Algoritmi e Strutture Dati

a.a. 2016/17

Compito del 26/05/2017

Cognome: ______________________________ Nome: ______________________________

Matricola: ______________________________ E-mail: ______________________________

Parte I

(30 minuti; ogni esercizio vale 2 punti)

1. In una tabella Hash di **m = 17** posizioni, inizialmente vuota, devono essere inserite le seguenti chiavi numeriche nell'ordine indicato:

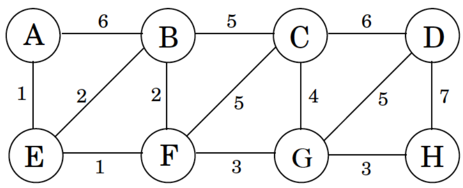
   30, 86, 52, 73

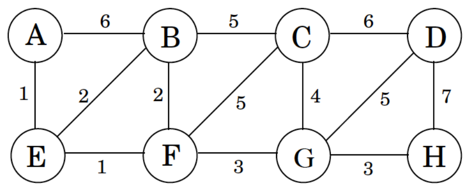
   La tabella è a indirizzamento aperto e la scansione è eseguita per doppio Hashing:

   $$h(k, i) = (k \bmod m + i * 2^{k \bmod 5}) \bmod m$$

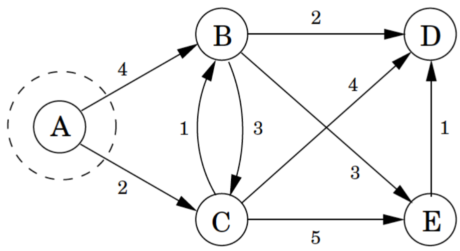
   Indicare per ogni chiave le posizioni scandite nella tabella e la posizione finale dove viene allocata.

2. Si determini un albero di copertura *minimo* ed uno di copertura *massimo* per il seguente grafo (si colorino i corrispondenti archi):





3. Si supponga di eseguire l'algoritmo di Dijkstra sul seguente grafo, utilizzando A come vertice sorgente:



   a) In quale ordine i vertici del grafo verranno estratti dall'algoritmo?
   b) Qual è il vertice avente la massima distanza da A, e qual è la sua distanza?

Algoritmi e Strutture Dati

a.a. 2016/17

Compito del 26/05/2017

Cognome: ______________________________ Nome: ______________________________

Matricola: ______________________________ E-mail: ______________________________

Parte II

(2.5 ore; ogni esercizio vale 6 punti)

1. Scrivere una versione del quicksort le cui prestazioni non degradino su array di input già ordinati o con ripetizioni.
   a. Mostrare un input per il quale questo algoritmo ha le prestazioni migliori, se esiste.
   b. Mostrare un input per il quale questo algoritmo ha le prestazioni peggiori, se esiste.
   c. Dire qual è la complessità dell'algoritmo nel caso medio.

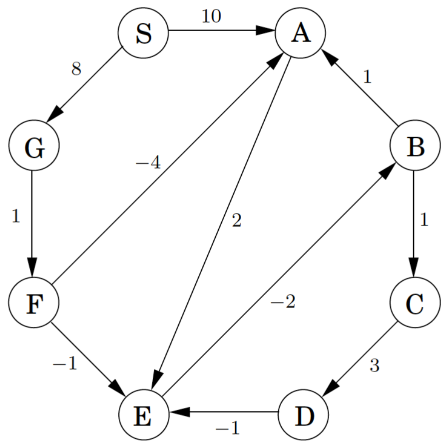
   Per l'esame da **12 CFU**, deve essere fornita **una procedura C**.
   Per l'esame da **9 CFU**, è sufficiente specificare lo pseudocodice.

2. a) Dare la definizione di max-heap.
   b) Utilizzando la **tecnica del divide-et-impera**, si realizzi una funzione *HeapSize(A)* che riceve in input un **array** *A* di dimensione *A.length* organizzato a max-heap, ma senza campo *heap-size*, e calcola il numero di elementi effettivamente contenuti nello heap. Gli elementi dell'array che non fanno parte dello heap hanno convenzionalmente valore *NULL*.
   c) Si valuti la complessità della funzione *HeapSize* assumendo come dimensione del problema la dimensione dell'array. Si deve scrivere e risolvere la ricorrenza.

3. Si consideri il seguente algoritmo, che accetta in ingresso un grafo orientato e pesato $G = (V, E)$, con funzione peso $w : E \rightarrow R$, e un vertice $s \in V$:

```
MyAlgorithm( G, w, s )
n = |V[G]|
for each u ∈ V[G] do
  d[u] = ∞
d[s] = 0
for i = 1 to n do
  for each u ∈ V[G]
    for each v ∈ Adj[u]     /* Adj[u] = insieme dei vertici adiacenti a u */
      d[v] = min { d[v], d[u] + w(u,v) }
return d
```

   a) Qual è la sua complessità?
   b) Quale problema risolve?
   c) L'algoritmo continua ad essere corretto se esegue un'iterazione in meno? Perché? (Giustificare formalmente la risposta.)
   d) Si simuli la sua esecuzione sul seguente grafo, costruendo una tabella avente 8 righe e 8 colonne (dove le righe rappresentano le iterazioni e le colonne i vertici) contenente i valori del vettore d, iterazione per iterazione:



e) Si ripeta la simulazione precedente sostituendo il peso attuale dell'arco (E, B) con il valore -4. Possiamo dire qualcosa sulla correttezza dell'algoritmo in questo caso?

4. Si definisca formalmente la relazione di "riducibilità polinomiale" tra problemi decisionali ($\leq_P$) e si stabilisca se **fra i problemi NP-completi** valgono le seguenti proprietà: *a*) riflessiva, *b*) simmetrica, *c*) transitiva (giustificando tecnicamente le risposte).

   Inoltre, siano $\mathcal{P}$ e $\mathcal{Q}$ due problemi in NP e si supponga $\mathcal{P} \leq_P \mathcal{Q}$. Si stabilisca se le seguenti affermazioni sono vere o false, fornendo una dimostrazione nel primo caso e un controesempio nel secondo:

   a) "Se $\mathcal{Q}$ è risolvibile in tempo quadratico, allora $\mathcal{P}$ è risolvibile in tempo quadratico"
   b) "Se $\mathcal{Q}$ è risolvibile in tempo esponenziale, allora $\mathcal{P}$ è risolvibile in tempo esponenziale"
   c) "Se $\mathcal{Q}$ è un problema NP-completo, allora $\mathcal{P}$ è NP-completo"
   d) "Se $\mathcal{P}$ è un problema NP-completo, allora $\mathcal{Q}$ è NP-completo"

   (**Nota:** in caso di discussioni poco formali l'esercizio non verrà valutato pienamente.)