OLIMPIADI DELL'INFORMATIVA 2019-2020

SELEZIONE SCOLASTICA

Soluzioni

Indice

1 Esercizi a carattere Logico-Matematico

1.1 Esercizio N° 1

Questo tipo di esercizio richiede che venga predisposto un sistema basandosi sulle indicazioni del testamento, indichiamo quindi con m l'importo che spetta a Martina mentre con v quello che spetta a Valerio.

Dalla frase

> "Vorrei che il doppio di quanto spetti a Martina sia pari al triplo di quanto spetti a Valerio."

è possibile ricavare la seguente equazione

$$2m = 3v \tag{1}$$

Successivamente ci viene inoltre detto

> "vorrei inoltre che il doppio di quanto spetti a Valerio, sommato con il triplo di quanto spetti a Martina, sia pari a 13.000 euro."

che ci permette di ottenere

$$2v + 3m = 13000 \tag{2}$$

Visto che dobbiamo fare in modo che valgano entrambe le equazioni, abbiamo bisogno di metterle a sistema

$$\begin{cases} 2m = 3v \\ 2v + 3m = 13000 \end{cases} \tag{3}$$

Da questo punto in poi è possibile proseguire con una tecnica risolutiva a piacere, ad esempio possiamo procedere per sostituzione

$$\begin{cases} m = \frac{3}{2}v \\ 2v + 3(\frac{3}{2}v) = 13000 \end{cases} \tag{4} \qquad \begin{cases} m = \frac{3}{2}v \\ 2v + \frac{9}{2}v = 13000 \end{cases} \tag{5} \qquad \begin{cases} m = \frac{3}{2}v \\ 4v + 9v = 26000 \end{cases} \tag{6}$$

$$\begin{cases} m = \frac{3}{2}v \\ 13v = 26000 \end{cases} \tag{7} \qquad \begin{cases} m = \frac{3}{2}v \\ v = 2000 \end{cases} \tag{8} \qquad \begin{cases} m = 3000 \\ v = 2000 \end{cases} \tag{9}$$

Dunque il risultato finale è **VALTOT = 5000**

1.2 Esercizio N° 2

In questo caso c'è poco da dire, quello che serve è rimanere tranquilli e analizzare la situazione

1. Immaginiamo di estrarre al primo tentativo un calzino Bianco (potete fare lo stesso ragionamento con uno Nero)
2. Ora, potremmo essere particolarmente fortunati ed estrarre al secondo tentativo un calzino Bianco (ed in quel caso avremmo una coppia), tuttavia immaginiamo di estrarre un calzino Nero
3. A questo punto, indipendentemente dal calzino che estrarremo, avremo per forza di cose una coppia

Per questo motivo **NUMCALZINI = 3**.

Notate che, supponendo di avere k tipi di calzini differenti, nel caso peggiore dobbiamo fare $k + 1$ estrazioni per avere la certezza di poter avere una coppia.

1.3 Esercizio N° 3

Questo esercizio, come molti, punta a confondervi sparando un sacco di informazioni. L'idea di base in ogni caso è la seguente:

- Avete un insieme A di n elementi (in questo caso particolare 12)
- il vostro obbiettivo è quello di contare il numero di k coppie di insiemi A_1 e A_2 tali che
  - $\mid A_1 \mid = \mid A_2 \mid$, che vuol dire che i due insiemi devono possedere la stessa cardinalità (cioè devono avere lo stesso numero di elementi)
  - $A_1 \cup A_2 = A$, cioè l'unione degli elementi dei due insiemi deve fare A, ciò significa che dobbiamo utilizzare tutti gli elementi per fare le due coppie di insiemi
  - $A_1 \cap A_2 = \emptyset$, cioè l'intersezione due due insiemi deve fare l'insieme vuoto, il che vuol dire che non possiamo utilizzare un elemento presente in A_1 anche in A_2 e viceversa
  - $somma(A_1) = somma(A_2)$, quindi la somma degli elementi in A_1 deve essere uguale alla somma degli elementi in A_2

Date queste premesse possiamo cominciare a lavorare: il punto è, siamo sicuri che è davvero possibile risolvere un esercizio di questo tipo? Data la difficoltà della richiesta ho deciso di cominciare ad analizzare l'esercizio facendo una serie di controlli banali:

1. Se davvero è possibile suddividere i l'insieme A in due insiemi tali che $\mid A_1 \mid = \mid A_2 \mid$, allora devono per forza esserci un numero pari di elementi. Sfortunatamente, il numero di elementi totali è 12: ci è andata male, ma possiamo andare avanti.
2. Visto che $somma(A_1) = somma(A_2)$, possono verificarsi solamente due casi:
   - $somma(A_1) = somma(A_2)$ corrisponde ad un numero dispari
   - $somma(A_1) = somma(A_2)$ corrisponde ad un numero pari

   Tuttavia, in ogni caso $somma(A_1)+somma(A_2)$ produrrà un numero pari, ciò vuol dire che $somma(A)$ dovrà essere un numero pari. A quanto pare stavolta la fortuna e l'intuizione sono dalla nostra perchè la somma complessiva degli elementi è 497

Dunque per questo motivo non è possibile separare il nostro insieme in due in modo che siano rispettate le condizioni elencate, per questo motivo la risposta corretta è la **(c)**, cioè 0 coppie.

1.3.1 Per chi volesse approfondire

Mentre leggevo il problema ho immaginato che non poteste mettervi ad elencare tutte le possibili coppie in quanto, il problema proposto, non è altro che una riedizione ancora più specifica (e quindi difficile) di un problema *NP-Completo* (che per i non addetti ai lavori vuol dire che potete mettervi a piangere prima di trovare una soluzione ben precisa, oppure che se la trovate vincete un sacco di soldi) che prende il nome di Partition, di cui lascio il link a chi fosse interessato.

