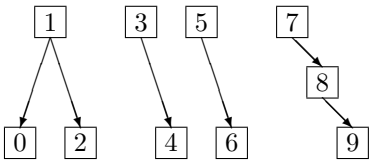
Complementi di Algoritmi e Strutture Dati

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

Prova scritta 18 luglio 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(1,3)
   union(3,6)
   union(4,9)
   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 – Sorting e alberi AVL (Punti 6) Lo schema astratto di heapsort per ordinare n elementi consiste nell'inserire prima tutti i numeri in uno heap, poi tirarli fuori uno alla volta dallo heap, sfruttando il fatto che lo heap li ritornerà ordinati (in modo crescente o decrescente a seconda che sia uno heap a minimo o a massimo).
Possiamo pensare di applicare la stessa idea usando un albero AVL invece che uno heap.

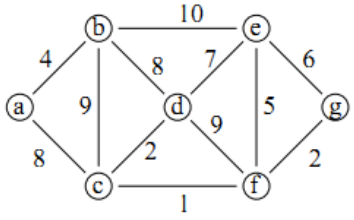
1. Scrivere lo pseudocodice di questo algoritmo "AVL-sort", assumendo la sequenza s implementata ad array. Nota: ovviamente, al contrario dello heap, l'albero AVL non potrà essere costruito "sul posto".

   Occorre indicare le operazioni dell'AVL che vengono usate (dire come si chiamano e che cosa fanno, non dare l'algoritmo).
2. Quale sarebbe la complessità temporale nel caso peggiore di un tale algoritmo "AVL-sort"? Giustificare la risposta.
3. Mostrare i passi dell'algoritmo per ordinare la sequenza $s = 28, 8, 36, 52, 55, 1, 30, 32, 15, 90$.

Esercizio 3 - Relazioni di ricorrenza (Punti 7) Per ognuna delle seguenti relazioni di ricorrenza, si dia la soluzione utilizzando il metodo delle sostituzioni successive e si provi la correttezza della soluzione per induzione aritmetica.

1. $T(n) = \begin{cases} 3T(n-1) & \text{se } n > 0 \\ 1 & \text{altrimenti} \end{cases}$
2. $T(n) = \begin{cases} 2T(n-1) - 1 & \text{se } n > 0 \\ 1 & \text{altrimenti} \end{cases}$

Esercizio 4 - Grafi (Punti 6) Si esegua, sul seguente grafo:



l'algoritmo di Prim a partire dal nodo a. Inizialmente quindi si avrà `dist`(a)=0 e `dist`=∞ per tutti gli altri nodi. Per ogni iterazione del ciclo while si dia:

- il nodo che viene estratto con la `getMin`
- i nodi per i quali viene modificata `dist` e come
- il minimo albero ricoprente alla fine dell'iterazione, evidenziando chiaramente la parte di albero definitiva.

Non dovete disegnare lo heap.

Esercizio 5 - Ordinamenti (Punti 7) Si consideri il seguente algoritmo.

```
ord(a,inf,sup)
  if (inf < sup)
    if (a[inf] > a[sup]) swap(a,inf,sup)
    ord(a,inf + 1,sup − 1)
    if (a[inf] > a[inf + 1]) swap(a,inf,inf + 1)
    ord(a,inf + 1,sup)
```

1. Questo algoritmo ordina correttamente `a[inf..sup]`? Si giustifichi la risposta.
2. Si scriva la relazione di ricorrenza che descrive il numero di confronti in funzione della lunghezza $n = \texttt{sup} - \texttt{inf} + 1$.
3. Si valuti la complessità dell'algoritmo.