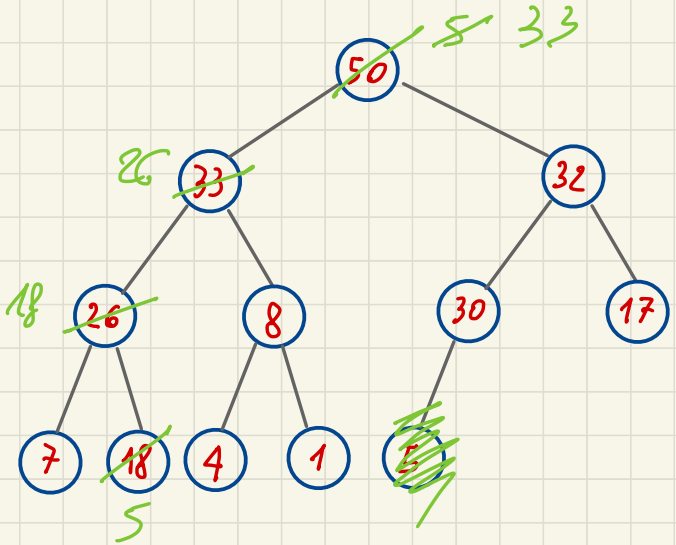
Algoritmi e Strutture Dati

Lezione 17

7 novembre 2022

Operazioni su heap

Operazioni su heap



- Trova elemento di chiave massima

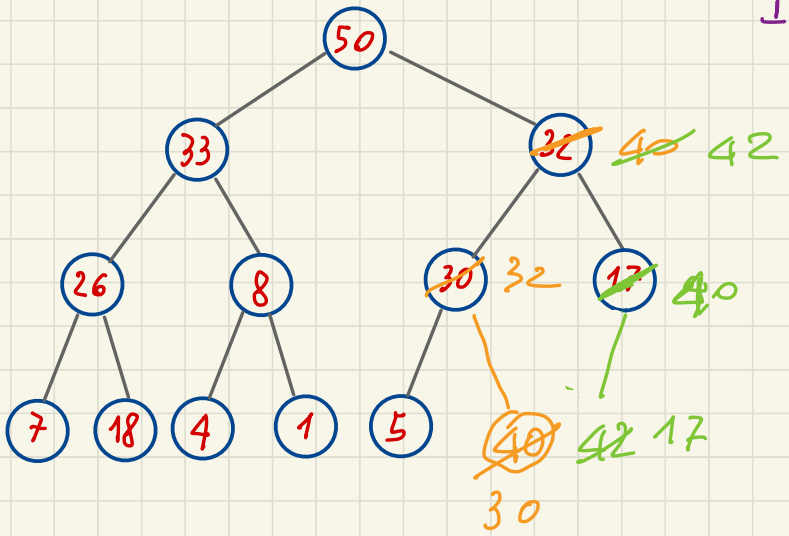
$\Theta(1)$

- Cancella elemento di chiave massima

$\Theta(\log n)$

Operazioni su heap

- Inserisci un nuovo elemento

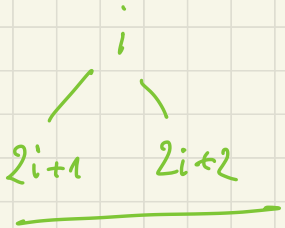


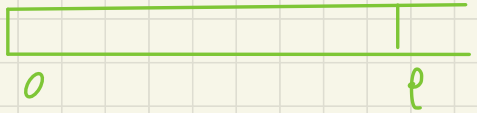
aggiungi 40

aggiungi 42

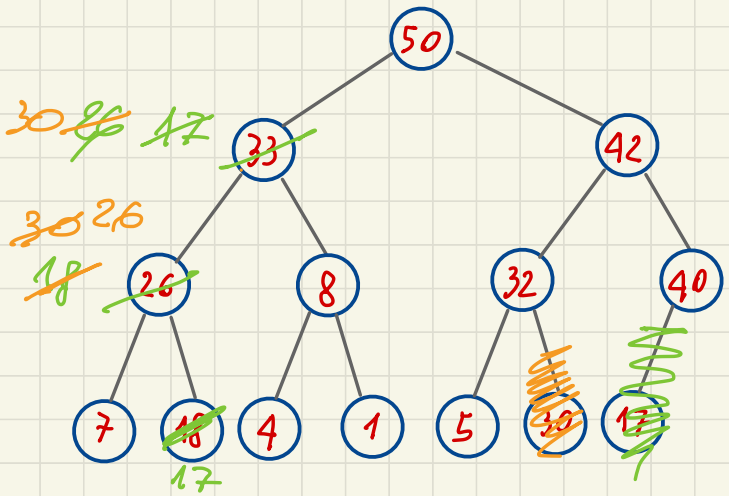
"risistemo dal basso"

$\Theta(\lg n)$ passi





Operazioni su heap



- Cancella un elemento di chiave x

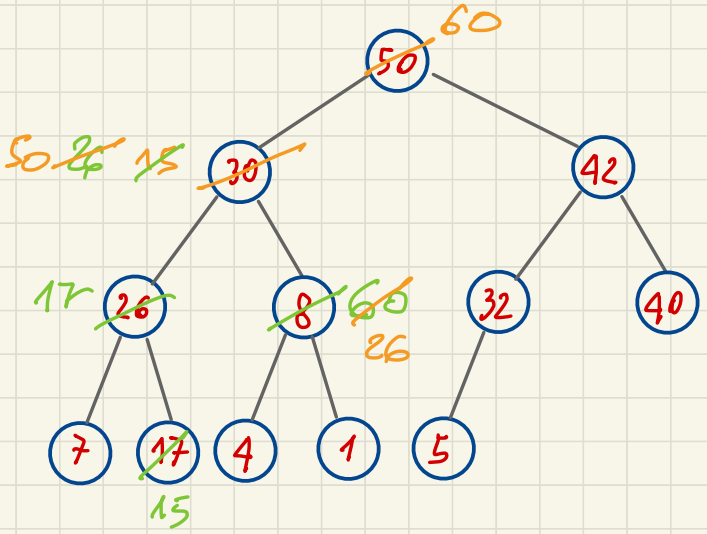
cancella 33

cancella 18

