Partiel – Lundi 7 novembre 2011

Durée : 2h. Notes de cours et documents non autorisés. Le sujet comprend 4 exercices indépendants.

La notation tiendra compte de la qualité de la rédaction et du soin apporté à la copie. Pour maximiser votre note, justifiez donc correctement toutes vos réponses et écrivez lisiblement. Ce n'est pas un concours de vitesse !
Si vous bloquez sur une question d'un exercice, vous pouvez admettre le résultat et passer à la suite.

Exercice 1. *Arbre couvrant de poids minimal*

Un graphe $G = (V, E)$ est un ensemble fini de sommets V et d'arêtes $E \subseteq \mathcal{P}_2(V)$ où $\mathcal{P}_2(V)$ est l'ensemble des parties à deux éléments de V. Dans la suite de cet exercice, nous considérons un graphe pondéré, c'est-à-dire qu'on associe un poids à chacune des arêtes. Plus précisément, on associe au graphe une fonction $\omega : E \to \mathbb{N}$.
Nous représentons les graphes par des matrices d'adjacences. Un graphe G à n sommets est donc représenté par une matrice symétrique M de taille $n \times n$ où chaque case $M(i, j)$ donne le poids de l'arête $\{i, j\}$ dans le graphe G. Si l'arête n'existe pas, on considère alors un poids infini.
Un arbre est un graphe connexe (il existe un chemin entre toute paire de sommets) sans cycle. Un arbre couvrant T est un sous-ensemble des arêtes tel que chaque sommet du graphe appartient à au moins une arête de T, et tel que le graphe constitué uniquement de ces arêtes est un arbre. Le poids d'un arbre couvrant est $\omega(T) = \sum_{e \in T} \omega(e)$.
L'algorithme de Prim est un algorithme permettant de trouver un arbre couvrant de poids minimum dans un graphe connexe.

```
Prim(G, ω)
1  Prendre s ∈ V un sommet quelconque du graphe;
2  Soit S est un ensemble de sommets;
3  Soit T un ensemble d'arêtes;
4  S ← {s};
5  T ← ∅;
6  tant que V \ S n'est pas vide faire
7  |  S' ← {{u, v} : u ∈ S, v ∈ V \ S};
8  |  {u, v} ← arête de poids minimum de S';
9  |  S ← S ∪ {v};
10 |  T ← T ∪ {{u, v}};
```

1. Considérant que l'accès (et donc la lecture) d'une case de la matrice est faite en temps unitaire, donner la complexité dans le pire des cas de cet algorithme.
2. Montrer la correction de cet algorithme, c'est-à-dire qu'à la fin de l'algorithme l'ensemble T est un arbre couvrant minimum.

Exercice 2. *Multiplication de deux entiers*

Soient deux entiers x et y codés sur n bits (on supposera que n est une puissance de 2). On souhaite multiplier ces deux entiers entre eux, en travaillant au niveau des bits. La multiplication de deux entiers de n bits se fait de manière triviale en $\mathcal{O}(n^2)$, la multiplication d'un entier par une puissance de 2, et l'addition de 2 entiers se font en temps linéaire $\mathcal{O}(n)$.

1. La méthode diviser pour régner n'est pas toujours meilleure qu'un algorithme naïf. Pour illustrer cela, donnez un algorithme diviser pour régner ayant une complexité en $\mathcal{O}(n^2)$ pour multiplier deux entiers x et y de n bits chacun.
2. Proposez un algorithme diviser pour régner ayant une complexité inférieure à $\mathcal{O}(n^2)$. Donnez la formule de récurrence pour votre algorithme et sa complexité finale (en $\mathcal{O}$). On supposera pour simplifier que l'addition/multiplication de deux nombres de taille n est un nombre de taille n.

