TD 1

Algorithmique

Exercice 1: Le Grand Saut

Le problème est de déterminer à partir de quel étage d'un immeuble sauter par la fenêtre est fatal. Vous êtes dans un immeuble à n étages (numérotés de 1 à n) et vous disposez de k étudiants. Il n'y a qu'une seule opération possible pour tester si la hauteur d'un étage est fatale : faire sauter un étudiant par la fenêtre. S'il survit, vous pouvez le réutiliser ensuite, sinon vous ne pouvez plus.

Vous devez proposer un algorithme pour trouver la hauteur à partir de laquelle un saut est fatal (renvoyer $n+1$ si on survit encore au n-ième étage) en faisant le minimum de sauts.

1. Si $k \geq \lceil \log_2(n) \rceil$, proposer un algorithme en $O(\log_2(n))$ sauts.
2. Si $k < \lceil \log_2(n) \rceil$, proposer un algorithme en $O(k + \frac{n}{2^{k-1}})$ sauts.
3. Si $k = 2$, proposer un algorithme en $2\sqrt{n}$ sauts.
4. Dans ce dernier cas, proposer aussi un algorithme en $\sqrt{2n}$ sauts.
5. Montrer dans ce dernier cas, que au moins $\sqrt{n}$ sauts sont nécessaires.

Exercice 2: Fibonacci

La suite de Fibonacci est définie par:

$$F_0 = F_1 = 1 \text{ et pour tout n } \geq 2, \quad F_n = F_{n-1} + F_{n-2}$$

On veut calculer le n-ième terme de la suite de manière efficace

1. avec un algorithme récursif. Donner le nombre d'appel récursif de cet algorithme.
2. proposer une variante où chaque valeur de la suite n'est calculée qu'une seule fois en utilisant de la mémoire. Éviter l'utilisation de mémoire trop grande.
3. Donner une solution où la complexité est logarithmique en n.

Exercice 3: Problème d'Optimisation : Bin Packing

Soit n objets de volume des rationnels $a_1, \ldots, a_n$ où $0 < a_i \leq 1$. Peut-on les partitionner en k boîtes $B_1, \ldots, B_k$ de capacité au plus 1, tel que

$$\sum_{j \in B_k} \leq 1$$

On cherche à minimiser la valeur k.

Une λ-approximation est un algorithme polynomial tel que, pour toute instance I du problème, $Sol_{algo}(I) \leq \lambda \cdot Opt(I)$.

1. L'algorithme Next Fit est le suivant:
   - prendre les objets dans un ordre quelconque
   - placer l'objet courant dans la dernière boîte utilisée s'il tient, sinon en créer une nouvelle

   Montrer que Next Fit est une 2-approximation. Montrer que la borne est fine.
2. L'algorithme Dec First Fit est le suivant:
   - Trier les a_i par ordre décroissant
   - Placer l'objet courant dans la première boîte utilisée où il tient, sinon en créer une nouvelle

   Montrer que Dec First Fit est une 3/2-approximation.

Exercice 4: Mariage

Un graphe biparti a ses sommets constitués de deux ensembles disjoints H et F et n'a d'arête qu'entre un sommet de H et un sommet de F. Un mariage est un ensemble d'arêtes n'ayant deux à deux aucun sommet commun. Si M est un mariage, on note $M(h)$, $h \in H$, l'unique élément f de F s'il existe tel que l'arête $\{h, f\}$ soit dans M. On pose $M(h) = \perp$ si f n'existe pas. On définit $M(f)$, $f \in F$, de manière analogue.

On se donne un graphe biparti complet à $2n$ sommets, ce qui signifie que chaque sommet h de H est lié à chaque sommet f de F. On se donne de plus pour chaque sommet h de H une fonction injective $r_h(f)$ qui affecte un entier $\leq n$ aux éléments de F. De même, on se donne pour chaque sommet f de F une fonction injective $r_f(h)$ qui affecte un entier aux éléments de H. Un mariage est *stable* s'il n'existe pas de couple (h, f), $h \in H$, $f \in F$ tel que:

- $\{h, f\} \notin M$
- $M(f)$ et $M(h)$ sont définis
- $r_h(f) > r_h(M(h))$
- $r_f(h) > r_f(M(f))$

Un mariage est *H-saturé* si pour tout $h \in H$, tel que $M(h)$ existe et pour tout $f \in F$ tel que $r_h(f) > r_h(M(h))$, on a $M(f) \neq \perp$.

1. Soit M un mariage stable H-saturé. Soit $h \in H$ tel que $M(h)$ soit non défini. On considère la réunion des ensembles

$$\{f \in F \mid M(f) = \perp\} \text{ et } \{f \in F \mid M(f) \neq \perp,\ r_f(h) > r_f(M(f))\}$$

   Soit f l'élément de la réunion de ces ensembles rendant $r_h(f)$ maximal. On définit M'
   - en ajoutant l'arête (h, f) à M si $M(f) = \perp$
   - en retirant de M l'arête $(f, M(f))$ et en ajoutant (h, f) sinon

   Montrer que M' est stable et H-saturé.
2. Proposer un invariant entier $n(M)$ qui augmente strictement quand on passe de M à M'.
3. Proposer un algorithme qui calcule un mariage stable ayant n arêtes. Montrer sa correction et évaluer sa complexité.