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

Algoritmi e Strutture Dati - Parte A - 04/02/2020

Esercizio -1 Iscriversi allo scritto entro la scadenza. In caso di inadempienza, -1 al voto finale.

Esercizio 0 Scrivere correttamente nome, cognome, numero di matricola, riga e colonna su tutti i fogli consegnati. Consegnare foglio A4 e foglio protocollo di bella. In caso di inadempienza, -1 al voto finale.

Esercizio A1 – Punti ≥ 8

Si calcoli la complessità computazionale della seguente procedura, dove buildHeap() è la funzione utilizzata in HeapSort per costruire un vettore max-heap a partire da un vettore non ordinato di interi.

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

    return funr$(A, 1, n)$

int funr(**int**[ ] A, **int** i, **int** j)

    if $i + 1 < j$ **then**
        int $n = (j - i + 1)$
        int[ ] $B =$ **new int**$[1 \ldots n]$
        for $k = i$ **to** j **do**
            $B[k - i + 1] = A[k]$
        buildHeap(B, n)
        int $m = \lfloor (i + j)/2 \rfloor$
        return funr$(A, i, m) +$ funr$(A, m + 1, j) + B[2]$
    else
        return 0

Esercizio A2 – Punti ≥ 10

Scrivere un algoritmo

boolean hasPath(**int**[ ][ ] A, **int** n)

che prenda in input una matrice quadrata di interi positivi di dimensione $n \times n$ e restituisca **true** se esiste un cammino ammissibile dalla cella $(1, 1)$ alla cella (n, n). Un cammino si dice *ammissibile* se non contiene celle ripetute ed è composto da mosse in orizzontale o verticale lunghe quanto il valore contenuto nella cella. In altre parole, dalla cella (i, j) le mosse possibili portano nelle celle seguenti (se esistenti):

$$(i - A[i][j], j)$$

$$(i, j - A[i][j]) \qquad\qquad (i, j + A[i][j])$$

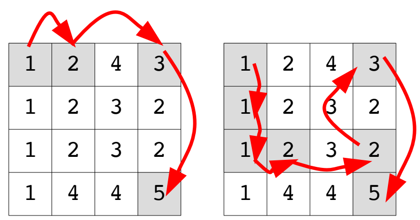
$$(i + A[i][j], j)$$

Discutere informalmente la correttezza dell'algoritmo e calcolare la sua complessità computazionale.

Nell'esempio seguente esistono 5 cammini ammissibili da $(1, 1)$ a $(4, 4)$:

$$C_1 = [(1,1), (1,2), (1,4), (4,4)]$$
$$C_2 = [(1,1), (2,1), (3,1), (3,2), (3,4), (1,4), (4,4)]$$
$$C_3 = [(1,1), (2,1), (3,1), (3,2), (1,2), (1,4), (4,4)]$$
$$C_4 = [(1,1), (2,1), (2,2), (2,4), (4,4)]$$
$$C_5 = [(1,1), (1,2), (3,2), (3,4), (1,4), (4,4)]$$

dei quali i primi due sono rappresentati tramite frecce rosse. Per questo input, l'algoritmo deve restituire **true**.



Esercizio A3 – Punti ≥ 12

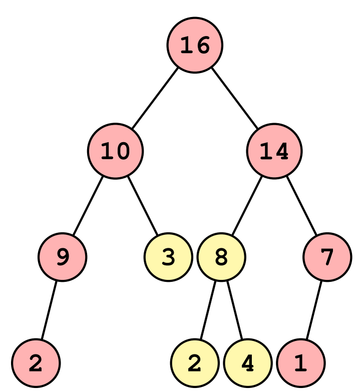
In un albero binario, un nodo è *angolare* se è il primo o l'ultimo del suo livello. Scrivere un algoritmo

int sumAngular(TREE T)

che prenda in input un albero binario che contenga valori numerici nel campo $t.value$ di ogni nodo t e restituisca la somma dei valori contenuti nei nodi angolari.

Discutere informalmente la correttezza dell'algoritmo e calcolare la sua complessità computazionale.

Nell'esempio seguente, i nodi rosa sono nodi angolari, i nodi gialli no. L'algoritmo dovrà quindi restituire: $16 + 10 + 14 + 9 + 7 + 2 + 1 = 59$.



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

Algoritmi e Strutture Dati - Parte B - 04/02/2020

Esercizio -1 Iscriversi allo scritto entro la scadenza. In caso di inadempienza, -1 al voto finale.

