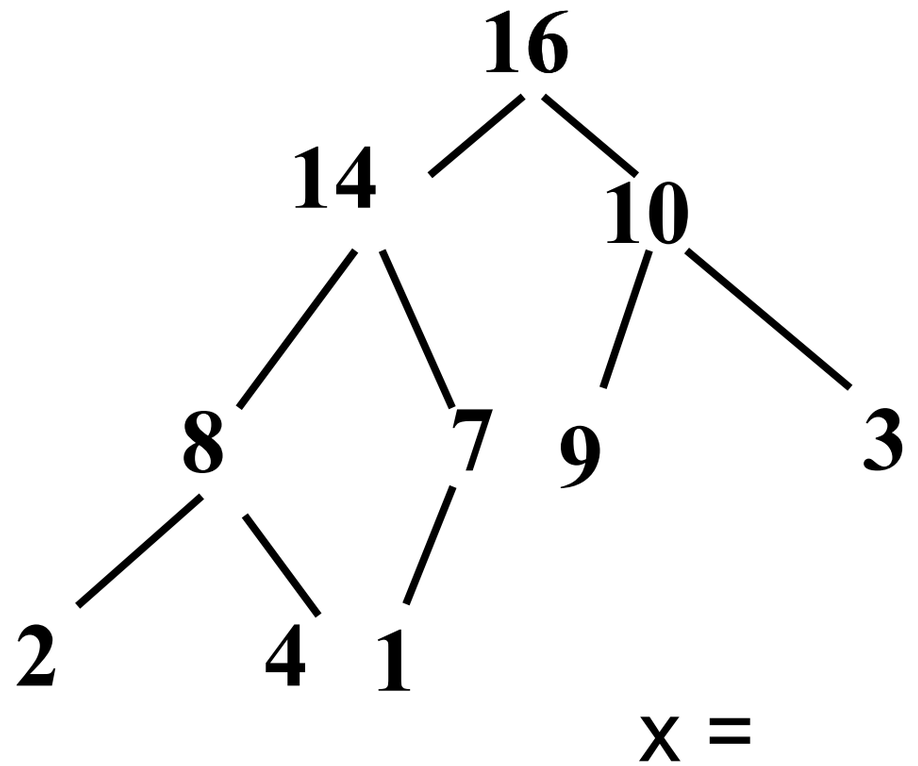
Determinare il massimo in un maxHeap è immediato

A=

16	14	10	8	7	9	3	2	4	1	4	0	7
1	2	3	4	5	6	7	8	9	10	11	12	13

```
Heap-Maximum (A)
  return A[1]
```



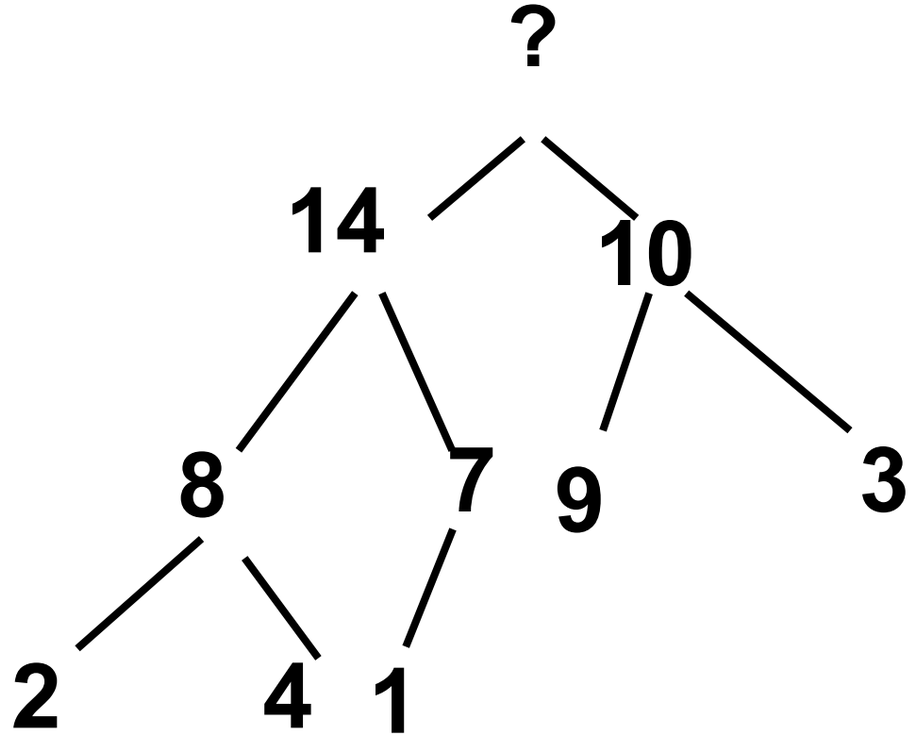
Θ(1)