1.3.2 Perchè la somma deve fare un numero pari?

Proviamo a generalizzare il problema e ad analizzarlo sotto l'aspetto matematico: supponiamo di avere due numeri p e q, entrambi pari o dispari (questo perchè sappiamo che $somma(A_1) = somma(A_2)$)

- Se p e q sono entrambi pari allora possiamo scrivere che, $\forall i, j : 0 \leq i \leq j \leq \infty$ (in parole povere possiamo scegliere due numeri i e j a scelta compresi tra 0 e infinito)
  - $p = 2i$ e $q = 2j$, in questo modo siamo certi che p e q siano effettivamente dei numeri pari
  - $p + q = 2i + 2j$
  - $(p + q) \texttt{ mod } 2 = (p \texttt{ mod } 2 + q \texttt{ mod } 2) \texttt{ mod } 2 = (2i \texttt{ mod } 2 + 2j \texttt{ mod } 2) \texttt{ mod } 2 = 0 \texttt{ mod } 2 = 0$

  Per due generici p e q pari dunque la loro somma dovrà per forza essere un numero pari, in quanto abbiamo appena dimostrato che la loro somma modulo 2 restituirà 0 (cioè il resto della loro divisione sarà esattamente 0 e quindi sarà divisibile per 2, definizione di numero pari). Vi faccio notare la proprietà dell'operatore modulo (che avremo modo di rivedere più avanti) e cioè che $(A + B) \texttt{ mod } M = (A \texttt{ mod } M + B \texttt{ mod } M) \texttt{ mod } M$

- Se invece p e q sono entrambi dispari allora possiamo scrivere che, $\forall i, j : 0 \leq i \leq j \leq \infty$
  - $p = 2i + 1$ e $q = 2j + 1$, in questo modo siamo certi che p e q siano effettivamente dei numeri dispari
  - $p + q = 2i + 1 + 2j + 1 = 2i + 2j + 2$
  - $(p + q) \texttt{ mod } 2 = (p \texttt{ mod } 2 + q \texttt{ mod } 2) \texttt{ mod } 2 =$
    $(2i \texttt{ mod } 2 + 2j \texttt{ mod } 2 + 2 \texttt{ mod } 2) \texttt{ mod } 2 = 0 \texttt{ mod } 2 = 0$

  Anche in questo caso possiamo osservare che la loro somma da origine ad un numero pari

1.4 Esercizio N° 4

In questo caso dobbiamo ricorrere alla cara amica probabilità per poter capire effettivamente quanti sono i numeri *palizeri* compresi tra 10^3 e 10^5 (estremi esclusi). Prima di tutto capiamo l'effettivo range di valori: possiamo utilizzare tutti i numeri $k : 1001 \leq k \leq 99999$, tuttavia, visto che i numeri tra 1001 e 9999 hanno un numero pari di cifre, non rispettano la definizione di palizeri. Per questo motivo il range a cui siamo veramente interessati è $10000 \leq k \leq 99999$.

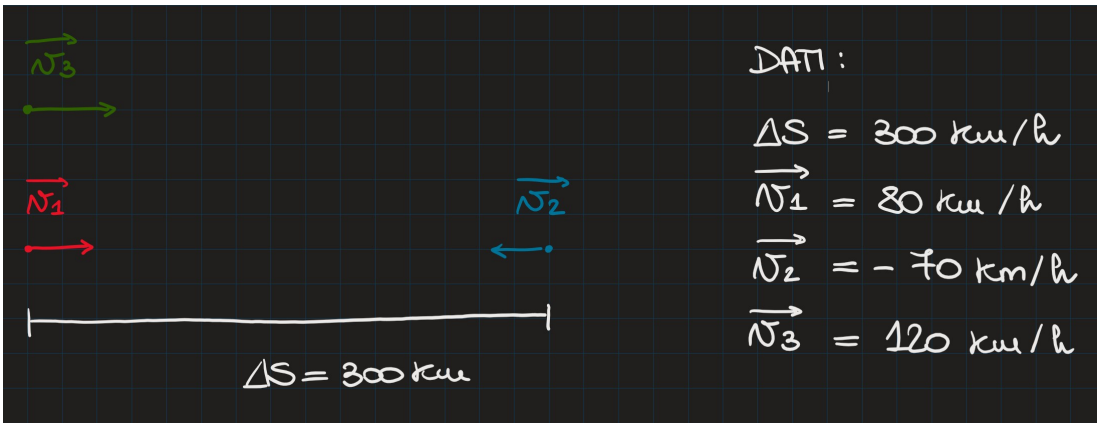
Cominciamo con la parte di probabilità forti del fatto che ci interessano solamente i casi per i numeri di 5 cifre

1. Per la prima cifra (quella più significativo, cioè più a sinistra), possiamo utilizzare un numero compreso tra 1 e 9, in quanto, se utilizzassimo lo 0, otterremmo un numero più piccolo di 5 cifre. Per questo motivo abbiamo *9 possibilità*
2. Per la seconda cifra invece non abbiamo vincoli particolari, in questo caso infatti possiamo scegliere un numero qualunque compreso tra 0 e 9 e quindi abbiamo un totale di *10 possibilità*
3. Per la terza cifra invece non abbiamo scelta: dobbiamo per forza avere il nostro zero in posizione centrale, abbiamo dunque *1 possibilità*
4. Per la quarta cifra abbiamo la necessità di rispettare la condizione di numero palindromo, cioè dobbiamo scegliere la stessa identica cifra che si trova in seconda posizione, dunque abbiamo soltanto *1 possibilità*
5. Per la quinta cifra dobbiamo rispettare gli stessi identici vincoli della cifra precedente, e quindi anche in questo caso avremo *1 possibilità*

Mettendo assieme quanto trovato fino ad ora possiamo dire che il numero totale di possibilità è pari a $9 \times 10 \times 1 \times 1 \times 1 = 90$, la risposta giusta in questo caso è dunque la **(b)**.

1.5 Esercizio N° 5

Dulcis in fundo abbiamo un problema di "fisica", iniziamo innanzitutto disegnando il sistema di riferimento per avere un'idea più chiara della situazione.



Indichiamo con v_1 la velocità della prima locomotiva, v_2 la velocità della seconda locomotiva (opposta a quella della prima) ed infine v_3 quella del colibrì (concorde con quella della prima locomotiva).

