In questa lezione

- **Code di priorità**

[CLRS10] cap. 6, par. 5

Coda di priorità: cos'è?

- **Una coda di priorità è una struttura dati che permette di gestire dei dati con chiave numerica, la loro priorità.**

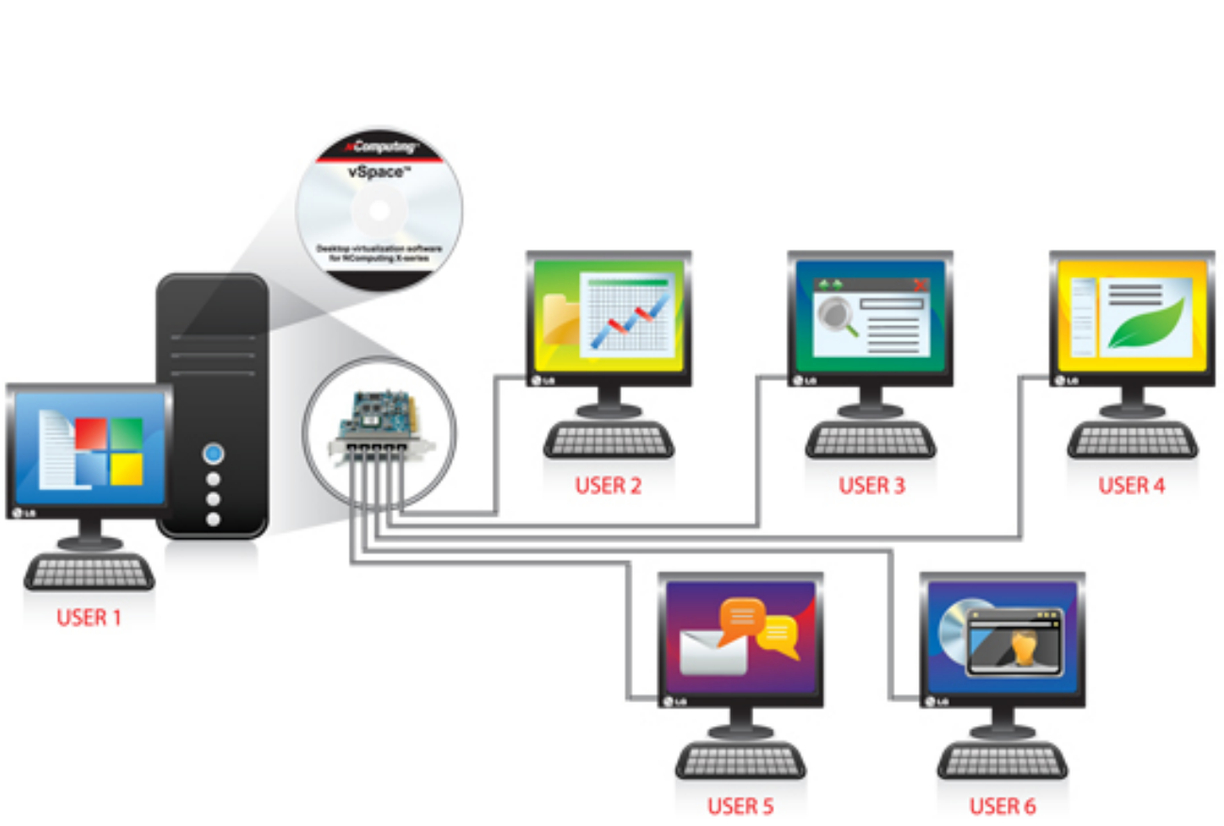
- **In una coda di priorità la modalità di prelievo si basa sulla priorità associata agli elementi, mentre in una coda la modalità di prelievo è basata sull'ordine di inserimento: il primo inserito è il primo a essere estratto**

Coda di max-priorità: le operazioni

- **Una coda di max-priorità offre le operazioni di**
  - **Insert(S,x): inserisce x in S**
  - **Maximum(S): restituisce l'elemento con priorità massima, senza estrarlo**
  - **Extract-Max(S): restituisce l'elemento con priorità massima eliminandolo dall'insieme**
  - **Increase-key(S,x,k): cambia la priorità di S[x] al valore k, se k è maggiore o uguale di quella presente**

