Titolo nota 26/02/2016

HEAPSORT

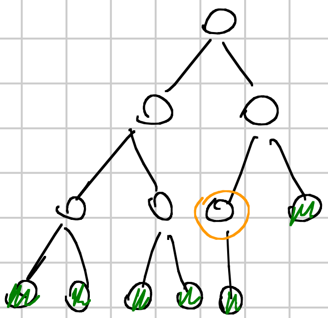
	CASO OTTIMO	CASO MEDIO	CASO PESSIMO	SUL POSTO
Selection sort	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	SI
Insertion sort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$	SI
Merge sort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	NO
Quick sort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n^2)$	SI MA RICORSIONE
Heapsort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	SI

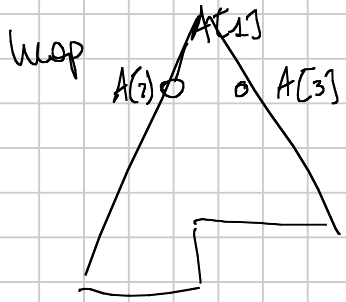
- Un algoritmo ordina sul posto <=> ad ogni istante al più un ~~ct~~ costante di elementi è memorizzato fuori dall'array

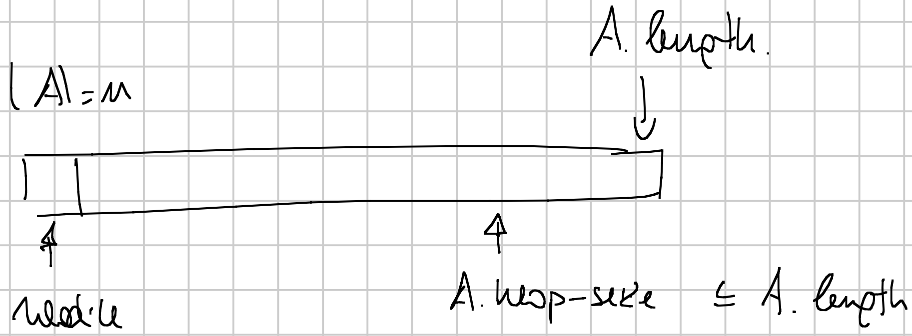
Heap: struttura dati

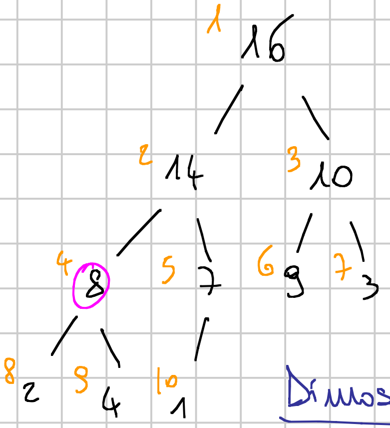
HEAP

Uno heap binario è un albero binario quasi completamente bilanciato









$|A| = 10$

1	2	3	4	5	6	7	8	9	10
16	14	10	8	7	9	3	2	4	1

padre(4) $i=4$ left(i) right(i)

REGOLE DI POSIZIONAMENTO

i è l'indice in A di un nodo:

- $PARENT(i) = \lfloor \frac{i}{2} \rfloor$
- $LEFT(i) = 2i$
- $RIGHT(i) = 2i+1$

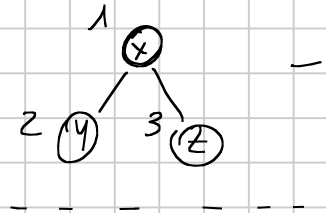
Dimostrazione di correttezza

per induzione su i:

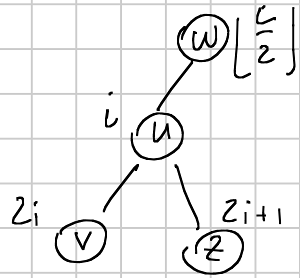
- CASO BASE $i=1$ $left(1)=2$ $right(1)=3$

