Note preliminari

Dati di ingresso ed uscita

Per ogni traccia, salvo diversa ed esplicita indicazione:

- i *dati in ingresso* devono essere sempre specificati dall'utente mediante riga di comando **e/o** lettura da file di testo;
- i *dati in uscita* devono essere sempre restituiti all'utente mediante riga di comando **e/o** file di testo.

Supponendo ad esempio di stare eseguendo il programma **`mio_programma.exe`**, il flusso di interazione con l'utente da prompt dei comandi/shell sarà del tipo:

```
$ mio_programma.exe
$ Benvenuto! Selezionare la modalità di lettura dei dati in ingresso (-f per file, -r per riga di comando)
$ -f
$ Grazie! Specificare il file da cui leggere!
$ # esecuzione del programma…
$ Programma terminato! Si vogliono visualizzare i risultati su file o su riga di comando?
$ -r
$ Array risultante: [1, 2, 3, 4, 5]
```

Fonti

Le tracce sono risolvibili con gli argomenti trattati durante il corso. Tuttavia, alcune contengono fonti per la contestualizzazione del problema.

Difficoltà relativa

Le tracce sono pensate come aventi la medesima difficoltà di implementazione.

Traccia 1

Sviluppare un tool mediante che, dati una figura geometrica ed i relativi parametri, ne restituisca area e perimetro. Le figure ammissibili sono (almeno) triangolo, rettangolo, parallelogramma, quadrato, rombo, trapezio e cerchio.

Traccia 2

Sviluppare un tool che, dato un certo numero n, restituisca tutti i numeri primi compresi tra 0 ed n utilizzando la tecnica del Crivello di Eratostene.

Fonte

Crivello di Eratostene

Traccia 3

Sviluppare un tool che, data in ingresso una generica funzione di primo o secondo grado, ne effettui lo studio analitico, considerando *almeno* lo studio di dominio, codominio, parità e disparità, intersezioni con gli assi cartesiani ed il segno.

Traccia 4

Sviluppare un tool che permetta di determinare se un numero è primo usando il test di Fermat.

Fonte

Test di Fermat

Traccia 5

Sviluppare un tool che stabilisca la natura di un triangolo a partire da tutte le combinazioni ammissibili di angoli e lati, sfruttando tecniche algebriche e trigonometriche.

Traccia 6

Sviluppare un tool crittografico che permetta di cifrare e decifrare un messaggio di lunghezza arbitraria, che può essere letto da un file di testo o inserito via riga di comando dall'utente. Il tool dovrà mascherare il messaggio basandosi sul cifrario di Cesare.

Fonte

Cifrario di Cesare

Traccia 7

La tecnica del *dollar-cost averaging* è una strategia di investimento nella quale un investitore divide il quantitativo totale di denaro investito in azioni in parti uguali, generando una serie di acquisti periodici di uno stesso asset nel tempo allo scopo di ridurre l'impatto delle fluttuazioni di mercato. Un esempio è il seguente:

- mese 1: acquisto di 100 € di azioni dell'azienda A, quotate 10 € ciascuna, per un totale di 10 azioni;
- mese 2: acquisto di 100 € di azioni dell'azienda A, che ha subito un ribasso a 5 €, per un totale di 20 azioni che, sommate alle precedenti, danno 30 azioni in nostro possesso;
- mese 3: acquisto di 100 € di azioni dell'azienda A, che si è rialzata a 10 €, per un totale di 10 azioni che, sommate alle precedenti, danno 40 azioni in nostro possesso.

Tale esempio si reitera nel tempo per intervalli periodici arbitrariamente lunghi.
Si sviluppi un tool che, dati i seguenti parametri (arbitrari), stabilisca il profitto (o la perdita) a metà e fine investimento:

- parametro 1: intervallo temporale di durata dell'investimento in mesi;
- parametro 2: periodicità dell'acquisto (es. mensile, bimestrale, etc.);
- parametro 3: cifra investita ad ogni acquisto;
- parametro 4: titoli nel nostro portafoglio azionario;
- parametro 5: per ognuno dei titoli inseriti nel portafoglio, la stima dell'andamento (in percentuale sul valore del titolo).

