correction du partiel d'algoritmique

Rannou Pierre et Morel Ivan

January 4, 2005

1 exercice 1

1.1 question 1

Pour que $|a-b|$ soit maximum il faut que a (Resp b) soit maximum et que b (Resp a) soit minimum. Le problème se résume donc à une recherche du min et du max qui se fait en $O(n)$.

1.2 question 2

Si $|c-d|$, est minimum pour c et d appartenant à S alors il n'existe pas e appartenant à S tel que $a > e > b$ ou $a < e < b$: sinon $|c-e| < |c-d|$. On trie donc S en $O(n \times logn)$ et on cherche ensuite le min des $s_i - s_{i+1}$, (s_i et s_{i+1} étant deux entier consécutifs de S) en $O(n)$.On obtient donc un algorithme en $O(n \times logn)$.

1.3 question 3

On commence par trier S en $O(n \times logn)$.Ensuite pour tout y appartenant à S on cherche si $m-y$ appartient à S par dicotomie (S est trié), la dicotomie se faisant en $O(logn)$ cette opération ce fait en $O(n \times logn)$. A la fin on obtient bien un algorithme en $O(n \times logn)$.

1.4 question 4

Il existe deux méthodes pour résoudre cette question.

1.4.1 première méthode

Cette méthode est basée sur la médiane. On note S_1 l'ensemble des élément de S inféreurs à la médiane et S_2 les éléments de S supérieurs à la médiane. Il est à noter que $\|S_1\| = \|S_2\| = \lceil \frac{\|S\|}{2} \rceil$.

début
- Calculer le min et le max de S ;
- **si** $n = 2$ **alors** retourner (Min,Max) ;
- **sinon**
  - extraire la médiane de S ; calculer S_1 S_2 ;
  - relancer l'algo sur S_1 ou S_2 suivant le cas.

fin

Preuve de l'algorithme : on pose $M(S)$ la moyenne pondérée de S, $n = \| S \|$, a le min de S, b le max de S, et m la médianne de S, deux cas sont alors possibles:

- n impair:

$$n = 2k + 1, \text{alors} \|S_1\| = \frac{n+1}{2} = k + 1$$
$$\text{donc :} M(S_1) = \frac{m-a}{k+1-1} = \frac{m-a}{k}$$
$$\text{et :} \|S_2\| = \frac{n+1}{2} = k + 1$$
$$\text{donc :} M(S_2) = \frac{b-m}{k+1-1} = \frac{b-m}{k}$$
$$\text{donc :} \frac{M(S_1) + M(S_2)}{2} = (\frac{m-a}{k} + \frac{b-m}{k}) \times \frac{1}{2} = \frac{b-a}{2k}$$
$$\text{or :} \frac{b-a}{2k} = \frac{b-a}{n-1} = M(S)$$
$$\text{donc :} 2 \times M(S) = M(S_1) + M(S_2)$$
$$\text{finalement :} M(S_1) \leq M(S) \text{ou} M(S_2) \leq M(S).$$

- n pair:

$$n = 2k, \text{alors} \|S_1\| = \frac{n+1}{2} = k$$
$$\text{donc :} M(S_1) = \frac{m-a}{k-1}$$
$$\text{et :} \|S_2\| = \frac{n+1}{2}$$
$$\text{donc :} M(S_2) = \frac{b-m}{k-1}$$
$$\text{donc :} \frac{M(S_1) + M(S_2)}{2} = (\frac{m-a}{k-1} + \frac{b-m}{k-1}) \times \frac{1}{2} = \frac{b-a}{2k-2}$$
$$\text{or :} \frac{b-a}{2k-2} \leq \frac{b-a}{n-1} = M(S)$$
$$\text{donc :} 2 \times M(S) \geq M(S_1) + M(S_2)$$
$$\text{finalement :} M(S_1) \leq M(S) \text{ou} M(S_2) \leq M(S).$$

Comme on sait que $M(S_1) \leq M(S)$ ou $M(S_2) \leq M(S)$ cela nous permet d'enclencher l'induction: soit sur S_1, soit sur S_2. Rechercher la moyenne se fait en $O(n)$, chercher le min et le max d'un ensemble aussi, donc:

$$C(n) = C(\lceil \frac{n}{2} \rceil) + O(n)$$

Ce qui donne par master théorème

$$C(n) = O(n)$$

1.4.2 deuxième méthode

On divise S en $(n-1)$ boites.

