ÉCOLE POLYTECHNIQUE - PSI/PT - ÉPREUVE FACULTATIVE D'INFORMATIQUE 2004

Corrigé rédigé par Alain Schauber - alain.schauber@prepas.org

Version de MAPLE : Classic Worksheet Maple 10.
Pour satisfaire à la convention des tableaux et des matrices à partir de 0, on utilisera dans ce corrigé la fonction Maple **array**.

Cercles d'amis

La relation d'amitié indirecte définit clairement une relation d'équivalence sur C, et il s'agit ici de modéliser la partition qu'elle définit à l'aide d'un tableau.
Dans tout le problème on admettra, ce qui semble sous-entendu, que les liens d'amitié réflexive ne sont pas cités, et par conséquent lorsqu'on écrit ci ~ cj on a i différent de j.

Question 1
Si **s** = 0, c'est que aucun lien d'amitié n'est défini. Les cercles d'amis sont donc les **n** singletons de C. Il en résulte que **E** = <0,1,2,...,n-1>.
Dans le cas **s** = 1 et ci ~ cj avec i < j, on a les deux tableaux possibles <0,1,..,j-1,i,j+1,..,n-1> et <0,1,..,i-1,j,i+1,..,n-1> selon l'élu du cercle {ci,cj}.

Question 2
La recherche de **l** nécessite dans le pire des cas le parcours complet du tableau **E**, donc s'effectue bien en temps linéaire en **n**.

```
> representant:=proc(E,i) local l;
    l:=i;
    while E[l] <> l do l:=E[l] od;
    l
  end;
```

Question 3
Il suffit de déterminer séparément les représentants cl et cl' des cercles d'amis respectifs de ci et de cj. ci et c j sont alors amis indirects si et seulement si l = l'.
La fonction **sontAmis** ainsi définie reste de complexité temporelle linéaire en **n**.

```
> sontAmis:=(E,i,j)->evalb(representant(E,i)=representant(E,j));
```

Question 4
On détemine d'abord séparément les représentants cl et cl' des cercles d'amis respectifs de $c_{i_{s+1}}$ et de $c_{j_{s+1}}$, puis on modifie le contenu de l'une des deux cellules d'indice l et l' en la faisant pointer vers l'autre. Cette dernière opération s'effectuant en temps constant, on a comme à la question 3 une fonction linéaire en **n**.

```
> ajout:=proc(E,i,j) E[representant(E,i)]:=representant(E,j); RETURN() end;
```

Question 5
Prenons comme exemple le tableau **E**=<0,2,3,..,n-1,n-1>, correspondant à la partition de C en deux cercles d'amis {c0} et {c1,..c(n-1)}, la deuxième représentée par c(n-1).
Supposons que l'on ajoute une relation d'amitié entre c0 et une autre personne quelconque (amitié indirecte universelle):
- si on fait pointer c0 vers c(n-1) (**E** -> <n-1,2,3,..,n-1,n-1>), on incrémente d'une unité le temps d'exécution de l'appel à **representant**(**E**,0), tout en laissant invariants les temps d'exécution des autres appels à cette fonction.
- si on fait pointer c(n-1) vers c0 (**E** -> <0,2,3,..,n-1,0>), on incrémente d'une unité le temps d'exécution de tous les appels à **representant**(**E**,i) pour i > 0, en ne laissant inchangé que le temps d'exécution de l'appel à **representant**(**E**,0).

Par conséquent, bien que restant dans les deux cas de complexité linéaire en **n**, le temps d'exécution moyen de la fonction **representant** est asyptotiquement inchangé dans le premier cas et incrémenté d'une unité dans le second.

Une façon de garantir un meilleur temps d'exécution serait de calculer le représentant cl du cercle d'amis de ci, puis de modifier à l'aide d'une boucle successivement toutes les relations d'amitié des personnes cj, cj1,...,cj_r,cl' (avec cj ~ cj1 ~ ... ~ cj_r ~cl' et cl' représentant du cercle d'amis de cj) pour qu'elles pointent vers cl.
En effet, cela raccourcit le temps d'exécution des appels à **representant**(E,j),..,**representant**(E,j_r), augmente d'une unité l'appel à **representant**(E,l), ce qui est de toutes façons inévitable et cela laisse inchangés les temps d'exécution des autres appels à la fonction **representant**.
n somme, par cette méthode, le temps d'exécution moyen de l'appel à la fonction **representant** reste asymptotiquement inchangé.