Traccia 8

Un cifrario a trasposizione colonnare opera in questo modo.
Dato una parola chiave ed un ordine di permutazione arbitrario, le parole del messaggio originario sono disposte secondo un numero di colonne pari a quelle della parola chiave; eventuali spazi e punteggiatura vengono trascurati. Il cifrario opera leggendo la matrice per colonne, nell'ordine di permutazione stabilito.

Nel seguente esempio, consideriamo come parola chiave per l'appunto "CHIAVE", e come ordine di permutazione "1-3-2-5-4-6", il che significa che leggeremo prima la prima colonna, poi la terza, poi la seconda, e via fino all'ultima. È importante sottolineare come la trasposizione possa essere regolare (matrice quadrata mediante l'aggiunta di lettere casuali) o meno.

C	H	I	A	V	E
1	3	2	5	4	6
M	E	S	S	A	G
G	I	O	D	I	E
S	E	M	P	I	O
S	E	M	P	L	I
C	E	D	X	B	V

Il messaggio originario è:
Messaggio di esempio semplice

Il messaggio che sarà inviato è:
MGSSCSOMMDEIEEEAIILBSDPPXGEOIV

La decifrazione avviene suddividendo il messaggio secondo la lunghezza della parola chiave, e riorganizzandolo secondo l'ordine di permutazione stabilito. Quindi:

Lunghezza messaggio: 30 caratteri
Lunghezza parola chiave: 6 caratteri
Lunghezza colonne: 5 caratteri

Suddivisione messaggio:
MGSSC SOMMD EIEEE AIILB SDPPX GEOIV

Riapplicando il metodo precedente, e leggendo per righe, possiamo risalire al messaggio originario.

1	3	2	5	4	6
M	E	S	S	A	G
G	I	O	D	I	E
S	E	M	P	I	O
S	E	M	P	L	I
C	E	D	X	B	V

Scrivere un tool che, dati in ingresso un messaggio di lunghezza ed un ordine di permutazione arbitrari, cifrino e decifrino il messaggio secondo questo metodo.

Fonte

WikiPedia

Traccia 9

Sviluppare il tool "Libretto dello Studente", che permette di aggiungere e rimuovere gli esami effettuati dallo studente, tenendo traccia di data e voto. Lo strumento dovrà inoltre ordinare gli esami per voto, e calcolare il voto medio di laurea. Come mezzo di memorizzazione, usare un file di testo.

Traccia 10

Sviluppare un tool che, data una matrice di dimensioni arbitrarie n ed m, ne calcoli l'inversa (se invertibile) usando l'algoritmo di Gauss - Jordan. Lo strumento dovrà ovviamente restituire un errore qualora l'utente passi in ingresso una matrice non invertibile.

Fonte

Invertibilità di una matrice

Traccia 11

Sviluppare un tool che permetta di generare delle tavole pitagoriche di dimensioni n per m.

Fonte

WikiPedia

Traccia 12

Una matrice quadrata è chiamata *quadrato magico* se la somma degli elementi su ogni riga e colonna, oltre che sulle due diagonali principali, risulta essere costante. Sviluppare un tool in grado di generare un quadrato magico di ordine n composto da numeri interi casuali positivi.

Traccia 13

Una serie di dati storici è caratterizzata almeno da due parametri: il primo è l'*andamento*, di solito calcolato in percentuale e può essere positivo o negativo, il secondo è legato alla *stagionalità*, ovvero a fenomeni che si ripetono nel tempo ad intervalli regolari. Una semplice tecnica di rimozione del contributo legato all'andamento (o *detrending*) consiste nel calcolare, per ogni dato, la differenza rispetto a quello immediatamente precedente. Sviluppare un tool che effettui il detrending a partire da una serie di dati storici di lunghezza arbitraria, passati in input sotto forma vettoriale in un file di testo. Si trascurino i parametri legati alla determinazione del periodo dei dati.

