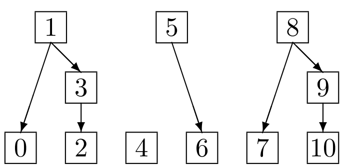
Complementi di Algoritmi e Strutture Dati

(III anno Laurea Triennale - a.a. 2017/18)

Prova scritta 6 giugno 2018

NB: I punteggi sono indicativi.

Esercizio 1 – Union find (Punti 6) Considerare la foresta union-find sottostante e un'implementazione di tipo *quick-union* che usa *union-by-size*.

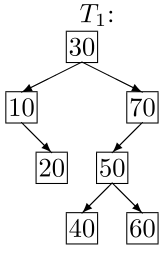


1. Indicare, per ogni radice, il "size" del corrispondente albero.
2. Eseguire nell'ordine le seguenti operazioni (ogni operazione va eseguita sul risultato della precedente):
   union(1,4)
   union(6,7)
   union(6,2)
   Per ogni operazione, spiegare che cosa viene fatto e perché, disegnare il nuovo albero e indicare il "size" della sua radice.
3. Considerare l'esecuzione delle operazioni di cui al punto precedente se l'implementazione adotta la *compressione dei cammini.* Se cambia qualcosa, dire che cosa e perché; se non cambia niente, dire perché.

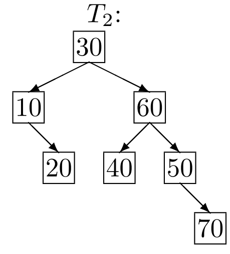
Esercizio 2 – Alberi AVL (Punti 5)

1. Per ciascuno di questi due alberi, dire se si tratta di un albero AVL oppure no, e perché.

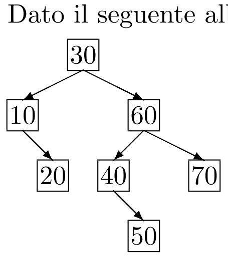
   T_1:

   T_2:

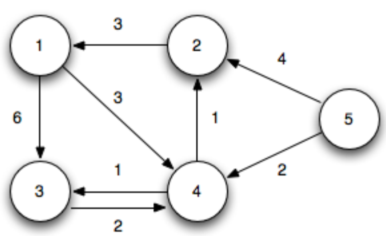
   

2. Dato il seguente albero AVL, indicare i fattori di bilanciamento dei nodi.

3. Sullo stesso albero AVL del punto precedente, eseguire in successione questi tre inserimenti, facendo vedere che cosa succede e spiegando perché:
   (a) inserire la chiave 35;
   (b) inserire la chiave 25;
   (c) inserire la chiave 56.

Esercizio 3 - (Punti 7) Si esegua l'algoritmo di Floyd-Warshall sul seguente grafo pesato:



Più precisamente:

1. Si scrivano sei matrici con righe e colonne indiciate 1, 2, 3, 4, 5, corrispondenti a considerare come nodi intermedi nei cammini: nessun nodo, solo 1, anche 2, anche 3, anche 4, anche 5. Per semplicità scrivete per ogni iterazione un'unica matrice le cui caselle contengono sia la distanza (matrice $\mathtt{D}^k$ nelle note) sia il predecessore (matrice Π^k nelle note). Per scrivere meno, potete indicare in ogni matrice solo le caselle modificate rispetto alla precedente.

2. Si disegni il sottografo dei cammini minimi determinato dalla matrice.

3. Si disegni l'albero dei cammini minimi a partire dal nodo 1.

Esercizio 4 - (Punti 8) Ricordiamo che un *insieme indipendente* in un grafo non orientato $G = (V, E)$ è un insieme $V' \subseteq V$ di nodi tale che per ogni coppia di essi non esiste l'arco che li collega. Abbiamo visto che il problema dell'*insieme indipendente* (determinare se in G esiste un insieme indipendente di dimensione k) è NP-completo. Invece, il problema dell'*insieme indipendente massimale* (dato G restituire un insieme indipendente tale che aggiungendo un qualunque altro nodo non sia più indipendente) è risolvibile in tempo polinomiale.

1. Dare un esempio di grafo con un insieme indipendente massimale di dimensione molto più piccola di un insieme indipendente di dimensione massima.

2. Dare un semplice algoritmo polinomiale (greedy) che risolva il problema dell'insieme indipendente massimale.

3. Provare la correttezza dell'algoritmo dato.

Esercizio 5 - (Punti 7) Consideriamo il problema di trovare il minimo e il massimo in un array $A[1..n]$. Un ovvio algoritmo, che richiede $2(n-1) = 2n-2$ confronti tra elementi, consiste nell'effettuare una prima scansione dell'array per trovare il minimo, e una seconda per trovare il massimo. Una soluzione migliore è un algoritmo divide-et-impera che divide l'array a metà e utilizza i risultati (coppie minimo/massimo) dei due sottoproblemi.

1. Si descriva in pseudocodice l'algoritmo divide-et-impera suggerito sopra.

2. Si calcoli il numero di confronti tra elementi richiesto da questo algoritmo.