Appunti Lezione – Capitolo 17
Algoritmi probabilistici

Alberto Montresor

11 Maggio, 2020

1 Domanda: Selezione a partire da mediana?

Potrei utilizzare l'algoritmo della mediana per scegliere il perno in perno(), e quindi ottenere sempre partizioni di dimensione $n/2$. A quel punto, la divisione darebbe sempre origine al caso migliore:

$$T(n) = T(n/2) + O(n) = O(n)$$

2 Domanda: Selezione a partire da mediana approssimata

Potrei utilizzare l'algoritmo della mediana approssimata per scegliere il pivot in perno(), e quindi ottenere partizioni che sono al più $\frac{3}{4}n$. A questo punto la divisione darebbe origine alla seguente equazione di ricorrenza:

$$T(n) = T(3n/4) + O(n)$$

Per il Master Theorem, otteniamo che la complessità è comunque $O(n)$.

3 Algoritmo in tempo deterministico lineare

Lemma 1

E' possibile dimostrare il Lemma 1 scrivendo un semplice algoritmo che trova ogni mediana di 5 elementi con al più 6 confronti. Un algoritmo (lungo, ma semplice) è il seguente:

```
int median5(int[] A)
  int m1, m2, m3                                      % 1°, 2°, 3° minimo
  if A[1] < A[2] then
      if A[2] < A[3] then
          m1 ← A[1]; m2 ← A[2]; m3 ← A[3]
      else
          m1 ← A[1]; m2 ← A[3]; m3 ← A[2]
  else
      if A[1] < A[3] then
          m1 ← A[2]; m2 ← A[1]; m3 ← A[3]
      else
          m1 ← A[2]; m2 ← A[3]; m3 ← A[1]
  if A[4] < m2 then
      if A[4] < m1 then
          m3 ← m2; m2 ← m1; m1 ← A[4]
      else
          m3 ← m2; m2 ← A[4]
  else
      if A[4] < m3 then
          m3 ← A[4]
  if A[5] < m2 then
      if A[5] < m1 then
          m3 ← m2; m2 ← m1; m1 ← A[5]
      else
          m3 ← m2; m2 ← A[5]
  else
      if A[5] < m3 then
          m3 ← A[5]
  return m3
```

Si noti che questo algoritmo ordina vettori di 5 elementi; la versione chiamata dagli algoritmi sui lucidi opera su sottovettori di 5 elementi all'interno di un vettore più grande, e nel caso dell'ultimo gruppo il numero di elementi può essere inferiore a 5; bisogna tenere conto di questi problemi per poter scrivere una versione completa di median5().

Se ci accontentiamo di un limite più alto, ma di un codice più semplice, è possibile vedere che è possibile utilizzare InsertionSort sui 5 elementi e poi ritornare il terzo. Il numero di confronti effettuati da InsertionSort su 5 elementi nel caso pessimo è pari a:

$$\sum_{i=1}^{4} i = 10$$

Altrimenti è possibile modificare SelectionSort per cercare 3 minimi invece che 5; in questo caso, il costo è pari a $4 + 3 + 2 = 9$ confronti.

Lemma 2

Facile, per definizione dei mediani.

Lemma 3

E' facile vedere che il mediano dei mediani è più alto o uguale di $\lceil\lceil n/5\rceil/2\rceil$ altri mediani; ognuno di questi mediani è più alto o uguale a 3 elementi, in ognuno dei gruppi di 5 elementi in cui è diviso il vettore A. Quindi, il mediano dei mediani è più alto o uguale di $3\lceil\lceil n/5\rceil/2\rceil$ elementi nel vettore.
Nel caso pessimo, la chiamata ricorsiva verra effettuata su $n-3\lceil\lceil n/5\rceil/2\rceil$ elementi, un valore che è limitato superiormente da $7n/10$.