Fonte

https://people.duke.edu/~rnau/411diff.htm#firstdiff

Traccia 14

Sviluppare un tool che verifichi la robustezza di una password passata in input dall'utente. Il tool dovrà assegnare diversi punteggi a diverse scelte fatte dall'utente: ad esempio, usare solo caratteri minuscoli darà un punteggio negativo, mentre inserire un carattere speciale potrebbe aumentare un punteggio di un fattore *q* arbitrario. I punteggi devono essere definiti (e motivati) dai candidati.

Traccia 15

Il metodo della *regressione lineare* si basa sul *metodo dei minimi quadrati* per determinare un'approssimazione di una funzione arbitraria. Sviluppare un tool che, data in ingresso una funzione arbitraria, utilizzi tale metodo per valutare la retta di regressione.

Fonte

Metodo dei minimi quadrati

Traccia 16

Scrivere un tool che, dati due testi di lunghezza arbitraria, permetta di verificare un eventuale plagio. Il tool dovrà confrontare la lunghezza dei due testi ed il loro contenuto. Si noti che l'analisi **non** dovrà essere effettuata dal punto di vista semantico.

Traccia 17

Un *filtro a media mobile* è un operatore che, dato in ingresso una matrice A a dimensioni arbitrarie, calcola la media dell'array bidimensionale di dimensione $n \times n$ attorno tutte le coppie (i, j) di A. Nel caso la posizione (i, j) si trovi nei pressi dei bordi di A, tutti gli elementi non esistenti sono considerati come a valore 0. Ad esempio, questo è il risultato dell'applicazione di un filtro 3×3 alla matrice a sinistra della freccia.

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} * \begin{bmatrix}3 & 2 & 4 & 2\\ 2 & 3 & 4 & 2\\ 2 & 3 & 4 & 3\\ 1 & 2 & 3 & 2\end{bmatrix} \Rightarrow \begin{bmatrix}1.11 & 2 & 1.89 & 1.33\\ 1.67 & 3 & 3 & 2.11\\ 1.33 & 2.67 & 2.89 & 2\\ 0.89 & 1.67 & 1.89 & 1.33\end{bmatrix}$$

Si noti che il simbolo $*$ indica l'operazione di *convoluzione*.
Realizzare un programma che, letto in ingresso un array bidimensionale di dimensioni arbitrarie, ed un parametro n che indichi le dimensioni del filtro a media mobile, restituisca in output la matrice filtrata. I valori di ingresso ed uscita devono essere letti e scritti a partire da file di testo.

Traccia 18

Dato un numero intero positivo, si definisce come *rotazione a sinistra* lo spostamento della cifra più significativa all'ultimo posto, mentre *rotazione a destra* lo spostamento della cifra meno significativa al primo posto. Scrivere un programma che, dato in ingresso un numero intero arbitrario, usi le operazioni di rotazione a destra ed a sinistra per ottenere la massima cifra possibile. Ad esempio:

$$101 \rightarrow 110$$
$$1024 \rightarrow 4102$$

Traccia 19

Scrivere un tool che permetta di prevedere la data di saturazione dei posti letto in un ospedale in una situazione pandemica. Il numero totale di posti letto a disposizione è un parametro arbitrariamente fissato. Il tasso di crescita dei ricoveri deve essere stimato sotto forma di funzione logistica, i cui parametri devono essere fissati in maniera arbitraria.
Errata corrige: visto e considerato che la presente traccia presuppone la conoscenza della regressione logistica, la frase "*Il tasso di crescita dei ricoveri deve essere stimato sotto forma di funzione logistica*" diventa "*Il tasso di crescita dei ricoveri deve essere mandato in input sotto forma di un vettore di dati reali concernenti almeno l'arco di trenta giorni*".

Fonte

Funzione logistica

Traccia 20

Si consideri il gioco "Mine", in cui una il giocatore esplora una griglia quadrata in cui in ogni elemento vi può essere un numero o meno una mina. Le dimensioni della griglia variano da partita a partita, e sono pari ad n. La griglia è occupata da diverse zone di colore diverso, con un numero arbitrario di elementi per zona.
Le regole per valutare la correttezza della griglia sono:

- ci devono essere solo i mine per riga;
- ci devono essere solo j per colonna;
- ci devono essere solo k per ogni zona colorata;
- negli elementi adiacenti ad una mina non ci devono essere mine.

