Cognome.....

Nome.....

Matricola.....

# Algoritmi e Strutture Dati

Prova scritta del 2 febbraio 2018

**TEMPO DISPONIBILE: 2 ore**

*Le risposte agli esercizi 1, 2, 3 devono essere scritte negli appositi riquadri su questo foglio (risposte scritte su altri fogli non saranno considerate). La soluzione dell'esercizio 4 va scritta su uno dei fogli di protocollo forniti. Le brutte copie NON devono essere consegnate.* *Ricordatevi di scrivere cognome e nome su tutto ciò che consegnate.*

**1.** Considerate funzione $f : \{\texttt{a}, \texttt{b}, \ldots, \texttt{z}\} \to \{0, 1, \ldots, 15\}$ definita come segue:

| $k$ | $f(k)$ |
|---|---|
| a | 0 |
| b | 1 |
| c | 2 |
| d | 3 |
| e | 4 |
| f | 5 |
| g | 6 |
| h | 6 |
| i | 7 |

| $k$ | $f(k)$ |
|---|---|
| j | 7 |
| k | 7 |
| l | 8 |
| m | 9 |
| n | 10 |
| o | 10 |
| p | 11 |
| q | 12 |
| r | 12 |

| $k$ | $f(k)$ |
|---|---|
| s | 13 |
| t | 13 |
| u | 14 |
| v | 14 |
| w | 15 |
| x | 15 |
| y | 15 |
| z | 15 |

Sia $h$ la funzione che trasforma ogni parola $k$ sull'alfabeto $\{\texttt{a}, \texttt{b}, \ldots, \texttt{z}\}$ nell'intero che si ottiene applicando $f$ al primo carattere di $k$ e $g$ la funzione che trasforma ogni parola $k$ nel *più piccolo numero primo* maggiore o uguale al valore che si ottiene applicando $f$ *all'ultimo* carattere di $k$.

*Esempi:* $h(\texttt{topo}) = 13$, $g(\texttt{topo}) = 11$, $h(\texttt{rinoceronte}) = 12$, $g(\texttt{rinoceronte}) = 5$.

Inserite nella tabella hash a destra, inizialmente vuota, le seguenti parole, nell'ordine indicato:

```
cinghiale criceto cigno calabrone fagiano panda
piccione ape
```

Come funzione hash utilizzate $h$. Per la gestione delle collisioni utilizzate l'hashing doppio mediante la funzione

$$c(k, i) = (\, h(k) + i \cdot g(k) \,) \bmod 16$$

| | |
|---|---|
| 0 | |
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | |
| 10 | |
| 11 | |
| 12 | |
| 13 | |
| 14 | |
| 15 | |

**2.** Considerate un albero AVL e un albero 2-3 ottenuti inserendo uno dopo l'altro, nell'ordine indicato, i seguenti numeri a partire da alberi inizialmente vuoti: `13 18 20 15 16 10 12`

| | |
|---|---|
| (a) Disegnate l'albero AVL. | (b) Disegnate l'albero 2-3. |
| (c) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in *ampiezza* dell'albero AVL. | (d) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in *ordine posticipato* dell'albero AVL. |

**3.** Considerate la seguente sequenza di numeri memorizzata in un array che deve essere ordinata in modo crescente:
`14 13 18 20 15 16 10 12`

(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`.

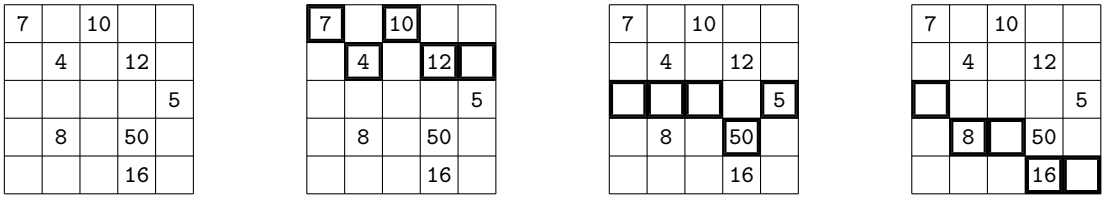
(b) Supponete di ordinare la sequenza mediante l'algoritmo `bubbleSort`. Indicate il contenuto dell'array dopo la prima iterazione del ciclo principale dell'algoritmo.

