CALCOLO DEI PERCENTILI
CALCOLO DELLE MEDIANE
STIMA DELLA DENSITÀ

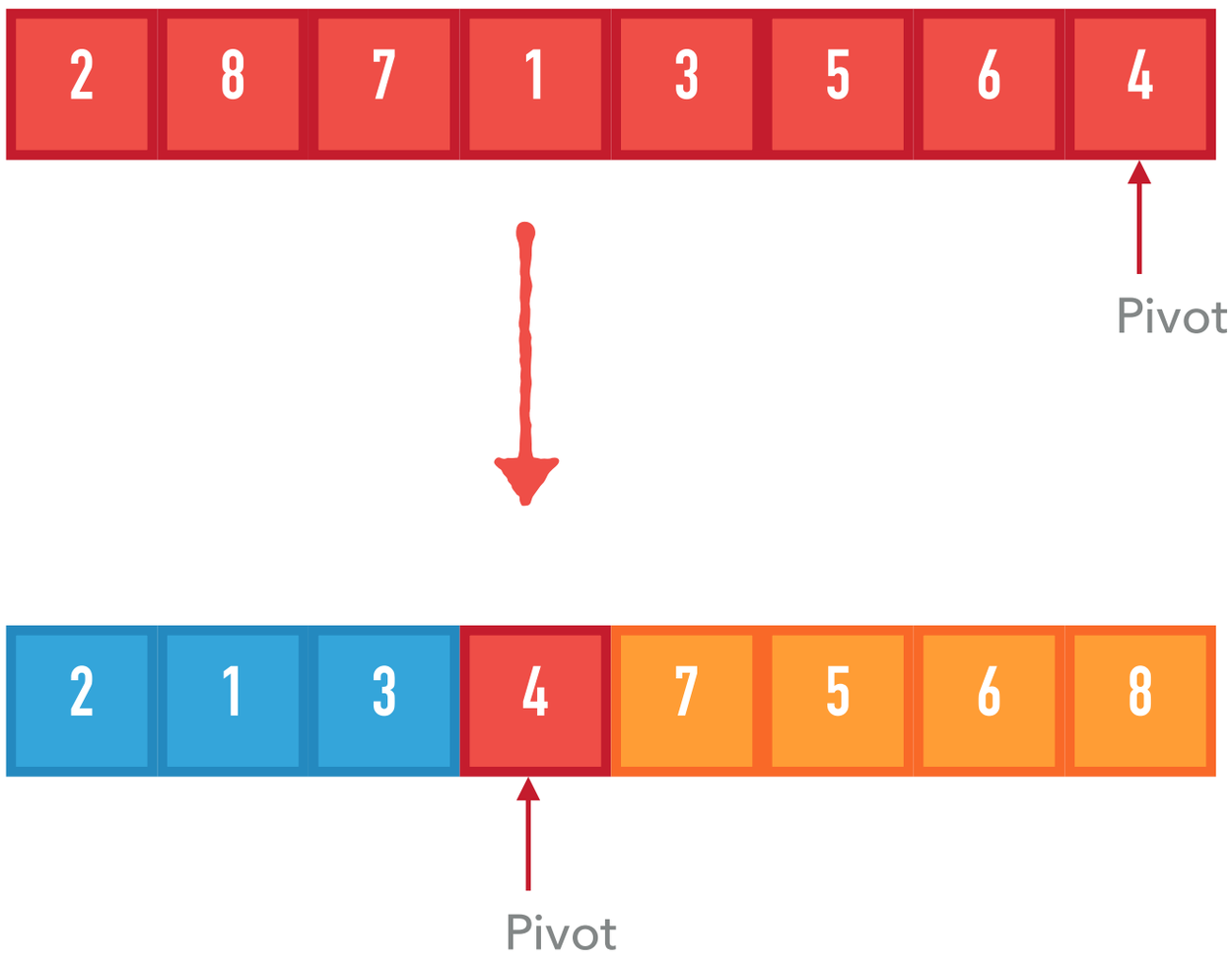
INFORMATICA

PERCENTILI

PERCENTILI

- Dato un array A di numeri (o elementi totalmente ordinabili), il p-esimo percentile è quel numero A[j] tale che p% dei rimanenti numeri dell'array sono minori di A[j].
- Il 50-simo percentile è la **mediana**.
- Un modo per trovare il p-simo percentile è ordinare l'array in senso crescente e ritornare A[j], con $j = \left\lceil \frac{p}{100} len(A) \right\rceil$
- Il costo di questo approccio è $O(n \log n)$.
- Trovare il massimo o il minimo costa $O(n)$. Esiste un metodo migliore per calcolare un singolo percentile?

QUICKSELECT



Supponiamo di cercare il 75simo percentile p.

