CONCOURS COMMUNS POLYTECHNIQUES

ÉPREUVE SPÉCIFIQUE-FILIÈRE MP

INFORMATIQUE

DURÉE : 2 heures

Les calculatrices programmables et alphanumériques sont autorisées, sous réserve des conditions définies dans la circulaire n° 86-228 du 28 juillet 1986.

PRÉAMBULE: Les trois parties qui composent ce sujet sont indépendantes et peuvent être traitées par les candidats dans un ordre quelconque.

Partie I : Logique et calcul des propositions

Une nouvelle série de composants informatiques dédiés au raisonnement logique a été conçue de manière à faciliter la détection de pannes. Chaque processeur effectue des raisonnements logiques et peut être, soit en état de fonctionnement normal, soit en état de panne. Il se comporte alors de la manière suivante :

- un processeur en état de fonctionnement normal ne peut affirmer que des propositions vraies;
- un processeur en état de panne ne peut affirmer que des propositions fausses.

Un ordinateur est composé de trois processeurs qui possèdent la même mémoire, donc les mêmes connaissances. Périodiquement, un ingénieur vient interroger l'ordinateur pour déterminer si certains processeurs sont en état de panne.

Lors d'une séance de test, l'ingénieur pose les deux questions suivantes au processeur n° 1 :

- « *Est-ce que les processeurs n° 2 et n° 3 sont en état de fonctionnement normal ?*»
- « *Est-ce que le processeur n° 2 est en état de fonctionnement normal ?*»

Le processeur n° 1 répond à la première question : « *Les processeurs n° 2 et n° 3 sont en état de fonctionnement normal.* »

Puis, il répond à la seconde question : « *Le processeur n° 2 est en état de panne.* »

Nous noterons P1 (respectivement P2 et P3) la proposition « *le processeur n° 1 (respectivement n° 2 et n° 3) est en état de panne* ».

Nous supposerons que l'état des trois processeurs ne peut pas changer entre les réponses aux deux questions.

Question I.1 *Exprimer la réponse à la première question sous la forme d'une formule du calcul des propositions.*

Tournez la page S.V.P.

Question I.2 *Exprimer la réponse à la deuxième question sous la forme d'une formule du calcul des propositions.*

Question I.3 *En utilisant le calcul des propositions (table de vérité ou formule de De Morgan), déterminer l'état de chaque processeur.*

Partie II : Algorithmique et programmation en CaML

Cette partie doit être traitée par les étudiants qui ont utilisé le langage CaML dans le cadre des enseignements d'informatique.

Le but de cette partie est l'étude de l'algorithme de Prim utilisé pour construire des réseaux d'interconnexion de coût minimal. Ses applications pratiques concernent, entre autres, la conception de réseaux routiers, de réseaux de distribution de gaz, d'eau, d'électricité, de téléphonie. Cette étude sera réalisée en deux étapes : l'étude d'un algorithme de construction d'un réseau quelconque, puis l'étude de l'algorithme de Prim qui produit un réseau de coût minimal.

Le problème posé consiste à connecter tous les points d'un ensemble de points de manière à ce que chaque point soit accessible depuis tous les autres points. De plus, le réseau construit ne devra pas contenir de cycles.

Lors de la construction du réseau, certaines connexions ne pourront pas être réalisées pour des raisons techniques. Certains points ne pourront donc être reliés que par un chemin composé d'une suite de connexions toutes distinctes. Pour distinguer ces deux cas, nous noterons $\langle A, B\rangle$ la connexion de A à B et $\langle\langle A, \cdots, B\rangle\rangle$ le chemin de A à B.

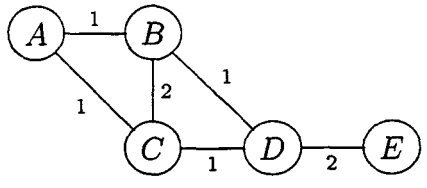
Nous considérerons uniquement des problèmes pour lesquels il est possible de construire un réseau reliant tous les points en utilisant les connexions techniquement réalisables (problème connexe).

Un problème de construction de réseau sera donc caractérisé par l'ensemble des points que l'on doit connecter et par l'ensemble des connexions possibles.

