Cognome.....

Nome.....

Matricola.....

# Algoritmi e Strutture Dati

Prova scritta del 20 settembre 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 la funzione $f : \{\mathtt{a}, \mathtt{b}, \ldots, \mathtt{z}\} \to \{0, 1, \ldots, 15\}$ definita come segue:

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

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

| $x$ | $f(x)$ |
|---|---|
| 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 $\{\mathtt{a}, \mathtt{b}, \ldots, \mathtt{z}\}$ 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 ottenuto applicando $f$ al terzo carattere $k$ (si ricordi che il più piccolo numero primo è 2).

*Esempi:* $h(\mathtt{ratto}) = 12$, $g(\mathtt{ratto}) = 13$, $h(\mathtt{cane}) = 2$, $g(\mathtt{cane}) = 11$.

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

```
salmone tonno rombo aringa cernia aragosta totano
ostrica
```

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.** La seguente sequenza di numeri, memorizzata in un array, deve essere ordinata in modo crescente:
`15 26 33 14 1 19 4 3`

(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`.

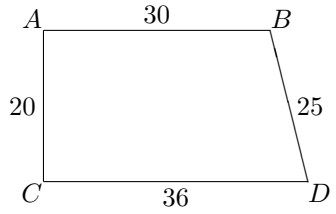
(b) Supponete di ordinare la sequenza mediante l'algoritmo `insertionSort`. Indicate il contenuto dell'array dopo la terza iterazione del ciclo principale dell'algoritmo.

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

(d) Supponete di ordinare la sequenza mediante l'algoritmo `heapSort`. Indicate il contenuto dell'array immediatamente dopo lo scambio che colloca `26` nella sua posizione finale.

**3.** Dato l'elenco delle strade che collegano *direttamente* le città di una nazione, con le rispettive lunghezze, si vuole ottenere un elenco delle distanze tra le coppie di città differenti tra loro (cioè escludendo le coppie formate dalla stessa città per le quali, ovviamente, la distanza è 0), in ordine di distanza non decrescente. Si suppone che ogni strada sia percorribile in entrambi i sensi di marcia, pertanto la lunghezza della strada dalla città $x$ alla città $y$ è uguale a quella dalla città $y$ alla città $x$.

*Esempio*
Per le 4 città $A, B, C, D$, con le strade schematizzate nella figura a sinistra, sarà prodotto l'elenco indicato a destra:




```
A C 20
B D 25
A B 30
C D 36
B C 50
A D 55
```

*Cosa si richiede*

Indichiamo le città con $c_1, \ldots, c_n$. Progettate un algoritmo che risolva il problema, ricevendo in ingresso una matrice $M$ dove, per $i, j = 1, \ldots, n$, l'elemento $m_{i,j}$ è così definito:

$$m_{i,j} = \begin{cases} 0 & \text{se } i = j, \\ \text{lunghezza strada che collega } \textit{direttamente } c_i \text{ e } c_j & \text{se tale strada esiste,} \\ \infty & \text{altrimenti.} \end{cases}$$

Nell'elenco in output potete fornire gli indici corrispondenti alle città (nell'esempio precedente, se $c_1 =$ `A` e $c_3 =$ `C`, la prima riga dell'elenco in output sarà `1 3 20` anziché `A C 20`).

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

(a) Descrivete *sinteticamente a parole* e poi *ad alto livello in pseudocodice* un algoritmo che risolva il problema. Potete adattare uno o più algoritmi presentati nel corso (indicate esplicitamente quali algoritmi utilizzate). Non è richiesta la codifica della parte relativa all'ordinamento. Pertanto potete richiamare l'ordinamento direttamente scrivendo `ordina(...)` dove al posto di `...` fornite la struttura da ordinare. Fate attenzione a indicare chiaramente la struttura che deve essere ordinata, eventualmente scrivendo lo pseudocodice per costruirla. Ricordatevi anche di indicare (per mezzo di un commento o in altro modo) *la chiave* rispetto alla quale dovrà essere effettuato l'ordinamento.

(b) Indicate il nome di un algoritmo di ordinamento, a vostra scelta, che può essere utilizzato per risolvere il problema.

(c) Fornire una stima del tempo di calcolo dell'intero algoritmo ottenuto in funzione del numero $n$ di città, indicando esplicitamente come la parte relativa all'ordinamento contribuisca a tale stima.

(d) Fornite una stima della memoria utilizzata in funzione del numero $n$ di città, tenendo conto dell'algoritmo di ordinamento scelto.

*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 massimo e il minimo tra $\infty$ e un valore $x$ sono rispettivamente $\infty$ e $x$.
- Non cambiate quanto stabilito nel testo dell'esercizio per i nomi della matrice di input $M$ e i nomi $c_1, \ldots, c_n$ (e relativi indici) utilizzati per indicare le città.
- Tutte le risposte devono essere adeguatamente giustificate.