www.tunisie-etudes.info

Ce document a été téléchargé depuis
www.tunisie-etudes.info

Des documents gratuits, devoirs, examens, cours, exercices, corrigés... Ainsi que toute une rubrique pour vous aider à trouver un emploi sans oublier les avis de concours en direct

Notre page Twitter :
http://www.twitter.com/TunisieEtudes
Notre page FaceBook :
http://www.facebook.com/TunisieEtudes



Merci d'avoir choisi www.tunisie-etudes.info
Bonne lecture et bon travail

MATHÉMATIQUES DE L'INFORMATIQUE

Lorsque l'on trie (c'est-à-dire lorsque l'on range par ordre croissant) n éléments pris dans un ensemble totalement ordonné, il est fréquent que, par suite de traitements préalables, l'ordre initial sur ces éléments ne soit pas "aléatoire". Par conséquent, les théorèmes généraux sur la qualité des algorithmes de tri ne s'appliquent plus.

On étudie ici un certain nombre de paramètres pouvant être utilisés comme "mesures (ou degrés) de préclassement" des suites de valeurs qui sont fournies en entrées aux algorithmes de tri. Puis, dans les dernières questions, on analyse un algorithme de tri "adaptatif" au sens où cet algorithme est particulièrement efficace sur des entrées ayant un fort degré de préclassement et on montre que cet algorithme est "optimal" en un certain sens.

Soit $X = (x_1, x_2, \ldots, x_n)$ une suite de n éléments d'un ensemble totalement ordonné $\mathcal{U}$. Une telle suite pourra également être notée par le mot $X = x_1 x_2 \ldots x_n$. L'entier n est la *longueur* de X. Pour simplifier, et comme il est justifié de le faire lors de l'analyse d'algorithmes qui ne procèdent que par comparaison de valeurs, on supposera souvent que les éléments x_i sont des entiers distincts strictement positifs. On s'intéressera tout particulièrement au cas où les x_i $(1 \leq i \leq n)$ sont les n premiers entiers auquel cas X sera identifiée à une permutation σ de S_n, le groupe symétrique sur $\{1, \ldots, n\}$.

Une suite $X = (x_1, x_2, \ldots, x_n)$ est *croissante* (resp. *décroissante*) si $x_i < x_{i+1}$ (resp. $x_i > x_{i+1}$) pour $i = 1, 2, \ldots, n-1$.

Pour tout ensemble E, on notera $\|E\|$ le cardinal de E. On notera également $\|X\|$ la longueur d'une suite X.

On note $\binom{n}{k}$ le coefficient binomial $\binom{n}{k} = \frac{n!}{k!(n-k)!}$. On désigne par $\log_2 n$ le logarithme de n en base 2. Le symbole de Kronecker $\delta_{n,k}$ vaut 1 si $n = k$ et 0 sinon.

1.– On appelle *nombre d'inversions* de X la quantité $Inv(X)$ définie par:

$$Inv(X) = \|\{(i,j) \mid 1 \leq i < j \leq n,\ x_i > x_j\}\|$$

1.1.– Que vaut $Inv(X)$ lorsque X est une suite croissante? Que vaut $Inv(X)$ lorsque X est une suite décroissante?

1.2.– On pose $Inv^+ = \max_{\sigma \in \mathrm{S}_n}\{Inv(\sigma)\}$. Déterminer

$$Inv_{n,+} = \{\sigma \mid \sigma \in \mathrm{S}_n,\ Inv(\sigma) = Inv^+\}$$

1.3.– On pose $Inv^- = \min_{\sigma \in \mathrm{S}_n}\{Inv(\sigma)\}$. Déterminer

$$Inv_{n,-} = \{\sigma \mid \sigma \in \mathrm{S}_n,\ Inv(\sigma) = Inv^-\}$$

1.4.– Soit $Inv_{n,k}$ le nombre de permutations σ de S_n telles que $Inv(\sigma) = k$ et soit $P_n(t) = \sum_k Inv_{n,k} t^k \in \mathbb{Z}[t]$. Montrer que:

$$P_n(t) = (1+t)(1+t+t^2)\ldots(1+t+\ldots+t^{n-1})$$

1.5.– En supposant les $n!$ permutations de S_n équiprobables, calculer la "valeur moyenne du paramètre Inv", à savoir le nombre:

$$\overline{Inv_n} = \frac{1}{n!}\sum_k k Inv_{n,k}$$

.2.– On appelle *nombre de descentes* de X la quantité $Desc(X)$ définie par:

$$Desc(X) = \|\{i \mid 1 \leq i < n,\ x_i > x_{i+1}\}\|$$

2.1.– Que vaut $Desc(X)$ lorsque X est une suite croissante? Que vaut $Desc(X)$ lorsque X est une suite décroissante?

