MASTER EN SCIENCES INFORMATIQUES

Logiciel d'aide à l'utilisateur pour la recherche d'un itinéraire des transports en commun

Gathy, Jean-Marc

Award date:
1984

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES N.D. DE LA PAIX

NAMUR

INSTITUT D'INFORMATIQUE

LOGICIEL D'AIDE A L'UTILISATEUR POUR LA RECHERCHE D'UN ITINERAIRE DES TRANSPORTS EN COMMUN

Mémoire présenté pour l'obtention du grade de Licencié et Maitre en informatique par

G A T H Y
Jean - Marc

Promoteur:

J. BERLEUR

Année académique 1983-1984

Je tiens à exprimer toute ma gratitude au Père Berleur pour l'aide et les conseils qu'il m'a prodigués tout au long de l'élaboration de ce mémoire.

Je remercie également mes parents pour leur aide lors de la rédaction de ce mémoire et pour leurs encouragements au cours de mes études.

TABLE DES MATIERES

Chapitre 5 - Module de gestion des données

Annexes

INTRODUCTION

Qui n'a jamais vécu le cauchemar de devoir compulser un indicateur officiel de la SNCB ou de la SNCV, à la recherche de l'itinéraire le mieux adapté à ses impératifs horaires. La chose est relativement aisée lorsque les deux points à joindre sont des centres urbains; ce n'est plus vrai lorqu'une des deux extrémités correspond à un endroit perdu de notre pays, et à fortiori lorsque la région ou se situe cet endroit est étranger au voyageur. Or la majorité des voyageurs occasionnels se trouvent dans cette situation.

C'est pourquoi, il est intéressant de construire un outil informatique automatisant cette recherche d'itinéraire, et par la même simplifiant de façon conséquente la vie de ce martyr des temps modernes.

Le but de ce mémoire est de concevoir un tel outil.

Il serait prétentieux d'affirmer que l'outil présenté constitue ce qui peut se faire de mieux, et que le résultat obtenu par la recherche automatique, correspond à coup sûr à la solution optimale du problème posé par l'utilisateur. Tout au plus s'agit-il d'une base, sérieuse certe, mais nécessitant d'être approfondie avant de pouvoir être considérée comme un résultat final.

Les quatre premiers chapitres sont consacrés à l'outil de recherche, tandis que le cinquième se situe quelque peu à part puisqu'il concerne un outil auxiliaire, utile à l'introduction des données constituant l'indicateur officiel. Il ne concerne donc pas directement l'utilisateur des services de transport en commun, mais plutôt le personnel de ces services.

Le premier chapitre présente les spécifications du problème, c'est-à-dire qu'on y trouve exprimés de manière claire et nette, les données du problème et le résultat escompté.

Le second chapitre est constitué par la présentation de quelques méthodes applicables pour résoudre le problème, et le choix d'une de ces méthode qui sera approfondie dans la suite.

On s'intéresse plus particulièrement à la méthode choisie dans le troisième chapitre. Ainsi on y trouve une analyse un peu plus poussée et les algorithmes correspondants.

Le quatrième chapitre traite de l'environnement général nécessaire pour pouvoir appliquer la méthode choisie, ainsi que les algorithmes en résultant.

Le cinquième chapitre constitue une brève présentation d'un outil permettant l'introduction et la gestion de l'ensemble des informations constituant l'indicateur officiel.

On trouvera en annexe quelques explications sur certaines techniques de stockage des matrices creuses ainsi que les données particulières relatives à une région donnée.

CHAPITRE 1

SPECIFICATION DU PROBLEME

1.0. Introduction

Il faut donc construire un logiciel permettant la résolution automatique du problème de recherche d'un itinéraire joignant deux endroits précis, desservis chacun par au moins une liaison appartenant à un réseau de transports en commun, cet itinéraire n'empruntant que des liaisons de ce même réseau.

Avant de pouvoir rechercher la meilleur méthode pour résoudre le problème, il est nécessaire de spécifier d'une manière claire et rigoureuse ce problème.

Les spécifications d'un problème sont constituées:

- d'une définition précise des différentes données du problème.
- d'une définition précise des entrées, c'est-à-dire des données particulières à un problème posé par un utilisateur.
- d'une définition précises des sorties, c'est-à-dire des résultats obtenus si le problème a été résolu.

On ne trouvera donc pas dans ce chapitre des indications quand à la manière de résoudre le problème, mais seulement des définitions du matériel dont on dispose et de l'objet créé à partir de ce matériel.

SPECIFICATION

- Un point d'arrêt est un lieu géographique desservi par une ou plusieurs liaisons du réseau SNCB/SNCV.
- Un point de départ est un point d'arrêt.
- Un point d'arrivée est un point d'arrêt.
- Un point de jonction est un point d'arrêt desservi par au moins deux liaisons distinctes du réseau SNCB/SNCV.
- Une heure de départ d'un point d'arrêt ou d'arrivée à un point d'arrêt est une heure ou ce point d'arrêt est servi par une liaison du réseau SNCB/SNCV.
- Un itinéraire est une suite de 0, un ou plusieurs points de jonction encadrés par un point de départ au début et un point d'arrivée à la fin. Chaque point de jonction est accompagné d'une heure d'arrivée et d'une heure de départ, ainsi que du numero de la liaison joignant ce point de jonction au point d'arrêt suivant sur l'itinéraire. Le point de départ est accompagné d'une heure de départ ainsi que du numero de la liaison le joignant au premier point de jonction de l'itinéraire. Le point d'arrivée est accompagné d'une heure d'arrivée.
- En entrée, l'utilisateur indique 3 renseignements:

  a) un point de départ

  b) un point d'arrivée

  c) au choix, un et un seul des deux renseignements suivants:

     1) heure minimale à laquelle il est disposé à partir du point de départ.

     2) heure maximale pour laquelle il doit être au point d'arrivée.

  Cette heure indiquée par l'utilisateur ne doit pas nécessairement correspondre à une heure de départ d'un point d'arrêt ou d'arrivée à un point d'arrêt.

- La sortie du programme doit être un itinéraire satisfaisant les conditions suivantes:

  a) le point de départ de l'itinéraire et le point de départ indiqué par l'utilisateur doivent correspondre

  b) le point d'arrivée de l'itinéraire et le point d'arrivée indiqué par l'utilisateur doivent correspondre.

  c) selon le choix de l'utilisateur

     1. si une heure minimale a été indiquée

        - l'heure de départ associée au point de départ de l'itinéraire doit être postérieure à l'heure minimale indiquée par l'utilisateur.

        - les deux quantités suivantes doivent être optimisées simultanément

          . délai entre heure minimale indiquée et heure de départ associée au point de départ de l'itinéraire.

          . délai entre heure de départ associée au point de départ de l'itinéraire et heure d'arrivée associée au point d'arrivée de l'itinéraire.

     2. si une heure maximale a été indiquée

        - l'heure d'arrivée associée au point d'arrivée de l'itinéraire doit être antérieure à l'heure maximale indiquée par l'utilisateur.

        - les deux quantités suivantes doivent être optimisées simultanément

          . délai entre heure d'arrivée associée au point d'arrivée de l'itinéraire et heure maximale indiquée.

          . délai entre heure de départ associée au point de départ de l'itinéraire et heure d'arrivée associée au point d'arrivée de l'itinéraire.

CHAPITRE 2

ANALYSE DU PROBLEME ET RECHERCHE D'UNE METHODE DE RESOLUTION

2.0. Introduction

Une fois que le problème à résoudre a été spécifié (voir chapitre précédent), il faut déterminer la méthode à employer pour résoudre ce problème. Cette détermination est faite au travers d'une analyse un peu plus poussée du problème.

Le rêve de tout informaticien, confronté à la recherche d'une méthode de résolution d'un problème, c'est de pouvoir trouver une méthode qui à la fois soit le plus rapide, soit d'un coût minimum et calcule de manière certaine la meilleur solution. Hélas, ceci est très rarement possible et bien souvent il doit se contenter d'une méthode ne possédant qu'une ou deux de ces qualités. Les méthodes obtenues pour résoudre le problème vérifient malheureusement à des degrés différents cette généralité.

On distingue 3 parties dans ce chapitre:

1) Un bref rappel de l'énoncé général du problème, des données et du type de résultat escompté, ainsi que quelques explications sur les notations utilisées par la suite.
2) Une présentation des différentes méthodes applicables pour résoudre le problème. Cette présentation est faite de manière progressive, c.à.d qu'une méthode découle de la précédente en ce sens qu'elle tente d'en améliorer un des aspects (rapidité, coût ou qualité) tout en conservant le plus possible les qualités acquises.
3) Le choix d'une de ces méthodes présentées, sur base des qualités offertes au niveau de chacun des trois aspects.

2.1. Enoncé du problème

Rappel

On dispose au départ d'un indicateur des chemins de fer et des chemins vicinaux; les indicateurs reprennent d'une part les différentes liaisons formant les réseaux SNCB/SNCV, et d'autre part les grilles horaires associées à ces liaisons. Chaque liaison est constituée d'un nombre fini de points d'arrêt, avec répétition possible d'un point d'arrêt sur une même liaison.
Le problème est de déterminer au travers de ces données, un itinéraire joignant deux points d'arrêt desservi par au moins une de ces liaisons, cet itinéraire étant tel que le temps de parcours soit le plus court possible et tel que les heures de passage associées aux différents points d'arrêt formant l'itinéraire, correspondent le plus possible au souhait de l'utilisateur.

Notations et définitions

- Une liaison est notée l_i, i étant le numero d'identification de la liaison.
- Un point d'arrêt est noté a_i, i étant le numero d'identification du point d'arrêt.

Un graphe est entièrement déterminé par l'ensemble de ses sommets et par l'ensemble des arcs reliant les sommets;

c.à.d un graphe G = (X,U) ou $X = \{s_i\}$ avec $s_i \equiv$ sommet

$U = \{u_{i,j}\}$ avec $u_{i,j} \equiv$ arc reliant les sommets s_i et s_j

- Un chemin reliant deux sommets s_p, s_q du graphe G est une suite de sommets $s_{i_o}, s_{i_1}, s_{i_n}$ avec : $s_{i_o} = s_p, s_{i_n} = s_q$

  et pour tout k = 0 jusque n-1, il existe un arc $u_{i_k, i_{k+1}}$
- Un chemin élémentaire est un chemin passant au plus une fois par chacun de ses sommets.

2.2 - Première méthode

Le type même des données du problème (liaisons et points d'arrêt) conduit directement à penser une méthode de résolution basée sur l'exploitation de la théorie des graphes. en effet, on peut considérer que l'ensemble des liaisons forme un graphe défini de la manière suivante:

- l'ensemble des sommets du graphe est formé par l'ensemble des points d'arrêt apparaissant au moins une fois dans une liaison.
- un arc relie deux sommets du graphe si il existe une liaison telle que les deux points d'arrêt représentés par les sommets soient consécutifs sur cette liaison (il y a autant d'arcs que de liaisons vérifiant cette condition).
- le poids associé à un arc est le temps nécessaire pour joindre les deux points d'arrêt représenté par les sommets extrémités de l'arc.
- le poids associé à un sommet est le temps d'attente entre une arrivée et un départ du point d'arrêt représenté par le sommet.

Deux phénomènes contribuent à accroitre de manière sensible la complexité de construction d'un tel graphe:

1. le poids associé à un arc n'est pas constant et varie d'une application à l'autre.

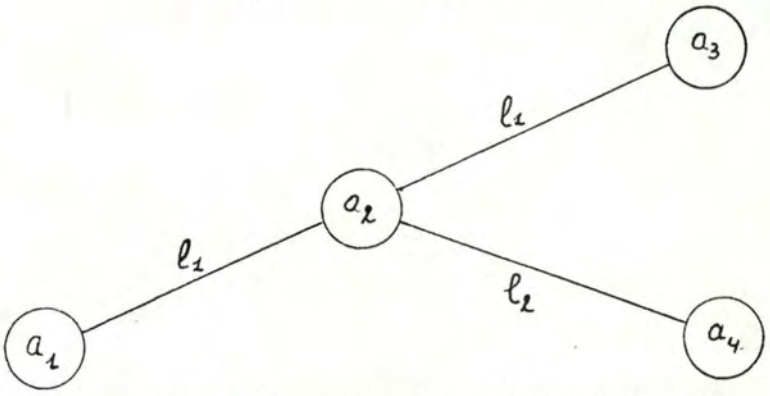
   En effet, le temps nécessaire pour joindre deux points d'arrêt consécutifs sur une liaison peut varier selon le moment choisi pour calculer ce temps; il peut aussi varier selon le sens de la liaison.
   Ex.: Liaison plus rapide les jours de vacances scolaires.

2. le poids associé à un sommet n'est pas constant et peut varier d'une application à l'autre.

   En effet, le temps d'attente en un point d'arrêt dépend du chemin suivi pour arriver à ce point d'arrêt et pour en repartir.

   Ex.: supposons le réseau restreint à deux liaisons

Le graphe construit est le suivant



Les différentes possibilités existantes si on se trouve au sommet a_2 sont les suivantes

Venir de			Aller vers	
a_1	$\xrightarrow{l_1}$	a_2	$\xrightarrow{l_1}$	a_3
a_1	$\xrightarrow{l_1}$	a_2	$\xrightarrow{l_2}$	a_4
a_3	$\xrightarrow{l_1}$	a_2	$\xrightarrow{l_1}$	a_1
a_3	$\xrightarrow{l_1}$	a_2	$\xrightarrow{l_2}$	a_4
a_4	$\xrightarrow{l_2}$	a_2	$\xrightarrow{l_1}$	a_1
a_4	$\xrightarrow{l_2}$	a_2	$\xrightarrow{l_1}$	a_3

Soit donc 6 possibilités pour le temps d'attente en a_2 et donc pour le poids associé au sommet correspondant.

Il est donc impossible de construire une fois pour toutes le graphe.

La méthode sera donc une méthode de sélection par extension successive, avec à chaque extension d'un chemin, le calcul du poids des arcs et du sommet correspondants. Cette méthode présente donc l'inconvénient de necessiter beaucoup de références aux grilles horaires (à chaque extension), mais d'un autre côté, la solution calculée est la solution optimale.

2.3 - Deuxième méthode

Pour éviter ce problème de la construction progressive du graphe, il est nécessaire de faire certaines simplifications sur les poids. Ces simplifications sont les suivantes:

1. le poids associé à un arc devient constant en considérant un temps moyen nécessaire pour joindre deux points d'arrêt.
2. le poids associé à un sommet devient constant en considérant le temps d'attente en un point d'arrêt comme étant nul.

On construit donc de manière définitive le graphe défini comme suit:

- l'ensemble des sommets du graphe est formé par l'ensemble des points d'arrêt apparaissant au moins une fois dans une liaison.
- un arc relie deux sommets si il existe une liaison telle que les deux points d'arrêt représentés par les sommets soient consécutifs sur cette liaison (il y a autant d'arcs que de liaisons).
- le poids associé à un arc est le temps moyen nécessaire pour joindre les deux points d'arrêt représentés par les sommets extrémités de l'arc, en suivant la liaison représentée par l'arc.
- le poids associé à un sommet est considéré comme nul.

En considérant ce graphe construit une fois pour toutes, la méthode à suivre pour résoudre le problème est la suivante:

Phase 1 - Appliquer sur le graphe un algorithme de recherche d'un chemin élémentaire de poids minimal joignant deux sommets. Le résultat sera une suite de points d'arrêt par lesquels il faut passer pour joindre notre point de départ à notre point d'arrivée, en empruntant les liaisons indiquées par les arcs parcourus. Ce parcours est celui qui en considérant les temps d'attente comme nuls en tout point d'arrêt, nécessitera en moyenne un temps minimum.

Phase 2 - Ayant trouvé un parcours, rechercher les heures de départ et d'arrivée, en respectant le choix de l'utilisateur, c'est-à-dire en partant du point de départ s'il a indiqué une heure minimale, en partant du point d'arrivée, s'il a indiqué une heure maximale.

Cette méthode ne nécessite plus la construction progressive du graphe, donc il y a moins de référence aux grilles horaires (uniquement lors de la phase 2 pour le chemin minimal), mais elle présente quand même un inconvénient majeur: la solution calculée ne représente qu'une approximation de la solution optimale et la précision de cette approximation peut parfois devenir très faible dans le cas de nombreux changements de liaison.

2.4 - Troisième méthode

Un moyen pour améliorer la précision de la solution calculée, c'est de donner une valeur non nulle aux poids associés aux sommets, c'est-à-dire aux temps d'attente, mais en veillant toujours à les garder constants pour éviter de retrouver le problème de la construction progressive du graphe. Une solution serait de calculer une valeur moyenne du temps d'attente en un point d'arrêt; mais l'écart entre la valeur nulle et la valeur moyenne peut encore être trop important et donc engendrer une imprécision sur la solution.

Exemple : 2 liaisons

ℓ_1 : ... a_1 — a_2 — a_3 ...

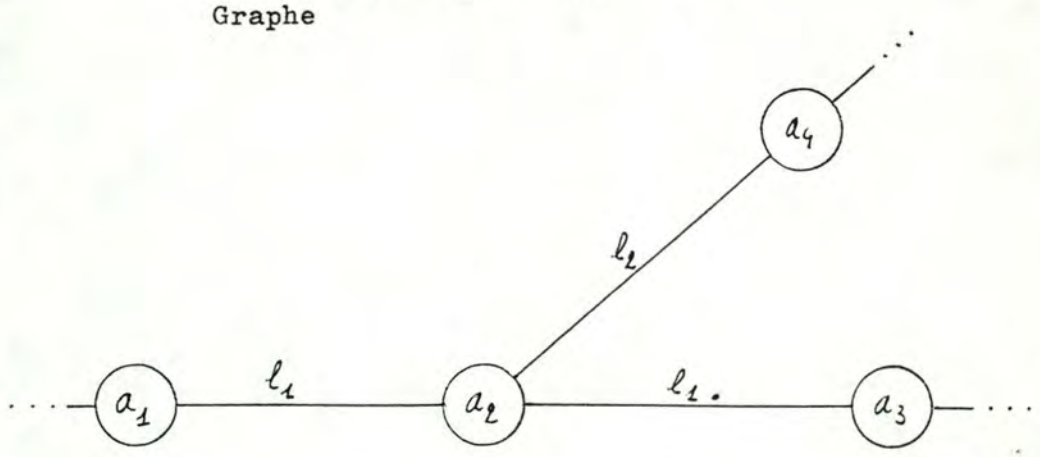
ℓ_2 : a_2 — a_4 ...

Temps de passage

ℓ_1	...	a_1	a_2	a_3	...
	...	9h50	10h20	10h40	... →
	...	11h00	10h30	10h10	... ←

ℓ_2	a_2	a_4	...
	11h00	11h30	... →
	10h00	9h30	... ←

Graphe



Calcul du poids du sommet a_2

Temps d'attente en a_2

venant de a_1 vers a_3 : 0 (même liaison)

a_1 a_4 : 40 (changement de liaison)

a_3 a_1 : 0 (même liaison)

a_3 a_4 : 30 (changement de liaison)

a_4 a_1 : 30 (idem)

a_4 a_3 : 20 (idem)

Temps moyen 20

Une autre solution serait de distinguer les points d'arrêt par la liaison à laquelle ils appartiennent, c'est-à-dire créer pour un point d'arrêt, autant de sommets que de liaisons passant par ce point et de créer de nouvelles liaisons fictives entre ces sommets.

On construit donc un graphe défini de la manière suivante:

- l'ensemble des sommets du graphe est formé par l'ensemble des paires (l_i, a_j) telles que le point d'arrêt a_j appartienne à la liaison l_i;
- un arc orienté relie un sommet à un autre sommet dans deux cas:
  1) les paires représentées par les deux sommets ont leur partie liaison commune et le point d'arrêt du sommet extrémité suit le point d'arrêt du sommet origine sur la liaison commune aux deux sommets.
  2) les paires représentées par les deux sommets ont leur partie point d'arrêt commune et leur partie liaison distinctes.

  En d'autres termes, un arc relie le sommet(l_i,a_j) au sommet (l_p,a_q) dans deux cas:

  1) $l_i=l_p$ et a_q suit a_j sur la liaison l_i
  2) $a_j=a_q$ et $l_i \neq l_p$

- le poids associé à un arc a une valeur dépendant du cas dans lequel on se trouve.

  Si on se trouve dans le premier cas, la valeur du poids est le temps mis pour joindre le point d'arrêt origine au point d'arrêt extrémité.

  Si on se trouve dans le second cas, la valeur du poids est le temps moyen d'attente au point d'arrêt commun entre une arrivée par la liaison origine et un départ sur la liaison extrémité.

- le poids associé à un sommet est nul.

Exemple:

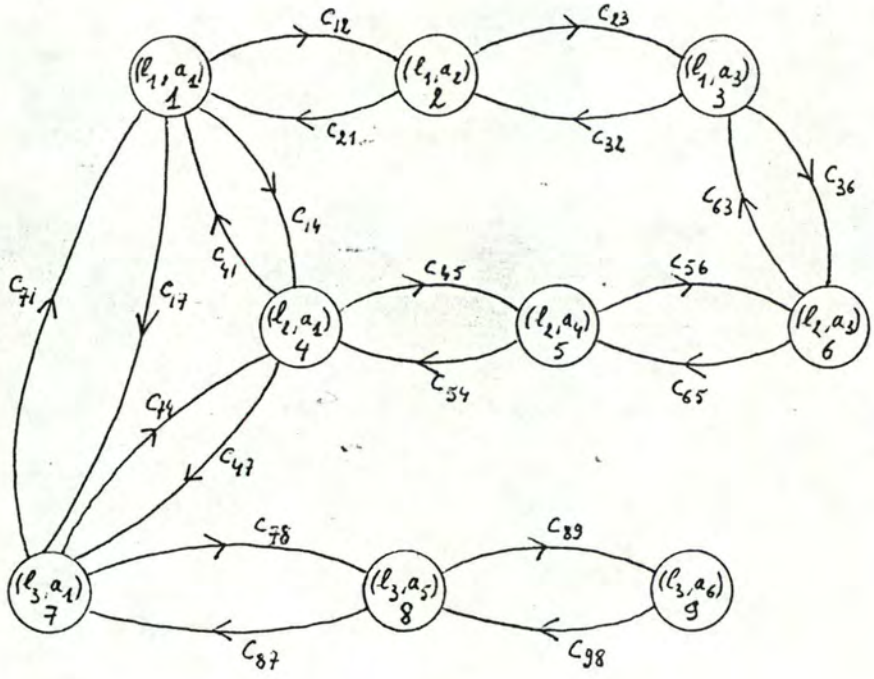
Supposons le réseau restreint à trois liaisons.

ℓ_1 : $a_1 — a_2 — a_3$

ℓ_2 : $a_1 — a_4 — a_3$

ℓ_3 : $a_1 — a_5 — a_6$

Le graphe correspondant est le suivant:



La méthode à suivre pour résoudre est la suivante, le graphe étant construit une fois pour toutes.

Phase 1 - Appliquer sur ce graphe un algorithme de recherche d'un chemin de poids minimal parmi les chemins élémentaires joignant deux sous ensembles de sommets. Le résultat sera une suite de paires (liaison, point d'arrêt) représentant un itinéraire avec les changements de liaison éventuels (deux paires $(\ell_p, a_i), (\ell_q, a_i)$ consécutives dans la suite). Cet itinéraire est celui qui, étant parcouru nécessite en moyenne le temps minimum.

Phase 2 - Ayant trouvé cet itinéraire, rechercher les heures de départ et d'arrivée en chaque point d'arrêt, en respectant le choix de l'utilisateur, c'est-à-dire en partant du point de départ s'il a indiqué une heure minimale; en partant du point d'arrivée s'il a indiqué une heure maximale.

Cette méthode ne nécessite plus de construction progressive du graphe, et la solution calculée devrait être une assez bonne approximation de la solution optimale; de plus elle est assez rapide, puisque il n'y a référence aux grilles horaires que dans la phase 2 et pour un seul chemin. Mais elle présente encore un gros inconvénient: la place nécessaire pour stocker le graphe. En effet si on reprend l'exemple décrit plus haut, la matrice associée au graphe est la suivante:

	1	2	3	4	5	6	7	8	9
1	0	c_{12}	0	c_{14}	0	0	c_{17}	0	0
2	c_{21}	0	c_{23}	0	0	0	0	0	0
3	0	c_{32}	0	0	0	c_{36}	0	0	0
4	c_{41}	0	0	0	c_{45}	0	c_{47}	0	0
5	0	0	0	c_{54}	0	c_{56}	0	0	0
6	0	0	c_{63}	0	c_{65}	0	0	0	0
7	c_{71}	0	0	c_{74}	0	0	0	c_{78}	0
8	0	0	0	0	0	0	c_{87}	0	c_{89}
9	0	0	0	0	0	0	0	c_{98}	0

soit donc une matrice de dimension 9 x 9 et de densité 0,25 , ce qui est assez important pour un petit exemple.

Si on applique cette méthode sur l'exemple décrit en annexe on obtient une matrice de dimension 99 x 99 et de densité 0,03 pour un problème à 23 liaisons et 43 points d'arrêt.

Pour éviter ce problème de place, ou du moins pour le diminuer, on peut employer des méthodes spéciales pour stocker des matrices creuses (voir annexe), mais toutes ces méthodes, si elles réduisent la place nécessaire pour stocker une matrice, augmentent aussi le nombre d'opérations nécessaires pour rechercher la valeur d'un élément et donc freinent l'exécution de la méthode.

2.5 - Quatrième méthode

Les trois méthodes proposées plus haut sont toutes basées sur l'utilisation d'un graphe dont les arcs possèdent des poids représentant un temps de parcours. Ainsi le chemin calculé est un chemin possédant un temps de parcours minimal. Le critère de sélection d'un chemin est donc le temps.

Mais il est possible d'utiliser un autre critère pour sélectionner un chemin, un critère moins fort que le temps, mais suffisant pour calculer une approximation de la solution optimale du problème. Ce critère, c'est le nombre de changements de liaison nécessaire pour joindre deux points d'arrêt.

Considérons le graphe construit de la manière suivante:

- l'ensemble des sommets du graphe est formé par l'ensemble des paires (ℓ_i, a_j) telles que le point d'arrêt a_j appartienne à la liaison ℓ_i .
- un arc relie deux sommets si une des conditions suivantes est satisfaite:

  1. Les deux paires représentées par les sommets ont la première partie (liaison) commune et les deux points d'arrêt formant la deuxième partie de chaque paire sont consécutifs sur la liaison commune.

  2. Les deux paires représentées par les sommets ont la seconde partie (point d'arrêt) commune,

  c'est-à-dire deux sommets quelconques (ℓ_i , a_j) et (ℓ_p , a_q) sont reliés par un arc si

  ou bien $\ell_i = \ell_p$ et a_j et a_q sont consécutif sur ℓ_i

  ou bien $a_j = a_q$

- le poids associé à un arc peut avoir deux valeurs
  - 1 si les deux sommets extrémités de l'arc ont leur deuxiéme partie (point d'arrêt) commune
  - 0 sinon
- le poids associé à un sommet est nul.

Parcourir un arc de poids nul correspond à passer au passer au point d'arrêt suivant sur la liaison, tandis que parcourir un arc de poids non nul correspond à changer de liaison en un point d'arrêt.

Exemple:

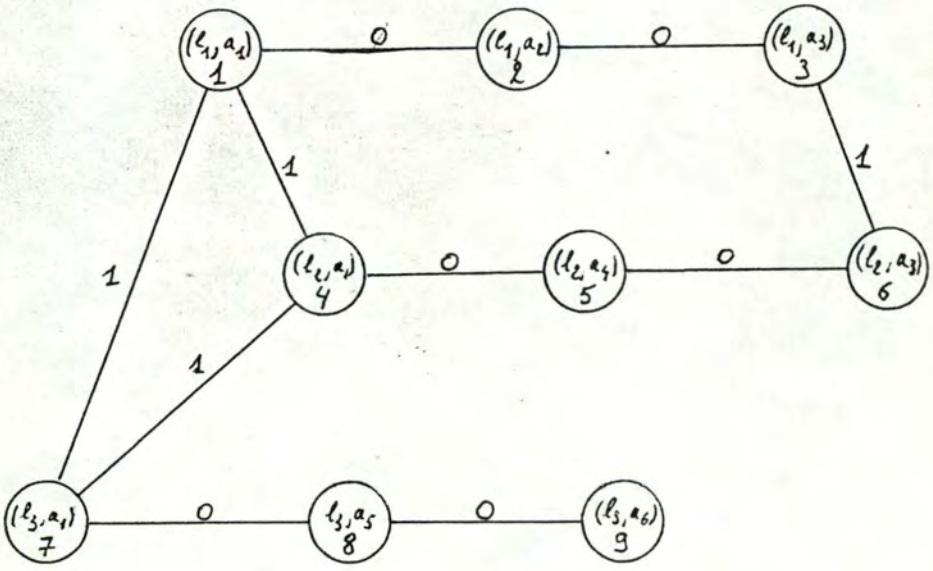
Reprenons l'exemple déjà décrit plus haut

ℓ_1 : $a_1 — a_2 — a_3$

ℓ_2 : $a_1 — a_4 — a_5$

ℓ_3 : $a_1 — a_5 — a_6$

Le graphe construit est le suivant:



La méthode à suivre pour résoudre le problème est la suivante

- Le graphe étant construit une fois pour toutes.

Phase 1 - Appliquer sur le graphe un algorithme de recherche d'un chemin de poids minimal parmi les chemins élémentaires joignant deux sous-ensembles de sommets. Le résultat sera une suite de paires (liaison, point d'arrêt) formant un itinéraire avec les éventuels changements de liaison (deux paires (ℓ_p, a_i), (ℓ_q, a_i) consécutives dans la suite). Cet itinéraire est celui qui, étant parcouru, nécessite le minimum de changement de liaison.

Phase 2 - A partir de l'itinéraire, rechercher les heures de départ et d'arrivée en chaque point d'arrêt, en respectant le choix de l'utilisateur, c'est-à-dire en partant du point de départ s'il a indiqué une heure minimale, en partant du point d'arrivée, s'il a indiqué une heure maximale.

Le graphe peut être stocké de deux manières différentes:

1 - Soit sous forme d'une matrice M carrée symétrique de dimension (ns x ns) où ns est le nombre de sommets du graphe, et dont les éléments M(i,j) peuvent avoir
la valeur 1 si un arc de poids 1 relie les sommets i et j
la valeur -1 si un arc de poids 0 relie les sommets i et j
la valeur 0 sinon

2 - Soit sous forme d'une matrice rectangulaire N de dimension n x m, ou n est le nombre de liaisons et m le nombre de points d'arrêt, et dont les éléments N (i,j) peuvent avoir la valeur k si le point d'arrêt aj appartient à la liaison lj, en k^{me} position,

0 sinon

Exemple:

Si on reprend l'exemple déjà décrit plus haut (3 liaisons), les deux matrices obtenues sont les suivantes:

1ère forme

	1	2	3	4	5	6	7	8	9
1	0	-1	0	1	0	0	1	0	0
2	-1	0	-1	0	0	0	0	0	0
3	0	-1	0	0	0	1	0	0	0
4	1	0	0	0	-1	0	1	0	0
5	0	0	0	-1	0	-1	0	0	0
6	0	0	1	0	-1	0	0	0	0
7	1	0	0	1	0	0	0	-1	0
8	0	0	0	0	0	0	-1	0	-1
9	0	0	0	0	0	0	0	-1	0

2ème forme

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} 1 & 2 & 3 & 0 & 0 & 0 \\ 1 & 0 & 3 & 2 & 0 & 0 \\ 1 & 0 & 0 & 0 & 2 & 3 \end{pmatrix} \end{array}$$

Remarque:

Si on utilise la deuxième forme pour stocker le graphe,

- un sommet du graphe est représenté par un élément non nul de la matrice N
- un arc de poids nul relie les deux sommets (lp, ai) et (lp,aj)
  ssi N(p, i) - N(p,j) est égal à 1 ou -1
- un arc de poids égal à 1 relie les deux sommets (lp,ai) et (lq,ai)
  ssi N(p,i) et N(q,i) sont différents de 0

Cette méthode, quelque soit le choix de matrice pour stocker le graphe, nécessite moins de place, mais elle présente encore un inconvénient. En effet, la solution fournie ne représente encore qu'une approximation plus ou moins précise de la solution optimale; néanmoins, il est possible de remédier en partie à cet inconvénient en modifiant en rien la méthode.

De plus, cette méthode est très rapide puisque le calcul du temps de parcours, et donc les accès aux grilles horaires, ne se fait que pour un seul chemin.

2.6 - Cinquième méthode

Dans la phase 1 de la méthode précédente, au lieu de calculer uniquement le chemin de poids minimal, on calcule un ensemble des meilleurs chemins, c'est-à-dire on considère tous les chemins dont le poids est inférieur ou égal au poids du chemin minimal calculé plus une certaine valeur.

Dans la phase 2, on recherche les heures de départ et d'arrivée en chaque point d'arrêt de chaque chemin en respectant le choix de l'utilisateur, c'est-à-dire en partant du point de départ s'il a indiqué une heure minimale, du point d'arrvée, s'il a indiqué une heure maximale. On ne garde alors que le chemin pour lequel le temps de trajet et le délai entre heure minimale et heure de départ ou heure maximale et heure d'arrivée sont optimaux.

