03MNO Algoritmi e Programmazione

Appello del 02/07/2019 - 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: 3-6, 6-2, 0-8, 5-3, 4-8, 4-7, 9-8, 3-5, 6-8, 9-0. Si applichi un algoritmo di on-line connectivity con **weighted** quickunion, riportando a ogni passo il contenuto del vettore e la foresta di alberi al passo finale. I nodi sono denominati con interi tra 0 e 9.

2. (2.5 punti)

Si inseriscano in sequenza i seguenti dati, composti da una stringa e da un intero che ne rappresenta la priorità, in una coda a priorità di indici inizialmente supposta vuota:

`"ab" 20  "cfd" 32  "FF" 19  "aAd" 51  "rt" 28`

Si ipotizzi di usare uno heap (priorità massima in radice, vettore di 5 celle) per la coda a priorità e che la tabella di hash di dimensione 11 per la tabella di simboli abbia il seguente contenuto:

0	1	2	3	4	5	6	7	8	9	10
aAD					FF			cfd	ab	rt
51					19			32	20	28
3					2			1	0	4

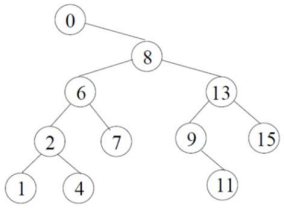
Si disegnino lo heap e il vettore `qp` ai diversi passi dell'inserzione. Al termine si cambi la priorità di `"rt"`da 28 a 60 e si disegnino i corrispondenti heap e vettore `qp`.

3. (3 x 0.5 punti)

Si disegni l'albero corrispondente alla seguente espressione in forma prefissa * A + – B C / + D E F e si trasformi l'espressione in forma infissa e postfissa.

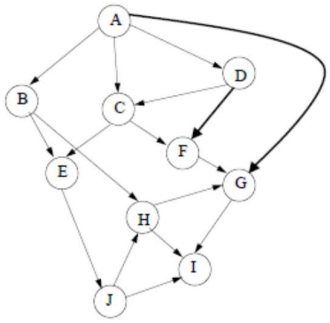
4. (2 punti)

Si aggiungano al seguente BST le informazioni che permettono di realizzate l'operazione di select e, indicando i passaggi, si identifichi la chiave di rango 8:



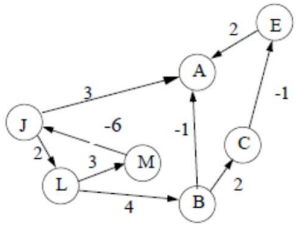
5. (2 punti)

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. (2+1 punti)

Si determinino per il seguente grafo orientato pesato mediante l'algoritmo di Bellman-Ford i valori di tutti i cammini minimi che collegano il vertice **J** con ogni altro vertice (**2punti**). Si verifichi l'eventuale presenza di un ciclo a peso negativo (**1 punto**).Si assuma, qualora necessario, un ordine alfabetico per i vertici e gli archi.



03MNO Algoritmi e Programmazione

Appello del 02/07/2019 - Prova di programmazione (12 punti)

1. (2 punti)

Si scriva un programma C che, dato un intero `N`, generi una matrice quadrata `NxN` con tutti 0 sulla diagonale principale, tutti 1 sulle 2 diagonali immediatamente sopra e sotto, tutti 2 su quelle sopra e sotto le precedenti, etc, come nel seguente esempio per `N=5`

0	1	2	3	4
1	0	1	2	3
2	1	0	1	2
3	2	1	0	1
4	3	2	1	0

2. (4 punti)

Una lista linkata, accessibile mediante il puntatore `head1` alla testa, contiene una sequenza di caratteri alfanumerici. Si definisca il tipo link/nodo utilizzati e scriva una funzione C che generi una seconda lista, accessibile mediante il puntatore `head2` alla testa, in cui sono state eliminate le eventuali sottosequenze di elementi delimitati da coppie di parentesi tonde (aperta all'inizio della sequenza, chiusa alla fine della sequenza), sostituendole con un solo elemento asterisco "*". Si assuma che la lista sia corretta, cioè che non possa contenere coppie di parentesi che si "intersecano" e che per ogni parentesi aperta esista una parentesi chiusa. Il prototipo della funzione sia:

```
link purgeList(link head1);
```

Esempio: data la lista

ab (a c g) b e ( ) a (x x) f

la lista di output sarà:

a b (*) b e (*) a (*) f

Ai fini dell'esercizio, non si può fare uso di funzioni di libreria.

3. (6 punti)

Sia dato un vettore `V` di `N` interi. Si definisce **sottosequenza** di `V` di lunghezza `k` (`k`≤`N`) un qualsiasi n-upla `Y` di `k` elementi di `V` con indici crescenti, ma non necessariamente contigui $i_0, i_1, \cdots, i_{k-1}$. La sottosequenza si dice **bitonica crescente/decrescente** se i suoi elementi sono ordinati prima in modo crescente e poi decrescente. Esempio: la sequenza 4, 5, 9, 7, 6, 3, 1 è bitonica.

Si scriva un programma C che, ricevuti come parametri il vettore V e l'intero N calcoli e visualizzi una qualsisasi sottosequenza bitonica di lunghezza massima.