2.2.– On pose $Desc^+ = \max_{\sigma \in \mathrm{S}_n}\{Desc(\sigma)\}$. Déterminer

$$Desc_{n,+} = \{\sigma \mid \sigma \in \mathrm{S}_n,\ Desc(\sigma) = Desc^+\}$$

2.3.– On pose $Desc^- = \min_{\sigma \in \mathrm{S}_n}\{Desc(\sigma)\}$. Déterminer

$$Desc_{n,-} = \{\sigma \mid \sigma \in \mathrm{S}_n,\ Desc(\sigma) = Desc^-\}$$

2.4.– On appelle *nombres eulériens* les entiers $A(n,k)$, $(n,k \in \mathbb{N})$ solutions de:

$$A(n,k) = kA(n-1,k) + (n-k+1)A(n-1,k-1) \quad (n,k > 0)$$

$$A(0,k) = \delta_{0,k} \quad (k \geq 0)$$

$$A(n,0) = \delta_{0,n} \quad (n \geq 0)$$

Tournez la page S.V.P.

Montrer que la "série génératrice double" des nombres eulériens

$$\mathcal{A}(t,u) = 1 + \sum_{n\geq 1}\sum_{k\geq 1} A(n,k)\frac{t^n}{n!}u^k$$

a pour expression

$$\mathcal{A}(t,u) = \frac{1-u}{1-ue^{t(1-u)}}$$

2.5.– Soit $Desc_{n,k}$ le nombre de permutations σ de S_n telles que $Desc(\sigma) = k$. Quelle relation existe-t-il entre les $Desc_{n,k}$ et les nombres eulériens?

2.6.– En supposant les $n!$ permutations de S_n équiprobables, calculer la valeur moyenne du paramètre *Desc*:

$$\overline{Desc_n} = \frac{1}{n!}\sum_k k Desc_{n,k}$$

3.– Une *sous-suite* Y de $X = (x_1, x_2, \ldots, x_n)$ est une suite

$$Y = (x_{f(1)}, x_{f(2)}, \ldots, x_{f(m)})$$

où $f : \{1,\ldots,m\} \to \{1,\ldots,n\}$ est injective et croissante. On considère la famille $I(X)$ des sous-suites croissantes de X (la suite vide, de longueur 0, est une sous-suite de X croissante).

On note $cpgssc(X)$ le cardinal de la plus grande sous-suite croissante de X, c'est-à-dire:

$$cpgssc(X) = \max\{\|Y\| \mid Y \in I(X)\}$$

et

$$Supp(X) = n - cpgssc(X)$$

On note $nbssc(X)$ le nombre de sous-suites croissantes de X, c'est-à-dire:

$$nbssc(X) = \|I(X)\|$$

3.1.– Que vaut $Supp(X)$ lorsque X est une suite croissante? Que vaut $Supp(X)$ lorsque X est une suite décroissante?

3.2.– On pose $Supp^+ = \max_{\sigma\in S_n}\{Supp(\sigma)\}$. Déterminer

$$Supp_{n,+} = \{\sigma \mid \sigma \in S_n,\ Supp(\sigma) = Supp^+\}$$

3.3.– On pose $Supp^- = \min_{\sigma\in S_n}\{Supp(\sigma)\}$. Déterminer

$$Supp_{n,-} = \{\sigma \mid \sigma \in S_n,\ Supp(\sigma) = Supp^-\}$$

3.4.– Que vaut $nbssc(X)$ lorsque X est une suite croissante? Que vaut $nbssc(X)$ lorsque X est une suite décroissante?