Esercizio 0 Scrivere correttamente nome, cognome, numero di matricola, riga e colonna su tutti i fogli consegnati. Consegnare foglio A4 e foglio protocollo di bella. In caso di inadempienza, -1 al voto finale.

Esercizio B1 – Punti ≥ 8

Scrivere un algoritmo

$$\textbf{int}\ \mathsf{countPaths}(\textbf{int}[\,][\,]\ A,\ \textbf{int}\ n)$$

che prenda in input una matrice quadrata di interi positivi di dimensione $n \times n$ e restituisca il numero di cammini ammissibili dalla cella $(1, 1)$ alla cella (n, n). Un cammino si dice *ammissibile* se non contiene celle ripetute ed è composto da mosse in orizzontale o verticale lunghe quanto il valore contenuto nella cella. In altre parole, dalla cella (i, j) le mosse possibili portano nelle celle seguenti (se possibile).

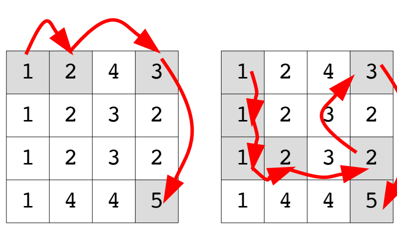
$$(i - A[i][j], j)$$
$$(i, j - A[i][j]) \qquad\qquad (i, j + A[i][j])$$
$$(i + A[i][j], j)$$

Discutere informalmente la correttezza dell'algoritmo e calcolare la sua complessità computazionale.

Nell'esempio seguente esistono 5 cammini ammissibili da $(1, 1)$ a $(4, 4)$:

$$C_1 = [(1, 1), (1, 2), (1, 4), (4, 4)]$$
$$C_2 = [(1, 1), (2, 1), (3, 1), (3, 2), (3, 4), (1, 4), (4, 4)]$$
$$C_3 = [(1, 1), (2, 1), (3, 1), (3, 2), (1, 2), (1, 4), (4, 4)]$$
$$C_4 = [(1, 1), (2, 1), (2, 2), (2, 4), (4, 4)]$$
$$C_5 = [(1, 1), (1, 2), (3, 2), (3, 4), (1, 4), (4, 4)]$$

dei quali i primi due sono rappresentati tramite frecce rosse. Per questo input, l'algoritmo deve restituire 5.



Esercizio B2 – Punti ≥ 10

Lo zero-sbilanciamento di un vettore contenente valori $0, 1$ è dato dal numero di valori zero meno il numero di valori uno contenuti in esso. Si scriva un algoritmo

$$\textbf{int}\ \mathsf{zeroUnbalance}(\textbf{int}[\,]\ A,\ \textbf{int}\ n)$$

che restituisca lo zero-sbilanciamento massimale del vettore A di lunghezza n, ovvero il più grande zero-sbilanciamento fra tutti i sottovettori contigui di A.

Discutere informalmente la correttezza dell'algoritmo e calcolare la sua complessità computazionale.

Ad esempio, il vettore `00001000` contiene 7 bit zero e 1 bit 1, quindi il suo zero-sbilanciamento è pari a 7-1=6.
Nel vettore $\texttt{11}\underline{\texttt{00001000}}\texttt{1}$, il sottovettore contiguo con massimo zero-sbilanciamento è `00001000`, sottolineato. Il vostro algoritmo dovrà quindi rispondere 6.

Esercizio B3 – Punti ≥ 12

Scrivere un algoritmo:

$$\textbf{int}\ \mathsf{largestSquare}(\textbf{int}[\,][\,]\ A,\ \textbf{int}\ n)$$

che prenda in input una matrice quadrata A di dimensione $n \times n$ contenente valori $0, 1$ e restituisca la dimensione della più grande sottomatrice quadrata che contiene solo valori 1 sui bordi.

Discutere informalmente la correttezza dell'algoritmo e calcolare la sua complessità computazionale.

Nella matrice 8×8 seguente, è possibile vedere una sottomatrice quadrata con bordi di valore 1 di dimensione 5×5, evidenziata in grigio; è possibile vedere che all'interno di essa esiste anche una sottomatrice quadrata con bordi di valore 1 di dimensione 4×4. Non è possibile vedere sottomatrici di dimensioni 6×6 o superiori. Per questo input, l'algoritmo deve restituire 5.

1	0	1	1	1	1	1	0
1	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1
1	1	1	0	0	1	1	1
0	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0
0	1	0	0	1	0	1	1