**4.** È data una scacchiera $n \times n$ su cui sono collocati alcuni pezzi. Ad ogni pezzo è associato un valore intero positivo. Un giocatore si muove sulla scacchiera partendo da una qualunque casella della colonna più a sinistra, fino a raggiungere una casella della colonna più a destra, utilizzando le seguenti regole:

- da una casella il giocatore si può muovere nella casella immediatamente a destra sulla stessa riga, se quest'ultima casella è vuota;
- da una casella il giocatore si può muovere nella casella immediatamente a destra sulla riga immediatamente superiore (se esiste), se quest'ultima casella contiene un pezzo;
- da una casella il giocatore si può muovere nella casella immediatamente a destra sulla riga immediatamente inferiore (se esiste), se quest'ultima casella contiene un pezzo.

Al giocatore viene attribuito un punteggio pari alla somma dei valori dei pezzi incontrati nel cammino percorso all'interno della scacchiera. In altre parole, partendo da zero, ogni volta che il giocatore raggiunge una casella contenente un pezzo, al punteggio viene aggiunto il valore del pezzo presente nella casella.

*Esempio:* La figura qui sotto a sinistra rappresenta una scacchiera con i valori dei pezzi. Nelle altre figure sono evidenziati tre possibili cammini di valori 33, 55 e 24, rispettivamente.




Si vuole costruire un algoritmo che data la scacchiera su cui sono collocati i pezzi con i relativi valori, determini il *punteggio massimo* ottenibile dal giocatore.

*Cosa si richiede:* Supponete che l'input del problema sia rappresentato da una matrice $M$ di dimensione $n \times n$, contenente numeri interi, in cui l'elemento $m_{i,j}$ è il valore del pezzo presente nella casella $(i, j)$ se questa non è vuota, ed è 0 altrimenti. Indichiamo con $c_{i,j}$ il massimo punteggio di un cammino che inizia in una qualunque casella della colonna di sinistra (cioè la numero 1) e termina nella casella $(i, j)$. Pertanto $c_{i,1} = m_{i,1}$, per $i = 1, \ldots, n$.
Qui a fianco sono mostrate la matrice $M$ e la matrice $C$ contenente i valori $c_{i,j}$ calcolati per l'esempio precedente.

$$M = \begin{bmatrix} 7 & 0 & 10 & 0 & 0 \\ 0 & 4 & 0 & 12 & 0 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 8 & 0 & 50 & 0 \\ 0 & 0 & 0 & 16 & 0 \end{bmatrix} \qquad C = \begin{bmatrix} 7 & 7 & 21 & 21 & 21 \\ 0 & 11 & 11 & 33 & 33 \\ 0 & 0 & 0 & 0 & 55 \\ 0 & 8 & 8 & 50 & 50 \\ 0 & 0 & 0 & 24 & 24 \end{bmatrix}$$

(a) Tenendo conto delle regole con cui il giocatore si può muovere, scrivete una o più formule per il calcolo dei punteggi massimi dei cammini che terminano nelle caselle di una colonna $j > 1$ (cioè i valori $c_{i,j}$), sulla base dei valori dei cammini che terminano nella colonna $j - 1$. Indicate come ottenere la soluzione del problema, cioè il punteggio massimo ottenibile dal giocatore, dai valori $c_{i,n}$.

(b) Descrivete *sinteticamente a parole* e poi *ad alto livello* in pseudocodice, un algoritmo basato sulla tecnica di *programmazione dinamica* che risolva il problema.