Exercice 3. *Bibliothèque*

La bibliothèque planifie son déménagement. Elle comprend une collection de n livres $b_1, b_2, \ldots, b_n$. Le livre b_i est de largeur w_i et de hauteur h_i. Les livres doivent être rangés dans l'ordre donné (par valeur de i croissante) sur des étagères identiques de largeur L.

1. On suppose que tous les livres ont la même hauteur $h = h_i, 1 \leq i \leq n$. Montrer que l'algorithme glouton qui range les livres côte à côte tant que c'est possible minimise le nombre d'étagères utilisées.
2. Maintenant les livres ont des hauteurs différentes, mais la hauteur entre les étagères peut se régler. Le critère à minimiser est alors l'encombrement, défini comme la somme des hauteurs du plus grand livre de chaque étagère utilisée.
   (a) Donner un exemple où l'algorithme glouton précédent n'est pas optimal.
   (b) Proposer un algorithme optimal pour résoudre le problème, et donner son coût.
3. On revient au cas où tous les livres ont la même hauteur $h = h_i, 1 \leq i \leq n$. On veut désormais ranger les n livres sur k étagères de même longueur L à minimiser, où k est un paramètre du problème. Il s'agit donc de partitionner les n livres en k tranches, de telle sorte que la largeur de la plus large des k tranches soit la plus petite possible. Proposer un algorithme pour résoudre le problème, et donner son coût en fonction de n et k.

Exercice 4. *Pauvres élèves...*

Au premier TD, nous avions étudié le problème algorithmique suivant : déterminer à partir de quel étage d'un immeuble, sauter par une fenêtre est fatal. On suppose qu'on a k étudiants et un immeuble à n étages, numérotés de 1 à n. Une seule opération est possible : jeter un étudiant par la fenêtre. S'il survit, on peut réutiliser l'étudiant, sinon on doit en prendre un nouveau. On veut alors *déterminer l'étage maximal n_0 ($0 \leq n_0 \leq n$) duquel un saut n'est pas fatal*.

On rappelle les résultats suivants démontrés en TD :
- Si $k \geq \lceil \log n \rceil$, il suffit de $\mathcal{O}(\log n)$ sauts.
- Si $k < \lceil \log n \rceil$, il suffit de $\mathcal{O}(k + n/2^{k-1})$ sauts.
- Si $k = 2$, il suffit de $\mathcal{O}(\sqrt{n})$ sauts.

On s'intéresse ici à l'optimalité de ces réponses. Pour cela, on change le point de vue du problème. Supposons qu'on dispose d'un immeuble avec un nombre infini (dénombrable) d'étages et notons $h < +\infty$ la hauteur maximale de laquelle un saut n'est pas fatal. On définit alors $H(k, s)$ comme étant la hauteur maximale telle qu'il existe un algorithme utilisant au plus k élèves et au plus s sauts et qui calcule

$$n_0 = \max\{n : 0 \leq n \leq H(k, s) \text{ et un saut de l'étage } n \text{ n'est pas fatal}\}.$$

On remarque que cette hauteur $H(k, s)$ ne dépend pas de h, autrement dit l'algorithme doit fonctionner quelque soit la valeur de h.

Par exemple, $H(2, 3) = 6$. On peut voir que $H(2, 3) \leq 6$ avec l'algorithme ci-dessous. Montrer que cette valeur est exactement 6 est l'objet de cet exercice. Plus précisément, on détermine dans cet exercice une expression de $H(k, s)$ pour tous k et s.

```
1  Faire sauter le premier élève à hauteur 3;
2  si le saut est fatal alors                                   // h < 3
3  |  Faire sauter le deuxième élève à hauteur 1, puis éventuellement 2;
4  |  Répondre 0, 1 ou 2 en fonction des sauts du deuxième élève;
5  sinon                                                        // h ≥ 3
6  |  Faire sauter le premier élève à hauteur 5;
7  |  si le saut est fatal alors                                // h < 5
8  |  |  Faire sauter le deuxième élève à hauteur 4;
9  |  |  Répondre 3 ou 4 en fonction du saut du deuxième élève;
10 |  sinon                                                     // h ≥ 5
11 |  |  Faire sauter le premier élève à hauteur 6;
12 |  |  Répondre 5 ou 6 en fonction de ce dernier saut;
```