Esempio: se N=9 e V = [4, 2, 5, 9, 7, 6, 10, 3, 1] una sottosequenza bitonica di lunghezza massima 7 è 4 5 9 7 6 3 1.

Si osservi che una sequenza crescente è bitonica di lunghezza massima, come pure una sequenza decrescente.
Esempio: se N=5 e V = [1, 2, 3, 4, 5], essa stessa è una sequenza bitonica di lunghezza massima 5.

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

- ***indicare nell'elaborato e nella relazione nome, cognome e numero di matricola.***
- ***se non indicato diversamente, è consentito utilizzare chiamate a funzioni <u>standard</u>, quali ordinamento per vettori, funzioni su FIFO, LIFO, liste, BST, tabelle di hash, grafi e altre strutture dati, considerate come librerie esterne.***
- ***gli header file devono essere allegati all'elaborato (il loro contenuto riportato nell'elaborato stesso). Le funzioni richiamate, inoltre, dovranno essere incluse nella versione del programma allegata alla relazione. I modelli delle funzioni ricorsive non sono considerati funzioni standard.***
- ***consegna delle relazioni (per entrambe le tipologie di prova di programmazione): entro venerdì 05/07/2019, alle ore 23:59, mediante caricamento su Portale. Le istruzioni per il caricamento sono pubblicate sul Portale nella sezione Materiale). <u>QUALORA IL CODICE CARICATO CON LA RELAZIONE NON COMPILI CORRETTAMENTE, VERRÀ APPLICATA UNA PENALIZZAZIONE.</u> Si ricorda che la valutazione del compito viene fatta, senza la presenza del candidato, <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.***

03MNO Algoritmi e Programmazione

Appello del 02/07/2019 - Prova di programmazione (18 punti)

Un'Agenzia della Mobilità deve pianificare in quali dei Comuni del suo territorio posizionare delle stazioni di ricarica pubbliche per veicoli elettrici. I dati noti sono:

- i Comuni del territorio coinvolto sono `N` e sono identificati dagli interi tra 0 e `N-1`
- il loro numero di abitanti è memorizzato in un vettore `pop` di `N` interi
- le loro distanze reciproche sono memorizzate in una matrice `dist NxN` di interi.

Si vuole ottimizzare la localizzazione delle stazioni di ricarica in base a funzioni obiettivo diverse e indipendenti:

- **funzione obiettivo 1**: data una distanza intera massima `distMax`, determinare il numero minimo `stazMin` di stazioni di ricarica e la loro localizzazione in modo che tutti i Comuni distino meno di `distMax` dalla stazione di ricarica più vicina. In ogni Comune è localizzata al più una stazione
- **funzione obiettivo 2**: dato un numero fisso intero di stazioni di ricarica posizionabili sull'insieme del territorio `numStaz`, dato il numero massimo di stazioni di ricarica localizzabili in ciascun Comune memorizzato in un vettore `stazComune` di interi ≥1, determinare una qualsiasi localizzazione che minimizzi la distanza, pesata in base alla popolazione e al numero di stazioni localizzate in un dato Comune, tra ogni Comune e la stazione di ricarica ad esso più vicina

$$\textbf{min} \sum_{i=0}^{N-1} pop_i * distMinDaStazione_i \,/numStazioniAdistMin_i$$

Esempio: vi siano 5 Comuni caratterizzati da:

dist	0	1	2	3	4
0	0	8	10	7	12
1	8	0	7	9	11
2	10	7	0	10	9
3	7	9	10	0	8
4	12	11	9	8	0

- **funzione obiettivo 1**: se `distMax=8,` il numero minimo di stazioni di ricarica da localizzare è `stazMin=2` e una soluzione è `sol=(1,3)`. La matrice sottostante riporta le distanze minime di ogni Comune dalla soluzione:

dist	0	1	2	3	4
0	0	8	10	7	12
1	8	0	7	9	11
2	10	7	0	10	9
3	7	9	10	0	8
4	12	11	9	8	0

- **funzione obiettivo 2**: se `numStaz=2,` se i vettori `stazComune` e `pop` sono quelli riportati sotto

stazComune	1	1	4	3	2
	0	1	2	3	4

pop	15	5	50	30	25
	0	1	2	3	4

la soluzione `sol=(2,3)` porta ad un valore minimo della funzione obiettivo 2 di:

340 = 7x15/1+7x5/1+0x50/1+0x30/1+8x25/1

La soluzione `sol=(2,2)` porta ad un valore **non** minimo della funzione obiettivo 2 di:

355 = 10x15/2+7x5/2+0x50/2+0x30/2+9x25/2

Si scriva un programma in C che, una volta acquisite le informazioni del problema:

- legga da file, con formato libero, una proposta di allocazione di risorse e verifichi se essa rispetti o meno le condizioni imposte dalla **funzione obiettivo 1**
- individui una soluzione ottima che rispetti i criteri della **funzione obiettivo 1** mediante un algoritmo ricorsivo
- individui una soluzione ottima che rispetti i criteri della **funzione obiettivo 2** mediante un algoritmo ricorsivo

L'introduzione di tecniche di pruning sarà oggetto di valutazione.

3MNO Algoritmi e Programmazione