Algorithmique avancée : Feuille de TD n^o 1

I Complexité des algorithmes

1 Complexité asymptotique

Déterminer les notations O, Ω, Θ et o des expressions suivantes :

1) $2n^2 + 5n + 10$
2) $\frac{1}{5} n \log_2(n) + n$
3) $\sum_{i=0}^{d} c_i n^i$

2 Analyse d'un algorithme simple

Pour les trois formulations suivantes du problème de calcul du plus grand diviseur d'un nombre entier $n \geq 2$, exprimer l'ordre de grandeur de la complexité temporelle maximale et de la complexité spatiale maximale. En conclure l'algoritme le plus rapide pour le calcul du plus grand diviseur d'un nombre.

1) Recherche descendante du plus grand diviseur

   Notons pgd(n) le plus grand diviseur de n. On a les propriétés suivantes :
   - $1 \leq \text{pgd}(n) \leq n - 1$
   - $\text{pgd}(n) = 1 \Leftrightarrow n$ est premier.

   On déduit de ces propriétés l'algorithme suivant :

```
Require: n ≥ 2
Ensure: r = pgd(n)
    r ← n − 1
    while n mod r ≠ 0 do r ← r − 1
    return r
```

2) Recherche ascendante du plus petit diviseur ≥ 2

   Nous pouvons remarquer que $\text{pgd}(n) = \frac{n}{p}$ où p est le plus petit diviseur de n qui soit ≥ 2, noté $\text{ppd}(n)$.

   On déduit de cette remarque l'algorithme suivant :

```
Require: n ≥ 2
Ensure: r = pgd(n)
    k ← 2
    while n mod k ≠ 0 do  k ← k + 1
    r ← n/k
    return r
```

3) Réduction de l'espace de recherche

   On peut remarquer que n non premier $\Rightarrow 2 \leq \text{ppd}(n) \leq \text{pgd}(n) \leq n - 1$. D'où, comme $ppd(n) \times pgd(n) = n$, n non premier $\Rightarrow (\text{ppd}(n))^2 \leq n$. On en déduit que si n ne possède pas de diviseurs compris entre 2 et $[\sqrt{n}\,]$, c'est qu'il est premier. Nous pouvons donc utiliser

cette remarque pour réduire l'espace de recherche et proposer l'algorithme suivant :

Require: $n \geq 2$
Ensure: $r = \texttt{pgd}(n)$

$k \leftarrow 2$
while *(n mod k ≠ 0) and (k ≤ n/k)* **do** $k \leftarrow k + 1$
if $k > n/k$ **then**
  $r \leftarrow 1$
else
  $r \leftarrow n/k$
return r

3 Analyse de l'algorithme de tri fusion

L'algorithme du tri fusion peut s'exprimer de la manière récursive suivante, dans laquelle la fonction **fusion**(T, i, j, k) effectue la fusion des deux sous-tableaux triés $T[i : j]$ et $T[j + 1 : k]$ pour fournir le sous-tableau trié $T[i : k]$ avec une complexité temporelle en $\Theta(k - i + 1)$.

Function FusionSort(T, i, k)

Require: T `: array`
i, k `integer so that` $T_m \leq i < k \leq T_M$ `with` T_m, T_M `the bounds of` T
Ensure: T `is sorted between` i `and` k

if $k > i$ **then**
  $j \leftarrow (k + i)/2$
  FusionSort(T, i, j)
  FusionSort$(T, j + 1, k)$
  fusion(T, i, j, k)

1) Dérouler cet algorithme sur le tableau

40	10	30	45	10

2) Calculer la complexité temporelle dans le pire cas de cet algorithme.

II Structures de données fondamentales

1 Structures de données pour la gestion de priorité

1) Construire le tas binaire maximal correspondant au tableau suivant :

10	1	6	12	25	8	14	29	18	11	17	38	27

2) Donner les nombres minimum et maximum d'éléments dans un tas binaire de hauteur h.
3) Montrer que, dans tout sous-arbre d'un tas maximal, la racine du sous-arbre à une clé supérieure à tous les éléments du sous arbre.
4) En supposant que tous les éléments d'un tas maximal sont différents, à quelle position se trouve la valeur minimale dans un tas maximal.
5) Montrer que la taille maximale d'un sous arbre d'un arbre presque plein de taille n est inférieure à $2n/3$.
6) (Travail personnel) Montrer qu'un tas binaire de n élément à une hauteur de $[\log n]$
7) (Travail personnel) Un tableau trié est-il un tas minimal ?
8) (Travail personnel) Montrer que dans un tableau représentant un tas binaire avec n éléments, les éléments d'indices $([n/2] + 1)..n$ sont des feuilles.

9) (Travail personnel) Montrer qu'il y a au plus $[\frac{n}{2^{h+1}}] + 1$ nœuds de hauteur h dans un tas de n éléments (Attention, ne pas confondre hauteur et profondeur d'un nœud [1]).

10) (Travail personnel) Ecrire un algorithme de tri dans l'ordre croissant, en place (i.e. en complexité spatiale $O(1)$) en utilisant un tas maximal. Montrer que cet algorithme de tri par tas à une complexité maximale en $O(n \log n)$.

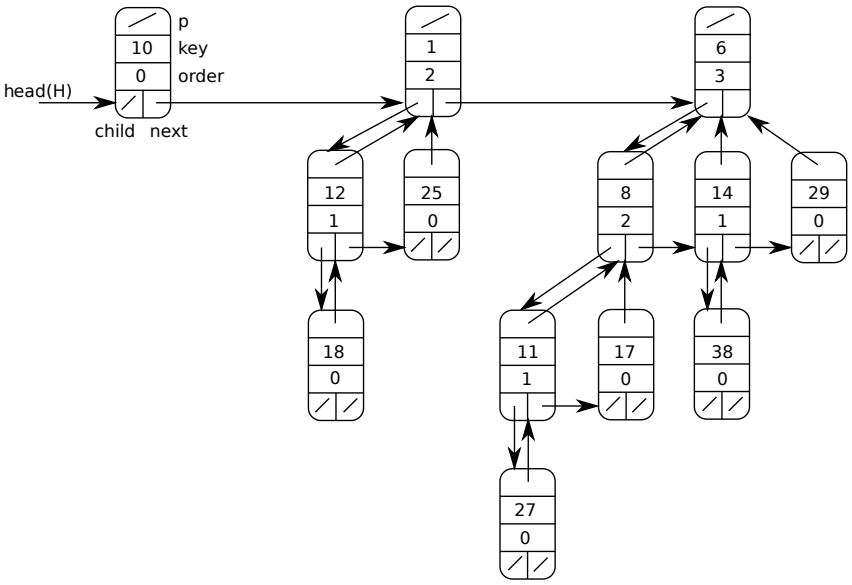

FIGURE 1 – Représentation d'un tas binomial.

11) Dessiner le tas binomial résultant de la suppression de l'élément minimum dans le tas binomial de la figure **??**.

12) Dessiner le tas binomial résultant de la suppression de l'élément 11 dans le tas binomial de la figure **??**.

13) (Travail personnel) Coder en binaire le nombre n d'éléments du tas binomial obtenu à la question précédente. Donner l'opération sur des nombres binaires qui a permis d'obtenir ce nombre binaire en vous servant de l'union de tas que vous avez réalisée lors de la question précédente.

1. Indice : procéder par récurrence.