

Selezione del k-esimo minimo

Gianluigi Zavattaro
Dip. di Informatica – Scienza e Ingegneria
Università di Bologna
gianluigi.zavattaro@unibo.it

Slide realizzate a partire da materiale fornito dal Prof. Moreno Marzolla

Original work Copyright © Alberto Montresor, University of Trento
(<http://www.dit.unitn.it/~montreso/asd/index.shtml>)

Modifications Copyright © 2009, 2010, Moreno Marzolla, Università di Bologna
(<http://www.moreno.marzolla.name/teaching/ASD2010/>)

This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/2.5/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.

Selezione del k-esimo minimo

- Consideriamo il seguente problema:
 - **Selezione del k-esimo minimo**: dato un array $A[1..n]$ di valori distinti e un valore $1 \leq k \leq n$, trovare l'elemento che è maggiore di esattamente $k-1$ elementi
- Caso particolare:
 - **Mediano**: il valore che occuperebbe la posizione $(n/2)$ se l'array fosse ordinato

Selezione del k-esimo minimo: a che serve?

Google [Ricerca avanzata](#)
[Preferenze](#)

Cerca: ☒ nel Web ☐ pagine in Italiano ☐ pagine provenienti da: Italia

Web Risultati 1 - 10 su circa **104.000** per algoritmi e strutture dati. (0,53 secondi)

Algoritmi e Strutture Dati
Ingegneria degli **Algoritmi** (Parte I). Contenuti: In questo modulo imparerai a progettare ed analizzare **algoritmi e strutture dati**. Docente: ...
[gauguin.info.uniroma2.it/~italiano/.../Algoritmi/](#) - [Copia cache](#) - [Simili](#)

Corso di Algoritmi e Strutture Dati
Il corso ha lo scopo di presentare i concetti fondamentali riguardanti l'analisi e il progetto di **algoritmi e strutture dati** efficienti. ...
[www.dit.unitn.it/~montreso/asd/index.shtml](#) - [Copia cache](#) - [Simili](#)

Sito "Algoritmi e strutture dati - Progetto di algoritmi e ..."
Progetto di **algoritmi e strutture dati** in Java. Camil Demetrescu, Irene Finocchi, Giuseppe F. Italiano, Umberto Ferraro Petrillo ...
[www.ateneonline.it/demetrescu/](#) - [Copia cache](#) - [Simili](#)

Sito "Algoritmi e strutture dati 2/ed", Camil Demetrescu, Irene ...
Questo libro offre un'introduzione allo studio degli **algoritmi** e delle **strutture dati**, cercando di conciliare comprensibilità, chiarezza di esposizione e ...
[www.ateneonline.it/demetrescu2e/homeA.asp](#) - [Copia cache](#) - [Simili](#)

Appunti di Algoritmi E Strutture Dati - Studenti.it
26 feb 2007 ... Tutte le risorse per **algoritmi e strutture dati**. Tantissimi riassunti e tesine appunti completamente gratis.
[www.studenti.it/appunti/.../algoritmi-e-strutture-dati/](#) - [Copia cache](#) - [Simili](#)

Algoritmi e strutture dati
M. Torelli, Appunti per il corso di **Algoritmi e Strutture Dati** ... C. Demetrescu, I. Finocchi, G. Italiano **Algoritmi e strutture dati**, McGraw-Hill, 2004. ...
[homes.dsi.unimi.it/~goldwurm/algo/](#) - [Copia cache](#) - [Simili](#)

Corso di Algoritmi e Strutture Dati
Corso di **Algoritmi e Strutture Dati**. Dott. Giorgio Terracina. A.A. 2008/2009. (Ultimo aggiornamento: 15/06/2009). AVVISO: Risultati dell'ultimo appello ...
[www.mat.unical.it/terracina/asd/](#) - [Copia cache](#) - [Simili](#)

Lezioni di algoritmi e strutture dati - Maniezzo Vittorio; Margara ...
Libro di Maniezzo Vittorio; Margara Luciano, Lezioni di **algoritmi e strutture dati**, Pitagora.
[www.ibs.it/code/.../lezioni-algoritmi-strutture.html](#) - [Copia cache](#) - [Simili](#)

Algoritmi e Strutture Dati I
[www.di.unipi.it/didadoc/asd1/](#) - [Copia cache](#) - [Simili](#)

Selezione del k-esimo minimo: a che serve?