Si realizzi un tool in grado di verificare la correttezza al variare dei parametri (n, i, j, k).

Traccia 21

Si definisca un tool in grado di valutare il numero massimo di contatti stretti avuti da un paziente che ha contratto la malattia Covid-19. Si supponga che il sistema di contact tracing abbia funzionato, ed il tool accetti di conseguenza in ingresso il grafo delle relazioni intraprese dal paziente negli ultimi k giorni. In tal senso, si supponga solo il valore medio relativo a queste date.

Traccia 22

Si sviluppi un tool che sia in grado di valutare se due vettori di lunghezza n contengono gli stessi elementi. I vettori devono essere specificati mediante file di testo. Si scriva il programma minimizzando il costo computazionale in termini di memoria e tempo.

Traccia 23

Scrivere un programma che legga in input una matrice $n \times m$ di numeri interi, ognuno dei quali può valere soltanto 0, 1 o 3. Ogni riga della matrice rappresenta i punti acquisiti dalle squadre di calcio nelle partite disputate nelle diverse giornate del campionato: 3 punti per le partite vinte, 1 punto per i pareggi e 0 della sconfitta. I risultati della k-ma giornata sono contenuti nelle righe della colonna di indice k. A partire da un valore i inserito dall'utente, il tool deve restituire la classifica relativa a quella giornata, con testa, zona Champions League (primi quattro valori), zona Europa League (quinto valore), e zona retrocessione (ultimi tre valori). Inoltre, il tool può restituire l'andamento *giornata per giornata* della posizione in classifica di qualsiasi squadra richiesta dall'utente.

Traccia 24

Dato un array di numeri casuali di dimensioni $n \times m$, e valori di ciascun elemento dell'array compresi nell'intervallo $[a, b]$, ricercare tutte le quaterne di elementi adiacenti in orizzontale, verticale e diagonale la cui somma sia compresa tra i valori $[x, y]$. La tupla (n, m, a, b, x, y) deve essere specificata dall'utente.

Traccia 25

Il *numero di inversioni* in un array permette di quantificare l'ordinamento dell'array stesso. Ad esempio, l'array [1 3 2 4] ha una sola inversione (ovvero quella tra 3 e 2). Definire un tool che valuti l'ordinamento dell'array in forma percentuale basandosi su questo concetto (ad esempio, per l'array precedente, il grado di ordinamento è del 50%).

Traccia 26

Sviluppare un algoritmo chiamato ***PEO*** (***Partition-Even-Odd***) che suddivide i numeri pari ed i numeri dispari in un array A. Il PEO ordinerà A in modo che vi siano prima tutti gli elementi pari, ordinati dal più piccolo al più grande, e poi tutti gli elementi dispari, ordinati dal più grande al più piccolo. L'algoritmo deve essere preferibilmente *in-place*, ovvero non usare array e strutture di supporto.

Traccia 27

L'algoritmo di codifica di Huffman crea un albero binario di simboli a partire da una stringa s mediante i seguenti passi.

1. I simboli vengono ordinati in una lista. Ad ogni simbolo è associato il conteggio delle sue occorrenze; ad esempio, nella parola *otto* avremo due occorrenze per il simbolo *o*, e due occorrenze per il simbolo *t*.
2. Fino a che non si ha un unico simbolo:
    a. consideriamo due simboli con frequenza di conteggio minore; questi sono i *figli* dell'albero binario, ed hanno un genitore in comune;
    b. assegnamo la somma del conteggio delle frequenze dei figli al genitore;
    c. cancelliamo i figli dalla lista.
3. Una volta completato l'albero, a partire dal nodo radice, assegnamo al ramo sinistro il codice 0 ed a quello destro il codice 1.

La codifica per ogni lettera sarà data dal percorso dalla radice al figlio.
Sviluppare un tool che calcoli in maniera automatica il codice di Huffman per una frase arbitraria.

