INFORMATIQUE

L'épreuve est constituée de deux parties indépendantes. Le candidat peut les traiter dans l'ordre de son choix à condition de respecter les numérotations.

Partie I - Algorithmique

On appelle *graphe* un ensemble fini de points du plan (nommés nœuds). Certains de ces nœuds sont reliés par un arc orienté. Un graphe permet de représenter simplement une relation binaire définie sur un ensemble fini.

I.A - Affectation de candidats à des postes

Dans cette partie, on s'intéresse au problème de l'affectation de candidats à des postes ouverts par des écoles. Chaque candidat classe les écoles dans lesquelles il souhaite obtenir un poste par ordre de préférence strictement décroissante. Chaque école offre un nombre connu de postes, et classe tous les candidats qui postulent par ordre de préférence strictement décroissante. Les choix des candidats et des écoles peuvent être représentés par un graphe dans lequel chaque nœud représente une candidature : les nœuds du graphe sont sur une grille à deux dimensions, les candidats étant placés en abscisses et les écoles en ordonnées ; ainsi les arcs verticaux représentent la relation de préférence des candidats pour les écoles et les arcs horizontaux la relation de préférence des écoles pour les candidats. Ces relations sont des relations d'ordre : elle sont donc transitives.

I.B - Notations

On note $\langle C_i, E_j\rangle$ la candidature du candidat C_i à un poste ouvert par l'école E_j. On note P_c la relation de préférence des candidats pour les écoles, et P_e la relation de préférence des écoles pour les candidats. Ainsi $P_c(\langle C_i, E_j\rangle, \langle C_i, E_k\rangle)$, indique que le candidat C_i préfère l'école E_j à l'école E_k, et $P_e(\langle C_j, E_i\rangle, \langle C_k, E_i\rangle)$ indique que l'école E_i préfère le candidat C_j au candidat C_k. On note N_i le nombre de postes ouverts par l'école E_i.

Dans toute cette partie $[1, n]$ désigne l'ensemble $\{1, \ldots, n\}$.

Filière MP

Concours Centrale SupElec 2004

I.C - Exemple

Considérons le graphe ayant pour sommets :

$\langle C_1, E_2\rangle$, $\langle C_1, E_3\rangle$, $\langle C_2, E_1\rangle$, $\langle C_2, E_2\rangle$, $\langle C_2, E_3\rangle$,

$\langle C_3, E_2\rangle$, $\langle C_3, E_3\rangle$, $\langle C_4, E_1\rangle$, $\langle C_4, E_2\rangle$

pour arcs « verticaux » :

$P_c(\langle C_1, E_3\rangle, \langle C_1, E_2\rangle)$,

$P_c(\langle C_2, E_3\rangle, \langle C_2, E_2\rangle)$, $P_c(\langle C_2, E_2\rangle, \langle C_2, E_1\rangle)$,

$P_c(\langle C_3, E_2\rangle, \langle C_3, E_3\rangle)$,

$P_c(\langle C_4, E_1\rangle, \langle C_4, E_2\rangle)$

et pour arcs « horizontaux » :

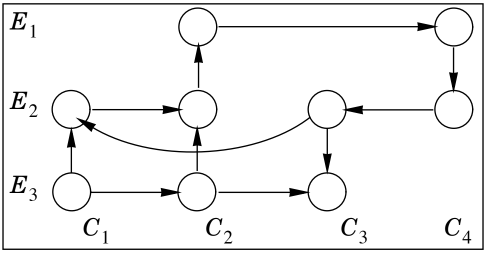
$P_e(\langle C_2, E_1\rangle, \langle C_4, E_1\rangle)$,

$P_e(\langle C_4, E_2\rangle, \langle C_3, E_2\rangle)$, $P_e(\langle C_3, E_2\rangle, \langle C_1, E_2\rangle)$, $P_e(\langle C_1, E_2\rangle, \langle C_2, E_2\rangle)$,

$P_e(\langle C_1, E_3\rangle, \langle C_2, E_3\rangle)$, $P_e(\langle C_2, E_3\rangle, \langle C_3, E_3\rangle)$

avec, comme nombres de postes ouverts, $N_1 = 1$, $N_2 = 2$ et $N_3 = 1$.

