Cognome.....

Nome.....

Matricola.....

Algoritmi e Strutture Dati

Prova scritta del 4 febbraio 2019

TEMPO DISPONIBILE: 2 ore

Le risposte agli esercizi 1 e 2 devono essere scritte negli appositi riquadri su questo foglio (risposte scritte su altri fogli non saranno considerate). La soluzione dell'esercizio 3 va scritta su uno dei fogli di protocollo forniti. Le brutte copie NON devono essere consegnate. Ricordatevi di scrivere cognome e nome su TUTTI i fogli (inclusi quelli di brutta).

1. Considerate l'albero AVL che si ottiene inserendo uno dopo l'altro, nell'ordine indicato, i seguenti numeri a partire da un albero AVL inizialmente vuoto:

`20 16 18 12 17 14 22`

(a) Disegnate l'albero ottenuto

(b) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in ampiezza

(c) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in profondità in ordine anticipato

(d) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in profondità in ordine simmetrico

(e) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in profondità in ordine posticipato

(f) L'albero ottenuto al punto (a) è un albero perfettamente bilanciato? ☐

(g) Se al punto (f) avete risposto SI scrivete la definizione di albero perfettamente bilanciato.
Se avete risposto NO indicate il contenuto di tutti i nodi che non rispettano la condizione di bilanciamento relativa agli alberi perfettamente bilanciati.

(h) Disegnate l'albero 2-3 che si ottiene inserendo il un albero 2-3 inizialmente vuoto la sequenza di numeri precedente, nell'ordine indicato.

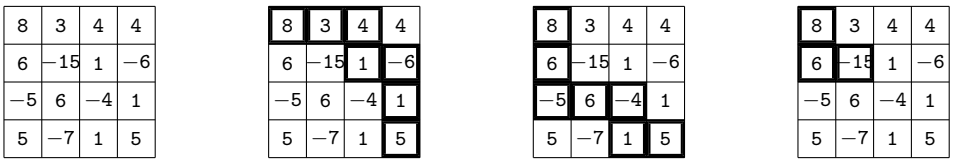
2. La seguente sequenza di numeri, memorizzata in un array, deve essere ordinata in modo crescente:
`18 16 29 12 17 14 22 19`

(a) Supponete di ordinare la sequenza mediante l'algoritmo `quickSort`, scegliendo come perno il primo elemento. Indicate il contenuto dell'array dopo avere effettuato la partizione, prima delle chiamate ricorsive di `quickSort`.

(b) Supponete di ordinare la sequenza mediante l'algoritmo `heapSort`. Indicate il contenuto dell'array dopo averlo trasformato in uno heap.

(c) Supponete di ordinare la sequenza mediante l'algoritmo `bubbleSort`. Indicate il contenuto dell'array dopo la terza iterazione del ciclo principale dell'algoritmo.

3. Un robot si muove su una griglia quadrata $n \times n$. In alcune caselle della griglia si trovano delle monete, che il robot raccoglie quando vi entra. In altre caselle è richiesto il pagamento di un pedaggio: se arrivando in una di queste caselle il robot non dispone del numero di monete richiesto resta bloccato. Ad ogni passo il robot può muoversi dalla casella in cui si trova nella casella immediatamente a destra sulla stessa riga della griglia (salvo nel caso in cui si trovi già nella colonna più a destra), o immediatamente sotto sulla stessa colonna (salvo nel caso in cui si trovi già nella riga più in basso). Obiettivo del robot è riuscire a *collezionare il maggior numero di monete partendo dalla casella in alto a sinistra della griglia, fino a raggiungere la casella in basso a destra*, muovendosi secondo quanto indicato sopra. *Esempio.* La figura qui sotto a sinistra rappresenta una griglia con l'indicazione delle monete presenti in ciascuna casella (numeri positivi) o del pedaggio da pagare (numeri negativi). Nelle altre figure sono evidenziati tre possibili cammini (nel primo alla fine il robot avrà 16 monete, nel secondo 17, nell'ultimo il robot resta bloccato).