Fonte

Codifica di Huffman

Traccia 28

Consideriamo il problema chiamato *subset-sum*. Dato un insieme di n numeri reali A, ed un numero x, stabilire se esiste un sottoinsieme $B \in A$ tale che la sommatoria di tutti gli elementi appartenenti a B sia pari ad x. I parametri n ed A devono essere arbitrariamente scelti dall'utente.

Traccia 29

Scrivere un tool che, dato in ingresso un grafo arbitrario, trovi la massima distanza possibile tra due diversi nodi dello stesso.

Traccia 30

Scrivere un tool che compari il tempo di esecuzione degli algoritmi di *insertion sort, merge sort, quick sort* e *selection sort* su un vettore arbitrario ad n elementi definito dall'utente.

Traccia 31

Scrivere un tool che converta i numeri reali in stringhe rappresentative del corrispondente numero romano. Ad esempio, 1500 diventa MD; 1492 diventa MCDCXII, e così via.

Traccia 32

Sviluppare un tool che, dato un intero positivo n, trovi il minor numero possibile di numeri quadrati (ovvero 1, 4, 9, 16, etc.) la cui somma è pari ad *n*. Ad esempio:

$$n = 12 = 4 + 4 + 4 \Rightarrow res = 3$$
$$n = 17 = 16 + 1 \Rightarrow res = 2$$

Traccia 33

Sviluppare un tool che, data una matrice $n \times n$ composta esclusivamente da 0 ed 1 determini la distanza dello 0 più vicino per ogni elemento della matrice. Si considerino due celle adiacenti come aventi distanza unitaria. Ad esempio:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

Traccia 34

Dato un vettore n scrivere un tool che cerchi tutte le coppie di indici (i, j) tali che $i \leq j$ e $n[i] > 2 \cdot n[j]$.

Traccia 35

Scrivere un tool che, data una generica lista di interi, la ordini e, successivamente, per ogni coppia di valori (i, j) nella lista consideri il rapporto tra i e j, ovvero i/j . In base a quest'ultimo, il tool deve restituire un array composto dai valori di i e j che hanno il rapporto più basso, ovvero:

$$res = [i, j]$$

Traccia 36

Dato un vettore a ad n elementi, definiamo come:

- *sotto-vettore* un vettore che comprende gli elementi contigui di a che vanno da $a[i]$ ad $a[j]$;
- *somma* di ciascun sotto-vettore come la sommatoria di tutti gli elementi dello stesso.

Dato un intero m, determinare il valore massimo in modulo m della somma di tutti i sotto-vettori di a. Ad esempio:

$$a = [1,2,3], m = 1$$
$$[1] \Rightarrow s = 1 \Rightarrow s\ mod_m = 1$$
$$[2] \Rightarrow s = 2 \Rightarrow s\ mod_m = 0$$
$$[3] \Rightarrow s = 3 \Rightarrow s\ mod_m = 1$$
$$[1,2] \Rightarrow s = 3 \Rightarrow s\ mod_m = 1$$
$$[2,3] \Rightarrow s = 5 \Rightarrow s\ mod_m = 1$$
$$[1,2,3] \Rightarrow s = 6 \Rightarrow s\ mod_m = 0$$

Il valore in output sarà quindi pari ad 1.

Traccia 37

Nel gioco "Spie", dove impersoniamo una spia con una lista di città da visitare per completare la sua missione segreta. Ogni città è rappresentata mediante una tupla (*latitudine, longitudine, altitudine, punti*). I valori della tupla sono diversi per ogni città. I punti totali della missione sono dati dalla somma dei punti di tutte le città indicate nella missione.
La nostra spia segue queste regole:

- iniziare la missione da una qualsiasi città;
- visitare le città in ordine di altitudine crescente;
- minimizzare la distanza in linea d'aria tra latitudine e longitudine del suo percorso.

Scriviamo un programma che aiuti la nostra spia a compiere la sua missione.

