Selezione

Il problema della Selezione

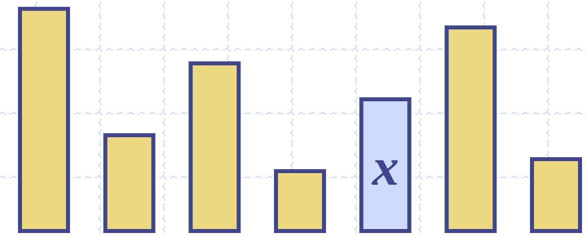
- Dato un intero k e n elementi $x_1, x_2, \ldots, x_n$, estratti da un ordine totale, trova il k-th elemento nell'insieme.
- Naturalmnte, possiamo ordinare l'insieme in tempo O(n log n) e quindi trovare il k-esimo elemento.

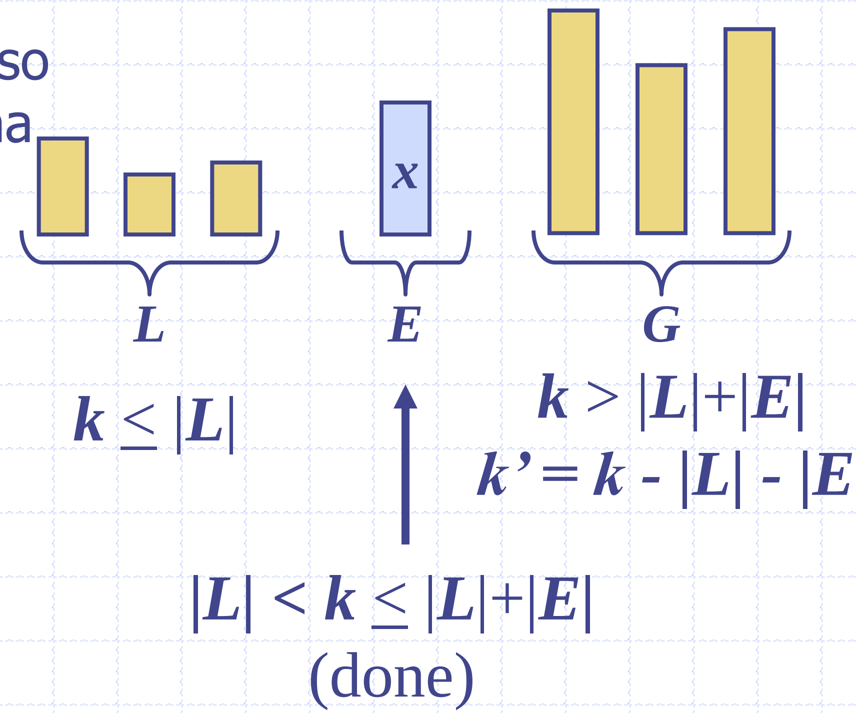
k=3 7 4 9 <u>6</u> 2 → 2 4 <u>6</u> 7 9

- Possiamo risolvere il problema piu' velocemente ?

Quick-Select (§ 11.7)

- Quick-select e' un algoritmo a **randomizzato di** selezione basato sul paradigma prune-and-search:
  - Prune: prendi un elemento a caso x (chiamato pivot) and partiziona S in
    - L elementi minori di x
    - E elementi eguali a x
    - G elementi maggiori di x
  - Search: sulla base di k, rispondi E, o ricorri in L o G





Partizione

- Partizioniamo una sequenza di input come nell'algoritmo di quick-sort:
  - Rimuoviamo ogni elemento y da S e
  - Inseriamo y in L, E or G, sulla base del risultato del confronto con il pivot x
- Ogni inserimento e rimozione e' all'inizio o alla fine della sequenza, e quindi richiede tempo $O(1)$
- Quindi, la partizione di quick-select richiede tempo $O(n)$

Algorithm *partition*(S, p)
    Input sequence S, position p of pivot
    Output subsequences L, E, G of the elements of S less than, equal to, or greater than the pivot, resp.
    $L, E, G \leftarrow$ empty sequences
    $x \leftarrow S.remove(p)$
    while $\neg S.isEmpty()$
        $y \leftarrow S.remove(S.first())$
        if $y < x$
            $L.insertLast(y)$
        else if $y = x$
            $E.insertLast(y)$
        else { $y > x$ }
            $G.insertLast(y)$
    return L, E, G

