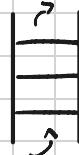


HEAP

↪ per implementare una struttura dati assezzata: CODA CON PRIORITÀ (LIFO)



- A parità di valore si considera il tempo di arrivo
- Si considera la Chiave (valore più piccolo).

0	1	2	3	4	5
5	1	2	3	9	4

le Chiavi inserite sono valori di priorità.

↪ sempre un valore
no posizione.

Per essere efficiente, le operazioni della struttura dati devono avere un costo basso.

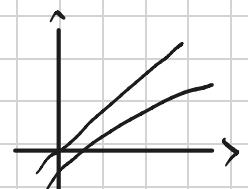
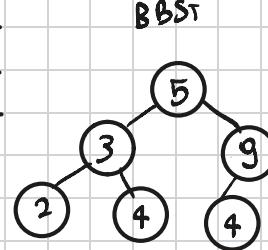
ESEMPIO

↪ Consideriamo un array disordinato

	INS	EXTR	Decr	MIN	OPERAZIONI
ARRAY	$\Theta(1)$	$\Theta(n)$	$\Theta(1)$	$\Theta(n)$	
ARRAY ORD.	$O(n)$	$\Theta(n)$	$O(n)$	$\Theta(1)$	
BBST	$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(\log n)$	

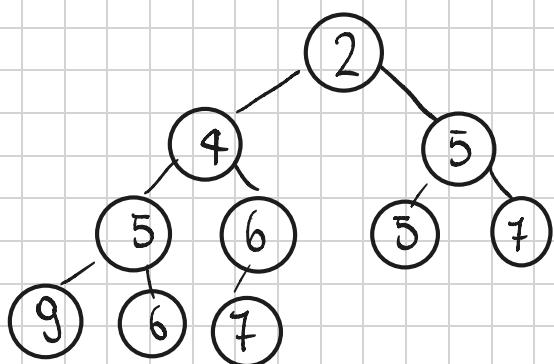
	0	1	2	3	4	5	
ARRAY	5	1	2	3	9	4	
ARRAY ORD.	0	1	2	3	4	5	
	2	3	4	5	7	9	

ARRAY ORDINATO



- **HEAP** → Struttura dati **NON LINEARE**. E' un **albero binario** dove ogni nodo ha al più due figli. E' **posizionale**, ha un figlio sinistro e uno destro, a sinistra più piccolo della chiave a destra più grande. E' **completo**, tutti i livelli sono pieni, l'ultimo livello può essere non pieno ma solo se i nodi sono allineati da sinistra verso destra

MIN HEAP → chiavi piccole priorità più alta



x x left x right

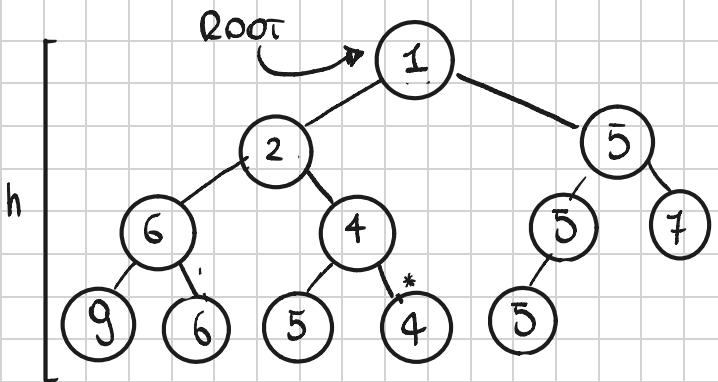
Key [x] ≤ Key [x left]

Key [x] ≤ Key [x right]

Se ho un heap e delle chiavi ho diversi modi per inserirle, ci sono tante combinazioni

MAX HEAP → priorità di valori più grandi

- La chiave del nodo ha priorità rispetto ai figli.



* Se devo aggiungere un nodo
lo devo aggiungere qui

DECREASE-KEY (x, k)

```

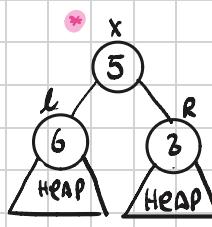
Key[x] <- k
P <- x.PARENT
WHILE (P ≠ NULL AND key[P] > key[x])
    SWAP(x, P)
    P <- x.PARENT
    x <- P
  
```

HEAPIFY (x) → Realizza un heap

```

l <- x.LEFT
R <- x.RIGHT
IF (l ≠ NULL AND key[l] < key[x])
    then MIN <- l
IF (R ≠ NULL AND key[R] < key[MIN])
    then MIN <- R
IF MIN ≠ x THEN
    SWAP(key[x], key[MIN])
    HEAPIFY(MIN)
  
```

	<u>INS</u>	<u>EXTR</u>	<u>Deca</u>	<u>min</u>
ARRAY	$\Theta(1)$	$\Theta(n)$	$\Theta(1)$	$\Theta(n)$
ARRAY and	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	$\Theta(1)$
B BST	$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(\log n)$
HEAP	$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(1)$



