GU : Licence Sciences Technologies (semestre 5) **UE :** INF209
Épreuve de : Algorithmique Avancée
Date : 2007 **Durée :** 3h
Documents autorisés
Épreuve de Monsieur Robson

1 [4 points]

Ecrire une fonction ou procédure en exalgo qui teste si un arbre binaire possède la propriété "équilibré par poids".
(Un arbre binaire est équilibré par poids si le rapport entre les tailles des deux sous-arbres à chaque noeud est au plus 2, sauf qu'il est permis qu'un noeud ait deux sous-arbres de taille 0 et 1.)
Réponse

```
fonction eqparpoids(val x:p): entier;
/* retourne -1 si l'arbre n'est pas équilibré; sinon sa taille */
var tailleg, tailled: entier;
debut
si x=NUL alors retourne 0 finsi;
/* il faut tester x=NUL avant de x^ pour éviter une erreur de segmentation */
tailleg=eqparpoids(x^.gauche);
si tailleg=-1 alors retourne -1 finsi;
tailled=eqparpoids(x^.droit);
si tailled=-1 alors retourne -1 finsi;
si tailled+tailleg=1 alors retourne 2 finsi;
/* (0,1) ou (1,0) est valable */
si tailled>tailleg*2 ou tailleg>tailled*2
   alors retourne -1
   sinon retourne tailled+tailleg+1
finsi
finfonction;
```

2 [6 points]

On veut calculer la plus grande somme des éléments de tous les intervalles d'un tableau, c'est-à-dire la valeur maximum de $T[i] + \cdots + T[j]$ pour $i \leq j$. (Il faut noter que des éléments du tableau peuvent être négatifs.) On remarque que cette somme est le maximum pour tout j de $V(j)$, définie comme la valeur de la plus grande somme d'un intervalle se terminant à T[j].

i. Donner des instructions exalgo pour calculer toutes les valeurs $V(j)$ par programmation dynamique

ii. et le calcul final de la somme maximum recherchée.

Réponse

```
procedure calcV (val T: TAB; ref V:TAB)
debut
V[1]=T[1]
pour i allant de 2 a 100 par pas de 1 faire
```

```
   si V[i-1]<0
      alors V[i]=T[i] /* intervalle d'un seul élément */
      sinon V[i]=V[i-1]+T[i]
   finsi
finpour
finprocedure;

fonction finir(val V: TAB): entier
var max: entier
debut
max=V[1];
pour i allant de 2 a 100 par pas de 1 faire
   si V[i]>max alors max=V[i]
finpour;
retourne max
finfonction;
```

3 [5 points]

Un tableau `T[1..n]` représente une "involution" de $\{1..n\}$ si pour chaque i en $\{1..n\}$ il existe un j en $\{1..n\}$, tel que $T[i] = j$ et $T[j] = i$.

i. Donner des instructions utilisant une boucle `tant que` pour vérifier qu'un tableau représente une involution.

ii. Donner un invariant qui suffirait pour démontrer que votre boucle soit correcte.

Réponse

```
debut
i=1; ok=vrai;
tant que i<=n et ok faire
   si T[i]<1 ou T[i]>n ou T[T[i]]!=i
      alors ok=faux
      sinon i:=i+1
   finsi
fintantque
/* postcondition: i>n si et seulement si T est une involution */
fin.

/* invariant: i>=1 et pour tout j en [1..i-1],
   1<=T[j]<=n et T[T[j]]=j
   et si ok=faux, i<=n et T n'est pas une involution*/
```

4 [5 points]

Déterminer (en fonction de n à $O(\ )$ près) le temps de calcul des boucles (a), (b), (c) (d), et (e)
(`P()` est une procédure dont le temps de calcul est constant.)

i. `pour i allant de 1 a n*n*n par pas de 1 faire P(); finpour;`
Réponse : $O(n^3)$

ii.
```
pour i allant de 1 a n par pas de 1 faire
   pour j allant de i a n par pas de 1 faire
      pour k allant de j à n par pas de 1 faire P(); finpour
```

```
   finpour
finpour;
```

Réponse : $O(n^3)$

iii.
```
i:=1;
tant que i<n-1 faire
   i:=i+(n-i)/2
fintantque;
```

Réponse : $O(\log n)$ (différence $n - i$ divisée par 2 à chaque itération)

iv.
```
j:=1;
pour i allant de 1 a n par pas de 1 faire
   tant que j<i*i faire
      j:=j+1
   fintantque
finpour;
```

Réponse : $O(n^2)$ (à la fin $j = n^2$)

v.
```
m:=1;
i:=1;
pour k allant de 1 a n par pas de 1 faire
   m:=m*k;
   pour j allant de 1 a m par pas de 1 faire P() finpour
finpour;
```

Réponse : $O(n!)$