Cette méthode ne nécessite pas plus de place que la précédente, mais offre une meilleurs précision sur la solution. De plus, elle ne demande pas énormément de travail supplémentaire lors de son application, puisqu'on ne calcule les temps que pour un nombre restreint de chemins.

2.7 - Choix d'une méthode

Trois critères peuvent intervenir dans le choix d'une méthode

1 - la rapidité d'exécution
2 - la place mémoire nécessaire
3 - la qualité de la solution obtenue

De l'analyse présentée ci-avant, on peut tirer le tableau théorique suivant:

Critères \ Méthodes	1	2	3	4	5
1	-	0/+	0	+	0/+
2	0/+	0/+	-	+	+
3	+	-	0/+	-	0/+

- : mauvais
0 : moyen
+ : bon

Si le but recherché est une bonne approximation de la solution optimale, alors la 5ème méthode représente le meilleur compromis, puisqu'elle calcule une assez bonne solution en nécessitant peu de place et en étant assez rapide. C'est donc celle-la qu'on va développer, tout en gardant éventuellement la possibilité de pouvoir comparer les résultats obtenus avec ceux obtenus par la 4ème méthode.

CHAPITRE 3

FORMALISATION DE LA METHODE CHOISIE

3.0. Introduction

On a donc choisi une méthode ayant l'avantage d'avoir un moindre coût au niveau de la place nécessaire, tout en calculant de manière assez rapide une bonne approximation de la meilleur solution.

Cette méthode se scinde en deux phases:

- Une phase ou on recherche par construction progressive, un ensemble de chemins reliant deux points d'arrêt, ces chemins ayant la particularité d'être ceux qui parmi tous les chemins possibles joignent les deux points d'arrêt, empruntent un nombre relativement peu important de liaisons On obtient non seulement les meilleurs chemins, mais aussi un certain nombre de chemins qui peuvent être considérés comme assez bons.
- Une phase ou on recherche pour chacun des chemins trouvés dans la première phase, les temps de passage aux différents points d'arrêt tels que d'une part, le temps de parcours soit minimal et tels que d'autre part, l'heure de départ ou l'heure d'arrivée satisfasse au mieux l'utilisateur.

Le chapitre 3 est constitué de 3 parties:

1) Un rappel des notations utilisées ainsi que certaines généralités sur le graphe auquel il est fait référence.
2) La formalisation de la 1ère phase qui comporte les algorithmes, ainsi que les structures des données nécessaires.
3) La formalisation de la 2ème phase, qui comporte les algorithmes ainsi que les structures des données nécessaires.

3.1 - Généralité et rappel

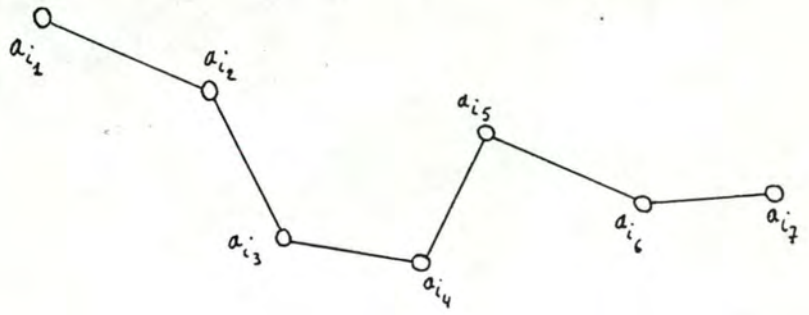
- On considère un réseau formé par

  n liaison, notée l_i , i = 1 jusque n

  m points d'arrêt distincts, notés a_j, j = 1 jusque m

- Une liaison est représentée par une liste de points d'arrêt. Cette liste est non vide et contient au moins deux éléments; en effet, il n'est pas pensable d'imaginer une liaison ne passant nulle part, ou ne comprenant qu'un seul point d'arrêt. De plus les points d'arrêt sont dans la liste, dans l'ordre où ils apparaissent sur la liaison.

  Ainsi, si on considère une liaison l_i constituée comme suit:



  La liste représentant la liaison sera

  soit $(a_{i_1}, a_{i_2}, a_{i_3}, a_{i_4}, a_{i_5}, a_{i_6}, a_{i_7})$

  soit $(a_{i_7}, a_{i_6}, a_{i_5}, a_{i_4}, a_{i_3}, a_{i_2}, a_{i_1})$

Une liaison peut-être parcourue dans deux sens

1er sens: dans l'ordre de la liste $\stackrel{déf}{=}$ sens direct.

2e sens: dans l'ordre inverse de la liste $\stackrel{déf}{=}$ sens indirect.

- Le graphe représentant le réseau est stocké sous forme d'une matrice rectangulaire N

Cette matrice est telle que

le point d'arrêt a_j apparait en k^{me} position dans la liste représentant la liaison l_i

ssi l'élément N (i,j) = k

Il est donc possible de repérer

1) les points d'arrêt constituant la liaison l_i en parcourant la i^{me} ligne de la matrice N

2) les liaisons passant par un point d'arrêt a_j en parcourant la j^{me} colonne de la matrice N

- En vertu de la définition des arcs joignant deux sommets du graphe, les seuls déplacements permis dans le graphe sont soit ceux entre deux sommets correspondant à un même point d'arrêt $\left((l_i,a_p) \rightarrow (l_j,a_p) \right)$, soit ceux entre deux sommets correspondant à une même liaison $\left((l_i,a_p) \rightarrow (l_i,a_q) \right)$. Donc, si on représente les déplacements dans la matrice N, seuls sont permis les déplacements horizontaux entre deux éléments non nuls, et dont la différence des deux valeurs vaut +/- 1 (parcours de la liaison entre deux points d'arrêt consécutifs), et les déplacements verticaux entre deux éléments de valeur non nulle (changement de liaison en un point d'arrêt).

- Un chemin reliant deux points d'arrêt sera donc constitué d'une suitealternée de déplacements verticaux et horizontaux, donc d'une suite alternée de parcours de liaison et de changements de liaison.

- Un chemin reliant deux points d'arrêt a_p et a_q sera représenté de la manière suivante:

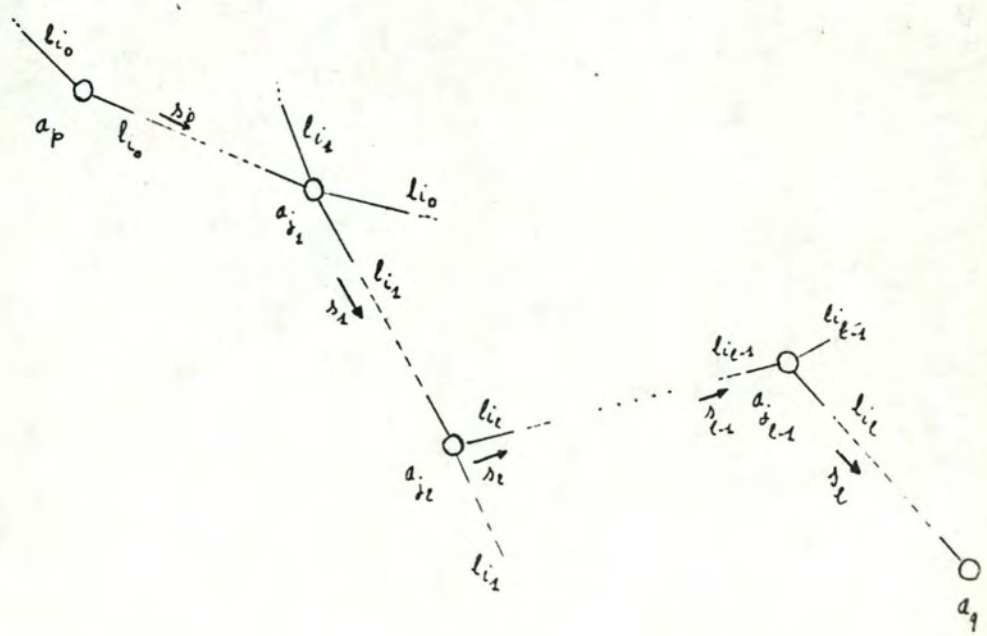
$$(p,i_o,s_o) = (j_o,i_o,s_o)\ ,\ (j_1,i_1,s_1),$$

$$(j_{e-1},i_e,{}^1{}_{e-1}),(j_e,i_e,s_e) = (q,0,0)$$

Chaque triple (j_k,i_k,s_k), k = 0 jusque l-1 signifiant:

au point d'arrêt a_{j_k}, prendre la liaison l_{i_k} dans le sens indiqué par s_k, jusqu'au prochain point d'arrêt nécessitant un changement de liaison, c'est-à-dire jusqu'au point d'arrêt $a_{j_{k+1}}$

Si on représente graphiquement le chemin:



3.2 - PHASE 1 : Recherche des "meilleurs" chemins reliant deux points d'arrêt a_p et a_q .

3.2.1 - Recherche des chemins joignant 2 points d'arrêt a_p et a_q.

- Pour trouver tous les chemins joignant deux points d'arrêt a_p et a_q, on va utiliser un procédé de construction par extension progressive, c'est-à-dire,

  on suppose qu'on a déjà trouvé un chemin élémentaire joignant a_p à un point d'arrêt a_k, distinct de a_q, on construit alors tous les prolongements possible à partir de a_k vers le point d'arrêt a_q .

- Chemin élémentaire joignant a_p à a_k $\overset{not}{=}$ (a_p,a_k) .

- Problème particulier: calculer tous les prolongement possible de (a_p,a_k) vers a_q.

  Pour calculer les prolongements possibles à partir de a_k, il faut parcourir l'ensemble des liaisons passant par a_k en essayant d'arriver soit en a_q, ce qui termine le chemin, soit en un point appartenant déjà au chemin (a_p,a_k), ce qui interdit tout prolongement dans cette direction, soit en un point d'arrêt tel qu'il soit possible à partir de ce point de calculer plusieurs prolongements distincts vers a_q, c.à.d. un point d'arrêt qui permette un changement de liaison. Une fois qu'un tel point d'arrêt est trouvé, on calcule tous les prolongements possibles de ce point vers a_q.

- Algorithme du problème particulier.

Parcourir l'ensemble des liaisons passant par a_k

Pour chaque liaison, prolonger si c'est possible vers a_q c.à.d. parcourir la liaison dans chacun des deux sens possible, jusqu'à se trouver dans une des situations suivantes:

1) On est en fin de liaison ou on est en un point d'arrêt appartenant déjà au chemin (a_p,a_k), c.à.d. prolongement impossible.
2) On est au point d'arrêt a_q, c.à.d. le chemin est terminé.
3) On est au point d'arrêt a_j permettant un changement de liaison, c.à.d. qu'il existe plusieurs prolongements possibles distincts à partir de ce point d'arrêt.

Selon la situation:

1) Passer à la liaison suivante.
2) Ajouter le point d'arrêt a_q au chemin $(a_p,a_k) \rightarrow$ chemin (a_p,a_q).
   éditer ou mémoriser le chemin (a_p,a_q)
   Retirer le point d'arrêt a_q du chemin $(a_p,a_q) \rightarrow$ chemin (a_p,a_k)
   Passer à la liaison suivante.
3) Ajouter le point d'arrêt a_j au chemin $(a_p,a_k) \rightarrow$ chemin (a_p,a_j).
   Calculer les prolongements possibles de a_j vers a_q.
   Retirer le point d'arrêt a_j du chemin $(a_p,a_j) \rightarrow$ chemin (a_p,a_k).
   Passer à la liaison suivante.

- Algorithme du sous-problème: ajouter a_j au chemin $(a_p, a_k) \rightarrow$ chemin (a_p, a_j).

. Si la longueur du chemin est nulle, c.à.d. si on calcule un prolongement à partir du point de départ, ou si la liaison utilisée pour prolonger n'est pas celle utilisée pour arriver au point d'arrêt a_k
(correspond à un changement de liaison)

- . indiquer dans le chemin la position du point d'arrêt de prolongement (a_k) sur la liaison utilisée pour prolonger (l_i).
- . indiquer dans le chemin le point d'arrêt vers lequel on prolonge (a_j)
- . indiquer dans le chemin la position de ce point d'arrêt sur la liaison de prolongement (l_i).
- . indiquer dans le chemin la liaison utilisée pour prolonger.
- . indiquer dans le chemin le sens emprunté.
- . incrémenter la longueur du chemin.
- . marquer le point d'arrêt a_j comme appartenant au chemin.

. Sinon (correspond à un parcours de liaison)

- . marquer le point d'arrêt a_j comme appartenant au chemin.
- . remplacer le point d'arrêt de prolongement (a_k) dans le chemin par celui vers lequel on prolonge (a_j)
- . remplacer dans le chemin la position du point de prolongement sur la liaison de prolongement par celle du point vers lequel on prolonge.

Explication: le point d'arrêt ou aura lieu l'éventuel prochain changement de liaison devient a_j, puisqu'on prolonge le parcours au dela de a_k jusque a_j).

- Algorithme du sous problème: Retirer a_j du chemin $(a_p, a_j) \longrightarrow$ chemin (a_p, a_k).

S'il n'y a pas eu de changement de liaison en a_k :

- marquer le point d'arrêt a_j comme n'appartenant plus au chemin.
- remplacer le point d'arrêt de prolongement dans le chemin (a_j) par a_k .
- remplacer la position du point d'arrêt de prolongement sur la liaison de prolongement par celle du point d'arrêt a_k.

Explication: on stoppe le parcours de la liaison en a_k

Sinon:

- marquer le point d'arrêt a_j comme n'appartenant plus au chemin.
- supprimer le point a_j du chemin.
- supprimer dans le chemin la liaison utilisée à partir de a_k pour prolonger vers a_j .
- supprimer le sens emprunté.
- diminuer la longueur du chemin de une unité.

- Problème général: calculer les prolongements possibles de a_p vers a_q .

3.2.2. - Expression des algorithmes en pseudo langage

- Données permanentes:

Pour permèttre un accès plus rapide aux informations nécessaires , la forme de stockage de la matrice N est modifiée.

Tout d'abord, on crée deux tableaux.

. NL [1..m] donnant pour chaque point d'arrêt, le nombre de liaisons passant par ce point d'arrêt,

c.à.d.

NL [i] = k ssi k liaisons passent par le point d'arrêt a_i

NL[i] correspond au nombre d'éléments non nuls de la $i^{ème}$ colonne de N.

. NA [1..n] donnant pour chaque liaison le nombre de points d'arrêt constituant cette liaison.

c.à.d.

NA[j] = K ssi la liaison l_j est constituée par k points d'arrêt.

NA[j] correspond au nombre d'éléments non nuls de la $j^{ème}$ ligne de N

Ensuite, à chaque point d'arrêt a_i , on associe deux tableaux.

. A_i [1..NL(i)] donnant la liste des liaisons passant par le point d'arrêt a_i .

. $POSL_i$ [1..NL (i)] donnant la position du point d'arrêt a_i dans les liaisons passant par a_i (position par rapport au sens direct).

c'est-à-dire

A_i [j] = K et $POSL_i$ [j] = p
ssi le point d'arrêt a_i apparait en $p^{ème}$ position sur la liaison $l_k \rightarrow N(k,i) = p$

Le tableau A_i reprend les indices des éléments non nuls de la $i^{ème}$ colonne de N, tandis que le tableau $POSL_i$ reprend la valeur de ces éléments.

On associe également à chaque liaison l_i un tableau

L_i [1..NA (i)] donnant la liste des points d'arrêt constituant la liaison l_i

c'est-à-dire

L_i [j] = K ssi le $j^{ème}$ point d'arrêt de la liaison l_i est $a_k \rightarrow N(i,k) = j$.

Le tableau L_i reprend les indices des éléments non nuls de la $i^{ème}$ ligne de N, classés par ordre de valeur croissante.

$\forall p \quad 1 \leq p \leq NA(i), \forall q \quad 1 \leq q \leq NA(i)$

$p \leq q = N(i,p) \leq N(i,q)$.

- Tableaux et variables utilisés.

l : représente la longueur du chemin en cours d'extension, exprimée en nombre de liaisons empruntées.

M [1..m] indique directement si un point d'arrêt appartient déjà au chemin en cours d'extension.
c.à.d. M[i] = 1
ssi le point d'arrêt a_i appartient au chemin

PA_o [o..n] donne les points d'arrêt formant le chemin, points d'arrêt ou intervient un changement de liaison.

Ch_o[o..n] donne les liaisons empruntées pour parcourir le chemin.

S_o[o..n] donne le sens des liaisons empruntées.

$POSA_o$ [o..n] donne la position précise du point d'arrêt dans la liaison utilisée pour parvenir en ce point.

$POSD_o$[o..n] donne la position précise du point d'arrêt dans la liaison utilisée pour quitter ce point.

c.à.d. $PA_o[j] = k$, $Ch_o[j] = i$, $S_o[k] = p$

$POSA_o[j] = p$, et $POSD_o[j] = q$.

ssi le $j^{ème}$ changement de liaison se produit au point d'arrêt a_k et il faut prendre la liaison l_i dans le sens direct n s = 1 dans le sens indirect n 1 = 2. De plus le point d'arrêt est en $q^{ème}$ position sur la sur la liaison l_i, et était en $p^{ème}$ position sur la liaison précédant l_i dans le chemin ($Ch_o[j-1]$).

De plus pour mémoriser les chemins

nc: nombre de chemins calculés.

l_i: longueur du $i^{ème}$ chemin calculé

PA_i $[0..l_i]$

Ch_i $[0..l_i]$

S_i $[0..l_i]$

$POSA_i$ $[0..l_i]$

$POSD_i$ $[0..l_i]$

} Même définition que PA., Ch., S., POSA., POSD

Remarques:

- Si l est la longueur du chemin en cours d'extension, alors le point d'arrêt de prolongement, c.â.d. à partir duquel on calcule un prolongement est $a_{PA_o}(l)$.

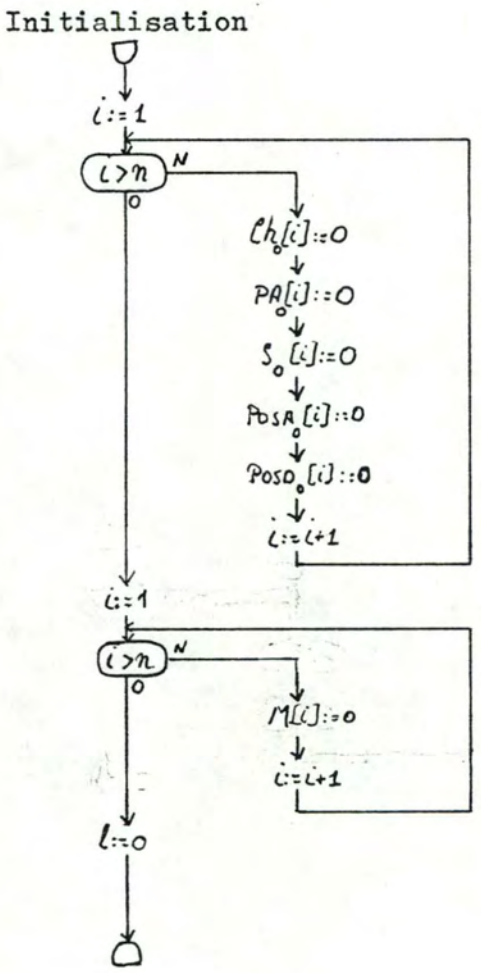
  La liaison utilisée pour arriver au point de prolongement est $l_{Ch_o}(l-1)$.

  La liaison utilisée pour prolonger sera $l_{Ch_o}(l)$.

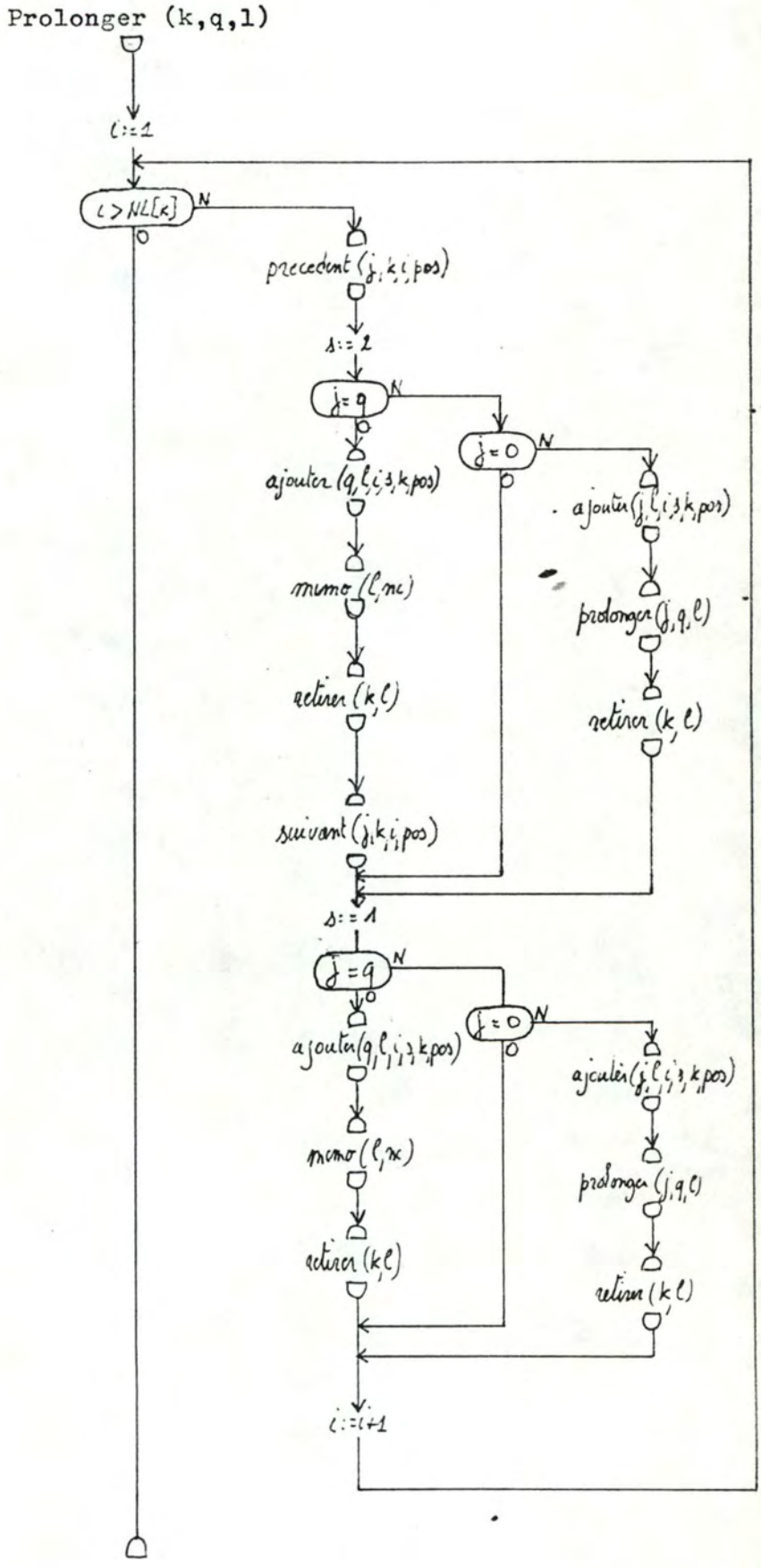
- Les deux tableaux $POSA_o$ et $POSD_o$ sont créés pour parer au cas d'un point d'arrêt apparaissant deux fois ou plus dans une liaison.

- Initialisation

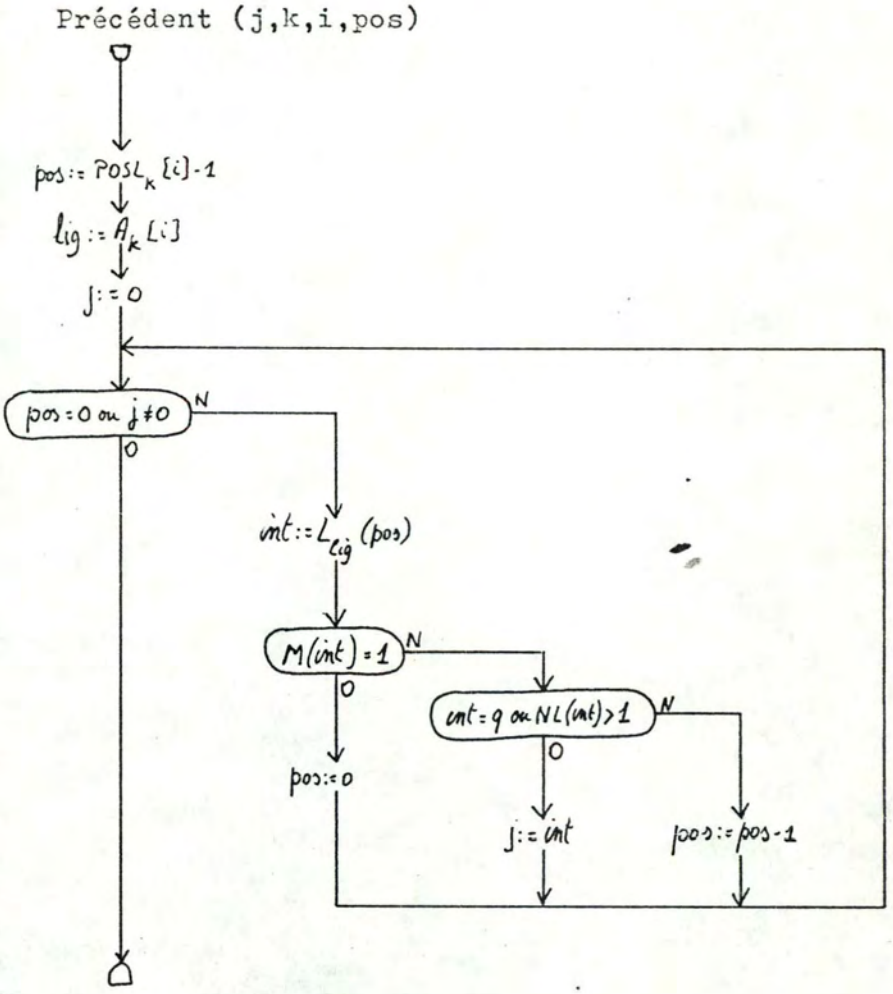
Au départ, le chemin doit être vide, donc les cinq tableaux M, Ch_o, PA_o, $POSA_o$ et $POSD_o$ doivent avoir tous leurs éléments nuls; de plus la longueur doit aussi être nulle.



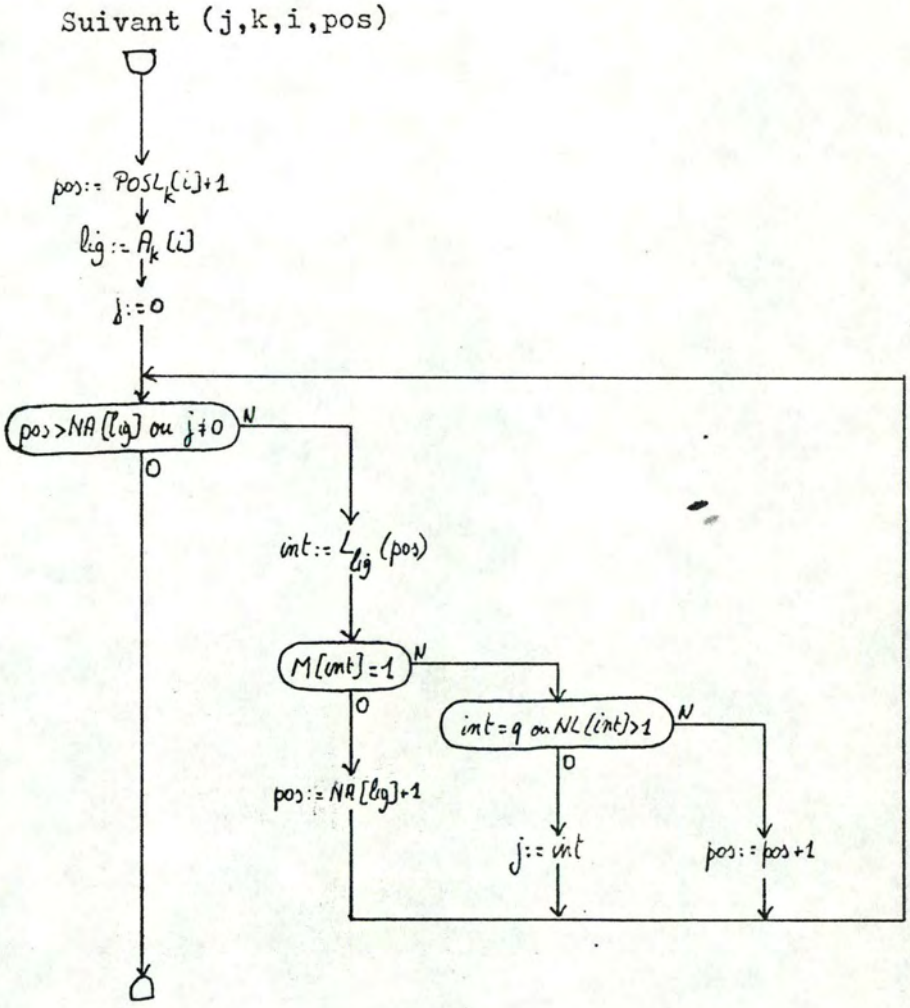
- Algorithme du problème particulier



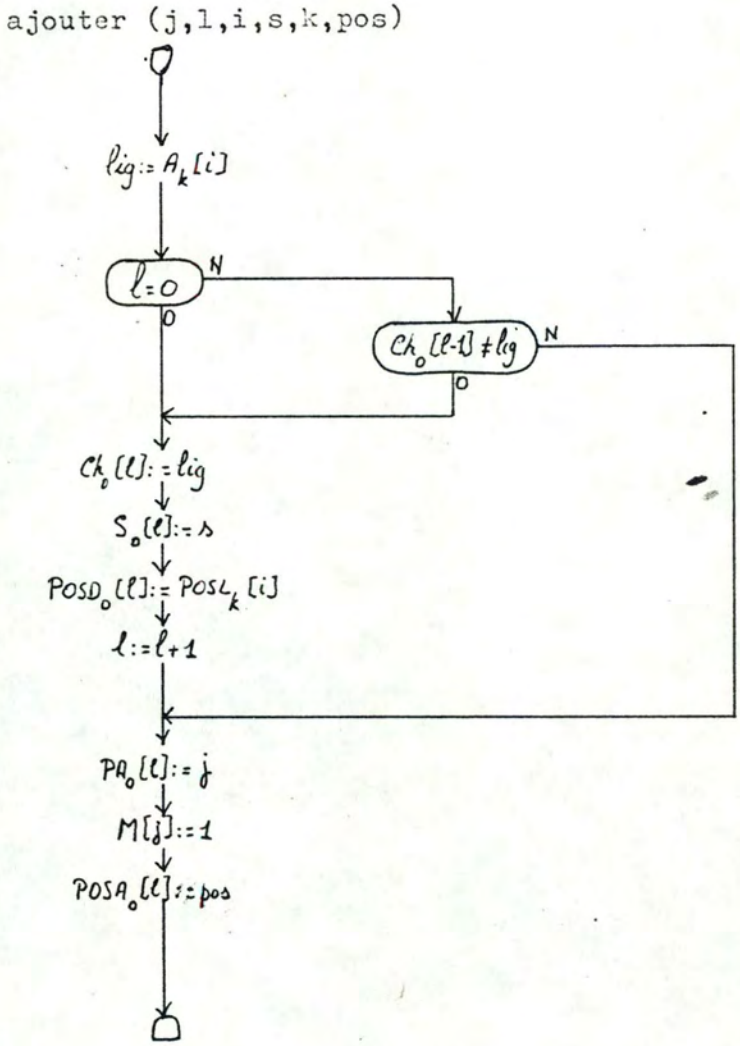
- Algorithme du sous problème: précédent



- Algorithme du sous problèle: suivant

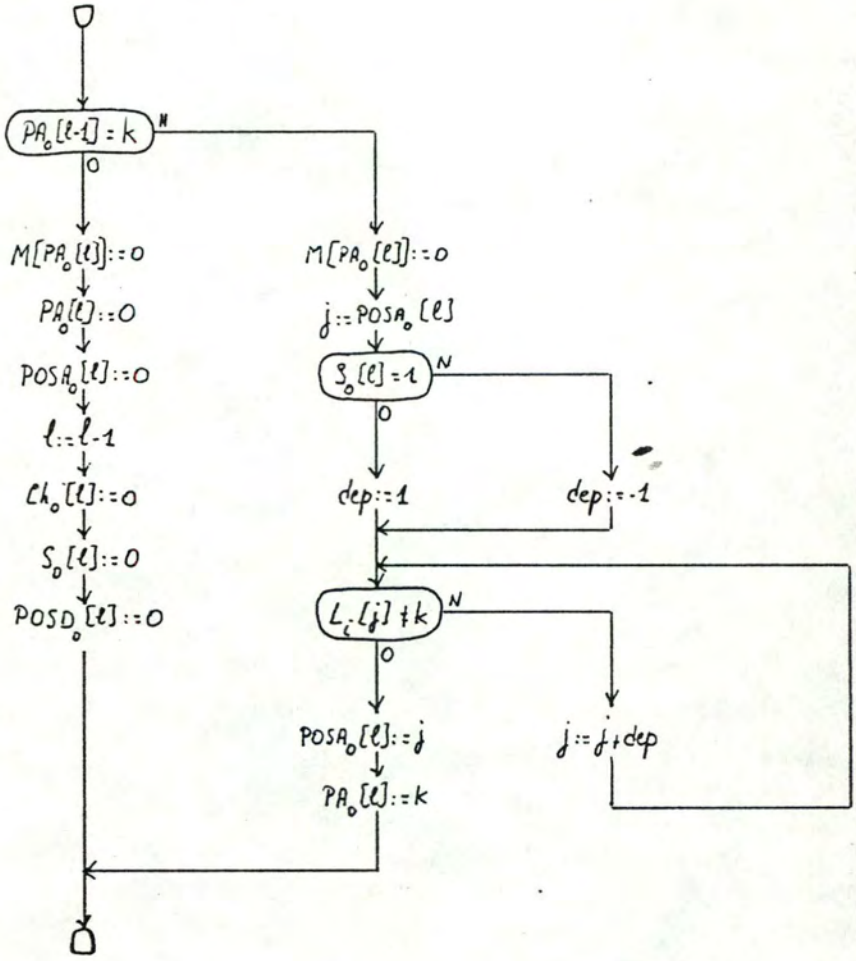


- Algorithme du sous problème: ajouter



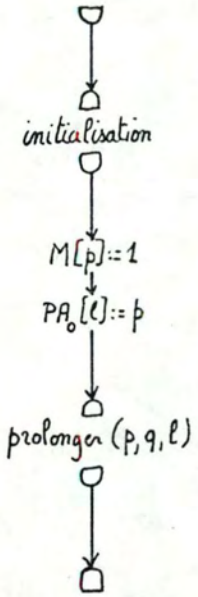
- Algorithme du sous problème: retirer

retirer (k,l)



- Algorithme du problème original

Programme principal



3.2.3. - Recherche du ou des chemins de poids minimal joignant deux points d'arrêt a_p et a_q .