Cosa si richiede. È data matrice M di dimensione $n \times n$, contenente numeri interi. Per ogni casella (i, j) $(i, j = 1, \ldots, n)$, un valore $m_{i,j} > 0$ indica il numero di monete disponibili nella casella, un valore $m_{i,j} < 0$ indica che nella casella deve essere pagato un pedaggio di $-m_{i,j}$ monete. Per le rimanenti caselle $m_{i,j} = 0$. Indichiamo con $c_{i,j}$ il massimo numero di monete che il robot può avere seguendo un percorso che inizia nella casella $(1, 1)$ (cioè quella in alto a sinistra) e termina nella casella (i, j). Nel caso il robot non sia in grado di raggiungere la casella (i, j) o resti bloccato in essa, assegnamo a $c_{i,j}$ il valore $-\infty$.

Due esempi di matrice M e corrispondente matrice C:

$$M = \begin{bmatrix} 8 & 3 & 4 & 4 \\ 6 & -15 & 1 & -6 \\ -5 & 6 & -4 & 1 \\ 5 & -7 & 1 & 5 \end{bmatrix}, C = \begin{bmatrix} 8 & 11 & 15 & 19 \\ 14 & -\infty & 16 & 13 \\ 9 & 15 & 12 & 14 \\ 14 & 8 & 13 & 19 \end{bmatrix}. \quad M = \begin{bmatrix} 0 & 3 & 5 & 0 \\ 4 & -10 & 4 & -6 \\ -5 & -6 & -4 & 1 \\ 4 & -7 & 1 & -3 \end{bmatrix}, C = \begin{bmatrix} 0 & 3 & 8 & 8 \\ 4 & -\infty & 12 & 6 \\ -\infty & -\infty & 8 & 9 \\ -\infty & -\infty & 9 & 6 \end{bmatrix}.$$

Risolvete i seguenti punti ***nell'ordine indicato.***

(a) Tenendo conto delle regole con cui il robot si può muovere, scrivete una o più formule che permettano di ricavare il generico valore $c_{i,j}$ $(i, j = 1, \ldots, n)$ sulla base di opportuni $c_{i',j'}$ con $i' < i$ o $j' < j$ e di $m_{i,j}$ o solo di quest'ultimo.

(b) Descrivete *sinteticamente a parole* e poi *ad alto livello* in pseudocodice, un algoritmo basato sulla tecnica di *programmazione dinamica* che ricevendo in ingresso la matrice M calcoli la matrice C (utilizzate quanto scritto al punto (a)) e stabilisca se esista almeno un percorso che permetta al robot partendo dalla casella $(1, 1)$ di raggiungere la casella (n, n) e, in caso positivo, indichi il numero massimo di monete che il robot può avere quando raggiunge la casella (n, n).

(c) Fornite una stima del tempo totale utilizzato dall'algoritmo in funzione di n.

(d) Descrivete sinteticamente a parole come ricavare dalla matrice C il cammino che permette al robot di ottenere il massimo numero di monete.

(e) Fornite una stima del tempo totale utilizzato dalla parte di algoritmo descritta al punto (d) in funzione di n.

Note. Potete effettuare operazioni direttamente con il valore $-\infty$. Il risultato dell'addizione e della sottrazione tra $-\infty$ e un valore finito è $-\infty$. Il minimo e il massimo tra $-\infty$ e un valore x sono rispettivamente $-\infty$ e x. Non cambiate i nomi stabiliti nel testo dell'esercizio per le matrici. Le risposte devono essere adeguatamente giustificate.

Cognome.....

Nome.....

Matricola.....

Algoritmi e Strutture Dati

Prova scritta del 4 febbraio 2019

TEMPO DISPONIBILE: 2 ore

Le risposte agli esercizi 1 e 2 devono essere scritte negli appositi riquadri su questo foglio (risposte scritte su altri fogli non saranno considerate). La soluzione dell'esercizio 3 va scritta su uno dei fogli di protocollo forniti. Le brutte copie NON devono essere consegnate. Ricordatevi di scrivere cognome e nome su TUTTI i fogli (inclusi quelli di brutta).

1. Considerate l'albero AVL che si ottiene inserendo uno dopo l'altro, nell'ordine indicato, i seguenti numeri a partire da un albero AVL inizialmente vuoto:

`21 17 19 13 18 15 23`

(a) Disegnate l'albero ottenuto

(b) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in ampiezza

(c) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in profondità in ordine anticipato

(d) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in profondità in ordine simmetrico

(e) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in profondità in ordine posticipato

(f) L'albero ottenuto al punto (a) è un albero perfettamente bilanciato? ☐

(g) Se al punto (f) avete risposto SI scrivete la definizione di albero perfettamente bilanciato.
Se avete risposto NO indicate il contenuto di tutti i nodi che non rispettano la condizione di bilanciamento relativa agli alberi perfettamente bilanciati.