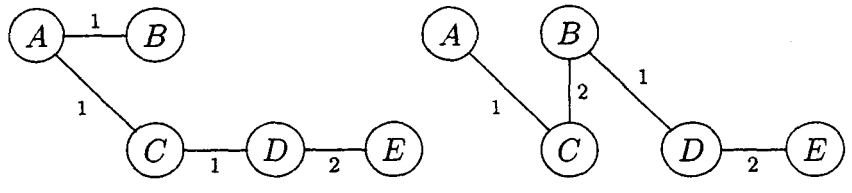
Nous appellerons réseau partiel un réseau connectant un sous-ensemble des points du problème.

De plus, chaque connexion possible aura un coût propre à sa réalisation. Le coût d'un réseau est alors égal à la somme des coûts de chaque connexion le composant.

Exemple II.1 *Le problème consistant à relier les points $\{A, B, C, D, E\}$ en utilisant les connexions possibles $\{\langle A, B\rangle, \langle A, C\rangle, \langle B, C\rangle, \langle B, D\rangle, \langle C, D\rangle, \langle D, E\rangle\}$ sera représenté par le graphe dont les connexions sont étiquetées par leur coût de réalisation :*



A partir du problème précédent, on peut construire plusieurs réseaux dont les deux réseaux suivants :



Question II.1 *Donner, de même, une représentation graphique de tous les réseaux qui peuvent être construits à partir de ce problème.*

Tournez la page S.V.P.

Question II.2 *Préciser, parmi les différents réseaux possibles, les réseaux de coût minimal.*

1 Représentation du problème en CaML

Les points sont représentés par des valeurs entières.

```
type point == int;;
```

Une connexion et son coût sont alors représentés par un triplet dont les deux premiers éléments sont les extrémités et le dernier est le coût.

```
type connexion == point * point * int;;
```

Un problème de construction d'un réseau est alors composé d'une paire dont le premier élément est une liste d'entiers (les points) et le second élément est une liste de triplets (les connexions).

```
type reseau == (point list) * (connexion list);;
```

Exemple II.2 *En associant les entiers* $1, 2, 3, 4, 5$ *aux points* A, B, C, D, E, *le problème donné en exemple est représenté par la valeur :*

```
( [ 1 ; 2 ; 3 ; 4 ; 5 ] ,
  [ ( 1 , 2 , 1 ) ; ( 1 , 3 , 1 ) ; ( 2 , 3 , 2 ) ;
    ( 2 , 4 , 1 ) ; ( 3 , 4 , 1 ) ; ( 4 , 5 , 2 ) ] )
```

Nous supposons prédéfinies les fonctions suivantes dont le calcul se termine quelles que soient les valeurs de leurs paramètres. Elles pourront éventuellement être utilisées dans les réponses aux questions :

- `appartient` de type `'a -> 'a list -> bool` telle que l'appel `(appartient e l)` renvoie la valeur vrai si `e` appartient à la liste `l`,
- `taille` de type `'a list -> int` telle que l'appel `(taille l)` renvoie le nombre d'éléments contenus dans la liste `l`.

2 Construction d'un réseau quelconque

Soit un problème connexe d'interconnexion des points de l'ensemble P en utilisant les connexions possibles de l'ensemble C (nous noterons $\mathcal{C}(C)$ l'ensemble des chemins que l'on peut construire à partir de l'ensemble de connexions C). L'algorithme construit une suite de réseaux partiels (P_i, C_i) en partant d'un réseau (P_0, C_0) comportant un seul point, puis en ajoutant un point et une connexion à chaque étape, jusqu'à atteindre une solution (P, C_k). La connexion ajoutée à chaque étape relie un point du réseau partiel à un point externe au réseau partiel. L'algorithme s'arrête lorsque tous les points ont été ajoutés.

2.1 Correction de l'algorithme

On remarquera que : $\forall p \in P, (\{p\}, \emptyset)$ est un réseau que l'on notera (P_0, C_0).