Equazione di Ricorrenza*

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

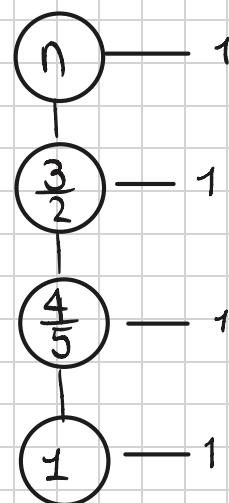
METODO MASTER

$$\begin{aligned}
 a &= 1 \\
 b &= \frac{3}{2} \\
 f(n) &= 1
 \end{aligned}
 \quad n^{\log_b a} = n^{\log_{\frac{3}{2}} 1} = 1$$

2° CASO

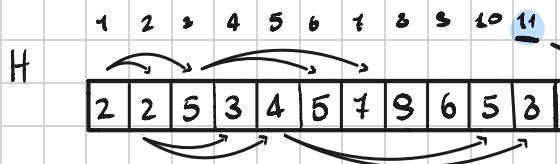
$$f(n) \in \Theta(n^{\log_b a}) \Rightarrow T(n) \in \Theta(\log n)$$

Albero di Ricorsione:



$$T(n) = \sum_{i=0}^{\log_{\frac{3}{2}} n} 1 = \log_{\frac{3}{2}} n \in O(\log n)$$

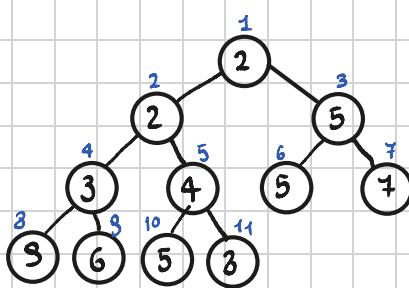
Heapsize (numero di nodi contenuti nell'heap)



HEAPIFY (H, i) *

```

 $l \leftarrow \text{left}(i), R \leftarrow \text{right}(i); \text{min} \leftarrow i$ 
IF  $l \leq \text{heapsize}$  AND  $H[l] < H[i]$ 
Then  $\text{min} \leftarrow l$ 
IF  $R \leq \text{heapsize}$  AND  $H[R] < H[\text{min}]$ 
Then  $\text{min} \leftarrow R$ 
IF  $(\text{min} \neq i)$  THEN
SWAP ( $H, i, \text{min}$ )
HEAPIFY ( $H, \text{min}$ )
    
```



Shift per moltiplicazione
 \uparrow
 $\text{Left}(i) = 2i = (i \ll 1)$
 $\text{Right}(i) = 2i + 1 = (i \ll 1) + 1$
 $\text{Parent}(i) = [i/2] = (i \gg 1)$

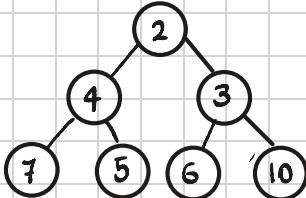
* questa versione utilizza indici e non punzetti

- Viene rappresentato come un array

COSTRUIRE UN HEAP

1) Partire da heap vuoto e inserire gli elementi

3	7	2	4	5	6	7
---	---	---	---	---	---	---

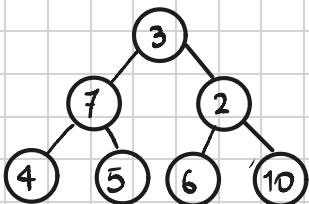


$O(n \log n)$ → a partire da n elementi

2) Prendo gli elementi e li metto in un array

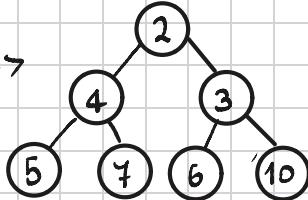
13	7	2	4	5	6	10
----	---	---	---	---	---	----

→ COSTRUZIONE dell'array $O(n)$



→ NON È ANCORA UN HEAP
(non è ordinato)

- Posso chiamare la funzione HEAPIFY sui nodi 7 e 2, perché posso usarla solo su heap



→ COSTRUISCE UN HEAP A PARTIRE DA UN ARRAY

BUILD-MIN-HEAP (H, n)

FOR $i \leftarrow [\frac{n}{2}]$ DOWN TO 1 DO

 HEAPIFY (H, i)

$\hookrightarrow \log n$

$O(n \log n)$

2	4	3	5	7	6	10
---	---	---	---	---	---	----

Un albero binario ha al più $\frac{n}{2^{h+1}}$ nodi di altezza h

$$\frac{n}{2^{h+2}} = \frac{n}{2^h} = \frac{n}{2} \rightarrow \text{ALTEZZA } O$$

$$\frac{n}{2} = \frac{n}{2^{\log_2 n}} = \text{MANCA QUALcosa}$$

$$T(n) = \sum_{h=0}^{\log n} \frac{n}{2^{h+2}} \quad O(n) = n \sum_{h=0}^{\log n} O\left(\frac{n}{2^h}\right) \leq n \sum_{h=0}^{\infty} \frac{n}{2^h} = n \sum_{h=0}^{\infty} K \left(\frac{1}{2}\right)^h = n \left(\frac{1}{1-\frac{1}{2}}\right)^2 = \Theta(n)$$

$$\sum_{k=0}^{\infty} kx^k = \frac{1}{(1-x)^2}$$
$$|x| < 1$$