(h) Disegnate l'albero 2-3 che si ottiene inserendo il un albero 2-3 inizialmente vuoto la sequenza di numeri precedente, nell'ordine indicato.

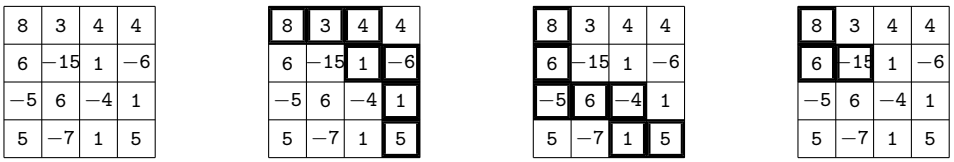
2. La seguente sequenza di numeri, memorizzata in un array, deve essere ordinata in modo crescente:
`19 17 30 13 18 15 23 20`

(a) Supponete di ordinare la sequenza mediante l'algoritmo `quickSort`, scegliendo come perno il primo elemento. Indicate il contenuto dell'array dopo avere effettuato la partizione, prima delle chiamate ricorsive di `quickSort`.

(b) Supponete di ordinare la sequenza mediante l'algoritmo `heapSort`. Indicate il contenuto dell'array dopo averlo trasformato in uno heap.

(c) Supponete di ordinare la sequenza mediante l'algoritmo `bubbleSort`. Indicate il contenuto dell'array dopo la terza iterazione del ciclo principale dell'algoritmo.

3. Un robot si muove su una griglia quadrata $n \times n$. In alcune caselle della griglia si trovano delle monete, che il robot raccoglie quando vi entra. In altre caselle è richiesto il pagamento di un pedaggio: se arrivando in una di queste caselle il robot non dispone del numero di monete richiesto resta bloccato. Ad ogni passo il robot può muoversi dalla casella in cui si trova nella casella immediatamente a destra sulla stessa riga della griglia (salvo nel caso in cui si trovi già nella colonna più a destra), o immediatamente sotto sulla stessa colonna (salvo nel caso in cui si trovi già nella riga più in basso). Obiettivo del robot è riuscire a *collezionare il maggior numero di monete partendo dalla casella in alto a sinistra della griglia, fino a raggiungere la casella in basso a destra*, muovendosi secondo quanto indicato sopra. *Esempio.* La figura qui sotto a sinistra rappresenta una griglia con l'indicazione delle monete presenti in ciascuna casella (numeri positivi) o del pedaggio da pagare (numeri negativi). Nelle altre figure sono evidenziati tre possibili cammini (nel primo alla fine il robot avrà 16 monete, nel secondo 17, nell'ultimo il robot resta bloccato).



Cosa si richiede. È data matrice M di dimensione $n \times n$, contenente numeri interi. Per ogni casella (i, j) $(i, j = 1, \ldots, n)$, un valore $m_{i,j} > 0$ indica il numero di monete disponibili nella casella, un valore $m_{i,j} < 0$ indica che nella casella deve essere pagato un pedaggio di $-m_{i,j}$ monete. Per le rimanenti caselle $m_{i,j} = 0$. Indichiamo con $c_{i,j}$ il massimo numero di monete che il robot può avere seguendo un percorso che inizia nella casella $(1, 1)$ (cioè quella in alto a sinistra) e termina nella casella (i, j). Nel caso il robot non sia in grado di raggiungere la casella (i, j) o resti bloccato in essa, assegnamo a $c_{i,j}$ il valore $-\infty$.

