FI2 (12 CFU), ASD Nettuno (6 CFU) - Compito di esame del 12-9-2012 – a.a. 2011-12
tempo a disposizione: 90 minuti

Problema 1

Un k-dado è un dado con k facce, numerate con i valori da 1 a k. Il metodo `kDadoProb(int k, int n, int p)` calcola e restituisce la probabilità di ottenere un risultato `p`, lanciando `n` `k`-dadi.

```
static double kDadoProb(int k, int n, int p) { // assumere: k > 0, n >= 0
    return kDadoConta(k, n, p) / (double)pow(k,n);
}

static long pow(int a, int b) { // assumere: a > 0, b >= 0
    if(b == 0) return 1;
    long f = pow(a, b >> 1);
    if(b%2 == 0) return f*f;
    else return f*f*a;
}

static long kDadoConta(int k, int n, int p) {
    if(n<=0) return 0;
    if(n == 1)
        if((p < 1) || (p > k)) return 0;
        else return 1;
    long cont = 0;
    for(int i = 1; i <= k; i++) cont += kDadoConta(k, n-1, p-i);
    return cont;
}
```

(a) Calcolare, motivandolo adeguatamente, il costo computazionale del metodo `kDadoProb(int k, int n, int p)`, in funzione di `k`, `n` e `p`.

(b) Riscrivere il costo calcolato al punto (a) in funzione della dimensione dell'input,

Problema 2

Si intende rappresentare, per finalità didattiche, lo schema di esecuzione di un algoritmo di merge-sort ricorsivo tramite un albero binario. A tal scopo è necessario progettare una apposita casse Java che, durante l'esecuzione del merge-sort, costruisce un corrispondente albero delle ricorsioni, i cui nodi sono associati a una singola attivazione dell'algoritmo ricorsivo e contengono i valori dei parametri espliciti. Assumendo per semplicità che i dati da ordinare siano numeri interi contenuti in un array, svolgere i seguenti punti:

(a) Definire due classi Java, `RecursionTree` e `RecursionNode`, per rappresentare alberi (binari) di ricorsione che descrivono la struttura dell'esecuzione di un algoritmo ricorsivo di merge-sort. Le classi debbono consentire le normali operazioni relative a un albero binario e debbono essere specificate definendo variabili membro e firma di metodi/costruttori.

(b) Definire un metodo Java `static RecursionTree mergeSort(int[] a)` della classe `RecursionTree`, che ordini l'array `a` e restituisca l'albero delle ricorsioni effettuate durante la sua esecuzione. Determinare il costo computazionale del metodo.

(c) Dato un albero delle ricorsioni, definire un metodo Java che visiti i nodi dell'albero nello stesso ordine in cui le relative esecuzioni sono state attivate, stampando a schermo i dati ad essi associati. Determinare il costo computazionale del metodo.

Problema 3

Svolgere i punti seguenti.

(a) Descrivere il problema della gestione efficiente di una collezione di insiemi disgiunti attraverso le tecniche di union-find, definire le operazioni previste e i relativi algoritmi, precisando i costi computazionali.

(b) Con riferimento al tipo astratto *grafo semplice* pesato sugli spigoli, definire il problema di determinare un minimum spanning tree (MST, minimo albero ricoprente) e illustrare un algoritmo efficiente per il calcolo del MST basato sulle tecniche di union-find di cui al punto (a).

(c) Con riferimento all'algoritmo fornito al punto (b) indicarne i costi computazionali nei casi: (c1) rappresentazione del grafo tramite liste di adiacenza; (c2) rappresentazione del grafo tramite matrice di adiacenza.