- L'algorithme utilisé pour la recherche de tous les chemin élémentaires joignant deux points d'arrêt peut être utili moyennant certaines petites modifications, pour recherche le ou les chemins de poids minimal.

- Une nouvelle variable est introduite: lmax. Cette variabl représente la longueur du plus petit chemin élémentaire complet trouvé. Au départ, lmax aura comme valeur une valeur théorique maximale n c.à.d. le nombre de liaisons.

- Les modifications apportées interviennent en deux endroit de l'algorithme:

  1) Lorqu'on passe en revue les liaisons passant par a_k , si pour pouvoir prolonger à partir de a_k sur la liaison l_i, il faut un changement de liaison en a_k, et qu'à ce moment la longueur du chemin est déjà égale à lmax, alors il est inutile de chercher des prolongements puisque en raison du changement de liaison, la longueur du chemin sera automatiquement augmentée et donc dépassera la longueur d'un chemin élémentaire dé trouvé. Donc le prolongement ne pourrait en aucun cas correspondre à un chemin de poids minimal.

  2) Lorqu'on mémorise un nouveau chemin complet, si la longueur de ce chemin est inférieure à lmax, il faut alors d'une part, changer la valeur de lmax et la remplacer par la longueur du nouveau chemin, et d'autre part, supprimer tous les chemins déjà existants, puisqu'ils sont de longueur supérieure à lmax.

3.2.4. - Recherche des meilleurs chemins joignant deux points d'arrêt a_p et a_q .

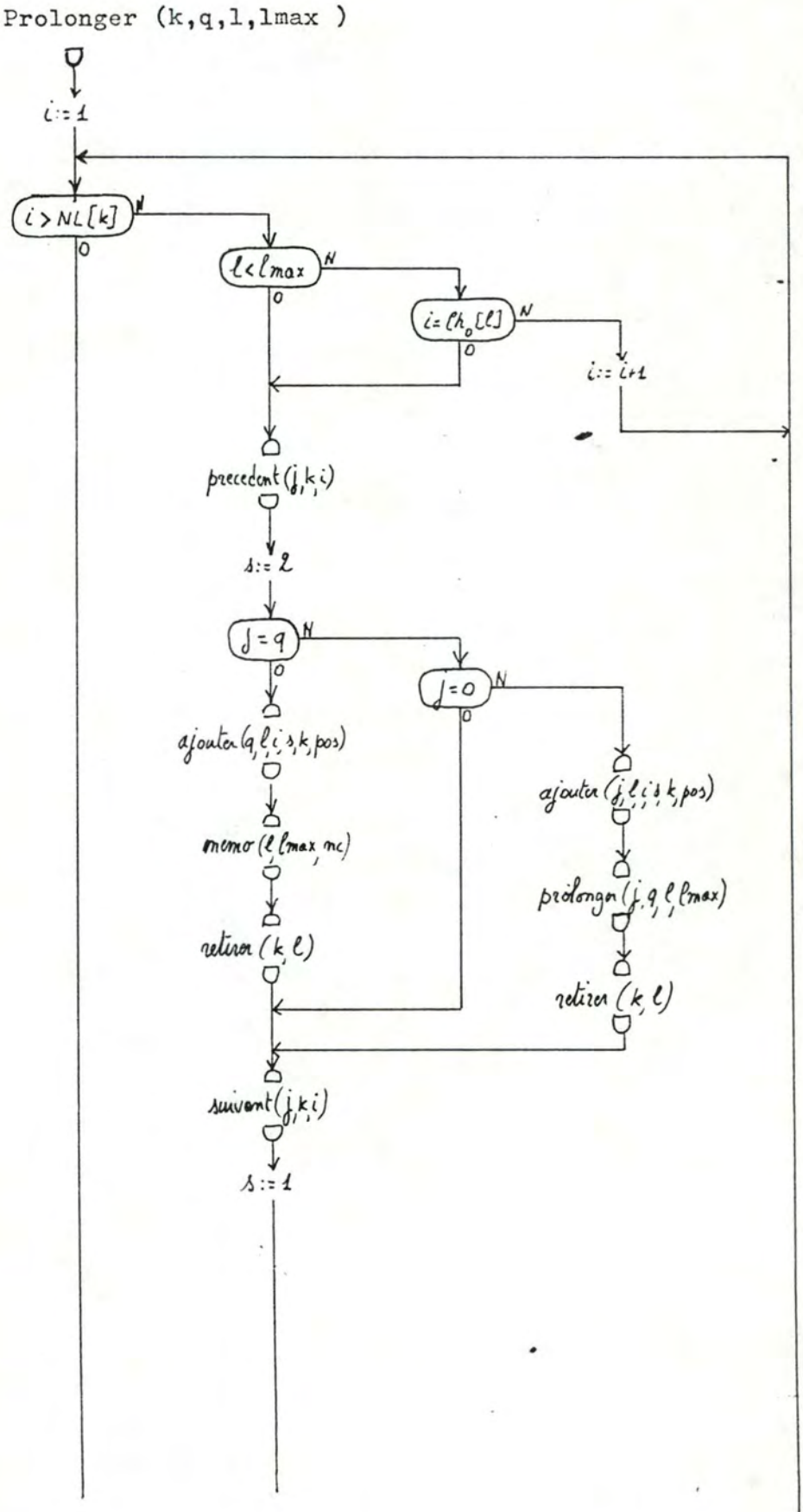
- De nouveau l'algorithme utilisé pour rechercher le ou les chemins de poids maximal, peut être utilisé pour rechercher les meilleurs chemins, à un détail près.

- Auparavant, la variable lmax correspondait à la valeur de la longueur du plus petit chemin élémentaire calculé. Maintenant lmax va représenter la borne supérieure pour la longueur d'un chemin; c'est-à-dire, on ne va plus considérer seulement les chemins qui ont une longueur minimale mais tous les chemins qui ont une longueur inférieure à la borne supérieure lmax. Ainsi à tout moment, lmax aura comme valeur la longueur du plus petit chemin complet calculé plus une certaine quantité qui représente l'écart maximum qui peut exister entre la longueur du chemin minimal et la longueur d'un chemin admissible, c.à.d. un chemin qui peut être considéré comme un des meilleurs chemins reliant les deux points d'arrêt.

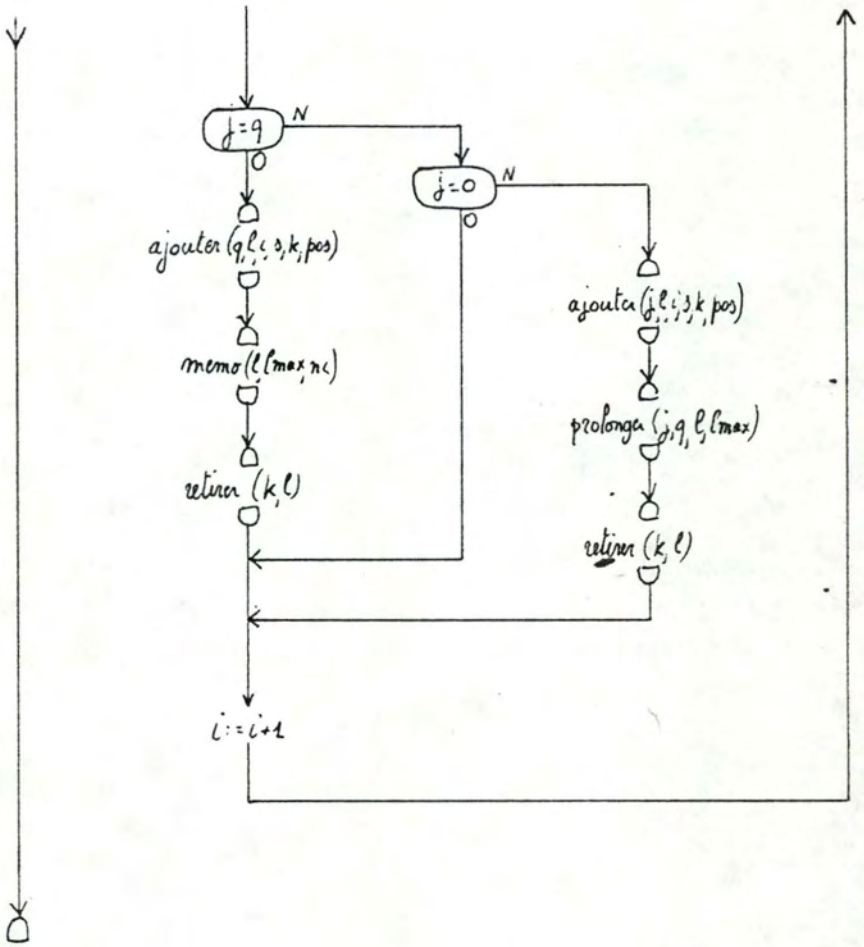
- De plus, lors de la mémorisation d'un chemin, si la longueur de ce chemin est inférieure à longueur du plus petit chemin calculé jusqu'alors, la valeur de lmax est mise à jour avec la valeur de la longueur du nouveau chemin et les chemins de longueur supérieure à la nouvelle valeur de lmax sont supprimés.

- Au départ, lmax aura encore comme valeur n.

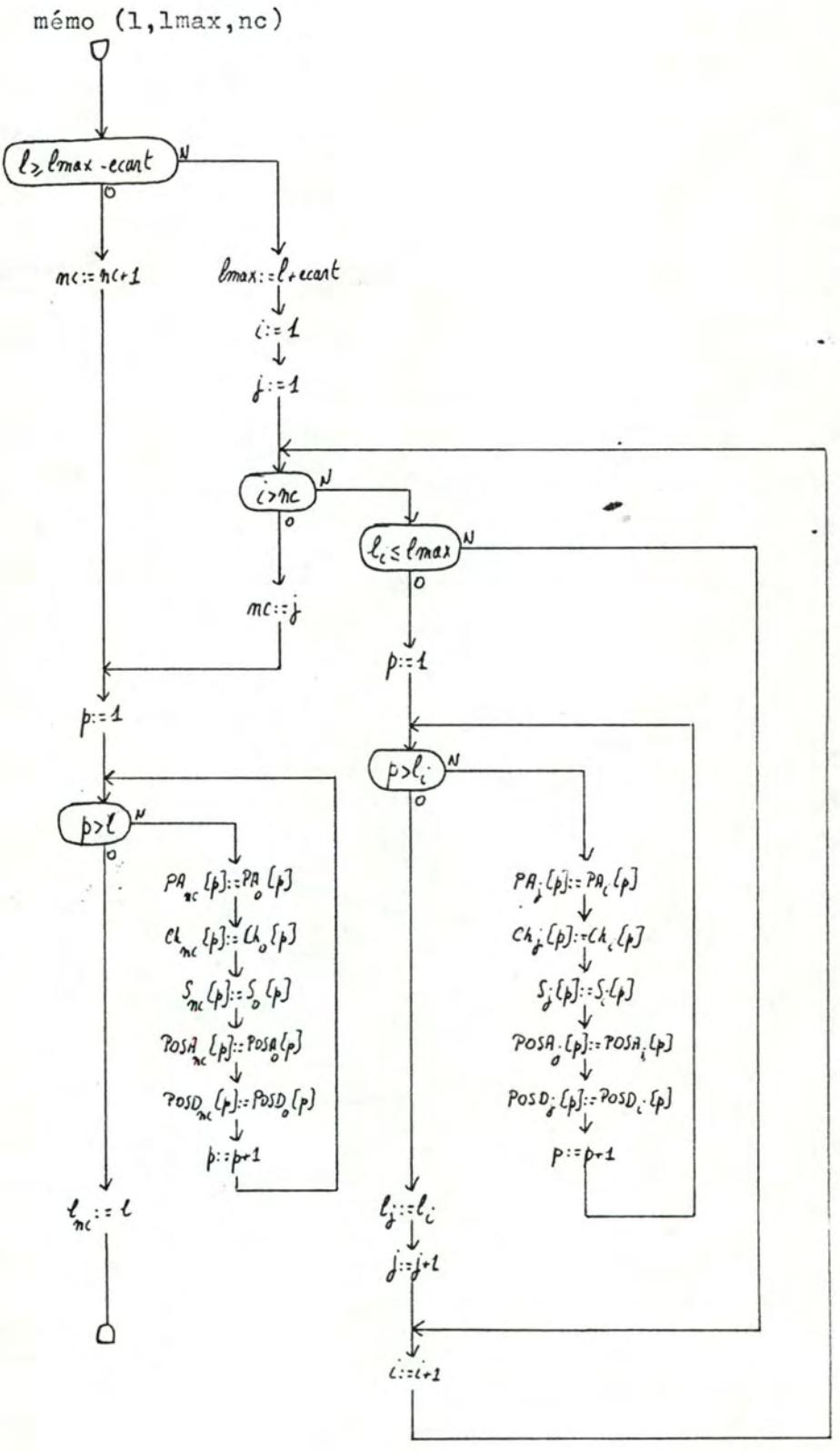
- Algorithme du problème particulié modifié



suite de page précédente



- Algorithme du sous problème modifié: mémoriser



3.3 - PHASE 2 : Recherche du chemin ayant un temps de de parcours minimal.

3.3.1. Introduction

- L'utilisateur accompagne sa demande de recherche d'itinéraire de plusieurs renseignements nécessaires à cette recherche.

  Tout d'abord, il indique une date pour laquelle la recherche doit être faite, ainsi que certaines informations relatives à cette date:

  - jour correspondant de la semaine
  - type de jour: jour férié
    congé scolaire
    congé payé.

  Ensuite il exprime son choix quand au type de recherche des heures de passage.

  - soit il désire partir au plus tôt pour une telle heure auquel cas il indique une heure minimale.
  - soit il désire arriver au plus tard pour une telle heure, auquel cas il indique une heure maximale.

- Le choix d'heure minimale ou maximale est important, car la méthode de recherche des heures de départ et d'arrivée aux points d'arrêt formant l'itinéraire sera différente selon le choix.
  En effet, si une heure minimale de départ est indiquée, la recherche se fera en partant du point de départ, tandis que si une heure maximale d'arrivée est indiquée, la recherche se fera en remontant le chemin à partir des points d'arrivée. C'est pourquoi il y aura un algorithme distinct pour chaque genre de recherche.

- Il est noter que parfois il n'existera pas d'horaire associé à une liaison, soit parce qu'il s'agit d'une nouvelle liaison qui vient d'être créée et dont l'horaire n'a pas encore été introduit; soit parce qu'il s'agit d'une liaison qui vient d'être modifiée et dont les modifications n'ont pas encore été représentées dans l'horaire. Dans ce cas, il n'est pas possible de déterminer les heures de passage sur cette liaison. Il n'est donc pas possible de calculer le temps de parcours d'un chemin empruntant cette liaison.

3.3.2. Algorithme de recherche des heures de passage et du temps de parcours d'un chemin.

En fait, il y a 2 algorithmes, chacun correspondant à un des deux cas suivants:

Cas 1 : heure minimale de départ
Cas 2 : heure maximale d'arrivée

- Cas 1

Passer en revue les triples successifs qui forment le chemin.

Pour chaque triple (j_k, i_k, s_k), $k = \emptyset$ jusque l-1

Si un horaire de la liaison l_{i_k} est disponible

Chercher dans l'horaire le passage tel que

. le sens du passage soit celui indiqué par s_k
. il y ait un arrêt effectif au point d'arrêt a_{j_k}, et au point d'arrêt jusqu'ou la liaison l_{i_k} doit être prise, c.à.d. $a_{j_{k+1}}$.

. la différence entre l'heure minimale et l'heure de passage au point d'arrêt a_{j_k} soit minimale, avec l'heure de passage supérieure à l'heure minimale.

. le passage soit compatible avec la date pour laquelle le temps de parcours est calculé.

Mémoriser l'heure de départ de a_{j_k} et l'heure d'arrivée en $a_{j_{k+1}}$

Passer au triple suivant $(j_{k+1}, i_{k+1}, s_{k+1})$

sinon, abandonner la recherche.

- Cas 2

Passer en revue les triples successifs qui forment le chemin, dans l'ordre inverse du chemin.

Pour chaque triple (j_k, i_k, s_k), k = 1 jusque 1

Si un horaire de la liaison $l_{i_{k-1}}$ est disponible

- Chercher dans l'horaire le passage tel que

. le sens du passage soit celui indiqué par s_k

. il y ait un arrêt effectif au point d'arrêt a_{j_k} et au point d'arrêt à partir duquel la liaison $l_{i_{k-1}}$ doit être prise c.à.d. à $a_{j_{k-1}}$

. la différence entre l'heure de passage au point a_{j_k} et l'heure maximale, soit minimale, avec l'heure de passage inférieure à l'heure maximale.

. le passage soit compatible avec la date pour laquelle le temps de parcours est calculé.

- Mémoriser l'heure de départ de $a_{j_{k-1}}$ et l'heure d'arrivée en a_{j_k} .
- Ajouter le temps d'attente (heure maximale - heure d'arrivée en a_{j_k}) et le temps de trajet $a_{j_{k-1}} \rightarrow a_{j_k}$ au temps total de parcours du chemin.
- Remplacer l'heure maximale par l'heure de départ de $a_{j_{k-1}}$
- Passer au triple suivant $(j_{k-1}, i_{k-1}, s_{k-1})$

sinon, abandonner la recherche.

3.3.3. Expression des algorithmes en pseudo langage

- Données permanentes.

On doit pouvoir accéder en permanence aux horaires des liaisons et aux caractéristiques de chaque passage (par exemple "supprimé le dimanche", "uniquement les jours scolaires", etc...)

On crée d'abord deux tableaux.

. NP^1 $[1..n]$ donnant pour chaque liaison le nombre de passages dans le sens direct de la liaison, c'est-à-dire

$NP^1[j] = k$

ssi il y a exactement k passage pour la liaison l_j dans le sens $a_{L_j(1)} \rightarrow a_{L_j(NA[j])}$

- NP^2 [1..n] donnant pour chaque liaison le nombre de passage dans le sens indirect de la liaison,

c'est-à-dire

NP2 [j] = k

ssi il y a exactement k passage pour la liaison l_j dans le sens $a_{L_j}(NA[j]) \rightarrow a_{L_j}(1)$

Ensuite on associe à chaque liaison deux matrices et deux tableaux.

H_i^1 [1..NA[i] , 1..NP^1[i]]

représentant les temps de passage aux différents points d'arrêt constituant la liaison l_i dans le sens direct de la liaison: Si il n'y a pas un arrêt effectif lors d'un passage en un point d'arrêt, alors l'élément correspondant aura une valeur "nulle",

c'est-à-dire

H_i^1 [k,p] = h

ssi

le $p^{ème}$ passage dans le sens direct de la liaison l_i au point d'arrêt $a_{L_i}(k)$ se fait à l'heure k (c.à.d. au point d'arrêt se trouvant en $k^{ème}$ position dans le sens direct)

CAK_i^1 [1..NP^1[i]] représentant sous forme codée les caractéristiques des passages de la liaison l_i dans le sens direct.

H_i^2 [1..NA[i], 1..NP^2[i]] idem H_i^1, mais dans le sens

$^{a}L_i$ (NA [i]) $\rightarrow$ $^{a}L_i$ (1)

c'est-à-dire

H_i^2 [k,p] = h

ssi

le $p^{ème}$ passage dans le sens indirect de la liaison l_i au point d'arrêt $a_{L_i}(k)$ se fait à l'heure h, c.à.d. au point d'arrêt se trouvant en $k^{ème}$ position dans le sens indirect).

CAR_i^2 [1..NP^2[i]] idem CAR_i^1, mais dans le sens

$^{a}L_i$ (NA [i]) $\rightarrow$ $^{a}L_i$ (1)

De plus on crée un nouveau tableau associé aux liaisons

D [1..n] indiquant pour chaque liaison si un horaire correct est disponible,

c'est-à-dire

D [i] = 'o' ssi un horaire correct de la liaison l_i est disponible sinon D [i] = 'n'.

Remarque sur H_i^1 et H_i^2

Si on parcours ces deux matrices de haut en bas, c.à.d. le long des colonnes, les éléments de H_i^1 sont alors classés par ordre croissant tandis que les éléments de H_i^2 sont classés par ordre décroissant.

- Tableaux et variables utilisés.

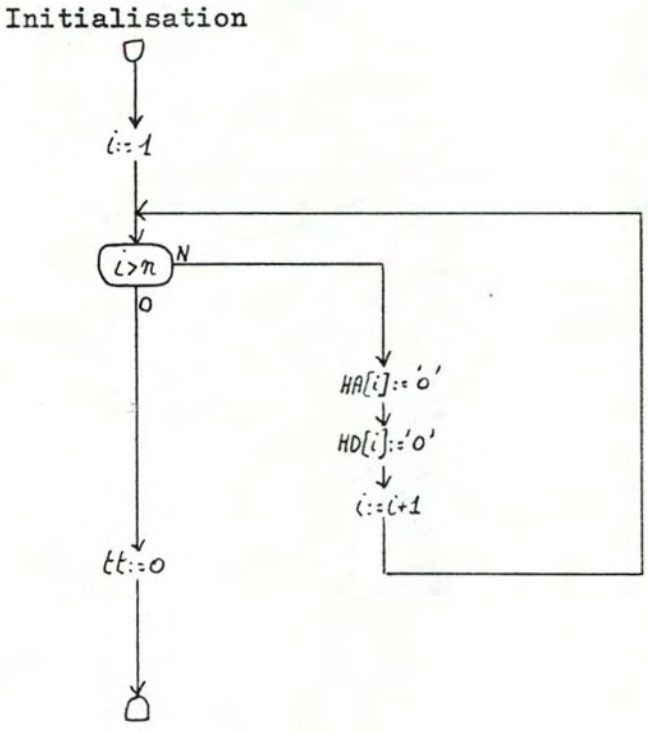
  HD [1..n] indiquant les heures de départ en chaque point d'arrêt du chemin.

  HA [1..n] indiquant les heures d'arrivée en chaque point d'arrêt du chemin.

  tt: temps total de parcours du chemin.

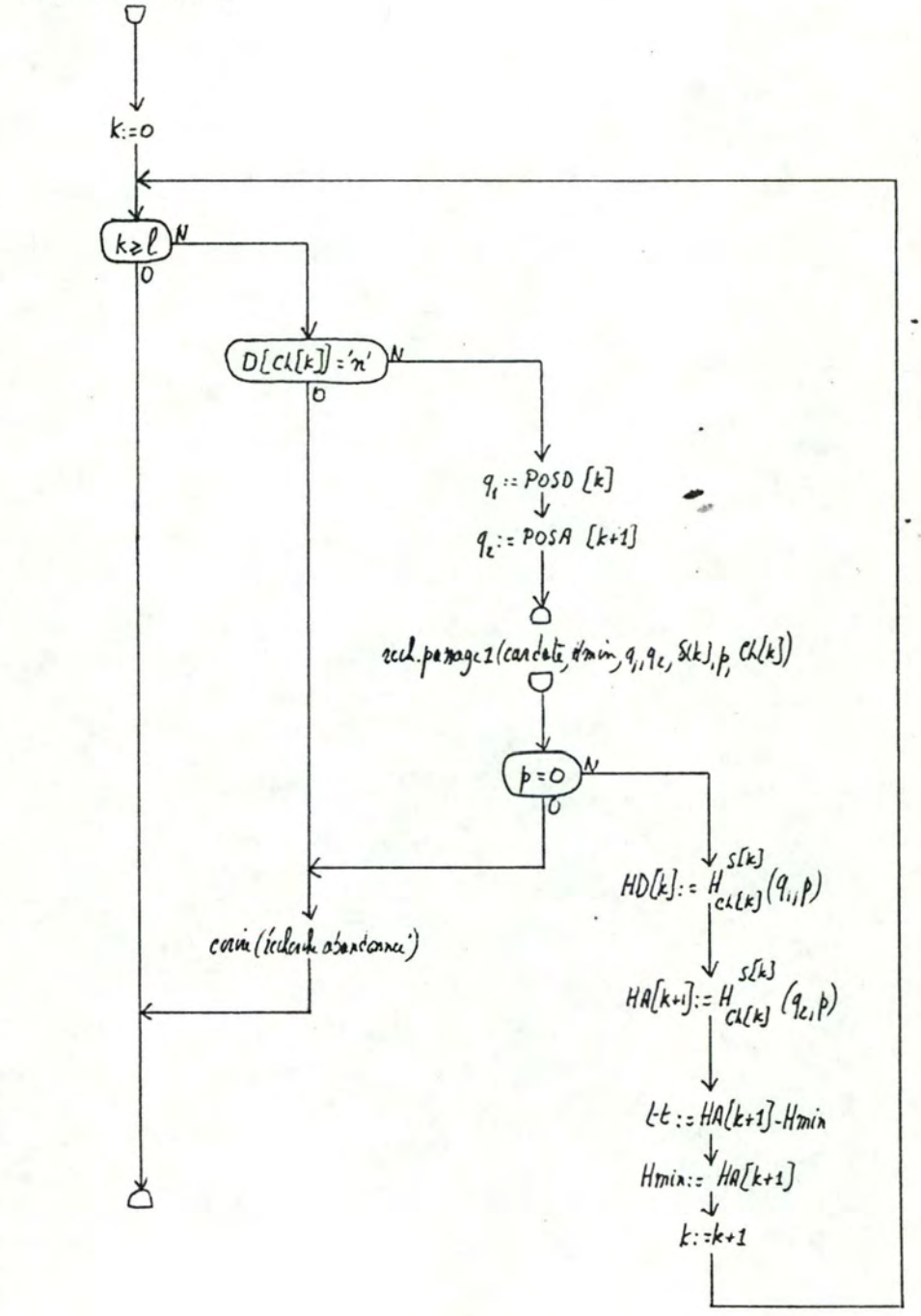
- Initialisation

  Au départ, les heures de passage sont inexistantes, donc les deux tableaux HA et HD doivent avoir tous leurs éléments nuls et le temps total de parcours doit aussi être nul.



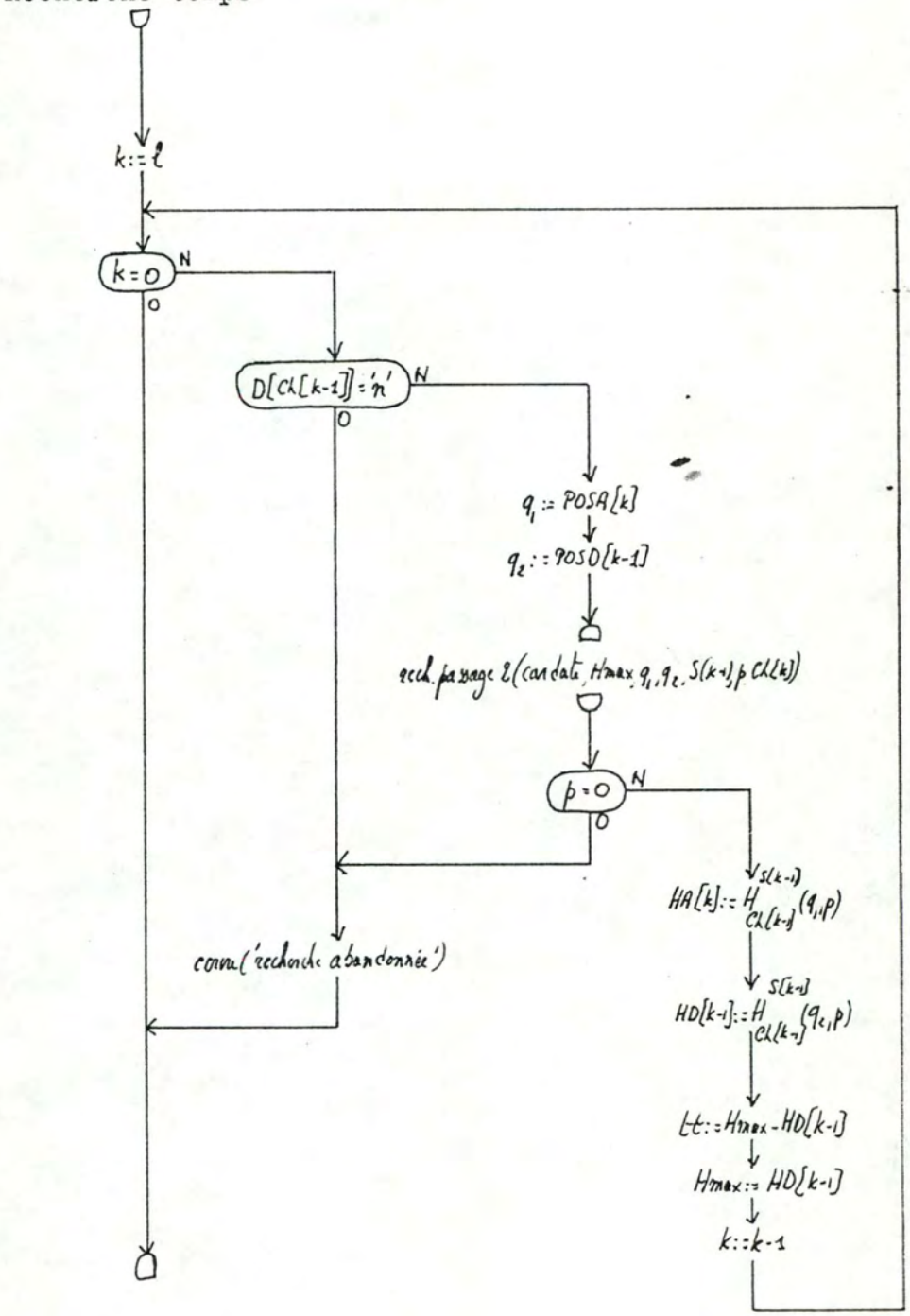
- Algorithme - Cas 1

Recherche temps



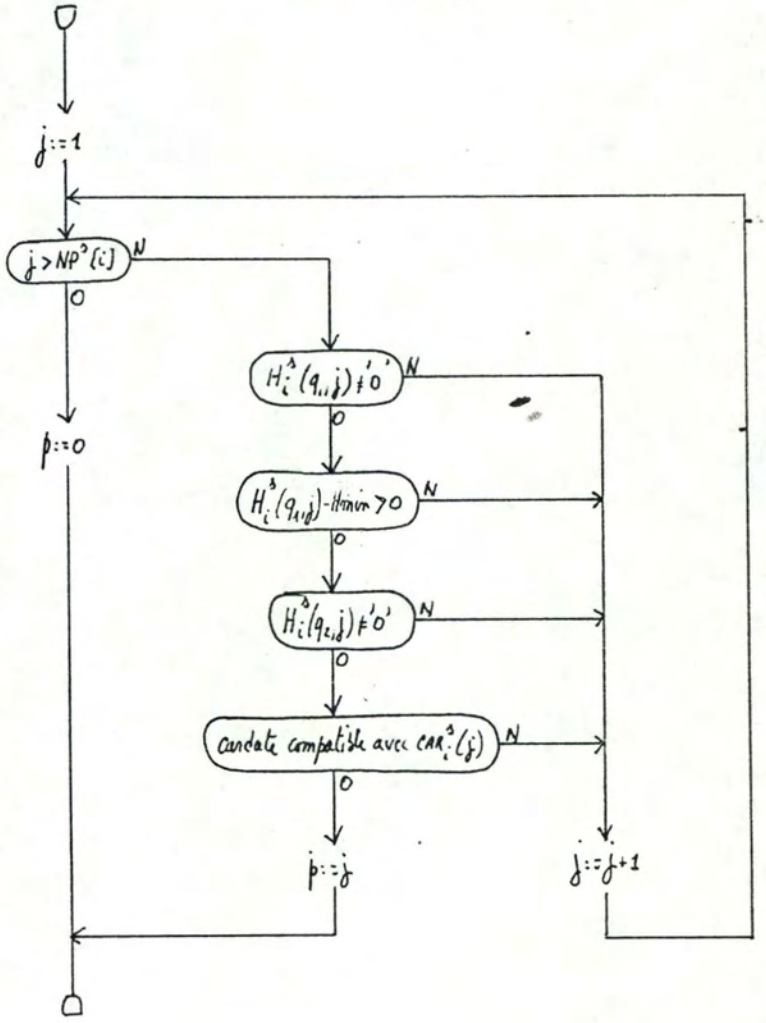
- Algorithme - Cas 2

Recherche temps



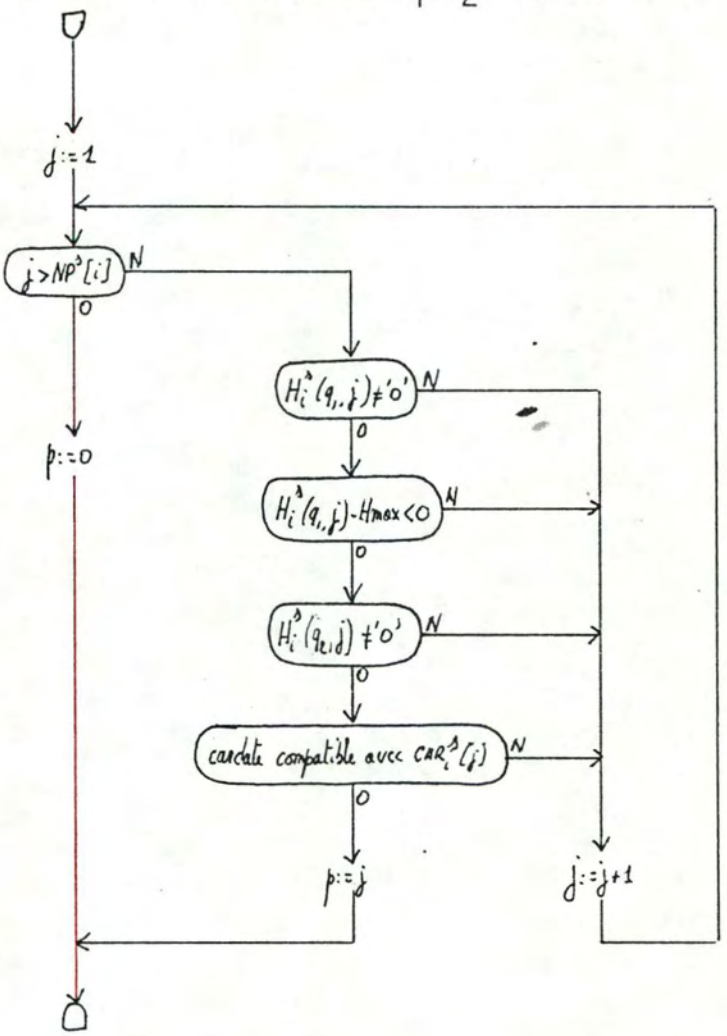
- Algorithme du sous problème: rechercher le passage dans le cas 1

Rech. passage 1 (cardate, Hmin, q_1, q_2, s, p, i)



- Algorithme du sous problème: rechercher le passage dans le cas 2

Rech. passage 2 (cardate, Hmax, q_1, q_2, s, p, i)



3.3.4. Algorithme de recherche du chemin ayant le temps de parcours minimum.

- La méthode utilisée pour la recherche du chemin ayant le temps de parcours minimum, suit d'assez près la méthode utilisée pour la recherche du chemin de poids minimal.

