Algorithmique

TD4 – Semaine du 27/10/2003

Éric Colin de Verdière, Yan Jurski, Sylvain Peyronnet

Exercice 1. Tri par sélection.

On considère un tableau T de taille n qui contient des entiers. Le tri par sélection consiste, pour chaque case i du tableau, à trouver la case j du tableau qui contient l'élément devant être placé à la case i, et à échanger les contenus des cases i et j. Ici, on parcourt le tableau de gauche à droite (i va de 1 à n) et on veut trier les éléments dans l'ordre croissant.

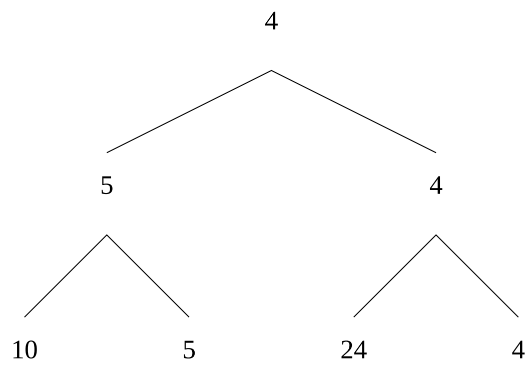
1. Écrire l'algorithme du tri par sélection en pseudo-langage.
2. Donner la complexité asymptotique de l'algorithme (en notation Θ) dans le meilleur cas et dans le cas le pire.

Exercice 2. Tri par sélection avec arbre.

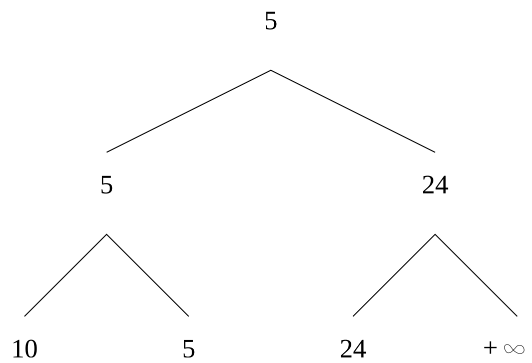
On va améliorer l'algorithme précédent en utilisant des arbres binaires. L'idée est de faire moins de comparaisons à chaque étape pour la recherche du minimum.

Pour simplifier, on suppose que la taille n du tableau à trier est une puissance de 2 : $n = 2^h$. (Si tel n'est pas le cas, rien n'empêcherait d'ajouter des nombres « $+\infty$ » au bout du tableau.)

Prenons un exemple : le tableau `[10 5 24 4]` est trié comme suit. On construit un arbre binaire complet où les feuilles lues de gauche à droite sont les éléments du tableau et où chaque nœud est le minimum de ses fils :



Ensuite, on prend le minimum (la racine) qu'on met dans un tableau auxiliaire, on le remplace dans le tableau par $+\infty$ et on recalcule les nœuds qui changent de valeur :



On recommence jusqu'à ce que tous les éléments soient enlevés.

1. Appliquer cet algorithme à l'exemple `[10 8 29 3 49 4 22 12]`.
2. Expliquer comment on peut efficacement :
   (a) écrire l'information dans les nœuds de l'arbre initial (on ne demande pas de décrire la structure de données utilisée pour le stockage de l'arbre, que l'on pourra supposer déjà créé avec des nœuds sans aucune information) ;
   (b) recalculer les nœuds qui changent de valeur après avoir remplacé un élément du tableau par $+\infty$.

   Cela fonctionne-t-il si deux cases du tableau initial ont la même valeur ?
3. Calculer le nombre exact de comparaisons entre éléments du tableau avec cet algorithme, dans le pire des cas, le meilleur des cas, le cas moyen.
4. Combien de fois compare-t-on $+\infty$ avec $+\infty$ au cours de cet algorithme ?

Exercice 3. Recherche d'élément dans un tableau trié.

On se donne un tableau de n entiers triés. On veut savoir si un entier donné est dans le tableau.

Est-il nécessaire de regarder les éléments un par un ? Donner en pseudo-langage un algorithme efficace pour ce problème (on donnera sa complexité en utilisant la notation Θ).