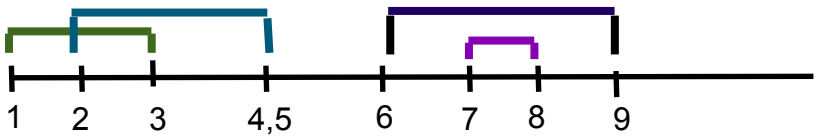
Introduzione agli algoritmi
Prova di esame del 1/2/2017
Prof.sse E. Fachini - R. Petreschi
Parte I

1. Si illustri l'algoritmo di costruzione di un heap massimo a partire da un array qualunque. Se ne discuta la correttezza e si dimostri che il tempo di esecuzione asintotico è O(n).

2. Si risponda ad ognuna delle seguenti domande, motivando la risposta:
   A) Se un algoritmo ha tempo di esecuzione $\Theta(n^2)$ nel caso peggiore, posso dedurne che nel caso migliore terminerà in $\Theta(n^2)$ passi?
   B) Se si dimostra che un algoritmo ha tempo di esecuzione $\Theta(n^2)$ nel caso migliore, è possibile che in qualche caso l'algoritmo termini in O(n) passi?
   C) Se si dimostra che un algoritmo ha tempo di esecuzione $\Theta(n^2)$ nel caso peggiore, è possibile che in qualche caso l'algoritmo termini in O(n) passi?

3. Dati n intervalli della retta reale, si descriva un algoritmo che calcoli la lunghezza della loro unione in O(n log n) passi. Per esempio, la lunghezza dell' intervallo [1, 3] è 2, la lunghezza dell'unione dei quattro intervalli [1, 3], [6, 9], [2, 4.5] e [7, 8] è 6.5. Si spieghi la correttezza dell'algoritmo descritto e se ne analizzi il tempo di esecuzione asintotico. Suggerimento: utilizzare un array di coppie <inizio, fine> come struttura dati per memorizzare gli intervalli. Nel disegno sottostante trovate i quattro intervalli dell'esempio raffigurati su una retta.



Introduzione agli algoritmi
Prova di esame del 1/2/2017
Prof.sse E. Fachini - R. Petreschi

Parte II

1. Si dimostri che l'altezza h di un AVL T (o rosso nero) è O(lg n), dove n è il numero di nodi di T.

2. Si consideri la seguente funzione:

```
analizzami(int n)
    c = 1
       m = n/2
    while m>1 do
          for j=1 to m do c++
          m=m - 2
    if n>1 then analizzami(n/2)
```

   Si imposti la relazione di ricorrenza che ne descrive la complessità e la si risolva utilizzando il metodo della sostituzione

3. Un albero si dice *albero binario di ricerca opposto* se, per ogni nodo v, tutti i nodi nel sotto albero sinistro di v hanno chiave ≥ chiave(v) e tutti i nodi nel sotto albero destro di v hanno chiave ≤ chiave(v). Progettare un algoritmo che, dato un albero binario di ricerca T_1, lo trasformi in un albero binario di ricerca opposto T_2, distruggendo T_1.