$i=2$ $PARENT(2) = 1$

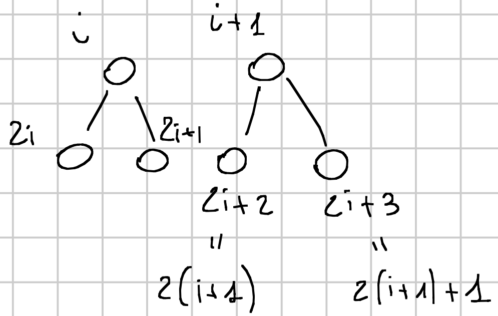
$i=3$ $PARENT(3) = \lfloor \frac{3}{2} \rfloor = 1$



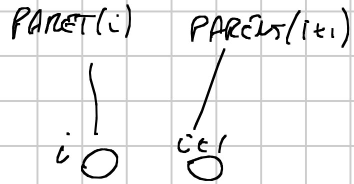
passo induttivo $\quad i \to i+1$

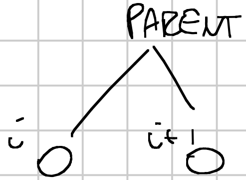


i+1 ?



left e right sono ok





caso 1

i pari → i+1 dispari

$$PARENT(i+1) = PARENT(i) = \left\lfloor \frac{i}{2} \right\rfloor = \frac{i}{2} = \left\lfloor \frac{i+1}{2} \right\rfloor$$

caso 2

i dispari → i+1 pari

$$PARENT(i+1) = PARENT(i) + 1 = \left\lfloor \frac{i}{2} \right\rfloor + 1 =$$

$$= \frac{i-1}{2} + 1 = \frac{i+1}{2} = \left\lfloor \frac{i+1}{2} \right\rfloor$$

CVD

$$PARENT(LEFT(i)) = PARENT(2i) = \left\lfloor \frac{2i}{2} \right\rfloor = i$$

$$PARENT(RIGHT(i)) = PARENT(2i+1) = \left\lfloor \frac{2i+1}{2} \right\rfloor = \frac{2i+1}{2} - \frac{1}{2} = \frac{2i}{2} = i$$

HEAP

Heap sort

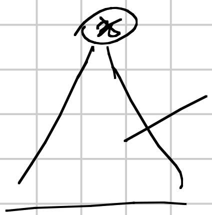
MAX-HEAP

MIN-HEAP

code di priorità

il valore di un nodo è al massimo il valore di suo padre

$$\boxed{A[PARENT(i)] \geq A[i]}$$



$\leq x$

- La profondità di un nodo è la sua distanza dalla radice
- L'altezza di un nodo è il # archi nel cammino più lungo fino ad una foglia
- L'altezza di uno heap è l'altezza della sua radice

foglia → altezza 0

PROPRIETÀ 1

Uno heap di n elementi ha altezza $O(\log n)$, e più precisamente $\lfloor \log n \rfloor$

Dimostrazione

Sia h l'altezza dello heap

$$2^h \le n \le 2^{h+1}-1 < 2^{h+1}$$

① $1+\sum_{k=0}^{h-1} 2^k = 1+2^h-1 = 2^h$

② $\sum_{k=0}^{h} 2^k = 2^{h+1}-1$

$$\Rightarrow 2^h \le n \le 2^{h+1}$$

$$\sum_{i=0}^{n} 2^i = 2^{n+1}-1$$

$$\sum_{i=0}^{n} x^i = \frac{x^{n+1}-1}{x-1}$$

$2^k \le n \le 2^{k+1}$

$k \le \log n < k+1$

$\Downarrow$

$\log n - 1 < k \le \log n$

$k = \lfloor \log n \rfloor$

def di $\lfloor \; \rfloor$

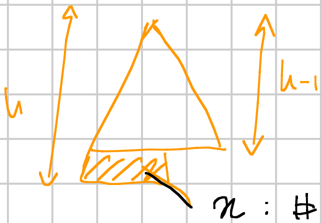
$x - 1 \le \lfloor x \rfloor \le x$

WD

PROPRIETÀ 2

Uno heap di n nodi contiene $\left\lceil \frac{n}{2} \right\rceil$ foglie

Dimostrazione

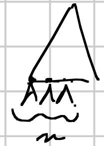


x : # foglie all'ultimo livello

$$n = 2^h - 1 + x \quad (*)$$

CASO 1

x è pari



$h-1$ ho 2^{h-1} nodi, di cui $\frac{x}{2}$ sono padri delle x foglie a livello h