Question II.3 *Montrer que : si (P_i, C_i) est un réseau et $p \in P_i$ et $q \in P_i$ et $\langle p, q\rangle \notin C_i$ alors $(P_i, C_i \cup \{\langle p, q\rangle\})$ contient au moins un cycle.*

Question II.4 *Montrer que : si (P_i, C_i) est un réseau et $p \notin P_i$ et $q \notin P_i$ alors $(P_i \cup \{p, q\}, C_i \cup \{\langle p, q\rangle\})$ n'est pas un réseau.*

Question II.5 *Montrer que : si (P_i, C_i) est un réseau et $p \in P_i$ et $q \notin P_i$ et $P_{i+1} = P_i \cup \{q\}$ et $C_{i+1} = C_i \cup \{\langle p, q\rangle\}$ alors (P_{i+1}, C_{i+1}) est un réseau.*

Question II.6 *Montrer que : si (P_k, C_k) est un réseau construit par l'algorithme précédent avec $k = card(P)$, alors ce réseau répond au problème (P, C).*

2.2 Préliminaire : Construction d'un cocycle

Soient un ensemble P de points et un ensemble C de connexions possibles entre les points de P, le cocycle d'un sous-ensemble Q de points de P est le sous-ensemble des connexions de C reliant un point de Q à un point qui n'appartient pas à Q (donc appartenant à $P \setminus Q$). Nous noterons $\omega(Q)$ le cocycle de Q dans (P, C). La connexion ajoutée au réseau (P_i, C_i) sera donc prise dans $\omega(P_i)$.

Question II.7 *Écrire en CaML une fonction récursive* `cocycle` *de type* `point list -> connexion list -> connexion list` *telle que l'appel* `(cocycle q c)` *calcule le cocycle du sous-ensemble de points* `q` *de l'ensemble de points P reliés par les connexions de l'ensemble* `c`.

Question II.8 *Montrer que le calcul de la fonction* `cocycle` *se termine quelles que soient les valeurs de ses paramètres.*

Question II.9 *Donner une estimation de la complexité de la fonction* `cocycle` *en nombre d'appels récursifs à* `cocycle`. *Justifier cette estimation.*

2.3 Algorithme de construction du réseau

Question II.10 *Appliquer* « à la main » *l'algorithme précédent sur l'exemple II.2. Utiliser pour cela la représentation graphique.*

Question II.11 *Écrire une fonction CaML* `construire`, *appliquant l'algorithme précédent, de type* `point list -> connexion list -> reseau` *telle que l'appel* `(construire p c)` *construit un réseau pour le problème connexe qui consiste à relier l'ensemble de points* `p` *en choisissant les connexions parmi l'ensemble* `c`. *Cette fonction fera appel à des fonctions récursives.*

Question II.12 *Montrer que le calcul de la fonction* `construire` *se termine quelles que soient les valeurs de ses paramètres.*

Tournez la page S.V.P.

Question II.13 *Donner une estimation de la complexité de la fonction* `construire` *en fonction de la complexité de* `cocycle` *et du nombre d'appels récursifs internes à* `construire`*. Justifier cette estimation.*

3 Construction d'un réseau de coût minimal

À partir d'un ensemble donné de connexions réalisables, il est généralement possible de construire plusieurs réseaux distincts répondant au problème d'interconnexion. Les coûts de construction de ces réseaux sont en général différents. Il est donc nécessaire d'adapter l'algorithme précédent de manière à construire un réseau de coût de réalisation minimal. Pour cela, la connexion ajoutée au réseau partiel à chaque étape de l'algorithme ne sera plus choisie au hasard dans le cocycle mais sera la connexion de coût minimal parmi les connexions possibles du cocycle. L'algorithme obtenu est appelé « *algorithme de Prim* ».

3.1 Correction de l'algorithme de Prim

Soit (X, Y) un réseau partiel pour le problème (P, C).
Soit $H = (P, Z)$ un réseau de coût minimal répondant à ce même problème tel que $Y \subseteq Z$.
Soit une connexion $c' = \langle p_0, p_n \rangle \in \omega(X)$ telle que $c' \notin Z$.
On remarquera alors que $(P, \{c'\} \cup Z)$ contient au moins un cycle (application de la question II.3) noté $\langle\langle p_0, \cdots, p_n, p_0 \rangle\rangle$ et composé des connexions $\{\langle p_0, p_1 \rangle, \langle p_1, p_2 \rangle, \cdots, \langle p_{n-1}, p_n \rangle, c'\}$.