Estrazione del massimo

A=

16	14	10	8	7	9	3	2	4	1	4	0	7
1	2	3	4	5	6	7	8	9	10	11	12	13

x =



Dopo l'operazione bisogna avere un MaxHeap con un elemento in meno.

Estrazione Massimo

Proprietà MaxHeap violata alla radice

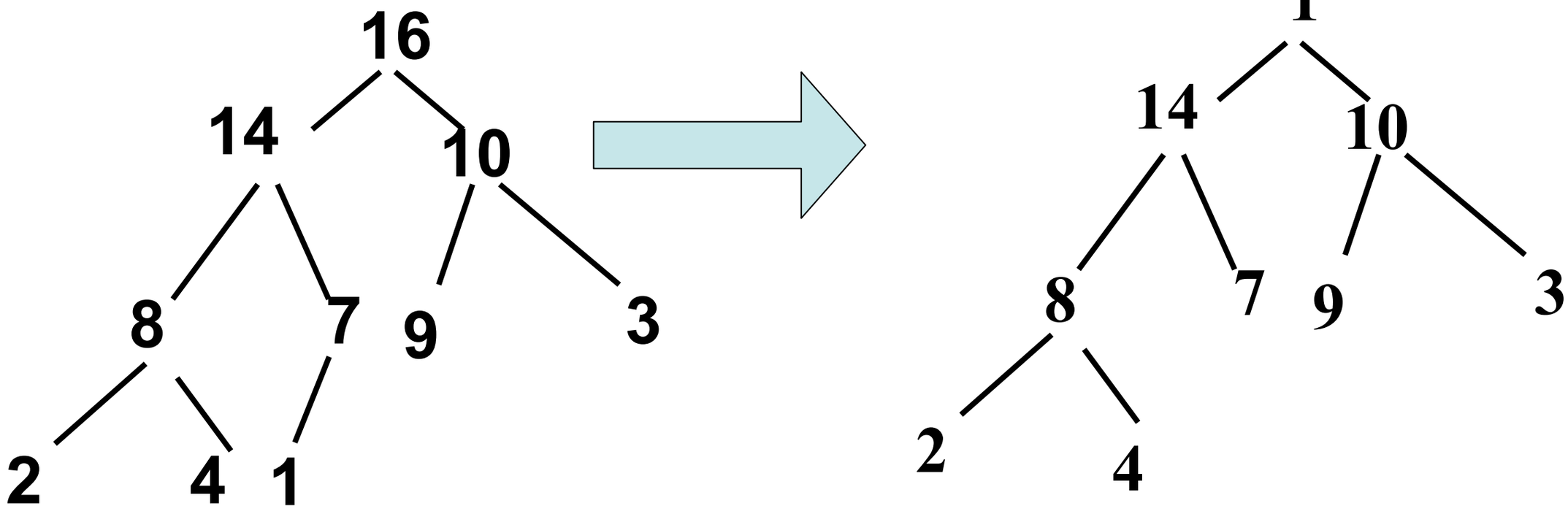
A=

16	14	10	8	7	9	3	2	4	1	4	0	7
1	2	3	4	5	6	7	8	9	10	11	12	13

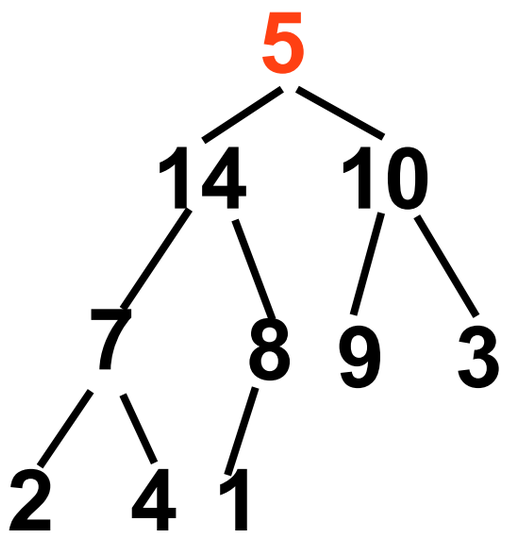
A=

1	14	10	8	7	9	3	2	4	1	4	0	7
1	2	3	4	5	6	7	8	9	10	11	12	13

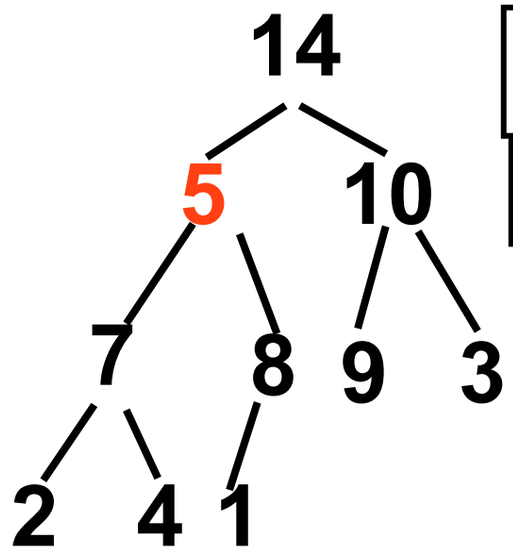
max =



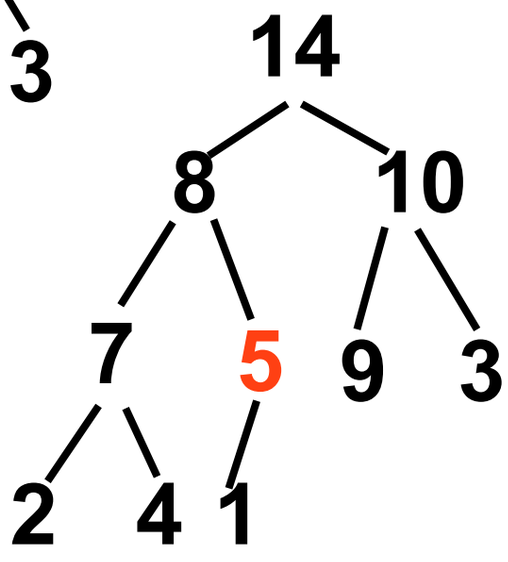
ripristino proprietà del Max-heap



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



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



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

Max-Heapify

Max-Heapify (A,i)

Input: A è un array e i è un indice

Prec: i figli o il figlio di A[i] sono radici di un max-heap, mentre A[i] può essere più piccolo di uno o di entrambi i suoi figli
Output: A[i] è radice di un max-Heap

se A[i] ha due figli prendiamo il figlio con valore massimo
se ha un figlio prendiamo quell'unico figlio
confrontiamo il valore di A[i] con quello del figlio selezionato
se il valore del figlio è maggiore di quello del padre, li scambiamo
e proseguiamo nello stesso modo sul figlio fino a quando la proprietà è ristabilita.