- On utilise une variable tmin qui, à tout moment, représente le temps de parcours minimum. Cette variable a au départ une valeur assez grande qui assure l'existence d'au moins un chemin dont le temps de parcours sera inférieur à tmin.

- L'algorithme de recherche du temps de parcours d'un chemin est modifié de telle manière, que lorque le temps de parcours intermédiaire calculé dépasse la valeur de tmin, la recherche est abandonnée.

- Algorithme de recherche du chemin de temps de parcours minimal.

  Passer en revue les chemins calculés lors de phase 1

  Pour chaque chemin:

    Initialiser les temps de passage et le temps de parcours du chemin.
    Rechercher le temps de parcours.
    Si le temps de parcours est inférieur à tmin

      Remplacer le meilleur chemin intermédiaire par celui dont on vient de calculer le temps de parcours (Chemin et temps de passage)

      Remplacer tmin par le temps de parcours du nouveau meilleur chemin.

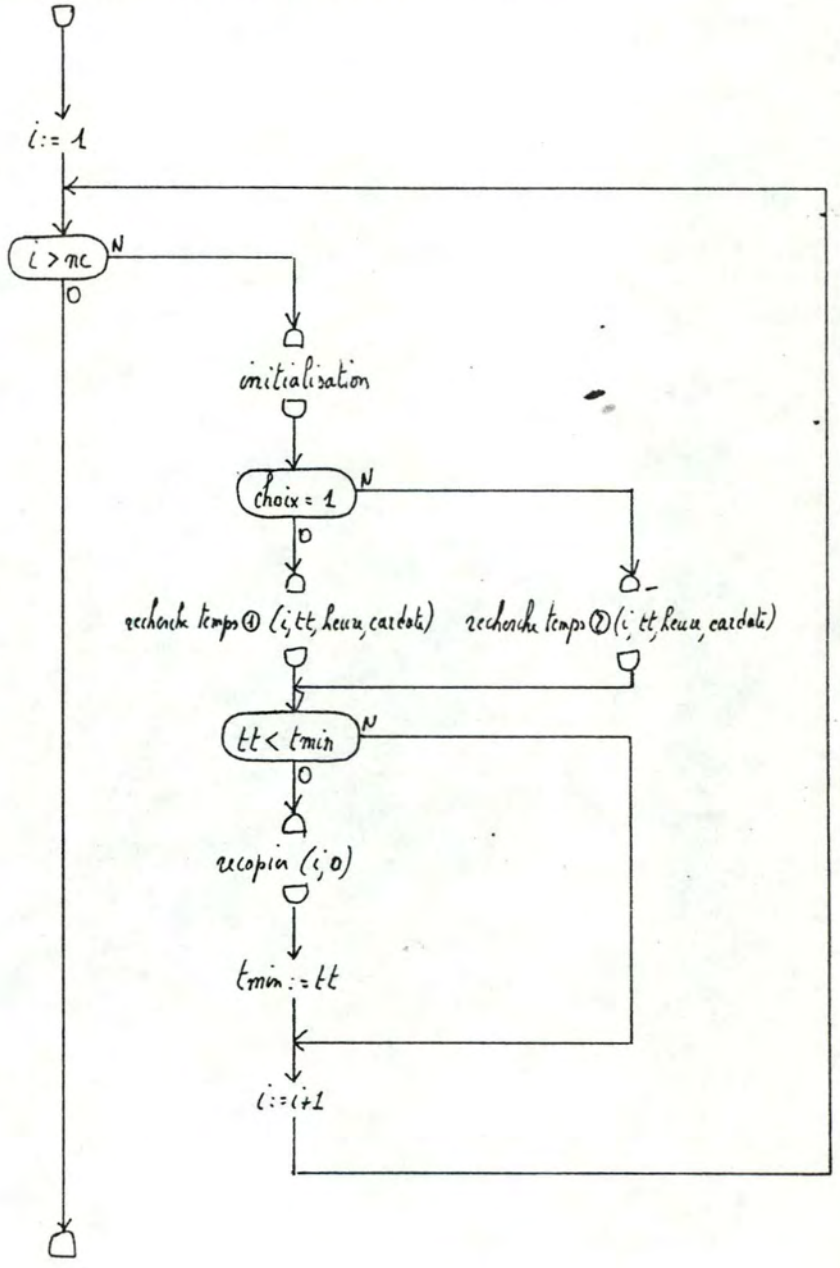
      Passer au chemin suivant.

    Sinon, passer au chemin suivant.

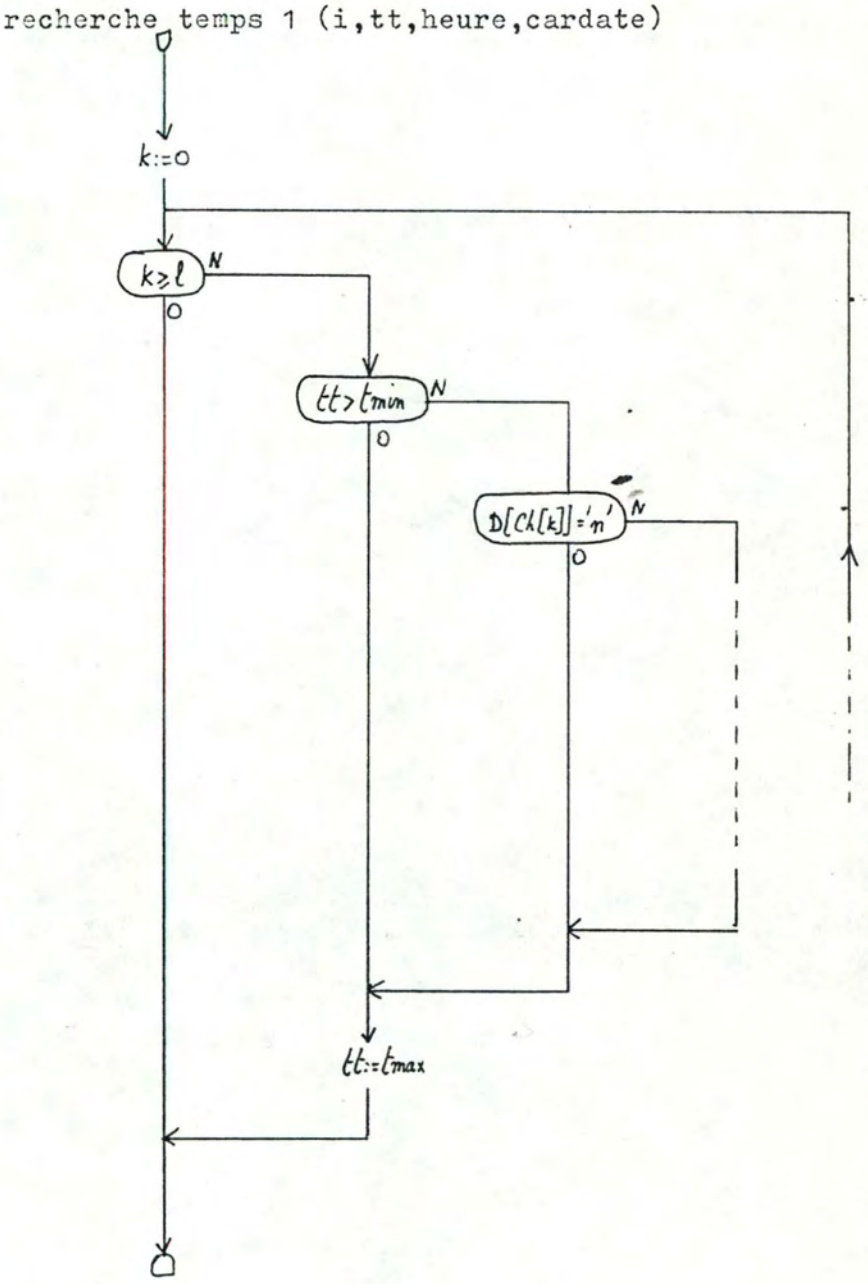
3.3.5. Expression de l'algorithme en pseudo langage

- Algorithme de recherche du meilleur chemin.

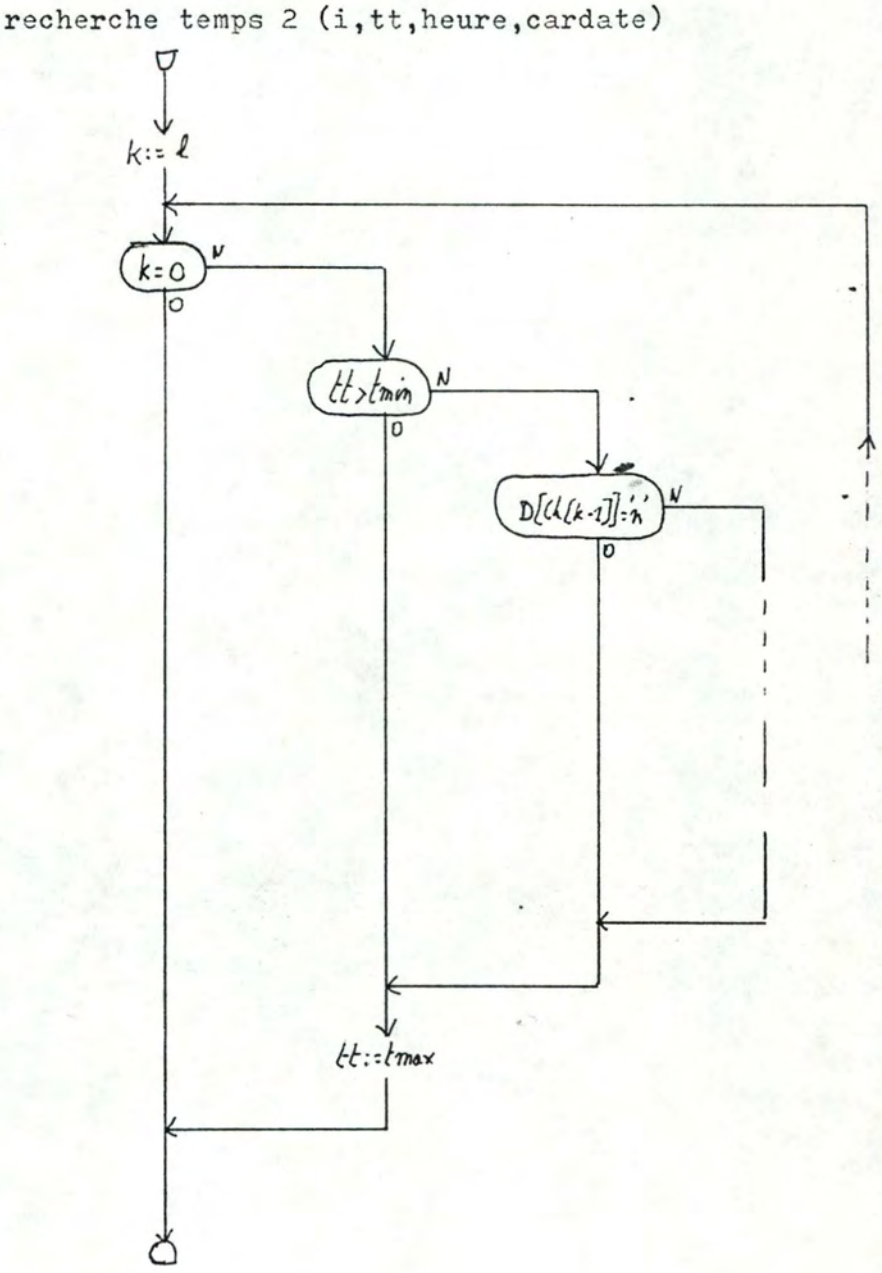
Rech. itinéraire (heure,cardate,choix)



- Algorithme de recherche du temps de parcours modifié cas 1



- Algorithme de recherche du temps de parcours modifié cas 2



CHAPITRE 4 - Description de l'environnement

4.0. Introduction

Le but de ce chapitre est de déterminer l'environnement nécessaire pour pouvoir appliquer la méthode décrite dans le chapitre précédent.

La première partie du chapitre définit la structure et le type des données, qu'elles soient permanentes (introduites une fois pour toutes: le contenu des liaisons, la liste des points d'arrêt, les horaires associés aux liaisons et toutes les informations s'y rapportant), ou temporaires (tableaux et variables créés et utilisés lors d'une application).

La seconde partie montre la correspondance entre les notations utilisées dans les algorithmes du chapitre précédent et ces structures de données.

La troisième partie décrit les données particulières à une application et le type de contraintes sur ces données.

La quatrième partie présente l'algorithme général qui permet de résoudre un problème particulier posé par l'utilisateur.

La cinquième partie enfin, présente l'expression en pseudo langage de cet algorithme et des sous problèmes qui en découlent.

4.1 Structures de données

4.1.1. Données permanentes

- On distingue trois types de données permanentes:

  1. les liaisons
  2. les points d'arrêt
  3. les horaires des liaisons

- Les renseignements concernant une liaison sont les suivantes:

  1. le numero de la liaison, c'est-à-dire le numero sous lequel cette liaison est connue dans le réseau SNCB/SNCV.
  2. la liste des points d'arrêt constituant la liaison.
  3. le nombre de points d'arrêt constituant la liaison.
  4. quelque chose permettant d'affirmer si un horaire correct de la liaison est disponible.

- Les renseignements concernant un point d'arrêt sont les suivants:

  1. la dénomination de ce point d'arrêt, c'est-à-dire la localité ou se situe le point d'arrêt et l'endroit précis dans la localité.
  2. le nombre de liaisons passant par ce point d'arrêt.
  3. la liste des liaisons passant par ce point d'arrêt ainsi que la position de ce point d'arrêt dans la liaison.

- Les renseignements concernant un horaire sont les suivants:

  1. le numero de la liaison concernée.
  2. le nombre de points d'arrêt constituant cette liaison.

3. la liste des points d'arrêt constituant cette liaison
4. le nombre de passage dand le sens direct.
5. le nombre de passage dans le sens indirect.
6. pour chaque passage dans le sens direct, les caractéristiques du passage et les heures de passage aux différents point d'arrêt.
7. pour chaque passage dans le sens indirect, les caractéristiques du passage et les heures de passage aux différents points d'arrêt.

Remarques:

- On appelle passage le parcours effectif de la liaison par un véhicule au cours d'une journée.

- Les caractéristiques d'un passage sont les limitations apportées quant aux jours de circulation. Etant donné le nombre important de ces limitations, ainsi que leurs caractères trop précis parfois (par ex.: avance de 8 minutes les jours de foire à Bastogne), on restreint ces limitations à celles reprises dans un indicateur horaire sous le sigle "signes conventionnels" c'est-à-dire

Circule uniquement	le dimanche	†
	les samedi et dimanche	ⓒ
	le lundi	①
	le mardi	②
	le mercredi	③
	le jeudi	④
	le vendredi	⑤
	le samedi	⑥
	le dimanche mais pas les jours fériés	⑦

Ne circule pas	le dimanche	
	le samedi et le dimanche	
	le samedi	
	pendant la période des congés annuels	
	les jours de vacances scolaires, ni le samedi et dimanche	

On ajoute cependant quelques limitations qui apparaissent assez fréquemment:

Ne circule pas	les jours scolaires
	le lundi
	le mardi
	le mercredi
	le jeudi
	le vendredi

- Pour définir les types des structures de données, les conventions de notation suivantes sont prises:

  N(l) : chaine de caractères numériques, de longueur l

  Z(l) : chaine de caractères alphanumériques de longueur l

  A(l) : chaine de caractères alphabétiques, de longueur l

  T [i..j] : tableau dont les indices peuvent prendre une valeur comprise entre i et j .

- Les structures de données sont les suivantes:

LIAISON

NUM_ID : entier
NUMERO : Z (6)
NBRE_POINT_ARRET : entier
DISP_HORAIRE : Booleen (vrai ou faux)
LISTE_POINT.ARRET : T [1..NBRE.POINT.ARRET] d'entier

POINT_ARRET

NUM_ID : entier
NOM
    LOCALITE : A (20)
    ENDROIT : A (20)
NBRE_LIAISON : entier
LISTE_LIAISON : T [1..NBRE_LIAISON] d'entier
POS_LIAISON : T [1..NBRE_LIAISON] d'entier

GRILLE_HORAIRE

NUM_ID : entier
NUMERO : Z (6)
NBRE_POINT.ARRET : entier
LISTE_POINT.ARRET : T [1..NBRE_POINT_ARRET] d'entier
NP1 : entier
NP2 : entier
HORAIRE 1 : T [1..NP1] de PASSAGE
HORAIRE 2 : T [1..NP2] de PASSAGE

PASSAGE
    CARACT. N (10)
    HEURE_PASSAGE : T [1..NBRE.POINT.ARRET] d'HEURE
        HEURE : N (4)

Remarque sur CARACT

Chaque caractère de CARACT indique s'il y a passage ou non, pour un type de jour, les correspondances entre caractère et type de jour étant les suivantes:

1er caractère : LUNDI
2ème caractère : MARDI
3ème caractère : MERCREDI
4ème caractère : JEUDI
5ème caractère : VENDREDI
6ème caractère : SAMEDI
7ème caractère : DIMANCHE
8ème caractère : JOUR FERIE
9ème caractère : JOUR SCOLAIRE
10me caractère : CONGES ANNUELS

ainsi un passage supprimé les samedis, dimanches et jours scolaires aura comme caractéristique

1	1	1	1	1	0	0	1	0	1

Remarque sur HEURE

Les caractères de HEURE doivent satisfaire aux conditions suivantes:

les deux premiers caractères doivent être compris entre '00' et '23'

les deux derniers caractères doivent être compris entre '00' et '59'.

4.1.2. Données temporaires

- Les différentes données temporaires sont celles concernant un itinéraire, c'est-à-dire un chemin à travers le réseau, et les heures de passage aux différents points d'arrêt formant le chemin. Il est évident qu'au départ nul ne saurait dire la longueur qu'aura un chemin, celle-ci pouvant aller de 1 pour un chemin joignant deux points d'arrêt appartenant à une même liaison jusqu'à un nombre qui peut être supérieur un nombre de liaisons. Il n'est donc pas possible de considérer des tableaux pour conserver la trace d'un chemin. Il faut passer par des structures de données dynamiques, c'est-à-dire pouvant être étendu indéfiniment. C'est pourquoi on utilise la notion de "pointeur".

On peut définir un pointeur comme étant une donnée particulière, indiquant comment accéder à une donnée d'un autre type. Ainsi, si on suppose que toute donnée possède une adresse qui lui est propre, un pointeur vers cette donnée aura pour valeur l'adresse de la donnée.

On utilise les conventions suivantes:

- si DONNEE est un type de donnée,

alors on note un type de donnée POINTEUR vers DONNEE : ↑DONNEE.

p. ex. : POINTEUR_DONNEE: ↑ DONNEE

- l'objet pointé par POINT de type POINTEUR_DONNEE sera désigné par (POINTEUR_DONNEE)↑

donc (POINT)↑ est de type DONNEE.

On peut représenter les pointeurs de la manière suivante:

- On distingue 4 types de données temporaires

  1. Une liste de chemins
  2. Une liste de parties de chemin, formant un chemin
  3. Une liste d'heures de passage associée à un chemin
  4. Un itinéraire

- Les renseignements concernant un chemin dans la liste des chemins sont les suivants:

  1. La longueur du chemin
  2. Le moyen de retrouver les différentes parties qui forment le chemin (dans les deux sens)
  3. Le moyen de retrouver le chemin suivant dans la liste.

- Les renseignements concernant une partie de chemin dans une liste de parties de chemins sont les suivants:

  1. Le point d'arrêt
  2. La liaison empruntée à partir du point d'arrêt
  3. Le sens de la liaison
  4. La position du point d'arrêt dans la liaison utilisée pour y arriver
  5. La position du point d'arrêt dans la liaison utilisée pour le quitter
  6. Le moyen de retrouver la partie de chemin suivante dans la liste
  7. Le moyen de retrouver la partie de chemin précédente dans la liste.

- Les renseignements concernant les heures de passage dans une liste d'heures de passage associée à un chemin sont les suivants :

  1. L'heure d'arrivée en un point d'arrêt du chemin
  2. L'heure de départ d'un point d'arrêt du chemin
  3. Le moyen de retrouver les heures de passage suivantes dans la liste
  4. Le moyen de retrouver les heures de passage précédentes dans la liste .

- Les renseignements concernant un itinéraire sont les suivants:

  1. La longueur du chemin
  2. Le temps de parcours du chemin
  3. Le moyen de retrouver les parties de chemin formant le chemin
  4. Le moyen de retrouver les heures de passage associées au chemin dans les deux sens.

- Les structures de données sont les suivantes:

```
PARTIE CHEMIN

    POINT_ARRET : entier
    LIAISON : entier
    SENS : entier
    POSA : entier
    POSD : entier
    SUIV : ↑ PARTIE_CHEMIN
    PREC : ↑ PARTIE_CHEMIN
```

LISTE_CHEMIN

LONG : entier
DEB_CHEMIN : ↑ PARTIE_CHEMIN
FIN_CHEMIN : ↑ PARTIE_CHEMIN
SUIV : CHEMIN

HEURE CHEMIN

HEURE_DEPART : N(4)
HEURE_ARRIVEE : N(4)
SUIV : ↑ HEURE_CHEMIN
PREC : ↑ HEURE_CHEMIN

ITINERAIRE

LONG : entier
TEMPS_PARCOURS : entier
DEB_CHEMIN : ↑ PARTIE_CHEMIN
DEB_HEURE_CHEMIN : ↑ HEURE_CHEMIN
FIN_CHEMIN : ↑ PARTIE_CHEMIN
FIN_HEURE_CHEMIN : ↑ HEURE_CHEMIN

CHEMIN

LONG : entier
DEB_CHEMIN : ↑ PARTIE_CHEMIN
FIN_CHEMIN : ↑ PARTIE_CHEMIN

4.2. Correspondances

4.2.1. Notations

NOM_DE_DONNEE . NOM_D_ITEM: signifie l'item NOM_D_ITEM de la donnée NOM_DE_DONNEE

ex. LIAISON . NUMERO signifie le NUMERO de la LIAISON

NOM DE DONNEE (:NOM_D_ITEM = 'valeur') signifie l'occurence se NOM_DE_DONNEE dont la valeur de NOM_D_ITEM égale 'valeur'

ex. LIAISON (:NUMERO = '126A1') signifie la liaison dont le NUMERO égal '126A1' .

4.2.2. Données permanentes

$A_i(j)$ <=> POINT_ARRET (:NUM_ID=i).LISTE_LIAISON (j)

$POSL_i(j)$ <=> POINT_ARRET (:NUM_ID=i).POS_LIAISON (j)

NL (j) <=> POINT_ARRET (:NUM_ID=j).NBRE_LIAISON

D (j) <=> POINT_ARRET (:NUM_ID=j).DISP_HORAIRE

L_j (i) <=> LIAISON (:NUM_ID=j).LISTE_POINT_ARRET (i)

NA (j) <=> LIAISON (:NUM ID=j).NBRE_POINT_ARRET

NP^1 (i) <=> GRILLE_HORAIRE (:NUM_ID=i).NP1

NP^2 (i) <=> GRILLE_HORAIRE (:NUM_ID=i).NP2

H_i^1 (p,q) <=> GRILLE_HORAIRE (:NUM_ID=i).HORAIRE1 (p). HEURE_PASSAGE(q)

H_i^2 (p,q) <=> GRILLE_HORAIRE (:NUM_ID=i).HORAIRE2 (p). HEURE_PASSAGE(q)

CAR_i^1 (p) <=> GRILLE_HORAIRE (:NUM_ID=i).HORAIRE1(p).CARACT

CAR_i^2 (p) <=> GRILLE_HORAIRE (:NUM_ID=i).HORAIRE2(p).CARACT.

4.3. Données particulières à une application

- Les données particulières à une application, sont celles qui correspondent aux renseignements relatifs aux désirs d'un utilisateur, c'est-à-dire

  . un point d'arrêt de départ
  . un point d'arrêt d'arrivée
  . un choix relatif aux contraintes sur l'itinéraire calculé
  . une heure qui sera minimale ou maximale
  . des caractéristiques relatives à la date pour laquelle l'itinéraire doit être calculé

- L'utilisateur ne connait un point d'arrêt que sous la forme d'un nom composé du nom de la localité ou se situe le point d'arrêt, et de l'endroit précis ou se trouve ce point d'arrêt.

  par exemple: nom de localité: NAMUR
  endroit : GARE

  Donc les points d'arrêt de départ et d'arrivée ne pourront être introduit par l'utilisateur que sous cette forme composée, d'ou deux données particulières:

  ARRET_DEPART
  ARRET_ARRIVEE

  toutes deux composées de

  LOCALITE: A (20)
  ENDROIT : A (20)

  Il est bien clair que ces deux points d'arrêt introduits doivent correspondre à deux points d'arrêt existant déjà comme données permanentes; de plus, ils doivent être distincts, car il est inutile de vouloir rechercher l'itinéraire reliant un point d'arrêt à ce même point d'arrêt.

- L'utilisateur doit alors indiquer son choix, une fois que les deux points d'arrêt ont été introduits. Ce choix, c'est soit partir au plus tôt pour une telle heure, soit arriver au plus tard pour une telle heure. La donnée CHOIX peut donc avoir deux valeurs entières: 1 ou 2, correspondant au cas (1) ou au cas (2).

- Ensuite l'utilisateur indique une heure qui est minimale ou maximale selon son choix, d'ou une nouvelle donnée particulière:

HEURE_CHOIX: N (4)

devant satisfaire aux conditions posées sur n'importe quelle heure, c'est-à-dire les deux premiers caractères doivent être compris entre '00' et '23' ; les deux derniers caractères doivent être compris entre '0,0' et '59'

- La dernière donnée particulière à une application concerne les caractéristiques d'une date. Cette donnée doit pouvoir être comparée aux caractéristiques associées à un passage d'une liaison, d'ou cette donnée est du même type que CARACT

CAR_DATE : N (10)

Chaque caractère pourra avoir comme valeur soit '1' soit '0' .

4.4. Algorithme général d'une application

4.4.1. L'algorithme général comporte 3 phases distinctes.
D'abord, il y a la phase d'introduction et de validation des données particulières à l'application; cette phase sera donc interactive.
Ensuite, ily a la phase de recherche de l'itinéraire correspondant aux données particulières.
Enfin, il y a phase d'édition des résultats, c'est-à-dire de l'itinéraire calculé par la phase 2 de l'algorithme général.

4.4.2. Phase 1.- Introduction et validation des données particulières.

- Les données particulières POINT_DEPART et POINT_D_ARRIVEE sont composées de LOCALITE et ENDROIT. Il est évident que parfois l'utilisateur ne connait d'une localité que l'endroit précis ou il se trouve ou qu'il désire rejoindre. Cet endroit ne correspond malheureusement pas toujours à un point d'arrêt appartenant au réseau. Il doit donc être possible à l'utilisateur de faire un choix parmi les points d'arrêt du réseau situés dans la localité qu'il désire quitter ou qu'il désire rejoindre. Il est donc nécessaire de pouvoir lui fournir une liste de ces points d'arrêt.
De plus, les algorithmes de la phase 2 utilisent comme données non pas un nom, mais un numero d'identification de point d'arrêt ($a_p \rightarrow p$), donc lorqu'un point d'arrêt a été introduit sous forme d'un nom, il faut établir la correspondance entre ce nom de point d'arrêt et son numero d'identification, ce qui est fait lors de la validation. Si le numero d'identification retourné est nul, alors il n'existe pas dans la liste des points d'arrêt une occurence de point d'arrêt dont le nom est

celui introduit par l'utilisateur. Dans ce cas, l'utilisateur doit pouvoir réintroduire un nouveau nom. Il doit encore être possible à l'utilisateur d'abandonner sa demande, en introduisant par exemple un nom fictif.

- L'introduction de CHOIX ne pose pas de problèmes. Sa validation non plus, puisqu'il s'agit juste de tester si la valeur introduite est bien égale à 1 ou 2.
  On peut rajouter la possibilité d'abandonner la demande à ce moment en introduisant comme valeur de CHOIX, une valeur nulle.

- L'introduction de HEURE_CHOIX ne pose pas non plus de problème. Pour la validation, il suffit de vérifier que la valeur introduite correspond bien à une heure. Comme dans les autres introductions de données, on peut ajouter la possibilité d'abandonner la demande en introduisant une heure nulle ('0000').

- Il reste à introduire les caractéristiques de la date pour laquelle la demande est faite. Les limitations sur le passage d'une liaison portent sur le type de jour, il est donc normal de donner comme caractéristiques d'une date, le type de jour correspondant à la date. L'introduction se fait par l'intermédiaire de questions posées à l'utilisateur. Ces questions sont les suivantes, chacune correspondant à un type de jour

  JOUR DE LA SEMAINE: ? ('Lu','Ma','Me','Je','Ve','Sa','Di'

  JOUR FERIE: ? ('o','n')

  JOUR SCOLAIRE: ? ('o','n')

  CONGES ANNUELS: ? ('o','n')

Le caractère de CARDATE correspond au jour répondu pour la première question prend la valeur '1'

('Lu': premier caractère, 'Ma': 2ème caractère,...)

