02MNO Algoritmi e Programmazione 01JKE APA I / 01JKF APA II

Appello del 03/09/2014 - Prova di teoria (12 punti)

1. (**1 punto**)
Sia data la seguente sequenza di coppie, dove la relazione i-j indica che il nodo i è adiacente al nodo j:

2-10, 4-2, 0-4, 6-1, 2-7, 4-3, 0-8, 0-4, 7-8, 10-3

si applichi un algoritmo di on-line connectivity con quickfind, riportando a ogni passo il contenuto del vettore e la foresta di alberi al passo finale. I vertici sono denominati con interi tra 0 e 10.

2. (**2 punti**)
Sia data la sequenza di interi, supposta memorizzata in un vettore:

21 2 11 3 7 33 13 5 16 9 17 1

- la si trasformi in un heap, ipotizzando di usare un vettore come struttura dati. Si riportino graficamente i diversi passi della costruzione dell'heap ed il risultato finale. Si ipotizzi che, alla fine, nella radice dell'heap sia memorizzato il valore massimo.
- si eseguano su tale heap i primi 2 passi dell'algoritmo di heapsort.

NB: la sequenza è già memorizzata nel vettore e rappresenta una configurazione intermedia per cui la proprietà di heap non è ancora soddisfatta.

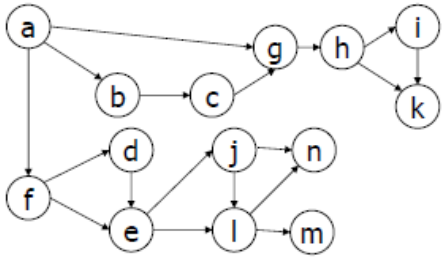
3. (2 punti)
Si inseriscano in sequenza in foglia in un BST, inizialmente supposto vuoto, le chiavi:

9 24 31 68 11 13 58 8 75

e al termine dell'inserzione si partizioni il BST secondo la chiave mediana.

4 (2 punti)
Sia data la sequenza di chiavi HUNGE$_1$RGAME$_2$S, dove ciascun carattere è individuato dal suo ordine progressivo nell'alfabeto (A=1, ..., Z=26) e da un eventuale pedice. Si riporti la struttura di una tabella di hash di dimensione 23, inizialmente supposta vuota, in cui avvenga l'inserimento della sequenza indicata. Si supponga di utilizzare l'open addressing con quadratic probing. Si selezionino opportuni valori per c_1 e c2.

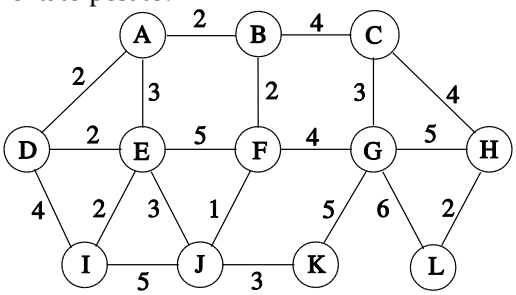
5. (2 punti)
Si ordini topologicamente il seguente DAG. Qualora necessario, si trattino i vertici secondo l'ordine alfabetico e si assuma che la lista delle adiacenze sia anch'essa ordinata alfabeticamente.



6. (1 punto)
Si trasformi il grafo dell'esercizio 5 nel corrispondente grafo non orientato e se ne determinino i punti di articolazione. Si consideri **a** come vertice di partenza e, qualora necessario, si trattino i vertici secondo l'ordine alfabetico.

7. (2 punti)

Sia dato il seguente grafo non orientato pesato:



se ne determini un minimum spanning tree applicando l'algoritmo di Kruskal, disegnando l'albero e ritornando come risultato il valore del peso minimo. Si esplicitino i passi intermedi.

02MNO Algoritmi e Programmazione 01JKE APA I / 01JKF APA II

Appello del 03/09/2014 - Prova di programmazione (12 punti)

1. (2 punti)

Si implementi la funzione

```
void searchStr (char *str, int *start, int *length);
```

che riceve in ingresso la stringa `str` e rintraccia in tale stringa la sequenza di caratteri uguali di lunghezza maggiore, ritornandone l'indice di inizio nella variabile `start` e la sua lunghezza nella variabile `length`.