Ce graphe peut être représenté comme suit :



Ce graphe indique que le candidat C_1 postule pour les écoles E_2 et E_3, et qu'il préfère E_3 à E_2. De même, le candidat C_2 postule pour les 3 écoles et préfère E_3 à E_2 et E_2 à E_1 et donc, par transitivité, il préfère E_3 à E_1. Le candidat C_3 postule pour E_2 et E_3, dans cet ordre de préférence décroissante, et C_4 postule pour E_1 et E_2 dans cet ordre. L'école E_1 ouvre un seul poste, et elle préfère la candidature de C_2 à celle de C_4. L' école E_2 ouvre 2 postes, elle préfère C_4 à C_3, C_3 à C_1 et C_1 à C_2 ; par transitivité, elle préfère donc C_4 à C_1, C_4 à C_2 et C_3 à C_2. Enfin E_3 n'ouvre qu'un poste et préfère C_1 à C_2 qu'elle préfère à C_3.

I.D - Affectations méritoires

Une affectation $\mathscr{A}$ est un ensemble de nœuds tel que dans chaque colonne, au plus un nœud appartient à l'affectation (un candidat ne peut pas être affecté à plusieurs postes) et tel que sur chaque ligne, le nombre de nœuds appartenant à l'affectation est au plus égal au nombre de postes ouverts par l'école correspondante. Une affectation vérifie donc les propriétés suivantes :

$A1 \qquad \forall i, \quad \big(\langle C_i, E_j\rangle \in \mathscr{A} \text{ et } \langle C_i, E_k\rangle \in \mathscr{A} \Rightarrow j = k\big)$

$A2 \qquad \big(\forall j, \exists n > N_j \,;\, \forall k \in [1, n], \langle C_{i_k}, E_j\rangle \in \mathscr{A}\big) \Rightarrow \exists p, q \in [1, n] \;, \begin{cases} p \neq q \\ i_p = i_q \end{cases}.$

Une affectation est dite « totale » si tous les postes ouverts sont attribués, *ou* si tous les candidats obtiennent un poste (le nombre de postes ouverts et le nombre de candidats ne sont pas forcément égaux). Une affectation $\mathscr{A}$ est dite « méritoire » si et seulement si pour tout nœud $\langle C_i, E_j\rangle$ du graphe l'une des propositions suivantes est vraie :

$M1 \;\; \langle C_i, E_j\rangle \in \mathscr{A}$

$M2 \;\; \exists \langle C_i, E_k\rangle \in \mathscr{A}, k \neq j \text{ et } P_c(\langle C_i, E_k\rangle, \langle C_i, E_j\rangle)$

$M3 \;\; \exists n_1, \ldots, n_{N_j} \text{ distincts, } \forall k \in [1, N_j], \begin{cases} n_k \neq i \\ \langle C_{n_k}, E_j\rangle \in \mathscr{A} \\ P_e(\langle C_{n_k}, E_j\rangle, \langle C_i, E_j\rangle) \end{cases}$

l'accolade dans $M3$ signifiant que les 3 propriétés sont vraies simultanément.

I.D.1) Que signifie en langage courant la définition d'une affectation méritoire ?

I.D.2) Une affectation méritoire est-elle nécessairement totale ?

I.E - Nœuds inutiles pour les écoles

Dans cette section on cherche un algorithme conduisant à une affectation méritoire privilégiant les vœux des candidats en donnant à chaque candidat son choix préféré.

On appelle « nœud inutile pour les écoles » tout nœud $\langle C_i, E_j\rangle$ tel qu'il existe N_j nœuds distincts $\langle C_{n_1}, E_j\rangle \ldots \langle C_{n_{N_j}}, E_j\rangle$, avec $n_k \neq i$ pour tout k, qui vérifient les deux propriétés suivantes :

$$\forall k \in [1, N_j],\; P_c(\langle C_{n_k}, E_p\rangle, \langle C_{n_k}, E_j\rangle) \Rightarrow (p = j) \qquad (1)$$

$$\forall k \in [1, N_j],\; P_e(\langle C_{n_k}, E_j\rangle, \langle C_i, E_j\rangle) \qquad (2)$$

I.E.1) Montrer que les affectations méritoires d'un graphe sont exactement celles du graphe obtenu en supprimant les nœuds inutiles pour les écoles du graphe initial, à condition que, lors de la suppression des nœuds inutiles, on prenne garde de maintenir les chaînes des préférences concernant les noeuds restants.