Question II.14 *Montrer que* $\omega(X) \cap \{\langle p_0, p_1 \rangle, \langle p_1, p_2 \rangle, \cdots, \langle p_{n-1}, p_n \rangle\} \neq \emptyset$.

Question II.15 *Soit c une connexion de* $\omega(X) \cap \{\langle p_0, p_1 \rangle, \langle p_1, p_2 \rangle, \cdots, \langle p_{n-1}, p_n \rangle\} \neq \emptyset$.
Montrer que $H' = (P, Z')$ *avec* $Z' = (Z \setminus \{c\}) \cup \{c'\}$ *est un réseau quelconque répondant au problème.*

Question II.16 *Donner une relation liant les coûts de* H' *et de* H.

Question II.17 *Montrer que : si* c' *est une connexion de coût minimal dans* $\omega(X)$ *alors* H' *est un réseau de coût minimal.*

Question II.18 *Montrer que l'application de l'algorithme de Prim produit un réseau minimal pour un problème donné.*

3.2 Algorithme de Prim : Construction d'un réseau de coût minimal

On dispose de plus d'une fonction récursive CaML `inserer` de type `connexion -> connexion list -> connexion list` telle que l'appel `(inserer c l)` sur une liste `l` ordonnée selon le coût des connexions, renvoie la liste `l` dans laquelle la connexion `c` a été insérée en respectant cet ordre. On ne demande pas d'écrire cette fonction, dont on admettra que la complexité est un $O(taille(l))$.

Question II.19 *Modifier la fonction* `cocycle` *de la question II.7 en utilisant la fonction* `inserer` *pour construire un cocycle ordonné.*

Question II.20 *Donner une estimation de la complexité dans le pire cas en fonction de la complexité de* `inserer` *et du nombre d'appels récursifs de la fonction* `cocycle` *modifiée. Justifier cette estimation.*

Question II.21 *Donner une estimation de la complexité dans le pire cas en fonction de la complexité de* `cocycle` *et du nombre d'appels récursifs internes à la fonction* `construire` *utilisant la nouvelle fonction* `cocycle`*. Justifier cette estimation.*

Tournez la page S.V.P.

Partie II : Algorithmique et programmation en PASCAL

Cette partie doit être traitée par les étudiants qui ont utilisé le langage PASCAL dans le cadre des enseignements d'informatique.

Le but de cette partie est l'étude de l'algorithme de Prim utilisé pour construire des réseaux d'interconnexion de coût minimal. Ses applications pratiques concernent, entre autres, la conception de réseaux routiers, de réseaux de distribution de gaz, d'eau, d'électricité, de téléphonie. Cette étude sera réalisée en deux étapes : l'étude d'un algorithme de construction d'un réseau quelconque, puis l'étude de l'algorithme de Prim qui produit un réseau de coût minimal.

Le problème posé consiste à connecter tous les points d'un ensemble de points de manière à ce que chaque point soit accessible depuis tous les autres points. De plus, le réseau construit ne devra pas contenir de cycles.

Lors de la construction du réseau, certaines connexions ne pourront pas être réalisées pour des raisons techniques. Certains points ne pourront donc être reliés que par un chemin composé d'une suite de connexions toutes distinctes. Pour distinguer ces deux cas, nous noterons $\langle A, B\rangle$ la connexion de A à B et $\langle\langle A, \cdots, B\rangle\rangle$ le chemin de A à B.

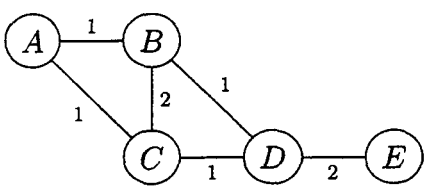
Nous considérerons uniquement des problèmes pour lesquels il est possible de construire un réseau reliant tous les points en utilisant les connexions techniquement réalisables (problème connexe).

Un problème de construction de réseau sera donc caractérisé par l'ensemble des points que l'on doit connecter et par l'ensemble des connexions possibles.

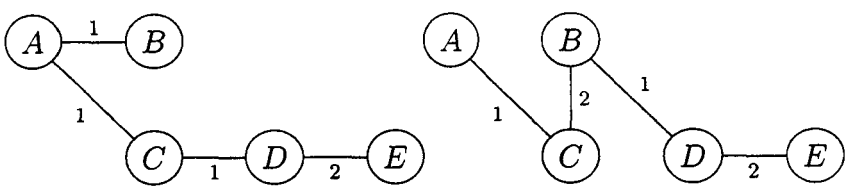
Nous appellerons réseau partiel un réseau connectant un sous-ensemble des points du problème.