Coloriage d'objets

Question 6

```
> couleurNO:=(T,i,j)->if i = 0 or j = 0 then 0 else T[i-1,j-1] fi;
  couleurN:=(T,i,j)->if i = 0 then 0 else T[i-1,j] fi;
  couleurO:=(T,i,j)->if j = 0 then 0 else T[i,j-1] fi;
```

Question 7
Il suffit d'écrire l'algorithme directeur de la fonction en suivant les instructions de l'énoncé.
Cette fonction est constituée de deux boucles imbriquées de longueur **R** chacune. Chaque tour de boucle de la boucle interne s'exécute en temps constant, sauf si le cas examiné amène à ajouter une équation (cas **else**). Dans ce cas-là, il est fait appel à la fonction **ajout**, dont le temps d'exécution est linéaire en **C**, comme on l'a vu à la question 4. Or la constante **C** est un majorant du nombre de couleurs nécessaire à la réalisation de l'algorithme, donc $\mathbf{C^2}$ est un majorant du nombre d'équations ainsi ajoutées.
On en déduit que le temps d'exécution de la fonction **marquage** est un O($\mathbf{R^2}$)+O($\mathbf{C^2}$).
NB : le nombre **p** d'objets dans la figure étant en réalité égal au nombre de classes d'équivalence modulo l'équivalence par équation des couleurs, on a en général **p** << **nc** et il faut donc choisir **C** >= **nc** en conséquence. Par exemple, dans l'image proposée dans l'énoncé, on a **p** = 5 et **nc** = 7.

```
> marquage:=proc(T)
    local i,j; global nc;
    for i from 0 to R-1 do
      for j from 0 to R-1 do
        if T[i,j] = 0 then next
        elif couleurNO(T,i,j) <> 0 then T[i,j]:=couleurNO(T,i,j)
        elif couleurO(T,i,j) <> 0 and couleurN(T,i,j) = 0 then
  T[i,j]:=couleurO(T,i,j)
        elif couleurO(T,i,j) = 0 and couleurN(T,i,j) <> 0 then
  T[i,j]:=couleurN(T,i,j)
        elif couleurO(T,i,j) = 0 and couleurN(T,i,j) = 0 then nc:=nc+1;
  T[i,j]:=nc
        elif couleurO(T,i,j) = couleurN(T,i,j) and couleurN(T,i,j) <> 0 then
  T[i,j]:=couleurN(T,i,j)
        else T[i,j]:=couleurO(T,i,j); ajout(E,couleurO(T,i,j),couleurN(T,i,j))
        fi
      od
    od;
    RETURN()
  end;
```

Question 8

On se sert du tableau **E** pour remplacer au cours du parcours de **T** chaque pixel par son représentant dans son "cercle d'amis" dans **E**.

On écrit donc de nouveau deux boucles imbriquées de longueur **R**, avec appel à à chaque tour à la fonction **representant** , qui est linéaire en **C**.

On en déduit un temps de calcul en $O(\mathbf{CR^2})$.

On peut penser qu'en ne faisant pas appel à la fonction **representant** pour les pixels à 0 on va gagner du temps, mais c'est illusoire car de toute manière 0 n'est en équation qu'avec lui-même, de telle sorte que l'appel à **representant**(**E**,0) se fait de toutes façons en temps constant. En revanche, on gagnerait sans doute significativement du temps en calculant avant la diffusion la projection canonique de l'ensemble {1,2,..,**nc**} dans l'ensemble de ses **p** classes d'équivalence, numérotées par leur représentant de plus petite valeur (par exemple sous la forme d'un tableau de couples associés de longueur **nc**).

C'est donc au sein de la fonction **colorier** qu'on effectue ce travail. On parcourt le tableau **E** de gauche à droite, de la manière suivante:

Si la cellule courante pointe vers une cellule antérieure, c'est qu'elle est équivalente à une couleur déjà représentée par un numéro antérieur.

Sinon il s'agit d'une nouvelle couleur, donc d'un nouvel objet, et on incrémente le compteur d'objet **p**. On fait alors une recherche itérative de toutes les couleurs amies d'indice supérieur ou égal, qu'on fait pointer vers le plus petit indice disponible dans **E**, c'est-à-dire **p**. Il ne reste plus alors qu'à mettre à jour le coloriage de l'image **T** avec les nouveaux numéros de couleur.

```
> diffusion:=proc(T) local i,j;
    for i from 0 to R-1 do
      for j from 0 to R-1 do
        T[i,j]:=representant(E,T[i,j]) # le cas T[i,j]=0 pourrait être réglé
  directement
      od
    od;
    RETURN()
  end;
> colorier:=proc(T)
    global nc,E; local k,m,c,i,j,p;
    k:=1;
    p:=0;
    while k <= nc do
      if E[k] >= k then
        p:=p+1;
        m:=k;
        while E[m] > k do c:=E[m]; E[m]:=p; m:=c od;
        E[k]:=p
      fi;
      k:=k+1
    od;
    for i from 0 to R-1 do
      for j from 0 to R-1 do
        T[i,j]:=E[T[i,j]]
      od
    od;
    RETURN()
  end;
```