La prima cosa che dobbiamo chiederci è, quando (in che istante t) si scontreranno le due locomotive? Per poter rispondere a questa domanda dobbiamo innanzitutto identificare che il moto che trattiamo è di tipo Uniforme, cioè non vi sono accelerazioni e, per questo motivo, possiamo sapere la posizione x all'istante t' di un corpo che viaggia a velocità v partendo dalla posizione x_0, semplicemente facendo

$$x = x_0 + v * t' \tag{10}$$

Il nostro obbiettivo è calcolare l'istante in cui le due locomotive impatteranno: per fare ciò, possiamo considerare che, al momento dell'impatto, la loro posizione sarà la stessa. Facendo questa assunzione possiamo dunque scrivere la formula come segue

$$x_{10} + v_1 * t = x_{20} + v_2 * t \tag{11}$$

Dove x_{10} e x_{20} indicano rispettivamente le posizioni iniziali di x_1 e x_2: per questo motivo la formula può essere dunque riscritta come segue

$$v_1 * t = \Delta S + v_2 * t$$

$$v_1 * t - v_2 * t = \Delta S$$

$$t * (v_1 - v_2) = \Delta S$$

$$t = \frac{\Delta S}{(v_1 - v_2)}$$

Sostituendo a questo punto (e fidatevi, conviene sempre farlo a formula completa) i numeri, otteniamo che

$$t = \frac{300 \text{ Km}}{(80 - (-70)) \text{ Km/h}} = \frac{300 \text{ Km}}{150 \text{ Km/h}} = 2 \text{ h} \tag{12}$$

Sapendo che i due treni si scontreranno tra esattamente 2 ore e che la velocità del colibrì è, sempre e comunque, 120 Km/h, la distanza percorsa in totale dal colibrì sarà pari a

$$\Delta S' = v_3 * t = 120 \text{ Km/h } * 2 \text{ h } = 240 \text{ Km} \tag{13}$$

Notare che è inutile calcolare le distanze parziali che il colibrì percorre tra una locomotiva e un'altra perchè, complessivamente, percorrerà sempre la stessa distanza sapendo che lo scontro avverrà tra due ore. La risposta giusta in questo caso è la **(c)**.

2 Esercizi di Programmazione

2.1 Esercizio N° 6

Con il codice davanti e solo dopo aver dato un'occhiata al suo funzionamento, analizziamo le risposte

(a) La prima affermazione fa particolare riferimento a riga 14 del codice, dove viene calcolata la distanza tra `par1` e `par2` e viene controllato se le coppie sono contenute in un cerchio di raggio `par3`; tuttavia possiamo osservare che la funzione banalmente ritorna 1 anche per i casi espressi a riga 4 e 10, differenti da quello illustrato.

(b) Se chiamiamo la funzione sulla terna (1, 0, 3), possiamo osservare che il primo if (riga 1) risulta essere falso, per cui consideriamo la condizione presente alla riga 8: essendo che è vera analizziamo anche quella della riga successiva. Visto che banalmente 1 < 3, ritorniamo dunque 1. Questo caso dovrebbe chiarire eventuali dubbi su quale dovrebbe effettivamente essere la risposta corretta

(c) Affermazioni di questo tipo potrebbero essere affrontate con la matematica alla mano ma... a volte trovare un controesempio andando a tentativi può non essere una cattiva idea, ad esempio

- se scegliamo $a = 0$, osserviamo che la terna diventa (0, 0, 0) e quindi, eseguiamo il `return 0` di riga 4
- per $a = 1$, la terna che otteniamo è (1, 2, 3), per cui dobbiamo eseguire il controllo a riga 14. Essendo che $\sqrt{1+4} = \sqrt{5} < 3 = \sqrt{9}$, allora ritorneremo 1, contraddicendo per l'appunto l'affermazione proposta

(d) Quest'ultima affermazione non ha particolarmente senso perchè tutti i casi sono opportunamente gestiti, dunque non vi è necessità di fare quanto richiesto

2.2 Esercizio N° 7

In questo caso possiamo fare poco se non analizzare completamente il codice per capirne il funzionamento. Notiamo che nel vettore che ci è chiesto di analizzare vi sono 8 elementi e, quindi il `while` più esterno ciclerà su tutti gli elementi da 0 fino ad `n-1`, nessuno escluso.

Possiamo osservare che all'inizio di ogni iterazione, salviamo il valore dell'elemento i-esimo all'interno della variabile b, e inizializziamo la variabile c a 0. Successivamente entriamo in un `while` se e solo se $b > 1$, quindi se il numero che abbiamo precedentemente salvato rispetta questa condizione. La prossima istruzione prevede di raddoppiare il valore di una variabile a inizializzata ad 1, fino a quando non supera il valore di b; finito ciò sottraiamo il valore di $a/2$ a b e lo sommiamo invece a c. Tutto ciò, come detto precedentemente, si ripeterà fino a quando b sarà maggiore di 1, cioè fino a quando non rimarremo con la sua parte decimale oppure con zero. Per poter essere sicuri di ciò proviamo ad osservare che cosa fa il codice ad esempio con il primo numero del vettore

1. $b = v[0] = 3.45$
2. $c = 0$
3. $b > 1$?
   (a) Se sì, imposta $a = 1$
   (b) Raddoppia a fino a quando non è strettamente maggiore di b
   (c) quando avremo $a = 4$, imposteremo
      - $b = 3.45 - 4/2 = 1.45$
      - $c = 0 + 4/2 = 2$
   (d) Ripetiamo da 3 ottenendo in questo caso $a = 2$ e quindi
      - $b = 1.45 - 2/1 = 0.45$
      - $c = 2 + 2/1 = 3$
   (e) verifichiamo la condizione del punto 3 e usciamo dal ciclo interno
4. verifichiamo se l'elemento si trova in posizione pari oppure dispari
   - se si trova in posizione pari, assegniamo c a $v[i]$
   - in caso contrario, assegniamo invece $c + 1$
   - In questo caso avremo che $v[0] = 3$
5. incrementiamo i e passiamo ad analizzare il prossimo elemento

Fondamentalmente andremo dunque a salvarci solamente la parte intera del numero che stiamo considerando e, nel caso in cui l'indice del numero fosse dispari, salveremo invece la parte intera + 1.

