ALGORITMI E STRUTTURE DATI

Prof. Manuela Montangero

A.A. 2022/23

STRUTTURE DATI:
Coda con Priorità

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Coda con priorità

Una coda con priorità memorizza sequenze di COPPIE **`(el,pr)`**:

- **`el`** appartiene ad un insieme di elementi
- **`pr`** è una priorità associabile ad un elemento **`el`**
  (ed appartiene ad un insieme totalmente ordinabile)

Massima priorità: minimo o massimo valore `pr`
(dipende dal problema)

Esempio:

- coppie in coda: (esame, data)
- priorità massima l'esame con la data più vicina (minore)

Esempio:

- coppie in coda: (compito, compenso)
- priorità massima il compito con il compenso più alto (massimo)

Coda con priorità

Una coda con priorità memorizza sequenze di COPPIE **`(el,pr)`**:

- **`el`** appartiene ad un insieme di elementi
- **`pr`** è una priorità associabile ad un elemento **`el`**
  (ed appartiene ad un insieme totalmente ordinabile)

Massima priorità: minimo o massimo valore `pr`

Algoritmo di Dijkstra (cammini minimi sorgente singola):

- coppie in coda (nodo, distanza dalla sorgente)
- ha la priorità massima il nodo con la distanza dalla sorgente minima

Algoritmo di Prim (albero di copertura minimo):

- coppie in coda (arco, peso arco)
- ha la priorità massima il nodo con peso minimo

Algoritmo di Huffman (codici a lunghezza variabile):

- coppie in coda (insieme di caratteri, somma frequenze di occorrenza)
- ha la priorità massima l'insieme con somma di frequenze minima

Coda con priorità

Una coda con priorità memorizza sequenze di COPPIE **`(el,pr)`**:

- **`el`** appartiene ad un insieme di elementi
- **`pr`** è una priorità associabile ad un elemento **`el`**
  (ed appartiene ad un insieme totalmente ordinabile)

PRIMITIVE (massima priorità -> minimo valore di priorità)

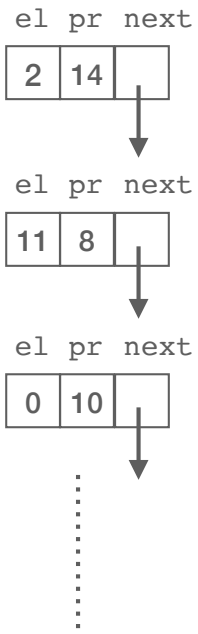
- `Make_priority_queue(Q')`: restituisce una coda con priorità che memorizza le coppie `(el,pr)` in un insieme di coppie `Q'`
- `is_empty_queue(Q)`: restituisce TRUE se la coda è vuota, FALSO altrimenti
- `EnQueue(Q,el,pr)`: modifica la coda con inserimento coppia `(el,pr)`
- `MinQueue(Q)`: restituisce l'elemento con priorità massima (valore minimo) in coda
- `DeQueue(Q)`: modifica la coda con eliminazione della coppia `(el,pr)` tale che `pr` sia minimo (massima priorità) e restituisce `el`
- `Decrease_Priority(Q,el,pr)`: modifica la coppia `(el,pr')` in coda aggiornando la priorità di `el` al nuovo valore `pr < pr'`

