Informatica Generale
Secondo Esonero

Giancarlo Bongiovanni, Tiziana Calamoneri, Ivano Salvo
Sapienza Università di Roma
11 Giugno 2018

Esercizio 1 (10 punti) Data una struttura dati ad albero T, si consideri il problema di eliminare la radice da T e ripristinare le proprietà di T eventualmente andate perse.

T sia dapprima un heap massimo e poi un albero binario di ricerca. Pertanto, un'istanza del problema è nel primo caso V (in cui V è il vettore che memorizza l'heap massimo) e nel secondo *pr* (in cui *pr* è il puntatore alla radice dell'albero binario di ricerca memorizzato come records e puntatori).

Si progettino due funzioni `Delete_Root_From_Heap` e `Delete_Root_From_ABR` che, presi in input V nel primo caso e *pr* nel secondo, eliminino il nodo radice dall'albero, ripristinandone poi la struttura.

Degli algoritmi proposti, si diano:

1. **(2+2 punti)** la descrizione a parole, evidenziando somiglianze e differenze;
2. **(2+2 punti)** lo pseudocodice;
3. **(1+1 punti)** il costo computazionale.

Esercizio 2 (10 punti) Una lista potrebbe contenere un puntatore che punta a un nodo precedente (generando quindi un ciclo): questa situazione va gestita con cautela, in quanto, ad esempio, le funzioni che abbiamo visto sulle liste assumono sempre la precondizione che la lista termini con un puntatore `NULL` e quindi non terminerebbero se ci fossero cicli. Definiamo:

1. *lista circolare* una lista in cui l'"ultimo" nodo punta al "primo", dove il primo nodo è il nodo puntato dalla variabile di tipo lista (vedi Fig. 1**a**);

2. *lasso* come una lista con un prefisso lineare che entra in una lista circolare (vedi Fig. 1**b**).

Scrivere una funzione C di prototipo: `char decomponi(list L, int n, list* P, int *p, int list *C, int *q)` che riceve in input una lista puntata da `L` e il numero di nodi distinti `n` e: (1) restituisce il carattere `c` se `L` è circolare, `l` se `L` è un lasso e `s` se `L` è una lista semplice. (2) carica in `*P` il puntatore alla testa del prefisso lineare di L (sarà `NULL` se L è circolare) e in `*p` la lunghezza di tale prefisso; (3) carica in `*C` il puntatore al ciclo di L (sarà `NULL` se L è una lista semplice) e carica in `*q` la lunghezza del ciclo (Fig. 1**c**).

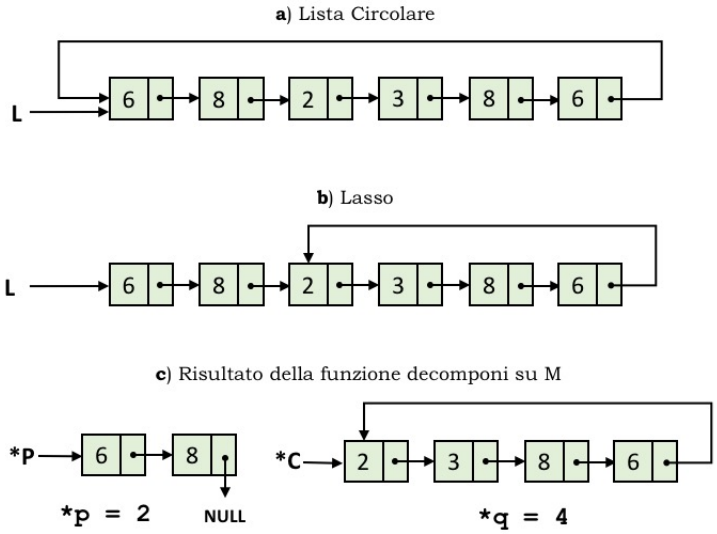

Figura 1: a) Lista circolare, b) Lasso c) Memoria dopo `decomponi` su M.

Esercizio 3 (**12 punti**) Dato un grafo $G = (V, E)$, un suo *triangolo* è un insieme di tre nodi distinti $\{u, v, w\}$ di G tali che i tre archi (u, v), (v, w) e (w, u) siano in E.

(**2 punti**) Si calcoli il numero dei triangoli in un grafo completo con n nodi.

Si progetti poi un algoritmo che, ricevendo in input un grafo G memorizzato tramite matrice di adiacenza, restituisca il numero dei suoi triangoli.

Di tale algoritmo:

1. (**3 punti**) si dia la descrizione a parole;
2. (**3 punti**) si scriva lo pseudo-codice;
3. (**2 punti**) si calcoli il costo computazionale;
4. (**2 punti** FACOLTATIVO) si discuta come varia il costo computazionale nel caso in cui G non sia dato tramite matrice di adiacenza ma tramite liste di adiacenza.