Dato v = {3.45, 5.67, 8.92, 2.12, 7.33, 8.21, 4.21, 9.03} e n = 8
otterremo v = {3.0, 6.0, 8.0, 3.0, 7.0, 9.0, 4.0, 10.0}, per cui la risposta corretta è la **(a)**.

2.3 Esercizio N° 8

Partiamo dal main osservando che viene allocato un vettore di 10 float, inizializzando ogni $v[i] = 25/4 = 6.25$. A questo punto viene eseguita la funzione $pp(vet, 10, 5)$.

La funzione possiamo osservare che si compone di un ciclo che viene eseguito 5 volte e, ogni volta si va a modificare l'elemento in posizione $v[dim-i-1]$ ponendolo uguale a 0: questo significa che verranno modificati gli indici compresi tra $10-1 = 9$ e $10-5 = 5$. Il vettore dunque risulterà così costituito alla fine della procedura

$vet = \{6.25, 6.25, 6.25, 6.25, 6.25, 0, 0, 0, 0, 0\}$, per cui la risposta corretta è la **(c)**.

2.4 Esercizio N° 9

La funzione $stampa()$ crea una matrice mat di dimensioni 5×5 inizializzando ogni cella al valore 0. Il passo successivo è quello di, scorrendo ogni valore, assegnare il valore a a quelle celle della matrice $mat[i][j]$ tali per cui

- l'indice delle righe identifica una posizione pari e l'indice della colonna fa riferimento alla seconda posizione (riga 18 del codice)
- l'indice delle righe identifica una posizione dispari mentre l'indice della colonna fa riferimento alla prima oppure alla terza posizione

Infine viene stampata a video la matrice completa, mostrando uno spazio nel caso in cui la posizione $mat[i][j] == 0$, altrimenti *

La matrice totale è dunque

	0	1	2	3	4
0			*		
1		*		*	
2			*		
3		*		*	
4			*		

Dunque la risposta corretta in questo caso è la **(d)**.

2.5 Esercizio N° 10

Il programma in questione inizializza una variabile c a 0 e due vettori v e w di interi per poi, considerando uno ad uno gli elementi del primo, confrontarli con quelli del secondo.

Possiamo notare che, ogni volta che $v[i] == w[j]$, il valore di c viene incrementato di uno: il compito di c è dunque quello di contare le occorrenze dei numeri in $v[i]$ che sono presenti anche in $v[j]$.

Una volta effettuata questa operazione, vengono eseguiti

- il rapporto tra c e il numero di elementi di v, moltiplicando infine il risultato per cento ed ottenendo dunque la *percentuale di elementi in comune tra* v *e* w *rispetto agli elementi presenti in* v
- il rapporto tra c e il numero di elementi di w, moltiplicando infine il risultato per cento ed ottenendo dunque la *percentuale di elementi in comune tra* v *e* w *rispetto agli elementi presenti in* w

Per poter analizzare le affermazioni e poter scegliere quella giusta notiamo innanzitutto che, dati v e w, $c = 4$, in quanto in comune tra i due insiemi vi sono i numeri {1, 4, 32, 6}

(a) Seppur $c/10 * 100 = 4 * 10 = 40$, sia effettivamente corretto, $4/14 * 100 = 2/7 * 100 = 200/7 = 28.57 \neq 35.71$, dunque questa affermazione è sbagliata

(b) Come detto precedentemente, questa affermazione è corretta in quanto il programma calcola la percentuale di elementi in comune tra v e w (cioè $v \cap w$) sul totale degli elementi del primo e del secondo insieme

(c) Ovviamente l'unione dei due insiemi da origine ad una percentuale completamente diversa e, tra le varie cose, maggiore di 100

(d) Per gli stessi motivi della domanda (a), questa affermazione è sbagliata

2.6 Esercizio N° 11

In questo esercizio ci viene richiesto di conteggiare il numero di chiamate ricorsive delle due funzioni $rec1$ e $rec2$ supponendo che venga fatta la chiamata $rec(4, 6)$ (che dobbiamo tra le altre cose conteggiare nel nostro conteggio).

1. **NUMREC1 = 1**, la prima è quella che fa partire la ricorsione e abbiamo che $a = 4$ e $b = 6$. In questo caso $c = 4 * 6 = 24$, per questo motivo abbiamo che $c \neq 0$ e quindi dobbiamo proseguire la ricorsione chiamando $rec(4, 24)$

2. **NUMREC2 = 1**, durante la prima chiamata ricorsiva a $rec2$ abbiamo che $a = 4$ e $b = 24$ per cui $d = 24 - 3 * 4 = 24 - 12 = 12$. Visto che $d\%27 \neq 0$, dobbiamo eseguire la chiamata ricorsiva $rec1(24, 12)$

3. **NUMREC1 = 2**, al secondo giro si ha che $c = 12 * 24 = 12 * 12 * 2 = 144 * 2 = 288$ ed ovviamente anche in questo caso non possiamo fermare la ricorsione ma dobbiamo chiamare $rec(24, 288)$

4. **NUMREC2 = 2**, calcoliamo $d = 288 - 24 * 3 = 288 - 72 = 216$. Visto che i numeri sono un po' grandi a questo punto un metodo rapido per poter verificare se il resto della divisione per 27 è pari a 0 è quello di dividere per tre volte 216 per 3 che, per quanto banale, vi semplifica il lavoro (ricordo anche che un numero è divisibile per 3 se e solo se la somma delle cifre restituisce un numero multiplo di 3). Visto che $((216/3)/3)/3 = (72/3)/3 = 24/3 = 8$, allora il resto della divisione per 27 è 0 dunque possiamo concludere la ricorsione eseguendo `return 0`.

La risposta corretta in questo caso sarà dunque **NUMREC1 = 2** e **NUMREC2 = 2**.

2.7 Esercizio N° 12

In questo caso direi che la prima cosa da fare è provare a mettere per iscritto le linee di codice di cui conosciamo già la posizione

```
1
2       if(n==1){
3           return 1;
4       }
5
6   }
7
8
9       risp = 0;
10
11          return 1;
12      }
13      int i;
14
15
16          risp = risp + f2(n-1);
17
18      }
19
20  }
```

Visto che conosciamo il fatto che il file contenesse due funzioni, è possibile partire dal riordinare le dichiarazioni: la `riga 6` ci aiuta a capire che quello è il punto dove finisce la prima funzione mentre la `riga 16` ci può essere d'aiuto per capire che, dalla riga 7 in poi, avremo la funzione f2 (potrebbe anche darsi il caso che le due funzioni si chiamino vicendevolmente ma non complichiamoci troppo la vita per il momento).