Max-Heapify (A,i)

Input: A è un array e i è un indice

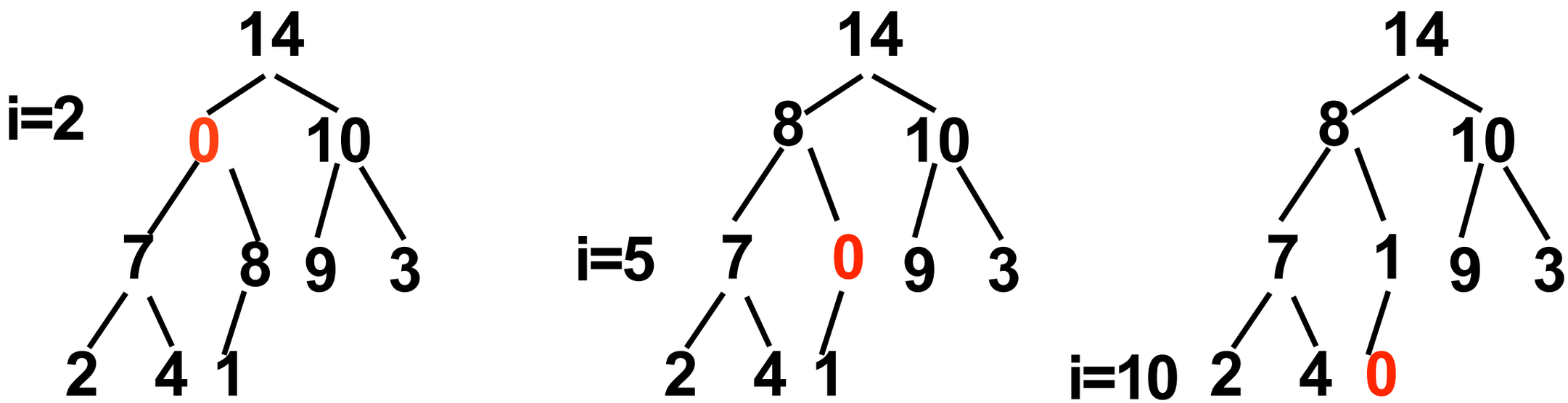
Prec: A[left(i)] e A[right(i)] sono radici di max-heap mentre A[i] può essere più piccolo dei suoi figli

Postc: A[i] è radice di un max-Heap

```
max = i
repeat
i = max
if (2i ≤ A.heap.size) and A[2i] > A[i] then max = 2i
if (2i +1 ≤ A.heap-size) and A[2i+1] > A[max] then max = 2i+1
if (i ≠ max) then scambia A[i] e A[max]
until max = i (CONDIZIONE DI USCITA DAL CICLO)
```

Ad ogni esecuzione del ciclo si "scende" di padre in figlio, quindi al più si va dalla radice alla foglia più lontana.
Caso peggiore: T(n) = Θ(log n)

Max-Heapify: esempio di esecuzione



Rimozione del massimo(A)

Heap-Extract-Max (A)

Input: A è un array

Prec: A è un max-heap

Postc: dà in output il massimo, che viene rimosso da A rispristinando la proprietà del max-heap

```
if A.heap.size < 1 then error "l'heap è vuoto"
max = A[1]
A[1]  = A[A.heap.size]
A.heap.size = A.heap.size - 1
Max-Heapify (A,1)
return max
```

O(lg n),
n = heap-size(A)

Max-Heap-insert(A,x)