De plus, chaque connexion possible aura un coût propre à sa réalisation. Le coût d'un réseau est alors égal à la somme des coûts de chaque connexion le composant.

Exemple II.1 *Le problème consistant à relier les points $\{A, B, C, D, E\}$ en utilisant les connexions possibles $\{\langle A, B\rangle, \langle A, C\rangle, \langle B, C\rangle, \langle B, D\rangle, \langle C, D\rangle, \langle D, E\rangle\}$ sera représenté par le graphe dont les connexions sont étiquetées par leur coût de réalisation :*



A partir du problème précédent, on peut construire plusieurs réseaux dont les deux réseaux suivants :



Question II.1 *Donner, de même, une représentation graphique de tous les réseaux qui peuvent être construits à partir de ce problème.*

Question II.2 *Préciser, parmi les différents réseaux possibles, les réseaux de coût minimal.*

1 Représentation du problème en PASCAL

Les points sont représentés par des valeurs entières (type `INTEGER`).

Une liste de points est représentée par le type de base `POINTS`.

Une connexion, composée de deux extrémités et d'un coût, est représentée par le type de base `CONNEXION`.

Une liste de connexions est représentée par le type de base `CONNEXIONS`.

Un problème de construction d'un réseau, composé de la liste des points et de la liste des connexions, est représenté par le type de base `RESEAU`.

Nous supposons prédéfinies les constantes et les fonctions suivantes dont le calcul se termine quelles que soient les valeurs de leurs paramètres. Elles pourront éventuellement être utilisées dans les réponses aux questions :

- `NIL` représente la liste vide de points ou de connexions,
- `FUNCTION LP_cons(t:INTEGER; q:POINTS):POINTS;` est une fonction qui renvoie une liste de points dont la tête est `t` et la queue `q`,
- `FUNCTION LP_tete(pts:POINTS):INTEGER;` est une fonction qui renvoie la tête de `pts`,
- `FUNCTION LP_queue(pts:POINTS):POINTS;` est une fonction qui renvoie la queue de `pts`,
- `FUNCTION LP_appartient(p:INTEGER; pts:POINTS):BOOLEAN;` est une fonction qui renvoie la valeur vrai si `p` appartient à la liste `pts`,
- `FUNCTION LP_taille(pts:POINTS):INTEGER;` est une fonction qui renvoie le nombre d'éléments contenus dans la liste `pts`,
- `FUNCTION C_cons(p1,p2:INTEGER; c:INTEGER):CONNEXION;` est une fonction qui renvoie une connexion dont les extrémités sont `p1` et `p2` et dont le coût est `c`,
- `FUNCTION C_extr1(cx:CONNEXION):INTEGER;` est une fonction qui renvoie la première extrémité de `cx`,
- `FUNCTION C_extr2(cx:CONNEXION):INTEGER;` est une fonction qui renvoie la seconde extrémité de `cx`,
- `FUNCTION C_cout(cx:CONNEXION):INTEGER;` est une fonction qui renvoie le coût de `cx`,
- `FUNCTION LC_cons(t:CONNEXION; q:CONNEXIONS):CONNEXIONS;` est une fonction qui renvoie une liste de connexions dont la tête est `t` et la queue `q`,
- `FUNCTION LC_tete(cxs:CONNEXIONS):CONNEXION;` est une fonction qui renvoie la tête de `cxs`,
- `FUNCTION LC_queue(cxs:CONNEXIONS):CONNEXIONS;` est une fonction qui renvoie la queue de `cxs`,
- `FUNCTION R_cons(pts:POINTS; cxs:CONNEXIONS):RESEAU;` est une fonction qui renvoie un réseau constitué des points de la liste `pts` reliés par les connexions de la liste `cxs`.

Tournez la page S.V.P.