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

Cognome.....

Nome.....

Matricola.....

# Algoritmi e Strutture Dati

Prova scritta del 2 febbraio 2018

**TEMPO DISPONIBILE: 2 ore**

*Le risposte agli esercizi 1, 2, 3 devono essere scritte negli appositi riquadri su questo foglio (risposte scritte su altri fogli non saranno considerate). La soluzione dell'esercizio 4 va scritta su uno dei fogli di protocollo forniti. Le brutte copie NON devono essere consegnate.* *Ricordatevi di scrivere cognome e nome su tutto ciò che consegnate.*

**1.** Considerate funzione $f : \{\texttt{a}, \texttt{b}, \ldots, \texttt{z}\} \to \{0, 1, \ldots, 15\}$ definita come segue:

| $k$ | $f(k)$ |
|---|---|
| a | 0 |
| b | 1 |
| c | 2 |
| d | 3 |
| e | 4 |
| f | 5 |
| g | 6 |
| h | 6 |
| i | 7 |

| $k$ | $f(k)$ |
|---|---|
| j | 7 |
| k | 7 |
| l | 8 |
| m | 9 |
| n | 10 |
| o | 10 |
| p | 11 |
| q | 12 |
| r | 12 |

| $k$ | $f(k)$ |
|---|---|
| s | 13 |
| t | 13 |
| u | 14 |
| v | 14 |
| w | 15 |
| x | 15 |
| y | 15 |
| z | 15 |

Sia $h$ la funzione che trasforma ogni parola $k$ sull'alfabeto $\{\texttt{a}, \texttt{b}, \ldots, \texttt{z}\}$ nell'intero che si ottiene applicando $f$ al primo carattere di $k$ e $g$ la funzione che trasforma ogni parola $k$ nel *più piccolo numero primo* maggiore o uguale al valore che si ottiene applicando $f$ *all'ultimo* carattere di $k$.

*Esempi:* $h(\texttt{topo}) = 13$, $g(\texttt{topo}) = 11$, $h(\texttt{rinoceronte}) = 12$, $g(\texttt{rinoceronte}) = 5$.

Inserite nella tabella hash a destra, inizialmente vuota, le seguenti parole, nell'ordine indicato:

```
coyote calamaro cavallo cane falco platessa pavone
alce
```

Come funzione hash utilizzate $h$. Per la gestione delle collisioni utilizzate l'hashing doppio mediante la funzione

$$c(k, i) = (\, h(k) + i \cdot g(k) \,) \bmod 16$$

| | |
|---|---|
| 0 | |
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | |
| 10 | |
| 11 | |
| 12 | |
| 13 | |
| 14 | |
| 15 | |

**2.** Considerate un albero AVL e un albero 2-3 ottenuti inserendo uno dopo l'altro, nell'ordine indicato, i seguenti numeri a partire da alberi inizialmente vuoti: `15 20 22 17 18 12 14`

| | |
|---|---|
| (a) Disegnate l'albero AVL. | (b) Disegnate l'albero 2-3. |
| (c) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in *ampiezza* dell'albero AVL. | (d) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in *ordine posticipato* dell'albero AVL. |

**3.** Considerate la seguente sequenza di numeri memorizzata in un array che deve essere ordinata in modo crescente:
`16 15 20 22 17 18 12 14`

(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`.

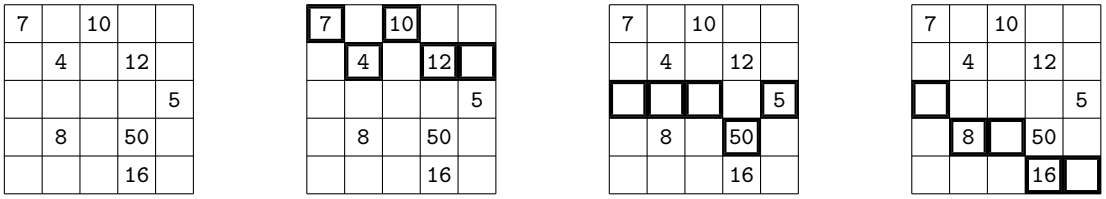
(b) Supponete di ordinare la sequenza mediante l'algoritmo `bubbleSort`. Indicate il contenuto dell'array dopo la prima iterazione del ciclo principale dell'algoritmo.