La `riga 9` ci suggerisce indubbiamente che la variabile $risp$ ha bisogno di una dichiarazione prima di essere utilizzata, mentre dalla `riga 13` e dalla presenza di un ciclo che sfrutta la variabile i dichiarata, abbiamo sicuramente bisogno di mettere un'inizializzazione.

Osserviamo i cambiamenti che abbiamo fatto fino a questo punto

```
1  int f1(int n){
2      if(n==1){
3          return 1;
4      }
5
6  }
7  int f2(int n){
8      int risp;
9      risp = 0;
10
11         return 1;
12     }
13     int i;
14     i = 1;
15
16         risp = risp + f2(n-1);
17
18     }
19
20 }
```

Visto che alla prima funzione manca soltanto un'istruzione possiamo senza problemi immaginare che alla `riga 5` vada inserito il costrutto `return` $n * f1(n-1)$, ad avvalorare la nostra ipotesi possiamo infatti osservare che $f1$ si occupa di calcolare $n!$, cioè il numero di permutazioni possibili di n oggetti.

Alla funzione f2 mancano diverse righe di codice, tuttavia, osservando l'indentazione, possiamo notare che le righe 10 e 15, prevederanno molto probabilmente l'utilizzo di un costrutto come un ciclo o un if statement. Il fatto che vi sia un while e una condizione di incremento non ancora utilizzati, fa pensare che questi debbano essere inseriti rispettivamente a `riga 15` e `riga 17`. Per esclusione dunque (e per indentazione) andremo a posizionare l'if a `riga 10`.

A questo punto, rimasti solo con l'istruzione `return` $risp$, non dobbiamo fare altro che inserirla a `riga 19`.

```
1  int f1(int n){
2      if(n==1){
3          return 1;
4      }
5      return n*f1(n-1);
6  }
7  int f2(int n){
8      int risp;
9      risp = 0;
10     if(n==1){
11         return 1;
12     }
13     int i;
14     i = 1;
15     while(i<=n){
16         risp = risp + f2(n-1);
17         i=i+1;
18     }
19     return risp;
20 }
```

Per poterci convincere al massimo, possiamo osservare che la funzione $f2$ non fa altro che chiamare n volte $f2(n-1)$, che corrisponde letteralmente alla chiamata presente a `riga 5`: modo che seppur contorto, assolutamente analogo.

La soluzione proposta è dunque

A = 10, E = 7, G = 1, H = 5, N = 8, P = 15, R = 14, S = 17, U = 19

3 Esercizi a Carattere Algoritmico

3.1 Esercizio N° 13

Mettiamo le due frasi a confronto

- $w_1 =$ Olimpiadi Italiane di Informatica
- $w_2 =$ Olinpiadi Italianer de Informatia

Sappiamo, dal problema, che la edit distance riguarda sia la modifica che l'inserimento di nuove parole in una qualsiasi delle due frasi w_1, w_2 che stiamo analizzando. Banalmente possiamo osservare che Olimpiadi è stato modificato in Olinpiadi, per questo l'edit distance di quella particolare parola è 1. La stessa cosa si può osservare dalla terza parola, e quindi tra di e de.

Analizzando invece la seconda parola, è possibile notare che Italianer ha una lettera in più rispetto a Italiane, dunque la loro edit distance è pari a 2.

Vista la semplicità dell'esercizio non mi dilungo oltre, ma vi faccio notare che la proprietà vale in entrambi i sensi e quindi che l'edit distance tra la parola Informatica e Informatia è sempre di 2, nonostante a differenza del caso precedente la prima parola appartenga a w_1 e la seconda a w_2.

Complessivamente **DIST = 6**.

3.2 Esercizio N° 14

Per questo tipo di esercizi è possibile concentrarsi sul trovare subito un candidato ideale oppure, come abbiamo già potuto vedere precedentemente, andare per tentativi. In questo caso, possiamo scegliere 2 sia perchè effettivamente vi sono già due elementi presenti nell'insieme A di cui 2 è divisore, oppure perchè non possiamo scegliere un divisore minore di 2 (e ci viene richiesto di trovare il più piccolo divisore c per cui è rispettata la relazione).

In qualsiasi caso per $c = 2$ possiamo osservare che

$$D(c, A) = \mid \{2, 8\} \mid > \mid \{42\} \mid = D(c, B) \tag{14}$$

per cui la relazione è rispettata.

La risposta corretta è dunque **C = 2**.

N.B. con $\mid \{2, 8\} \mid$ si fa riferimento alla *cardinalità* dell'insieme, cioè al numero di elementi contenuti all'interno di tale insieme (in questo caso particolare $\mid \{2, 8\} \mid = 2$)

3.3 Esercizio N° 15

Vi ricordo innanzitutto alcune proprietà dell'operatore *modulo*

- il risultato restituito è sempre compreso tra 0 e $N-1$, in quanto il resto della divisione per N può essere al massimo $N-1$ (altrimenti la divisione non è stata effettuata correttamente)
- $(A + B) \texttt{ mod } M = (A \texttt{ mod } M + B \texttt{ mod } M) \texttt{ mod } M$

- $(A * B) \texttt{ mod } M = (A \texttt{ mod } M * B \texttt{ mod } M) \texttt{ mod } M$

L'operazione che dobbiamo applicare è $2x \texttt{ mod } 7$, che possiamo scrivere anche come $(2 \texttt{ mod } 7 \cdot x \texttt{ mod } 7) \texttt{ mod } 7$ e che ci potrà tornare utili per calcoli più complessi.