3.5.– On pose $nbssc^+ = \max_{\sigma\in S_n}\{nbssc(\sigma)\}$. Déterminer

$$nbssc_{n,+} = \{\sigma \mid \sigma \in S_n,\ nbssc(\sigma) = nbssc^+\}$$

3.6.– On pose $nbssc^- = \min_{\sigma\in S_n}\{nbssc(\sigma)\}$. Déterminer

$$nbssc_{n,-} = \{\sigma \mid \sigma \in S_n,\ nbssc(\sigma) = nbssc^-\}$$

3.7.– En supposant les $n!$ permutations de S_n équiprobables, montrer que le nombre moyen de sous-suites croissantes d'une permutation σ de S_n vaut:

$$\overline{nbssc_n} = \sum_{0\leq k\leq n}\binom{n}{k}\frac{1}{k!}$$

4.– Y est dite une *sous-suite consécutive* de X, de longueur m, s'il existe un i $(1 \leq i \leq n-m+1)$ tel que $Y = x_i x_{i+1}\ldots x_{i+m-1}$. L'entier i est le *début* de la sous-suite.

Y est une sous-suite consécutive croissante *maximale* si

- $x_{i-1}x_i\ldots x_{i+m-1}$ n'est pas croissante ou $i = 1$, et
- $x_i x_i \ldots x_{i+m-1}x_{i+m}$ n'est pas croissante ou $i = n-m+1$.

Une sous-suite consécutive croissante maximale est appelée une *monotonie*.

On note $nmon(X)$ le *nombre de monotonies* de X.

4.1.– Que vaut $nmon(X)$ lorsque X est une suite croissante? Que vaut $nmon(X)$ lorsque X est une suite décroissante?

4.2.– En supposant les $n!$ permutations de S_n équiprobables, calculer la valeur moyenne du paramètre *nmon*.

4.3.– On note $M_{n,k}$ le nombre total de monotonies de longueur k contenues dans toutes les permutations σ de S_n.

Tournez la page S.V.P.

4.3.1.– On note $N_{n,k,l}$ le nombre de permutations $\sigma = \sigma_1\sigma_2\ldots\sigma_n$ de S_n telles que l'élément σ_l est le début d'une monotonie de longueur supérieure ou égale à k. Montrer que:

$$N_{n,k,l} = n!\frac{k+\delta_{l,1}}{(k+1)!} \quad \text{si} \quad l \leq n-k+1$$

$$N_{n,k,l} = 0 \quad \text{sinon}$$

4.3.2.– En déduire que:

$$M_{n,k} = n!\left((n+1)\frac{k^2+k-1}{(k+2)!} - \frac{k^2-k-1}{(k+1)!}\right) \quad \text{si} \quad 1 \leq k \leq n-1$$

$$M_{n,n} = 1$$

4.3.3.– En supposant les $n!$ permutations de S_n équiprobables, calculer la valeur moyenne du paramètre "nombre de monotonies de longueur k".

4.3.4.– Montrer que la longueur moyenne d'une monotonie a pour valeur asymptotique $2 + O(1/n)$.

5.– Etant donné une permutation $\sigma = \sigma_1\sigma_2\ldots\sigma_n$ de S_n, on dit que $i \in \{1,\ldots,n-1\}$ est une *succession* si $\sigma_{i+1} = \sigma_i + 1$. On note $Succ(\sigma)$ le nombre de successions de σ.

5.1!– Que vaut $Succ(\sigma)$ lorsque σ est la permutation associée à une suite croissante? Que vaut $Succ(\sigma)$ lorsque σ est la permutation associée à une suite décroissante?

5.2.– On pose $Succ^+ = \max_{\sigma\in S_n}\{Succ(\sigma)\}$. Déterminer

$$Succ_{n,+} = \{\sigma \mid \sigma \in S_n,\ Succ(\sigma) = Succ^+\}$$

5.3.– On pose $Succ^- = \min_{\sigma\in S_n}\{Succ(\sigma)\}$. Déterminer

$$Succ_{n,-} = \{\sigma \mid \sigma \in S_n,\ Succ(\sigma) = Succ^-\}$$

5.4.– Soit D_n le nombre de *dérangements* de S_n, c'est-à-dire le nombre de permutations σ de S_n telles que $\sigma_i \neq i$ pour tout i de $\{1,\ldots,n\}$. Montrer que la "série génératrice exponentielle de dénombrement" des dérangements de S_n a pour expression:

$$\sum_{n\geq 0} D_n\frac{t^n}{n!} = \frac{e^{-t}}{1-t}$$

5.5.– Soit $Succ_{n,k}$ le nombre de permutations σ de S_n telles que $Succ(\sigma) = k$. Montrer que, pour $n \geq 1$:

$$Succ_{n,k} = \binom{n-1}{k}[D_{n-k} + D_{n-k-1}]$$

5.6.– Calculer une expression de la série génératrice double de dénombrement des permutations par nombre de successions:

$$\sum_{n\geq 0,\, k\geq 0} Succ_{n,k}\frac{t^n}{n!}u^k$$

5.7.– En supposant les $n!$ permutations de S_n équiprobables, montrer que le nombre moyen de successions d'une permutation de S_n a pour valeur asymptotique 1.

6.– Etant donné une suite $X = x_1\ldots x_n$, on définit, pour tout i de $\{1,\ldots,n\}$, l'ensemble $Crois(x_i)$ des *croisements* de x_i par:

$$Crois(x_i) = \{j \mid 1 \leq j < n, \min\{x_j, x_{j+1}\} < x_i < \max\{x_j, x_{j+1}\}\}$$

On appelle *nombre d'oscillations* de X la quantité $Osc(X)$ définie par:

$$Osc(X) = \sum_{i=1}^{n} \|Crois(x_i)\|$$

6.1.– Que vaut $Osc(X)$ lorsque X est une suite croissante? Que vaut $Osc(X)$ lorsque X est une suite décroissante?

6.2.– Calculer $Osc^+ = \max_{\sigma\in S_n} Osc(\sigma)$.

6.3.– On pose $Osc^- = \min_{\sigma\in S_n} Osc(\sigma)$. Déterminer

$$Osc_{n,-} = \{\sigma \mid \sigma \in S_n,\ Osc(\sigma) = Osc^-\}$$

6.4.– Montrer que pour toute suite X de longueur n:

$$Osc(X) \leq 4Inv(X)$$

6.5.– Montrer que pour toute suite X de longueur n:

$$Osc(X) \leq 2nDesc(X) + n$$

Tournez la page S.V.P.

7.– Les paramètres *Inv*, *Desc* et *Osc* étudiés plus haut sont souvent considérés comme de "bons" critères de "préclassement".

On appelle *mesure de préclassement* toute fonction rélle M définie sur l'ensemble des suites à éléments dans $\mathcal{U}$. Soit A un algorithme de tri qui exécute $T_A(X)$ comparaisons pour trier une suite X. On dit que l'algorithme A est *M-optimal* ou *optimal vis-à-vis de la mesure M* s'il existe une constante $c > 0$ telle que, pour toute suite $X = x_1 x_2 \dots x_n$:

$$T_A(X) \leq c \ \max\{\|X\|,\ log(\|sous(X, M)\|)\}$$

où

$$sous(X, M) = \{\sigma \mid \sigma \in \mathrm{S}_n,\ M(\sigma(X)) \leq M(X)\}$$

avec $\sigma(X) = x_{\sigma_1} x_{\sigma_2} \dots x_{\sigma_n}$, et $log(n) = \max\{1, \log_2 n\}$.

7.1.– Commenter le choix d'une telle définition en situant en particulier la notion introduite par rapport à la notion d'algorithme de tri optimal.

7.2.– X étant une suite de longueur n, montrer que $log(\|sous(X, Osc)\|)$ est borné inférieurement par $c\, n\, log(Osc(X)/n)$ pour un $c > 0$.

(On pourra utiliser, pour n divisant p, les ensembles

$$A_i = \left\{1 + i\left(\frac{p}{n} - 2\right), 2 + i\left(\frac{p}{n} - 2\right), \dots, \left(\frac{p}{n} - 2\right) + i\left(\frac{p}{n} - 2\right)\right\}$$

pour construire une suite Y telle que $\|sous(Y, Osc)\| \geq (p/n - 2)!^{\lceil n^2/(p-2n)\rceil}$.)