**4.** È data una scacchiera $n \times n$ su cui sono collocati alcuni pezzi. Ad ogni pezzo è associato un valore intero positivo. Un giocatore si muove sulla scacchiera partendo da una qualunque casella della colonna più a sinistra, fino a raggiungere una casella della colonna più a destra, utilizzando le seguenti regole:

- da una casella il giocatore si può muovere nella casella immediatamente a destra sulla stessa riga, se quest'ultima casella è vuota;
- da una casella il giocatore si può muovere nella casella immediatamente a destra sulla riga immediatamente superiore (se esiste), se quest'ultima casella contiene un pezzo;
- da una casella il giocatore si può muovere nella casella immediatamente a destra sulla riga immediatamente inferiore (se esiste), se quest'ultima casella contiene un pezzo.

Al giocatore viene attribuito un punteggio pari alla somma dei valori dei pezzi incontrati nel cammino percorso all'interno della scacchiera. In altre parole, partendo da zero, ogni volta che il giocatore raggiunge una casella contenente un pezzo, al punteggio viene aggiunto il valore del pezzo presente nella casella.

*Esempio:* La figura qui sotto a sinistra rappresenta una scacchiera con i valori dei pezzi. Nelle altre figure sono evidenziati tre possibili cammini di valori 33, 55 e 24, rispettivamente.




Si vuole costruire un algoritmo che data la scacchiera su cui sono collocati i pezzi con i relativi valori, determini il *punteggio massimo* ottenibile dal giocatore.

*Cosa si richiede:* Supponete che l'input del problema sia rappresentato da una matrice $M$ di dimensione $n \times n$, contenente numeri interi, in cui l'elemento $m_{i,j}$ è il valore del pezzo presente nella casella $(i, j)$ se questa non è vuota, ed è 0 altrimenti. Indichiamo con $c_{i,j}$ il massimo punteggio di un cammino che inizia in una qualunque casella della colonna di sinistra (cioè la numero 1) e termina nella casella $(i, j)$. Pertanto $c_{i,1} = m_{i,1}$, per $i = 1, \ldots, n$.
Qui a fianco sono mostrate la matrice $M$ e la matrice $C$ contenente i valori $c_{i,j}$ calcolati per l'esempio precedente.

$$M = \begin{bmatrix} 7 & 0 & 10 & 0 & 0 \\ 0 & 4 & 0 & 12 & 0 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 8 & 0 & 50 & 0 \\ 0 & 0 & 0 & 16 & 0 \end{bmatrix} \qquad C = \begin{bmatrix} 7 & 7 & 21 & 21 & 21 \\ 0 & 11 & 11 & 33 & 33 \\ 0 & 0 & 0 & 0 & 55 \\ 0 & 8 & 8 & 50 & 50 \\ 0 & 0 & 0 & 24 & 24 \end{bmatrix}$$

(a) Tenendo conto delle regole con cui il giocatore si può muovere, scrivete una o più formule per il calcolo dei punteggi massimi dei cammini che terminano nelle caselle di una colonna $j > 1$ (cioè i valori $c_{i,j}$), sulla base dei valori dei cammini che terminano nella colonna $j - 1$. Indicate come ottenere la soluzione del problema, cioè il punteggio massimo ottenibile dal giocatore, dai valori $c_{i,n}$.

(b) Descrivete *sinteticamente a parole* e poi *ad alto livello* in pseudocodice, un algoritmo basato sulla tecnica di *programmazione dinamica* che risolva il problema.