Le 8ème caractère de CARDATE est mis à '1' si la réponse à la 2ème question est 'o'

à 'o' sinon

Le 9ème caractère de CARDATE est mis à '1' si la réponse à la 3ème question est 'o'

à 'o' sinon

Le 10me caractère de CARDATE est mis à '1' si la réponse à la 4ème question est 'o'

à 'o' sinon

4.4.3. Phase 2.- Recherche de l'itinéraire

Cette phase se limite à l'application des deux algorithmes de recherche de chemin, avec les données particulière.

4.4.4. Phase 3.- Edition de résultats

L'itinéraire est édité liaison par liaison, en donnant pour chaque liaison

- le numero de la liaison
- le point d'arrêt de départ de la liaison
- l'heure de départ de ce point d'arrêt
- le point d'arrêt d'arrivée de la liaison
- l'heure d'arrivée au point d'arrêt

ainsi le résultat obtenu sera de la forme

LIAISON N°:.....

A PARTIR DE:..... DEPART A:.....h..
JUSQUE:..... ARRIVEE A:.....h..

LIAISON N°:.....

A PARTIR DE:..... DEPART A:.....h..
JUSQUE..... ARRIVEE A:.....h..

.....

4.4.5. Sous problème créés

- Identifier un nom de point d'arrêt.

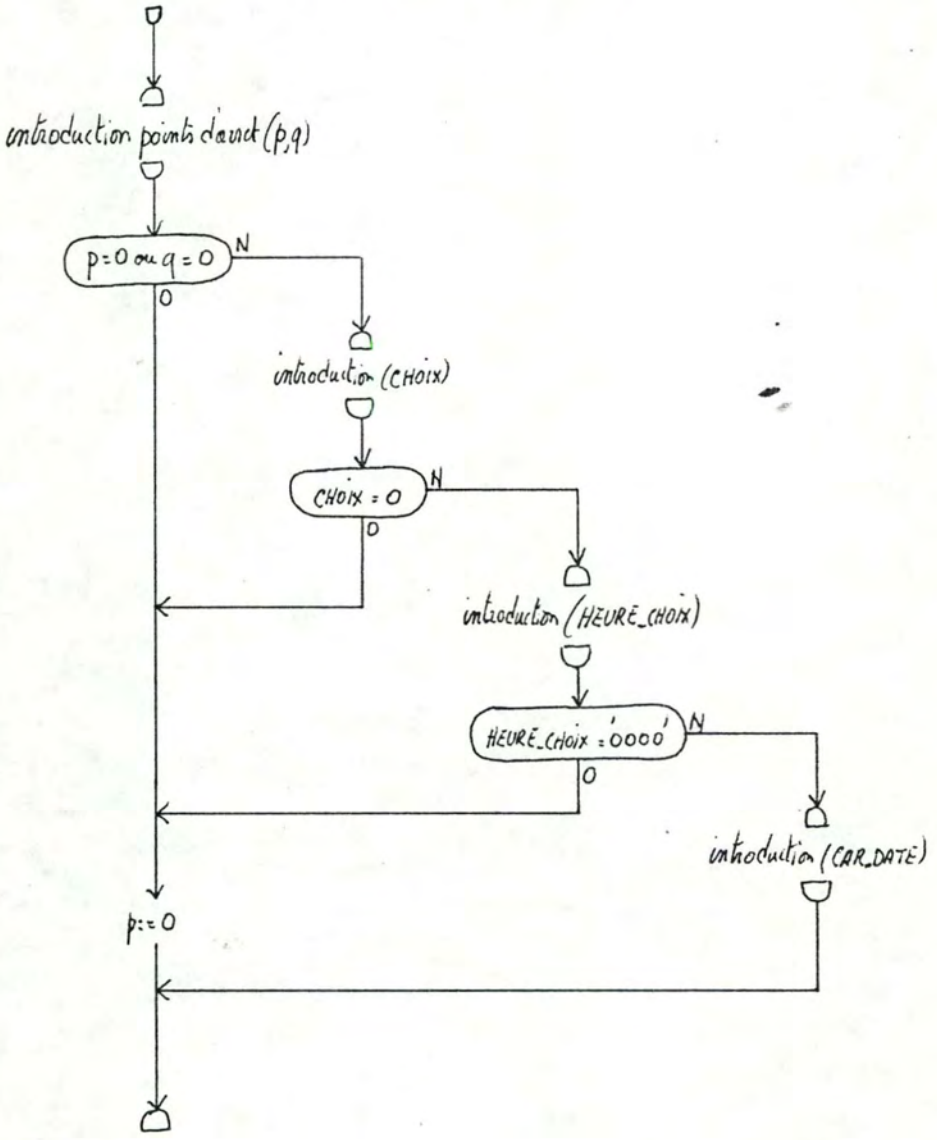
Pour identifier un nom de point d'arrêt, il faut passer en revue la liste des points d'arrêt, jusqu'a trouver un point d'arrêt correspondant. C'est-à-dire si on suppose qu'on dispose d'un fichier POINT_ARRET dont les articles sont classés par ordre alphabétique, on effectue sur ce fichier une recherche dichotomique sur base de l'item NOM de POINT_ARRET. Une recherche dichotomique signifie qu'on compare le nom recherché avec la valeur du nom se situant au milieu de l'ensemble sur lequel on fait la recherche. Si le nom recherché est égal à cette valeur, tout est parfait et la recherche est terminée; si le nom recherché est plus petit, on recommence avec la 1ère moitié de l'ensemble et si le nom recherché est plus grand, on recommence avec la seconde moitié de l'ensemble.
On recommence jusqu'au moment où soit, on a trouvé le nom, soit il n'y a plus moyen de couper en deux l'ensemble de recherche.
Si la recherche aboutit, on retrouve le numero d'ordre de l'article correspondant, sinon on retourne la valeur Ø.

- Fournir une liste de points d'arrêt se situant dans une localité.

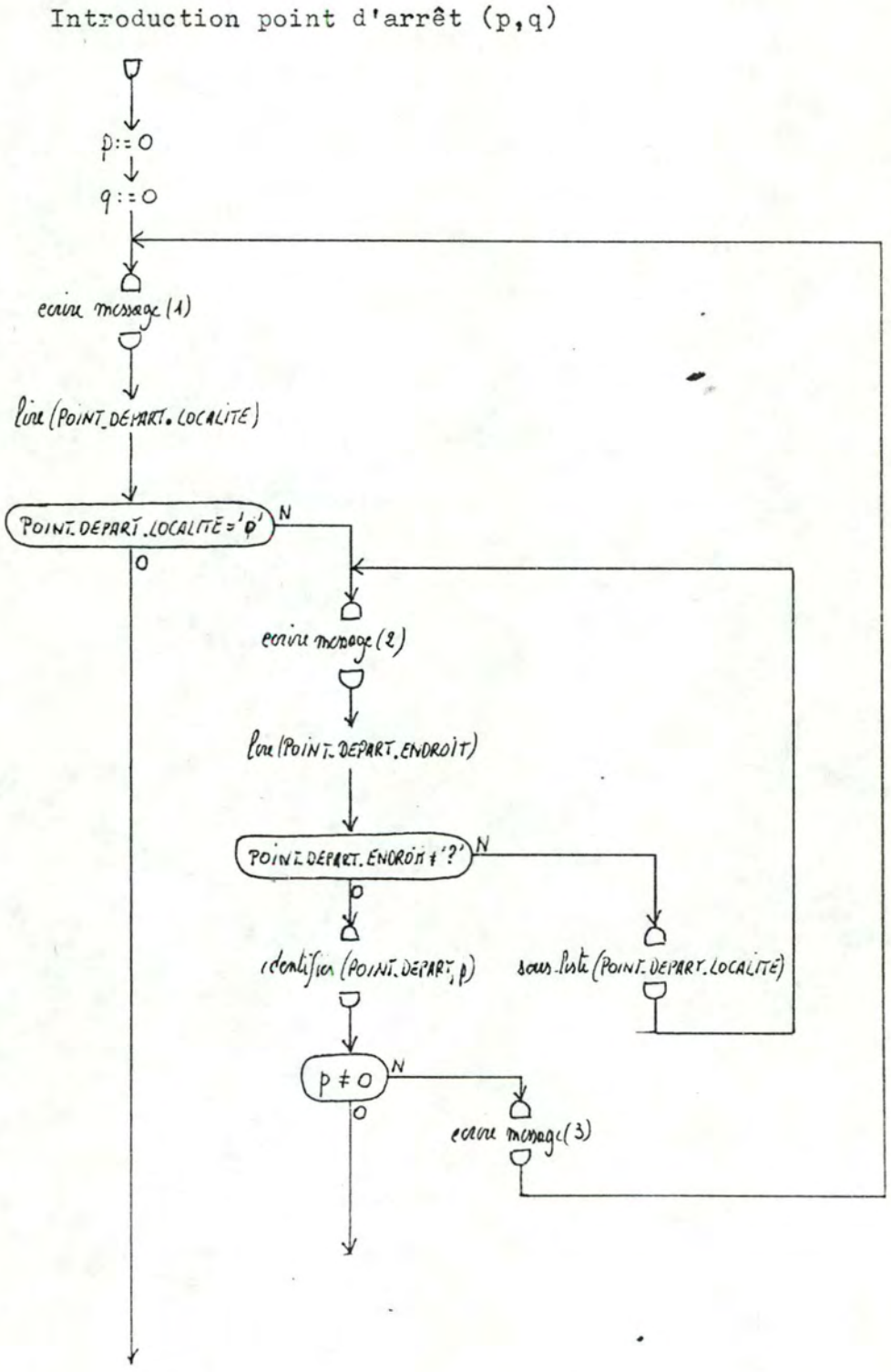
  Il faut d'abord faire une recherche dichotomique dans le fichier POINT_ARRET sur base d'un nom de localité. Lorsque la recherche a abouti, on remonte dans le fichier jusqu'a trouver un article dont l'item LOCALITE est différent du nom de localité sur base duquel la recherche a été faite. Ensuite on parcourt le fichier jusqu'au moment un l'item LOCALITE redevient différent du nom de localité.

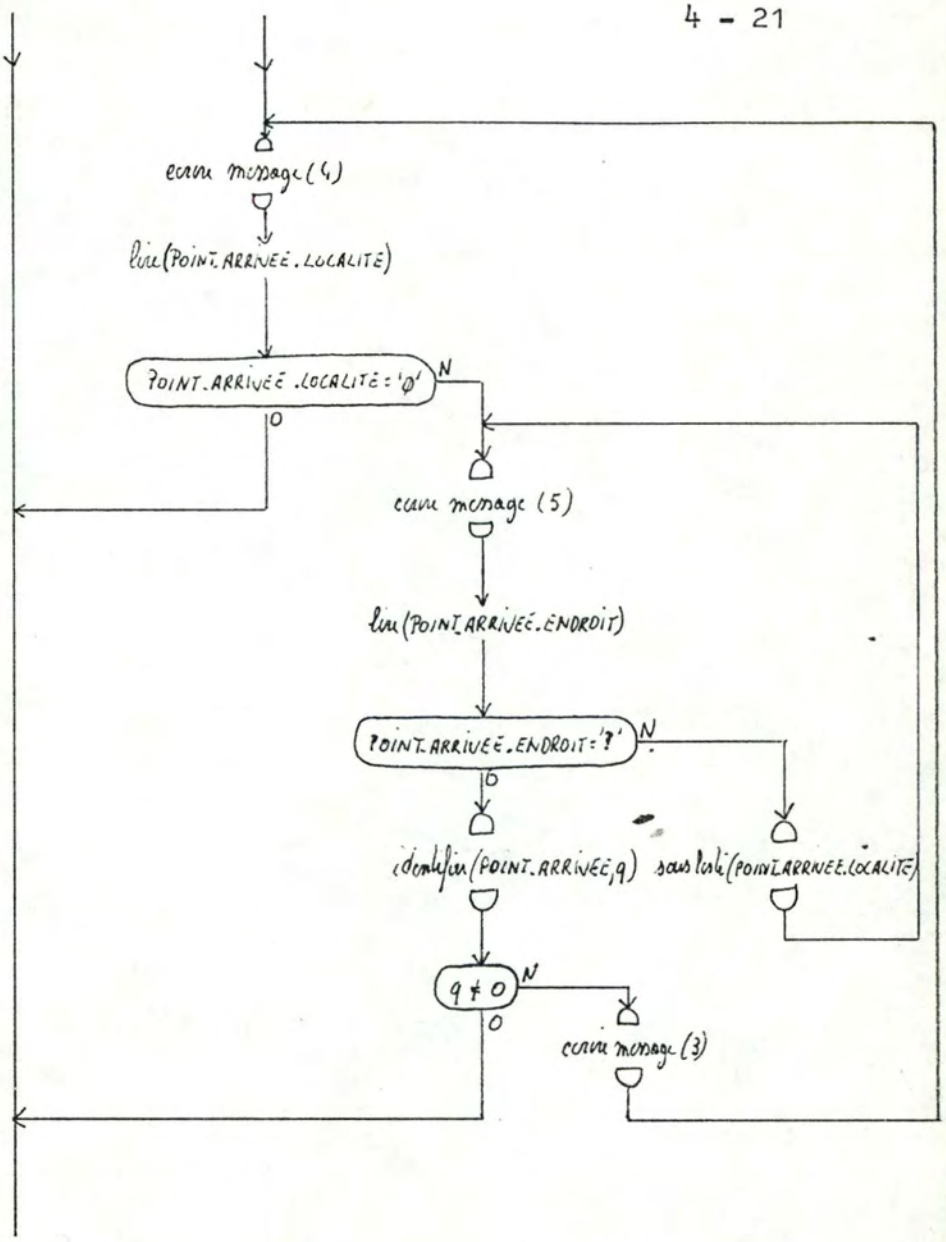
4.4.6 Algorithme de la phase 1

Phase 1 (p,q,CHOIX,HEURE_CHOIX,CARDATE)

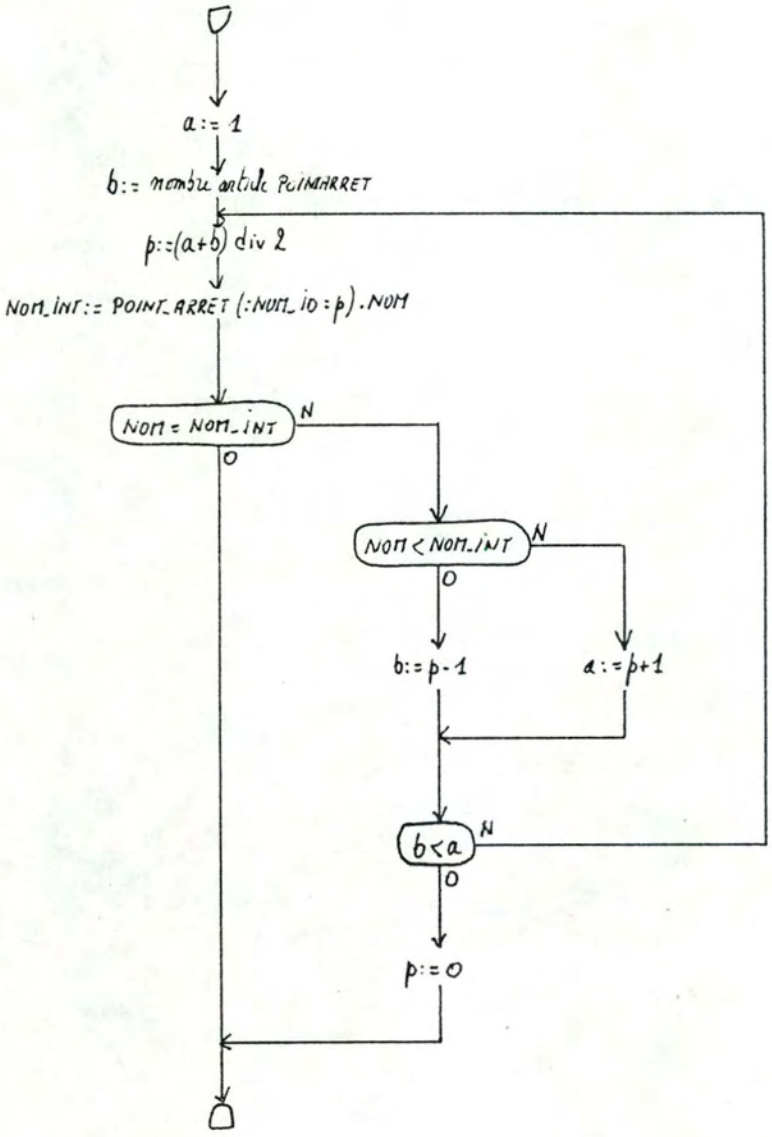


4.4.7. Algorithme des sous problèmes inhérents à la phase 1

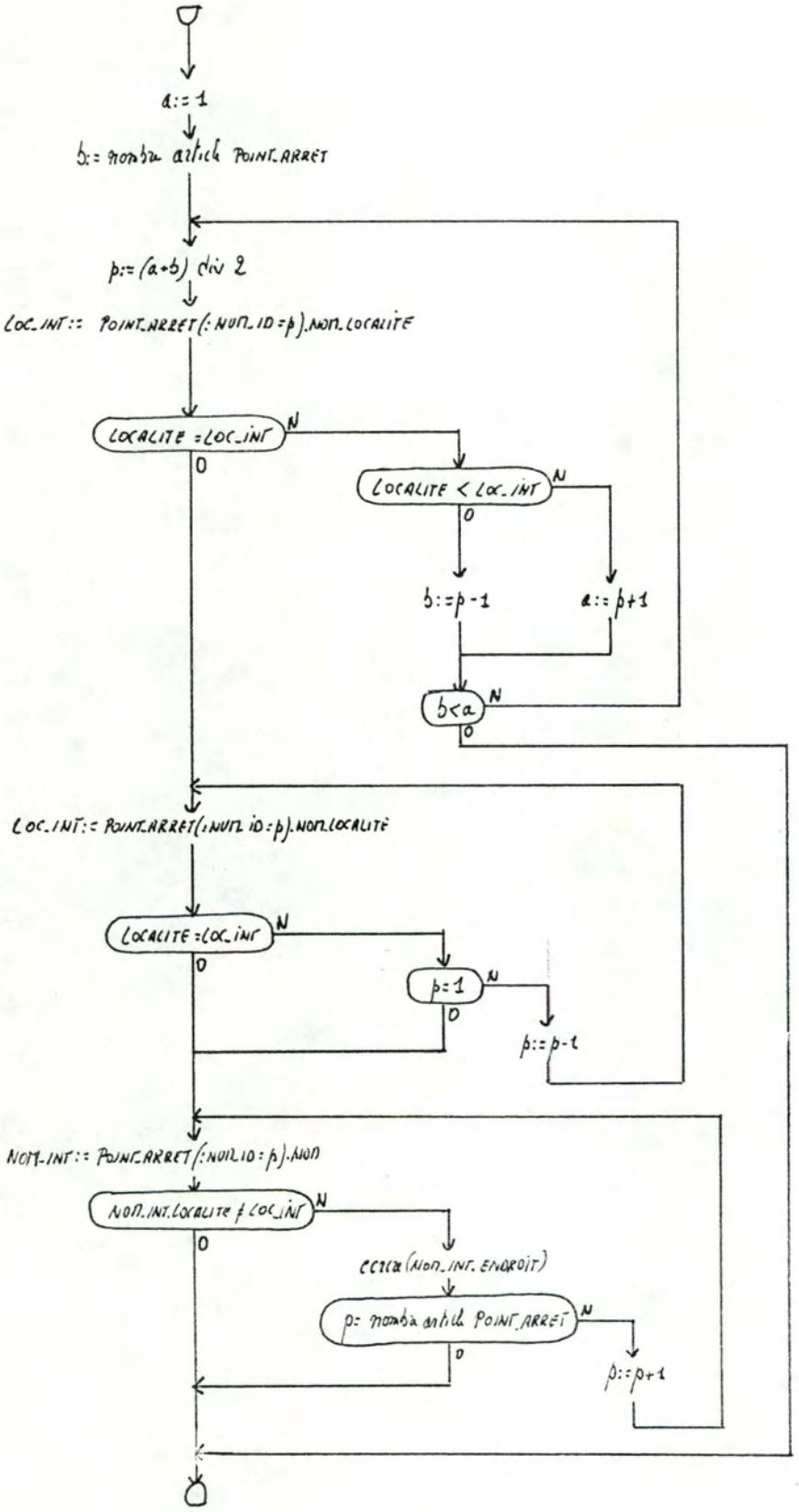



ecrire message (4)
lire (POINT_ARRIVEE.LOCALITE)
POINT_ARRIVEE.LOCALITE = 'Ø'
N
O
ecrire message (5)
lire (POINT_ARRIVEE.ENDROIT)
POINT_ARRIVEE.ENDROIT = '?'
N
O
donlefin (POINT_ARRIVEE, q)
sous local (POINT_ARRIVEE.LOCALITE)
q ≠ 0
N
O
ecrire message (3)

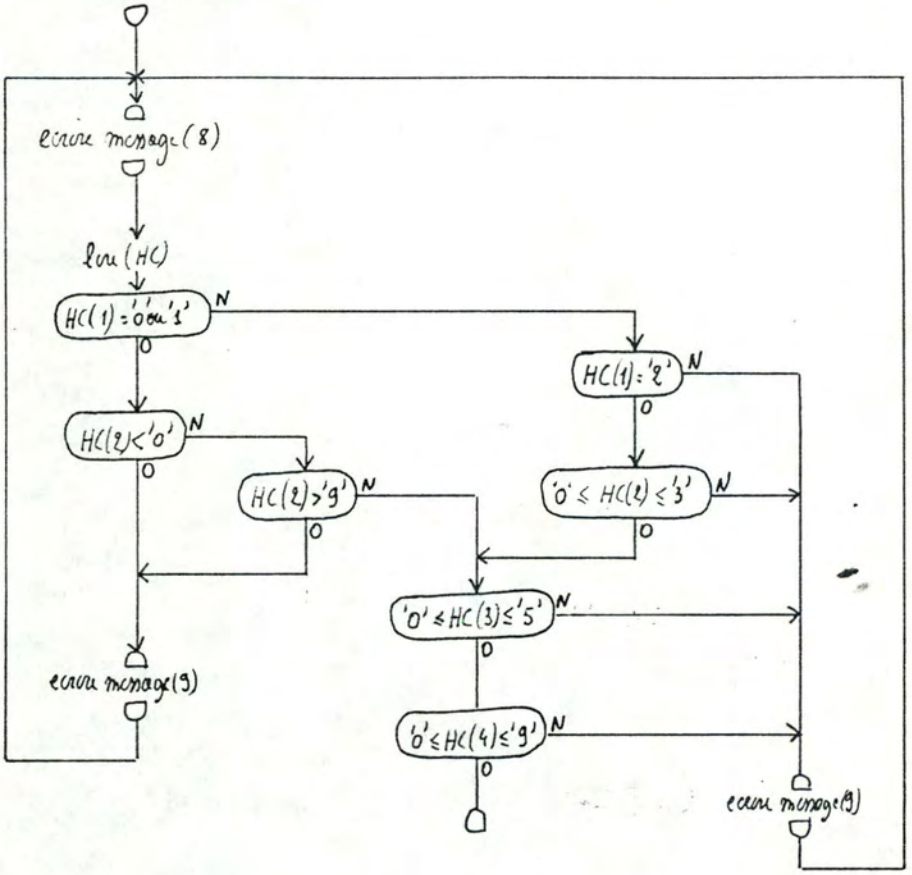
- Identifier (NOM,p)



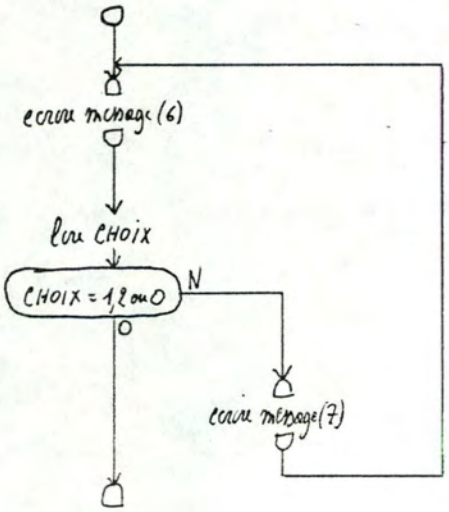
- Sous liste (LOCALITE)



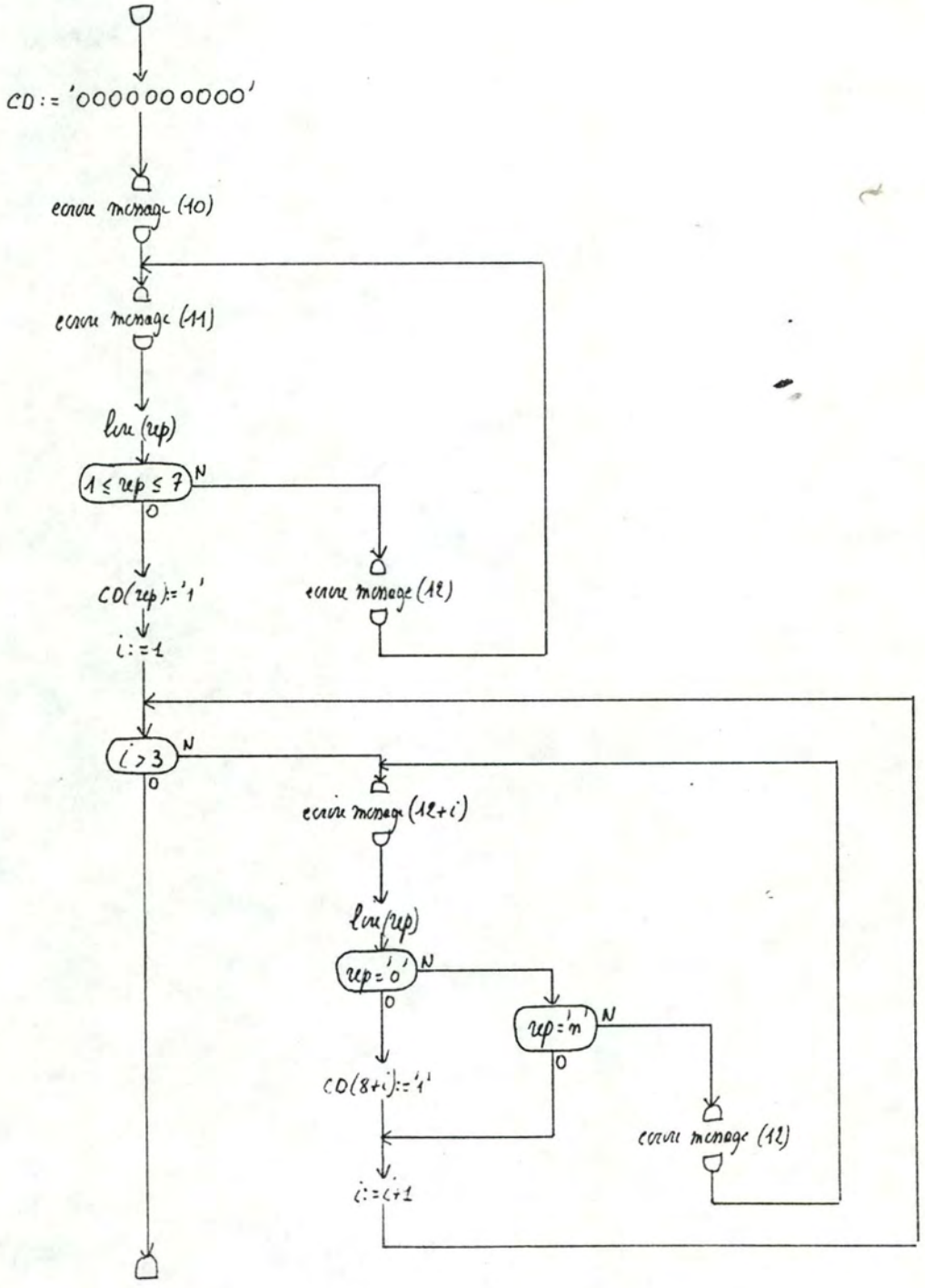
- Introduction (H C)



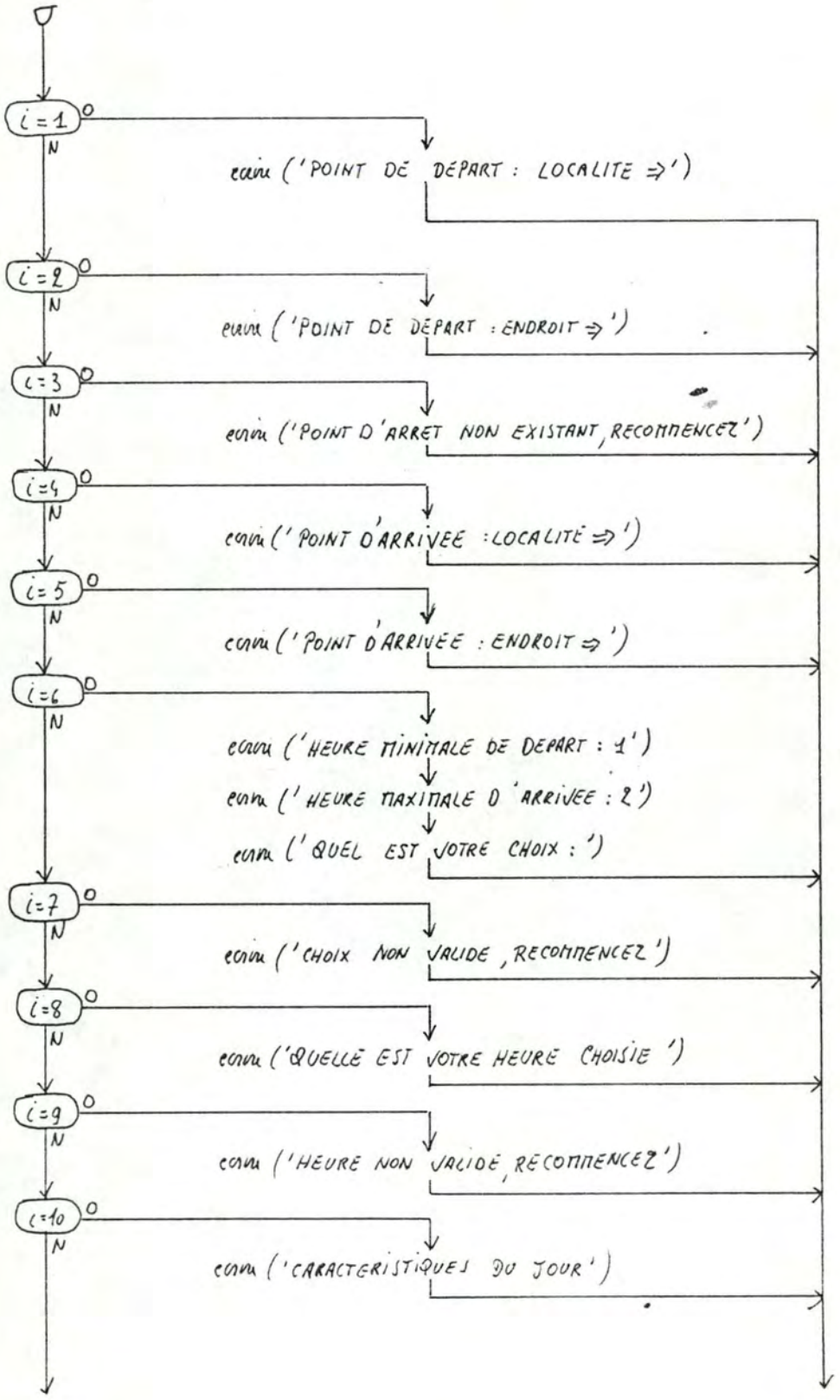
- Introduction (CHOIX)