7.3.– En déduire que tout algorithme de tri opérant par comparaisons effectue au moins $c\, n\, log(Osc(X)/n)$ comparaisons pour trier une suite X de longueur n.

8.– On rappelle que l'algorithme de *Tri_en_Tas* ("Heapsort" en anglais) est un algorithme de tri dont le temps de calcul (compté en nombre de comparaisons) est $\Theta(n\, logn)$ (c'est-à-dire est borné inférieurement et supérieurement par des quantités de la forme $c\, n\, logn$). Le *Tri_en_Tas* est donc un algorithme optimal dans la classe des algorithmes qui procèdent par comparaison. L'algorithme *Tri_en_Tas* commence, dans un premier temps, par construire une structure de données appelée *tas* ("heap" en anglais) qui contient les n éléments à trier puis, dans un deuxième temps, effectue l'opération *Supprime_Max* (suppression de l'élément maximal) n fois sur la structure. L'opération *Supprime_Max* peut être exécutée en temps logarithmique en la taille du tas.

On considère ici une variante de l'algorithme *Tri_en_Tas*, appelée *Tri_par_Arbre_Max*.

L'*arbre maximal* d'une suite $X = x_1 x_2 \dots x_n$ est l'arbre binaire construit récursivement de la manière suivante: sa racine est

$$x_i = \max\{x_1, x_2, \dots, x_n\}$$

son sous-arbre gauche est l'arbre maximal de la suite $x_1 x_2 \dots x_{i-1}$, son sous-arbre droit est l'arbre maximal de la suite $x_{i+1} \dots x_n$; enfin, l'arbre maximal de la suite vide est l'arbre binaire vide.

8.1.– Donner un algorithme de construction de l'arbre maximal d'une suite X donnée par la suite de ses éléments. Analyser son temps de construction dans le cas le plus défavorable.

On admettra qu'il existe un algorithme linéaire de construction de l'arbre maximal d'une suite.

On supposera dans la suite que l'arbre maximal a été construit en temps linéaire.

8.2.– L'algorithme *Tri_en_Tas* peut être décrit comme suit:

Procédure *Tri_par_Arbre_Max*
  Construire l'arbre maximal de X
  pour i **de** 1 **à** n **faire**
    Extraire de l'arbre maximal les éventuels nouveaux "max-candidats"
    Insérer le(s) nouveau(x) max-candidat(s) dans le tas
    Exécuter *Supprime_Max* sur le tas
  finfaire

Les *max-candidats* sont, à chaque étape, les fils dans l'arbre maximal du dernier sommet supprimé dans le tas. Au début, ce sont les fils de la racine.

Montrer que le nombre d'éléments dans le tas, quand x_i est supprimé, est inférieur ou égal à $\lfloor Osc(x_i)/2 \rfloor + 2$.

8.3.– Montrer que sur une entrée $X = x_1 x_2 \dots x_n$, l'algorithme *Tri_par_Arbre_Max* effectue au plus $O(n + \sum_i log(\|Osc(x_i)\|))$ comparaisons.

8.4.– Montrer que *Tri_par_Arbre_Max* est un algorithme optimal vis-à-vis de la mesure de préclassement *Osc*.

9.– Soit $X = x_1 x_2 \dots x_n$ une suite de longueur n.

9.1.– Montrer que tout algorithme de tri opérant par comparaisons effectue au moins $c\, n\, log(Inv(X)/n)$ comparaisons pour trier X.

9.2.– Montrer que tout algorithme de tri opérant par comparaisons effectue au moins $c\, n\, log(Desc(X))$ comparaisons pour trier X.

9.3.– Montrer que *Tri_par_Arbre_Max* est un algorithme optimal vis-à-vis de chacune des mesures de préclassement *Inv*, *Desc* et *Osc*.

www.tunisie-etudes.info

Ce document a été téléchargé depuis
www.tunisie-etudes.info

Des documents gratuits, devoirs, examens, cours, exercices, corrigés... Ainsi que toute une rubrique pour vous aider à trouver un emploi sans oublier les avis de concours en direct

Notre page Twitter :
http://www.twitter.com/TunisieEtudes
Notre page FaceBook :
http://www.facebook.com/TunisieEtudes



Merci d'avoir choisi www.tunisie-etudes.info
Bonne lecture et bon travail