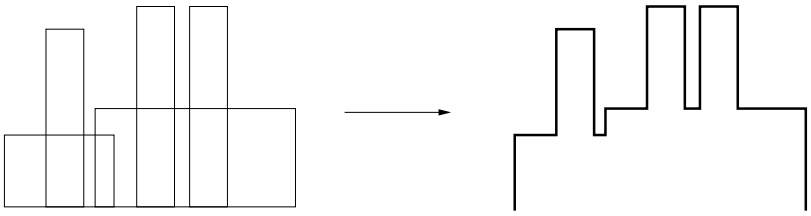
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Sections d'Informatique et de Systèmes de Communication

Série d'exercices 6 26 Octobre 2007

1. Diviser pour régner : Le problème du skyline

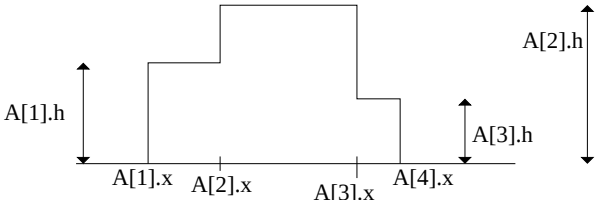
On décrit un bâtiment par un triplet $(x_1, x_2, h) \in \mathbb{R}^3$ où x_1 décrit la première coordonnée sur l'axe x et x_2 la deuxième. La hauteur du bâtiment est décrite par h. Étant donné une liste de bâtiments, la tâche est de trouver la silhouette (skyline) formée par nos bâtiments. Illustration :



La silhouette est représentée par la liste croissante de coordonnées x et la hauteur correspondante.

a) Etant donné deux skylines on veut écrire un programme pour les combiner (c'est-à-dire trouver le skyline obtenu avec tous les bâtiments du premier et du deuxième).

On suppose qu'on nous donne deux tableaux A et B dont les entrées contiennent toutes deux valeurs : x et h. Ces tableaux décrivent les deux skylines qu'on veut combiner. Le output est aussi un skyline, c'est-à-dire un tableau C avec le même type d'entrées, qui décrit le skyline obtenu en combinant les skylines de A et B.



Ecrire un algorithme qui, étant donné comme input les tableaux A et B, retourne comme output le tableau C. (*Indication :* Regarder MergeSort).

b) Utiliser le fait qu'on puisse combiner deux skylines pour construire un algorithme diviser-pour-régner qui résout le problème Skyline en $O(n \log(n))$.

2. Algorithmes

Considérons le problème suivant : étant donné une suite triée (croissante) d'éléments de $\mathbb{N}$ et un nombre $x \in \mathbb{N}$, déterminer s'il existe dans la suite deux éléments distincts dont la somme vaut x.

Par exemple pour l'input $\big((1, 2, 2, 3, 3, 10, 11),\ 6\big)$ l'output serait *vrai*, alors que pour l'input $\big((1, 1, 1, 2, 5, 6, 6, 7),\ 4\big)$ l'output serait *faux*.

a) Donner une spécification formelle de ce problème.

b) Quel est le temps de parcours de l'algorithme naïf pour ce problème ?

c) Supposons que la suite est stockée dans un tableau a de taille n. Donner (en pseudo code) un algorithme qui résout ce problème en temps $O(n)$.

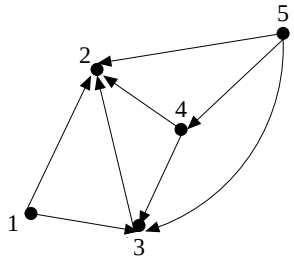
3. Algorithmes par induction : Le problème de la célébrité

Nous aimerions trouver une célébrité dans un ensemble V de n personnes. Pour modéliser ce problème, nous caractérisons une célébrité comme suit : C'est une personne qui est connue de tout le monde, mais qui ne connait personne.

Pour résoudre ce problème, nous avons le droit de poser un seul type de question : si $a, b \in V$ sont des personnes, nous pouvons demander à a s'il connait b, et il nous répondrera par "oui" ou "non". Le problème est donc le suivant : Etant donné un groupe de personnes V contenant une célébrité, trouver la célébrité en posant de telles questions.

Remarquons qu'on peut modéliser ce problème comme un problème de graphe orienté (V, E) : L'ensemble V est l'ensemble des personnes, et on met une flèche de $a \in V$ à $b \in V$ si et seulement si a connait b. Poser la question "a connait-il b?" revient alors à vérifier si l'élément (a, b) de la matrice d'adjacence vaut 1.

a) Dans le graphe suivant y-a-t il une célébrité ?



b) Supposons que nous avons un ensemble de personnes V dont certaines se connaissent (on obtient donc un graphe orienté (V, E)). Montrer qu'il peut y a voir au plus une célébrité dans V.

c) Supposons que le sommet i est une célébrité. Que peut-on dire sur la ligne i et la colonne i de la matrice d'adjacence du graphe ?

d) Décrire l'algorithme naïf pour résoudre le problème de la célébrité, étant donné la matrice d'adjacence du graphe. Donner l'ordre du nombre de questions qu'il faut poser (c'est-à-dire le nombre d'entrées dans la matrice qu'il faut regarder) en fonction du nombre de personnes $n = |V|$.

e) Observons que pour tout $a, b \in V$, on a
   - Si a connaît b, alors a ne peut pas être une célébrité.
   - Si a ne connaît pas b, alors b ne peut pas être une célébrité.

   Nous voulons utiliser ce fait pour construire par induction un algorithme $O(n)$ pour résoudre le problème de la célébrité.
   - *(i)* Base. Montrer que pour $n = 2$ on peut résoudre le problème en posant une seule question.
   - *(ii)* Pas. Supposons que nous savons résoudre le problème pour n personnes en posant $T(n)$ questions. Montrer qu'on peut le résoudre pour $n+1$ personnes en posant $T(n)+1$ questions.
   - *(iii)* En déduire un algorithme pour résoudre le problème pour n personnes, et donner le nombre de questions qui sont nécessaires.