Due esempi di matrice M e corrispondente matrice C:

$$M = \begin{bmatrix} 8 & 3 & 4 & 4 \\ 6 & -15 & 1 & -6 \\ -5 & 6 & -4 & 1 \\ 5 & -7 & 1 & 5 \end{bmatrix}, C = \begin{bmatrix} 8 & 11 & 15 & 19 \\ 14 & -\infty & 16 & 13 \\ 9 & 15 & 12 & 14 \\ 14 & 8 & 13 & 19 \end{bmatrix}. \quad M = \begin{bmatrix} 0 & 3 & 5 & 0 \\ 4 & -10 & 4 & -6 \\ -5 & -6 & -4 & 1 \\ 4 & -7 & 1 & -3 \end{bmatrix}, C = \begin{bmatrix} 0 & 3 & 8 & 8 \\ 4 & -\infty & 12 & 6 \\ -\infty & -\infty & 8 & 9 \\ -\infty & -\infty & 9 & 6 \end{bmatrix}.$$

Risolvete i seguenti punti ***nell'ordine indicato.***

(a) Tenendo conto delle regole con cui il robot si può muovere, scrivete una o più formule che permettano di ricavare il generico valore $c_{i,j}$ $(i, j = 1, \ldots, n)$ sulla base di opportuni $c_{i',j'}$ con $i' < i$ o $j' < j$ e di $m_{i,j}$ o solo di quest'ultimo.

(b) Descrivete *sinteticamente a parole* e poi *ad alto livello* in pseudocodice, un algoritmo basato sulla tecnica di *programmazione dinamica* che ricevendo in ingresso la matrice M calcoli la matrice C (utilizzate quanto scritto al punto (a)) e stabilisca se esista almeno un percorso che permetta al robot partendo dalla casella $(1, 1)$ di raggiungere la casella (n, n) e, in caso positivo, indichi il numero massimo di monete che il robot può avere quando raggiunge la casella (n, n).

(c) Fornite una stima del tempo totale utilizzato dall'algoritmo in funzione di n.

(d) Descrivete sinteticamente a parole come ricavare dalla matrice C il cammino che permette al robot di ottenere il massimo numero di monete.

(e) Fornite una stima del tempo totale utilizzato dalla parte di algoritmo descritta al punto (d) in funzione di n.

Note. Potete effettuare operazioni direttamente con il valore $-\infty$. Il risultato dell'addizione e della sottrazione tra $-\infty$ e un valore finito è $-\infty$. Il minimo e il massimo tra $-\infty$ e un valore x sono rispettivamente $-\infty$ e x. Non cambiate i nomi stabiliti nel testo dell'esercizio per le matrici. Le risposte devono essere adeguatamente giustificate.

Cognome.....

Nome.....

Matricola.....

Algoritmi e Strutture Dati

Prova scritta del 4 febbraio 2019

TEMPO DISPONIBILE: 2 ore

Le risposte agli esercizi 1 e 2 devono essere scritte negli appositi riquadri su questo foglio (risposte scritte su altri fogli non saranno considerate). La soluzione dell'esercizio 3 va scritta su uno dei fogli di protocollo forniti. Le brutte copie NON devono essere consegnate. Ricordatevi di scrivere cognome e nome su TUTTI i fogli (inclusi quelli di brutta).

1. Considerate l'albero AVL che si ottiene inserendo uno dopo l'altro, nell'ordine indicato, i seguenti numeri a partire da un albero AVL inizialmente vuoto:

`22 18 20 14 19 16 24`

(a) Disegnate l'albero ottenuto

(b) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in ampiezza

(c) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in profondità in ordine anticipato

(d) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in profondità in ordine simmetrico

(e) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in profondità in ordine posticipato

(f) L'albero ottenuto al punto (a) è un albero perfettamente bilanciato? ☐

(g) Se al punto (f) avete risposto SI scrivete la definizione di albero perfettamente bilanciato.
Se avete risposto NO indicate il contenuto di tutti i nodi che non rispettano la condizione di bilanciamento relativa agli alberi perfettamente bilanciati.

(h) Disegnate l'albero 2-3 che si ottiene inserendo il un albero 2-3 inizialmente vuoto la sequenza di numeri precedente, nell'ordine indicato.

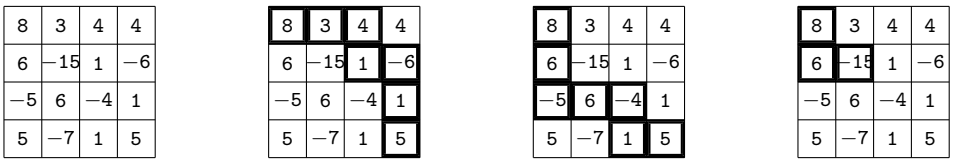
2. La seguente sequenza di numeri, memorizzata in un array, deve essere ordinata in modo crescente:
`20 18 31 14 19 16 24 21`

(a) Supponete di ordinare la sequenza mediante l'algoritmo `quickSort`, scegliendo come perno il primo elemento. Indicate il contenuto dell'array dopo avere effettuato la partizione, prima delle chiamate ricorsive di `quickSort`.

