03MNO Algoritmi e Programmazione

Appello del 29/01/2018 - Prova di teoria (12 punti)

1. **(2.5 punti)**
Si risolva la seguente equazione alle ricorrenze mediante il metodo dello sviluppo (unfolding):

$$T(n) = 3T(n/4) + 1 \qquad n>1$$
$$T(1) = 1 \qquad n=1$$

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

1 10 72 41 71 0 8 55 91 14 32 19 13 73 7 3 31

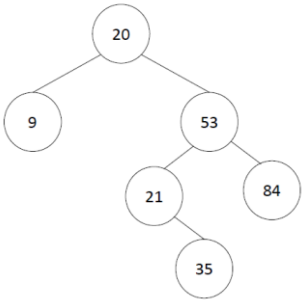
Si eseguano i primi 2 passi dell'algoritmo di quicksort per ottenere un ordinamento **discendente.** NB: I passi sono da intendersi, impropriamente, come in ampiezza sull'albero della ricorsione, non in profondità. Si chiede, pertanto, che siano ritornate le 2 partizioni del vettore originale e le due partizioni delle partizioni trovate al punto precedente.

3. **(2.5 punti)**
Data la catena di matrici (A_1, A_2, A_3, A_4) di dimensioni (5x7), (7x6), (6x3) e (3x5) rispettivamente, si determini mediante un algoritmo di programmazione dinamica la parentesizzazione ottima del prodotto di matrici che minimizza il numero di moltiplicazioni.

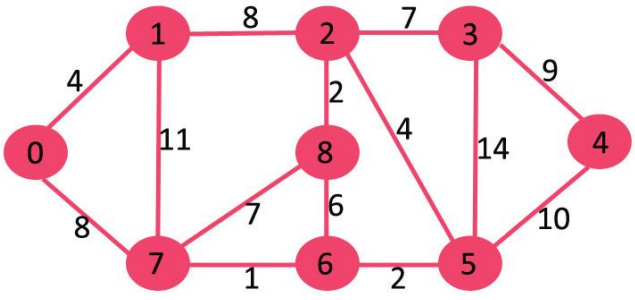
4. **(1 punto)**
Si converta la seguente espressione da forma infissa a forma postfissa:

(A – B) / ((C / D) + (E / (F - G)))

5. **(2 punti)**
Si inseriscano in **radice** nel BST di figura in sequenza le chiavi: 16, 19 e 22 e poi si cancelli la chiave 19. Si disegni l'albero ai passi significativi.



6. **(2 punti)**
Si determini mediante l'algoritmo di Kruskal l'albero ricoprente minimo per il grafo non orientato, pesato e connesso in figura illustrando i passaggi intermedi del procedimento adottato. Si disegni l'albero e si ritorni come risultato il valore del peso minimo.



03MNO Algoritmi e Programmazione

Appello del 29/01/2018 - Prova di programmazione (18 punti)

1. (**18 punti**)

Ad un'agenzia per il lavoro si rivolgono *N* persone in cerca di impiego e *N* aziende ciascuna delle quali è alla ricerca di una persona da assumere. L'agenzia organizza gli N^2 colloqui delle *N* persone in cerca di impiego presso le *N* aziende ed al termine chiede ad ogni persona di elencare le aziende in ordine di preferenza decrescente e ad ogni azienda di elencare le persone in ordine di preferenza decrescente. Una volta acquisiti questi dati, l'agenzia cerca di assegnare ogni persona a una e una sola azienda (matching tra persone e aziende). Un matching è un vettore di *N* coppie (persona, azienda).

Dato un matching, una qualsiasi delle *N(N-1)* coppie persona/azienda (*p, a*) **alternative** al matching è una **fonte di instabilità** per quel matching se e solo se sono vere entrambe le seguenti condizioni (AND logico):

- la persona *p* preferisce l'azienda *a* all'azienda a cui è stata assegnata nel matching
- l'azienda *a* preferisce la persona *p* alla persona che le è stata assegnata nel matching.

Un matching è **perfetto** se e solo se nessuna delle *N(N-1)* coppie persona/azienda **alternative** al matching è una **fonte di instabilità.** In altri termini, in un matching perfetto nessuna delle coppie **alternative** al matching è tale per cui, per entrambi i membri, l'assegnazione proposta da questa coppia sarebbe stata migliore di quella assegnata dal matching ad ognuno dei membri.