- sostituisci l'elemento da cancellare con l'ultima foglia, che viene rimossa sia f il valore che c'era nell'ultima foglia
  - Se f < x → applica risalita
  - Se f > x → applica risalita del barro

$\Theta(\lg n)$ passi (conoscendo la posizione iniziale di x)

Operazioni su heap



- Modifica la chiave di un elemento

$30 \rightarrow 15$

se chiave decresce → risistemo

$8 \rightarrow 60$

se chiave cresce → risistemo dal basso

$\Theta(\lg n)$ passi (se si conosce la posizione del nodo da modificare)

Code con priorità

Code con priorità

- Collezioni di dati da cui gli elementi vengono prelevati secondo un *criterio di priorità*

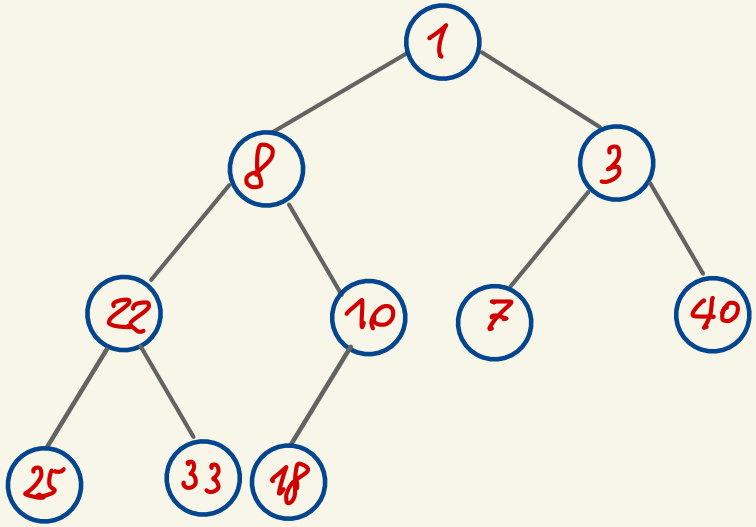
- Ogni elemento ha una *chiave*:
  Chiave più bassa indica priorità maggiore

