Complexité

Florian Bourse

Notation de Landau :

Soient f et g deux fonctions de $\mathbb{N}$ dans $\mathbb{R}^+$. f est *dominée asymptotiquement* par g, $f = O(g)$ si et seulement si $\exists c \in \mathbb{R}^{+*}$, $\exists n_0 \in \mathbb{N}$ tels que $\forall n > n_0, f(n) \leq cg(n)$.
Soient f et g deux fonctions de $\mathbb{N}$ dans $\mathbb{R}^+$. f et g sont du *même ordre de grandeur asymptotique*, $f = \theta(g)$ si et seulement si $\exists c_1, c_2 \in \mathbb{R}^{+*}$, $\exists n_0 \in \mathbb{N}$ tels que $\forall n > n_0, c_2 g(n) \leq f(n) \leq c_1 g(n)$.

Question 1. On considère les fonctions suivantes :

C

```c
int f (int n) {
  int x = 0;
  for (int i = 1; i < n; i = i + 1) {
    for (int j = i; j < i + 5; j = j + 1) {
    x = x + j ;
    }
  }
  return x ;
}

int g (int n) {
  int y = 0;
  for (int i = 0; i < f(n ); i = i + 1) {
    y = y + i ;
  }
  return y ;
}
```

1. Calculer $c_f(n)$, le nombre d'additions effectuées par f, en fonction de n. Uiliser O ou θ pour qualifier au mieux $c_f(n)$.
2. Calculer $f(n)$, puis son ordre de grandeur.
3. Donner un ordre de grandeur $c_g(n)$, le nombre d'additions effectuées par g en fonction de n.
4. Donner en fonction de n un ordre de grandeur du résultat de $g(n)$.
5. Réécrire la fonction g pour diminuer sa complexité. Quelle est alors sa complexité ?

Question 2. Donner la complexité des fonctions suivantes :

```
let rec factoriel n =
  if n = 0 then 1
  else n * factoriel (n-1)
```

```
let rec triangle n =
  if n = 0 then 0
  else n + triangle (n-1)

let rec somme_triangles n =
  if n = 0 then 0
  else triangle n + somme_triangles (n-1)
```

```
let prod_mat m1 m2 =
  let n1 = Array.length m1 in
  let n2 = Array.length m1.(0) in
  assert (Array.length m2 = n2);
  let n3 = Array.length m2.(0) in
  let m3 = Array.make_matrix n1 n3 0 in
  for i = 0 to n1-1 do
    for j = 0 to n3-1 do
      for k = 0 to n2-1 do
        m3.(i).(j) <- m3.(i).(j) + m1.(i).(k) * m2.(k).(j)
      done
    done
  done;
  m3
```

```
let mystere n =
  let rec aux n acc =
    if n = 0 then acc
    else aux (n-1) 0 + aux (n-1) 1
  in aux n 0
```

Question 3.

1. Proposez une fonction efficace (en nombre de comparaisons) qui prend en arguments un tableau trié et son nombre d'éléments n et qui renvoie le maximum des valeurs $|x - y|$ où x et y prennent les différentes valeurs contenues dans le tableau.
   Par exemple, si le tableau est $[1, 3, 5, 8]$, x et y prennent les valeurs $1, 3, 5$ et 8 et le maximum de $|x - y|$ est 7. Il est obtenu pour $x = 1$ et $y = 8$.
   Quelle est sa complexité en nombre de comparaisons ?
2. Proposez une fonction efficace (en nombre de comparaisons) qui prend en arguments un tableau non trié et son nombre d'éléments n et qui renvoie le maximum des valeurs $|x - y|$ où x et y prennent les différentes valeurs contenues dans le tableau.
   Quelle est sa complexité en nombre de comparaisons ?
3. Proposez une fonction efficace (en nombre de comparaisons) qui prend en arguments un tableau trié et son nombre d'éléments n et qui renvoie le minimum des valeurs $|x - y|$ où x et y prennent les différentes valeurs contenues dans le tableau.
   Par exemple, si le tableau est $[1, 3, 5, 8]$, x et y prennent les valeurs $1, 3, 5$ et 8 et le minimum de $|x - y|$ est 2. Il est obtenu pour $x = 1$ et $y = 3$.
   Quelle est sa complexité en nombre de comparaisons ?
4. Proposez une fonction efficace (en nombre de comparaisons) qui prend en arguments un tableau non trié et son nombre d'éléments n et qui renvoie le minimum des valeurs $|x - y|$ où x et y prennent les différentes valeurs contenues dans le tableau.
   Quelle est sa complexité en nombre de comparaisons ?

Question 4.

1. Implémenter en C une fonction qui permet de trier un tableau de taille n en $O(n^2)$.
2. Un tri par comparaison (n'utilisant aucune information sur les données du tableau et n'utilisant que les opérations de comparaison et d'échange de place) ne peut pas être plus efficace que $O(n \log n)$ pour trier un tableau de taille n dans le pire des cas. Peut-on trier un tableau de taille n plus rapidement que $O(n \log n)$ lorsque le tableau ne contient que des 0 et des 1 ?
3. Quid des tableaux ne contenant que des 0, des 1 et des 2 ?