$$\# \text{ foglie } \quad 2^{h-1} - \frac{x}{2} + x =$$

$$= 2^{h-1} + \frac{x}{2} = \frac{2^h + x}{2} = \left\lceil \frac{2^h + x}{2} \right\rceil = \left\lceil \frac{2^h + x - 1}{2} \right\rceil \stackrel{(*)}{=} \left\lceil \frac{n}{2} \right\rceil$$

CASO 2

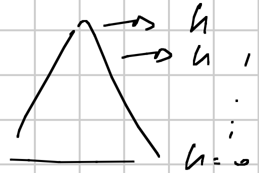
x è dispari

$$2^{h-1} - \frac{x+1}{2} + x = 2^{h-1} + \frac{2x - x - 1}{2} = 2^{h-1} + \frac{x-1}{2} = \frac{2^h + x - 1}{2} \stackrel{(*)}{=} \frac{n}{2} = \left\lceil \frac{n}{2} \right\rceil$$

PROPRIETÀ 3

In uno heap di n nodi, ci sono al massimo $\left\lceil \frac{n}{2^{h+1}} \right\rceil$ nodi di altezza h

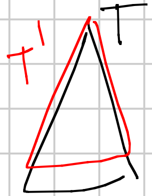
(ed esattamente $= \left\lceil \frac{n}{2^{h+1}} \right\rceil$ se lo heap è ABCB)



Dimostrazione

per induzione su h

CASO BASE $h=0$ $\qquad \left\lceil \frac{n}{2} \right\rceil = \left\lceil \frac{n}{2^{h+1}} \right\rceil$



PASSO INDUTTIVO $\quad h-1 \rightsquigarrow h$

n_h : # nodi ad altezza h in un albero di n nodi

T' : albero ottenuto da T rimuovendo tutte le foglie

T' ha n' nodi

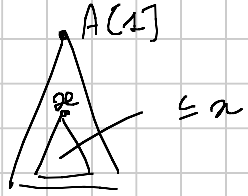
$n' = n - n_0 =$ $\qquad n_0$ # foglie di T

$= n - \left\lceil \frac{n}{2} \right\rceil = \left\lfloor \frac{n}{2} \right\rfloor$

n'_{h-1} # nodi di altezza $h-1$ in T'

$$n_h = n_{h-1} \underset{\text{hp induttiva}}{\le} \left\lceil \frac{n'}{2^h} \right\rceil = \left\lceil \frac{\lfloor \frac{n}{2} \rfloor}{2^h} \right\rceil \le \left\lceil \frac{\frac{n}{2}}{2^h} \right\rceil = \left\lceil \frac{n}{2^{h+1}} \right\rceil$$

MAX-HEAP

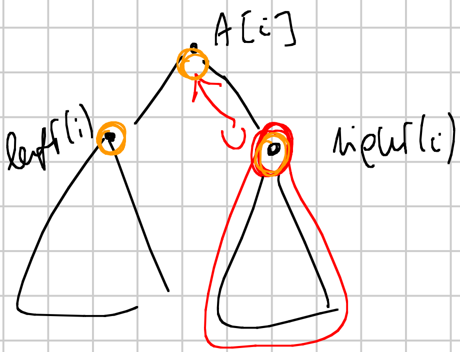


Titolo nota 26/02/2016

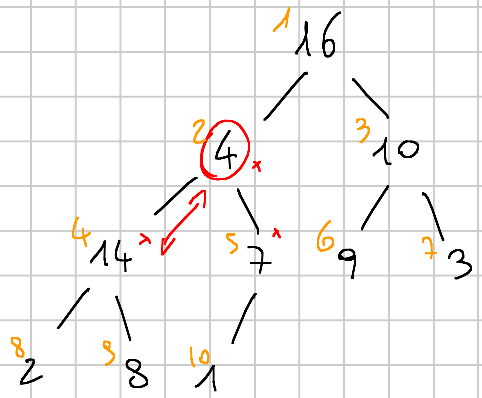
MAX-HEAPIFY (A, i)

Max-heapify, quando viene chiamata, assume che i sottoalberi left(i) e right(i) siano MAX-HEAP, e A[i] violi la proprietà di MAX-HEAP.