Analizziamo le affermazioni

(a) Se in **BF1** fosse presente il numero 4, dovrebbe valere che $BF1[8 \texttt{ mod } 7] = 1$, tuttavia osserviamo che $BF1[1] \neq 1$, dunque ciò non è possibile. Ci tengo inoltre a sottolineare che questa frase non è completamente corretta in quanto, seppur ci fossimo trovati nel caso per cui $BF1[1] = 1$, non avremmo mai potuto avere la certezza che il calcolo facesse riferimento espressamente al numero 4, infatti $\forall i : 0 \leq i \leq \infty$ (quindi per un qualsiasi i compreso tra 0 ed infinito), $(2 \cdot (4 + 7i)) \texttt{ mod } 7 = 1$, il che vuol dire che otterremo lo stesso risultato per 4, 11, 18, ...

(b) Se in **BF1** non fosse presente 19, allora dovrebbe valere che $BF1[(19 * 2) \texttt{ mod } 7] = BF1[10 \texttt{ mod } 7] = BF1[3] = 0$; osservando la tabella però siamo in grado di dire che ciò non è vero in quanto $BF1[3] = 1$. Rifacendomi all'affermazione precedente, mi soffermo sul fatto che il numero 19 *potrebbe non essere presente nella tabella*, in quanto è possibile fare lo stesso giochetto di prima e trovare un altro numero che da lo stesso risultato (ad esempio 12); tuttavia questo non esclude il fatto che sia proprio 19 il numero incriminato.

(c) Se in **BF1** fosse presente 6, allora dovrebbe valere che $BF1[12 \texttt{ mod } 7] = BF1[5] = 1$, tuttavia $BF1[5] = 0$ per cui sicuramente 6 non è presente nella tabella.

(d) Se in **BF1** fosse presente 12, allora dovrebbe valere che $BF1[(12*2) \texttt{ mod } 7] = BF1[10 \texttt{ mod } 7] = BF1[3] = 1$; visto che $BF1[3] = 1$ e che tutte le altre affermazioni sono incorrette è possibile affermare con certezza che questa sia quella corretta.

3.4 Esercizio N° 16

Per poter capire qual è il numero minimo di mosse conviene concentrarsi sugli angoli della nostra matrice: l'unico modo per poter accendere queste luci è

- cliccare direttamente sull'angolo
- cliccare una cella adiacente (sopra, sotto, a destra o a sinistra) all'angolo

Provando a cliccare direttamente sugli angoli tuttavia non porterà a nessuna soluzione in quanto rimarrà spenta tutta la sezione centrale (quadrato 2x2) e, cliccando su una delle celle rimanenti, ne spegneremo alcune di quelle già accese, aumentando ancora di più il numero di mosse da dover fare.

Se invece cliccassimo una delle celle adiacenti agli angoli, potremmo effettivamente accorgerci che, in realtà il numero di mosse minime è pari a 4

	0	1	2	3
0	(1)	(4)	click(4)	(4)
1	click(1)	(1)	(4)	(3)
3	(1)	(2)	(3)	click(3)
4	(2)	click(2)	(2)	(3)

Cliccando per l'appunto nelle posizioni indicate da *click(i)*, verranno illuminate anche le celle adiacenti *(i)*. Inoltre, possiamo essere certi che la soluzione sia 4 mosse perchè, essendo che si illuminano al più sempre 5 celle (quella cliccata più le 4 adiacenti), non c'è modo di fare una soluzione con 3 mosse: la matrice contiene infatti $16 > 3 * 5$ celle.

NUMMOSSE = 4

3.5 Esercizio N° 17

L'esercizio richiede di generare una stringa al massimo lunga 13 caratteri tale che, tutte le stringhe proposte, siano *sottostringhe* di quella ottenuta (cioè A è sottostringa di B se e solo se B contiene una sottosequenza contigua di caratteri B' tale che $A = B'$).

In questo caso possiamo subito notare che la stringa 101, che deve essere sottostringa di w, in realtà è già una sottostringa ad esempio di $00\underline{101}$: questo significa che possiamo ignorarla tranquillamente.

Fatto ciò il problema si riduce banalmente ad ordinare le stringhe in modo da sfruttare le parti in comune e non ripeterle, cioè

- una parte di $1111\underline{001}$ è in comune anche con $\underline{001}01$
- lo stesso vale per $00\underline{101}$ e $\underline{101}0101$

Accorgendosi di ciò è possibile quindi dedurre che la stringa 1111001 dovrà essere utilizzata all'inizio, a seguire 00101 ed infine 1010101, ottenendo dunque

$$w = 1111001010101 \tag{15}$$

che è soluzione dell'esercizio in quanto $\underline{1111001}010101$, $1111\underline{001010}101$, $111100\underline{1010101}$ ed infine $11110\underline{1010}101$ (anche se in questo ultimo caso sono possibili diverse sottostringhe per 101)

3.6 Esercizio N° 18

La richiesta, a differenza del problema di un'edizione precedente, è quello di verificare *quale sia il percorso non corretto* tra i 4: il mio consiglio è quello di mettere tenere sott'occhio entrambe le facciate in modo da poter arrivare più velocemente possibile alla soluzione, in quanto il codice è molto spesso simile.

Utilizzando questa tecnica osserviamo subito che la seconda istruzione dell'alternativa C prevede che la tartaruga prosegua in avanti di l, tuttavia questo vorrebbe dire, visto che siamo nella condizione di *pennagiù*, disegnare al di fuori del tratto tratteggiato.

Il codice che non disegnerà correttamente la figura sarà dunque l'opzione **(c)**.