Coda con priorità

Realizzazione con

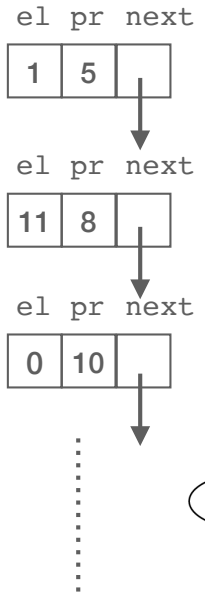
non ordinate

lista di coppie non ordinate



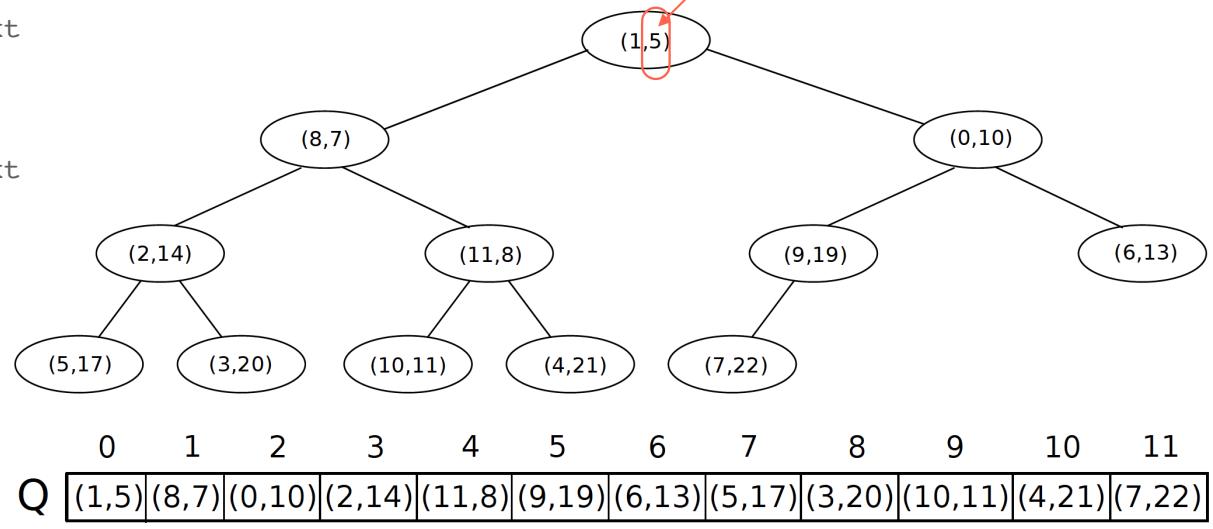
liste ordinate

lista di coppie ordinate per priorità crescente



min-heap

min heap che memorizza coppie,
le chiavi dell'heap sono le priorità



Coda con priorità

Realizzazione con

non ordinate

lista di coppie non ordinate

liste ordinate

lista di coppie ordinate per priorità crescente

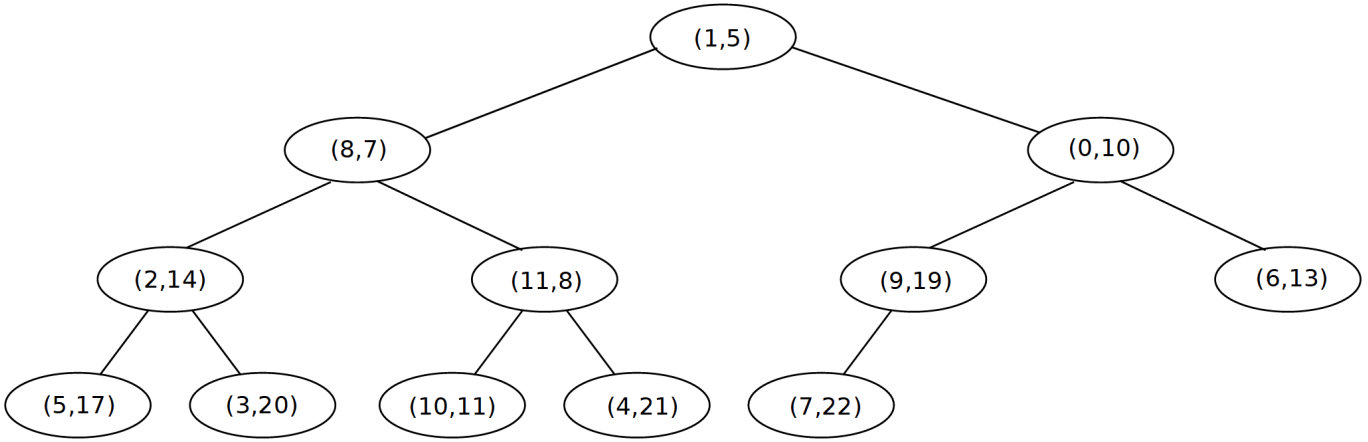
min-heap

min heap che memorizza coppie,
le chiavi dell'heap sono le priorità

PRIMITIVA	Heap	Lista ordinata	Lista non ordinata
BUILDQUEUE(Q)	$O(n)$	$O(n \log n)$	$O(n)$
MINQUEUE(Q)	$O(1)$	$O(1)$	$O(n)$
DEQUEUE(Q)	$O(\log n)$	$O(1)$	$O(n)$
ENQUEUE(Q, e, p)	$O(\log n)$	$O(n)$	$O(1)$