- I motori di ricerca producono molti risultati a fronte di una singola query
- I risultati vengono mostrati in **pagine**, in ordine decrescente di rilevanza
 - Nella prima pagina i risultati più rilevanti
 - Nella seconda quelli meno, e così via
- È inutile ordinare tutti i risultati in base alla rilevanza
 - Quanti vanno frequentemente oltre la quarta pagina di risultati della ricerca?
- È quindi utile selezionare i primi k risultati, e via via i successivi, se l'utente seleziona le altre pagine

Selezione: casi particolari

Ricerca del minimo

```
algorithm minimum(array A[1..n]) → elem  
  min := A[1];  
  for i := 2 to n do  
    if (A[i] < min) then  
      min = A[i];  
    endif  
  endfor  
  return min;
```

- $T(n) = n-1 = \Theta(n)$ confronti

Selezione: casi particolari

- Ricerca del secondo minimo
 - Trovare il secondo elemento più piccolo dell'array $A[]$
 - Costo: $2n-3$ confronti nel caso peggiore (il caso peggiore si verifica quando i valori sono in ordine decrescente)

```
algorithm minimum2(array  $A[1..n]$ )  $\rightarrow$  elem  
    min1 := A[1];  
    min2 := A[2];  
    if (min2 < min1) then  
        swap(min1, min2);  
    endif  
    for i:= 3 to n do  
        if (A[i] < min2) then  
            min2 = A[i];  
            if (min2 < min1) then  
                swap(min1, min2);  
            endif  
        endif  
    endfor  
    return min2;
```

Selezione del k-esimo minimo

```
algorithm select(array A[1..n], int k) → elem
  for i:=1 to k do
    minIndex := i;
    minValue := A[i];
    for j:=i+1 to n do
      if (A[j] < minValue) then
        minIndex := j;
        minValue := A[j];
      endif
    endfor
    swap A[i] and A[minIndex];
  endfor
  return A[k];
```

- In sostanza un Selection Sort incompleto
 - Si ferma al k-esimo elemento
- Costo $\Theta(kn)$

Selezione per piccoli valori di k

- Costruisco un min-heap a partire dai valori
 - Costo $O(n)$
- Estraggo per $k-1$ volte il minimo
 - Costo $O(k \log n)$
- Il k -esimo minimo è l'elemento minimo che rimane
 - Costo complessivo: $O(n + k \log n)$

```
algorithm heapselect(array  $A[1..n]$ , int  $k$ )  $\rightarrow$  elem  
  min-heapify( $A$ );  
  for  $i := 1$  to  $k-1$  do  
    deleteMin( $A$ );  
  endfor  
  return findMin( $A$ );
```

Questa funzione
costruisce un min-heap

Esempio

- Estrarre i primi k elementi da una query che ha fornito n match costa $O(n + k \log n) = O(n)$ se k è $O(n/\log n)$
 - Esempio: k=10, n=104000

The screenshot shows a Google search interface with the query "algoritmi e strutture dati". The search bar includes a "Cerca" button and links to "Ricerca avanzata" and "Preferenze". Below the search bar, there are radio buttons for "nel Web", "pagine in Italiano", and "pagine provenienti da: Italia". The search results are displayed under the heading "Web" and show "Risultati 1 - 10 su circa 104.000 per algoritmi e strutture dati. (0,53 secondi)".

The first result is titled "Algoritmi e Strutture Dati" and is from the website "gauguin.info.uniroma2.it/~italiano/.../Algoritmi/". It describes a module for learning to design and analyze algorithms and data structures.

The second result is titled "Corso di Algoritmi e Strutture Dati" and is from the website "www.dit.univr.it/~montreso/asd/index.shtml". It describes a course that presents fundamental concepts regarding analysis and the design of efficient algorithms and data structures.

The third result is titled "Sito 'Algoritmi e strutture dati - Progetto di algoritmi e ...'" and is from the website "www.ateneonline.it/demetrescu/". It describes a project of algorithms and data structures in Java, by Camil Demetrescu, Irene Finocchi, Giuseppe F. Italiano, Umberto Ferraro Petrillo, and others.

The fourth result is titled "Sito 'Algoritmi e strutture dati 2/ed', Camil Demetrescu, Irene ..." and is from the website "www.ateneonline.it/demetrescu2/homeA.asp". It describes a book that offers an introduction to the study of algorithms and data structures, aiming to provide a clear and accessible exposition.

