Algoritmi e Strutture Dati

Docente: Sabrina De Capitani di Vimercati

Appello online del 28 Gennaio 2017

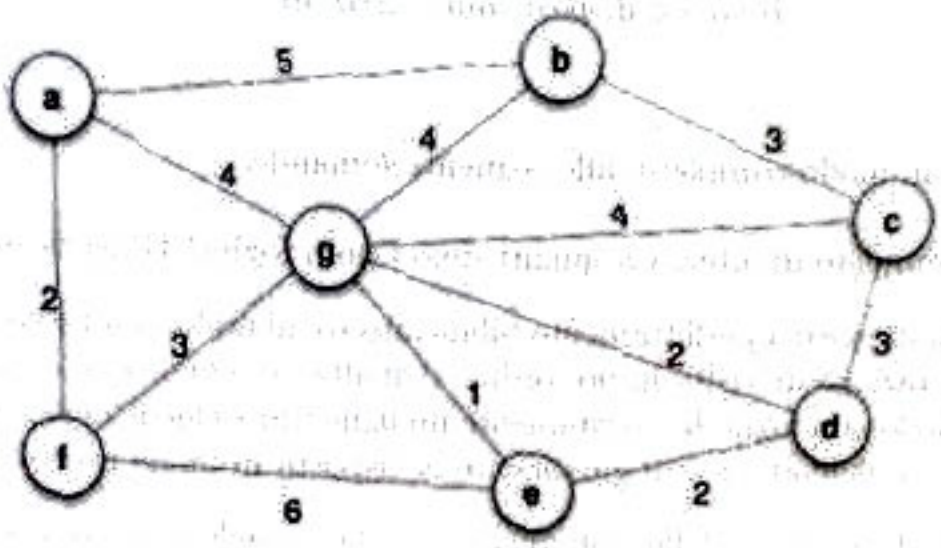
Tempo a disposizione: 2:30 ore

Domanda 1)
Rispondere brevemente, ma in modo completo, alle seguenti domande.

1. In un albero binario completo di altezza h quanti sono i nodi foglia? Giustificare la risposta.
2. Dato un albero binario di ricerca perfettamente bilanciato (ogni nodo non foglia ha due figli) quale è la complessità dell'operazione di visita in pre-ordine dell'albero? Quale è la complessità di tale operazione quando ogni nodo non foglia ha esattamente un figlio? Si richiede di esprimere la complessità in funzione dell'altezza h dell'albero e di giustificare la risposta in modo preciso.
3. Si richiede di definire il *problema della massima diversità*. Quale è la complessità dell'algoritmo Greedy più efficiente per la risoluzione di tale problema? L'uso di uno heap migliora la complessità (giustificare la risposta)?
4. Si richiede di descrivere la realizzazione di una coda di priorità con uno *heap* e di mostrare un esempio sul quale si richiede anche di eseguire l'operazione di cancellamin.
5. Si richiede di descrivere il *problema della cricca* e di mostrare un esempio (con k=20).
6. Si richiede di dimostrare per induzione che $\Sigma_{i=1}^{n} i = \frac{n(n+1)}{2}$.
7. Cosa si intende per *albero di copertura DFS*? Si richiede di mostrare un esempio, illustrando anche il tipi di archi di cui si compone tale albero (in T, in avanti, all'indietro di attraversamento).
8. Si richiede di enunciare il *Teorema di Cook-Levin* e di descrivere le sue conseguenze.
9. A cosa serve l'*algoritmo di Kruskal*? Quale è la sua complessità?
10. Si richiede di dire cosa fa il seguente algoritmo e di calcolarne la complessità computazione (Osservazione: a è un array di numeri di dimensione n). Si richiede di mostare i passi svolti per calcolare la complessità computazionale.

```
Algo(a,ss,dd) {
  if (ss == dd) {
    return false;
  }
  else {
    var = false;
    c = (ss+dd)/2;
    for (i=ss; i<=c; i++) {
      for (j=c+1; j<=dd; j++) {
        if ((a[i]==j)&&(a[j]==i))
          var = true;
      }
    }
    var1 = Algo(a,ss,c);
    var2 = Algo(a,c+1,dd);
    return (var or var1 or var2);
} }
```

Esercizio 1)
Si richiede di determinare due alberi diversi di copertura minima del seguente grafo utilizzando l'algoritmo di Kruskal. Mostrare il procedimento seguito per ottenere tali alberi.



Esercizio 2)
Si richiede di ordinare in ordine crescente l'array contenente i valori [12, 6, 17, 5, 8, 15, 19, 16, 7] utilizzando l'algoritmo heapsort.

Esercizio 3)
Dato un grafo non orientato $G = (V, E)$ rappresentato con una matrice di adiacenza, si richiede di scrivere un algoritmo (pseudocodice) che calcoli il numero di triangoli in G (nota: un triangolo è un insieme di tre nodi distinti $\{a, b, c\}$ tale per cui $\{a, b\}$; $\{b, c\}$ e $\{a, c\}$ appartengono ad E).