Lezione_14_DeA

[La parte precedente è la risoluzione di esercizi sugli alberi]

---

5. Priority queue

5.1. Entry

> Una *Entry* è una coppia `(chiave, valore)`, dove la chiave proviene da un dominio K e il valore da un dominio V.

```
public interface Entry<K,V> {
    /** Returns the key of the entry */
    K getKey();
    /** Returns the value of the entry */
    V getValue();
}
```

5.1.1. Esempio

$$\text{Entry} \equiv \text{STUDENTE} \begin{cases} \text{key} = \text{numero matricola} \\ \text{value} = \text{info come anagrafica, esami , ecc.} \end{cases}$$

5.2. Priority Queue - definizione

> Una *Priority Queue* è una collezione di entry le cui chiavi rappresentano *priorità* e provengono da un universo totalmente ordinato K.
> Come tipo di dato astratto, la Priority Queue deve permettere di *trovare/rimuovere la entry di massima priorità* e *inserire una nuova entry*.
> Le chiavi delle entry *non* sono necessariamente distinte.

Convenzionalmente si assume che più piccolo è il valore della chiave e più alta è la priorità, ma vedremo anche utilizzi in cui vale l'opposto.

```
public interface PriorityQueue<K,V> {
    int size();
    boolean isEmpty();
    /** Inserts and returns a new entry (key,value) */
```

```
    Entry<K,V> insert(K key, V value);
    /** Returns an entry with min key, without removing it */
    Entry<K,V> min();
    /** Returns and removes an entry with min key */
    Entry<K,V> removeMin();
}
```

> **Osservazione**
>
> Se esistono più entry con chiave minima, `min` e `removeMin` ne restituiscono (e `removeMin` ne rimuove) una arbitraria.

5.2.1. Esempio

Le seguenti sono una sequenza di operazioni a partire da una Priority Queue vuota.

Operazione	Output	Priority Queue risultante
`insert(5,A)`	$(5, A)$	$(5, A)$
`insert(9,C )`	$(9, C)$	$(5, A)(9, C)$
`insert(3,B)`	$(3, B)$	$(5, A)(9, C)(3, B)$
`insert(7,D)`	$(7, D)$	$(5, A)(9, C)(3, B)(7, D)$
`min()`	$(3, B)$	$(5, A)(9, C)(3, B)(7, D)$
`removeMin()`	$(3, B)$	$(5, A)(9, C)(7, D)$
`size()`	3	$(5, A)(9, C)(7, D)$
`isEmpty()`	$FALSE$	$(5, A)(9, C)(7, D)$

> **Applicazioni delle Priority Queue**
>
> - Algoritmo di Dijkstra per trovare i cammini minimi su un grafo;
> - Pattern discovery;
> - Scheduling di processi in sistema operativo o di richieste di banda nelle reti in base a priorità per garantire QoS;
> - Simulazione discreta a eventi.