---

début

- **si** *il n'y a qu'une boite* **alors** on renvoie deux de ses éléments;
- **sinon**
  - **si** *le nombre d'éléments des* $\lfloor \frac{n-1}{2} \rfloor$ *premières boites est supérieur à* $\lfloor \frac{n-1}{2} \rfloor$.
  - **alors** la moyenne pondérée sur les $\lfloor \frac{n-1}{2} \rfloor$ premières boites est inférieure à celle de S, et on relance l'algorithme sur ces boites. ;
  - **sinon**
    - le nombre d'éléments des $\lceil \frac{n+1}{2} \rceil$ dernières boites est supérieur à $\lceil \frac{n+1}{2} \rceil$, et alors la moyenne pondérée sur les $\lceil \frac{n+1}{2} \rceil$ dernières boites est inférieure à celle de S, et on relance l'algorithme sur celles-ci.

fin

Quant à la complexité elle est en $O(n)$:

$$C(n) = C(\lceil \frac{n}{2} \rceil + 1) + O(n)$$

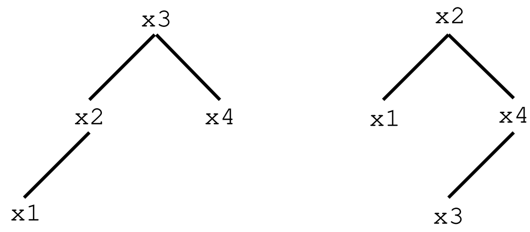
Ce qui donne par master théorème

$$C(n) = O(n)$$

2 exercice 2

2.1 question 1

il suffit de dessiner l'ensemble des graphes possibles (il y en a 14) pour trouver les deux graphes minimaux.



2.2 question 2

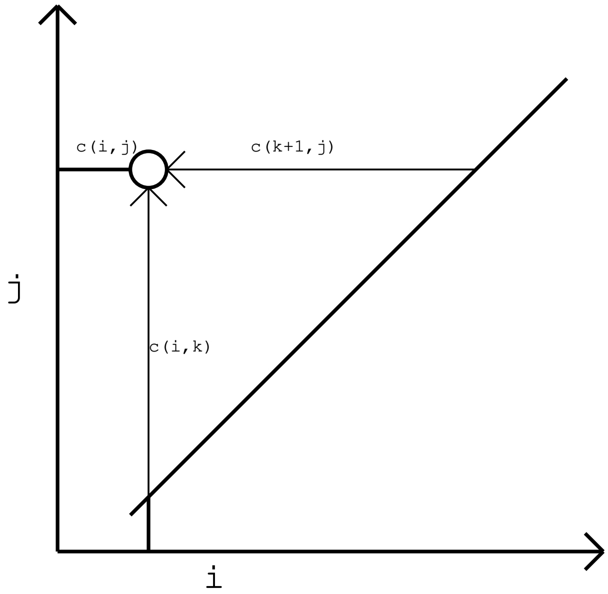
Le poids d'un arbre est le poid de ses fils, plus le poids de sa racine, plus le poids de tous les noeuds de ses fils (on leur rajoute un niveau). On appelle $t[i,j]$ le poid de l'arbre, de poids minimum contenant les sommets de i à j, et C_k le poid du sommet p. donc:

$$t[i,j] = min_{i \leq k \leq j}(t[i,k-1] + C_k + t[k+1,j] + \sum_{p=i}^{k-1}(C_p) + \sum_{p=k+1}^{j}(C_p))$$

soit:

$$t[i,j] = min_{i \leq k \leq j}(t[i,k-1] + t[k+1,j]) + \sum_{p=i}^{j}(C_p)$$

On initialise l'algo avec $t[i,i-1] = 0$.



la complexité est en $O(n^3)$: une boucle sur les i une autre sur les j puis une dernière sur les k.

3 exercice 3

3.1 question 1

- Pour une longueur inférieure à deux l'algo est réduit à l'échange ce qui donne un tableau trié.
- Pour i supérieur à trois la première ligne ne sert à rien. On notera $X \leq Y$ si $\forall x \in X\ \forall y \in Y : x \leq y$. On pose $l = \|T\|$,on pose A l'ensemble des $(\lfloor \frac{l+1}{3} \rfloor)$ premier élément de T, C l'ensemble des $(\lceil \frac{l}{3} \rceil)$ élément suivants , on pose B l'ensemble des $(\lfloor \frac{l}{3} \rfloor)$ dernier élément. On commence par trier les éléments de $A \cup C$ par induction: comme $\|A \cup C\| = (\lceil \frac{2l}{3} \rceil) < l$. Donc on obtient $A < C$. On trie maintenant $C \cup B$ par induction : comme $\|A \cup B\| \leq (\lceil \frac{2l}{3} \rceil) < l$.Alors $\|B\| \leq \|C\|$ ce qui implique que si un élément b de B avant le tri se retrouve dans B après, c'est qu'il existe un élément c dans C avant le tri tel que $c < b$ or comme c est supérieur à tous les éléments de A b aussi donc après le tri $C \leq B$ et $A \leq B$ ou encore $C \cup A \leq B$. Finalement on retrie $A \cup C$ et on obtient $A < B < C$ avec A, B, C, trié donc T est trié.

3.2 question 2

Pour trier un ensemble de n (n supérieur à trois) éléments on trie trois ensembles à $\lceil \frac{2n}{3} \rceil$ éléments et on fait un échange. Pour trier un ensemble de 2 éléments on fait un échange. Donc:

$$C(n) = 1 + C(\lceil \frac{2n}{3} \rceil)$$

et:

$$C(1) = 2$$

donc par master théorème:

$$C(n) = O(n^{log_{\frac{3}{2}}(3)}) \simeq O(n^{2,75})$$

C'est mauvais.