3.1 Errata corrige del libro rispetto all'algoritmo di selezione

Per trovare il k-esimo elemento più piccolo senza ordinare il vettore, si può utilizzare la procedura perno() [cfr. § **??**]. Ricordiamo che la procedura $\mathsf{perno}(A, \mathit{primo}, \mathit{ultimo})$ scambia tra loro elementi di $A[\mathit{primo} \ldots \mathit{ultimo}]$ e restituisce l'indice j del "perno" tale che $A[i] \leq A[j]$, per $\mathit{primo} \leq i \leq j$, e $A[i] \geq A[j]$, per $j \leq i \leq \mathit{ultimo}$. La seguente procedura selezione(), proposta da Hoare (1961), richiama la perno() ed individua così il q-esimo elemento più piccolo di $A[\mathit{primo} \ldots \mathit{ultimo}]$, dove $q = j - \mathit{primo} + 1$ è il numero di elementi di $A[\mathit{primo} \ldots j]$. Se $k \leq q$, allora selezione() è riapplicata ad $A[\mathit{primo} \ldots j]$; se invece $k > q$, allora selezione() è riapplicata ad $A[j+1 \ldots \mathit{ultimo}]$, ma per ricercare il $(k-q)$-esimo elemento più piccolo. La chiamata iniziale è $\mathsf{selezione}(A, 1, n, k)$.

```
ITEM selezione(ITEM[] A, int primo, int ultimo, int k)
if primo = ultimo then
    return A[primo]
else
    int j ← perno(A, primo, ultimo)
    int q ← j − primo + 1
    if k = q then
        return A[j]
    else if k < q then
        return selezione(A, primo, j − 1, k)
    else
        return selezione(A, j + 1, ultimo, k − q)
```

Assumendo che perno() restituisca con la stessa probabilità una qualsiasi posizione j del vettore A, il numero medio di confronti $T(n)$ tra elementi di A effettuati da selezione() è dato dalla relazione di ricorrenza

$$
\begin{aligned}
T(n) &= 0, && \text{per } n = 0, 1 \\
T(n) &= n + \frac{1}{n} \sum_{q=1}^{n} T\left(\max\{q-1, n-q\}\right) \\
&\leq n + \frac{2}{n} \sum_{q=\lfloor n/2 \rfloor}^{n-1} T(q), && \text{per } n \geq 2
\end{aligned}
$$

Infatti, nel caso n sia pari, è facile vedere che i valori $\lfloor n/2\rfloor, \lfloor n/2\rfloor + 1, \ldots, n-1$ vengono ottenuti due volte dall'espressione $\max\{q-1, n-q\}$, per $q = 1 \ldots n$; se invece n è dispari, i valori $\lfloor n/2\rfloor + 1$, $\lfloor n/2\rfloor + 2, \ldots, n-1$ vengono ottenuti due volte, mentre $\lfloor n/2\rfloor$ viene ottenuto una volta sola – da cui la disequazione.
Effettuiamo un ragionamento informale per cercare di ricavare una buona soluzione sulla quale applicare la tecnica di dimostrazione per tentativi. In media, selezione() dovrebbe essere richiamata ogni volta su una porzione dimezzata del vettore A. Poiché la perno() richiede tempo lineare nella lunghezza della porzione

di A, il numero complessivo di confronti dovrebbe crescere come $n + n/2 + n/4 + \cdots + 1 \leq 2n$. Pertanto, una soluzione promettente di $T(n)$ con la quale tentare sembra essere $O(n)$.
Tentiamo quindi con $T(n) \leq cn$ per qualche costante c. La base dell'induzione è banalmente verificata perché $0 = T(0) = T(1) \leq c$. Si assuma che valga l'ipotesi induttiva $T(n') \leq cn'$ per ogni $n' \leq n-1$. Sostituendo, si ricava:

$$
\begin{aligned}
T(n) &\leq n + \frac{1}{n} \sum_{q=\lfloor n/2 \rfloor}^{n-1} 2 \cdot cq \leq n + \frac{2c}{n} \sum_{q=\lfloor n/2 \rfloor}^{n-1} q && \text{Sostituzione, raccolgo } 2c \\
&\leq n + \frac{2c}{n} \sum_{q=n/2-1}^{n-1} q && \text{Estensione sommatoria} \\
&= n + \frac{2c}{n} \left(\sum_{q=1}^{n-1} q - \sum_{q=1}^{n/2-2} q \right) && \text{Sottrazione prima parte} \\
&= n + \frac{2c}{n} \cdot \left(\frac{n(n-1)}{2} - \frac{(n/2-1)(n/2-2)}{2} \right) \\
&= n + \frac{\cancel{2}c}{n} \cdot \frac{\left(n^2 - n - \left(1/4\, n^2 - 3/2\, n + 2\right)\right)}{\cancel{2}} \\
&= n + c/n \cdot \left(3/4\, n^2 + 1/2\, n - 2\right) \\
&\leq n + c/n \cdot \left(3/4\, n^2 + 1/2\, n\right) = n + 3/4\, cn + 1/2 \stackrel{?}{\leq} cn && \text{Vera per } c \geq 6 \text{ e } n \geq 1
\end{aligned}
$$

Pertanto, la funzione selezione() trova il k-esimo elemento più piccolo di A in tempo lineare nel caso medio.