Complementi di Algoritmi e Strutture Dati

(III anno Laurea Triennale - a.a. 2016/17)

Prova scritta 19 luglio 2017

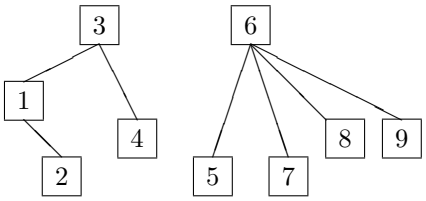
NB: I punteggi sono indicativi.

Esercizio 1 – Ordinamenti (Punti 6) Si consideri il seguente array, sul quale è stata già eseguita la prima fase dell'algoritmo heapsort, cioè l'array è già stato trasformato in uno heap a massimo:

80	65	60	50	40	17	25	30	15	35	20

1. Disegnare la rappresentazione ad albero dello heap.
2. Eseguire la seconda fase dell'algoritmo heapsort, cioè quella che produce l'array ordinato. Ad ogni passo mostrare il nuovo stato di tutto l'array e la rappresentazione ad albero della parte di array che è heap.

Esercizio 2 – Union-find (Punti 5) Considerare la foresta union-find sottostante e un'implementazione di tipo *quick-union*.



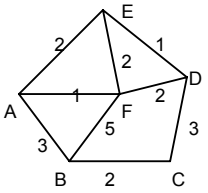
1. Indicare, per ogni radice, il "size" e il "rank" del corrispondente albero.
2. Eseguire l'operazione *union*(4,7) nelle due versioni:
   a) supponendo *union-by-size*
   b) supponendo *union-by-rank*
   SENZA compressione dei cammini. Spiegare che cosa viene fatto e perché, disegnare il nuovo albero e indicare il "size" e il "rank" della sua radice.
3. Domanda identica alla precedente ma CON compressione dei cammini.
4. Domanda identica alla domanda 2 ma per l'operazione *union*(2,7).
   Nota bene: eseguire l'operazione sulla foresta iniziale (NON sul risultato dell'operazione di cui alla domanda 2.
5. Domanda identica alla precedente ma CON compressione dei cammini.

Esercizio 3 - Analisi di algoritmi (Punti 7) Si consideri il seguente algoritmo ricorsivo.

```
fun(n)
if (n>1)
  a = 0
  for (i = 1; i < n; i++)
    for (j =i+1; j <=n; j++) a = a + 2 * (i + j)
  for (i = 1; i <= 16; i++) a = a + fun(n/4);
  return a
else return n-1
```

Scrivere e risolvere la relazione di ricorrenza che descrive il costo computazionale di `fun` in funzione di `n`.

Esercizio 4 - Grafi (Punti 7) Si consideri il seguente grafo pesato.



Si costruisca il minimo albero ricoprente utilizzando:

1. l'algoritmo di Kruskal (per ogni iterazione, si diano l'arco esaminato e la foresta corrente)
2. l'algoritmo di Prim a partire dal nodo A (per ogni iterazione, si diano le distanza provvisorie `dist` di tutti i nodi e l'albero corrente, evidenziandone la parte definitiva)

Nel caso di più scelte possibili si usi come convenzione l'ordine alfabetico.

Esercizio 5 - Analisi di algoritmi (Punti 8) Si consideri il seguente *coffee can problem* (*problema del barattolo di caffè*), dovuto a David Gries. Sia C un barattolo che contiene chicchi bianchi o neri, e indichiamo con $N(C)$ il numero di chicchi contenuti in C. Supponiamo inoltre di avere un numero illimitato di chicchi neri a disposizione fuori dal barattolo, che possono essere inseriti dentro.

```
//Pre: N(C) ≥ 2
while (N(C)>1)
  estrai due chicchi da C
  se hanno lo stesso colore eliminali e inserisci un chicco nero in C
  altrimenti ri-inserisci il chicco bianco in C ed elimina il nero
```

1. Si provi che l'algoritmo termina.
2. Si provi, utilizzando un'opportuna invariante, che il colore dell'ultimo chicco rimasto è bianco se e solo se il numero di chicchi bianchi presenti inizialmente è dispari.