(b) Supponete di ordinare la sequenza mediante l'algoritmo `heapSort`. Indicate il contenuto dell'array dopo averlo trasformato in uno heap.

(c) Supponete di ordinare la sequenza mediante l'algoritmo `bubbleSort`. Indicate il contenuto dell'array dopo la terza iterazione del ciclo principale dell'algoritmo.

3. Un robot si muove su una griglia quadrata $n \times n$. In alcune caselle della griglia si trovano delle monete, che il robot raccoglie quando vi entra. In altre caselle è richiesto il pagamento di un pedaggio: se arrivando in una di queste caselle il robot non dispone del numero di monete richiesto resta bloccato. Ad ogni passo il robot può muoversi dalla casella in cui si trova nella casella immediatamente a destra sulla stessa riga della griglia (salvo nel caso in cui si trovi già nella colonna più a destra), o immediatamente sotto sulla stessa colonna (salvo nel caso in cui si trovi già nella riga più in basso). Obiettivo del robot è riuscire a *collezionare il maggior numero di monete partendo dalla casella in alto a sinistra della griglia, fino a raggiungere la casella in basso a destra*, muovendosi secondo quanto indicato sopra. *Esempio.* La figura qui sotto a sinistra rappresenta una griglia con l'indicazione delle monete presenti in ciascuna casella (numeri positivi) o del pedaggio da pagare (numeri negativi). Nelle altre figure sono evidenziati tre possibili cammini (nel primo alla fine il robot avrà 16 monete, nel secondo 17, nell'ultimo il robot resta bloccato).



Cosa si richiede. È data matrice M di dimensione $n \times n$, contenente numeri interi. Per ogni casella (i, j) $(i, j = 1, \ldots, n)$, un valore $m_{i,j} > 0$ indica il numero di monete disponibili nella casella, un valore $m_{i,j} < 0$ indica che nella casella deve essere pagato un pedaggio di $-m_{i,j}$ monete. Per le rimanenti caselle $m_{i,j} = 0$. Indichiamo con $c_{i,j}$ il massimo numero di monete che il robot può avere seguendo un percorso che inizia nella casella $(1, 1)$ (cioè quella in alto a sinistra) e termina nella casella (i, j). Nel caso il robot non sia in grado di raggiungere la casella (i, j) o resti bloccato in essa, assegnamo a $c_{i,j}$ il valore $-\infty$.

Due esempi di matrice M e corrispondente matrice C:

$$M = \begin{bmatrix} 8 & 3 & 4 & 4 \\ 6 & -15 & 1 & -6 \\ -5 & 6 & -4 & 1 \\ 5 & -7 & 1 & 5 \end{bmatrix}, C = \begin{bmatrix} 8 & 11 & 15 & 19 \\ 14 & -\infty & 16 & 13 \\ 9 & 15 & 12 & 14 \\ 14 & 8 & 13 & 19 \end{bmatrix}. \quad M = \begin{bmatrix} 0 & 3 & 5 & 0 \\ 4 & -10 & 4 & -6 \\ -5 & -6 & -4 & 1 \\ 4 & -7 & 1 & -3 \end{bmatrix}, C = \begin{bmatrix} 0 & 3 & 8 & 8 \\ 4 & -\infty & 12 & 6 \\ -\infty & -\infty & 8 & 9 \\ -\infty & -\infty & 9 & 6 \end{bmatrix}.$$

Risolvete i seguenti punti ***nell'ordine indicato.***

(a) Tenendo conto delle regole con cui il robot si può muovere, scrivete una o più formule che permettano di ricavare il generico valore $c_{i,j}$ $(i, j = 1, \ldots, n)$ sulla base di opportuni $c_{i',j'}$ con $i' < i$ o $j' < j$ e di $m_{i,j}$ o solo di quest'ultimo.