Lanciamo Partition. Dove si trova p rispetto al pivot?

QUICKSELECT

QUICKSELECT

```
QuickSelect(A, l, r, p)
    if l = r return A[l]
    q = Partition(A, l, r)
    i = q-l+1   //p è relativo a r-l+1, q a len(A)
    if p=i return A[q]
    else if p < i return QuickSelect(A, l, q-1, p)
    else return  QuickSelect(A, q, r, p-i)
```

QUICKSELECT

```
QuickSelect(A, l, r, p)
    if l = r return A[l]
    q = Partition(A, l, r)
    i = q-l+1   //p è relativo a r-l+1, q a len(A)
    if p=i return A[q]
    else if p < i return QuickSelect(A, l, q-1, p)
    else return  QuickSelect(A, q, r, p-i)
```

Correttezza: per induzione

QUICKSELECT

```
QuickSelect(A, l, r, p)
    if l = r return A[l]
    q = Partition(A, l, r)
    i = q-l+1   //p è relativo a r-l+1, q a len(A)
    if p=i return A[q]
    else if p < i return QuickSelect(A, l, q-1, p)
    else return  QuickSelect(A, q, r, p-i)
```

Correttezza: per induzione

Ci sono tre casi da considerare rispetto alla relazione tra *p* e i = *q-l+1:*

- *p< i:* il percentile sta tra l e q-1
- p = i: il percentile è proprio A[q]
- p> i: il percentile sta tra q+1 e r.

QUICKSELECT – COMPLESSITÀ

QUICKSELECT – COMPLESSITÀ

```
QuickSelect(A, l, r, p)
    if l = r return A[l]
    q = Partition(A, l, r)
    i = q-l+1   //p è relativo a r-l+1, q a len(A)
    if p=i return A[q]
    else if p < i return QuickSelect(A, l, q-1, p)
    else return  QuickSelect(A, q, r, p-i)
```

QUICKSELECT – COMPLESSITÀ

```
QuickSelect(A, l, r, p)
    if l = r return A[l]
    q = Partition(A, l, r)
    i = q-l+1   //p è relativo a r-l+1, q a len(A)
    if p=i return A[q]
    else if p < i return QuickSelect(A, l, q-1, p)
    else return  QuickSelect(A, q, r, p-i)
```

Caso peggiore

QUICKSELECT – COMPLESSITÀ

```
QuickSelect(A, l, r, p)
    if l = r return A[l]
    q = Partition(A, l, r)
    i = q-l+1   //p è relativo a r-l+1, q a len(A)
    if p=i return A[q]
    else if p < i return QuickSelect(A, l, q-1, p)
    else return  QuickSelect(A, q, r, p-i)
```

Caso peggiore

Pivot ritorna una partiziona in due array di lunghezza 0 e n-1,
ed il percentile sta in quello più lungo:
$T(n) = T(n-1) + O(n) \quad => \quad T(n) = O(n^2)$

QUICKSELECT – COMPLESSITÀ MEDIA

$$T(n) \leq \frac{1}{n}\left(\sum_{k=\lceil \frac{n}{2} \rceil}^{n-1} T(k)\right) + \Theta(n)$$

QUICKSELECT – COMPLESSITÀ MEDIA

Caso ottimo

$$T(n) \leq \frac{1}{n}\left(\sum_{k=\lceil \frac{n}{2} \rceil}^{n-1} T(k)\right) + \Theta(n)$$

QUICKSELECT - COMPLESSITÀ MEDIA

Caso ottimo

Pivot ritorna una partiziona in due array di lunghezza n/2:

$$T(n) = T\left(\frac{n}{2}\right) + \Theta(n) \quad => \quad T(n) = \Theta(n)$$

$$T(n) \leq \frac{1}{n}\left(\sum_{k=\lceil \frac{n}{2} \rceil}^{n-1} T(k)\right) + \Theta(n)$$

QUICKSELECT - COMPLESSITÀ MEDIA

Caso ottimo

Pivot ritorna una partiziona in due array di lunghezza n/2:

$$T(n) = T\left(\frac{n}{2}\right) + \Theta(n) \quad => \quad T(n) = \Theta(n)$$

Caso medio

$$T(n) \leq \frac{1}{n}\left(\sum_{k=\lceil \frac{n}{2} \rceil}^{n-1} T(k)\right) + \Theta(n)$$

QUICKSELECT - COMPLESSITÀ MEDIA

Caso ottimo

Pivot ritorna una partiziona in due array di lunghezza n/2:

$$T(n) = T\left(\frac{n}{2}\right) + \Theta(n) \quad => \quad T(n) = \Theta(n)$$

Caso medio

Assumiamo la variante randomized-select (pivot scelto a caso).
Partition ritorna due partizioni di lunghezza k e n-k-1 con probabilità 1/n.
Assumiamo il percentile stia sempre in quella più lunga.