I.E.2) Déduire de la question précédente un algorithme pour trouver une affectation méritoire.

I.E.3) Appliquer cet algorithme (pas à pas) au graphe donné en exemple.

On va maintenant s'intéresser à l'affectation qui privilégie les vœux des écoles.

I.F - Dualité candidat-école

I.F.1) Donner la définition d'un « nœud inutile pour les candidats ».

I.F.2) Montrer que les nœuds inutiles pour les candidats peuvent eux-aussi être supprimés du graphe sans que cela change les affectations méritoires.

I.F.3) En déduire un algorithme pour obtenir l'affectation méritoire qui privilégie le choix des écoles.

I.F.4) Appliquer cet algorithme au graphe donné en exemple.

I.G - Graphe réduit

I.G.1) Donner un algorithme permettant de supprimer tous les nœuds inutiles (aussi bien pour les écoles que pour les candidats) d'un graphe.

I.G.2) Appliquer cet algorithme au graphe donné en exemple, et en déduire toutes les affectations méritoires de ce graphe.

Mamouni My Ismail
Professeur agrégé de mathématiques
Enseignant en classes de MP
CPGE My Youssef
Rabat, Maroc
mamouni.myismail@gmail.com
myprepa.ifrance.com

Corrigé abrégé
Epreuve Informatique
Concours Centrale Sup Elec, 2004

```
STUDENT > restart:
```

On commence par déclarer les écoles, le nombre de postes ouverts pour chaque école

```
STUDENT > Ecoles:=[E1,E2,E3];N:=[1,2,1];candidats:=[C1,C2,C3,C4];
```

$$Ecoles := [E1, E2, E3]$$

$$N := [1, 2, 1]$$

$$candidats := [C1, C2, C3, C4]$$

Puis le classement de chaque école pour les candidats

```
STUDENT > E1:=[C2,C4];E2:=[C4,C3,C1,C2];E3:=[C1,C2,C3];
```

$$E1 := [C2, C4]$$

$$E2 := [C4, C3, C1, C2]$$

$$E3 := [C1, C2, C3]$$

Les voeux de chaque candidat classés dans l'ordre de préference, la 1ére case répresente un numéro pour idéntifier le candidat

```
STUDENT > C1:=[1,3,2];C2:=[2,3,2,1];C3:=[3,2,3];C4:=[4,1,2];
```

$$C1 := [1, 3, 2]$$

$$C2 := [2, 3, 2, 1]$$

$$C3 := [3, 2, 3]$$

$$C4 := [4, 1, 2]$$

Pour chaque école, on va chercher les candidats à éliminer, en commençant bien sûr par le dernier classée pour cette même école. Pour cela on dénombre les candidats les mieux classés pour lesquels cette école représente le 1ér choix, si ce nombre dépasse le nombre de poste ouverts par la dite école, alors notre candidat est élinminé.

```
STUDENT > choix_ecole:=proc(i) local
          Ecole,j,Nbr,k,candidat,a,b,NewEcole;
STUDENT > Ecole:=op(i,Ecoles):NewEcole:=NULL:
STUDENT > for j from nops(Ecole) to 1 by -1 do
STUDENT > Nbr:=0;
STUDENT > for k from j-1 to 1 by -1 do
STUDENT > candidat:=op(k,Ecole):
STUDENT > if op(2,candidat)=i then Nbr:=Nbr+1:
STUDENT > else
```

```
STUDENT > fi:
STUDENT > od:
STUDENT > if Nbr>=op(i,N) then print(`Le candidat eliminé est
          numéro`=op(1,op(j,Ecole)));
STUDENT > else print(`Le candidat non eliminé
          `=op(1,op(j,Ecole)));NewEcole:=NewEcole,C[op(1,op(j,Ecol
          e))]:
STUDENT > fi:
STUDENT > od:
STUDENT > print(`Le choix de l'école est
          `=[NewEcole]);RETURN([NewEcole]);
STUDENT > end:
```