(b) Descrivete *sinteticamente a parole* e poi *ad alto livello* in pseudocodice, un algoritmo basato sulla tecnica di *programmazione dinamica* che ricevendo in ingresso la matrice M calcoli la matrice C (utilizzate quanto scritto al punto (a)) e stabilisca se esista almeno un percorso che permetta al robot partendo dalla casella $(1, 1)$ di raggiungere la casella (n, n) e, in caso positivo, indichi il numero massimo di monete che il robot può avere quando raggiunge la casella (n, n).

(c) Fornite una stima del tempo totale utilizzato dall'algoritmo in funzione di n.

(d) Descrivete sinteticamente a parole come ricavare dalla matrice C il cammino che permette al robot di ottenere il massimo numero di monete.

(e) Fornite una stima del tempo totale utilizzato dalla parte di algoritmo descritta al punto (d) in funzione di n.

Note. Potete effettuare operazioni direttamente con il valore $-\infty$. Il risultato dell'addizione e della sottrazione tra $-\infty$ e un valore finito è $-\infty$. Il minimo e il massimo tra $-\infty$ e un valore x sono rispettivamente $-\infty$ e x. Non cambiate i nomi stabiliti nel testo dell'esercizio per le matrici. Le risposte devono essere adeguatamente giustificate.

Cognome.....

Nome.....

Matricola.....

Algoritmi e Strutture Dati

Prova scritta del 4 febbraio 2019

TEMPO DISPONIBILE: 2 ore

Le risposte agli esercizi 1 e 2 devono essere scritte negli appositi riquadri su questo foglio (risposte scritte su altri fogli non saranno considerate). La soluzione dell'esercizio 3 va scritta su uno dei fogli di protocollo forniti. Le brutte copie NON devono essere consegnate. Ricordatevi di scrivere cognome e nome su TUTTI i fogli (inclusi quelli di brutta).

1. Considerate l'albero AVL che si ottiene inserendo uno dopo l'altro, nell'ordine indicato, i seguenti numeri a partire da un albero AVL inizialmente vuoto:

`23 19 21 15 20 17 25`

(a) Disegnate l'albero ottenuto

(b) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in ampiezza

(c) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in profondità in ordine anticipato

(d) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in profondità in ordine simmetrico

(e) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in profondità in ordine posticipato

(f) L'albero ottenuto al punto (a) è un albero perfettamente bilanciato? ☐

(g) Se al punto (f) avete risposto SI scrivete la definizione di albero perfettamente bilanciato.
Se avete risposto NO indicate il contenuto di tutti i nodi che non rispettano la condizione di bilanciamento relativa agli alberi perfettamente bilanciati.

(h) Disegnate l'albero 2-3 che si ottiene inserendo il un albero 2-3 inizialmente vuoto la sequenza di numeri precedente, nell'ordine indicato.

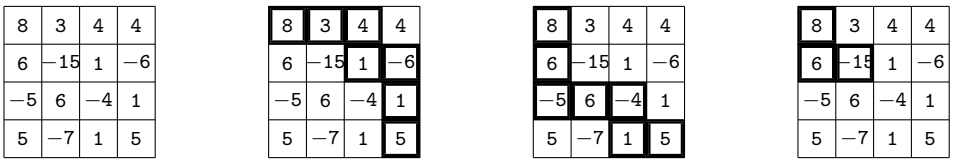
2. La seguente sequenza di numeri, memorizzata in un array, deve essere ordinata in modo crescente:
`21 19 32 15 20 17 25 22`

(a) Supponete di ordinare la sequenza mediante l'algoritmo `quickSort`, scegliendo come perno il primo elemento. Indicate il contenuto dell'array dopo avere effettuato la partizione, prima delle chiamate ricorsive di `quickSort`.

(b) Supponete di ordinare la sequenza mediante l'algoritmo `heapSort`. Indicate il contenuto dell'array dopo averlo trasformato in uno heap.

(c) Supponete di ordinare la sequenza mediante l'algoritmo `bubbleSort`. Indicate il contenuto dell'array dopo la terza iterazione del ciclo principale dell'algoritmo.