Exemple II.2 *En associant les entiers* $1, 2, 3, 4, 5$ *aux points* A, B, C, D, E, *le problème donné en exemple est représenté par la valeur :*

```
R_cons(
   LP_cons( 1 , LP_cons( 2 , LP_cons( 3 ,
      LP_cons( 4 , LP_cons( 5 , NIL ))))) ,
   LC_cons( C_cons( 1 , 2 , 1 ) , LC_cons( C_cons( 1 , 3 , 1 ) ,
      LC_cons( C_cons( 2 , 3 , 2 ) , LC_cons( C_cons( 2 , 4 , 1 ) ,
      LC_cons( C_cons( 3 , 4 , 1 ) , LC_cons( C_cons( 4 , 5 , 2 ) ,
      NIL )))))))
```

2 Construction d'un réseau quelconque

Soit un problème connexe d'interconnexion des points de l'ensemble P en utilisant les connexions possibles de l'ensemble C (nous noterons $\mathcal{C}(C)$ l'ensemble des chemins que l'on peut construire à partir de l'ensemble de connexions C). L'algorithme construit une suite de réseaux partiels (P_i, C_i) en partant d'un réseau (P_0, C_0) comportant un seul point, puis en ajoutant un point et une connexion à chaque étape, jusqu'à atteindre une solution (P, C_k). La connexion ajoutée à chaque étape relie un point du réseau partiel à un point externe au réseau partiel. L'algorithme s'arrête lorsque tous les points ont été ajoutés.

2.1 Correction de l'algorithme

On remarquera que : $\forall p \in P, (\{p\}, \emptyset)$ est un réseau que l'on notera (P_0, C_0).

Question II.3 *Montrer que : si* (P_i, C_i) *est un réseau et* $p \in P_i$ *et* $q \in P_i$ *et* $\langle p, q\rangle \notin C_i$ *alors* $(P_i, C_i \cup \{\langle p, q\rangle\})$ *contient au moins un cycle.*

Question II.4 *Montrer que : si* (P_i, C_i) *est un réseau et* $p \notin P_i$ *et* $q \notin P_i$ *alors* $(P_i \cup \{p, q\}, C_i \cup \{\langle p, q\rangle\})$ *n'est pas un réseau.*

Question II.5 *Montrer que : si* (P_i, C_i) *est un réseau et* $p \in P_i$ *et* $q \notin P_i$ *et* $P_{i+1} = P_i \cup \{q\}$ *et* $C_{i+1} = C_i \cup \{\langle p, q\rangle\}$ *alors* (P_{i+1}, C_{i+1}) *est un réseau.*

Question II.6 *Montrer que : si* (P_k, C_k) *est un réseau construit par l'algorithme précédent avec* $k = card(P)$, *alors ce réseau répond au problème* (P, C).

2.2 Préliminaire : Construction d'un cocycle

Soient un ensemble P de points et un ensemble C de connexions possibles entre les points de P, le cocycle d'un sous-ensemble Q de points de P est le sous-ensemble des connexions de C reliant un point de Q à un point n'appartenant pas à Q (donc appartenant à $P \setminus Q$). Nous noterons $\omega(Q)$ le cocycle de Q dans (P, C). La connexion ajoutée au réseau (P_i, C_i) sera donc prise dans $\omega(P_i)$.

Question II.7 *Écrire en PASCAL une fonction récursive* `cocycle(pts:POINTS; cxs:CONNEXIONS):CONNEXIONS;` *telle que l'appel* `cocycle( q, c)` *calcule le cocycle du sous-ensemble de points* `q` *de l'ensemble de points* `P` *reliés par les connexions de l'ensemble* `c`.

Question II.8 *Montrer que le calcul de la fonction* `cocycle` *se termine quelles que soient les valeurs de ses paramètres.*

Question II.9 *Donner une estimation de la complexité de la fonction* `cocycle` *en nombre d'appels récursifs à* `cocycle`*. Justifier cette estimation.*

2.3 Algorithme de construction du réseau

Question II.10 *Appliquer* « à la main » *l'algorithme précédent sur l'exemple II.2. Utiliser pour cela la représentation graphique.*

Question II.11 *Écrire une fonction PASCAL* `construire(pts:POINTS; cxs:CONNEXIONS):RESEAU;`*, appliquant l'algorithme précédent, telle que l'appel* `construire( p, c)` *construit un réseau pour le problème connexe qui consiste à relier l'ensemble de points* `p` *en choisissant les connexions parmi l'ensemble* `c`*. Cette fonction fera appel à des fonctions récursives.*