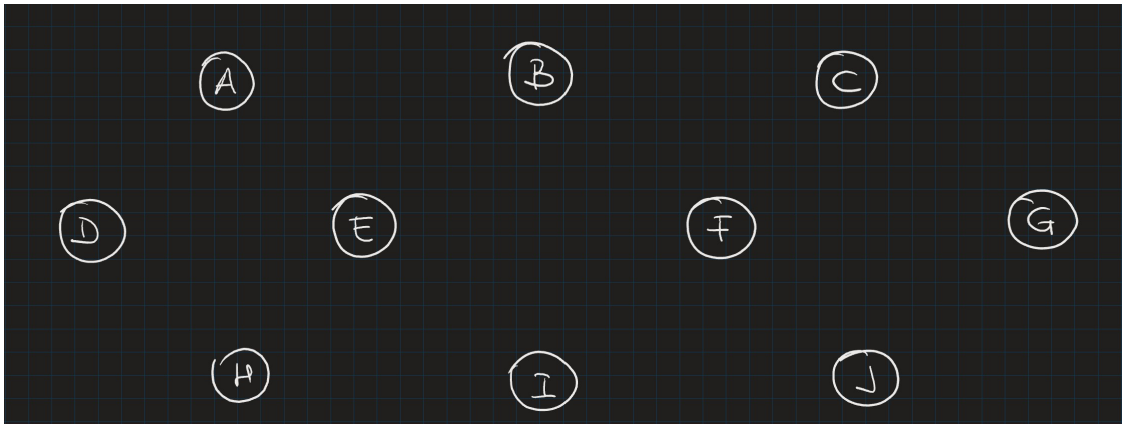
3.7 Esercizio N° 19

L'esercizio richiede che vengano identificati i 9 archi che permettono di connettere i 10 PC facendo in modo che il costo totale di queste connessioni sia minimo. Formalmente questo problema viene definito Minimum Spanning Tree o Albero di Copertura di Peso minimo ed esistono due algoritmi, rispettivamente di Kruskal e Prim, che permettono di individuarlo con relativa facilità.

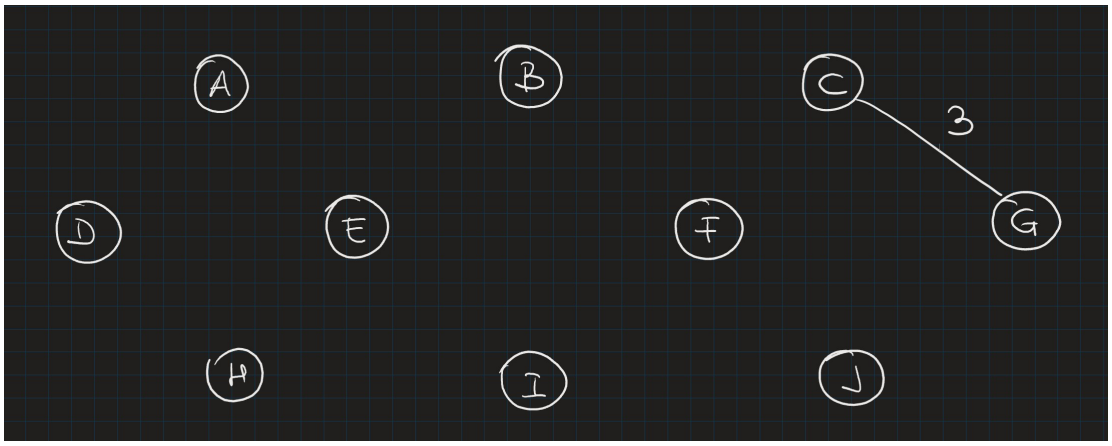
Nonostante le premesse non sembrino delle più rosee, l'idea alla base è relativamente facile ed intuitiva:

1. Prendete i nodi della vostra rete di PC (numerati rispettivamente da A fino a J) e scollegateli
2. considerate ora l'arco di peso minimo che connette tra di loro due nodi che non avete ancora connesso tra loro
3. connettete i due nodi
4. ripetete dal punto 2 fino a che non avete connesso tutti e dieci i nodi

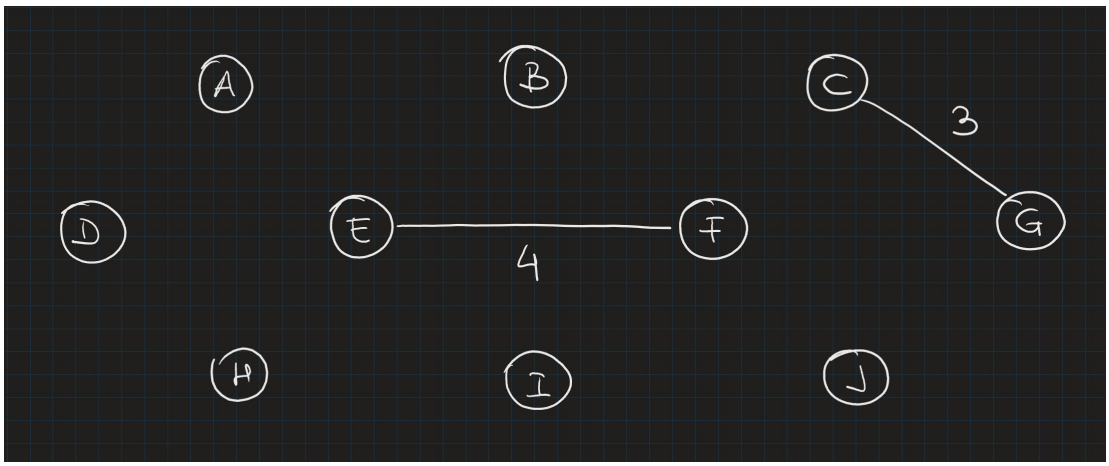
Per capire meglio facciamo la risoluzione passo passo dell'esercizio. Innanzitutto scolleghiamo tutti gli archi del nostro grafo



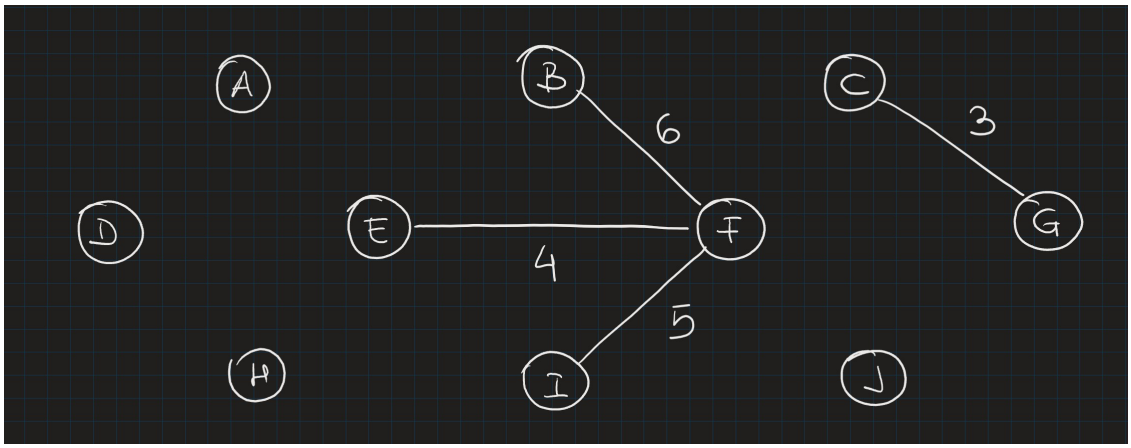
A questo punto selezioniamo l'arco di peso minimo all'interno del nostro grafo originale, in questo caso parliamo dell'arco con costo 3 che connette C e G: colleghiamo dunque i due nodi con tale arco.