Algorithme pour $k = 2$, $s = 3$ et hauteur maximale testée 6

1. **Préliminaires.**
   - **(a)** Calculer $H(0, s)$ et $H(1, s)$ pour tout s.
   - **(b)** Que peut-on dire de $H(k, s)$ pour $k > s$?
2. **Avec 2 élèves.**

   On s'intéresse ici au calcul de $H(2, s)$. On suppose donc qu'on dispose de deux élèves.
   - **(a)** Montrer que le premier saut doit être effectué à hauteur au plus $(1 + H(1, s - 1))$.
   - **(b)** En déduire que $H(2, s) \leq \binom{s+1}{2}$.
   - **(c)** Montrer que cette borne est atteignable en donnant un algorithme sur l'entrée n détermine l'étage maximal non fatal $n_0 \leq n$ en utilisant au plus deux élèves et au plus s_n sauts où $s_n = \min\{s : n \leq H(2, s)\}$.
   - **(d)** Estimer s_n et comparer avec la valeur obtenue au TD 1 avec l'algorithme suivant : on coupe les étages en $\lceil \sqrt{n} \rceil$ tranches de $\lceil \sqrt{n} \rceil$ étages (sauf la dernière éventuellement plus petite) puis on fait sauter le premier élève aux étages $\lceil \sqrt{n} \rceil$, $2\lceil \sqrt{n} \rceil$, etc... jusqu'à arriver à un étage fatal, puis on fait sauter le deuxième élève dans la tranche précédente pour déterminer exactement la hauteur fatale.

3. **Cas général : calcul de la série génératrice.**
Pour étudier le cas général, on va avoir recours aux séries génératrices. On définit

$$\mathcal{G}(z,t) = \sum_{k\geq 0}\sum_{s\geq 0} H(k,s) z^s t^k.$$

On définit également, à k fixé,

$$\mathcal{G}_k(z) = \sum_{s\geq 0} H(k,s) z^s.$$

(a) Calculer $\mathcal{G}_0(z)$ et $\mathcal{G}_1(z)$.

(b) En s'inspirant du cas de deux élèves, exprimer $H(k,s)$ en fonctions des $H(k-1,u)$ pour $u < s$.

(c) Exprimer $\mathcal{G}_k(z)$ en fonction de $\mathcal{G}_{k-1}(z)$ et en déduire que

$$\mathcal{G}_k(z) = \frac{z}{(1-z)(1-2z)} \times \left(1 - \frac{z^k}{(1-z)^k}\right).$$

(d) En utilisant le fait que $\mathcal{G}(z,t) = \sum_{k\geq 0} \mathcal{G}_k(z) t^k$, montrer que

$$\mathcal{G}(z,t) = \frac{zt}{(1-z)(1-t)(1-z-zt)}.$$

4. **Cas général : utilisation de la série génératrice.**

(a) Montrer que

$$\frac{t+1}{1-z-zt} = \sum_{s\geq 0}\sum_{k=0}^{s+1} \binom{s+1}{k} z^s t^k.$$

(b) En déduire que

$$\frac{z(t+1)}{(1-t)(1-z-zt)} = \sum_{s\geq 1}\sum_{k\geq 0}\sum_{i=0}^{k} \binom{s}{i} z^s t^k.$$

(c) Décomposer $\mathcal{G}(z,t)$ en éléments simples *en fonction de* z, c'est-à-dire sous la forme

$$\mathcal{G}(z,t) = \frac{\alpha(z,t)}{1-z} + \frac{\beta(z,t)}{1-z-zt}.$$

(d) Déduire des questions précédentes une expression de $H(k,s)$.