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

Cognome.....

Nome.....

Matricola.....

# Algoritmi e Strutture Dati

Prova scritta del 2 febbraio 2018

**TEMPO DISPONIBILE: 2 ore**

*Le risposte agli esercizi 1, 2, 3 devono essere scritte negli appositi riquadri su questo foglio (risposte scritte su altri fogli non saranno considerate). La soluzione dell'esercizio 4 va scritta su uno dei fogli di protocollo forniti. Le brutte copie NON devono essere consegnate.* *Ricordatevi di scrivere cognome e nome su tutto ciò che consegnate.*

**1.** Considerate funzione $f : \{\texttt{a}, \texttt{b}, \ldots, \texttt{z}\} \to \{0, 1, \ldots, 15\}$ definita come segue:

| $k$ | $f(k)$ |
|---|---|
| a | 0 |
| b | 1 |
| c | 2 |
| d | 3 |
| e | 4 |
| f | 5 |
| g | 6 |
| h | 6 |
| i | 7 |

| $k$ | $f(k)$ |
|---|---|
| j | 7 |
| k | 7 |
| l | 8 |
| m | 9 |
| n | 10 |
| o | 10 |
| p | 11 |
| q | 12 |
| r | 12 |

| $k$ | $f(k)$ |
|---|---|
| s | 13 |
| t | 13 |
| u | 14 |
| v | 14 |
| w | 15 |
| x | 15 |
| y | 15 |
| z | 15 |

Sia $h$ la funzione che trasforma ogni parola $k$ sull'alfabeto $\{\texttt{a}, \texttt{b}, \ldots, \texttt{z}\}$ nell'intero che si ottiene applicando $f$ al primo carattere di $k$ e $g$ la funzione che trasforma ogni parola $k$ nel *più piccolo numero primo* maggiore o uguale al valore che si ottiene applicando $f$ *all'ultimo* carattere di $k$.

*Esempi:* $h(\texttt{topo}) = 13$, $g(\texttt{topo}) = 11$, $h(\texttt{rinoceronte}) = 12$, $g(\texttt{rinoceronte}) = 5$.

Inserite nella tabella hash a destra, inizialmente vuota, le seguenti parole, nell'ordine indicato:

```
calabrone cigno criceto cinghiale fagiano puma
piccione ape
```

Come funzione hash utilizzate $h$. Per la gestione delle collisioni utilizzate l'hashing doppio mediante la funzione

$$c(k, i) = (\, h(k) + i \cdot g(k) \,) \bmod 16$$

| | |
|---|---|
| 0 | |
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | |
| 10 | |
| 11 | |
| 12 | |
| 13 | |
| 14 | |
| 15 | |

**2.** Considerate un albero AVL e un albero 2-3 ottenuti inserendo uno dopo l'altro, nell'ordine indicato, i seguenti numeri a partire da alberi inizialmente vuoti: `17 22 24 19 20 14 16`

| | |
|---|---|
| (a) Disegnate l'albero AVL. | (b) Disegnate l'albero 2-3. |
| (c) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in *ampiezza* dell'albero AVL. | (d) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in *ordine posticipato* dell'albero AVL. |

**3.** Considerate la seguente sequenza di numeri memorizzata in un array che deve essere ordinata in modo crescente:
`18 17 22 24 19 20 14 16`

(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`.

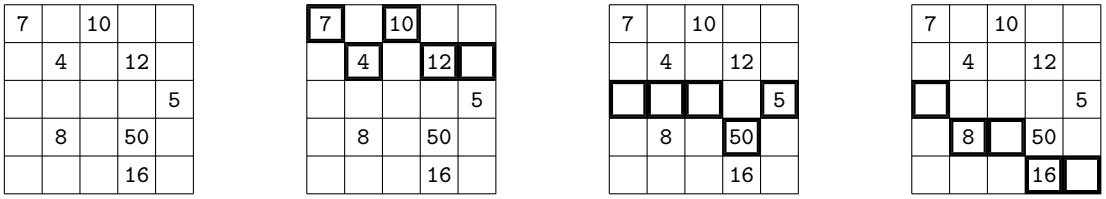
(b) Supponete di ordinare la sequenza mediante l'algoritmo `bubbleSort`. Indicate il contenuto dell'array dopo la prima iterazione del ciclo principale dell'algoritmo.