$$T(n) \leq \frac{1}{n}\left(\sum_{k=\lceil \frac{n}{2} \rceil}^{n-1} T(k)\right) + \Theta(n)$$

QUICKSELECT – COMPLESSITÀ MEDIA

$$T(n) \leq \frac{1}{n}\left(\sum_{k=\lceil \frac{n}{2} \rceil}^{n-1} ck\right) + dn$$

$$\frac{1}{n}\left(\sum_{k=\lceil \frac{n}{2} \rceil}^{n-1} ck\right) + dn \leq \frac{c}{n}\left(\sum_{k=1}^{n-1} k\right) + dn = \frac{c}{2n}\left(n^2 - n\right) + dn = \frac{cn}{2} - \frac{c}{2} + dn$$

QUICKSELECT – COMPLESSITÀ MEDIA

Siamo ottimisti ed assumiamo che la complessità media vada come quella ottima. Dimostriamo per sostituzione che $T(n) \leq cn$ per qualche $c>0$.

$$T(n) \leq \frac{1}{n}\left(\sum_{k=\lceil \frac{n}{2} \rceil}^{n-1} ck\right) + dn$$

$$\frac{1}{n}\left(\sum_{k=\lceil \frac{n}{2} \rceil}^{n-1} ck\right) + dn \leq \frac{c}{n}\left(\sum_{k=1}^{n-1} k\right) + dn = \frac{c}{2n}\left(n^2 - n\right) + dn = \frac{cn}{2} - \frac{c}{2} + dn$$

QUICKSELECT – COMPLESSITÀ MEDIA

Siamo ottimisti ed assumiamo che la complessità media vada come quella ottima. Dimostriamo per sostituzione che $T(n) \leq cn$ per qualche *c>0*.

$$T(n) \leq \frac{1}{n}\left(\sum_{k=\lceil \frac{n}{2} \rceil}^{n-1} ck\right) + dn$$

$$\frac{1}{n}\left(\sum_{k=\lceil \frac{n}{2} \rceil}^{n-1} ck\right) + dn \leq \frac{c}{n}\left(\sum_{k=1}^{n-1} k\right) + dn = \frac{c}{2n}\left(n^2 - n\right) + dn = \frac{cn}{2} - \frac{c}{2} + dn$$

Segue $T(n) \leq cn$ per c > 2d

MEDIAN OF MEDIANS

Can one find a partition scheme that has worst case running time O(n)?

1. Divide the n elements of the input array into $\lfloor n/5 \rfloor$ groups of 5 elements each and at most one group made up of the remaining n mod 5 elements.
2. Find the median of each of the $\lceil n/5 \rceil$ groups by first insertion-sorting the elements of each group (of which there are at most 5) and then picking the median from the sorted list of group elements.
3. Use SELECT recursively to find the median x of the $\lceil n/5 \rceil$ medians found in step 2. (If there are an even number of medians, then by our convention, x is the lower median.)
4. Partition the input array around the median-of-medians x using the modified version of PARTITION. Let k be one more than the number of elements on the low side of the partition, so that x is the kth smallest element and there are $n-k$ elements on the high side of the partition.
5. If $i = k$, then return x. Otherwise, use SELECT recursively to find the ith smallest element on the low side if $i < k$, or the $(i-k)$th smallest element on the high side if $i > k$.

Why this works?

Intuitively, the approximate median found is always between the 30th and the 70th percentile: half medians of groups and 2 other elements of these groups are smaller

$$T(n) \le \begin{cases} O(1) & \text{if } n < 140\,, \\ T(\lceil n/5 \rceil) + T(7n/10 + 6) + O(n) & \text{if } n \ge 140\,. \end{cases}$$

Exercise: compare experimentally execution time of QuickSelect using random pivot selection vs median of medians

MEDIAN OF MEDIANS

MEDIAN OF MEDIANS

Esiste anche un algoritmo che ha complessità $\Theta(n)$ nel caso peggiore.

MEDIAN OF MEDIANS

Esiste anche un algoritmo che ha complessità $\Theta(n)$ nel caso peggiore.

MoM_Select(A, l, r, p)
    B = array di lunghezza $m = \lceil (r - l + 1)/5 \rceil$, il cui elemento j
        è la mediana del j-simo blocco di lunghezza 5 di A[p:r]
_,w = *MoM_Select*(B, 1, m, m/2)
swap(A[r], A[l+5w+2])
q = *Partition*(A, l, r)
if l = r **return** A[l]
i = q-l+1 *//p è relativo a r-l+1, q a len(A)*
if p=i **return** A[q]
else if p < i **return** *MoM_Select*(A, l, q-1, p)
else return *MoM_Select*(A, q, r, p-i)

MEDIAN OF MEDIANS

MEDIAN OF MEDIANS

Come costruisco B?