Coda con priorità

Realizzazione con min-heap



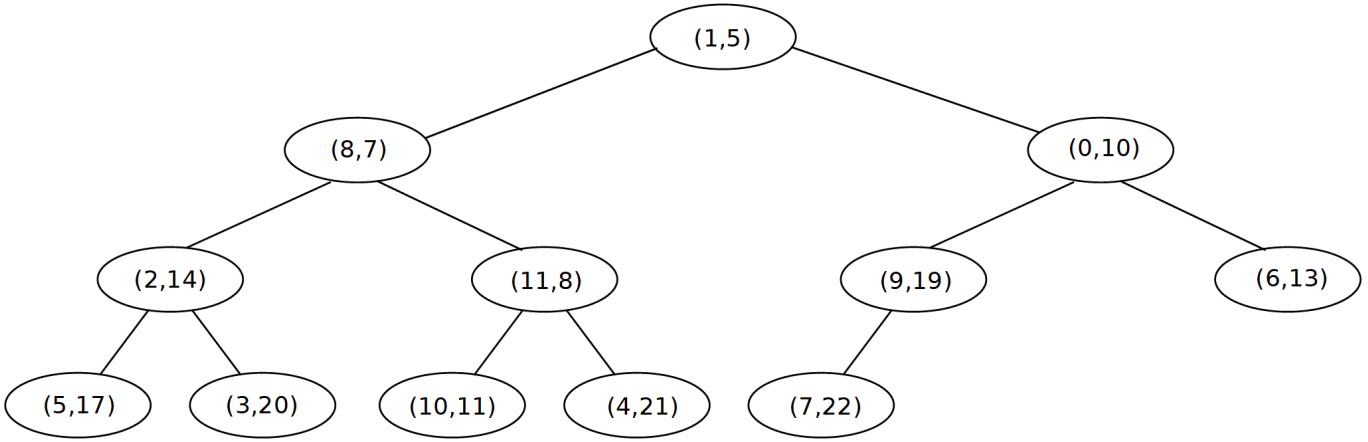
	0	1	2	3	4	5	6	7	8	9	10	11
Q	(1,5)	(8,7)	(0,10)	(2,14)	(11,8)	(9,19)	(6,13)	(5,17)	(3,20)	(10,11)	(4,21)	(7,22)

`Q[i]` è una coppia:

- per accedere al primo elemento: `Q[i].el`
- per accedere al secondo elemento: `Q[i].pr`

Coda con priorità

Realizzazione con min-heap



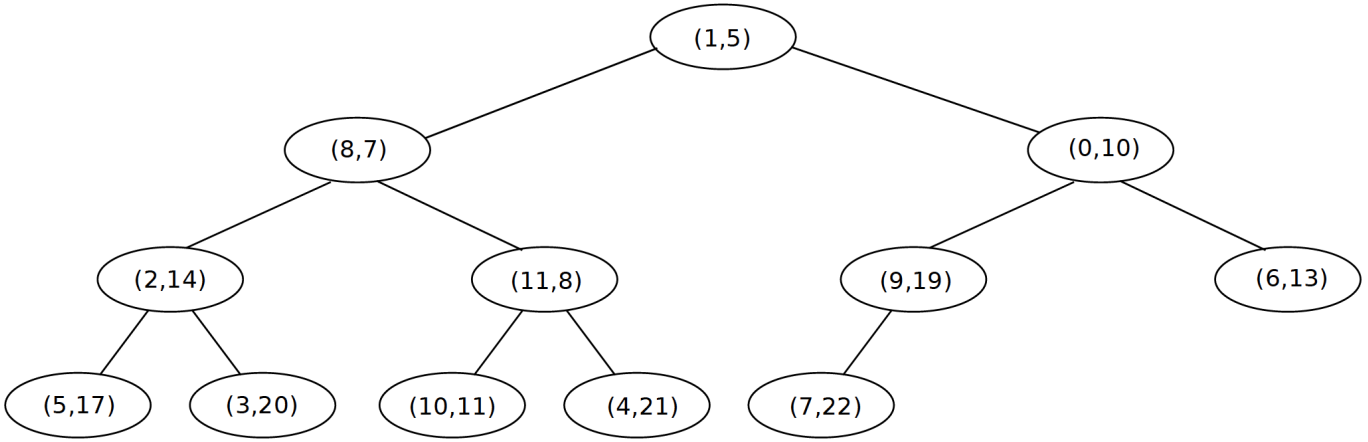
`Decrease_Priority(Q,4,19)`

La primitiva dell'heap ha bisogno dell'indice in Q in cui è memorizzata la coppia (4,21)

`DecrementaChiaveHeap(Q,10,21)`

Coda con priorità

Realizzazione con min-heap



Q

	0	1	2	3	4	5	6	7	8	9	10	11
C	(1,5)	(8,7)	(0,10)	(2,14)	(11,8)	(9,19)	(6,13)	(5,17)	(3,20)	(10,11)	(4,21)	(7,22)

	0	1	2	3	4	5	6	7	8	9	10	11
H	2	0	3	8	10	7	6	11	1	5	9	4

Handle

```
Decrease_Priority(Q,4,19)
```

```
DecrementaChiaveHeap(C,10,21)
```