Applicazioni

In caso di condivisione di un computer fra più utenti: viene svolto per primo il lavoro con più alta priorità



Coda di min-priorità: le operazioni

- **Una coda di min-priorità offre le operazioni di**

  - **Insert(S,x): inserisce x in S**

  - **Minimum(S): restituisce l'elemento con priorità minima**

  - **Extract-Min(S): restituisce l'elemento con priorità minima e lo rimuove dall'insieme**

  - **Decrease-key(S,x,k): porta a k la priorità di S[x], se k è minore di quella presente**

Applicazioni

Simulazione guidata da eventi: Gli elementi della coda sono quelli da simulare e ad ogni elemento è associato il tempo nel quale si può verificare.

Implementazione: caso max-priorità

- **Per implementare una coda con priorità si utilizza un array, in cui sono memorizzate le priorità**
- **Si utilizza un handle, aggancio (un puntatore, un intero), per legare la priorità nell'array a un elemento e viceversa (in tal caso l'handle dell'elemento è l'indice nell'array in cui compare la sua priorità).**

Implementazione: caso max-priorità

Usando un semplice array per rappresentare le priorità, si ottengono le seguenti prestazioni, nel caso peggiore:

implementazione	insert	Max	Extract-Max	Increase-key
array non ordinato	Θ(1)	Θ(n)	Θ(n)	Θ(n)
array ordinato crescente	Θ(n)	Θ(1)	Θ(1)	Θ(n)
heap binario	Θ(lg n)	Θ(1)	Θ(lg n)	?