Traccia 38

L'obiettivo del gioco *Fabbrica di Cioccolato* è quello di creare n barrette di cioccolato usando m macchine per la fabbricazione ed l lavoratori. In un round possiamo quindi creare al più $m \times l$ barrette. Al successivo, possiamo venderne p per comprare o assumere un totale di p tra macchine e lavoratore. Supponiamo che il costo di una macchina o di un lavoratore sia quindi unitario. Vogliamo minimizzare il numero di round necessario ad ottenere le n barrette. Ad esempio, partendo da $m = 1, l = 2$, vogliamo accumulare $n = 60$ barrette. Per farlo:

1. creiamo *1* x *2* barrette, e compriamo due macchine; p varrà quindi 2;
2. creiamo *3 x 2* barrette, e compriamo tre macchine e tre lavoratori; p varrà quindi 6;
3. creiamo *6* x *5* barrette, e teniamole; p varrà quindi 0.
4. reiteriamo il punto 3, ed arriviamo a 60.

Implementiamo un tool che ci permetta di minimizzare il numero di step necessari dati arbitrari valori di (n, m, l). Il tool dovrà restituire il valore di p ottenuto ad ogni step, oltre che il numero di barrette prodotto.

Traccia 39

Immaginiamo che in città ci siano N ciclisti ed M biciclette. Ognuno di loro vuole partecipare alla *Gara dell'Anno*, organizzata da UniBa ogni 30 febbraio. Sfortunatamente, solo K ciclisti possono partecipare alla gara. Alice, che sta organizzando la gara, e vuole fare in modo che questa inizi il prima possibile, chiede a tutti di correre verso la bicicletta più vicina. Supponendo che la città sia fatta da isolati perfettamente quadrati di m metri, sviluppare un tool che, partendo dalla posizione iniziale delle bici e dei ciclisti, stimi il tempo necessario ad iniziare la

Gara dell'Anno in secondi. Si supponga, per ogni ciclista, una velocità costante pari ad h metri al secondo.

Traccia 40

La giornata dello *Studente Quadratico Medio* presenta una serie impressionante di task da completare. Per completare il task i, ad esempio, abbiamo bisogno di ben M_i minuti, e la scadenza relativa a questo task è all'istante T_i. Supponiamo, per renderci la vita più semplice, che non dobbiamo completare un compito tutto d'un fiato: possiamo suddividerlo in compiti più piccoli, e passare da un compito all'altro in maniera dinamica.
Sviluppare un tool che permetta di organizzare al meglio questi task.

Traccia 41

Definiamo una coppia di nodi (a, b) in un albero *simile* se:

- il nodo a è il genitore del nodo b;
- vale la relazione $|(a - b)| \leq k$

Sviluppare un tool che, dato un albero dove ogni nodo è etichettato da 1 ad n, restituisca tutte le possibili coppie di nodi simili.

Traccia 42

Dato un vettore di interi, creare un tool che, dato un intero x:

- aggiunga x ad ogni elemento dell'array, modificandolo in maniera permanente;
- trovi il valore assoluto di ogni elemento nell'array, e mostri a schermo la somma dei valori assoluti dell'array.

Il tool deve operare in maniera iterativa fino a che non individua un input di terminazione dell'utente mediante la lettera q.

Traccia 43

Bob sta disponendo le piante nel suo giardino di N metri quadri. Il giardino è organizzato secondo un sistema di coordinate cartesiane Oxy, discretizzato nell'ordine del metro. L'obiettivo è quello di disporre $N/2$ piante come segue:

- il centro di ogni pianta deve essere collocato su una coppia di valori interi del sistema di riferimento cartesiano (ad esempio, $(1,1)$);
- le coordinate delle $N/2$ piante devono essere distinte;
- la distanza euclidea di ogni pianta dall'origine O non deve essere un intero;
- la somma di tutte le distanze euclidee tra tutti i punti e l'origine deve essere quasi intera (ciò significa che la differenza tra questa e l'intero più vicino deve essere trascurabile ed inferiore a 10^{-6}).

Scrivere un tool che permetta a Bob di automatizzare le operazioni di assegnazione delle coordinate.