Code con priorità: operazioni

- `findMin()`
  Restituisce l'elemento minimo della coda (senza rimuoverlo)
- `deleteMin()`
  Restituisce l'elemento minimo della coda e lo rimuove
- `insert(elemento e, chiave k)`
  Inserisce nella coda un elemento e con associata una chiave (priorità) k
- `delete(elemento e)`
  Cancella l'elemento e
- `changeKey(elemento e, chiave d)`
  Modifica la priorità dell'elemento e, assegnando come nuovo valore d

Code con priorità

Implementazione mediante *MinHeap*



- `findMin()` $O(1)$
- `deleteMin()` $\Theta(\log n)$
- `insert(elemento e, chiave k)` $\Theta(\log n)$
- `delete(elemento e)` $\Theta(\log n)$
- `changeKey(elemento e, chiave d)` $\Theta(\log n)$

→ essendo già posizionato nell'albero

Algoritmi di ordinamento basati su confronti

Ordinamento

Problema dell'ordinamento:

Input: n elementi $x_1, x_2, \ldots, x_n$
provenienti da un dominio D su cui è definita una relazione $\leq$ di *ordine totale*

Output: Sequenza $x_{j_1}, x_{j_2}, \ldots, x_{j_n}$
con $(j_1, j_2, \ldots, j_n)$ permutazione di $(1, 2 \ldots, n)$
tale che $x_{j_1} \leq x_{j_2} \leq \cdots \leq x_{j_n}$

$n \cdot (n-1) \cdot (n-2) \cdots 1 = n!$

permutazioni di n elementi

Ordinamento: metodi basati su confronti

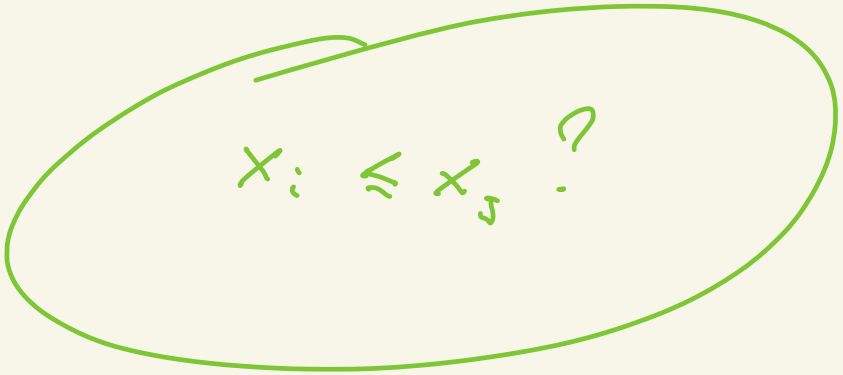
elem

avanz.