**4.** È data una scacchiera $n \times n$ su cui sono collocati alcuni pezzi. Ad ogni pezzo è associato un valore intero positivo. Un giocatore si muove sulla scacchiera partendo da una qualunque casella della colonna più a sinistra, fino a raggiungere una casella della colonna più a destra, utilizzando le seguenti regole:

- da una casella il giocatore si può muovere nella casella immediatamente a destra sulla stessa riga, se quest'ultima casella è vuota;
- da una casella il giocatore si può muovere nella casella immediatamente a destra sulla riga immediatamente superiore (se esiste), se quest'ultima casella contiene un pezzo;
- da una casella il giocatore si può muovere nella casella immediatamente a destra sulla riga immediatamente inferiore (se esiste), se quest'ultima casella contiene un pezzo.

Al giocatore viene attribuito un punteggio pari alla somma dei valori dei pezzi incontrati nel cammino percorso all'interno della scacchiera. In altre parole, partendo da zero, ogni volta che il giocatore raggiunge una casella contenente un pezzo, al punteggio viene aggiunto il valore del pezzo presente nella casella.

*Esempio:* La figura qui sotto a sinistra rappresenta una scacchiera con i valori dei pezzi. Nelle altre figure sono evidenziati tre possibili cammini di valori 33, 55 e 24, rispettivamente.




Si vuole costruire un algoritmo che data la scacchiera su cui sono collocati i pezzi con i relativi valori, determini il *punteggio massimo* ottenibile dal giocatore.

*Cosa si richiede:* Supponete che l'input del problema sia rappresentato da una matrice $M$ di dimensione $n \times n$, contenente numeri interi, in cui l'elemento $m_{i,j}$ è il valore del pezzo presente nella casella $(i, j)$ se questa non è vuota, ed è 0 altrimenti. Indichiamo con $c_{i,j}$ il massimo punteggio di un cammino che inizia in una qualunque casella della colonna di sinistra (cioè la numero 1) e termina nella casella $(i, j)$. Pertanto $c_{i,1} = m_{i,1}$, per $i = 1, \ldots, n$.
Qui a fianco sono mostrate la matrice $M$ e la matrice $C$ contenente i valori $c_{i,j}$ calcolati per l'esempio precedente.

$$M = \begin{bmatrix} 7 & 0 & 10 & 0 & 0 \\ 0 & 4 & 0 & 12 & 0 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 8 & 0 & 50 & 0 \\ 0 & 0 & 0 & 16 & 0 \end{bmatrix} \qquad C = \begin{bmatrix} 7 & 7 & 21 & 21 & 21 \\ 0 & 11 & 11 & 33 & 33 \\ 0 & 0 & 0 & 0 & 55 \\ 0 & 8 & 8 & 50 & 50 \\ 0 & 0 & 0 & 24 & 24 \end{bmatrix}$$

(a) Tenendo conto delle regole con cui il giocatore si può muovere, scrivete una o più formule per il calcolo dei punteggi massimi dei cammini che terminano nelle caselle di una colonna $j > 1$ (cioè i valori $c_{i,j}$), sulla base dei valori dei cammini che terminano nella colonna $j - 1$. Indicate come ottenere la soluzione del problema, cioè il punteggio massimo ottenibile dal giocatore, dai valori $c_{i,n}$.

(b) Descrivete *sinteticamente a parole* e poi *ad alto livello* in pseudocodice, un algoritmo basato sulla tecnica di *programmazione dinamica* che risolva il problema.

