Cognome: **Nome**: **# Matricola**: **Riga**: **Col**:

Algoritmi e Strutture Dati – Parte A – 04/09/2020

Esercizio A1, punti ≥ 8

Trovare i limiti superiori e inferiori più stretti possibili per la seguente famiglia di equazioni di ricorrenza, per valori di a interi positivi maggiori o uguali a 2:

$$T_a(n) = \begin{cases} 2T(\lfloor n/a \rfloor) + n^{\lceil a/2 \rceil} & n \geq 2 \\ 1 & n < 2 \end{cases}$$

Esercizio A2 – Punti ≥ 10

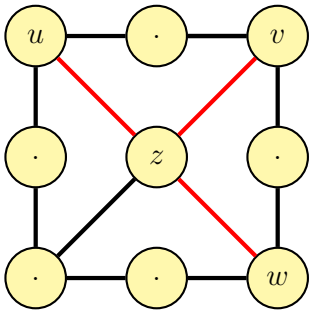
Sia G un grafo non orientato e connesso che rappresenta una rete stradale e siano u, v, w tre nodi (città) in questa rete. Tre amici che abitano in questi tre nodi vogliono incontrarsi in un quarto nodo z, in modo tale che la distanza che devono percorrere (misurata come somma del numero minimo di archi che ognuno di essi deve attraversare per partire da u, v, w e raggiungere z) sia la minore possibile. Nel caso più nodi abbiano distanza minore, andrà bene uno qualunque di essi. Scrivere un algoritmo

NODE meetingPoint(GRAPH G, NODE u, NODE v, NODE w)

che prenda in input G, u, v, w e restituisca il nodo z.

Discutere correttezza e complessità computazionale dell'algoritmo proposto.

Nel grafo seguente, il nodo z è quello al centro; la distanza totale di u, v, w è pari a $1+1+1=3$. Scegliere v dà origine ad una distanza totale pari $2+0+2=4$, da tutti gli altri nodi la distanza è superiore a 4.



Notate che è possibile che la città scelta sia una di quelle di partenza, se questo minimizza il numero di archi da percorrere. Per esempio, se nel grafo precedente gli amici abitassero nelle città u, z, w, la città prescelta sarebbe di nuovo z.

Esercizio A3 – Punti ≥ 12

Scrivere un algoritmo

int longestSingle(**int**[] A, **int** n)

che prenda in input un vettore A di dimensione n, contenente interi compresi fra 1 ed n, e restituisca la lunghezza del più lungo sottovettore contiguo che non contiene valori duplicati.

Ad esempio, nel sottovettore $[1, 2, \mathbf{3, 2, 4,} \mathbf{1}, 3]$ di lunghezza $n = 7$, i sottovettori $[3, 2, 4, 1]$ e $[2, 4, 1, 3]$ hanno lunghezza massima. L'algoritmo dovrà restituire 4.

Discutere correttezza e complessità computazionale dell'algoritmo proposto.

Cognome: **Nome**: **# Matricola**: **Riga**: **Col**:

Algoritmi e Strutture Dati – Parte B - 04/09/2020

Esercizio B1, punti ≥ 8

Siete responsabile di un ristorante che propone menù a prezzo fisso. Avete un insieme di n piatti; il costo del piatto i-esimo è contenuto nel vettore A nella posizione $A[i]$. Il costo totale che volete proporre al pubblico è pari a k. Volete contare quanti menù diversi possono dare origine a quel costo, ottenuto sommando i costi dei singoli piatti (selezionati al massimo una volta per menù)
Scrivere un algoritmo

int countMenu(**int**[ ] A, **int** n, **int** k)

che prenda in input un vettore A contenente n interi positivi e un intero positivo k, e restituisca il numero di modi diversi per cui è possibile ottenere il valore k come somma di un sottoinsieme di elementi di A.

Discutere correttezza e complessità computazionale dell'algoritmo proposto. Specificate cosa succede nel caso più piatti abbiano lo stesso costo.

Ad esempio, se $A = [1, 3, 5, 7, 8, 4]$ e $k = 8$, è possibile ottenere tale valore come $1 + 3 + 4, 1 + 7, 3 + 5, 8$. L'algoritmo dovrà restituire 4.

Esercizio B2, punti ≥ 11

Scrivere un algoritmo

int LCSubstring(ITEM[ ] T, ITEM[ ] U, **int** n, **int** m)

che prenda in input due stringhe T, U di lunghezza n e m e restituisca la lunghezza della più lunga sottostringa comune ad entrambe. Notate che questo problema è diverso da LCS, che è basato su sottosequenze non necessariamente contigue. In questo problema le sottostringhe devono essere contigue.

Per esempio, la più lunga sottostringa comune contenuta nelle stringhe $T =$ "`abcdef`" e $U =$ "`abxcdexf`" è "`cde`", e l'algoritmo dovrà restituire 3.

Discutere correttezza e complessità computazionale dell'algoritmo proposto.

Esercizio B3, punti ≥ 11

Scrivere un algoritmo

int printIncreasing(**int**[ ] A, **int** n)

che prenda in input un vettore A e stampi tutte le sottosequenze crescenti del vettore A.

Ad esempio, $A = [1, 3, 6, 4, 5]$ dà origine a questo output, non necessariamente in quest'ordine:

[1, 3, 4, 5], [3, 4, 5], [1, 4, 5], [4, 5], [1, 3, 5], [3, 5], [1, 5], [5], [1, 3, 4], [3, 4], [1, 4], [4], [1, 3, 6], [3, 6], [1, 6], [6], [1, 3], [3], [1], []

Discutere correttezza e complessità computazionale dell'algoritmo proposto.