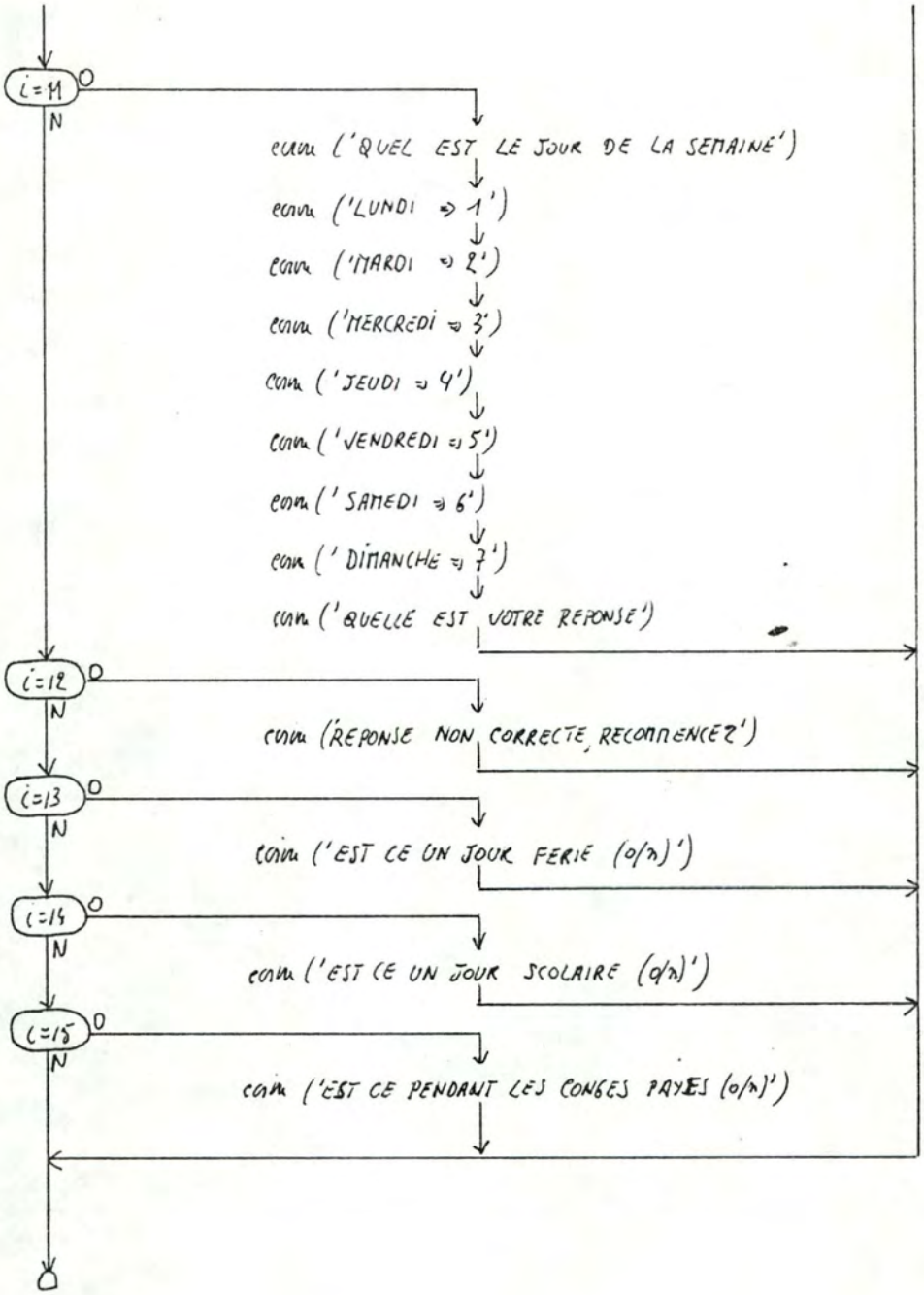


- Introduction (CD)



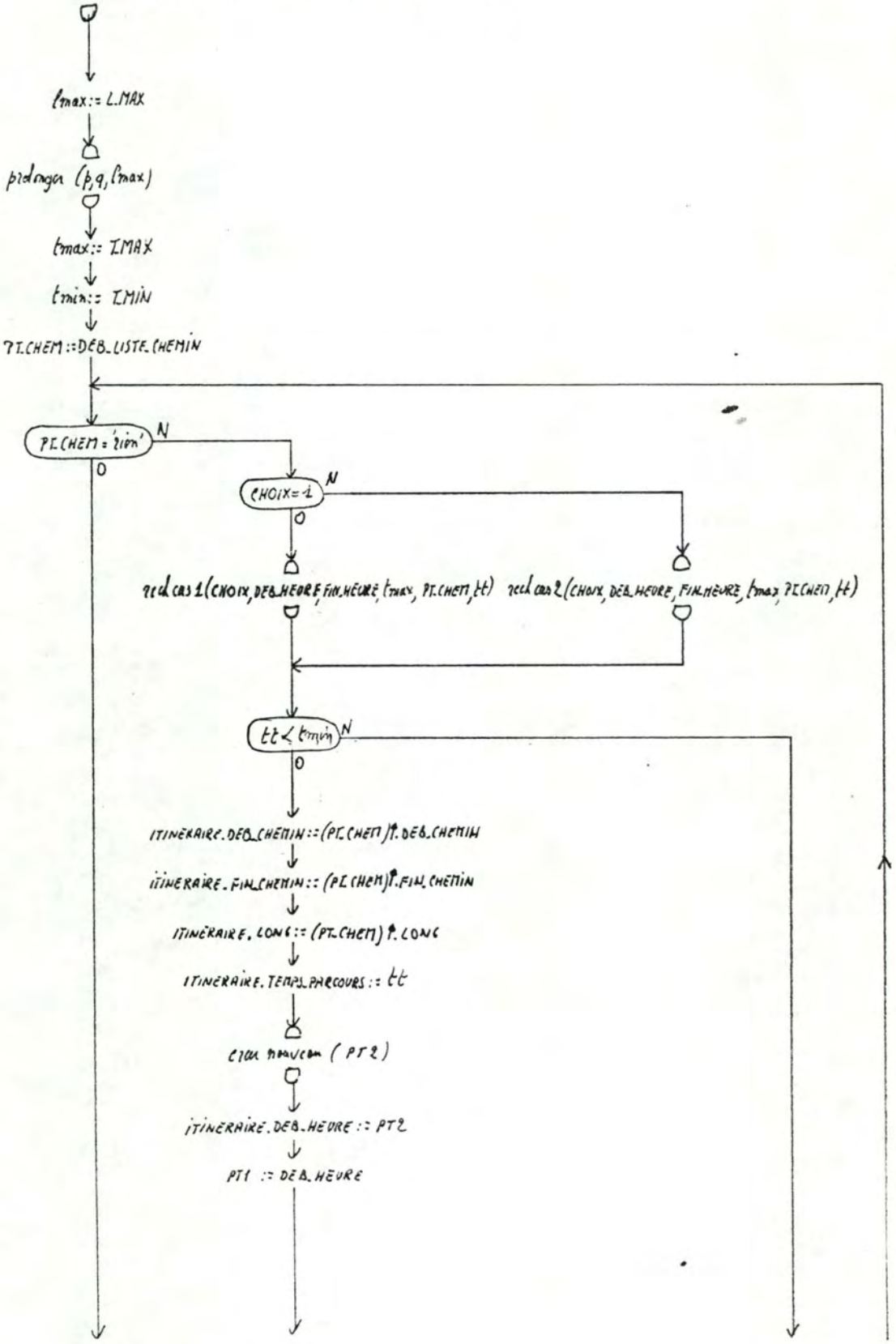
- Ecrire message (i)

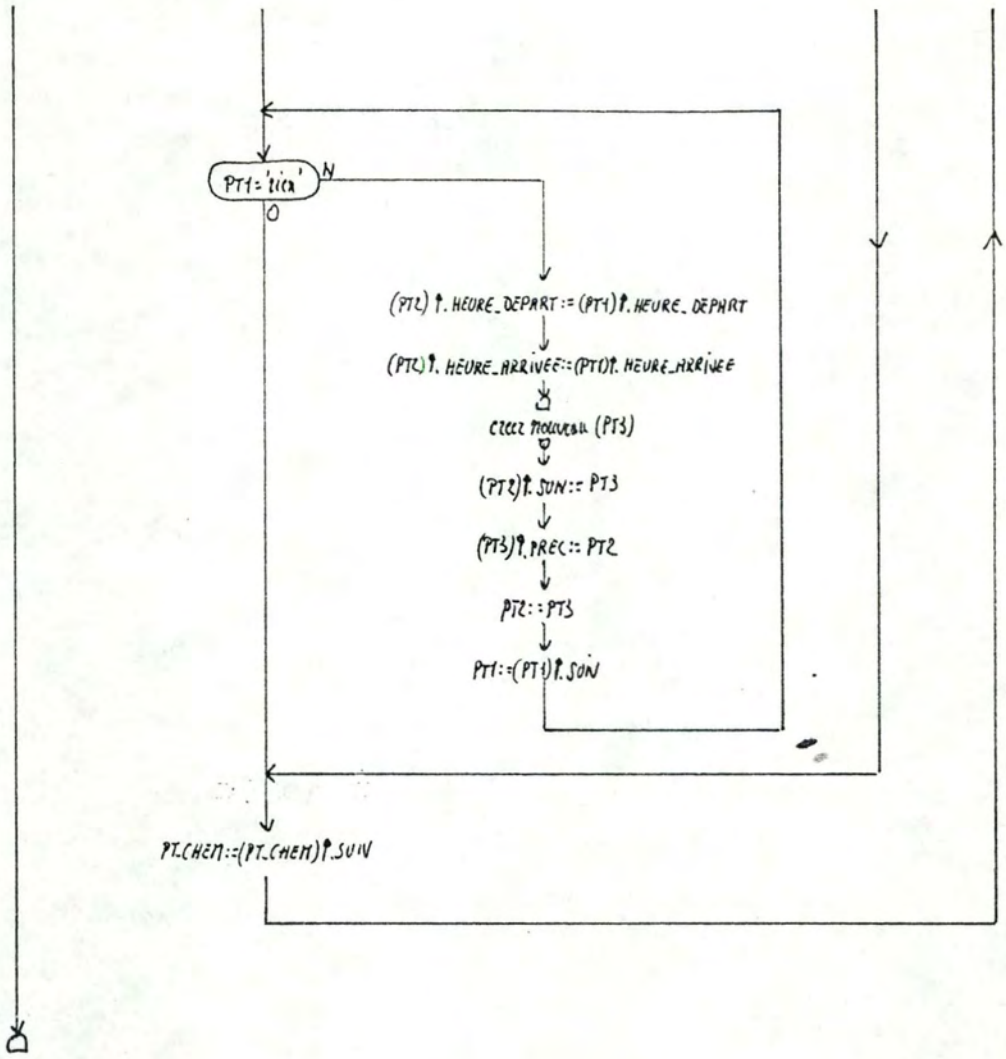



i=11
O
N
conv ('QUEL EST LE JOUR DE LA SEMAINE')
conv ('LUNDI => 1')
conv ('MARDI => 2')
conv ('MERCREDI => 3')
conv ('JEUDI => 4')
conv ('VENDREDI => 5')
conv ('SAMEDI => 6')
conv ('DIMANCHE => 7')
conv ('QUELLE EST VOTRE REPONSE')
i=12
O
N
conv (REPONSE NON CORRECTE, RECOMMENCEZ')
i=13
O
N
conv ('EST CE UN JOUR FERIE (o/n)')
i=14
O
N
conv ('EST CE UN JOUR SCOLAIRE (o/n)')
i=15
O
N
conv ('EST CE PENDANT LES CONGES PAYES (o/n)')

4.4.8. Algorithme phase 2

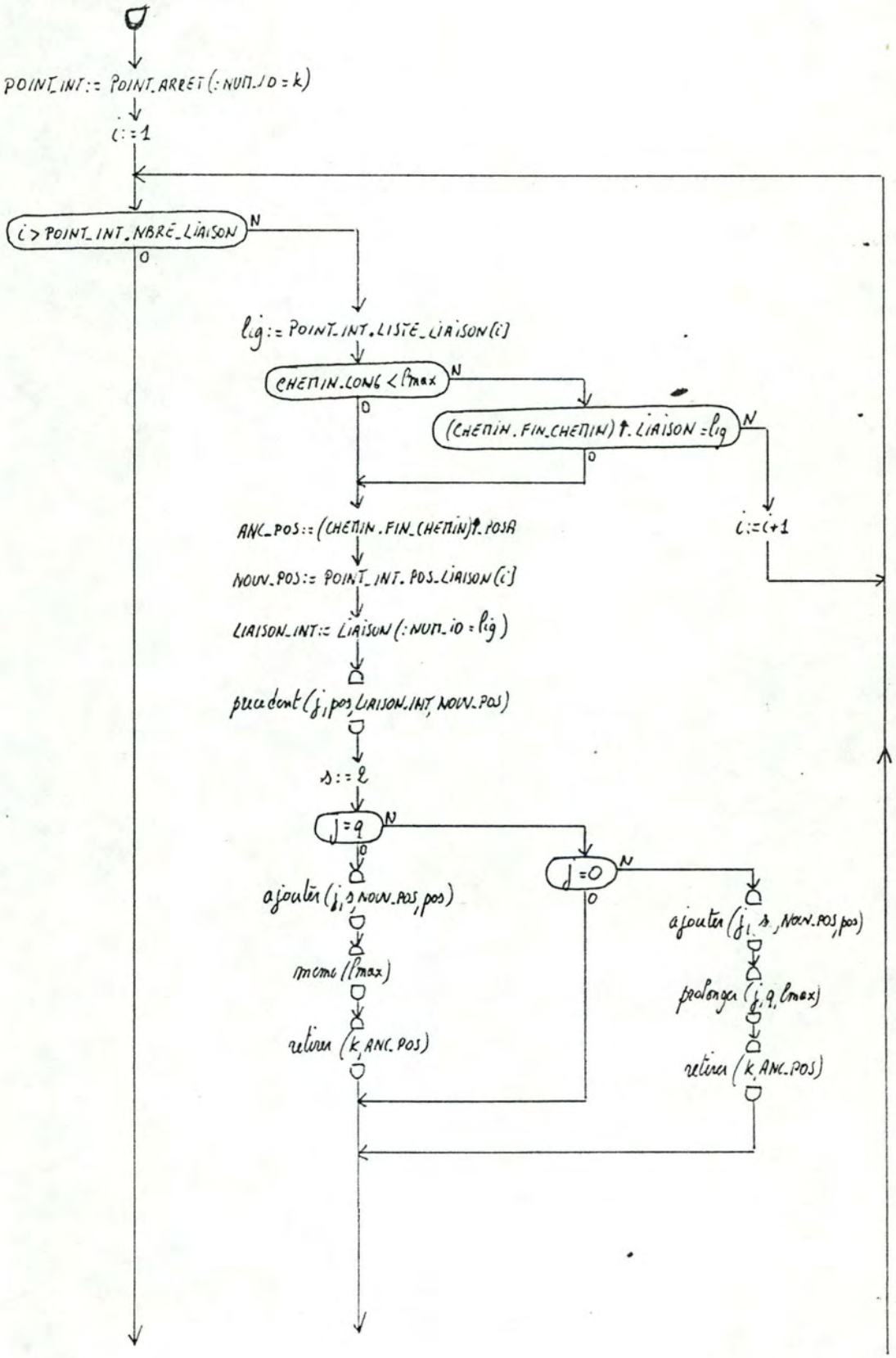
- Recherche chemin (ITINERAIRE p,q)

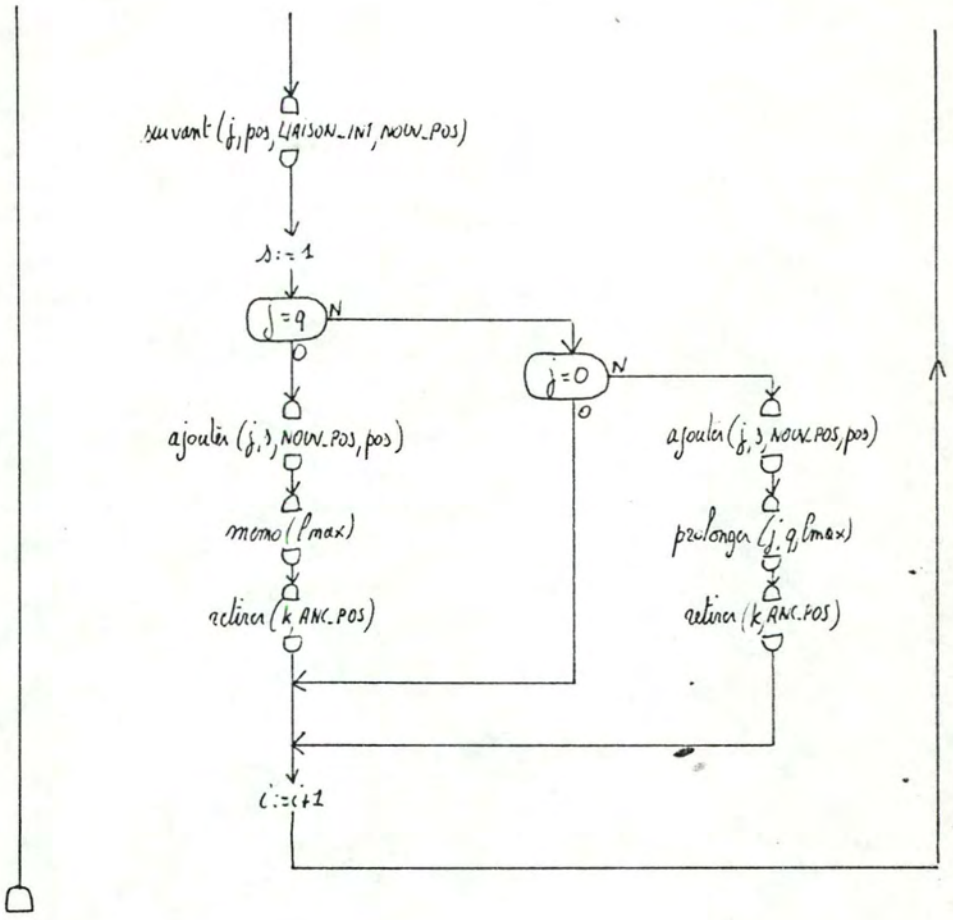



PT1 = 'lien'
N
O
(PT2)↑.HEURE_DEPART := (PT1)↑.HEURE_DEPART
(PT2)↑.HEURE_ARRIVEE := (PT1)↑.HEURE_ARRIVEE
CREER NOUVEAU (PT3)
(PT2)↑.SUIV := PT3
(PT3)↑.PREC := PT2
PT2 := PT3
PT1 := (PT1)↑.SUIV
PT_CHEM := (PT_CHEM)↑.SUIV

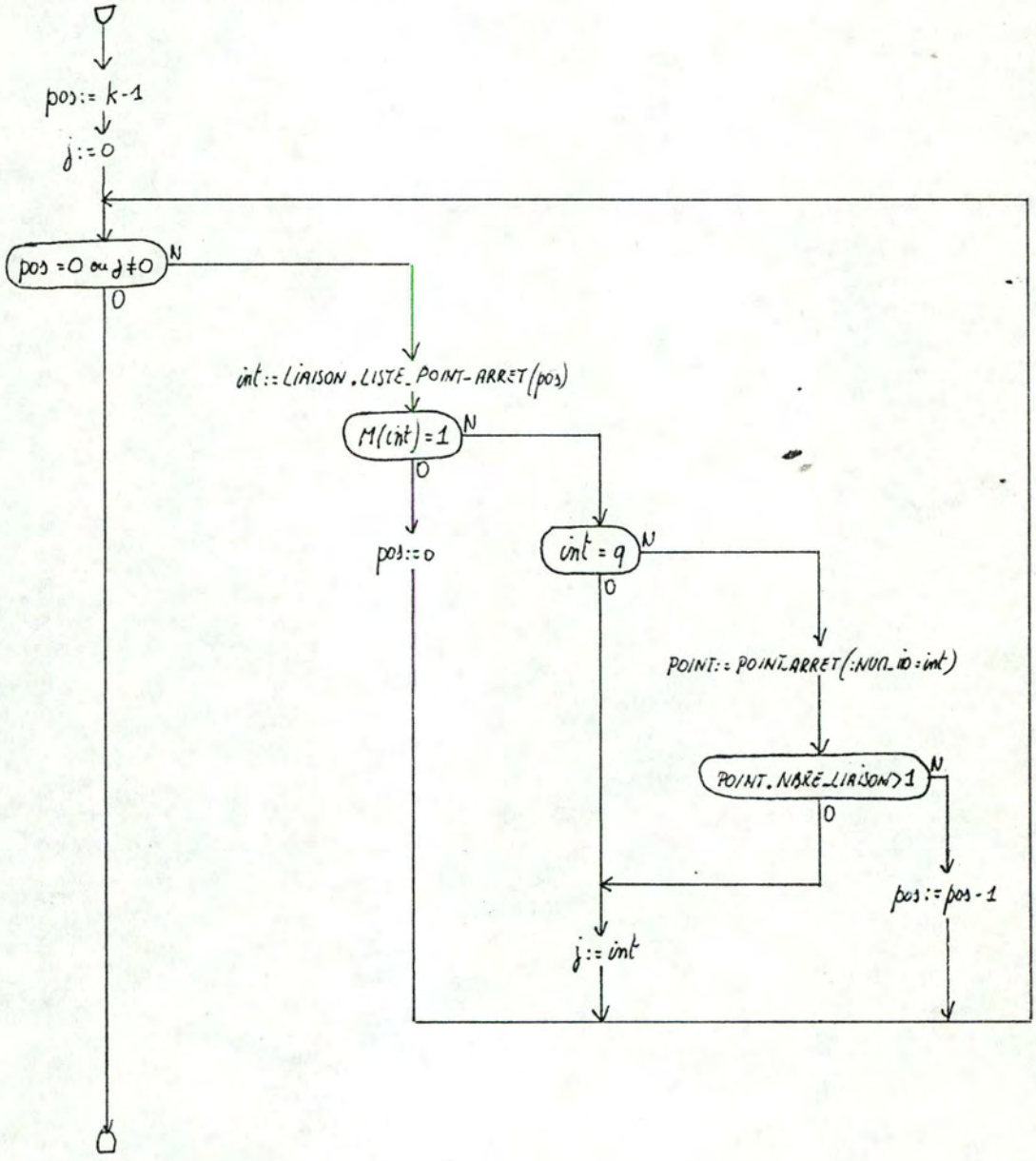
4.4.9. Algorithme des sous problèmes inhérents à la phase 2

- Prolonger (k,q,lmax)

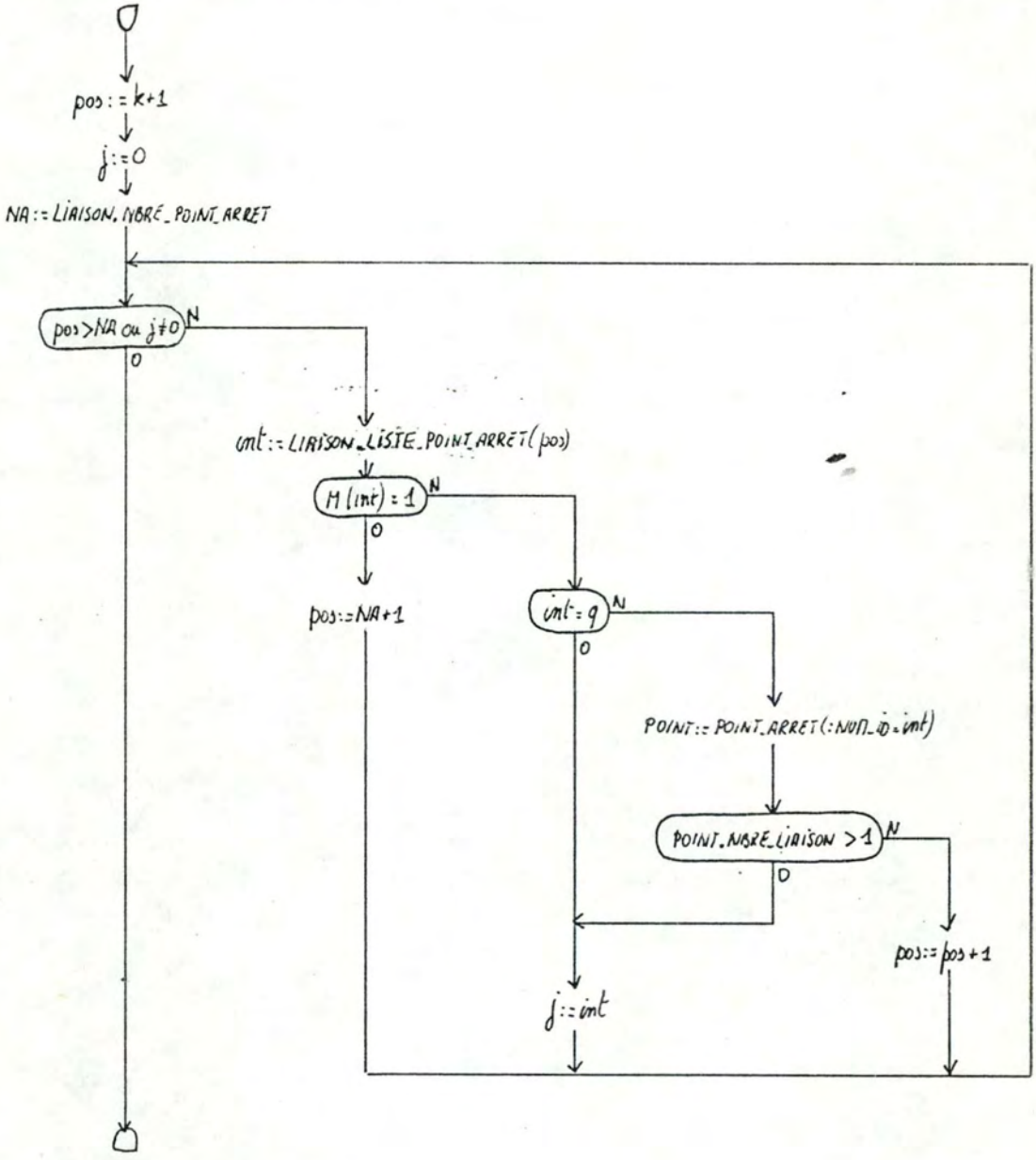



suivant (j, pos, LIAISON_INT, NOUV_POS)
i := 1
j = q
N
O
j = 0
N
O
ajouter (j, i, NOUV_POS, pos)
mémo (lmax)
retirer (k, ANC_POS)
ajouter (j, i, NOUV_POS, pos)
prolonger (j, q, lmax)
retirer (k, ANC_POS)
i := i+1

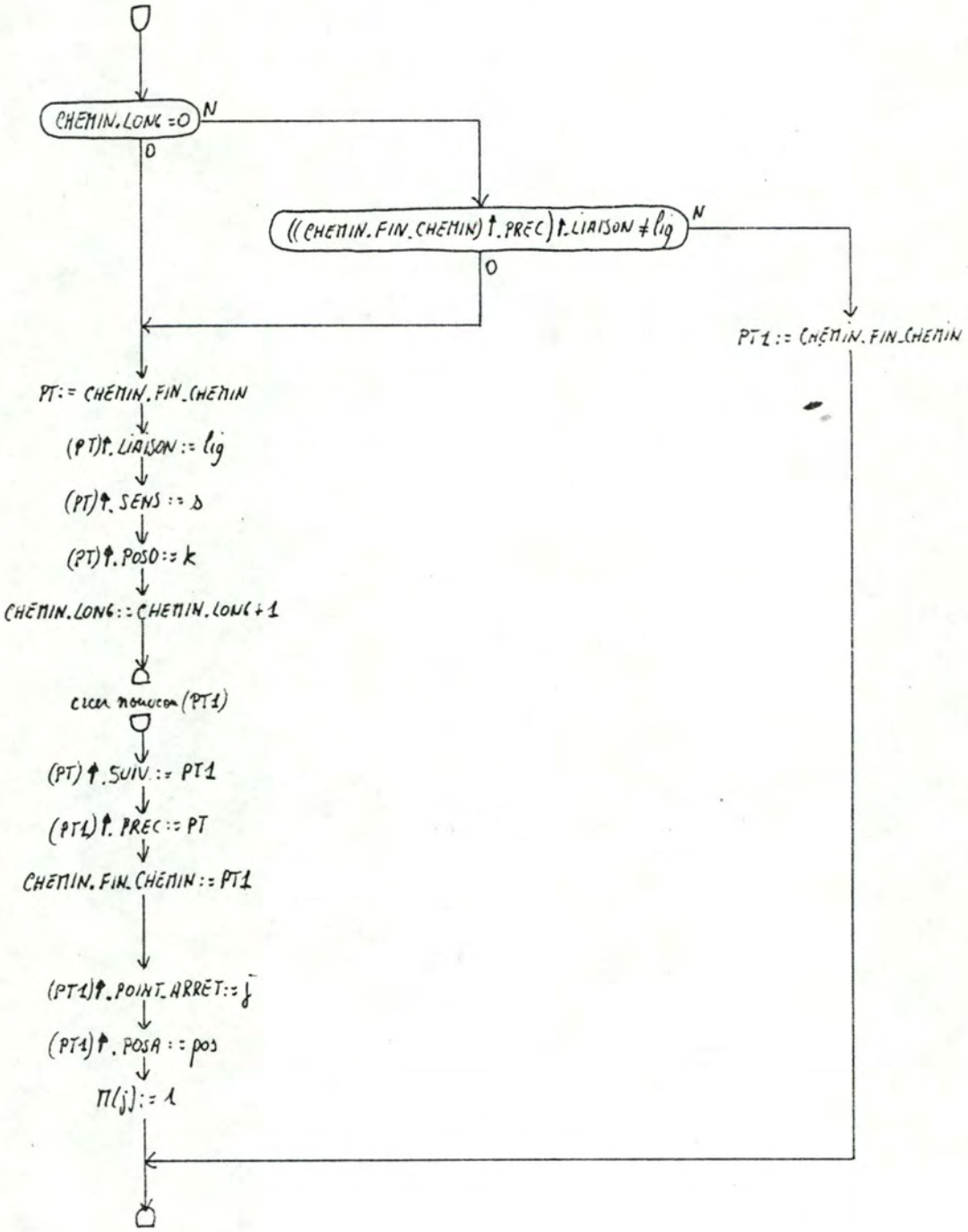
- Précédent (j,pos,LIAISON,k)



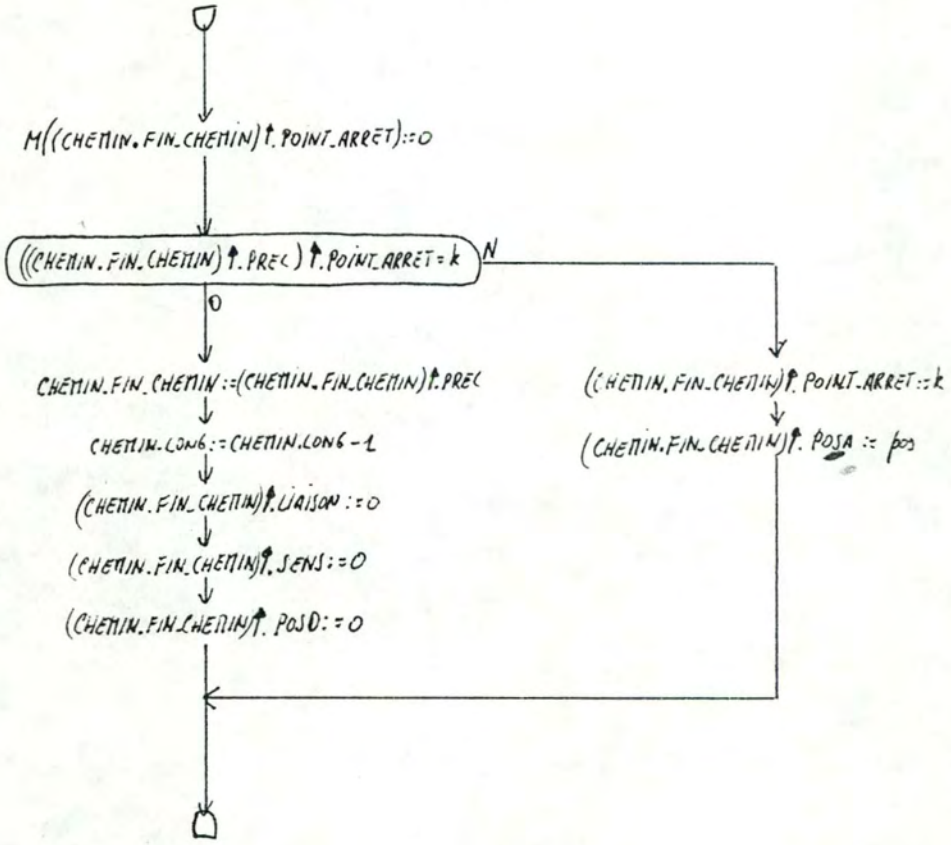
- Suivant (j,pos,LIAISON,k)