The fifth result is titled "Appunti di Algoritmi E Strutture Dati - Studenti.it" and is from the website "www.studenti.it/appunti/.../algoritmi-e-strutture-dati/". It describes a collection of notes and exercises for the course of algorithms and data structures, available for free.

The sixth result is titled "Algoritmi e strutture dati" and is from the website "homes.dsi.unimi.it/~goldwurm/algo/". It describes a course of algorithms and data structures, by M. Torelli, Camil Demetrescu, I. Finocchi, G. Italiano, and others.

The seventh result is titled "Corso di Algoritmi e Strutture Dati" and is from the website "www.mat.unical.it/terraccina/asd/". It describes a course of algorithms and data structures, by Dott. Giorgio Terraccina, A.A. 2008/2009.

The eighth result is titled "Lezioni di algoritmi e strutture dati - Maniezzo Vittorio, Margara ..." and is from the website "www.its.it/codex/.../lezioni-algoritmi-strutture.html". It describes a book of algorithms and data structures, by Maniezzo Vittorio, Margara Luciano, and Pitagora.

The ninth result is titled "Algoritmi e Strutture Dati" and is from the website "www.di.unipi.it/didattoc/asd1/". It describes a course of algorithms and data structures, by the University of Pisa.

Esempio

- Le cose vanno meno bene se k è del medesimo ordine di grandezza di n
- Esempio: voglio calcolare il valore mediano
 - Cioè il valore che occuperebbe la posizione centrale se l'array fosse ordinato
- In questo caso $k = n/2$, e per valori sufficientemente grandi di n il costo è

$$O(n + k \log n) = O(n + (n/2) \log n) = O(n \log n)$$

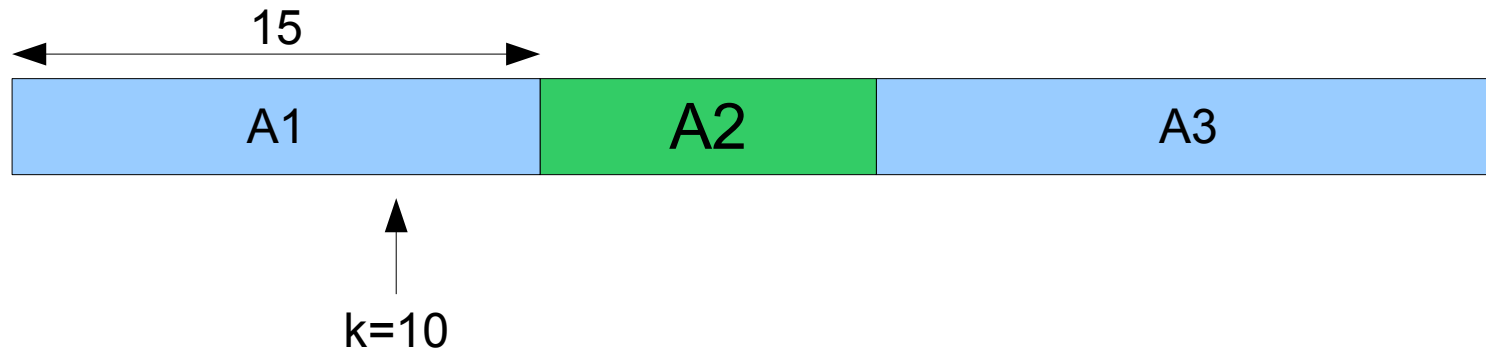
Adattamento di quicksort al problema della selezione

```
algorithm select1( array A[1..n], int k ) -> elem
  scegli un elemento x in A
  A1 := {y in A: y < x }
  A2 := {y in A: y = x }
  A3 := {y in A: y > x }
  quicksort(A1);
  quicksort(A3);
  ritorna il k-esimo elemento della concatenazione di A1, A2, A3
```

