Tri par segmentation (Quicksort)

Concepts : Analyse de coût, coût moyen,
Méthodes : Probabilité, espérance, log

Présentation

Le problème est de trier les éléments d'un ensemble de taille n. Les éléments de l'ensemble sont tous comparables deux à deux (unicité des clés) et on suppose qu'il n'y a pas de doublons (ordre total). L'algorithme consiste à choisir un élément pivot, segmenter l'ensemble par rapport à la valeur du pivot puis à trier récursivement les éléments plus petits, puis plus grand et assembler l'ensemble

TriSegmentation(E)
Données : Un ensemble E de n éléments comparables
Résultat : Les éléments par ordre croissant
if $E \neq \emptyset$
  Pivot = **Extrait** (E) `// retire l'élément pivot de` E
  (E_1, E_2) = **Segmentation** (E,pivot) `//` E_1 `(resp.` E_2`) éléments plus petits (resp. plus grand) que pivot`
  return Assemble (**TriSegmentation** (E_1),*pivot,* **TriSegmentation** (E_2))

Algorithme 1: Algorithme du tri par segmentation

La preuve de cet algorithme est laissée en exercice.

Complexité

L'objectif est d'analyser le coût de cet algorithme en nombre de comparaisons effectuées entre des éléments. Une première étape consiste à se donner des exemples pour comprendre le déroulement de l'algorithme. Ensuite on recherche des exemples présentant des "cas extrêmes" au pire et au mieux. Puis on formalise le problème en écrivant les équations vérifiées par le coût de l'algorithme. On résout ces équations, ou bien, si on ne trouve pas de solution on essaye de majorer/minorer le coût (en ordre de grandeur). Si il y a une grande différence entre le coût au pire et le coût au mieux (par exemple s'il ne sont pas sur la même échelle), on évalue l'ordre de grandeur du coût moyen.

Etape 1 Faire une trace de l'algorithme sur l'exemple $E = \{4, 3, 6, 7, 2, 5, 1\}$.
L'opération extraire prend les éléments de E dans l'ordre, le pivot étant le premier élément de l'ensemble. Total 11 comparaisons (voir la trace d'exécution en appendice). On sent que l'algorithme va être efficace si on arrive à segmenter E en 2 ensembles de même taille.

Etape 2 Exemples extrémaux.
Un premier cas consiste à segmenter au mieux l'ensemble, c'est à dire $|E1| = |E2| = \frac{|E|-1}{2}$, par exemple si E={4,2,1,3,6,5,7} on fera 6+2.2=10 comparaisons. Pour un ensemble de taille $n = 2^k - 1$ notre exemple ferait de l'ordre de $k2^k$ comparaisons (voir plus loin), soit de l'ordre de $n \log n$.
A contrario si la segmentation est très déséquilibrée, par exemple pour E={1,2,3,4,5,6,7} on aura 6+5+4+3+2+1=21 comparaisons. Pour le même exemple de taille n on aurait $1+2+\cdots+n-1 = \frac{n(n-1)}{2}$.
On peut remarquer que sur 2 exemples on obtient des valeurs de coût ayant des ordres de grandeur différents, ce qui encourage à pousser davantage l'analyse du coût.

Tri par segmentation

L'opération de traitement du maximum courant est supposée coûteuse, on s'intéresse donc au nombre d'appels à cette fonction durant l'exécution de l'algorithme.

Il est clair que le nombre d'opérations Traiter est au mieux égal à 1 si le plus grand élément du tableau est en première position et il y a $(n-1)!$ configurations du tableau telles que le $\max$ soit en première position. D'autre part, l'algorithme effectue au pire n opérations de traitement dans le cas où tous les éléments sont rangés par ordre croissant.

Etape 3 Formalisation.

Notons $C(E)$ le coût de l'algorithme en nombre de comparaisons pour l'entrée E de taille n. L'opération de segmentation nécessite $n-1$ opérations donc le coût de l'algorithme vérifie

$$C(E) = n - 1 + C(E1) + C(E2).$$

avec $C(E) = 0$ si $n = 0$ ou 1. Evidemment ce coût dépend fortement du choix du pivot, c'est à dire de la taille respective de $E1$ et $E2$. Les seules contraintes que l'on connaisse sur $E1$ et $E2$ sont

$$0 \leqslant |E1| \leqslant n-1;\ 0 \leqslant |E2| \leqslant n-1;\ |E1| + |E2| = n-1.$$

Comme d'habitude, notons

$$C_{max}(n) = \max_{|E|=n} C(E) \text{ et } C_{min}(n) = \min_{|E|=n} C(E).$$

Etape 4 Majorants/minorants.

Comme on ne peut pas calculer facilement $C_{max}(n)$, montrons par récurrence que $C_{max}(n) \leqslant n^2$.

