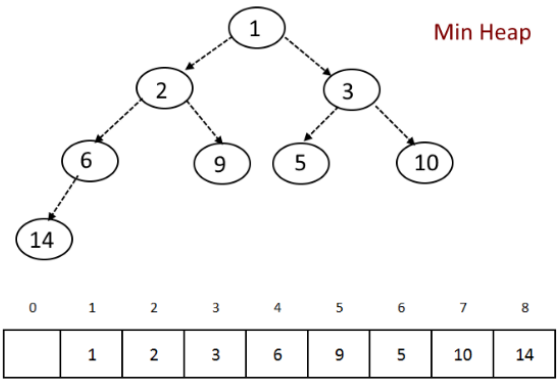
Programmazione ad Oggetti - parte B: Esercitazione di laboratorio 8

Esercizio 1:

Scrivere una classe MinHeapPQ per rappresentare una struttura dati min-heap. Un min-heap è una struttura dati che soddisfa le seguenti condizioni:

1. è un albero binario completo realizzato con un array a partire dall'elemento di indice 1
2. il valore di un nodo figlio è maggiore o uguale a quello del nodo padre

Un min-heap è una struttura dati duale rispetto alla struttura dati "heap" vista a lezione (che più propriamente è chiamata max-heap). Un esempio di min-heap è rappresentato nella seguente figura:



Scrivere il codice del costruttore, distruttore e delle seguenti funzioni membro:

- `int empty()`
- `void insert(Item item)`
- `Item getmin() per estrarre l'elemento più piccolo`

Scrivere anche il codice di un programma principale di esempio per verificare il funzionamento della classe MinHeapPQ nel caso in cui contenga elementi interi (MinHeapPQ<int>). Per risolvere l'esercizio partite dalla classe PQ vista a lezione e modificatela (è necessario modificare anche le funzioni fixUp e fixDown).

Esercizio 2:

Si vogliono attaccare assieme L funi di lunghezza diversa una dopo l'altra. Il costo dell'operazione di attaccare due funi una con l'altra è dato dalla somma della lunghezza delle due funi.

Scrivere un algoritmo che utilizzi un min-heap di numeri interi per calcolare il costo minimo possibile.

Per calcolare il costo minimo possibile si seguano i seguenti passi:

1. creare un min-heap e inserire le lunghezze di tutte le funi
2. inizializzare a 0 il costo totale
3. Ripetere i seguenti passaggi finchè il min-heap non contiene un solo elemento:
    a. Estrarre dal min-heap il minimo elemento e il secondo minimo elemento

   b. Sommare al costo totale i valori dei due elementi estratti
   c. Inserire nel min-heap la somma dei due elementi estratti
4. Stampare a video il costo totale

Per esempio avendo a disposizione 4 funi le cui lunghezze sono salvate in un array

```
int len[L] = { 4, 3, 2, 6 };
```

il costo totale minimo, secondo la procedura proposta è: 5+9+15=29

Esercizio 3:

Scrivere una procedura per creare un unico array di dimensione k*n ordinato in modo crescente partendo da k array (di n elementi ciascuno) già ordinati memorizzati in una matrice.

```
Input:
k = 3, n =  4
arr[k][n] = { {1, 3, 5, 7},
              {2, 4, 6, 8},
              {0, 9, 10, 11}} ;

Output: 0 1 2 3 4 5 6 7 8 9 10 11
```

Per risolvere l'esercizio si seguano i seguenti passi:

1. Creare l'array di output di dimensione k*n
2. Creare un min-heap di numeri interi (di dimensione massima k) e inserire il primo elemento di ciascuno dei k array nel min-heap
3. Ripetere le seguenti istruzioni n*k volte:
   a. Estrarre l'elemento più piccolo dal min-heap e salvarlo nella prima posizione libera disponibile nell'array
   b. Inserire nel min-heap l'elemento successivo (se esiste) preso dallo stesso array a cui apparteneva l'elemento estratto dal min-heap
4. Stampare a video l'array di output ordinato

Suggerimento: definire un min-heap di oggetti di una classe Item che contiene tre membri:

1. Il valore (numero intero) dell'elemento
2. L'indice dell'array di provenienza dell'elemento
3. L'indice dell'elemento successivo nell'array di provenienza dell'elemento