- Idea

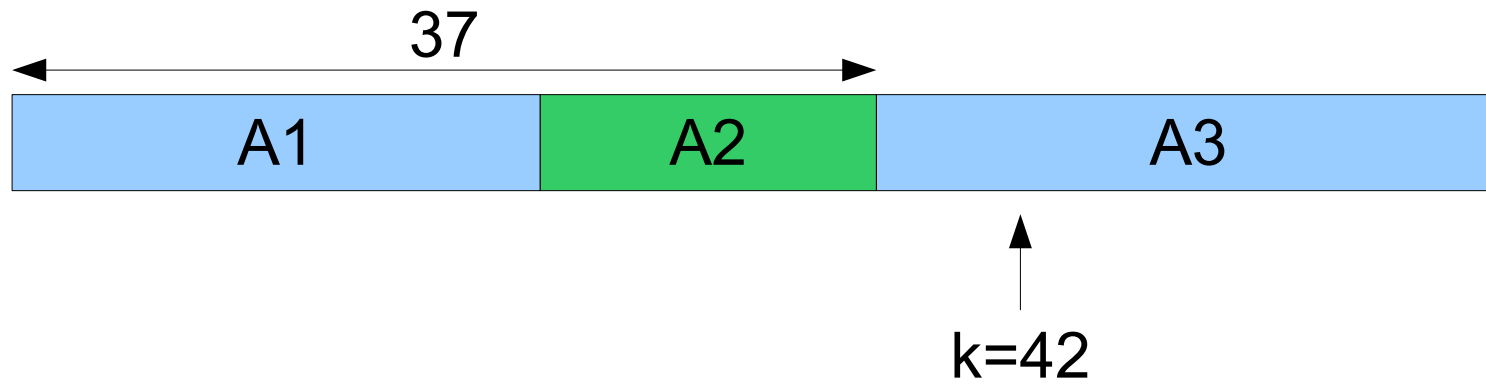
- Approccio divide-et-impera simile al QuickSort...
- ...però essendo un problema di ricerca, non è necessario cercare in tutte le partizioni, basta cercare in una sola

Adattamento di quicksort al problema della selezione



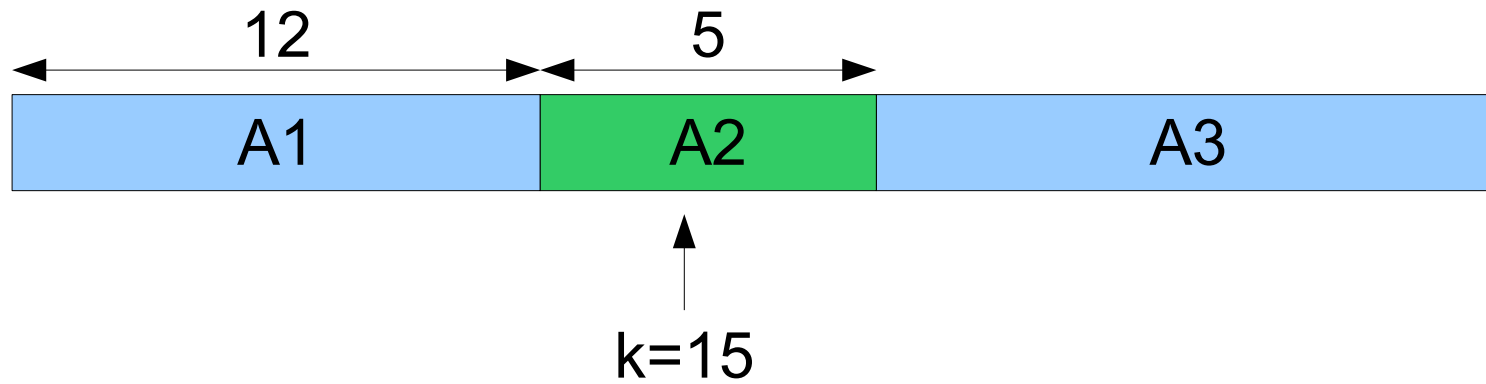
- In realtà non serve considerare tutto il vettore!
 - Supponiamo di cercare il 10mo minimo;
 - Supponiamo che A1 abbia 15 elementi
 - Il valore cercato sicuramente sta in A1

Adattamento di quicksort al problema della selezione



- In realtà non serve considerare tutto il vettore!
 - Supponiamo di cercare il 42mo minimo;
 - Supponiamo che A1 e A2 abbiano 37 elementi
 - Il valore cercato è il $(42-37)=5$ di A3