A=

16	14	10	8	7	9	3	2	4	1	4	0	7
1	2	3	4	5	6	7	8	9	10	11	12	13

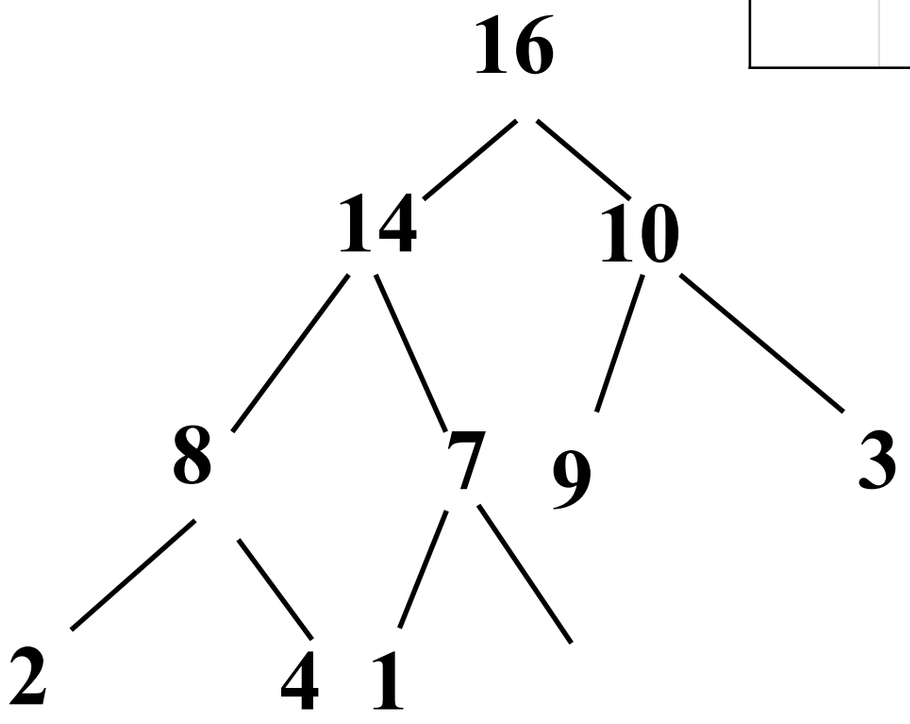
heap-size(A)=10

Insert(A,15)

A=

16	15	10	8	14	9	3	2	4	1	7	0	7
1	2	3	4	5	6	7	8	9	10	11	12	13

heap-size(A)=11



15

Max-Heap-insert

Max-Heap-insert(A,key)

input: A è un max-heap e key una chiave dello stesso tipo di quelle in A

output: Inserisce key in A, ripristinando la proprietà del max-heap

```
if A.heap.size = A.length then error "l'heap è pieno"
A.heap.size = A.heap.size +1
i = A.heap-size
A[i] = key
    while (i > 1 and A[i/2] < A[i])
          scambia  A[i/2] con  A[i]
          i = i/2
```

Caso peggiore: Θ(lg n),

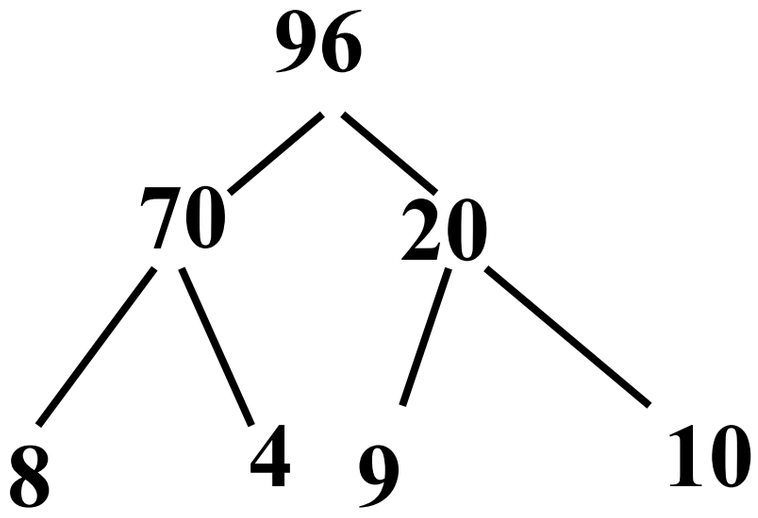
n = heap-size(A)

Esercizio estrazione massimo

Dato un MaxHeap di n > 2 elementi diversi, qual'è il minimo numero di scambi padre-figlio necessari nella rimozione del massimo, indipendentemente dal valore di n?
Si dia un esempio di MaxHeap con n = 7 elementi in cui si realizza tale minimo.