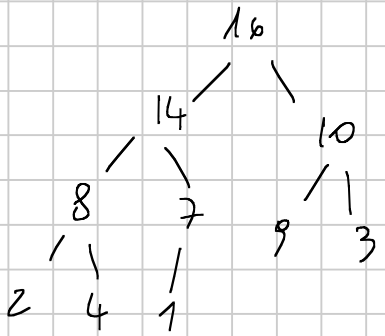
MAX-HEAP ripristina la proprietà su tutto lo heap di radice A[i]



MAX



MAX HEAPIFY $(A, 2)$



Un sottoalbero di dimensione n ho figli che hanno al massimo $\frac{2}{3}n$ nodi ciascuno

Titolo nota 26/02/2016

MAX-HEAPIFY (A, i)

< confronto A[i], A[left(i)], A[right(i)] e metto il max in massimo] $\Theta(1)$

if massimo $\neq$ i then { scambio $A[i] \leftrightarrow A[\text{massimo}]$ $\Theta(1)$

$\Theta(1)$

MAX-HEAPIFY (A, massimo) $T\left(\frac{2}{3}n\right)$

}

$T(n) = T\left(\frac{2n}{3}\right) + \Theta(1)$

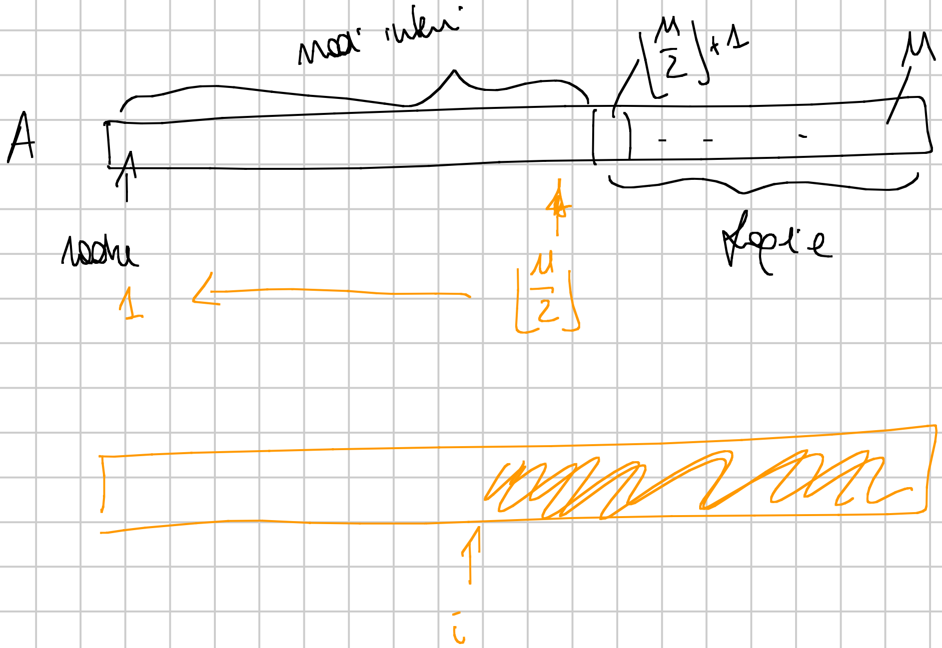
$f(n) = \Theta(1)$

$n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$

caso ②

$\Theta(\log n)$

BUILD-MAX-HEAP



$\lceil \frac{n}{2} \rceil$ foglie

$\lfloor \frac{n}{2} \rfloor$ nodi interni o radice