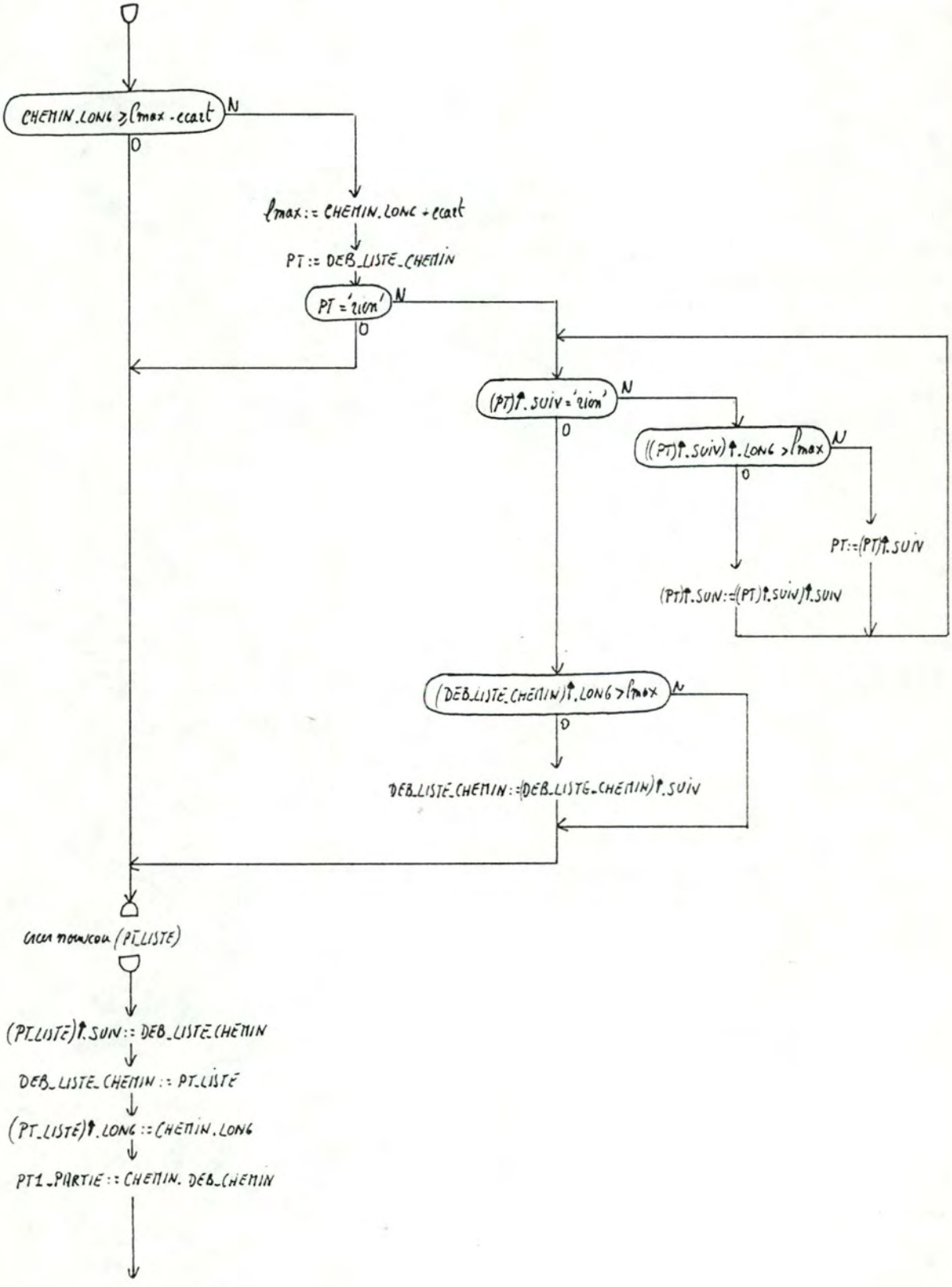
- Ajouter (j, s,k,pos)

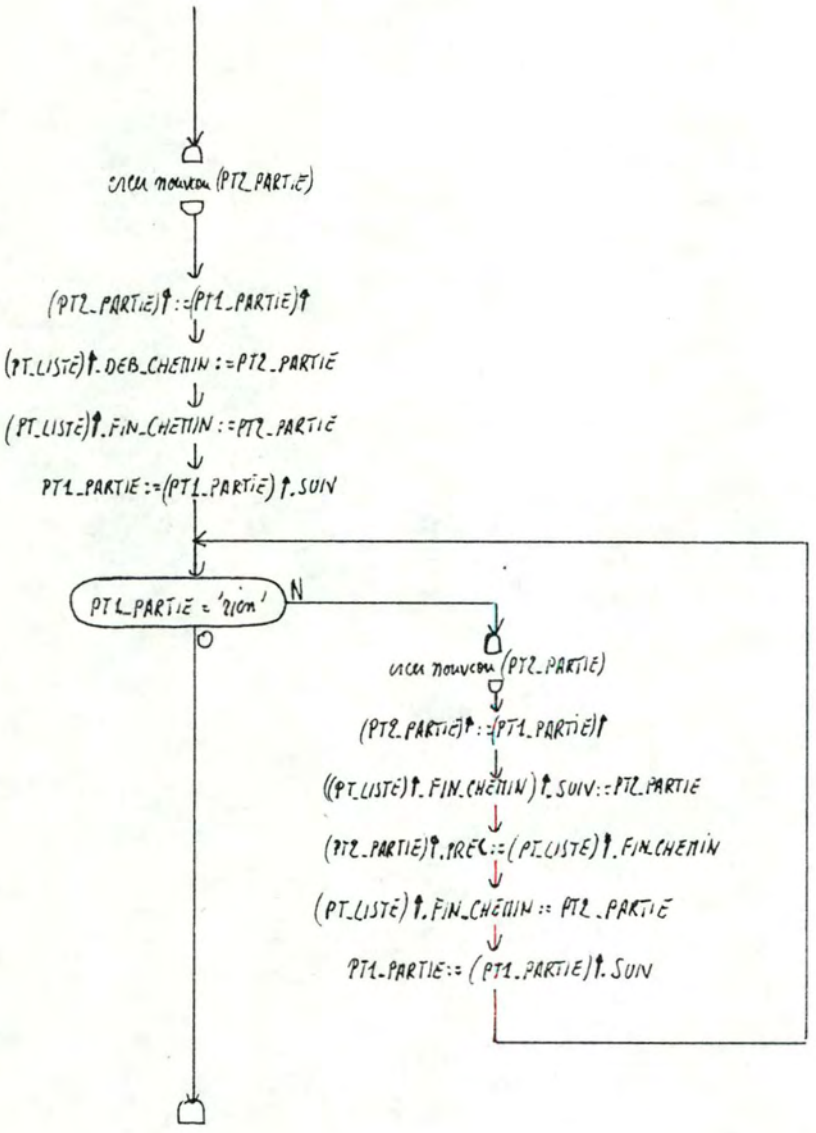


- Retirer (k, pos)

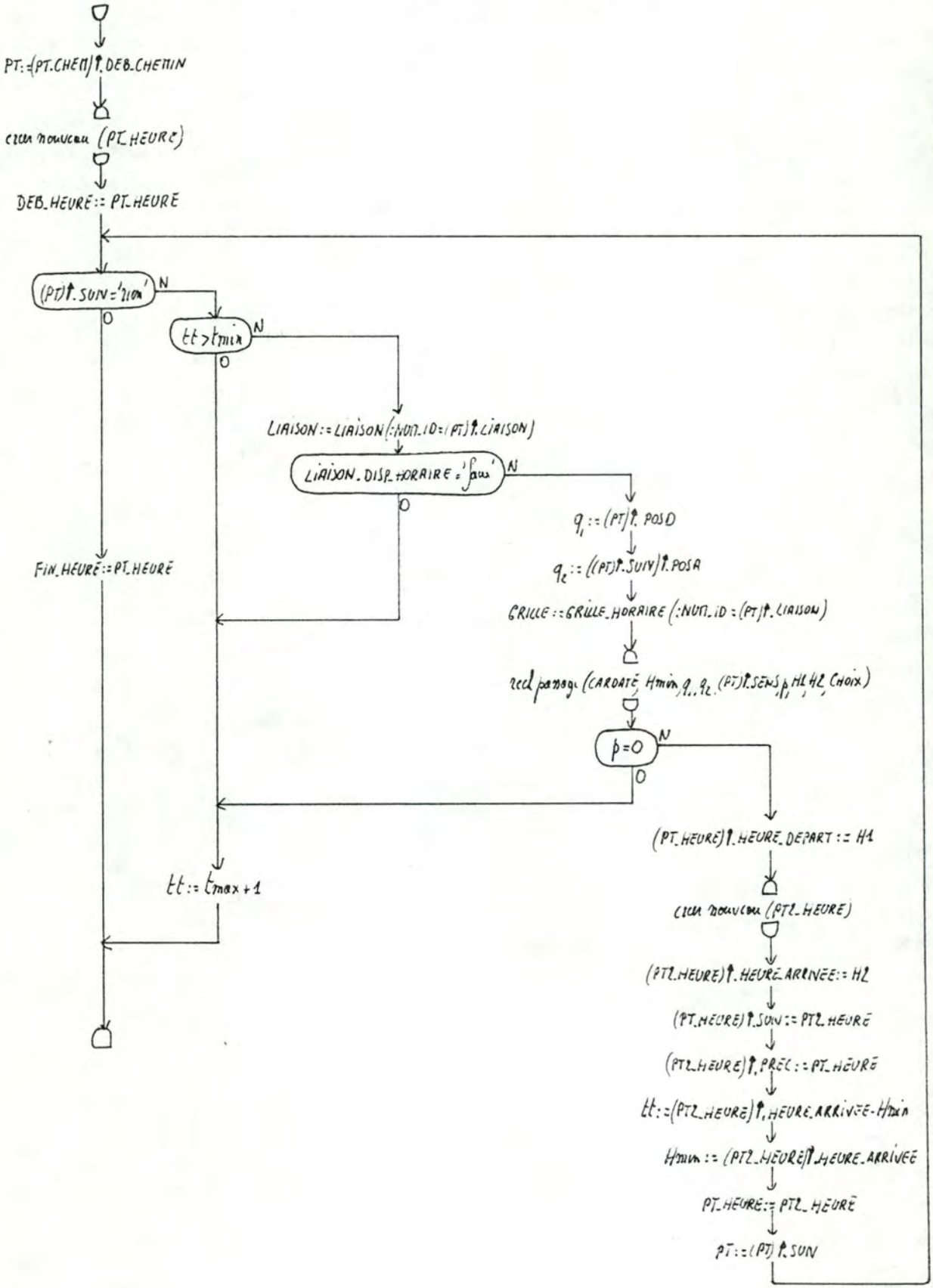


- Mémo (lmax)

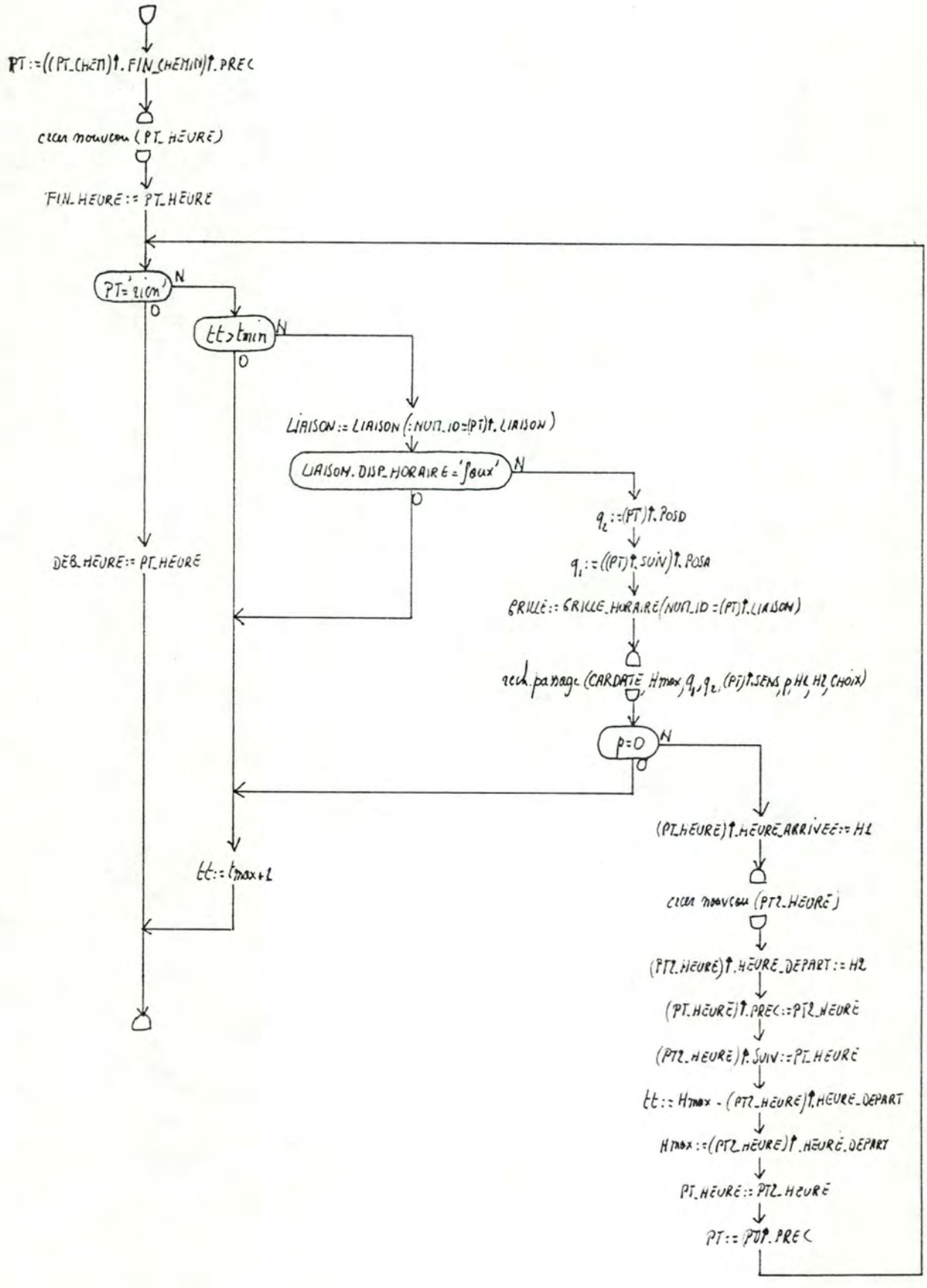

CHEMIN.LONG ≥ lmax - ecart
N
O
lmax := CHEMIN.LONG + ecart
PT := DEB_LISTE_CHEMIN
PT = 'rien'
N
O
(PT)↑.SUIV = 'rien'
N
O
((PT)↑.SUIV)↑.LONG > lmax
N
O
PT := (PT)↑.SUIV
(PT)↑.SUIV := ((PT)↑.SUIV)↑.SUIV
(DEB_LISTE_CHEMIN)↑.LONG > lmax
N
O
DEB_LISTE_CHEMIN := (DEB_LISTE_CHEMIN)↑.SUIV
Créer nouveau (PT_LISTE)
(PT_LISTE)↑.SUIV := DEB_LISTE_CHEMIN
DEB_LISTE_CHEMIN := PT_LISTE
(PT_LISTE)↑.LONG := CHEMIN.LONG
PT1_PARTIE := CHEMIN.DEB_CHEMIN


ALLOUER NOUVEAU (PT2_PARTIE)
(PT2_PARTIE)↑ := (PT1_PARTIE)↑
(PT_LISTE)↑.DEB_CHEMIN := PT2_PARTIE
(PT_LISTE)↑.FIN_CHEMIN := PT2_PARTIE
PT1_PARTIE := (PT1_PARTIE)↑.SUIV
PT1_PARTIE = 'NIL'
N
O
ALLOUER NOUVEAU (PT2_PARTIE)
(PT2_PARTIE)↑ := (PT1_PARTIE)↑
((PT_LISTE)↑.FIN_CHEMIN)↑.SUIV := PT2_PARTIE
(PT2_PARTIE)↑.PREC := (PT_LISTE)↑.FIN_CHEMIN
(PT_LISTE)↑.FIN_CHEMIN := PT2_PARTIE
PT1_PARTIE := (PT1_PARTIE)↑.SUIV

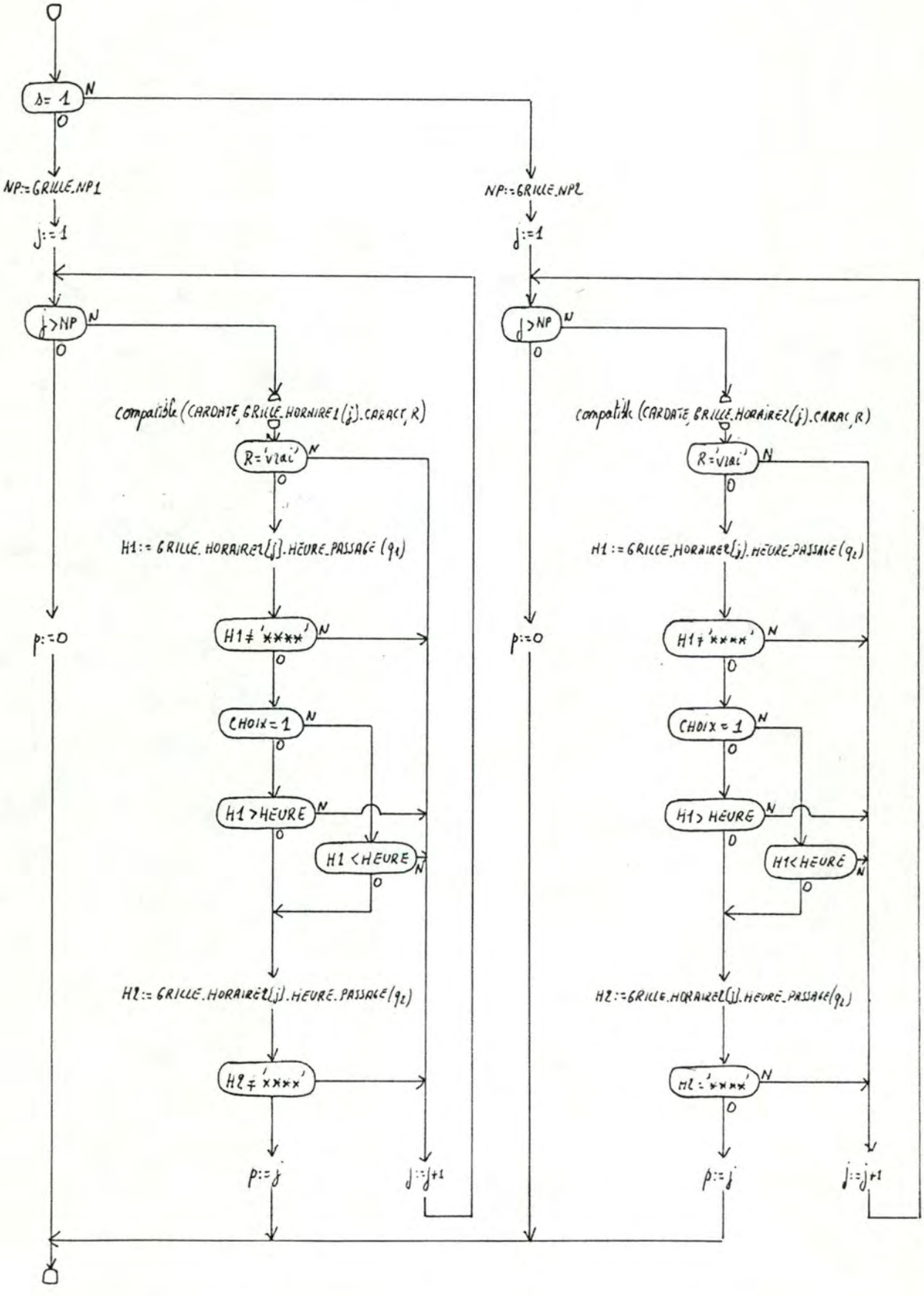
- Rech, cas 1 (CHOIX,DEB_HEURE,FIN_HEURE,tmax,PT CHEM,tt)



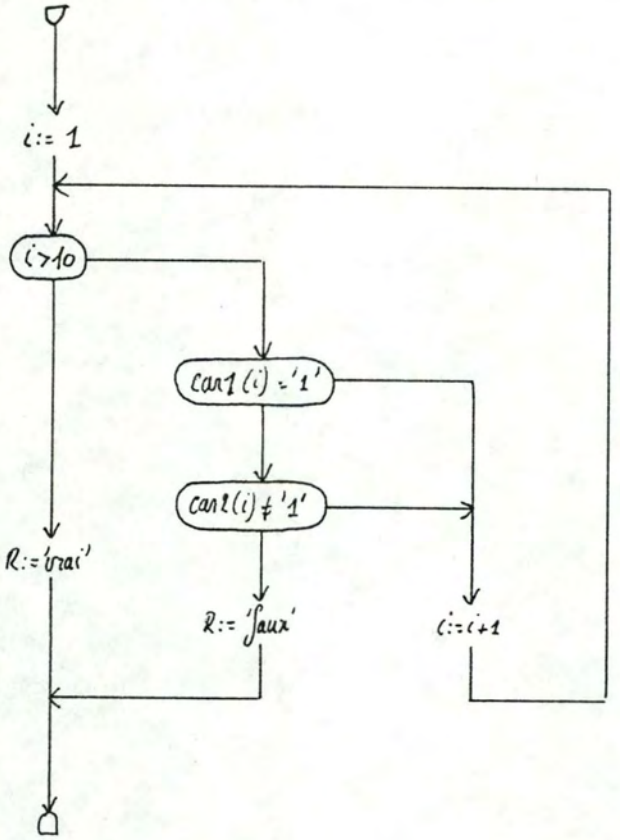
- Rech. cas 2 (CHOIX,DEB_HEURE,FIN_HEURE,tmax,PT CHEM,tt)



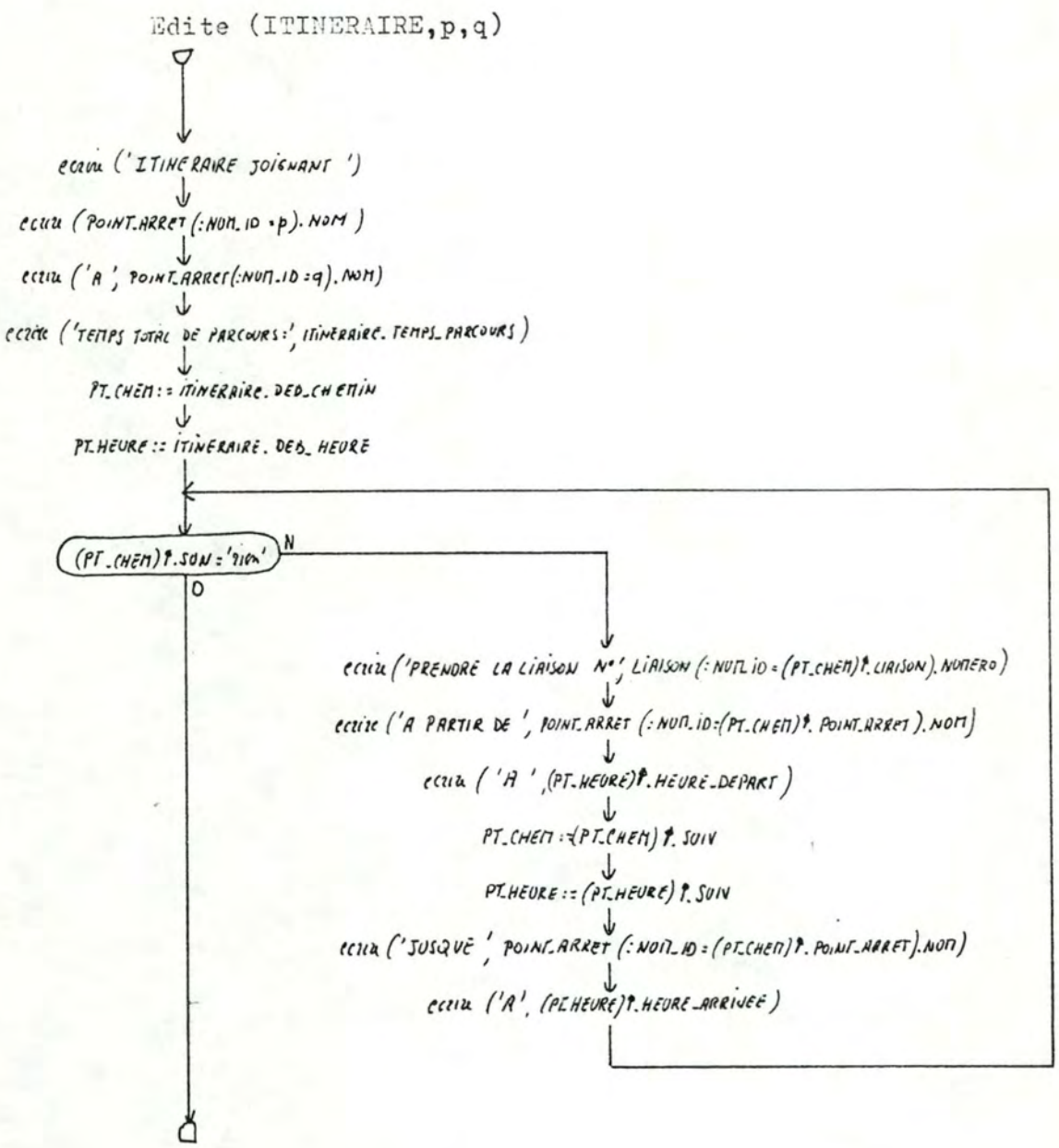
- Rech. passage (CAR_DATE,HEURE,q_1,q_2,s,ø,H1,H2,CHOIX)



- Compatible (cas 1, cas 2,R)



4.4.10. Algorithme phase 3



CHAPITRE 5

MODULE DE GESTION DES DONNEES

Introduction

La recherche d'un chemin reliant deux points d'arrêt nécessite l'introduction préalable d'un certain nombre d'informations concernant les liaisons, c'est-à-dire, les points d'arrêt qui constituent la liaison et les grilles horaires associées à cette liaison; toutes ces informations étant celles qu'on peut trouver dans les indicateurs officiels de la SNCB:SNCV.

Ces informations sont assez nombreuses et peuvent parfois être modifiées; c'est pourquoi il est intéressant et même nécessaire d'avoir un outil permettant de gérer (introduire, modifier, supprimer) toutes ces informations. Cet outil, qu'on appellera le module de gestion des données, est présenté dans ce chapitre.

Pour permettre la gestion des données, un certain nombre de facilités sont offertes, présentées sous forme de commandes. Ainsi le module de gestion est construit autour d'un gérant de commandes.

La première partie du chapitre présente une description du module de gestion des données, ainsi que des différentes facilités offertes.

La seconde partie présente la description de la méthode de réalisation de ce module de gestion, c'est-à-dire la description des fonctions utilisées pour réaliser chaque commande ainsi que la description du gérant des commandes.

MODULE DE GESTION DES DONNEES

Objectif -

Permettre la gestion pratique d'un ensemble d'informations représentant les liaisons du réseau SNCB/SNCV.

Description -

Pour permettre une gestion pratique, le module doit offrir à l'utilisateur, un certain nombre de facilités qui sont les suivantes:

1 - Au niveau des liaisons, il doit être possible de créer de toute pièce une nouvelle liaison, de modifier une liaison, de supprimer une liaison et restaurer une liaison supprimée par erreur. Il doit aussi être possible de pouvoir obtenir une liste de liaisons, ou du moins une liste de certaines liaisons, ou encore de pouvoir visualiser complètement le contenu d'une liaison.

2 - Au niveau des points d'arrêt, il doit être possible de supprimer un arrêt et de le restaurer en cas d'erreur. Il doit encore être possible d'obtenir une liste des points d'arrêt ou une liste de certains arrêts, ou encore de pouvoir visualiser complètement un point d'arrêt.

3 - A niveau des horaires, il doit être possible d'intriduire un nouvel horaire associé à une ligne existante, de modifier un horaire existant ou encore de visualiser un horaire.

DESCRIPTION DES FACILITES OFFERTES

Au niveau des liaisons

Créer une nouvelle liaison

- Créer une nouvelle liaison nécessite l'introduction d'un numero de liaison et d'un ensemble de points d'arrêt constituant cette liaison

- Le numero de la nouvelle liaison étant identifiant de cette liaison, il ne peut en aucun cas exister une autre liaison possédant déjà ce numero. Dans le cas ou une autre liaison possède déjà ce numero, c'est celle-ci qui est prise en compte et la nouvelle liaison est refusée.

Modifier une liaison existante

- Modifier une liaison nécessite l'introduction d'un numero de liaison et d'un ensemble de modifications.

- Pour modifier une liaison, il faut que celle-ci existe; il faut donc que le numero de liaison introduit soit associé à une liaison.

- Les modifications apportées à la liaison peuvent être de 3 ordres:

  1) La modification d'un ou plusieurs points d'arrêt
  2) La suppression d'un ou plusieurs points d'arrêt
  3) L'insertion d'un ou plusieurs points d'arrêt

Supprimer une liaison existante

- Supprimer une liaison nécessite l'introduction d'un numero de liaison.
- Pour supprimer une liaison, il faut que celle-ci existe, il faut donc que le numero de liaison introduit soit associé à une liaison active.
- Pour permettre de rattraper une erreur, il s'agira d'une suppression logique et non d'une suppression physique.

Restaurer une liaison

- Restaurer une liaison nécessite l'introduction d'un numero de liaison.
- Pour restaurer une liaison, il faut que celle-ci ait été supprimée; il faut donc que le numero de liaison introduit soit associé à une liaison inactive.
- La restauration d'une liaison n'est possible que si il n'y a pas eu de suppression physique depuis la suppression logique (pour gagner de la place par ex.).

Liste de liaisons

- La liste peut être demandée sur base de certains critères:
  - . un préfixe du numero de liaison
  - . l'état de la liaison
- La liste peut éventuellement être vide

- Les renseignements fournis pour une liaison sont les suivants:

  . le numero de liaison
  . l'état de la liaison
  . le nombre de points d'arrêt actifs
  . les extrémités de la liaison

Visualiser une liaison

- Visualiser une liaison nécessite l'introduction d'un numero de liaison.

- Pour visualiser une liaison, il faut que celle-ci existe; il faut donc que le numero de liaison introduit soit associé à une liaison.

- Les renseignements fournis sont les suivants:

  . l'état de la liaison
  . le nombre de points d'arrêt actifs
  . l'intitulé de ces points d'arrêt.

Au niveau des points d'arrêt

Supprimer un arrêt

- Supprimer un arrêt nécessite l'introduction d'un nom d'arrêt.
- Pour supprimer un arrêt, il faut que celui-ci existe; il faut donc que le nom d'arrêt introduit soit associé à un arrêt actif.
- Pour permettre de rattraper une erreur, il s'agira d'une suppression logique et non d'une suppression physique.

Restaurer un arrêt

- Restaurer un arrêt nécessite l'introduction d'un nom d'arrêt.
- Pour restaurer un arrêt, il faut que celui-ci ait été supprimé; il faut donc que le nom d'arrêt soit associé à un arrêt inactif.
- La restauration d'un arrêt n'est possible que si il n'y a pas eu de suppression physique depuis la suppression logique (pour gagner de la place, par ex.).

Liste des points d'arrêt

- La liste peut-être demandée sur base de certains critères:
  - un préfixe du nom d'arrêt
  - l'état du point d'arrêt
- La liste peut-être éventuellement vide.
- Les renseignements fournis pour un point d'arrêt sont les suivants:
  - le nom de l'arrêt
  - l'état de l'arrêt
  - le nombre de liaisons actives passant par ce point d'arrêt.

Visualiser un point d'arrêt

- Visualiser un point d'arrêt nécessite l'introduction d'un nom d'arrêt.
- Pour visualiser un point d'arrêt, il faut que celui-ci existe; il faut donc que le nom d'arrêt introduit soit associé à un point d'arrêt.
- Les renseignements fournis sont les suivants:
  - l'état de l'arrêt
  - le nombre de liaisons actives passant par ce point d'arrêt
  - le numero de ces liaisons.

Au niveau des horaires

Introduire un horaire

- Introduire un horaire nécessite l'introduction d'un numero de liaison et d'un ensemble de temps de passage pour chaque sens de la liaison, chaque ensemble de temps de passage étant composé d'une part des restrictions sur ces temps de passage, et d'autre part des différents temps de passage à chaque point d'arrêt constituant la liaison, avec la possibilité de sauter un ou plusieurs points d'arrêt.

- Pour introduire un horaire d'une liaison, il faut que celle-ci existe; il faut donc que le numero de liaison introduit soit associé à une liaison.

- Il faut aussi qu'à l'intérieur de chaque ensemble de temps de passage, les temps de passage soient introduits en ordre croissant.

Modifier un horaire

- Modifier un horaire nécessite l'introduction d'un numero de liaison et d'un ensemble de modifications.

- Pour modifier un horaire d'une liaison, il faut que celle-ci existe; il faut donc que le numero de liaison introduit soit associé à une liaison. De plus, il faut aussi qu'un horaire de cette liaison soit déjà existant.

- Les modifications apportées à un horaire peuvent être de 3 ordres:

    1) Suppression d'un ou plusieurs ensembles de temps de passage
    2) Modification d'un ou plusieurs ensembles de temps de passage
    3) Insertion d'un ou plusieurs ensembles de temps de passage

- Les modifications apportées doivent encore garantir le caractère croissant des temps de passage.

- Les modifications d'un ensemble de temps de passage peuvent être de 4 ordres:

    1) Modification des restrictions sur les temps de passage
    2) Suppression d'un ou plusieurs temps de passage
    3) Modification d'un ou plusieurs temps de passage
    4) Insertion d'un ou plusieurs temps de passage.

Visualiser un horaire

- Visualiser un horaire nécessite l'introduction d'un numero de liaison.

- Pour visualiser un horaire d'une liaison, il faut que celle-ci existe; il faut donc que le numero de liaison introduit soit associé à une liaison. De plus, il faut aussi qu'un horaire de cette liaison soit déjà existant.

- Les renseignements fournis sont:

  . La liste des noms d'arrêts constituant la liaison.
  . L'ensemble des temps de passage pour les deux sens de la liaison (restriction et temps de passage).

REALISATION

Introduction

- Les différentes facilités sont présentées à l'utilisateur sous forme de commandes.