Ad esempio, se la funzione ricevesse la stringa

abbcccddddeeeee

dovrebbe rintracciare la sequenza eeeee e ritornare i valori start = 10 e length = 5.

2. (4 punti)

Una stringa contiene delle sottostringhe, costituite da caratteri esclusivamente alfabetici e numerici, delimitate dal carattere punto "`.`". Si scriva la funzione:

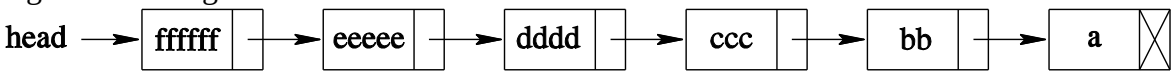
```
node_t *splitStr (char *str);
```

in grado di ricevere la stringa `str` con un tale formato e di restituire il puntatore a un lista in cui ciascun elemento contiene una delle sottostringhe della stringa originaria, definita dinamicamente. Si riporti inoltre la definizione del nodo della lista.

Ad esempio, se la funzione ricevesse la stringa

a.bb.ccc.dddd.eeeee.fffff

dovrebbe generare la seguente lista:



nella quale le stringhe sono allocate dinamicamente, restituendo il puntatore `head`.

3. (6 punti)

Gli alberi di grado *n* possono essere rappresentati mediante nodi ciascuno contenente *n* puntatori (Fig. 1(a)) oppure, per evitare di appesantire ciascun nodo di un albero con un numero eccessivo di puntatori, mediante la tecnica del *left-child right-sibling.* Con questa tecnica in ogni nodo vi sono un puntatore al figlio più a sinistra e al fratello del nodo immediatamente a destra. La Fig. 1(b) mostra lo stesso albero della Fig. 1(a) rappresentato con la tecnica del *left-child right-sibling.*

Si supponga che ciascun nodo dell'albero rappresentato come *left-child right-sibling* sia di tipo `node_t` e che includa due stringhe dinamiche (cognome e nome) e il voto di un esame (valore intero). Si definisca la struttura C relativa a un nodo dell'albero. Si scriva inoltre la funzione ricorsiva in grado di visualizzare tutto il contenuto dell'albero stesso una volta ricevuta la radice dell'albero quale parametro.

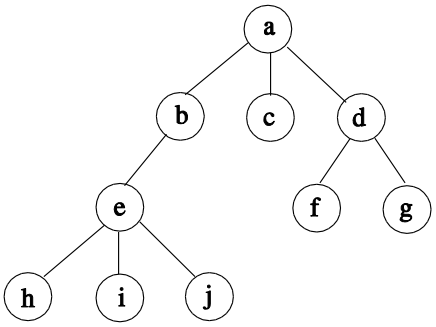


Fig. 1(a)

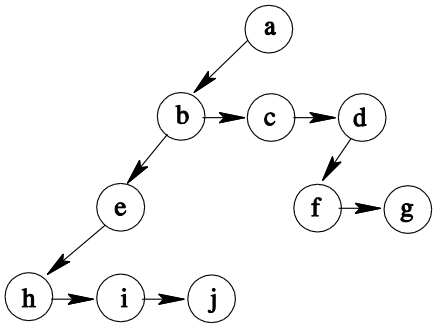


Fig. 1(b)

02MNO Algoritmi e Programmazione 01JKE APA I / 01JKF APA II

Appello del 03/09/2014 - Prova di programmazione (18 punti)

Un progettista deve configurare una rete, della quale inizialmente sono dati solo i nodi. L'obiettivo del programma sarà quindi di determinare gli archi:

- o verificando una soluzione generata manualmente
- o generandola automaticamente.

Un file contiene l'elenco dei nodi uno per riga come stringa di caratteri univoca di lunghezza pari al massimo a 20 caratteri. Il numero di nodi non è noto a priori. Il nome del file compare sulla riga di comando.