Traccia 44

Consideriamo una matrice $M = [m_0, m_1, \dots, m_{n-1}]$ di n interi. Effettuiamo q query su M, ognuna delle quali chiede di ordinare tutti gli elementi nel sottosegmento $m_{l_i}, m_{l_i+1}, \dots, m_{r_i}$. Sviluppare

un tool che, data M, stampi a schermo il valore all'indice k, con $0 \leq k < n$, dopo aver effettuato q query.

Traccia 45

Si scriva un programma che, dato un vettore ad n elementi da soli zero, ed una lista di k coppie (i, j), per ognuna delle operazioni aggiunga un valore ad ognuno degli elementi dell'array tra gli indici i e j. Una volta che tutte le operazioni sono state effettuate, restituire il valore massimo dell'array.

Traccia 46

Bob ha completato il suo giardino, ed inizia a trattare ciascuna delle N piante con il pesticida per evitare tediosi inconvenienti. Ogni giorno, se una pianta ha più pesticida di quella immediatamente alla sua sinistra, essendo più debole di quest'ultima, muore.
Sviluppare un programma che, dato il valore iniziale del pesticida in ciascuna delle piante, stabilisca per quanti giorni la pianta riesce a sopravvivere (ovverosia il tempo dopo il quale non ci sono piante con più pesticida rispetto alla pianta alla loro sinistra).

Traccia 47

Consideriamo un vettore A ad n elementi, con $A = \{a_0, a_1, \dots, a_{n-1}\}$. Sviluppare un tool che, dato un intero d, calcoli il risultato della seguente espressione:

$$\min_{0 \leq i \leq n-d} \left(\max_{i \leq j < i+d} a_j \right)$$

L'utente deve essere in grado di calcolare questa espressione q volte, ogni volta con un intero d inserito da riga di comando. Il programma termina quando l'utente inserisce il carattere #.

Traccia 48

Supponiamo di avere una strada di forma circolare ai cui bordi ci sono N pompe di benzina. A ciascuna tra queste è associato un indice che va da 0 ad $(N-1)$. Abbiamo due informazioni per ognuna delle pompe di petrolio: la prima è il quantitativo q_i di petrolio che ciascuna pompa è in grado di erogare, e la seconda è la distanza $D(i, i+1)$ tra due pompe di benzina adiacenti. Scrivere un tool che, data un'auto con serbatoio di capacità C litri, inizi ad attraversare la strada in un qualsiasi punto. Considerando un consumo di k litri al chilometro, stimare di quanto dovrà rifornirsi l'auto ad ogni pompa di benzina. Si supponga che la distanza tra ogni coppia di stazione sia paragonabile ad una frazione rilevante di $k * C$.

Traccia 49

Si consideri un carattere minuscolo in italiano, e lo si contraddistingua con c. Un'operazione di *shift* avviene quando c si trasforma nella lettera successiva dell'alfabeto. Ad esempio, $b = shift(a)$.
Dato un array composto da n caratteri s, effettuiamo q operazioni sull'array del seguente tipo:

1. *operazione* $[i, j, t]$*:* tutte le lettere incluse nel range che va da i a j sono traslate di t caratteri;

2. *operazione* $[i, j]$*:* consideriamo tutti gli indici nel range che va da i a j. Troviamo il numero di sottoinsiemi di caratteri non vuoti $c_1, c_2, \dots, c_k$ con $i \leq idx(c_1) < idx(c_2) < \cdots < idx(c_k) \leq j$, tali che i caratteri $c_1, c_2, c_3, \dots, c_k$ possano essere nuovamente disposti sotto forma di palindromo. Due sottoinsiemi di questo tipo sono considerati differenti se i loro caratteri vengono da indici differenti dell'array originario. Si trascurino i sottoinsiemi di cardinalità unitaria.

Ad esempio:

$$i = 2, j = 10$$

s = "Ieri Anna è andata a casa di Bob"

L'operazione $[i, j]$ ci darà come risultato 1, dato che abbiamo soltanto un possibile palindromo nell'intervallo considerato.