DATI & STRUTTURE DATI

TIPO DI DATO → SPECIFICA UNA COLLEZIONE DI OGGETTI E DELLE OPERAZIONI SU QUESTA COLLEZIONE.
ES. DIZIONARIO CON INSERIMENTO, CANCELLAZIONE ...

STRUTTURA DATI → ORGANIZZAZIONE DEI DATI CHE PERMETTE DI MEMORIZZARE ED EFFETTUARE LE OPERAZIONI USANDO MENO RISORSE POSSIBILI

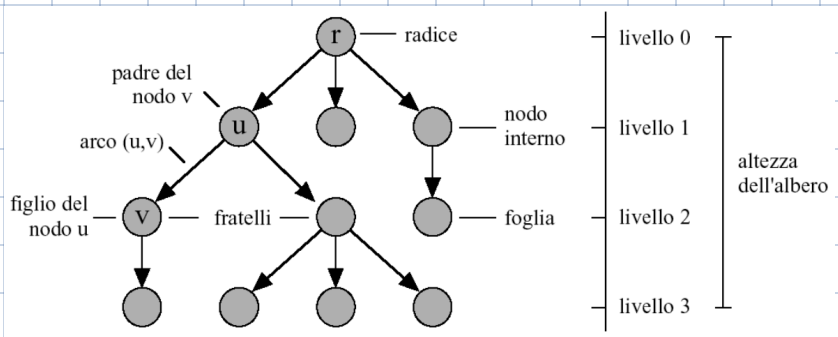
↓

CRUCIALE → IMPLEMENTARE UNA STRUTTURA DATI H SU CUI:
- DATO UN ARRAY A GENERARE VELOCEMENTE H
- TROVARE IL MAX IN H
- CANCELLARE IL MAX DA H

↓

TIPO DI DATO: CODA CON PRIORITÀ

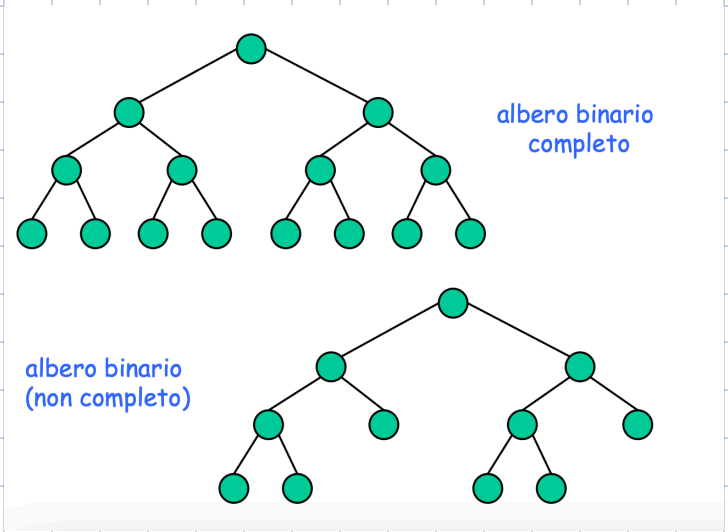
DEFINIZIONI DI ALBERI



ALBERO D-ARIO → ALBERO IN CUI TUTTI I NODI INTERNI HANNO AL PIÙ D FIGLI.

SE D = 2 → ALBERO BINARIO

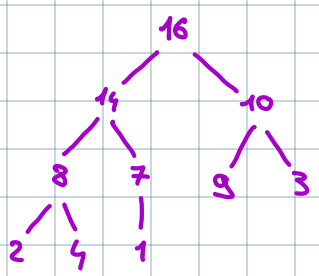
ALBERO COMPLETO → TUTTI I NODI INTERNI HANNO ESATTAMENTE D FIGLI E LE FOGLIE SONO SULLO STESSO LIVELLO.



ON

STRUTTURA DATI HEAP

STRUTTURA ⟶ ALBERO BINARIO RADICATO CON LE SEGUENTI PROPRIETÀ:

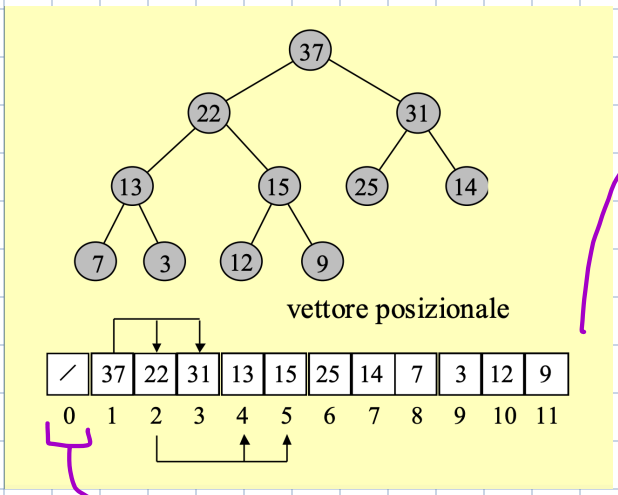


- ALBERO BINARIO COMPLETO, OGNI NODO FINO AL PENULTIMO LIVELLO DEVE AVERE 2 FIGLI, L'ULTIMO LIVELLO PUÒ AVERE LE FOGLIE COMPATTATE A SINISTRA.
- ELEMENTI MEMORIZZATI NEI NODI, OGNI NODO V CONTIENE UN SOLO ELEMENTO DENOTATO CON CHIAVE (v).
- CHIAVE (PADRE(v)) ≥ CHIAVE (v) ∀ NODO v DIVERSO DALLA RADICE.

ALTRE PROPRIETÀ:

- IL MASSIMO È CONTENUTO NELLA RADICE
- L'ALBERO CON m NODI HA ALTEZZA $O(\log m)$
- HEAP CON STRUTTURA RAFFORZATA POSSONO ESSERE RAPPRESENTATI CON UN ARRAY DI DIMENSIONE m.

RAPPRESENTAZIONE TRAMITE VETTORE



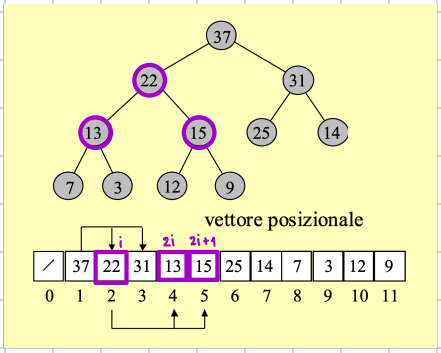
I NUMERI NELL'ARRAY SONO INSERITI LIVELLO x LIVELLO DA SX A DX

POSSIAMO CALCOLARE LE VARIE POSIZIONI COSÌ:

$$\text{sin}(i) = 2i$$
$$\text{des}(i) = 2i+1$$
$$\text{padre}(i) = \lfloor i/2 \rfloor$$

* DI OGGETTI INDICATO CON: HEAPSIZE[A]
DI SOLITO IN POSIZIONE 0, QUINDI IN POSIZ. 0 AVREMMO IL NUMERO DI OGGETTI DELL'ARRAY

ESEMPIO



$i = 2$ QUINDI ARRAY $[i] = 22$

$SIN(i) = 2i = 4$ QUINDI ARRAY $[4] = 13$

ELEMENTO A SX DI 22 NELL' ALBERO

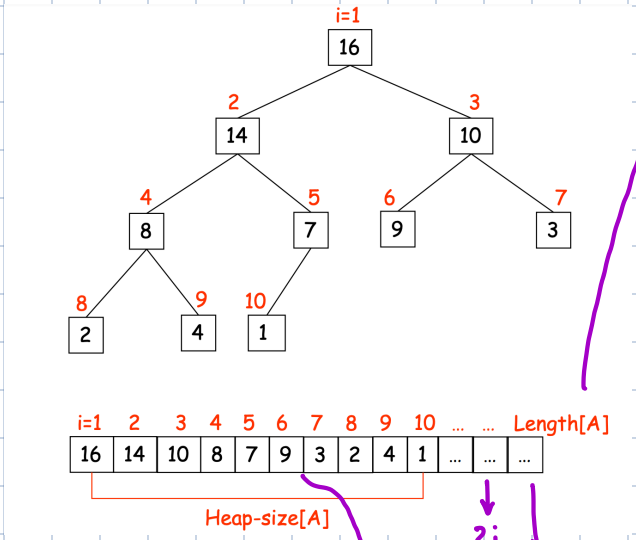
$DIS(i) = 2i + 1 = 5$ QUINDI ARRAY $[5] = 15$

ELEMENTO A DX DI 22 NELL' ALBERO

$PADRE(i) = \lfloor i/2 \rfloor$ QUINDI ARRAY $[1] = 37$

PADRE DELL' ELEMENTO 22 NELL' ALBERO

HEAPSIZE & LENGTH



$i = 6$ È UNA FOGLIA

$2i$

$2i + 1$

CONTENGONO NULL

IL VETTORE NON CONTIENE SOLAMENTE GLI ELEMENTI DELL' ALBERO MA CONTIENE ANCHE DEGLI SLOT FINALI, PER OGNI FOGLIA SI HANNO 2 SLOT FINALI CONTENENTI NULL

PERCHÉ?