Implementazione Increase-Key

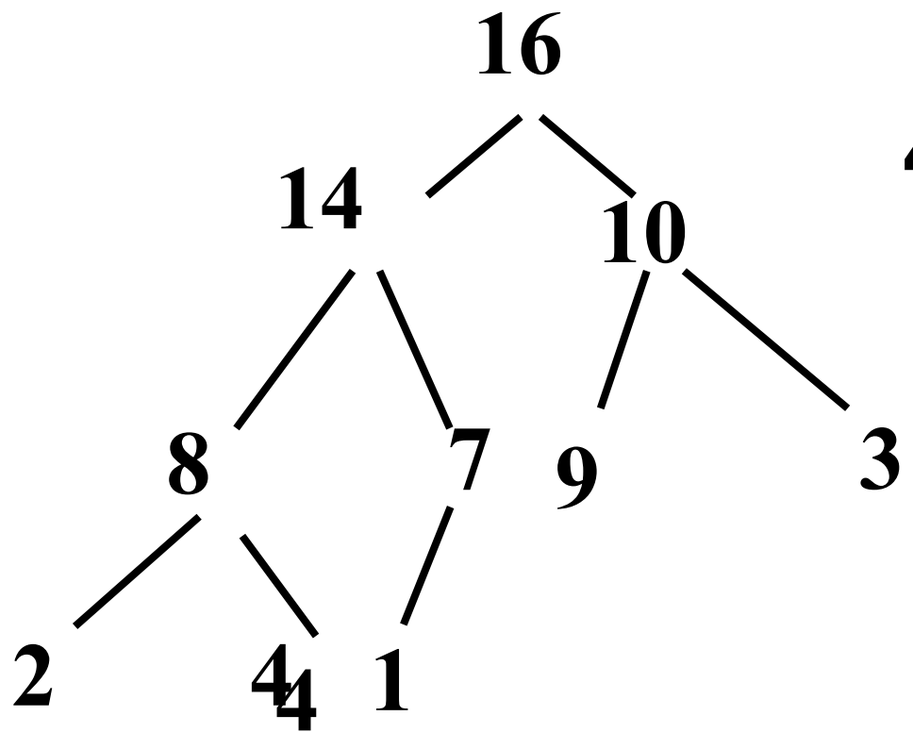
A=

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

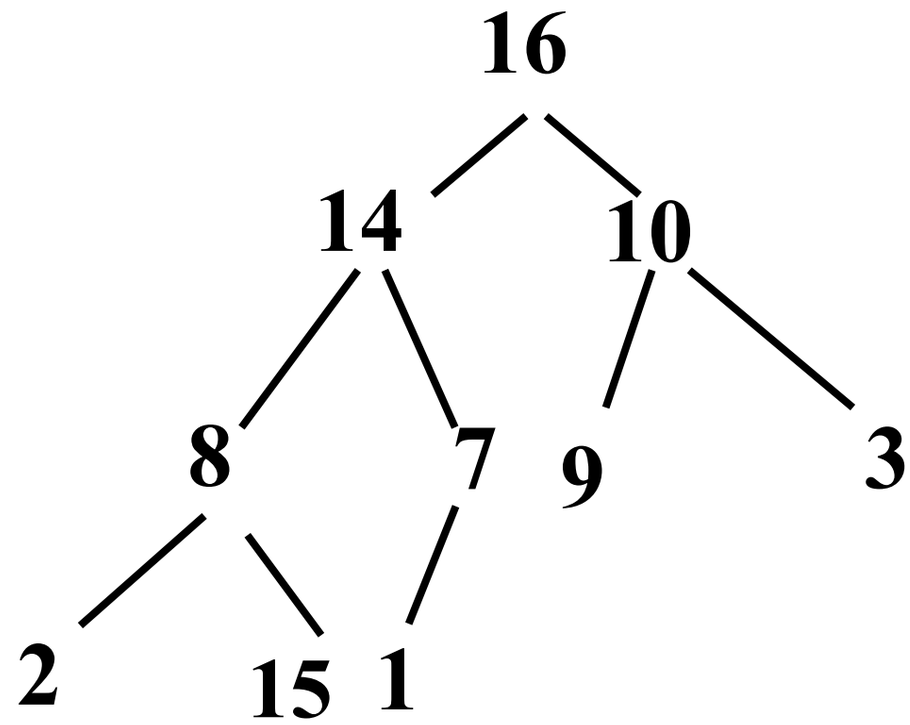
Increase-key(A,9,15)

A=

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



4 ⇨ 15



Implementazione Increase-Key

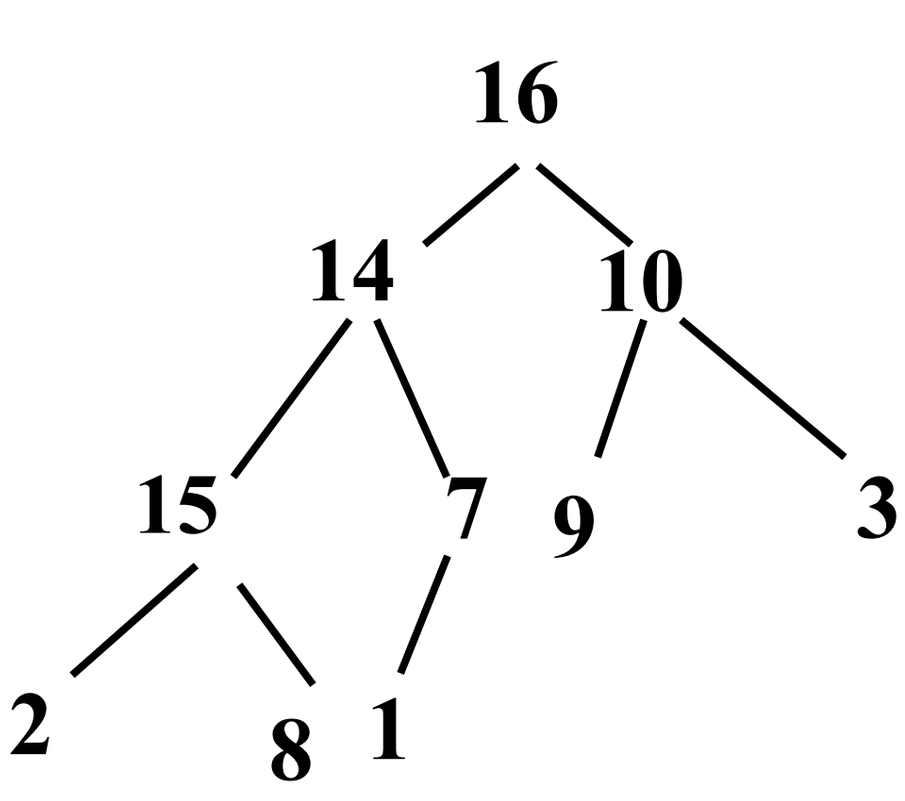
A=

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

Increase-key(A,9,15)