Adattamento di quicksort al problema della selezione

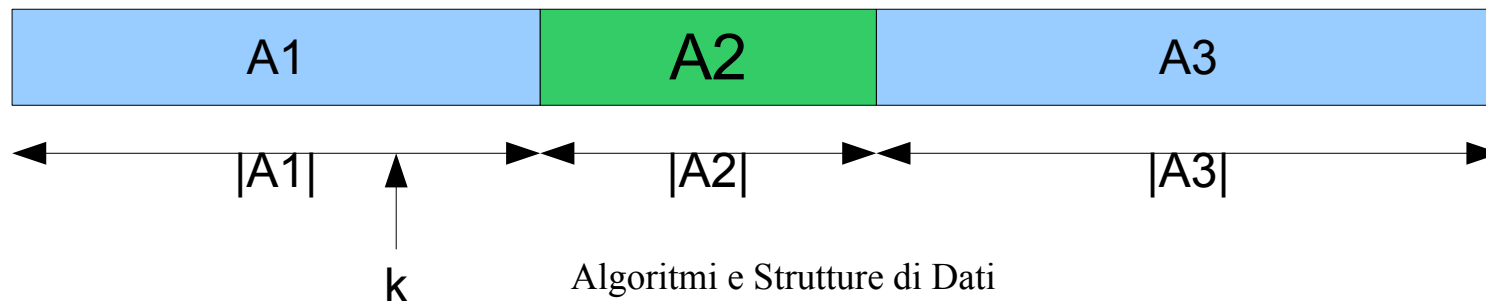


- In realtà non serve considerare tutto il vettore!
 - Supponiamo di cercare il 15mo minimo;
 - Supponiamo che A1 abbia 12 elementi e A2 ne abbia 5
 - Il valore cercato si trova in A2

Algoritmo quickSelect()

```
algorithm quickSelect( Array A, int k ) → elem  
  scegli un elemento x in A  
  A1 := {y in A: y < x }  
  A2 := {y in A: y = x }  
  A3 := {y in A: y > x }  
  if ( k ≤ |A1| ) then  
    return quickSelect( A1, k );  
  else  
    if ( k > |A1| + |A2| ) then  
      return quickSelect( A3, k - |A1| - |A2| );  
    else  
      return x;  
    endif  
  endif
```

A1, A2, e A3 possono essere determinati con l'algoritmo della "bandiera nazionale", una modifica della "partition" di quicksort che raggruppa i valori uguali al pivot sempre in O(n)



Esempio

(nota: il partizionamento è fatto in modo arbitrario)

↓ pivot

k=6

3	75	4	14	28	16	22	5	67	54	37	91	12	17
---	----	---	----	----	----	----	---	----	----	----	----	----	----

Cerchiamo
il k=5 in A3

3	75	4	14	28	16	22	5	67	54	37	91	12	17
---	----	---	----	----	----	----	---	----	----	----	----	----	----

k=5

↓

75	4	14	28	16	22	5	67	54	37	91	12	17
----	---	----	----	----	----	---	----	----	----	----	----	----

Cerchiamo
il k=5 in A1

4	14	28	16	22	5	67	54	37	12	17	75	91
---	----	----	----	----	---	----	----	----	----	----	----	----

k=5

↓


4	14	28	16	22	5	67	54	37	12	17
---	----	----	----	----	---	----	----	----	----	----

Cerchiamo
il k=4 in A3

4	14	28	16	22	5	67	54	37	12	17
---	----	----	----	----	---	----	----	----	----	----

Esempio (cont.)

k=4




14	28	16	22	5	67	54	37	12	17
----	----	----	----	---	----	----	----	----	----

Cerchiamo
il k=1 in A3

5	12	14	16	22	67	54	37	28	17
---	----	----	----	----	----	----	----	----	----

k=1



16	22	67	54	37	28	17
----	----	----	----	----	----	----

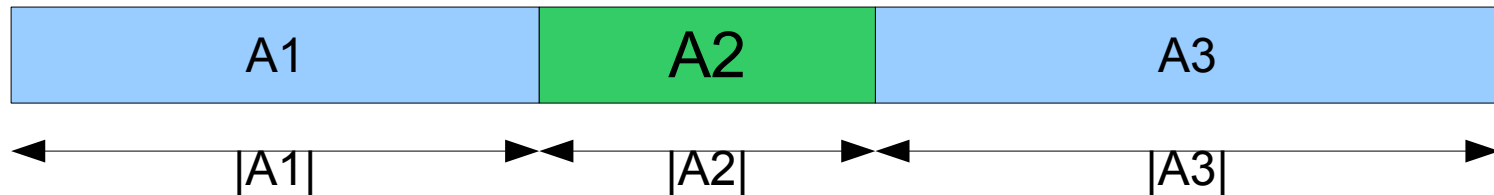
16	22	67	54	37	28	17
----	----	----	----	----	----	----

Trovato! il k=6
elemento di A è 16

Analisi dell'algoritmo quickSelect()

- Costo nel caso ottimo
 - $T(n) = \Theta(n)$
 - Esecuzione del partizionamento
- Costo nel caso pessimo
 - $T(n) = T(n-1) + n = \Theta(n^2)$
 - Dimostrazione come nel caso di Quick Sort
- ...e nel caso medio?

