TD
N° 3

LST–A (S3)
2009-2010
Algorithmes et Programmation Impérative 2

12 octobre 2009

Exercice 1.

Question 1.1. *Donner le temps de calcul de l'instruction* `B` *suivante :*

```
1  for i:=1 to n do begin
2    for j:=1 to i do A ;
3  end {for} ;
```

Exercice 2.

L'algorithme suivant permet de vérifier si un élément `x` appartient ou pas à un tableau `A[1..n]` où $n \geqslant 1$.

```
1 Algo(A,x)
2   trouve := false ;
3   i := low(A) ;
4   while not trouve and (i <= high(A) do begin
5     if A[i] = x then trouve := true
6     else i := i + 1 ;
7   end {while} ;
8   return trouve ;
```

On évalue le coût de cet algorithme en comptant le nombre d'évaluations de la condition `A[i] = x` en fonction de `n = high(A)-low(A)+1` (la taille du tableau). On note $T(n)$ cette complexité.

Question 2.1. *Quelle est la complexité dans le meilleur cas ? (l'élément* `x` *se trouve dans la première case du tableau* `A`*).*

Question 2.2. *Quelle est la complexité dans le pire cas ? (l'élément* `x` *se trouve au mieux dans la dernière case du tableau* `A`*).*

On va calculer la *complexité moyenne* dans le cas où les valeurs du tableau sont prises entre 1 et un entier $p > 1$. On suppose que les éléments du tableau ainsi que x sont des entiers appartenant à l'intervalle $[1, p]$ de $\mathbb{N}$. On va montrer que

$$T(n) \quad = \quad p - \frac{(p-1)^n}{p^{n-1}} \tag{1}$$

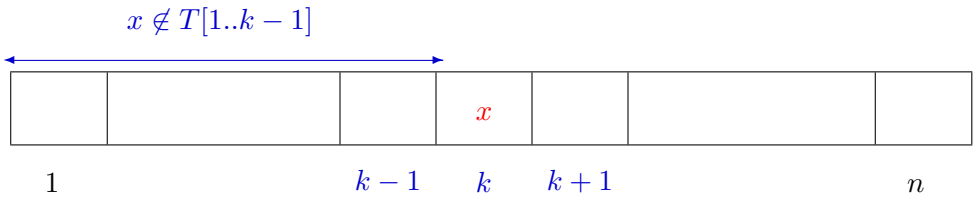
Le tableau est de taille n et les valeurs sont dans $[1, p]$. Il y a donc p^n tableaux différents.

Question 2.3. *Combien de tableaux, parmi les p^n, vont avoir leurs premières $n-1$ cases différentes de x et la dernière égale ou différente aussi de x ? Quelle est la complexité de l'algorithme pour ces tableaux.*

Soit maintenant $k \in [1, n-1]$. Le nombre de tableaux pour lesquels les $k-1$ premiers éléments sont différents de x, le $k-$ième égal à x et les $n-k$ derniers éléments quelconques dans $[1, p]$ est

$$(p-1)^{k-1} \cdot p^{n-k} \tag{2}$$

car il y a $(p-1)^{k-1}$ façons de choisir les éléments différents de x des $k-1$ premières cases et il y a p^{n-k} façons de choisir les éléments des $n-k$ dernières cases parmi les valeurs de $[1, p]$.



Question 2.4. *Quelle est la complexité de l'algorithme pour ces tableaux ?*

Question 2.5. *Montrer par récurrence sur n que*

$$p(p-1)^{n-1} + \sum_{k=1}^{n-1}(p-1)^{k-1}p^{n-k} \quad = \quad p^n \tag{3}$$

La complexité moyenne est la somme des complexités dans les différents cas divisée par le nombre de cas. Le coût moyen est donc

$$T(n) \quad = \quad \frac{1}{p^n}(n \cdot p(p-1)^{n-1} + \sum_{k=1}^{n-1} k \cdot (p-1)^{k-1}p^{n-k})$$

On pose

$$S(n) \quad = \quad \sum_{k=1}^{n-1} k \cdot (p-1)^{k-1}p^{n-k} \quad = \quad p^{n-1}\sum_{k=1}^{n-1} k \cdot (\frac{p-1}{p})^{k-1}$$

Question 2.6. *Montrer par récurrence sur n l'égalité :*

$$\sum_{k=1}^{n-1} kx^{k-1} \quad = \quad \frac{(n-1)x^n - nx^{n-1} + 1}{(x-1)^2}$$

Question 2.7. *Utiliser cette formule pour montrer que*

$$S(n) \quad = \quad p\big((n-1)(p-1)^n - np(p-1)^{n-1} + p^n\big)$$

Question 2.8. *En déduire que*

$$T(n) \quad = \quad p - \frac{(p-1)^n}{p^{n-1}}$$

Application : On récupère un paquet de 1000 copies d'un partiel noté en points entiers entre 0 et 20. On se demande s'il y a un étudiant qui a obtenu 20.

- Avec l'algorithme ci-dessus, en moyenne (avec $p = 21$ notes possibles, et $n = 1000$ taille du tableau) l'examen des 21 premières copies permettra de conclure.
- En posant la question au correcteur, on a la réponse immédiatement ;-)

Exercice 3.

Question 3.1. *Quelle est la complexité spatiale de l'algorithme ci-dessus ?*

Question 3.2. *Déterminer les 3 complexités en temps de l'algorithme suivant :*

```
1  trouve := false ;
2  for i:=low(A) to high(A) do begin
3    if A[i] = x then trouve := true ;
4  end {for} ;
5  return trouve ;
```