ASD (FI2 12 CFU - tutti), ASD (5 CFU)

Appello del 20-7-2016 – a.a. 2015-16 – Tempo a disposizione: *120 minuti* – somma punti: 32 + 3

Problema 1

`[Punti: (a) 3/30; (b) 3/30]]`

Si considerino i metodi Java di seguito illustrati.

```
// assumere tutti i parametri >= 0

static long succ(long i) { return i+1; }

static long sum(long a, long b) { return sum2(a, b, 0); }

static long sum2(long a, long b, long count) {
    if(b == 0) return a;
    if(count == b) return a;
    return sum2(succ(a), b, succ(count));
}

static long prod(long a, long b) {
    if(b == 0) return 0;
    if(b == 1) return a;
    return prod2(a, b, 1, a);
}

static long prod2(long a, long b, long count, long res) {
    if(count < b) return prod2(a, b, succ(count), sum(res, a));
    return res;
}

static long pot(long a, long b) {
    if(a == 0) return 0;
    if(b == 0) return 1;
    if(b == 1) return a;
    return pot2(a, b, 1, a);
}

static long pot2(long a, long b, long count, long res) {
    if(count < b) return pot2(a, b, succ(count), prod(res, a));
    return res;
}
```

Sviluppare, *argomentando adeguatamente* (il 50% del punteggio dell'esercizio sarà sulle argomentazioni addotte), quanto segue:

(a) Determinare il costo asintotico dell'algoritmo descritto da `sum(long, long)` in funzione di z, dimensione dell'input.

(b) Determinare il costo asintotico dell'algoritmo descritto da `prod(long, long)` in funzione di z, dimensione dell'input.

Opzionale (3 p.ti): Determinare il costo asintotico dell'algoritmo descritto da `pot(long, long)` in funzione di z, dimensione dell'input.

Problema 2

`[Punti: 9/30]`

Dati due BST t_1 e t_2, aventi rispettivamente $n_1 > 0$ ed $n_2 > 0$ chiavi (ciascun BST ha chiavi distinte), con $n_1 \leq n_2$, scrivere un algoritmo (Java oppure C) non necessariamente in-place, che determini in tempo $O(n_1 + n_2)$ se le chiavi di t_1 sono un sottoinsieme di quelle presenti in t_2.

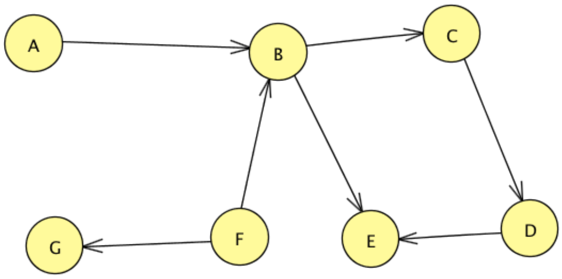
Problema 3

Miscellanea argomenti.

1. Descrivere la tecnica di gestione delle collisioni nota come *linear probing.* (3 punti)

2. Descrivere (pseudo-codice) un algoritmo efficiente (tempo e spazio) per ordinare un array di elementi già quasi ordinato (max k inversioni presenti, con k costante non negativa). (3 punti)

3. Descrivere (pseudo-codice) un algoritmo di topological sort e mostrarne il funzionamento sul grafo in figura. (3 punti)



Problema 4

`[Punti: (a) 5/30; (b) 3/30]`

Data una mappa stradale rappresentata da un opportuno grafo pesato e orientato $G = (V, E, w)$, essendo V l'insieme dei nodi, $E \subseteq V \times V$ l'insieme degli archi orientati e $w : E \mapsto \mathbb{R}^+$ la funzione di pesatura (costo) degli archi, si vuole risolvere il problema di determinare un percorso di costo minimo fra due nodi assegnati.

In particolare, si richiede di:

(a) Presentare un algoritmo (pseudo-codice) che dato il grafo $G = (V, E, w)$ e due nodi $u, v \in V$, restituisca un percorso orientato di costo minimo da u a v, o NULL se tale percorso non esiste.

(b) Determinare i costi dell'algoritmo nei casi di rappresentazione per: liste di adiacenza, matrice delle adiacenze. Il 50% del punteggio sarà associato alla spiegazione dei costi.