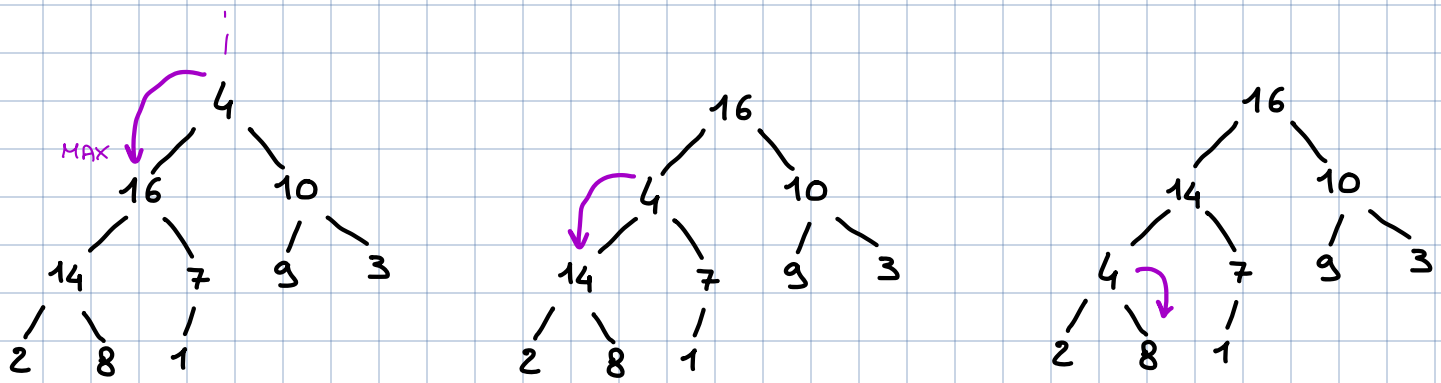
PERCHÉ COME FACCIAMO A CAPIRE NELL' ARRAY QUALI SONO LE FOGLIE?
CALCOLIAMO SINISTRA & DESTRA CON:

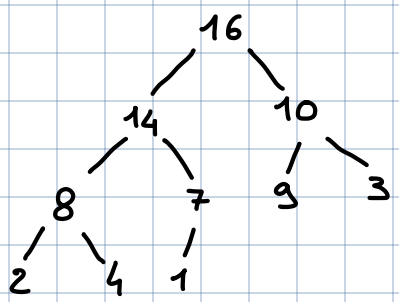
- $SIN(i) = 2i$
- $DES(i) = 2i + 1$

E VEDREMO COSÍ CHE GLI SLOT DI SINISTRA & DESTRA CALCOLATI AVRANNO DENTRO NULL E CAPIREMO COSÍ CHE IL NODO i È UNA FOGLIA.

FIX HEAP

VIENE USATO PER FIXARE UN ALBERO IN CUI È PRESENTE UN' ANOMALIA, CIOÈ UN PUNTO IN CUI ABBIAMO UN NODO PADRE CHE È MINORE DEI FIGLI.





COMPLESSITÀ = $O(\log n)$

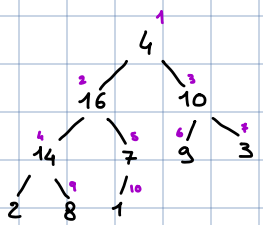
4 (1)	16 (2)	10 (3)	14 (4)	7 (5)	9 (6)	3 (7)	2 (8)	8 (9)	1 (10)	NULL (11)	NULL (12)	NULL (13)	NULL (14)	NULL (15)	NULL (16)

HEAPSIZE = 10

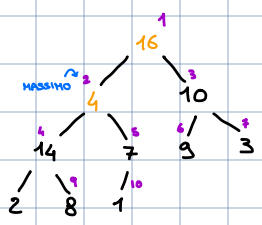
fixHeap (i,A)