  Ainsi, il y a 6 commandes accompagnées de paramètres

  Créer (type, numero) avec type = L(iaison) ou H(oraire)
  Modifier (type, numero) avec type = L(iaison) ou H(oraire)
  Supprimer (type, numero ou nom) avec type = L(iaison) ou P(oint d'arrêt)
  Restaurer (type, numero ou nom) avec type = L(iaison) ou P(oint d'arrêt)
  Visualiser (type, numero ou nom) avec type = L(iaison) ou P(oint d'arrêt) ou H(oraire)
  Lister (type) avec type = L(iaison) ou P(oint d'arrêt)

  Les spécifications de ces commandes sont telles que si un des paramètres est incorrect, alors la commande est refusée. (Voir description des facilités)

  La structure du module de gestion des données est composée d'un gérant des commandes analysant la commande introduite et passant ensuite, si tout est exact, la main à la fonction spécifique réalisant la fonction.

Structures de données

- Les données manipulées étant celles qui sont utilisées lors de la recherche d'un itinéraire, les structures de données permanentes sont celles décrites au chapitre précédent, avec cependant quelques ajouts.

Ainsi on ajoute à LIAISON deux items:

1) L'état de la liaison, c'est-à-dire quelque chose qui permet de dire si cette liaison a été supprimée logiquement ou non.
   ETAT: A(1) ('A' ou 'I')

2) Quelque chose permettant d'identifier précisément un point d'arrêt si celui-ci apparait plusieurs fois dans la liaison.

   POS_POINT_ARRET : T (1..NBRE_POINT_ARRET) d'entier

   si LIAISON (:NUM_ID=p).POS_POINT_ARRET [q] = i

   et LIAISON (:NUM_ID=p).LISTE_POINT_ARRET [q] = j

   alors

   POINT ARRET (:NUM_ID=p).LISTE_LIAISON [i] = p

   et POINT_ARRET (:NUM_ID=j).POS_LIAISON [i]= q

On ajoute aussi un item à POINT_ARRET

- L'état du point d'arrêt, c'est-à-dire quelque chose permettant de dire si ce point d'arrêt a été supprimé logiquement ou non.

  ETAT : A(1) ('A' ou 'I')

Il est nécessaire aussi de définir quelques variables qui seront utilisées:

N_LIAISON: le premier numero d'identification disponible pour une liaison

N_POINT ARRET: le premier numero d'identification disponible pour un point d'arrêt.

Fonctions utilisées dépendant de l'implémentation

- Fonction LIRE (DONNEE,X,Y)

Spécification:

donner à X qui est du même type que DONNEE, la valeur de l'occurence de DONNEE dont la valeur du numero d'identification est Y.

Condition d'exécution:

il existe une occurence de DONNEE dont le numero d'identification est Y.

- Fonction ECRIRE (DONNEE,X,Y)

Spécification:

donne à l'occurence de DONNEE dont le numero d'identification est Y, la valeur de X qui est de même type que DONNEE.

Condition d'exécution:

il existe une occurence de DONNEE dont le numero d'identification est Y.

le numero d'identification de X est aussi Y.

- Fonction CREER (DONNEE,Y)

Spécification:

crée une occurence de DONNEE dont le numero d'identification sera Y.

Condition d'exécution:

Il n'existe pas encore d'occurence de DONNEE dont le numero d'identification est Y.

Autres fonctions utilisée

- Fonction AJOUT_LIAISON (p,q,pos)

Spécification:

effectue la mise à jour de l'occurence de POINT_ARRET dont le numero d'identification est p, pour y ajouter le passage de la liaison dont le numero d'identification est q, la position du point d'arrêt sur la liaison étant pos.

Condition d'exécution:

on suppose qu'il existe une occurence de POINT_ARRET dont le numero d'identification est p.

Description:

Si le point d'arrêt était inactif, on le rend actif et on le signale.
On indique la nouvelle liaison dans la liste et la position.
Le nombre de liaisons passant par ce point est incrémenté de un.

- Fonction IDENT_ARRET (X,p)

Spécification:

Retourne comme valeur de p le numero d'identification de l'occurence de POINT_ARRET dont la valeur NOM est X.
Si cette occurence n'existe pas, retourne 0.

Condition d'exécution:

Il existe au plus une occurence de POINT_ARRET dont la valeur d'item NOM est X.

Description:

Rechercher l'occurence de POINT_ARRET dont la valeur d'item NOM est X ; si on en trouve une, p prend la valeur du numero d'identification sinon p prend la valeur 0.

- Fonction IDENT_LIAISON (X,p)

Spécification:

Retourne comme valeur de p le numero d'identification de l'occurence de LIAISON dont la valeur d'item NUMERO est X.
Si cette occurence n'existe pas, retourne 0.

Condition d'exécution:

Il existe au plus une occurence de LIAISON dont la valeur d'item NUMERO est X.

Description:

Rechercher l'occurence de LIAISON dont la valeur d'item NUMERO est X.
Si on en trouve une, p prend la valeur du numero d'identification, sinon p prend la valeur 0.

- Fonction PRINT_LISTE_LIAISON (préfixe, état)

Spécification:

Toutes les occurences de LIAISON dont l'item NUMERO correspond à préfixe et dont l'item ETAT correspond à état sont affichées à l'écran.

Condition d'exécution:

Description:

Toutes les occurences de LIAISON sont passées en revue.
Pour chaque occurence, on compare la valeur de l'item NUMERO avec préfixe et la valeur de l'item ETAT avec état. S'il y a concordance, la valeur des items NUMERO, ETAT, DISP_HORAIRE, NBRE_POINT_ARRET est affichée ainsi que les items NOM des occurences de POINT_ARRET correspondant aux extrémités de la liaison.

- Fonction PRINT_LISTE_ARRET (préfixe, état)

Spécification:

Toutes les occurences de POINT_ARRET dont l'item NOM correspond à préfixe et dont l'item ETAT correspond à état sont affichées à l'écran.

Condition d'exécution:

—

Description:

Toutes les occurences de POINT_ARRET sont passées en revue.
Pour chaque occurence, on compare la valeur de l'item NOM avec préfixe et la valeur de l'item ETAT avec état. S'il y a concordance, la valeur des items NOM, ETAT, NBRE_LIAISON est affichée.

Fonctions réalisant les commandes

- Fonction CREATION_LIAISON (NUMERO,p)

Spécification:

Permet l'introduction des points d'arrêt constituant la liaison.
Effectue la mise à jour des différentes concurences de POINT_ARRET concernées par la liaison.
Crée une nouvelle occurence de LIAISON avec tous les renseignements introduits.

Condition d'exécution:

Il n'existe pas encore d'occurence de LIAISON dont le numero d'identification est p.
Il n'existe pas encore d'occurence de LIAISON dont la valeur d'item NUMERO est numero.

Description:

Les noms de point d'arrêt sont introduits les uns après les autres. L'introduction d'un nom bidon signifie la fin de l'introduction (par exemple le nom 'O'). Chaque nom d'arrêt introduit est identifié.
Si le point d'arrêt n'existe pas encore, alors on crée une occurence de POINT_ARRET et on donne à l'item NOM la valeur de nom introduite.
La mise à jour de l'occurence de POINT_ARRET est effectuée.
Lorsque tous les points d'arrêt ont été introduits, au moins deux points d'arrêt ont été introduits, on crée une nouvelle occurence de LIAISON et on y écrit les renseignements introduits. L'item DISP_HORAIRE prend la valeur 'faux'.
On crée aussi une nouvelle occurence de GRILLE_HORAIRE dont le numero d'identification sera p. Les items HORAIRE1 et HORAIRE2 de cette occurence sont remplis de renseignements bidons.

- Fonction CREATION_HORAIRE (numero, p)

Spécification:

Permet l'introduction des renseignements formant l'horaire associé à la liaison dont le numero d'identification est p.

Condition d'exécution:

Il existe une occurence de LIAISON dont le numero d'identification est p.
Il n'y a pas encore d'horaire disponible pour cette liaison.

Description:

Les différents ensembles d'heure de passage sont introduits les uns après les autres avec chaque fois, à la fin de l'introduction d'un ensemble, l'introduction des caractéristiques du passage.
Si pour un passage, il n'y a pas d'arrêt effectif en un point d'arrêt, on introduit une heure bidon (par exemple * * h * *).
Lors de l'introduction des heures de passage, on vérifie que les heures soient introduites de manière croissante.

- Fonction MODIFICATION_LIAISON (p)

Spécification:

Permet la modification du contenu d'une liaison.
Les différentes modifications possible sont:

. Insertion d'un ou plusieurs points d'arrêt
. Suppression d'un ou plusieurs points d'arrêt
. Modification d'un ou plusieurs points d'arrêt.

Répercute ces modifications sur l'horaire correspondant

Condition d'exécution:

Il existe une occurence de LIAISON dont le numero d'identification est p.

Description:

Insertion d'un point d'arrêt

Le nouveau point d'arrêt est introduit puis identifier. S'il n'existe pas encore, on crée une nouvelle occurence de POINT_ARRET et on donne à l'item NOM la valeur de nom introduite.
On effectue la mise à jour de l'occurence POINT_ARRET.
Le point d'arrêt est ajouté à la liaison.

Suppression d'un point d'arrêt

On effectue la mise à jour de l'occurence POINT_ARRET correspondant.
On retire le point d'arrêt de la liaison.

Modification d'un point d'arrêt

On effectue la suppression du point d'arrêt puis l'insertion d'un nouveau point d'arrêt.

Chaque fois qu'un point d'arrêt est inséré dans la liaison, on insère une nouvelle colonne correspondante dans les items HORAIRE1 et HORAIRE2 de l'occurence de GRILLE_HORAIRE dont le numero d'identification est p.
Chaque fois qu'on supprime un point d'arrêt, on supprime la colonne correspondante dans les items HORAIRE1 et HORAIRE2 de l'occurence de GRILLE_HORAIRE dont le numero d'identification est p.

- Fonction MODIFICATION_HORAIRE (p)

Spécification:

Permet la modofication d'un horaire associé à une liaison. Les différentes modifications sont les suivantes:

. Modification des caractéristiques d'un passage
. Modification des heures de passage
. Insertion d'un passage
. Suppression d'un passage

Condition d'exécution:

Il existe une occurence de LIAISON et une occurence de GRILLE_HORAIRE dont le numero d'identification est p .

Description:

Insertion d'un passage

On introduit les différentes heures de passage, en vérifiant que l'ordre croissant soit conservé, ensuite on introduit les caractéristiques de passage.

Suppression d'un passage

La ligne correspondante de l'item HORAIRE1 ou HORAIRE2 est supprimée.

Modification d'un passage

On passe en revue les heures de passage formant le passage en introduisant, s'il y a lieu, une nouvelle heure de passage. Ensuite on introduit, si on le désire, des nouvelles caractéristiques de passage.

- Fonction VISUAL_ARRET (nom, p)

Spécification:

Affiche à l'écran les renseignements concernant un point d'arrêt, c'est-à-dire
l'état du point d'arrêt, le nombre de p liaison passant par ce point d'arrêt et la liste de ces liaisons.

Condition d'exécution:

Il existe une occurence de POINT_ARRET dont le numero d'identification est p .

Description:

Ecrire à l'écran le nom du point d'arrêt.
Ecrire à l'écran la valeur des items ETAT et NBRE_LIAISON de l'occurence de POINT ARRET dont le numero d'identification est p .
Ensuite pour chaque élément non nul de l'item LISTE_LIAISON de cette même occurence, écrire à l'écran la valeur de l'item NUMERO de l'occurence de LIAISON dont le numero d'identification est la valeur de cet élément non nul.

- Fonction VISUAL_LIAISON (numero, p)

Spécification:

Affiche à l'écran les renseignements concernant une liaison, c'est-à-dire
l'état de la liaison, le nombre de points d'arrêt constituant la liaison, la disponibilité d'un horaire correct et la liste des points d'arrêt qui constituent la liaison.

Condition d'exécution:

Il existe une occurence de LIAISON dont le numero d'identification est p .

Description:

Ecrire à l'écran le numero de la liaison.
Ecrire à l'écran la valeur des items ETAT, DISP_HORAIRE et NBRE POINT ARRET de l'occurence de LIAISON dont le mumero d'identification est p .
Ensuite, pour chaque élément non nul de l'item LISTE_POINT_ARRET de cette même occurence, écrire à l'écran la valeur de l'item NOM de l'occurence de POINT_ARRET dont la numero d'identification est la valeur de cet élément non nul.

- Fonction VISUAL_HORAIRE (numero, p)

Spécification:

Affiche à l'écran les renseignements concernant l'horaire associé à la liaison dont le numero d'identification est p , c'est-à-dire
la liste des noms de point d'arrêt constituant la liaison, les temps de passage à ces points d'arrêt ainsi que les caractéristiques des passages sous forme de code.

Condition d'exécution:

Il existe une occurence de LIAISON dont le numero d'identification est p .
Il existe une occurence de GRILLE_HORAIRE dont le numero d'identification est p .

Description:

Pour chaque élément non nul de l'item LISTE_POINT_ARRET de l'occurence de GRILLE_HORAIRE dont le numero d'identification est p, écrire à l'écran la valeur de l'item NOM de l'occurence POINT_ARRET dont le numero d'identification est la valeur de cet élément non nul.
Ecrire à l'écran la valeur des items HORAIRE1 et HORAIRE2 de l'occurence de GRILLE_HORAIRE dont le numero d'dentification est p .

- Fonction SUPPRIME_ARRET (p)

Spécification:

Supprime logiquement le point d'arrêt dont le numero d'identification est p , c'est-à-dire
marque le point d'arrêt comme inactif et supprime le passage par ce point d'arrêt pour toutes les liaisons auxquelles il appartient.

Condition d'exécution:

On suppose qu'il existe une occurence de POINT_ARRET dont le numero d'identification est p et que ce point d'arrêt est actif.

Description:

L'item ETAT du point d'arrêt prend la valeur 'I'nactif et pour chaque liaison passant par ce point d'arrêt, la position de la liaison dans la liste du point d'arrêt est mise à 0.

- Fonction SUPPRIME_LIAISON (q)

Spécification:

Supprime logiquement la liaison dont le numero d'identification est q , c'est-à-dire
marque cette liaison comme inactive et supprime le passage de cette liaison en tous ses points d'arrêt.

Condition d'exécution:

On suppose qu'il existe une occurence de LIAISON dont le numero d'identification est q, et que cette liaison est active.

Description:

L'item ETAT de la liaison prend comme valeur 'I'nactif et pour chaque point d'arrêt appartenant à la liaison, la position du point d'arrêt sur la liaison est mise à 0.

- Fonction RESTAURE_LIAISON (q)

Spécification:

Restaure la liaison dont le numero d'identification est q, et qui avait été supprimée logiquement,c.à.d. marque cette liaison comme active et restaure le passage de cette liaison par tous les points d'arrêt qui la forment.

Condition d'exécution:

On suppose qu'il existe une occurence de LIAISON dont le numero d'identification est q et que cette liaison est inactive.

Description:

L'item ETAT de la liaison prend la valeur 'A'ctif et pour chaque point d'arrêt appartenant à la liaison la position du point d'arrêt sur la liaison reprend sa vraie valeur.

- Fonction RESTAURE_ARRET (p)

Spécification:

Restaure le point d'arrêt dont le numero d'identification est p, et qui avait été supprimé logiquement, c'est-à-dire
marque ce point d'arrêt comme actif et restaure le passage par ce point d'arrêt pour toutes les liaisons auxquelles il appartient.

Condition d'exécution:

On suppose qu'il existe une occurence de POINT_ARRET dont le numero d'identification est p, et que ce point d'arrêt est inactif.

Description:

L'item ETAT du point d'arrêt prend la valeur 'A'ctif et pour chaque liaison passant par ce point, la position de la liaison dans la liste des points d'arrêt reprend sa vraie valeur.

- Fonction LISTE_ARRET

Spécification:

Permet l'introduction des paramètres pour la recherche d'une sous liste de points d'arrêt, c'est-à-dire un préfixe pour le nom et pour l'état du point d'arrêt Effectue la recherche et l'impression de la sous liste.

Condition d'exécution:

Description:

Introduction du préfixe. Si ce préfixe est la chaine vide, cela signifie que la recherche ne se fait pas sur base du préfixe.
Introduction de l'état. Si cet état est la chaine vide, cela signifie que la recherche ne se fait pas sur base de l'état.
Recherche et impression de la sous liste sur base de paramètres intriduits.

- Fonction LISTE_LIAIS.

Spécification:

Permet l'introduction des paramètres pour la recherche d'une sous liste de liaisons, c'est-à-dire un préfixe pour le numero et/ou l'état de la liaison. Effectue la recherche et l'impression de la sous liste.

Condition d'exécution:

Description:

Introduction du préfixe. Si ce préfixe est la chaine vide, cela signifie que la recherche ne se fait pas sur base du préfixe.
Introduction de l'état. Si cet état est la chaine vide, cela signifie que la recherche ne se fait pas sur base de l'état.
Recherche et impression de la sous liste sur base des paramètres introduits.

Gérant des commandes

- Format d'une commande

Une commande est représentée par une chaine de caractères alphanumériques.

Le premier caractère de cette chaine constitue le type de commade

C pour Créer
M pour Modifier
S pour Supprimer
R pour Restaurer
V pour Visualiser
L pour Listes

Le second caractère représente le premier paramètre de la commande, c'est-à-dire le type de structure de donnée associé à la commande

H pour Horaire
L pour Liaison
P pour Point d'arrêt

Le reste de la chaine est constitué par le second paramètre de la commande, s'il y en a un, encadré par deux «'».
De plus, si ce paramètre est un nom de point d'arrêt, alors le nom de la localité est séparé du nom de l'endroit par un /
→ Z (45)

Exemples:

Créer la liaison de numero 126A1
CL '126A1'

Supprimer le point d'arrêt NAMUR, GARE
SP 'NAMUR/GARE'

Visualiser l'horaire de la liaison 296
VH '296'

- Deux commandes supplémentaires sont ajoutées:

  Q pour quitter le module de gestion des données.
  ? pour obtenir à l'écran une description des différentes commandes disponibles.

- Description

La commande est introduite.
Ensuite cette commande est validée, c'est-à-dire la chaine de caractères est analysée, et les paramètres sont validés en fonction de la commande demandée sur base d'un tableau

(voir page suivante)

Une fois que la commande a été analysée et validée, on exécute la fonction réalisant cette commande.

COMMANDE	Premier PARAMETRE	Deuxième PARAMETRE
C	L	$\nexists$p tq. LIAISON(:NOM_ID=p).NUMERO=numero
	H	$\exists$p tq. LIAISON(:NUM_ID=p).NUMERO=numero et LIAISON(:NUM_ID=p).DISP_HORAIRE='faux'
M	L	$\exists$p tq. LIAISON(:NUM_ID=p).NUMERO=numero
	H	$\exists$p tq. LIAISON(:NUM_ID=p).NUMERO=numero et LIAISON(:NUM_ID=p).DISP_HORAIRE='faux'
S	L	$\exists$p tq. LIAISON(:NUM_ID=p).NUMERO=numero et LIAISON(:NUM_ID=p).ETAT='A'
	P	$\exists$p tq. POINT_ARRET(:NUM_ID=p).NOM=nom et POINT_ARRET(:NUM_ID=p).ETAT='A'
R	L	$\exists$p tq. LIAISON(:NUM_ID=p).NUMERO=numero et LIAISON(:NUM_ID=p).ETAT='I'
	P	$\exists$p tq. POINT_ARRET(:NOM_ID=p).NOM=nom et POINT_ARRET(:NOM_ID=p).ETAT='I'
V	L	$\exists$p tq. LIAISON(:NUM_ID=p).NUMERO=numero
	P	$\exists$p tq. POINT_ARRET(:NUM_ID=p).NOM=nom
	H	$\exists$p tq. LIAISON(:NUM_ID=p).NUMERO=numero et LIAISON(:NUM_ID=p).DISP_HORAIRE='vrai'
L	L ou P	quelconque

CONCLUSION

Il n'est guère possible de tirer des conclusions importantes d'un travail, qui pour être complet, aurait nécessité beaucoup plus de temps qu'il ne m'était possible d'y consacrer (travail prévu pour deux personnes au départ). Ainsi, il aurait été intéressant de pouvoir présenter une formalisation plus mathématique du problème, ainsi que l'application concrète d'un problème particulier.

Néanmoins, malgré son aspect heuristique, il est raisonnable de penser que la méthode choisie constitue une bonne approche pour la résolution du problème, et que la solution devant résulter de son application devrait convenir à l'utilisateur.

Annexe 1

Techniques de stockage de matrices creuses

- Compactage par ligne ou par colonne.

Principe: Seuls sont stockés les éléments non nuls de la matrice Ainsi un premier vecteur contient les éléments non nuls de la matrice, rangés par ligne ou par colonne. Un second vecteur contient la paire d'indices ligne et colonne correspondant à chaque élément non nul.

Exemple:
$$\begin{pmatrix} 0 & 2 & 0 & 0 \\ 6 & 0 & 4 & 0 \\ 3 & 0 & 0 & 0 \\ 7 & 9 & 0 & 5 \end{pmatrix}$$

Compactage par ligne

V
2
6
4
3
7
9
5

L	C
1	2
2	1
2	3
3	1
4	1
4	2
4	4

Compactage par colonne

V
6
3
7
2
9
4
5

L	C
2	1
3	1
4	1
1	2
4	2
2	3
4	4

C'est une méthode simple, mais guère efficace
ainsi pour chercher l'élément (3,3) par exemple, dans le cas d'un compactage par ligne, il faut réaliser les opérations suivantes:

Parcourir le vecteur des lignes jusqu'au moment où se trouve la valeur 3. Ensuite, à partir de cet endroit, continuer à parcourir le vecteur des lignes jusqu'au moment où il y a changement de ligne, ou jusqu'au moment ou l'indice colonne

correspondant prend soit la valeur 3, soit une valeur supérieur

3 , 1
4 , 1 ← changement de ligne ⇒ élément (3,3) = 0

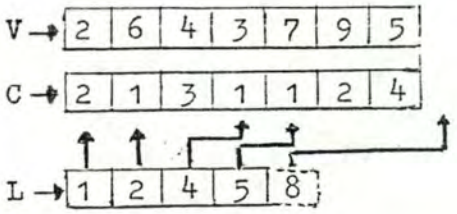
Cherchons l'élément (2,3)

1) 1 ,
ligne = 2 → 2) 2 , 1
3) 2 , 3 ← colonne = 3 et ligne = 2
=) l'élément 3 du tableau des valeurs correspond à l'élément (2,3) de la matrice.

- Autre méthode

Un vecteur qui contient les éléments non nul de la matrice,
Un vecteur qui contient les indices de colonne (ligne) correspondant.
Un troisième vecteur contient la position du premier élément non nul de chaque ligne (colonne), position par rapport au vecteur qui contient les valeurs des éléments non nuls.

Exemple:



Pour chercher l'élément (3,3):

Se positionner dans C sur l'élément L(3) = 4, parcourir C jusque l'élément L(4)=5
C(4)=1 ≠ 3 , C(5) correspond à la ligne 4.
→ l'élément (3,3)=0.

Pour chercher l'élément (2,3):

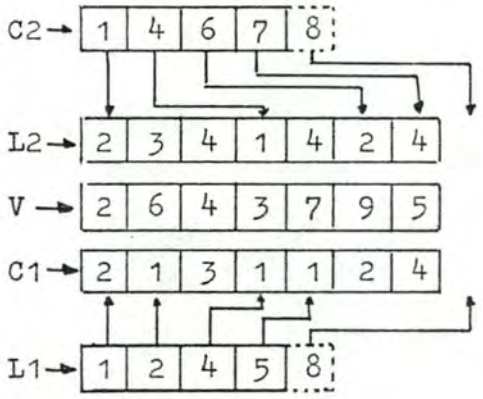
Se positionner dans C sur l'élément L(2)=2, parcourir C jusque l'élément L(3)=4
C(2)=1 ≠ 3, C(3)=3 ok.
l'élément (3,3) = V(3)=4

Si on veut tous les éléments non nuls d'une ligne, c'est assez simple

-- parcourir le vecteur V de L(i) jusque L(i+1)-1

Mais si on veut tous les éléments non nuls d'une colonne, il faut ajouter le compactage par colonne

ce qui donne:

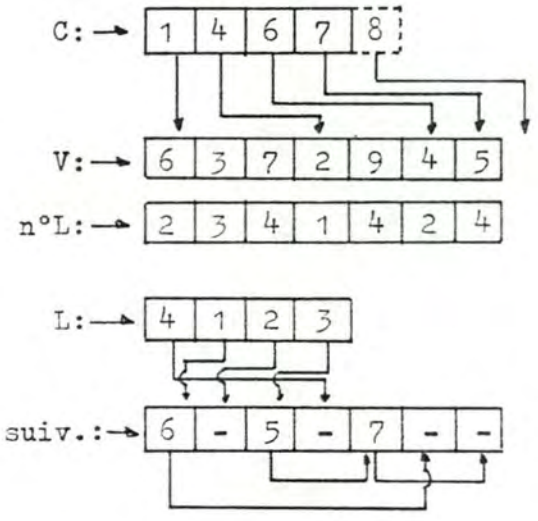


Cette méthode, si elle permet un peu plus d'accés et d'opérations, nécessite plus de place.
Elle empêche aussi toute extension de la matrice.

- Une autre méthode pour permettre l'accès en ligne (colonne), si on utilise un compactage par colonne (ligne), est la suivante:

  Si on utilise un compactage par colonne (ligne), on ajoute un vecteur donnant pour chaque ligne (colonne) la position du premier élément non nul de la ligne (colonne) dans le vecteur des valeurs, et un vecteur parallèle au vecteur des valeurs, qui donne la position de l'élément non nul suivant dans la ligne ou indique qu'il n'y en a plus.

Exemple:

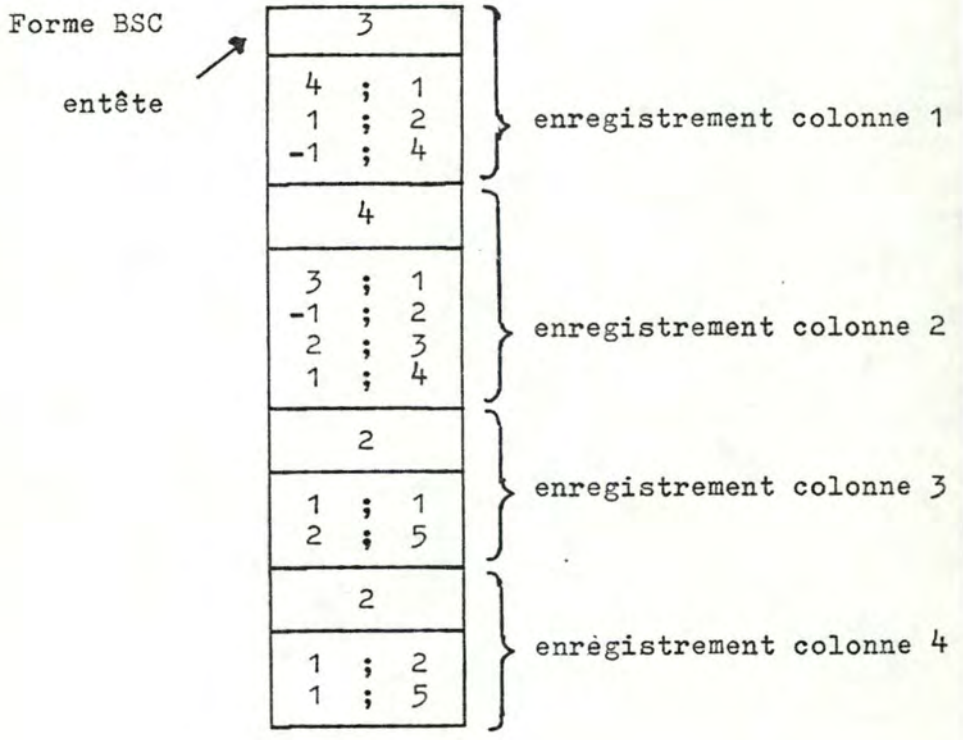


Remarque: Toutes ces méthodes ont le gros désavantage de ne pouvoir être conserver les matrices dans un fichier.

- Forme colonne séquentielle de base (B.S.C.)

Exemple: soit la matrice

$$\begin{pmatrix} 4 & 3 & 1 & 0 \\ 1 & -1 & 0 & 1 \\ 0 & 2 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 2 & 1 \end{pmatrix}$$



- Entête -- nombre d'éléments non nuls dans la colonne.
- Paire (valeur de l'élément, indice de ligne)

Cette méthode permet l'extension en colonne, mais pas en ligne

Annexe 2

Données particulières à une région choisie: Florenville

Liste des lignes (23)

163a, 165, 165a, 975, 989, 990, 991, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1015, 1016, 1018, 1019, 1020, 1021, 1022, 1030, 1033.

Liste des points desservis

ALLE : dépôt SNCV 975, 991, 1008, 1020, 1021
Choir 975
Village 975

BELLEVAUX : Menuchenet 989, 990, 1019

BOHAN: Centre 991, 1008, 1021

BOUILLON: Pont de Liège 989, 990, 1008
Dépôt SNCV 989, 990, 1008
Pont de France 990, 1008
Place ST Arnould 1018, 1019

CHASSEPIERRE: Eglise 1008, 1016

CHINY: Eglise 1005, 165a

CORBION: Route de Bouillon 989
Route de la Semois 1008

DOHAN: Eglise 1008, 1019

FLORENVILLE: Place 165a, 1004, 1005, 1006, 1008, 1015, 1016
Gare SNCB 1004, 1005, 1006, 1008, 1016
EPE 1016

FONTENOILLE: Monument 1016
4 Arbres 1016

IZEL: Institut 165a, 1004

LACUISINE: Pont 1005, 1015
Eglise 1015

LES HAYONS: - 1019

MUNO: - 163a
Douane 1016
Eglise 1016

NOIREFONTAINE: Gare 975
Eglise 1019

OIZY: Eglise 975
Centre 1020

ORCHIMONT: Village 991

POUPEHAN: Hôtel de la Semois 1008

ROCHEHAUT: Centre 1008

SENSENRUTH: Maison communale 989, 990, 1008, 1019

Ste CECILE: Eglise 163a
Place 1008
Centre 1016

SUGNY: Eglise 990
Pensionnat 1021

SUXY: Place 1005

UCIMONT: 989

VILLERS-devant-Orval: Place 1006

VRESSE: Eglise 991, 1008, 1020, 1021

Liste des lignes avec leurs arrêts

163a

Sainte Cécile (Eglise)
Muno

165a

Florenville (Place)
...
Izel (institut)
...
Virton (Eglise)
Virton (gare SNCB)

975

Oizy (Eglise)
Oizy (Centre)
Alle sur Semois (Choir C.)
Alle sur Semois (Village)
Alle sur Semois (Dépôt SNCV)

989

Bouillon (Dépôt SNCV)
Bouillon (Pont de Liège)
Noirefontaine (Gare)
Sensenruth (Maison communale)
...
Bellevaux (Menuchenet)

990

Sugny (Eglise)
Corbion (Route de Bouillon)
Bouillon (Pont de France)
Bouillon (Dépôt SNCV)
Bouillon (Pont de Liège)
Noirefontaine (Gare)
Sensenruth (Maison communale)

990 (suite)

Ucimont (Route de Mogimont)

...

Bellevaux (Menuchenet)

...

Noirefontaine (Eglise)

Bellevaux (Eglise)

991

Alle sur Semois (Dépôt SNCV)

...

...

Vresse (Eglise)

...

Bohan (Centre)

...

Vresse (Eglise)

Orchimont (Village)

...

Gedinne (Village)

Gedinne (Gare SNCB)

1004

Florenville (Gare SNCB)

Florenville (Place)

...

Izel (Institut)

1005

Florenville (Place)

Florenville (Gare SNCB)

Pont de Lacuisine

Chiny (Eglise)

Chiny (Embarc.)

Chiny (Camping)

Suxy

1006

Florenville (Gare SNCB)
Florenville (Place)
...
Villers devant Orval (Place)
...
Virton (Hôtel de Ville)
Virton (Pierrard)

1008

Florenville (Gare SNCB)
Florenville (Place)
Chassepierre (Eglise)
Sainte-Cécile (Place)
...
...
...
Dohan (Eglise)
Noirefontaine (Gare)
Bouillon (Pont de Liège)
Bouillon (SNCV)
Bouillon (Pont de France)
Corbion (Route de la Semois)
Poupehan (Hôtel de la Semois)
Rochehaut (Centre)
Alle (Dépôt SNCV)
...
Vresse (Eglise)
...
Bohan (Centre)

1015

Florenville (Place)
...
Lacuisine (Eglise)
Lacuisine (Pont)
Florenville (Place)

1016

Florenville (Gare SNCB)
Florenville (Place)
Florenville (E.P.E.)
Florenville (Place)
...
Chassepierre (Eglise)
Sainte-Cécile (Centre)
Fontenoille (Monument)
Fontenoille (4 Arbres)
...
...
Muno (Douane)
Muno (Eglise)

1018

Sedan (Gare SNCF)
Sedan (Place Turenne)
Sedan (Place d'Alsace)
Sedan (Place Nassan)
...
Bouillon (Place St Arnould)

1019

Bouillon (Place St Arnould)
Noirefontaine (Eglise)
...
Dohan (Eglise)
Les H yons

1020

Alle (Dépôt SNCV)

...

...

...

Vresse (Eglise)

...

Vresse (Eglise)

...

...

...

Oizy (Centre)

...

Gedinne (Gare SNCB)

Gedinne (Ecole Moyenne)