4 exercice 4

4.1 question 1

Pour appareiller les boulons et les écrous, il suffit de prendre un boulon arbitrairement, de le tester avec tous les écrous.On trouve alors le bon en au plus n tests et il suffit alors de recommencer avec tous les autres boulons.On obtient donc le résultat en au plus $\frac{n(n-1)}{2} = O(n^2)$ tests.

4.2 question 2

Le principe est de numéroter les boulons et les écrous de 1 à n, de manière arbitraire, et de faire progresser des compteurs (par exemple i et j) pour marquer le minimun courant dans l'une des catégories, et la structure en cours de test dans l'autre.

---

début
- **while** *(i ≤ n) && (j ≤ n)* **do**
  - **si** *i=j=n* **alors** sortir de la boucle ;
  - **si** *ecrou.i = boulon.j* **alors** s'en souvenir et faire $i := i + 1$;
  - **si** *ecrou.i < boulon.j* **alors** $j := j + 1$; $min = ecrou$;
  - **si** *ecrou.i > boulon.j* **alors** $i := i + 1$; $min = boulon$;

fin

A la fin de cette boucle,

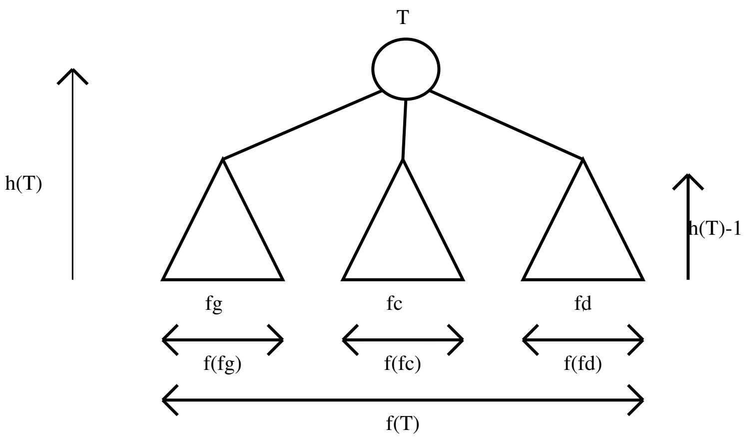
- si min=écrou, l'écrou i est le plus petit.
- si min=boulon, le boulon j est le plus petit. Et dans les deux cas, on sait déjà quel est l'élément qui correspond : en effet, le compteur de l'autre catégorie vaut n+1 (ou n dans un cas spécial),et on a donc déjà rencontré le minimum. la seule raison possible pour laquelle il n'a pas été conservé est donc qu'il ait été testé avec l'autre minimum. Pour le cas spécial ou i=j=n, le dernier test est inutile : en effet, on sait que l'un des deux, est minimum, et que une fois le boulon minimum atteind, j reste fixe : d'où le boulon n est minimum, et son homologue a soit déjà été trouvé, soit est l'écrou restant.

Cette boucle effectue donc au plus $2 * n - 2$ tests.

4.3 question 3

On note $f(T)$ le nombre de feuilles de l'arbre et $h(T)$ sa hauteur. Si T est un arbre ternaire $h(T) \geq \lceil log_3 f(T) \rceil$. Par induction sur la hauteur de T:

- Pour $h(T) = 0$ l'arbre à une feuille donc on a bien $h(T) \geq \lceil log_3 f(T) \lceil$.
- Pour $h(T) > 0$ un arbre ternaire a trois fils, le fils gauche fg, le fils central fc, et le fils droit fd de hauteur inférieur à $h(T) - 1$.On suppose que $h(T) \geq \lceil log_3 f(T) \rceil$ est vrai pour $h(T) \leq k$ k étant fixé on veut démontrer que cette propriété est vrai aussi pour $h(T) = k + 1$. Les feuilles d'un arbre sont aussi celles de ses fils. Donc



$$f(T) = f(fd) + f(fg) + f(fd)$$

de plus :

$$h(T) \leq h(fd) + 1$$
$$h(T) \leq h(fg) + 1$$
$$h(T) \leq h(fc) + 1$$

or par induction comme $h(T) = k + 1$, $h(fg) \leq k$, $h(fg) \leq k$,et $h(fg) \leq k$ on a:

$$k = h(fc) \leq \lceil log_3 f(fc) \rceil$$

$$h(fd) \leq \lceil log_3 f(fd) \rceil$$
$$h(fg) \leq \lceil log_3 f(fg) \rceil$$

Donc $f(fc), f(fg), f(fd) \leq 3^k$ or:

$$f(T) = f(fc) + f(fg) + f(fd)$$

donc:

$$f(T) \leq 3 \times 3^k = 3^{k+1}$$

D' où on en déduit que:

$$h(T) \geq log_3 f(T)$$

Il y a $!n$ agencements boulons-écrous possibles donc l'arbre de décision, qui est ternaire a pour hauteur: $log_3!n \sim nlog_3n$, donc la complexité dans le pire cas de l'algo est de: $\Omega(n \times logn)$.