Algoritmo	*Numero confronti*	*Spazio*	*Note*	*Stabile*
`selectionSort`	$\Theta(n^2)$ sempre	$O(1)$	in loco	no
`insertionSort`	$\Theta(n^2)$ caso peggio. $n-1$ array ord	$O(1)$	in loco	si
`bubbleSort`	$\Theta(n^2)$ caso peggio. $n-1$ array ord	$O(1)$	in loco	si
`mergeSort`	$\Theta(n \lg n)$	$\Theta(n)$	$\Theta(n)$ array ausiliario per merge $\Theta(\lg n)$ altezza stack	si
`quickSort`	$\Theta(n^2)$ caso peggiore $\Theta(n \lg n)$ caso migliore $\approx 1.39 n \lg_2 n$ in media	$\Theta(n)$ $\Theta(\lg n)$	in loco stack ricorsione - altezza $\Theta(n)$ caso peggio. - altezza $\Theta(\lg n)$ vers. migliorata	no
`heapSort`	$\Theta(n \lg n)$	$O(1)$	in loco	no

Ordinamento: metodi basati su confronti

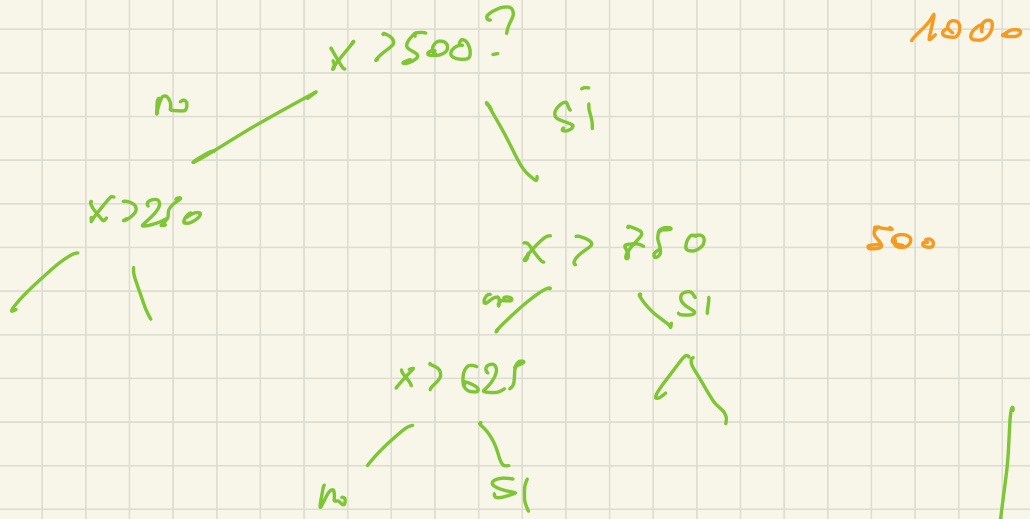
È possibile ordinare array di n elementi utilizzando un numero di confronti tra chiavi che cresca meno di $n \log n$?