En effet, cette assertion est vraie pour $n = 0$, supposons-la vérifiée pour tout k avec $0 \leqslant k \leqslant n-1$ et soit E une instance réalisant le coût maximum. On a

$$C_{max}(n) = C(E) = n-1+C(E1)+C(E2) \leqslant n-1+\max_{0\leqslant k\leqslant n-1} \{C_{max}(k) + C_{max}(n-1-k)\}.$$

On applique l'hypothèse de récurrence

$$C_{max}(n) \leqslant n-1+\max_{0\leqslant k\leqslant n-1} k^2 + (n-1-k)^2 = (n-1)^2 \leqslant n-1+(n-1)^2 = n(n-1) \leqslant n^2.$$

(la fonction $x^2 + (n-1-x)^2$ est une parabole tournée vers le haut donc maximale sur l'une des extrémité de l'intervalle $[0, n-1]$.) La propriété est donc héréditaire, on en déduit que pour tout n

$$C_{max}(n) \leqslant n^2,$$

comme il existe un exemple dont le coût est $\frac{n(n-1)}{2}$ en en conclut que

$$C_{max}(n) = \Theta(n^2)$$

Exercice : Montrer par récurrence que $C_{max}(n) = \frac{n(n-1)}{2}$.

Pour le coût minimal on raisonne de la même manière en montrant que $C_{min}(n) \geqslant \frac{1}{2} n \log n$

Etape 5 Analyse en moyenne.

Pour faire l'analyse en moyenne on suppose que les permutations ont toutes la même probabilité. Cela revient à choisir unifrmément le pivot parmi les n éléments de E.

Ceci nous donne (en utilisant la linéarité de la moyenne)

$$C_{moy}(n) = \mathbb{E}C(E) = n - 1 + \mathbb{E}[C(E1)] + \mathbb{E}[C(E2)];$$

Tri par segmentation

On conditionne par la taille de $E1$ (valeur de la position du pivot dans E moins 1)

$$C_{moy}(n) = n - 1 + \sum_{i=0}^{n-1}(C_{moy}(i) + C_{moy}(n-1-i))\frac{1}{n};$$

en simplifiant la somme

$$C_{moy}(n) = n - 1 + \frac{2}{n}\sum_{i=0}^{n-1} C_{moy}(i).$$

On en déduit, en utilisant la même équation de récurrence au rang $n-1$:

$$\begin{aligned} C_{moy}(n) &= n - 1 + \frac{2}{n}C_{moy}(n-1) + \frac{2}{n}\sum_{i=0}^{n-2}(C_{moy}(i); \\ &= n - 1 + \frac{2}{n}C_{moy}(n-1) + \frac{2}{n}\frac{n-1}{2}\left(C_{moy}(n-1) - (n-2)\right); \\ &= (n-1)\frac{2}{n} + \frac{n+1}{n}C_{moy}(n-1). \end{aligned}$$

En divisant par $n+1$ on déduit

$$\frac{C_{moy}(n)}{n+1} = \frac{4}{n+1} - \frac{2}{n} + \frac{C_{moy}(n-1)}{n} = \sum_{i=1}^{n+1}\frac{2}{i} \simeq 2\log(n+1).$$

D'où

$$C_{moy}(n) = \mathcal{O}(n\log n);$$

ce qui est excellent puisque proche (et au moins sur la même échelle) de la complexité du problème du tri.

Autre approche à partir de variables indicatrices, soit $X_{i,j}$ la variable booléenne indiquant si les objets de rang i et j ont été comparés.

Soit C_n la variable aléatoire indiquant le nombre de comparaisons effectuées par l'algorithme. Clairement

$$C_n = \sum_{1\leqslant i<j\leqslant n} X_{i,j};\ \text{d'où}\ \mathbb{E}C_n = \sum_{1\leqslant i<j\leqslant n}\mathbb{E}X_{i,j}.$$

Il reste donc à calculer $\mathbb{E}, X_{i,j} = \mathbb{P}(X_{i,j} = 1)$. Or pour que l'élément de rang i et l'élément de rang j soient comparés il faut que le pivot ne soit jamais choisi entre l'élément de rang i et de rang j avant le choix de i ou de j. Comme le choix du pivot est uniforme, soit i soit j est choisi comme pivot parmi les éléments de rang i à j donc avec la probabilité

$$\mathbb{P}(X_{i,j} = 1) = \frac{2}{|j-i+1|}.$$

on en déduit

$$\mathbb{E}C_n = \sum_{1\leqslant i<j\leqslant n}\frac{2}{|j-i+1|} = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{2}{|j-i+1|} = \sum_{i=1}^{n-1}\sum_{k=2}^{n-i+1}\frac{2}{k} \leqslant 2\sum_{i=1}^{n-1}H_i \leqslant 2nH_n$$

et finalement

$$\mathbb{E}C_n = \mathcal{O}(n\log n).$$

Tri par segmentation

Trace d'exécution

pivot ← 4
E1 ← {3,2,1}
E2 ← {6,7,5}
⇒ 6 comparaisons
tri-rapide(E1)
  pivot ← 3
  E1 ← {2,1}
  E2 ← {}
  ⇒ 2 comparaisons
  tri-rapide(E1)
    pivot ← 2
    E1 ← {1}
    E2 ← {}
    ⇒ 1 comparaisons
    tri-rapide(E1)
      pivot ← 1
      E1 ← {}
      E2 ← {}
      ⇒ 0 comparaisons
    tri-rapide(E2)
      ⇒ 0 comparaisons
  tri-rapide(E2)
    ⇒ 0 comparaisons
tri-rapide(E2)
  pivot ← 6
  E1 ← {5}
  E2 ← {7}
  ⇒ 2 comparaisons

Références

[1] Thomas Cormen, Charles Leiserson, Ronald Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3 edition, 2009.