Algoritmi e Strutture Dati

a.a. 2020/21

Compito del 17/01/2022

Cognome: _____ Nome: _____

Matricola: _____ E-mail: _____

Parte I

(30 minuti; ogni esercizio vale 2 punti)

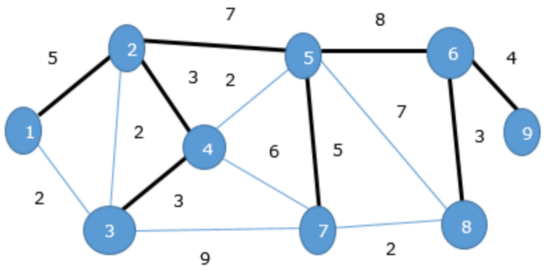
Avvertenza: Si giustifichino tecnicamente tutte le risposte. In caso di discussioni poco formali o approssimative gli esercizi non verranno valutati pienamente.

1. Completare la seguente tabella indicando la complessità delle operazioni che si riferiscono a un dizionario di n elementi. Per l'operazione **Predecessore** si assuma di essere sull'elemento x a cui si applica l'operazione.

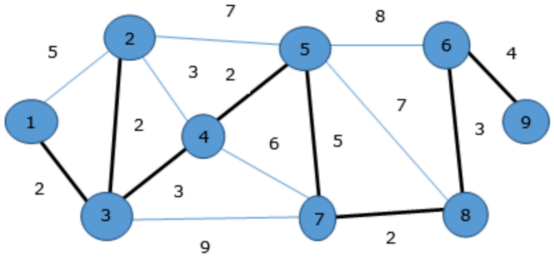
	Ricerca	**Minimo**	**Predecessore**	**Costruzione**
Tabelle Hash a indirizzamento aperto (caso medio)*				
Array ordinato in senso crescente				
Alberi binari di ricerca				

*La Tabella Hash ha dimensione m e il fattore di carico è α.

2. Si dica se nei grafi sottostanti gli archi indicati in grassetto formano o meno un albero di copertura minimo:



(a)



(b)

3. Siano $\mathcal{P}$ e $\mathcal{Q}$ due problemi in NP e si supponga $\mathcal{P} \leq_P \mathcal{Q}$. Si stabilisca se la seguente affermazione è vera o falsa: "Se $\mathcal{P}$ è un problema NP-completo, allora $\mathcal{Q}$ è NP-completo". Potremmo dire la stessa cosa se $\mathcal{Q} \leq_P \mathcal{P}$? Perché?

Algoritmi e Strutture Dati

a.a. 2020/21

Compito del 17/01/2022

Cognome: _____ Nome: _____

Matricola: _____ E-mail: _____

Parte II

(2.5 ore; ogni esercizio vale 6 punti)

Avvertenza: Si giustifichino tecnicamente tutte le risposte. In caso di discussioni poco formali o approssimative gli esercizi non verranno valutati pienamente.

1. Scrivere una funzione **efficiente** in C o in C++ `blackHeight(u)` che dato in input la radice `u` di un albero binario, i cui nodi `x` hanno, oltre ai campi `key`, `left` e `right`, un campo `col` che può essere `B` (per "black") oppure `R` (per "red"), verifica se per ogni nodo, il cammino da quel nodo a qualsiasi foglia contiene lo stesso numero di nodi neri (altezza nera del nodo `x`). In caso negativo, restituisce -1, altrimenti restituisce l'altezza nera della radice.
   a. Valutare la complessità della funzione, **indicando eventuali relazioni di ricorrenza**.
   b. Scrivere il tipo del nodo dell'albero in C o in C++ a seconda del linguaggio scelto.

2. Si consideri la struttura dati Max-Heap implementata tramite un vettore (secondo lo schema visto a lezione).
   a) Si scriva la definizione di Max-Heap.
   b) Si realizzi in modo **efficiente** in **pseudocodice** la funzione Differenza(H1,H2) che dati due Max-Heap H1 e H2 contenenti rispettivamente n1 e n2 interi (anche ripetuti), ritorna in output un nuovo Max-Heap contenente gli elementi che appartengono a H1 ma non appartengono a H2. In presenza di duplicati se x compare k1 volte in H1 e k2 volte in H2, nel Max-Heap differenza x dovrà comparire max{0, k1-k2}.
   c) Si determini e giustifichi la complessità in funzione di n1 e n2.

   Si devono scrivere le eventuali funzioni/procedure ausiliarie utilizzate.

3. Per un certo problema sono stati trovati due algoritmi risolutivi (A_1 e A_2) con i seguenti tempi di esecuzione (dove n rappresenta la dimensione dei dati di ingresso):

   A_1 : $T(n) = 4 \cdot T(n/4) + n^2$
   A_2 : $T(n) = T(n/4) + T(3n/4) + n$

   Si dica quale dei due algoritmi è preferibile per input di dimensione sufficientemente grande. (Suggerimento: nel secondo caso si utilizzi la tecnica dell'albero di ricorsione e si consideri la lunghezza del cammino più lungo dalla radice alle foglie.)

4. Sia $G = (V, E)$ un grafo orientato con funzione peso $w : E \rightarrow \mathrm{R}$ e vertici numerati da 1 a n: $V = \{1, 2, \ldots, n\}$. Si scriva un algoritmo che, per ogni coppia di vertici $i, j \in V$, determini la lunghezza del cammino minimo tra i e j i cui vertici intermedi non superino $n - 1$. Si dimostri la correttezza dell'algoritmo proposto e si determini la sua complessità.