Question II.12 *Montrer que le calcul de la fonction* `construire` *se termine quelles que soient les valeurs de ses paramètres.*

Question II.13 *Donner une estimation de la complexité de la fonction* `construire` *en fonction de la complexité de* `cocycle` *et du nombre d'appels récursifs internes à* `construire`*. Justifier cette estimation.*

3 Construction d'un réseau de coût minimal

À partir d'un ensemble donné de connexions réalisables, il est généralement possible de construire plusieurs réseaux distincts répondant au problème d'interconnexion. Les coûts de construction de ces réseaux sont en général différents. Il est donc nécessaire d'adapter l'algorithme précédent de manière à construire un réseau de coût de réalisation minimal. Pour cela, la connexion ajoutée au réseau partiel à chaque étape de l'algorithme ne sera plus choisie au hasard dans le cocycle mais sera la connexion de coût minimal parmi les connexions possibles du cocycle. L'algorithme obtenu est appelé « *algorithme de Prim* ».

3.1 Correction de l'algorithme de Prim

Soit (X, Y) un réseau partiel pour le problème (P, C).
Soit $H = (P, Z)$ un réseau de coût minimal répondant à ce même problème tel que $Y \subseteq Z$.
Soit une connexion $c' = \langle p_0, p_n \rangle \in \omega(X)$ telle que $c' \notin Z$.
On remarquera alors que $(P, \{c'\} \cup Z)$ contient au moins un cycle (application de la question II.3) noté $\langle\langle p_0, \cdots, p_n, p_0 \rangle\rangle$ et composé des connexions $\{\langle p_0, p_1 \rangle, \langle p_1, p_2 \rangle, \cdots, \langle p_{n-1}, p_n \rangle, c'\}$.

Question II.14 *Montrer que* $\omega(X) \cap \{\langle p_0, p_1 \rangle, \langle p_1, p_2 \rangle, \cdots, \langle p_{n-1}, p_n \rangle\} \neq \emptyset$.

Question II.15 *Soit c une connexion de* $\omega(X) \cap \{\langle p_0, p_1 \rangle, \langle p_1, p_2 \rangle, \cdots, \langle p_{n-1}, p_n \rangle\} \neq \emptyset$.
Montrer que $H' = (P, Z')$ *avec* $Z' = (Z \setminus \{c\}) \cup \{c'\}$ *est un réseau quelconque répondant au problème.*

Tournez la page S.V.P.

Question II.16 *Donner une relation liant les coûts de H' et de H.*

Question II.17 *Montrer que : si c' est une connexion de coût minimal dans $\omega(X)$ alors H' est un réseau de coût minimal.*

Question II.18 *Montrer que l'application de l'algorithme de Prim produit un réseau minimal pour un problème donné.*

3.2 Algorithme de Prim : Construction d'un réseau de coût minimal

On dispose de plus d'une fonction récursive PASCAL
`inserer(c:CONNEXION; cxs:CONNEXIONS):CONNEXIONS;` telle que l'appel `inserer( c, l)` sur une liste `l` ordonnée selon le coût des connexions, renvoie la liste `l` dans laquelle la connexion `c` a été insérée en respectant cet ordre. On ne demande pas d'écrire cette fonction, dont on admettra que la complexité est un $O(taille(l))$.

Question II.19 *Modifier la fonction `cocycle` de la question II.7 en utilisant la fonction `inserer` pour construire un cocycle ordonné.*

Question II.20 *Donner une estimation de la complexité dans le pire cas en fonction de la complexité de `inserer` et du nombre d'appels récursifs de la fonction `cocycle` modifiée. Justifier cette estimation.*

Question II.21 *Donner une estimation de la complexité dans le pire cas en fonction de la complexité de `cocycle` et du nombre d'appels récursifs internes à la fonction `construire` utilisant la nouvelle fonction `cocycle`. Justifier cette estimation.*

Partie III : Automates et langages

Le but de cet exercice est l'étude des propriétés de l'opération $\times$ calculant le produit de deux automates finis déterministes.