Esempio di file

`Nodi.txt`

```
nodoA
nodoB
nodoC
nodoD
```

Il progettista deve stabilire quali sono gli archi (bidirezionali) tra i nodi per ottenere una rete connessa che rispetti tutti i seguenti vincoli:

- il numero di archi deve essere minimo
- per ogni coppia di vertici (v_i, v_j) la lunghezza del cammino minimo che li connette non deve superare un valore intero *k* che compare sulla riga di comando
- ogni nodo ha grado non superiore a *m*,, valore intero che compare sulla riga di comando.

Si ricordi che per rete connessa si intende una rete in cui per ogni coppia di vertici (v_i, v_j) esiste un cammino che li connette).

Si scriva un programma C che:

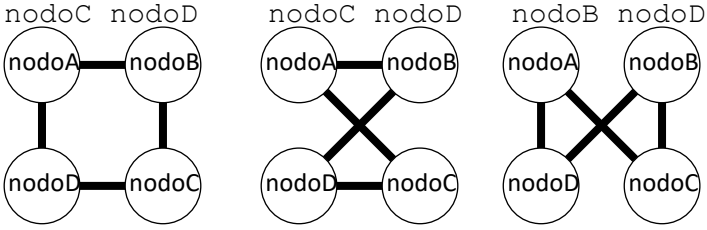
- legga il file contenente i nodi e memorizzi le sue informazioni in un'opportuna struttura dati
- generi gli archi in una delle seguenti 2 modalità alternative con scelta da input:
  - legga un secondo file contenente un elenco di archi, uno per riga nel formato v_i v_j, e verifichi se questo insieme soddisfa le seguenti condizioni:
    - ogni coppia di vertici (v_i, v_j) è connessa da un cammino lungo al massimo *k*
    - ogni vertice ha grado non superiore a *m*

    Si assuma che tutti gli archi contenuti nel file siano coerenti con l'elenco dei nodi del primo file.
  - generi automaticamente una qualsiasi soluzione ottima che soddisfi i vincoli e la scriva sotto forma di elenco di archi, uno per riga, in un file di output il cui nome compare sulla riga di comando.

Esempio:

per `Nodi.txt` con *k*=2 e *m*=2 le seguenti 3 soluzioni rispettano i vincoli:

```
nodoA nodoB     nodoA nodoB   nodoA nodoC
nodoA nodoD     nodoA nodoC   nodoA nodoD
nodoB nodoC     nodoB nodoD   nodoB nodoC
nodoC nodoD     nodoC nodoD   nodoB nodoD
```



PER ENTRAMBE LE PROVE DI PROGRAMMAZIONE (18 o 12 punti):

- ***indicare nell'elaborato e nella relazione (oltre a nome, cognome e numero di matricola) anche il nome del corso per cui si sta sostenendo l'esame (AP, APA I+II).***
- ***È consentito utilizzare chiamate a funzioni <u>standard</u>, quali ordinamento per vettori, inserzione/estrazione/ricerca relative a FIFO, LIFO, liste, BST, tabelle di hash e altre strutture dati, considerate come librerie esterne. Gli header file delle librerie utilizzate devono essere allegati all'elaborato. Le funzioni richiamate, inoltre, dovranno essere incluse nella versione del programma allegata alla relazione.***
- ***Consegna delle relazioni (per entrambe le tipologie di prova di programmazione): entro lunedì 08/09/2014, alle ore 23:59, via e-mail all'indirizzo: danilo.vendraminetto@polito.it, usando come subject (oggetto) la stringa APA#<m>, essendo <m> il proprio numero di matricola. L'allegato alla mail deve essere costituito da <u>un unico file</u>: un archivio compresso, contenente sia il codice corretto, sia la relazione (<u>NO eseguibili</u>). QUALORA IL CODICE SPEDITO CON LA RELAZIONE NON COMPILI CORRETTAMENTE, VERRÀ APPLICATA UNA PENALIZZAZIONE. Si ricorda che la valutazione del compito viene fatta senza discussione o esame orale, <u>sulla base dell'elaborato svolto in aula</u>. Non verranno corretti i compiti di cui non sarà stata inviata la relazione nei tempi stabiliti.***