Algoritmi e Strutture Dati

a.a. 2013/14

Compito del 12/06/2014

Cognome: _____ Nome: _____

Matricola: _____ E-mail: _____

Parte I

(30 minuti; ogni esercizio vale 2 punti)

1. Dato un albero binario di ricerca T, scrivere la funzione *Tree-maximum(Node x)* che restituisce un nodo con chiave massima di un sottoalbero di T radicato nel nodo x *(*x è un nodo dell'albero T *e* $x \neq NULL$*)*. Quale è la complessità di questa funzione? Quale è il caso migliore? E il caso peggiore?

2. Si confrontino le seguenti funzioni utilizzando le relazioni O, Ω e Θ:

$f(n)$	$g(n)$
$100\,n + \log n$	$n + (\log n)^2$
$\log n$	$\log n^3$
2^n	3^n

3. Si riempia la tabella sottostante, specificando le complessità degli algoritmi indicati in funzione della tipologia di grafo utilizzato:

	Grafo sparso	Grafo denso
Dijkstra (array)		
Dijkstra (heap)		
Bellman-Ford		

Supponendo che il grafo sia aciclico, quale algoritmo conviene usare? Perché?

Algoritmi e Strutture Dati

a.a. 2013/14

Compito del 12/06/2014

Cognome: _____ Nome: _____

Matricola: _____ E-mail: _____

Parte II

(2.5 ore; ogni esercizio vale 6 punti)

1. In una tabella Hash di **m = 17** posizioni, inizialmente vuota, devono essere inserite le seguenti chiavi numeriche nell'ordine indicato:

   23, 40, 15, 58, 85

   Indicare per ogni chiave la posizione finale dove viene allocata in questi due casi:

   a. Funzione Hash: h(k) = k mod m

      Risoluzione delle collisioni mediante concatenamento

   b. La tabella è a indirizzamento aperto e la scansione è eseguita per doppio Hashing:

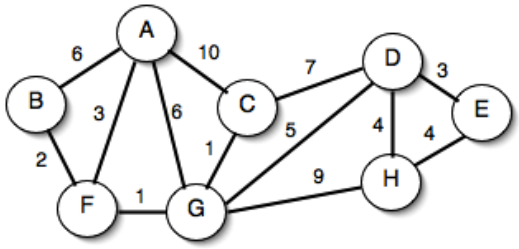
      $$h(k, i) = (k \bmod m + i * 2^{k \bmod 5}) \bmod m$$

      Si devono indicare tutte le posizioni scandite nella tabella.

2. Progettare un algoritmo **efficiente** per stabilire se un albero binario è **quasi completo**, cioè tutti i livelli dell'albero sono completamente riempiti, tranne eventualmente l'ultimo che ha le foglie addossate a sinistra. Calcolare la complessità al caso pessimo dell'algoritmo indicando, e risolvendo, la corrispondente relazione di ricorrenza.
   La rappresentazione dell'albero binario utilizza esclusivamente i campi **left, right** e **key.**

   Per l'esame da **12 CFU**, deve essere fornita **una funzione C**.
   Per l'esame da **9 CFU**, è sufficiente specificare lo pseudocodice.

3. Si scriva l'algoritmo di Kruskal, si dimostri la sua correttezza, si fornisca la sua complessità computazionale e si simuli accuratamente la sua esecuzione sul seguente grafo



4. Si supponga di voler cambiare una banconota da P euro in monete da $a_1, a_2, \ldots, a_k$ euro. E' possibile? In caso affermativo, qual è il minimo numero di monete da utilizzare per effettuare il cambio? Per esempio, se si volesse cambiare una banconota da 5 euro in monete da 1 e da 2 euro, sarebbero necessarie almeno 3 monete. Se si disponesse soltanto di monete da 2 euro, invece, il cambio non sarebbe possibile.

   Si formuli questo problema come un problema di cammini minimi su grafi. (Suggerimento: si costruisca un grafo con $P + 1$ vertici numerati da 0 a P e si inseriscano archi e pesi in modo tale che i cammini dal vertice 0 al vertice P corrispondano a possibili sequenze di monete da utilizzare per il cambio.)

   A titolo esemplificativo, si consideri il caso di una banconota da $P = 10$ euro e monete da $a_1 = 1$ e $a_2 = 2$ euro. Si costruisca il grafo corrispondente e si utilizzi l'algoritmo di Dijkstra per risolvere il problema.