1. s=sin(i) ⟶ S = POSIZIONE DEL NODO SX NELL' ARRAY (2i)
2. d=des(i) ⟶ D = POSIZIONE DEL NODO DX NELL' ARRAY (2i+1)
3. **if** (s ≤ heapsize[A] e A[s] >A[i]) ⟶ * CONTROLLO CHE S SIA DENTRO L'HEAPSIZE, CON ▲ CONTROLLO SE IL NODO A SX È > DEL PADRE
4. **then** massimo=s ⟶ SE SI, IL MASSIMO DIVENTA IL NODO SINISTRO
5. **else** massimo=i ⟶ SE NO, IL MASSIMO RIMANE IL PADRE
6. **if** (d ≤ heapsize[A] e A[d] >A[massimo]) ⟶ STESSA COSA DI PRIMA MA ORA CONTROLLO SE IL NODO DX > DEL MASSIMO (CHE IN BASE A PRIMA È IL MAX TRA (NODO SX e PADRE))
7. **then** massimo=d ⟶ SE SI, IL MASSIMO DIVENTA IL NODO DX
8. **if** (massimo≠i) ⟶ CONTROLLO CHE IL MASSIMO NON SIA L'INDEX DEL PADRE, ALTRIMENTI ANDREBBE BENE COSÌ
9. **then** scambia A[i] e A[massimo] ⟶ SE IL MASSIMO NON È IL PADRE SCAMBIO IL MASSIMO CON IL PADRE FACENDO COSÌ DIVENTARE IL MASSIMO IL PADRE E IL VECCHIO PADRE IL FIGLIO
10. fixHeap(massimo,A)

↳ CHIAMO NUOVAMENTE FIXHEAP PASSANDO COME "NODO DA ANALIZZARE" IL MASSIMO CIOÈ LA POSIZIONE CHE CONTIENE IL VECCHIO PADRE CHE ADESSO È IL PADRE DEGLI ELEMENTI SOTTO DI ESSO

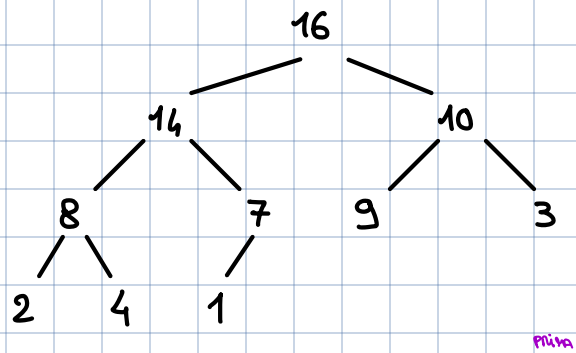


i = 1
S = 2
D = 3
MASSIMO = 2



FixHeap(MASSIMO (2), A)

ESTRAZIONE DEL MASSIMO

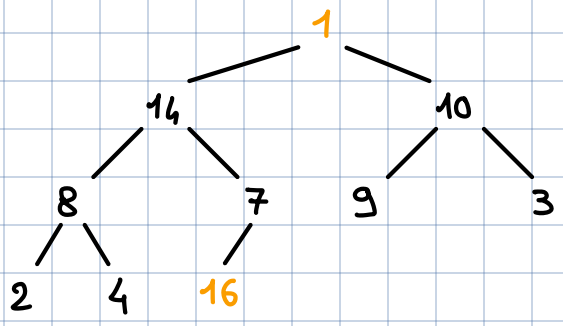


RAPPRESENTATO CON:

16	14	10	8	7	9	3	2	4	1	NULL	NULL	NULL	NULL	NULL	NULL

HEAPSIZE

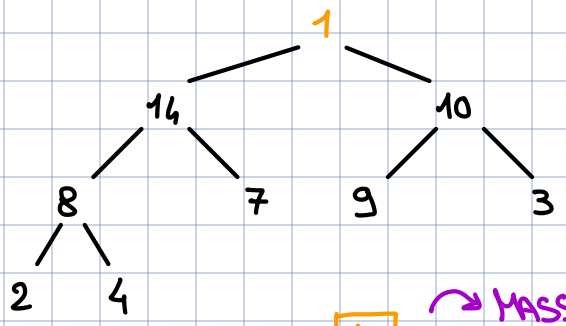
IL MASSIMO SI TROVA ALLA RADICE (PRIMA POS. NELL'ARRAY) E DOBBIAMO TIRARLO FUORI
INNANZITUTTO SPOSTIAMO SCAMBIAMO LA RADICE CON LA FOGLIA PIÙ A DESTRA: (ULTIMA POSIZ OVVERO L'HEAPSIZE)



