ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Sections d'Informatique et de Systèmes de Communication

Série d'exercices 9 23 Novembre 2009

1. *Application de* QuickSort *et* HeapSort

Voici une suite de 30 nombres:

$$23, 29, 84, 15, 58, 19, 81, 17, 48, 15, 36, 49, 91, 26, 89,$$
$$22, 63, 57, 33, 10, 50, 56, 85, 4, 10, 63, 1, 72, 10, 48.$$

a) Trier cette suite avec QuickSort (en prenant toujours comme pivot le dernier élément, comme dans le cours). S'il vous reste des sous-suites de ≤ 5 éléments à trier, vous pouvez le faire à la main.

b) Trier les quinze premiers éléments ci-dessus avec HeapSort.

2. *Dégénérescence de* QuickSort

Rappelons que QuickSort nécessite en moyenne $O(N \log N)$ comparaisons si la permutation des clés est aléatoire et uniformément choisie. Néanmoins, il faut à QuickSort $\Omega(N^2)$ comparaisons dans le pire des cas.

Intuitivement, QuickSort aura une meilleure performance si le pivot choisi est plus proche de la médiane de la suite (pour qu'elle soit coupée en deux sous-suites de même taille lors de l'appel récursif). Par contre, si le pivot se trouve parmi les plus grands (ou plus petits) éléments de la suite, il faudra faire plus d'appels récursifs, et donc plus de comparaisons. Le choix du pivot est donc crucial.

a) Si on choisit toujours comme pivot le dernier élément de la suite, montrer que pour tout N on peut trouver une suite de N éléments pour laquelle QuickSort nécessite $\Omega(N^2)$ comparaisons.

b) Considérons la stratégie suivante: on prend comme pivot toujours la médiane des éléments au début, au milieu et à la fin. Puis on échange cet élément avec le dernier élément de la suite (si nécessaire), et on procède ensuite comme dans le cours. (C'est la *3-median strategy*.) Montrer que pour tout N on peut trouver une suite de N éléments pour laquelle QuickSort nécessite $\Omega(N^2)$ comparaisons.

3. *Nombre moyen d'échanges avec* SelectionSort

Nous voulons calculer le nombre moyen d'échanges dont a besoin l'algorithme SelectionSort pour trier une suite de longueur N. (Ici *moyen* veut dire la moyenne sur tous les inputs possibles, c'est-à-dire toutes les permutations d'une suite de longueur N.)

a) Supposons que $N = 3$. Nous avons donc une suite de 3 éléments à ordonner. Supposons que les clés valent $0, 1$ et 2. Pour chacune des 6 permutations possibles de ces 3 éléments, combien faut-il à SelectionSort d'échanges pour ordonner la suite?

En supposant que chacune de ces permutations a la même probabilité, quel est le nombre moyen d'échanges nécessaires?

b) Supposons maintenant que N est quelconque. L'algorithme fait $N-1$ itérations (dans le programme la variable i va de 0 à $N-2$).

Pendant l'itération i il faudra faire un échange si et seulement si le $i^{\text{ème}}$ élément de la suite n'est pas à la bonne position. On voit aussi qu'au début de l'itération i les éléments $0,\ldots,i-1$ sont tous à la bonne place.

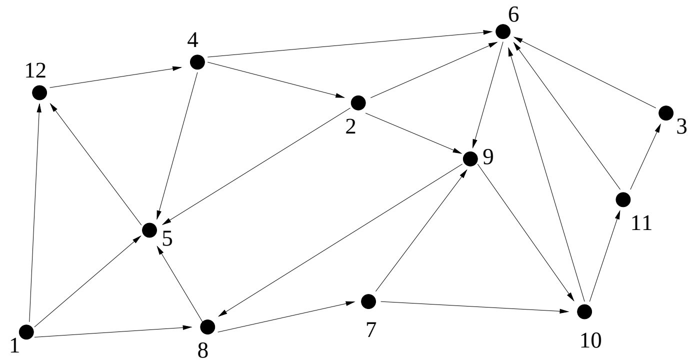
(i) Etant donné une permutation aléatoire de $0,\ldots,k-1$, quelle est la probabilité que 0 soit à la bonne place (c'est-à-dire en premier)?

(ii) Montrer que le nombre moyen d'échanges dont a besoin SelectionSort pour trier une suite à N éléments est égal à

$$\frac{N-1}{N}+\frac{N-2}{N-1}+\ldots+\frac{2}{3}+\frac{1}{2}.$$

4. Traverser des graphes

Voici un graphe orienté:



a) Parcourir ce graphe avec la méthode DepthFirstSearch, en commençant au sommet 1.

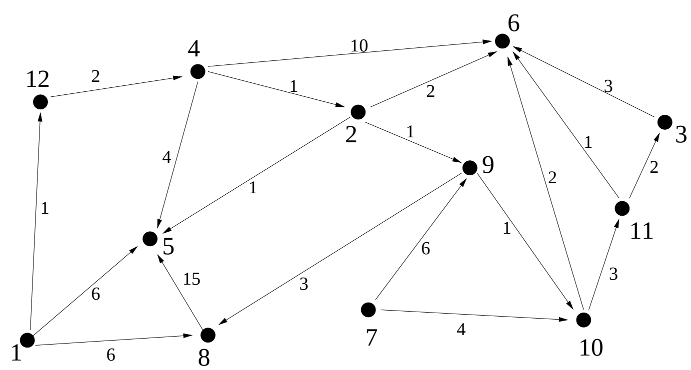
b) Parcourir ce graphe avec la méthode BreadthFirstSearch, en commençant au sommet 1.

c) Modifier l'algorithme BFS pour qu'il retourne pour chaque sommet sa distance par rapport à 1.

d) Modifiez l'algorithme AbstractTraversal pour qu'il compte aussi le nombre de sommets qui sont atteignables depuis 1.

5. L'algorithme de Dijkstra

Considérer le graphe suivant:



a) Appliquer l'algorithme de Dijkstra sur ce graphe. Commencer au sommet 1.

b) Modifier l'algorithme de Dijkstra pour qu'il fournisse pour chaque sommet v le plus court chemin du sommet de départ s à v et l'ensemble des sommets qui font partie de ce chemin.