On applique notre petit programme pour le 1ére école

```
STUDENT > E[1]:=choix_ecole(1);
```

$$\textit{Le candidat non eliminé } = 4$$

$$\textit{Le candidat non eliminé } = 2$$

$$\textit{Le choix de l'école est } = [C_4, C_2]$$

$$E_1 := [C_4, C_2]$$

On applique notre petit programme pour le 2éme école

```
STUDENT > E[2]:=choix_ecole(2);
```

$$\textit{Le candidat non eliminé } = 2$$

$$\textit{Le candidat non eliminé } = 1$$

$$\textit{Le candidat non eliminé } = 3$$

$$\textit{Le candidat non eliminé } = 4$$

$$\textit{Le choix de l'école est } = [C_2, C_1, C_3, C_4]$$

$$E_2 := [C_2, C_1, C_3, C_4]$$

On applique notre petit programme pour le 3éme école

```
STUDENT > E[3]:=choix_ecole(3);
```

$$\textit{Le candidat eliminé est numéro} = 3$$

$$\textit{Le candidat eliminé est numéro} = 2$$

$$\textit{Le candidat non eliminé } = 1$$

$$\textit{Le choix de l'école est } = [C_1]$$

$$E_3 := [C_1]$$

Maintenant chaque candidat va éliminer toute école qu'elle juge inutile, c'est à dire pour la quelle il est sûr d'avoir mieux. Comment? il commence par la derniére et il vérifie si une école mieux classée va le prendre dans sa liste prinicipale.

```
STUDENT > for i from 1 to nops(Ecoles) do
STUDENT > liste_principale[i]:=[seq(op(j,E[i]),j=1..N[i])];
STUDENT > od;
```

$$\textit{liste\_principale}_1 := [C_4]$$

$$liste\_principale_2 := [C_2, C_1]$$

$$liste\_principale_3 := [C_1]$$

STUDENT >

On s'inspirant de cet exemple on inverse les role écoles-candidats pour trouver les écoles à éliminer pour chaque candidat....A vous de le faire

Mamouni My Ismail
Professeur agrégé de mathématiques
Enseignant en classes de MP
CPGE My Youssef
Rabat, Maroc
mamouni.myismail@gmail.com
myprepa.ifrance.com

ECOLE POLYTECHNIQUE
ECOLE SUPERIEURE DE PHYSIQUE ET CHIMIE INDUSTRIELLES

CONCOURS 2003 **FILIERE MP-OPTION SCIENCES INDUSTRIELLES**

FILIERE PC

EPREUVE FACULTATIVE D'INFORMATIQUE

***

l'enclos du robot

Frontière Sud-Ouest

```
STUDENT > restart;
```

Question 1.

```
STUDENT > sudouest:=proc(P,Q) local x,y;
STUDENT > x:=[P[1],Q[1]]:y:=[P[2],Q[2]]:
STUDENT > if x[1]<=x[2] and y[1]<=y[2] then 1
STUDENT > else 0
STUDENT > fi;
STUDENT > end:
STUDENT > nordouest:=proc(P,Q) local x,y;
STUDENT > x:=[P[1],Q[1]]:y:=[P[2],Q[2]]:
STUDENT > if x[1]<=x[2] and y[1]>=y[2] then 1
STUDENT > else 0
STUDENT > fi;
STUDENT > end:
STUDENT > sudest:=proc(P,Q) local x,y;
STUDENT > x:=[P[1],Q[1]]:y:=[P[2],Q[2]]:
STUDENT > if x[1]>=x[2] and y[1]<=y[2] then 1
STUDENT > else 0
STUDENT > fi;
STUDENT > end:
STUDENT > nordest:=proc(P,Q) local x,y;
STUDENT > x:=[P[1],Q[1]]:y:=[P[2],Q[2]]:
STUDENT > if x[1]>=x[2] and y[1]>=y[2] then 1
STUDENT > else 0
STUDENT > fi;
STUDENT > end:
```

