Codice segreto

Alice e Bob devono comunicare in segreto. Bob se ne esce fuori con il seguente codice crittografico: la lettera "A" corrisponde a 1, la lettera "B" corrisponde a 2, la lettera "C" corrisponde a 3 e così via fino a "Z" che corrisponde a 26. Ad esempio, la parola "ACME" è tradotta con "13135". Ma Alice si accorge subito di un problema: il codice non è univoco, ovvero "13135" può essere interpretato anche come "MACE", "MME" e "ACACE".

Scrivere un algoritmo che dato un codice numerico composto da n cifre rappresentato da un vettore T di interi, restituisca il numero di modi possibili in cui è possibile intepretare il codice. Se non è possibile interpretare il codice, l'algoritmo deve restituire 0.

Discutere correttezza e complessità dell'algoritmo proposto.

Somma di quadrati

Ogni intero positivo n può essere scritto come somma di quadrati di interi; ad esempio, $7 = 2^2 + 1^2 + 1^2 + 1^2$, mentre $13 = 3^2 + 2^2$. Ovviamente, esistono più modi per esprimere un numero come somma di quadrati; 13 può essere espresso anche come $2^2 + 2^2 + 2^2 + 1^2$.

Scrivere un algoritmo che, preso in input n, restituisce il **numero** minimo di quadrati la cui somma è pari ad n. Ad esempio, nel caso di 13, $3^2 + 2^2$ richiede 2 quadrati, mentre $2^2 + 2^2 + 2^2 + 1^2$ richiede 4 quadrati. Discuterne correttezza e complessità.

Somma di quadrati

Scrivere un algoritmo che, dato n, stampa **tutti** i modi possibili per esprimere n come somma di quadrati, discutendo correttezza e complessità; ad esempio, con $n = 13$, stamperà

$$1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2$$
$$2^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2$$
$$2^2 + 2^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2$$
$$2^2 + 2^2 + 2^2 + 1^2$$
$$3^2 + 1^2 + 1^2 + 1^2 + 1^2$$
$$3^2 + 2^2$$

Supersequenza comune minimale

Una stringa P è una supersequenza di una stringa T se T è una sottosequenza di P. Scrivere un algoritmo che restituisce la lunghezza della *supersequenza comune minimale* di due stringhe P, T, ovvero la più piccola supersequenza di entrambe le stringhe. Discutere correttezza e complessità dell'algoritmo proposto.

Esempio: L'unica supersequenza comune minimale di `AB` e `BC` è `ABC`, e la sua lunghezza è pari a 3.

Esempio: Esistono due supersequenze comuni minimali di `DAB` e `DCB`, ovvero `DACB` e `DCAB`, e la loro lunghezza è pari a 4.

High Line

La High Line è un parco lineare di New York realizzato su una sezione in disuso della ferrovia sopraelevata chiamata West Side Line.

- E' un rettilineo lungo L metri corredato da aiuole e piante. Lungo il parco, esistono n irrigatori.
- L'irrigatore i-esimo è collocato ad una distanza $D[i]$ dall'inizio della High Line e ha un raggio di azione pari a $R[i]$, ovvero innaffia la sezione di High Line che va da $D[i] - R[i]$ a $D[i] + R[i]$.

Il vostro compito è scrivere un algoritmo che restituisca il minimo numero di irrigatori che vanno attivati per innaffiare l'intera linea, oppure -1 se è impossibile innaffiare l'intera linea.
Discutere correttezza e complessità dell'algoritmo proposto.

Siete responsabile dei corsi di nuoto organizzati da una piscina pubblica. I corsi vengono gestiti settimanalmente.
Ogni corso dura un'ora. Ogni corso può avere al massimo 6 bambini.
Ogni corso ha un orario di inizio che può andare dalle 9 alle 18 (per un totale di dieci corsi al giorno).
Ogni bambino sceglie a quanti corsi partecipare alla settimana, ma questo numero non può essere superiore a 5 e non si può fare lezione più di due volte lo stesso giorno. Poiché i poveri bambini moderni sono stressati da mille altri corsi (calcio, musica, etc), ogni bambino può dare un certo numero di preferenze (ore in cui può partecipare ad un corso).
Ogni corso deve essere insegnato da un insegnante. Ogni insegnante può dare un certo numero di disponibilità (ore in cui può insegnare un corso).
Infine, ogni bambino che segue un corso paga 10 euro. Trovate un algortmo che di assegnamento bambini-corsi-insegnanti che massimizzi il guadagno, descrivendo per bene input, output. Discutere la complessità.

Stringhe primitive

Scrivere un algoritmo che, dati un insieme S contenente m stringhe dette *primitive* ed una stringa $X[1 \ldots n]$, conti in quanti modi diversi X è ottenibile dalla concatenazione di stringhe primitive.

Ad esempio, dato $S = \{\texttt{01}, \texttt{10}, \texttt{011}, \texttt{101}\}$:

- $X = \texttt{0111010101}$: ci sono 3 modi
  ($\texttt{011} - \texttt{10} - \texttt{10} - \texttt{101}$, $\texttt{011} - \texttt{10} - \texttt{101} - \texttt{01}$ e $\texttt{011} - \texttt{101} - \texttt{01} - \texttt{01}$)
- $X = \texttt{0110001}$: non c'è modo di ottenere tre 0 consecutivi

Per comodità, supponete che la lunghezza di una stringa s sia $|s|$ e di avere a disposizione una primitiva $\mathsf{check}(X, s, i)$ che ritorna vero se la stringa s è contenuta nella stringa X a partire dalla posizione i. Il costo della chiamata a $\mathsf{check}()$ è $O(|s|)$. Ad esempio, se $X = \texttt{1001}$ e $s = \texttt{00}$, $\mathsf{check}(X, s, 2)$ ritorna vero, per tutti gli altri indici i ritorna falso.

Good vs bad guys

Fra ogni coppia di wrestler professionisti, può esserci una rivalità oppure no. Per ragioni di marketing, è una buona idea dividere i wrestler professionisti in "buoni" e "cattivi".
Supponete di avere rappresentato i wrestler e le loro rivalità tramite un grafo $G = (V, E)$, dove $(u, v) \in E$ significa che il wrestler u è un rivale del wrestler v e viceversa.
Scrivere un algoritmo che restituisce **true** se è possibile suddividere i wrestler professionisti in due sottoinsiemi non vuoti ("buoni" e "cattivi"), non necessariamente della stessa dimensione, in modo tale che non ci siano rivalità all'intero dei due gruppi; **false** altrimenti.

Spoiler alert!