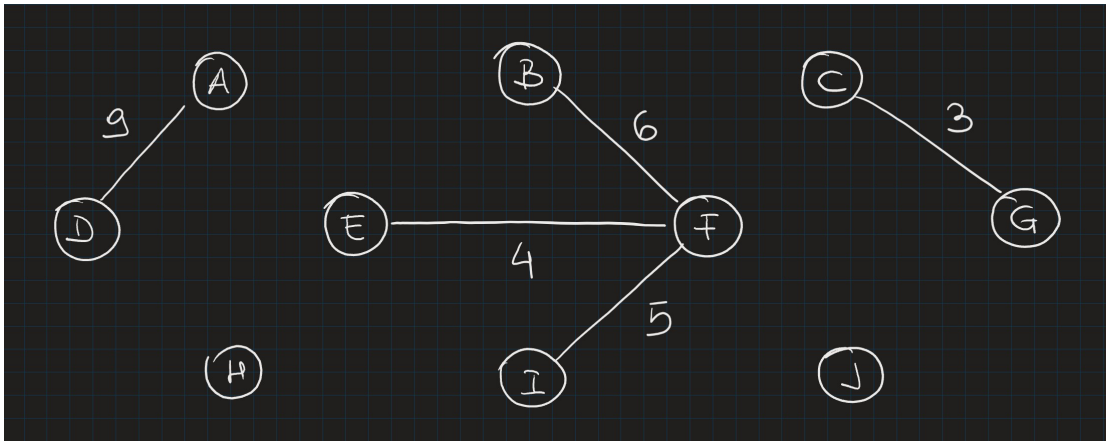
Proseguiamo selezionando il prossimo arco di peso minimo: in questo caso parliamo dell'arco di costo 4 che connette E ed F.



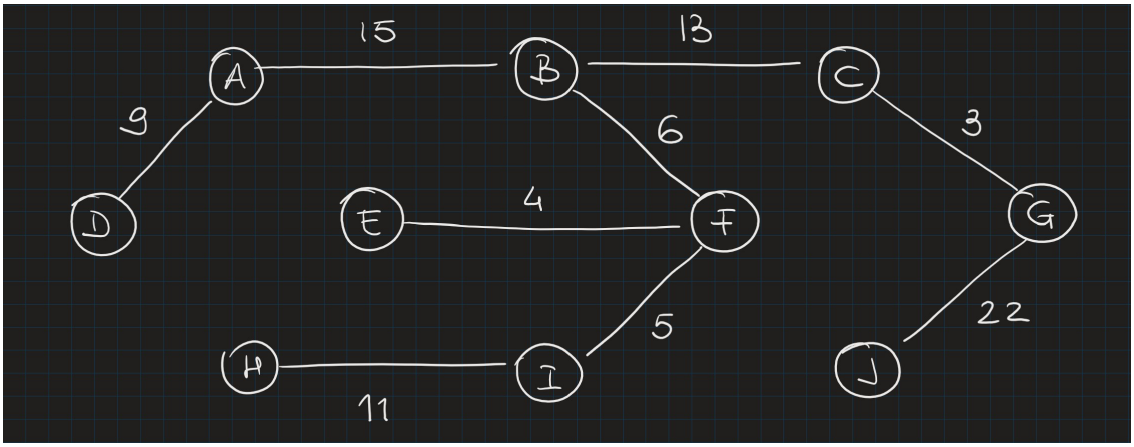
Capito il procedimento di base siamo già a metà dell'opera, tuttavia osserviamo con attenzione questa situazione



Se a questo punto selezionassimo infatti l'arco di peso minimo, andremmo a scegliere l'arco di costo 7 che connette il nodo E con I; tuttavia, abbiamo già un cammino (cioè possiamo attraversare una determinata sequenza di archi per poter raggiungere un nodo X partendo dal nodo Y) che permette di raggiungere sia I che E. Per questo motivo dobbiamo ignorare tale arco e passare al prossimo arco di peso minimo: in questo caso, ignorando l'arco citato prima, facciamo riferimento all'arco di costo 9 che connette A e D.



Proseguendo in questo modo dovreste ad arrivare ad una soluzione di questo tipo



Il costo totale di questa struttura sarà dunque pari alla somma degli archi, di conseguenza **TOTCOSTO = 88**

3.8 Esercizio N° 20

Per risolvere questo esercizio dobbiamo selezionare correttamente la riga e la colonna che ci viene indicata nei due casi:

A Nel primo caso è la squadra S ad iniziare e ci viene richiesto di selezionare la riga dove è massimo il numero minimo di filetti di salmone nel piatto, un modo particolarmente contorto per dire che dobbiamo selezionare la riga dove, nel caso in cui si scelga il piatto contenente meno salmone, questo conterrà un numero di filetti di salmone maggiore rispetto al quello (minimo) delle altre righe. Possiamo dunque osservare che tale scelta non può che ricadere su **R1**, mentre la colonna selezionata dalla squadra T sarà **C2** in quanto conterrà la quantità di tonno maggiore tra i piatti sulla riga R1.

B Applicando lo stesso ragionamento per il secondo caso, la squadra T, iniziando stavolta per prima, si ritroverà a scegliere per prima la colonna **C2** e la squadra S la riga **R3**.

La soluzione all'esercizio è dunque

RA = 1, CA = 2
RB = 3, CB = 2