3. Un robot si muove su una griglia quadrata $n \times n$. In alcune caselle della griglia si trovano delle monete, che il robot raccoglie quando vi entra. In altre caselle è richiesto il pagamento di un pedaggio: se arrivando in una di queste caselle il robot non dispone del numero di monete richiesto resta bloccato. Ad ogni passo il robot può muoversi dalla casella in cui si trova nella casella immediatamente a destra sulla stessa riga della griglia (salvo nel caso in cui si trovi già nella colonna più a destra), o immediatamente sotto sulla stessa colonna (salvo nel caso in cui si trovi già nella riga più in basso). Obiettivo del robot è riuscire a *collezionare il maggior numero di monete partendo dalla casella in alto a sinistra della griglia, fino a raggiungere la casella in basso a destra*, muovendosi secondo quanto indicato sopra. *Esempio.* La figura qui sotto a sinistra rappresenta una griglia con l'indicazione delle monete presenti in ciascuna casella (numeri positivi) o del pedaggio da pagare (numeri negativi). Nelle altre figure sono evidenziati tre possibili cammini (nel primo alla fine il robot avrà 16 monete, nel secondo 17, nell'ultimo il robot resta bloccato).



Cosa si richiede. È data matrice M di dimensione $n \times n$, contenente numeri interi. Per ogni casella (i, j) $(i, j = 1, \ldots, n)$, un valore $m_{i,j} > 0$ indica il numero di monete disponibili nella casella, un valore $m_{i,j} < 0$ indica che nella casella deve essere pagato un pedaggio di $-m_{i,j}$ monete. Per le rimanenti caselle $m_{i,j} = 0$. Indichiamo con $c_{i,j}$ il massimo numero di monete che il robot può avere seguendo un percorso che inizia nella casella $(1, 1)$ (cioè quella in alto a sinistra) e termina nella casella (i, j). Nel caso il robot non sia in grado di raggiungere la casella (i, j) o resti bloccato in essa, assegnamo a $c_{i,j}$ il valore $-\infty$.

Due esempi di matrice M e corrispondente matrice C:

$$M = \begin{bmatrix} 8 & 3 & 4 & 4 \\ 6 & -15 & 1 & -6 \\ -5 & 6 & -4 & 1 \\ 5 & -7 & 1 & 5 \end{bmatrix}, C = \begin{bmatrix} 8 & 11 & 15 & 19 \\ 14 & -\infty & 16 & 13 \\ 9 & 15 & 12 & 14 \\ 14 & 8 & 13 & 19 \end{bmatrix}. \quad M = \begin{bmatrix} 0 & 3 & 5 & 0 \\ 4 & -10 & 4 & -6 \\ -5 & -6 & -4 & 1 \\ 4 & -7 & 1 & -3 \end{bmatrix}, C = \begin{bmatrix} 0 & 3 & 8 & 8 \\ 4 & -\infty & 12 & 6 \\ -\infty & -\infty & 8 & 9 \\ -\infty & -\infty & 9 & 6 \end{bmatrix}.$$

Risolvete i seguenti punti ***nell'ordine indicato.***

(a) Tenendo conto delle regole con cui il robot si può muovere, scrivete una o più formule che permettano di ricavare il generico valore $c_{i,j}$ $(i, j = 1, \ldots, n)$ sulla base di opportuni $c_{i',j'}$ con $i' < i$ o $j' < j$ e di $m_{i,j}$ o solo di quest'ultimo.

(b) Descrivete *sinteticamente a parole* e poi *ad alto livello* in pseudocodice, un algoritmo basato sulla tecnica di *programmazione dinamica* che ricevendo in ingresso la matrice M calcoli la matrice C (utilizzate quanto scritto al punto (a)) e stabilisca se esista almeno un percorso che permetta al robot partendo dalla casella $(1, 1)$ di raggiungere la casella (n, n) e, in caso positivo, indichi il numero massimo di monete che il robot può avere quando raggiunge la casella (n, n).

(c) Fornite una stima del tempo totale utilizzato dall'algoritmo in funzione di n.

(d) Descrivete sinteticamente a parole come ricavare dalla matrice C il cammino che permette al robot di ottenere il massimo numero di monete.

(e) Fornite una stima del tempo totale utilizzato dalla parte di algoritmo descritta al punto (d) in funzione di n.

Note. Potete effettuare operazioni direttamente con il valore $-\infty$. Il risultato dell'addizione e della sottrazione tra $-\infty$ e un valore finito è $-\infty$. Il minimo e il massimo tra $-\infty$ e un valore x sono rispettivamente $-\infty$ e x. Non cambiate i nomi stabiliti nel testo dell'esercizio per le matrici. Le risposte devono essere adeguatamente giustificate.