Question 2

```
STUDENT > echange:=proc(a,b,i,j) local p,q,a1,b1;
```

```
STUDENT > p:=min(i,j);q:=max(i,j);
STUDENT > if p=1 then
STUDENT >      if q=nops(a) then
          a1:=[a[q],seq(a[k],k=2..nops(a)-1),a[1]];
STUDENT >
          b1:=[b[q],seq(b[k],k=2..nops(b)-1),b[1]];
STUDENT >      else
          a1:=[a[q],seq(a[k],k=2..q-1),a[1],seq(a[k],k=q+1..nops(a
          ))];
STUDENT >
          b1:=[b[q],seq(b[k],k=2..q-1),b[1],seq(b[k],k=q+1..nops(b
          ))];
STUDENT >      fi;
STUDENT > else
STUDENT >      if q=nops(a) then
          a1:=[seq(a[k],k=1..p-1),a[q],seq(a[k],k=p+1..q-1),a[p]];
          b1:=[seq(b[k],i=1..p-1),b[q],seq(b[k],k=p+1..q-1),b[p]];
STUDENT >      else
STUDENT > a1:=[seq(a[k],k=1..p-1),a[q],seq(a[k],k=p+1..q-1),a[p],s
          eq(a[k],k=q+1..nops(a))];
          b1:=[seq(b[k],k=1..p-1),b[q],seq(b[k],k=p+1..q-1),b[p],s
          eq(b[k],k=q+1..nops(b))];     fi;
STUDENT > fi;
STUDENT > RETURN([a1,b1]);
STUDENT > end:
Warning, `k` in call to `seq` is not local
Warning, `k` in call to `seq` is not local
Warning, `k` in call to `seq` is not local
Warning, `k` in call to `seq` is not local
Warning, `k` in call to `seq` is not local
Warning, `k` in call to `seq` is not local
Warning, `k` in call to `seq` is not local
Warning, `k` in call to `seq` is not local
Warning, `i` in call to `seq` is not local
Warning, `k` in call to `seq` is not local
Warning, `k` in call to `seq` is not local
Warning, `k` in call to `seq` is not local
Warning, `k` in call to `seq` is not local
Warning, `k` in call to `seq` is not local
Warning, `k` in call to `seq` is not local
Warning, `k` in call to `seq` is not local
```

Question 3 *c'est le point P1*

Question 4

Ecrivons d'abord la fonction testSO qui retourne 0 si un point donné est dans la frontière SudOuest, et une valeur non nulle dans le cas contraire

```
STUDENT > testSO:=proc(a,b,i) local a1,b1,N,j;
STUDENT > a1:=op(1,echange(a,b,1,i));
STUDENT > b1:=op(2,echange(a,b,1,i));
STUDENT > N:=0:for j from 2 to nops(a) do
          N:=N+sudouest([a1[j],b1[j]],[a1[1],b1[1]]);
STUDENT > od;
```

```
STUDENT > RETURN(N);
STUDENT > end:
```

Terminons enfin par récupérer les points qui se trouvent sur la frontière SudOuest, c'est à dire pour lesquels la valeur retournée par testSO est 0

```
STUDENT > frontiereSO:=proc(a,b) local aSO,bSO,i; global nSO;
STUDENT > aSO:=NULL:
STUDENT > bSO:=NULL:
STUDENT > for i from 1  to nops(a) do
STUDENT > if testSO(a,b,i)=0 then aSO:=aSO,a[i];bSO:=bSO,b[i];
STUDENT > else fi;
STUDENT > od;
STUDENT > nSO:=nops([aSO]);
STUDENT > RETURN(seq([op(i,aSO),op(i,bSO)],i=1..nSO));
STUDENT > end:
```

Question 5

```
STUDENT > testNO:=proc(a,b,i) local a1,b1,N,j;
STUDENT > a1:=op(1,echange(a,b,1,i));
STUDENT > b1:=op(2,echange(a,b,1,i));
STUDENT > N:=0:for j from 2 to nops(a) do
          N:=N+nordouest([a1[j],b1[j]],[a1[1],b1[1]]);
STUDENT > od;
STUDENT > RETURN(N);
STUDENT > end:
STUDENT > frontiereNO:=proc(a,b) local aNO,bNO,i; global nNO;
STUDENT > aNO:=NULL:
STUDENT > bNO:=NULL:
STUDENT > for i from 1  to nops(a) do
STUDENT > if testNO(a,b,i)=0 then aNO:=aNO,a[i];bNO:=bNO,b[i];
STUDENT > else fi;
STUDENT > od;
STUDENT > nNO:=nops([aNO]);
STUDENT > RETURN(seq([op(i,aNO),op(i,bNO)],i=1..nNO));
STUDENT > end:
```