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

Cognome.....

Nome.....

Matricola.....

# Algoritmi e Strutture Dati

Prova scritta del 2 febbraio 2018

**TEMPO DISPONIBILE: 2 ore**

*Le risposte agli esercizi 1, 2, 3 devono essere scritte negli appositi riquadri su questo foglio (risposte scritte su altri fogli non saranno considerate). La soluzione dell'esercizio 4 va scritta su uno dei fogli di protocollo forniti. Le brutte copie NON devono essere consegnate.* *Ricordatevi di scrivere cognome e nome su tutto ciò che consegnate.*

**1.** Considerate funzione $f : \{\texttt{a}, \texttt{b}, \ldots, \texttt{z}\} \to \{0, 1, \ldots, 15\}$ definita come segue:

| $k$ | $f(k)$ |
|---|---|
| a | 0 |
| b | 1 |
| c | 2 |
| d | 3 |
| e | 4 |
| f | 5 |
| g | 6 |
| h | 6 |
| i | 7 |

| $k$ | $f(k)$ |
|---|---|
| j | 7 |
| k | 7 |
| l | 8 |
| m | 9 |
| n | 10 |
| o | 10 |
| p | 11 |
| q | 12 |
| r | 12 |

| $k$ | $f(k)$ |
|---|---|
| s | 13 |
| t | 13 |
| u | 14 |
| v | 14 |
| w | 15 |
| x | 15 |
| y | 15 |
| z | 15 |

Sia $h$ la funzione che trasforma ogni parola $k$ sull'alfabeto $\{\texttt{a}, \texttt{b}, \ldots, \texttt{z}\}$ nell'intero che si ottiene applicando $f$ al primo carattere di $k$ e $g$ la funzione che trasforma ogni parola $k$ nel *più piccolo numero primo* maggiore o uguale al valore che si ottiene applicando $f$ *all'ultimo* carattere di $k$.

*Esempi:* $h(\texttt{topo}) = 13$, $g(\texttt{topo}) = 11$, $h(\texttt{rinoceronte}) = 12$, $g(\texttt{rinoceronte}) = 5$.

Inserite nella tabella hash a destra, inizialmente vuota, le seguenti parole, nell'ordine indicato:

```
cane cavallo calamaro coyote falco puzzola pavone
alce
```

Come funzione hash utilizzate $h$. Per la gestione delle collisioni utilizzate l'hashing doppio mediante la funzione

$$c(k, i) = (\, h(k) + i \cdot g(k) \,) \bmod 16$$

| | |
|---|---|
| 0 | |
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | |
| 10 | |
| 11 | |
| 12 | |
| 13 | |
| 14 | |
| 15 | |

**2.** Considerate un albero AVL e un albero 2-3 ottenuti inserendo uno dopo l'altro, nell'ordine indicato, i seguenti numeri a partire da alberi inizialmente vuoti: `19 24 26 21 22 16 18`

| | |
|---|---|
| (a) Disegnate l'albero AVL. | (b) Disegnate l'albero 2-3. |
| (c) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in *ampiezza* dell'albero AVL. | (d) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in *ordine posticipato* dell'albero AVL. |

**3.** Considerate la seguente sequenza di numeri memorizzata in un array che deve essere ordinata in modo crescente:
`20 19 24 26 21 22 16 18`

(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`.

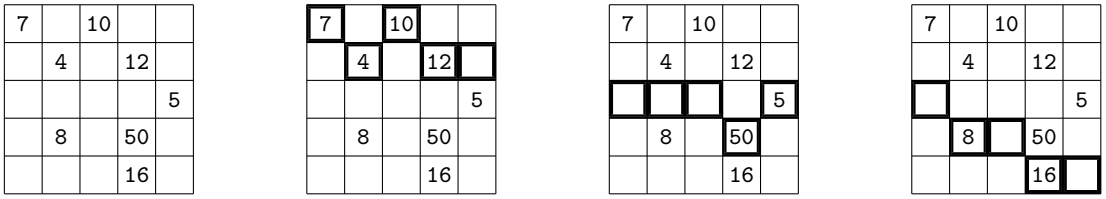
(b) Supponete di ordinare la sequenza mediante l'algoritmo `bubbleSort`. Indicate il contenuto dell'array dopo la prima iterazione del ciclo principale dell'algoritmo.