ALBERO DI DECISIONE

nodi interni: domande sì/no

foglie: possibili risultati



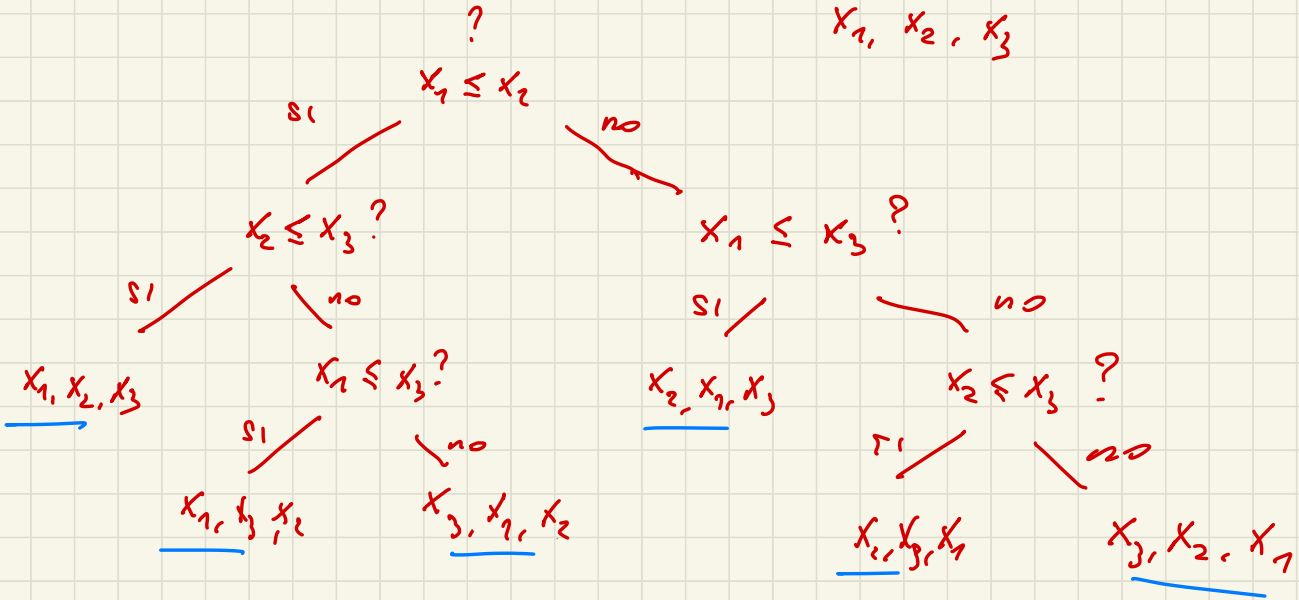
1000

500

$2 \log_2 1000 = 16$

1

1000



almeno una foglia per ogni permutazione

$\rightarrow n!$ foglie

$\Rightarrow$ altezza $\geq \log_2 n!$

Formula di Stirling $n! \simeq \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$

$$\log_2(n!) \simeq \log_2\left(\sqrt{2\pi n}\left(\frac{n}{e}\right)^n\right)$$

$$= \log_2\sqrt{2\pi} + \log_2\sqrt{n} + \log_2\left(\frac{n}{e}\right)^n$$

$$= \log_2\sqrt{2\pi} + \frac{1}{2}\log_2 n + n\log_2 n - n\log_2 e$$

$$= \Theta(n\log n)$$

$\Rightarrow$ Ogni algoritmo di ordinamento basato su cfr. nel caso peggiore effettua almeno $\Omega(n\log_2)$ confronti

Ordinare senza confrontare

Problema: ordinare n interi in $[0..b-1]$

A

1	0	4	2	0	2

Y

$b=5$

$\emptyset_2$ 2	$\emptyset_1$	$\emptyset_{1_2}$	0	$\emptyset_1$
0	1	2	3	4

integerSort

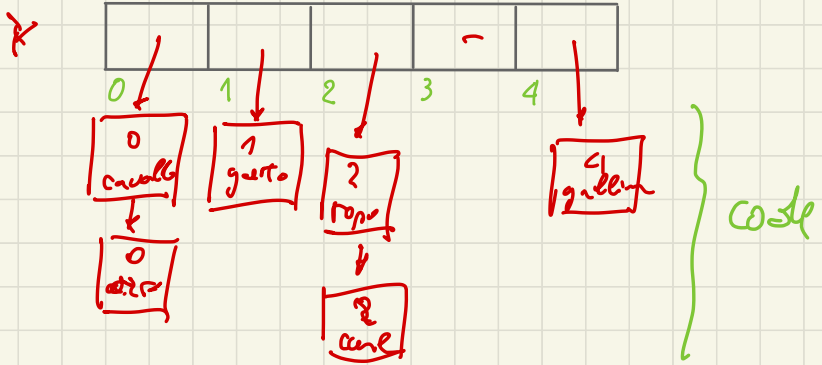
A

1 0	$\emptyset_0$	4 1	2_2	$\emptyset_2$	2_4

Problema: ordinare n record con chiavi intere in $[0..b-1]$

A

1	0	4	2	0	2
gatto	cavallo	gallina	topo	orso	cane



bucketSort

A

1 0	0 0	4 1	2 2	0 2	2 4
gatto cavallo	cavallo orso	gallina gatto	topo cane	orso cane	cane gallina