Question 6

```
STUDENT > testSE:=proc(a,b,i) local a1,b1,N,j;
STUDENT > a1:=op(1,echange(a,b,1,i));
STUDENT > b1:=op(2,echange(a,b,1,i));
STUDENT > N:=0:for j from 2 to nops(a) do
          N:=N+sudest([a1[j],b1[j]],[a1[1],b1[1]]);
STUDENT > od;
STUDENT > RETURN(N);
STUDENT > end:
STUDENT > frontiereSE:=proc(a,b) local aSE,bSE,i; global nSE;
STUDENT > aSE:=NULL:
STUDENT > bSE:=NULL:
STUDENT > for i from 1  to nops(a) do
STUDENT > if testSE(a,b,i)=0 then aSE:=aSE,a[i];bSE:=bSE,b[i];
```

```
STUDENT > else fi;
STUDENT > od;
STUDENT > nSE:=nops([aSE]);
STUDENT > RETURN(seq([aSE[i],bSE[i]],i=1..nSE));
STUDENT > end:
STUDENT > testNE:=proc(a,b,i) local a1,b1,N,j;
STUDENT > a1:=op(1,echange(a,b,1,i));
STUDENT > b1:=op(2,echange(a,b,1,i));
STUDENT > N:=0:for j from 2 to nops(a) do
          N:=N+nordest([a1[j],b1[j]],[a1[1],b1[1]]);
STUDENT > od;
STUDENT > RETURN(N);
STUDENT > end:
STUDENT > frontiereNE:=proc(a,b) local aNE,bNE,i; global nNE;
STUDENT > aNE:=NULL:
STUDENT > bNE:=NULL:
STUDENT > for i from 1  to nops(a) do
STUDENT > if testNE(a,b,i)=0 then aNE:=aNE,a[i];bNE:=bNE,b[i];
STUDENT > else fi;
STUDENT > od;
STUDENT > nNE:=nops([aNE]);
STUDENT > RETURN(seq([aNE[i],bNE[i]],i=1..nNE));
STUDENT > end:
STUDENT > frontiereSO(a,b);
```

$$[1, 1]$$

```
STUDENT > with(plots):
```

Question 7

On définit maintenant la fonction SO qui permet de tracer la frontière SudOuest en joignant ses point à l'aide de la fonction plot

```
STUDENT > tracer:=proc(P) local Pts,i;
STUDENT > Pts:=NULL:
STUDENT > for i from 1 to nops(P)-1 do
STUDENT > Pts:=Pts,P[i],[P[i][1],P[i+1][2]];
STUDENT > od;
STUDENT > Pts:=Pts,P[nops(P)];
STUDENT > end:
STUDENT > P:=[[0,11],[2,0],[2,1],[3,8],[3,7],[4,8],[4,2],[4,0],[5,
          3],[5,6],[6,9],[6,12],[6,11],[7,9],[9,3],[9,11],[10,8],[
          10,6],[10,10],[11,1]];a:=[seq(P[i][1],i=1..nops(P))];b:=
          [seq(P[i][2],i=1..nops(P))];
```

$$P := [[0, 11], [2, 0], [2, 1], [3, 8], [3, 7], [4, 8], [4, 2], [4, 0], [5, 3], [5, 6], [6, 9], [6, 12], [6, 11], [7, 9], [9, 3], [9, 11], [10, 8], [10, 6], [10, 10], [11, 1]]$$

$$a := [0, 2, 2, 3, 3, 4, 4, 4, 5, 5, 6, 6, 6, 7, 9, 9, 10, 10, 10, 11]$$

$$b := [11, 0, 1, 8, 7, 8, 2, 0, 3, 6, 9, 12, 11, 9, 3, 11, 8, 6, 10, 1]$$

```
STUDENT > P1:=[frontiereNO(a,b),frontiereNE(a,b),frontiereSE(a,b),
          frontiereSO(a,b),frontiereNO(a,b)];
```

$P1 := [[6, 12], [6, 12], [9, 11], [10, 10], [11, 1], [4, 0], [11, 1], [2, 0], [6, 12]]$

```
STUDENT > plot([tracer(P1)]);
```