Analisi del caso medio



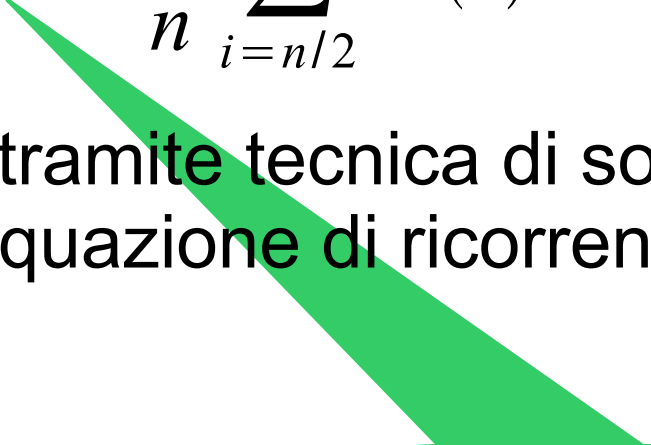
- Vedremo che, anche sotto ipotesi “pessimista”, il costo medio sarà lineare
- Ipotesi “pessimista”: si effettua chiamata ricorsiva su un vettore di lunghezza $\max(|A1|, |A3|)$
 - Scartiamo A2 (altrimenti termineremmo subito) e il sottovettore più corto tra A1 e A3
- Il numero di valori su cui si ricorre è fra $n/2$ e $n-1$:
 - Assumiamo uguale frequenza per ogni valore fra $n/2$ e $n-1$. Ogni valore appare quindi con frequenza $1/(n/2) = 2/n$

Analisi del caso medio

- Si ha quindi la seguente relazione di ricorrenza per esprimere il numero $T(n)$ di confronti richiesti:

$$T(n) = n - 1 + \frac{2}{n} \sum_{i=n/2}^{n-1} T(i)$$

- **Teorema:** si mostra, tramite tecnica di sostituzione, che la soluzione all'equazione di ricorrenza di cui sopra è $T(n) \leq 4n$



Costo del partizionamento
usando la modifica di
partition() che raggruppa gli
elementi uguali al pivot

Dimostrazione

- Per sostituzione, dimostriamo che $T(n) \leq cn$ per una opportuna costante c

$$\begin{aligned} T(n) &= n - 1 + \frac{2}{n} \sum_{i=n/2}^{n-1} T(i) \\ &\leq n - 1 + \frac{2}{n} \sum_{i=n/2}^{n-1} ci \\ &= n - 1 + \frac{2c}{n} \left(\sum_{i=1}^{n-1} i - \sum_{i=1}^{n/2-1} i \right) \end{aligned}$$

Dimostrazione

$$T(n) \leq n - 1 + \frac{2c}{n} \left(\sum_{i=1}^{n-1} i - \sum_{i=1}^{n/2-1} i \right)$$

$$= n - 1 + \frac{2c}{n} \left(\frac{n^2}{2} - \frac{n^2}{8} - \frac{n}{4} \right)$$

$$= \left(1 + \frac{3c}{4} \right) n - 1 - \frac{c}{2}$$

$$\leq \left(1 + \frac{3c}{4} \right) n \leq cn$$

Ricordiamo che

$$\sum_{i=1}^n i = n(n+1)/2$$

- L'induzione funziona quando $(1+3c/4) \leq c$, ossia $c \geq 4$

Possiamo fare ancora meglio?

- Sì, anche se non facilmente
- Esiste un algoritmo deterministico per la selezione del k-esimo elemento che ha costo $O(n)$ nel caso peggiore
 - Trovate la descrizione dell'algoritmo (che usa una tecnica chiamata “mediano dei mediani”) nel libro di testo
 - Nota: il costo non dipende da k come nel caso di quickSelect