Appello del 5 luglio 2005
Laboratorio di Algoritmi e Strutture Dati – Proff. Blundo e De Bonis
Anno Accademico 2004/2005– Matricole Dispari-Pari e Pari-Dispari

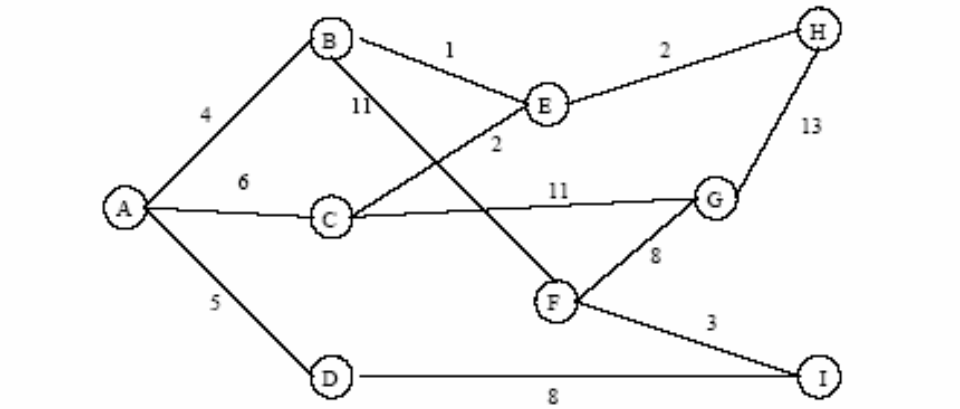
Cognome e Nome: Numero di Matricola:

Spazio riservato alla correzione

1	2	3	4	5	6	7	totale
/13	/12	/18	/10	/13	/14	/10	/90

1 Scrivere lo pseudocodice dell'algoritmo di Dijkstra. Quale è la sua complessità? Giustificare la risposta.

2 Si consideri il grafo pesato in figura



Rappresentare il grafo mediante liste di adiacenza (nella lista i nodi vengono inseriti in ordine lessicografico) ed eseguire l'algoritmo di Dijkstra su di esso per il calcolo dei cammini minimi, a partire dal nodo A. Si riporti per ogni ciclo principale dell'algoritmo: il contenuto dell'insieme Q dei nodi non ancora visitati con le relative priorità; il contenuto dell'insieme R dei nodi già visitati; il vettore delle distanze.

3. Si scriva la funzione **Iterator sameDepth(Tree t, int k)** che ricevuti in input un albero **T** ed un intero **k**, restituisca in output un iteratore su tutti i nodi di **T** che si trovano a profondità **k**. Per l'implementazione non è possibile utilizzare i metodi **elements()** e **positions().** Quale è la complessità del metodo proposto (giustificare la risposta).

4. Aggiungere alla classe **LinkedList** il metodo **Object getAtIndex(int i)** che restituisce l'elemento i-esimo della lista. Se tale elemento non esiste deve lanciare l'eccezione **ElementNotFound**.

5. Aggiungere alla classe D, che implementa un dizionario non ordinato, il metodo

public Iterator EQ().

Il metodo **EQ** restituisce un iteratore di tutti gli elementi il cui valore e` uguale a quello della propria chiave. (Si tenga presente che un elemento puo` essere di un tipo che non e` confrontabile con quello delle chiavi).

Il metodo deve essere scritto senza fare assunzioni sul modo in cui la classe D implementa l'interfaccia Dictionary.

6. Aggiungere alla classe **P** che implementa l'interfaccia **PriorityQueue** il metodo

Object decreaseKey(Object elem, Object NewKey).

Se il valore di **NewKey** e` inferiore a quello della chiave associata ad **elem** allora il metodo pone il valore della chiave associata ad **elem** uguale a **NewKey** e restituisce **elem;** in caso contrario il metodo non effettua alcun aggiornamento e restituisce **null**.

Il metodo deve essere scritto senza fare assunzioni sul modo in cui la classe P implementa PriorityQueue.

7. Il tipo astratto DoppioStack consente le seguenti operazioni:

- **headPush():** inserimento in testa.
- **tailPush():** inserimento in coda.
- **headPop():** rimozione e restituzione dell'elemento in testa.
- **tailPop():** rimozione e restituzione dell' elemento in testa.
- **headTop():** restituzione (senza rimozione) dell'elemento in testa.
- **tailTop():** restituzione (senza rimozione) dell'elemento in coda.

Si implementi in Java tale tipo usando **LinkedList**.