Esempio: se con *N* =4 le persone p0, p1, p2 e p3 hanno espresso le preferenze per le aziende a0, a1, a2 e a3 riportate nella matrice P e le aziende a0, a1, a2 e a3 hanno espresso le preferenze per le persone p0, p1, p2 e p3 riportate nella matrice A

P

a1	a3	a0	a2
a2	a0	a3	a1
a1	a2	a0	a3
a3	a0	a2	a1

A

p1	p0	p3	p2
p3	p2	p0	p1
p0	p3	p2	p1
p1	p0	p3	p2

Il matching `match` = (p0, a0), (p1, a2), (p2, a1), (p3, a3) non è perfetto in quanto la coppia (p0, a3) è fonte di instabilità: la persona p0 preferisce l'azienda a3 all'azienda a0 cui è stato assegnata && l'azienda a3 preferisce la persona p0 alla persona p3 che le è stata assegnata.
I `match` = (p0, a3), (p1, a2), (p2, a1), (p3, a0) e (p0, a3), (p1, a0), (p2, a1), (p3, a2) sono perfetti perché nessuna coppia tra le *N(N-1)* coppie alternative al matching è fonte di instabilità rispetto al matching.

Si scriva un programma C che, ricevuti sulla riga di comando:

- come primo argomento un intero *N*
- come secondo argomento il nome di un file che contiene *N* gruppi di 2 righe, dove la prima contiene il nome della persona e la seconda N nomi di aziende in ordine di preferenza decrescente
- come terzo argomento il nome di un file che contiene *N* gruppi di 2 righe, dove la prima contiene il nome dell'azienda e la seconda N nomi di persone in ordine di preferenza decrescente
- come quarto argomento il nome di un file che contiene un possibile matching persone/aziende (*N* coppie di nomi persona/azienda)

- acquisisca in opportune strutture dati le preferenze delle persone e delle aziende
- verifichi se il matching sia perfetto o meno. Sarà valutata anche la complessità dell'algoritmo di verifica
- determini e visualizzi un matching perfetto mediante un algoritmo completo.

Le stringhe sono di al massimo 10 caratteri. Si può supporre corretto il file.

03MNO Algoritmi e Programmazione

Appello del 29/01/2018 - Prova di programmazione (12 punti)

1. (**2 punti**)
Sia dato un vettore *V* di *N* interi. Si scriva una funzione C che visualizzi tutti i sottovettori di dimensione massima formati da celle contigue contenenti dati non nulli.

Esempio: dato il vettore 1 3 4 0 1 0 9 4 2 0, i 2 sottovettori di dimensione massima contenenti dati non nulli sono 1 3 4 e 9 4 2.

2. (**4 punti**)
Sia data una lista di interi ordinati in ordine ascendente. Alla lista si accede mediante puntatore alla testa. Si scriva una funzione C che, senza usare array o liste di appoggio, completi la sequenza di interi, inserendo nella posizione corretta all'interno della lista tutti i numeri mancanti e visualizzi la lista così generata. La chiamata alla funzione può essere alternativamente:

```
1. n = aggiungi(&head);
2. n = aggiungi(head);
```

dove `head` rappresenta la testa della lista. Si realizzi una delle 2 alternative motivando la scelta. La funzione ritorna come valore intero il numero di nodi aggiunti.

Esempio: se la lista in input contiene 4, 7, 10, quella in output deve contenere 4,5,6,7,8, 9,10

3. (6 punti)
Un intervallo aperto (l_i, h_i) è caratterizzato da un estremo inferiore l_i, un estremo superiore h_i e una durata $d_i = h_i - l_i$. Estremi e durata sono interi. Una collezione di intervalli *S* è memorizzata in un vettore `v` di *N* `struct interv` aventi come campi estremo inferiore ed estremo superiore. Si suppongano corretti i dati contenuti nel vettore ($\forall\ i\ l_i \leq h_i$). Due intervalli *i* e *j* sono **incompatibili** se e solo se si intersecano o sovrappongono:

$$(l_i < h_j)\ \&\&\ (l_j < h_i) \Leftrightarrow i \cap j \neq \varnothing$$

Si scrivano la funzione wrapper e una funzione ricorsiva in C in grado di determinare e visualizzare il sottoinsieme *S* di intervalli **compatibili** che massimizza la somma delle durate:

$$\sum_{k \in S} d_k = \text{MAX} \quad \&\& \quad \forall (k_1, k_2) \in S\ k_1 \cap k_2 = \varnothing$$

Il prototipo della funzione wrapper sia:

```
void intervSel(int N, interv *v);
```

Si identifichi, giustificando la scelta, il modello combinatorio sottostante. L'algoritmo deve essere completo (non sono ammesse soluzioni greedy).

Esempio: se S =((1,2), (2,4), (2,5), (3,5), (5,7), (6,8)), uno dei sottoinsiemi di *S* di intervalli **compatibili** che massimizza la somma delle durate è (1,2), (2,5), (6,8) per una somma delle durate pari a 6.

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 giovedì 01/02/2018, alle ore 14:00, mediante caricamento su Portale. Le istruzioni per il caricamento sono pubblicate sul Portale nella sezione Materiale). QUALORA IL CODICE CARICATO CON LA RELAZIONE NON COMPILI CORRETTAMENTE, VERRÀ APPLICATA UNA PENALIZZAZIONE. 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.***