1	14	10	8	7	9	3	2	4	16	NULL	NULL	NULL	NULL	NULL	NULL

HEAPSIZE

ADESSO RIMUOVO/STACCO L'ELEMENTO IN POSIZIONE HEAPSIZE OVVERO IL MASSIMO

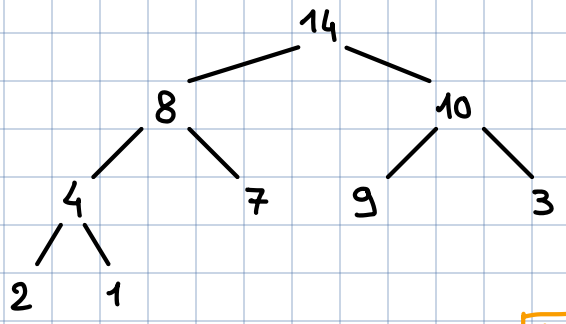


1	14	10	8	7	9	3	2	4	NULL	NULL	NULL	NULL

HEAPSIZE

$\boxed{16}$ MASSIMO ESTRATTO

ADESSO HO UN VALORE MOLTO BASSO ALLA RADICE, MI BASTA CHIAMARE IL FIXHEAP PER SISTEMARLO:



14	8	10	4	7	9	3	2	1	NULL	NULL	NULL	NULL

HEAPSIZE

$\boxed{16}$ MASSIMO ESTRATTO

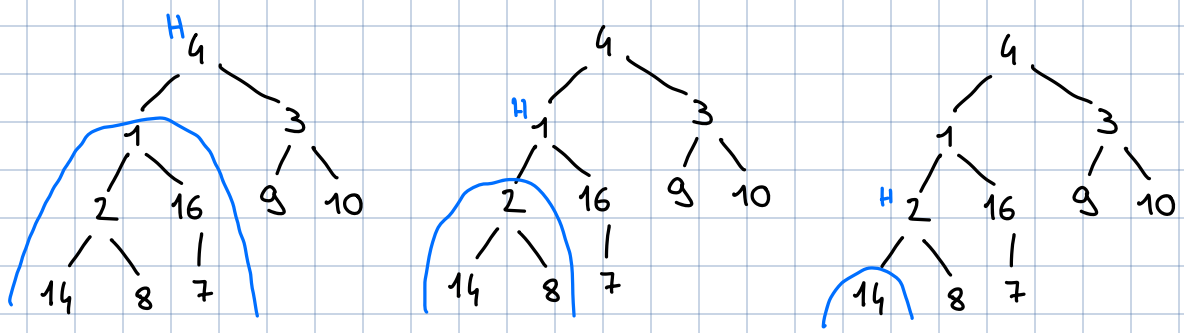
COSTO: $O(\log m)$

HEAPIFY

UTILIZZATO PER TRASFORMARE UN INTERO ALBERO/ARRAY NON ORDINATO IN UN HEAP

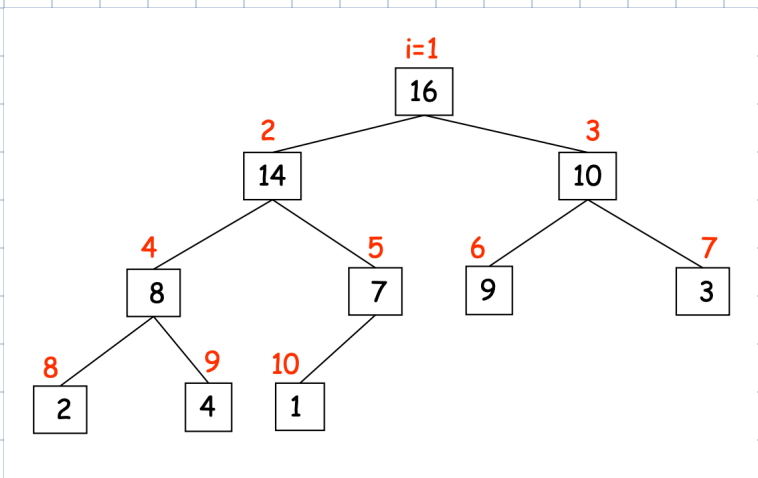
```
heapify(heap H)
    if (H non è vuoto) then
        heapify(sottoalbero sinistro di H)
        heapify(sottoalbero destro di H)
        fixHeap(radice di H,H)
```

!! SEGUI PASSO PASSO CON LE DITA UN ILLUSTRAZIONE COSÍ DA CAPIRE MEGLIO



E VIA DICENDO

AVREMO POI UN RISULTATO FINALE COSÍ:



→ É UN HEAP

ANALISI DELLA COMPLESSITÀ DI HEAPIFY

SIA h L'ALTEZZA DI UN HEAP CON m ELEMENTI.
DIAMO ADESSO UN UPPERBOUND USANDO UN ALBERO COMPLETO COME UPPERBOUND INVECE DI UN ALBERO CON LE FOGLIE A SX.
AVREMO QUINDI:

- m' ELEMENTI DELL'ALBERO COMPLETO E QUINDI $T(m) \leq T(m')$ E $m' \leq 2m$

  PIÙ GRANDI m MA NON ABBASTANZA PER $2m$

$$T(m') = 2T\left(\frac{m'-1}{2}\right) + O(\log m')$$

IGNORO IL (-1) $\rightarrow$ $T(m') = 2T\left(\frac{m'}{2}\right) + O(\log m')$

$m-1$ PERCHÉ ALLA RADICE NON VIENE FATTA NESSUNA CHIAMATA

$\frac{}{2}$ PERCHÉ PRENDE LA METÀ DI SX E LA METÀ DI DX

COSTO DEL FIXHEAP

APPLICANDO IL TEOREMA MASTER

$a = 2$
$b = 2$
$f(m) = O(\log m')$

$m^{\log_b a} = m^{\log_2 2} = m^1$ E ABBIAMO m vs $\log m$

$\log m = O(m)$ E SIAMO NEL CASO 1 DEL T.M. QUINDI $T(m') = \Theta(m')$

LO RISCRIVO COME O-GRANDE

$$T(m') = O(m')$$

HEAP SORT ALGORITMO

SENTI LA REGISTRAZIONE AUDIO !!!

COSTRUISCE UN HEAP TRAMITE HEAPIFY

heapSort (A)

1. Heapify(A) $O(m)$ → COSTRUISCE L'HEAP
2. Heapsize[A]=n → ULTIMO ELEMENTO A_0 =<16,14,10,8,7,9,3,2,4,1> ↓ HEAPSIZE
3. **for** i=n **down to** 2 **do**
4. scambia A[1] e A[i] → SCAMBIA L'ULTIMO & IL PRIMO ELEMENTO
5. N-1 VOLTE Heapsize[A] = Heapsize[A] -1 → SPOSTA HEAPSIZE DI UNA POSIZIONE INDIETRO STACCANDO LA FOGLIA PIÙ A DX DOVE C'È L'ELEMENTO PIÙ GRANDE
6. $O(\log m)$ fixHeap(1,A) → FIXA PORTANDO IL MASSIMO IN CIMA

COMPLESSITÀ = $O(m \cdot \log m)$

HEAP

→ ALBERO BINARIO CON ALTEZZA $\log m$, COMPLETO ALMENO FINO AL PENULTIMO LIVELLO. OGNI NODO PADRE HA CHIAVE CON VALORE > DI QUELLA DEI FIGLI

FIXHEAP

→ AGGIUSTA UN'ANOMALIA NELL'ALBERO, TROVA IL MASSIMO TRA IL FIGLIO SX & IL FIGLIO DX E LO SCAMBIA CON IL PADRE, CHIAMA RICORSIVAMENTE FIXHEAP SUL VECCHIO PADRE CHE ORA È UN FIGLIO XCHÈ È STATO SCAMBIATO. $O(\log m)$

ESTRAZIONE MASSIMO

→ X LE PROPRIETÀ DELL'ALBERO IL MASSIMO SI TROVA ALLA RADICE, PER STACCARLO LO SCAMBIO CON LA FOGLIA PIÙ A DESTRA DELL'ULTIMO LIVELLO OVVERO IN POSIZIONE HEAPSIZE, LA STACCO E CHIAMO FIXHEAP SULLA RADICE. $O(\log m)$

HEAPIFY

→ SCENDE IN TUTTI I SOTTOALBERI E MAN MANO CHE RISALE CHIAMA FIXHEAP SUI SOTTOALBERI. $O(m)$

HEAPSORT

→ TRASFORMA L'ALBERO IN UN HEAP, SCAMBIA L'ULTIMA FOGLIA CON LA RADICE, ADESSO NELL'ULTIMA FOGLIA ABBIAMO IL NUMERO PIÙ GRANDE, STACCHIAMO LA FOGLIA (DECREMENTANDO HEAPSIZE)

E CHIAMIAMO FIXHEAP SULLA RADICE, ORDINIAMO QUINDI DA DX VERSO SX IN MODO DECRESCENTE.

$O(m \log m)$