Esercizio Heap

Soluzione: basterà un solo scambio padre figlio se, per esempio, la foglia più a destra è nel sotto albero destro, la radice del sotto albero sinistro ha un valore maggiore di tutti gli elementi nel sotto albero destro mentre quelli nel sotto albero sinistro diversi dalla sua radice sono minori dell'elemento nell'ultima foglia. Esempio, estrazione del massimo:



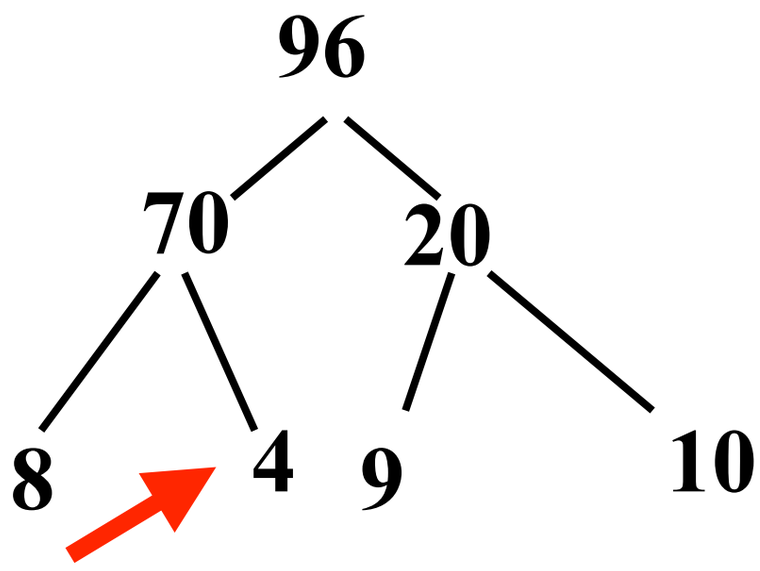
Esercizio il minimo

Dato un max-heap dove può trovarsi il minimo tra le chiavi memorizzate?
Si scriva un algoritmo iterativo per calcolare il minimo in un max-heap.

Esercizio il minimo

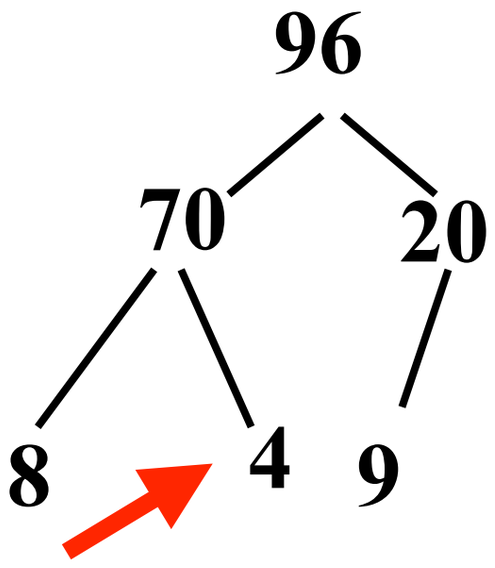
Dato un max-heap dove può trovarsi il minimo tra le chiavi memorizzate?
Si scriva un algoritmo iterativo per calcolare il minimo in un max-heap.

Soluzione: il minimo può trovarsi solo tra le foglie. Infatti i figli di ogni elemento sono più piccoli di lui.



Pseudocodice per il minimo

Soluzione: il minimo può trovarsi solo tra le foglie. Infatti i figli di ogni elemento sono più piccoli di lui.




```
MinimoInMax-Heap (A)
input: un max-heap A
output: il minimo in A
n= A.heapsize
minInd = n/2+1
for i = n/2+2 to n do
if A[minInd] > A[i] then minInd = i
return minInd
```