MEDIAN OF MEDIANS

Come costruisco B?

// B array di lunghezza $m = \lceil (r - l + 1)/5 \rceil$
for j,i in enumerate(range(l,r,5))
    insertion_sort(A,j,j+5)
    B[i] = A[j+2]

MEDIAN OF MEDIANS

Come costruisco B?

// B array di lunghezza $m = \lceil (r - l + 1)/5 \rceil$
for j,i in enumerate(range(l,r,5))
    insertion_sort(A,j,j+5)
    B[i] = A[j+2]

B è un array di mediane di lunghezza n/5 e cerco ricorsivamente la sua mediana. Poi uso questa mediana di mediane come elemento di pivot nella procedura di **select**.

La correttezza è identica a quella di **quick_select** - posso ignorare il passo di cercare la mediana di mediane.

MEDIAN OF MEDIANS – COMPLESSITÀ

MEDIAN OF MEDIANS - COMPLESSITÀ

MoM_Select si chiama ricorsivamente due volte, la prima su una istanza di dimensione n/5, la seconda sul risultato di partition.

MEDIAN OF MEDIANS – COMPLESSITÀ

MoM_Select si chiama ricorsivamente due volte, la prima su una istanza di dimensione n/5, la seconda sul risultato di partition.

Intuizione: la mediana di mediane x è più grande (piccola) della metà delle mediane dei gruppi di 5 elementi di A, che a loro volta sono più grandi (piccole) di altri due elementi del gruppo.

Quindi x è più grande (piccola) di $3 \cdot \frac{1}{2} \cdot \frac{n}{5} = \frac{3}{10}n$ elementi.

MEDIAN OF MEDIANS – COMPLESSITÀ

MoM_Select si chiama ricorsivamente due volte, la prima su una istanza di dimensione n/5, la seconda sul risultato di partition.

Intuizione: la mediana di mediane x è più grande (piccola) della metà delle mediane dei gruppi di 5 elementi di A, che a loro volta sono più grandi (piccole) di altri due elementi del gruppo.

Quindi x è più grande (piccola) di $3 \cdot \frac{1}{2} \cdot \frac{n}{5} = \frac{3}{10}n$ elementi.

Sempre intuitivamente: $T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + O(n)$

MEDIAN OF MEDIANS – COMPLESSITÀ

MoM_Select si chiama ricorsivamente due volte, la prima su una istanza di dimensione n/5, la seconda sul risultato di partition.

Intuizione: la mediana di mediane x è più grande (piccola) della metà delle mediane dei gruppi di 5 elementi di A, che a loro volta sono più grandi (piccole) di altri due elementi del gruppo.

Quindi x è più grande (piccola) di $3 \cdot \frac{1}{2} \cdot \frac{n}{5} = \frac{3}{10}n$ elementi.

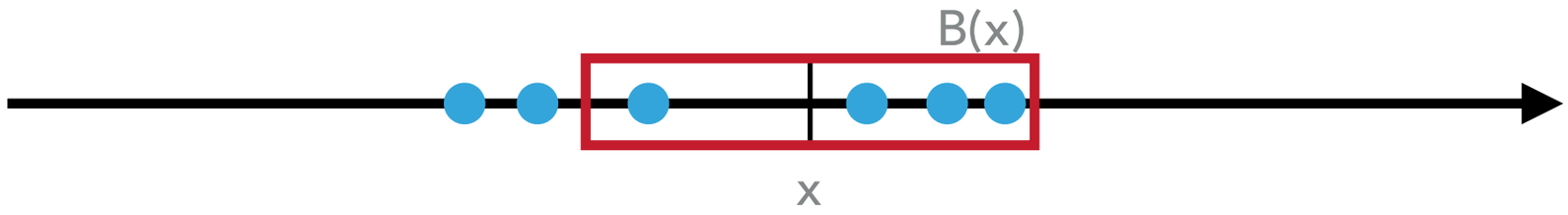
Sempre intuitivamente: $T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + O(n)$

Per sostituzione $T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + O(n)$ implica $T(n) = O(n)$

K-NN DENSITY ESTIMATION

Dati N osservazioni (in una dimensione) di una variabile continua, possiamo stimare la densità in x come p(x) = K/NV(x), dove V(x) è il volume della palla B(x)=[x-a,x+a] centrata in x che contiene esattamente K punti, ovvero 2a in questo caso.

Come possiamo calcolare efficientemente V(x), per ogni x?



Come possiamo gestire lo scenario online, in cui nuovi punti arrivano di tanto in tanto?