Visualizzazione di Quick-Select

- Un’esecuzione di quick-select puo’ essere vizualizzata attraverso un cammino di ricorsione
  - Ogni nodo rappresenta una chiamata ricorsiva di quick-select, e memorizza k e la sequenza rimanente

k=5, S=(7 4 9 3 2 6 5 1 8)

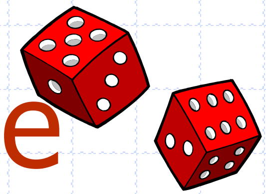
k=2, S=(7 4 9 6 5 8)

k=2, S=(7 4 6 5)

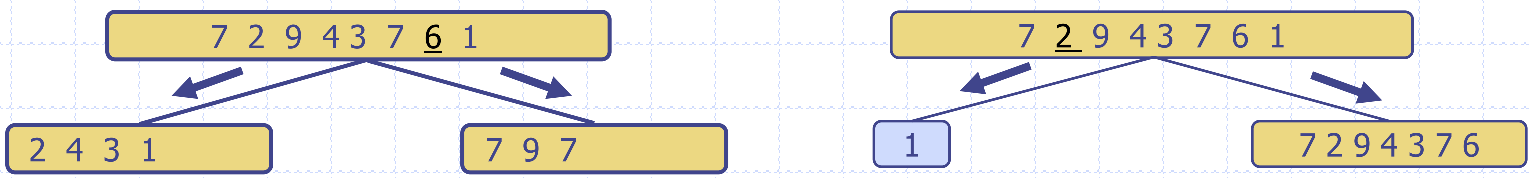
k=1, S=(7 6 5)

5

Tempo atteso di esecuzione

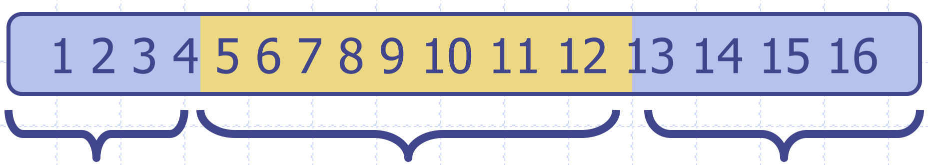


- Considera una chiamata ricorsiva di quick-select su una sequenza di dimensione s
  - **Good call:** le dimensioni di L e G sono minori di $3s/4$
  - **Bad call:** L o G hanno dimensione maggiore di $3s/4$



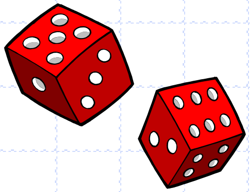
Good call **Bad call**

- Una chiamata e' good con probabilita' 1/2
  - 1/2 dei possibili pivot determina good calls:



Bad pivots **Good pivots** **Bad pivots**

Tempo atteso di esecuzione, Parte 2



- Fatto #1: Il numero atteso di lanci di monete richiesto per ottenere una testa e' 2
- Fatto #2: Il valore atteso e' uan funzione lineare:
  - $\boldsymbol{E(X + Y) = E(X) + E(Y)}$
  - $\boldsymbol{E(cX) = cE(X)}$
- Sia T(n) il tempo atteso di esecuzione di quick-select.
- Dal Fatto #2,
  - $T(n) \leq T(3n/4) + bn$*(# atteso di chiamate prima di una good call)
- Dal Fatto #1,
  - $T(n) \leq T(3n/4) + 2bn$
- Cioe', T(n) e' una serie geometrica: $(3/4)^2 n + 2b(3/4)^3 n + \ldots$
- Quindi T(n) e' O(n).
- Possiamo risolvere il problema della selezione in tempo atteso O(n).

Selezione Deterministica

- Possiamo eseguire la selezione in tempo O(n) anche nel caso peggiore.
- Idea: ricorsivamente usa l'algoritmo di selezione per trovare un buon pivot per quick-select:
  - Dividi S in n/5 insiemi di 5 elementi ciascuno
  - Trova il mediano in ogni insieme
  - Ricorsivamente trova il mediano tra i mediani degli insiemi di 5 elementi