**4.** È data una scacchiera $n \times n$ su cui sono collocati alcuni pezzi. Ad ogni pezzo è associato un valore intero positivo. Un giocatore si muove sulla scacchiera partendo da una qualunque casella della colonna più a sinistra, fino a raggiungere una casella della colonna più a destra, utilizzando le seguenti regole:

- da una casella il giocatore si può muovere nella casella immediatamente a destra sulla stessa riga, se quest'ultima casella è vuota;
- da una casella il giocatore si può muovere nella casella immediatamente a destra sulla riga immediatamente superiore (se esiste), se quest'ultima casella contiene un pezzo;
- da una casella il giocatore si può muovere nella casella immediatamente a destra sulla riga immediatamente inferiore (se esiste), se quest'ultima casella contiene un pezzo.

Al giocatore viene attribuito un punteggio pari alla somma dei valori dei pezzi incontrati nel cammino percorso all'interno della scacchiera. In altre parole, partendo da zero, ogni volta che il giocatore raggiunge una casella contenente un pezzo, al punteggio viene aggiunto il valore del pezzo presente nella casella.

*Esempio:* La figura qui sotto a sinistra rappresenta una scacchiera con i valori dei pezzi. Nelle altre figure sono evidenziati tre possibili cammini di valori 33, 55 e 24, rispettivamente.




Si vuole costruire un algoritmo che data la scacchiera su cui sono collocati i pezzi con i relativi valori, determini il *punteggio massimo* ottenibile dal giocatore.

*Cosa si richiede:* Supponete che l'input del problema sia rappresentato da una matrice $M$ di dimensione $n \times n$, contenente numeri interi, in cui l'elemento $m_{i,j}$ è il valore del pezzo presente nella casella $(i, j)$ se questa non è vuota, ed è 0 altrimenti. Indichiamo con $c_{i,j}$ il massimo punteggio di un cammino che inizia in una qualunque casella della colonna di sinistra (cioè la numero 1) e termina nella casella $(i, j)$. Pertanto $c_{i,1} = m_{i,1}$, per $i = 1, \ldots, n$.
Qui a fianco sono mostrate la matrice $M$ e la matrice $C$ contenente i valori $c_{i,j}$ calcolati per l'esempio precedente.

$$M = \begin{bmatrix} 7 & 0 & 10 & 0 & 0 \\ 0 & 4 & 0 & 12 & 0 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 8 & 0 & 50 & 0 \\ 0 & 0 & 0 & 16 & 0 \end{bmatrix} \qquad C = \begin{bmatrix} 7 & 7 & 21 & 21 & 21 \\ 0 & 11 & 11 & 33 & 33 \\ 0 & 0 & 0 & 0 & 55 \\ 0 & 8 & 8 & 50 & 50 \\ 0 & 0 & 0 & 24 & 24 \end{bmatrix}$$

(a) Tenendo conto delle regole con cui il giocatore si può muovere, scrivete una o più formule per il calcolo dei punteggi massimi dei cammini che terminano nelle caselle di una colonna $j > 1$ (cioè i valori $c_{i,j}$), sulla base dei valori dei cammini che terminano nella colonna $j - 1$. Indicate come ottenere la soluzione del problema, cioè il punteggio massimo ottenibile dal giocatore, dai valori $c_{i,n}$.

(b) Descrivete *sinteticamente a parole* e poi *ad alto livello* in pseudocodice, un algoritmo basato sulla tecnica di *programmazione dinamica* che risolva il problema.

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