Soit l'alphabet X, un ensemble de symboles, soit Λ le symbole représentant le mot vide ($\Lambda \notin X$). un automate fini sur X est un quintuplet $A = (Q, X, i, T, \delta)$ composé de :

- Un ensemble d'états : Q,
- L'état initial : $i \in Q$,
- Un ensemble d'états terminaux : $T \subseteq Q$,
- Une fonction de transition : $\delta : X \times Q \to Q$

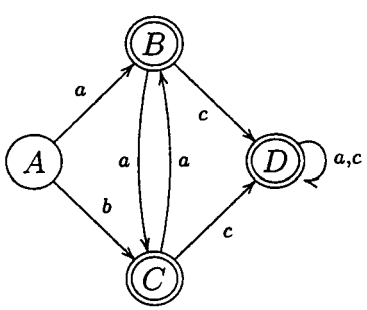
Les valeurs de la fonction de transition δ seront représentées par un graphe dont les nœuds sont les états. Un état initial sera entouré d'un cercle (i) et un état final sera entouré d'un double cercle $((t))$.

Le langage sur $X^\star$ reconnu par cet automate fini est :

$$L(A) = \{m \in X^\star \mid t \in T, \delta^\star(m, i) = t\} \text{ avec } \begin{array}{l} \delta^\star(\Lambda, q) = q \\ \delta^\star(x, q) = \delta(x, q) \\ \delta^\star(m.x, q) = \delta(x, \delta^\star(m, q)) \end{array}$$

Étude de l'exemple $\mathcal{E}_1$

Soit l'automate fini déterministe $\mathcal{E}_1 = (\{A, B, C, D\}, \{a, b, c\}, A, \{B, C, D\}, \delta_{\mathcal{E}_1})$ dont la fonction de transition est définie par :

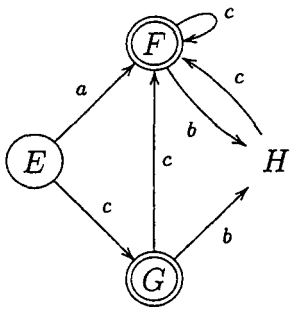


Question III.1 *Caractériser le langage reconnu par $\mathcal{E}_1$ par une expression régulière ou ensembliste.*

Tournez la page S.V.P.

Étude de l'automate $\mathcal{E}_2$

Soit l'automate fini déterministe $\mathcal{E}_2 = (\{E, F, G, H\}, \{a, b, c\}, E, \{F, G\}, \delta_{\mathcal{E}_2})$ dont la fonction de transition est définie par :



Question III.2 *Caractériser le langage reconnu par $\mathcal{E}_2$ par une expression régulière ou ensembliste.*

Construction de l'automate « produit » : $\mathcal{E}_1 \times \mathcal{E}_2$

Soit l'opération interne $\times$ sur les automates finis déterministes définie par :

Déf. III.1 (Produit d'automates) *Soient $A_1 = (Q_1, X, q_1, T_1, \delta_1)$ et $A_2 = (Q_2, X, q_2, T_2, \delta_2)$ deux automates finis déterministes, l'automate produit $A = A_1 \times A_2$ est défini par :*

$$A = (Q_1 \times Q_2, X, (q_1, q_2), T_1 \times T_2, \delta_{1\times 2})$$
$$\delta_{1\times 2}(a, (x, y)) = (x', y') \text{ si } \delta_1(a, x) = x' \text{ et } \delta_2(a, y) = y'$$

Question III.3 *Construire l'automate $\mathcal{E}_1 \times \mathcal{E}_2$.*

Question III.4 *Caractériser le langage reconnu par $\mathcal{E}_1 \times \mathcal{E}_2$ par une expression régulière ou ensembliste. Comparer ce langage avec les langages reconnus par $\mathcal{E}_1$ et $\mathcal{E}_2$.*

Étude de l'automate « produit »

Question III.5 *Montrer que $\delta^*_{1\times 2}(m, (q_1, q_2)) = (\delta^*_1(m, q_1), \delta^*_2(m, q_2))$*

Question III.6 *Montrer que : $m \in L(A_1 \times A_2) \Leftrightarrow (m \in L(A_1) \wedge m \in L(A_2))$.*

Question III.7 *Quelle relation liant les langages reconnus par A_1, A_2 et $A_1 \times A_2$ peut-on en déduire ?*

Fin de l'énoncé