A=

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



Implementazione Increase-Key

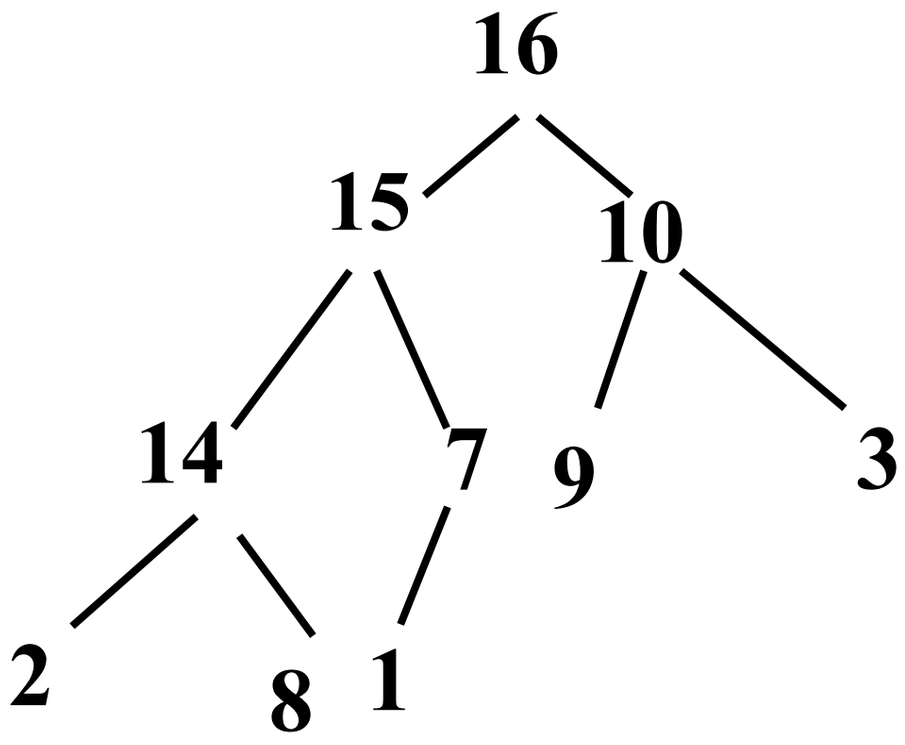
A=

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

Increase-key(A,9,15)

A=

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



Increase-key

Heap-increase-key(A,i,key)

prec: A è un max-heap e 1≤i≤A.heap-size

postc: sostituisce key ad A[i], se key ≥ A[i] e ripristina la proprietà del max-heap

if key < A[i] then "errore: la nuova chiave è più piccola"

A[i] = key

while (i > 1 and A[i/2] < A[i])

L'array A[1...heap-size] soddisfa la proprietà del max-heap tranne al più nella posizione i

scambia A[i/2] con A[i]

i = i/2 "si risale al padre di A[i]"

Increase-key: complessità

Heap-increase-key(A,i,key)

▷ A è un max-heap. Sostituisce key ad A[i], se key ≥ A[i] e ripristina la proprietà del max-heap

```
if key < A[i] then "errore: la nuova chiave è più piccola"
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)

Sommario dei costi

Complessità nel caso peggiore di diverse implementazioni di una coda di max-priorità con n elementi

implementazione	insert	Max	Extract-Max	Increase-key
array non ordinato	$\Theta(1)$	$\Theta(n)$	$\Theta(n)$	$\Theta(1)$
array ordinato crescente	$\Theta(n)$	$\Theta(1)$	$\Theta(1)$	$\Theta(n)$
heap binario	$\Theta(\lg n)$	$\Theta(1)$	$\Theta(\lg n)$	$\Theta(\lg n)$

Riassumendo

La MAX-HEAPIFY, che ha complessità logaritmica nel numero degli elementi, è il metodo chiave per mantenere la proprietà del max-heap, quando la proprietà è violata in un'entrata i dell'array i cui figli soddisfano la proprietà di essere max-heap.

I metodi
MAX-HEAP-INSERT,
HEAP-EXTRACT-MAX,
HEAP-INCREASE-KEY, e
HEAP-MAXIMUM,
tutti di complessità al più logaritmica nel numero degli elementi, realizzano un'implementazione di una coda di priorità con il max-heap molto efficiente.

Cancellare un elemento

Può essere inoltre utile l' operazione

Delete(A,i): cancella A[i] da A