D20

Dati n dadi, con il dado i-esimo dotato di $F[i]$ facce numerate da 1 a $F[i]$, scrivere un algoritmo che restituisca il numero di modi diversi con cui è possibile ottenere una certa somma X sommando i valori di tutti i dadi.

- Ad esempio, avendo due dadi a quattro facce numerati da 1 a 4, il valore 7 è ottenibile in un solo modo non contando le possibili permutazioni: $3 + 4$.
- Avendo tre dadi sempre a 4 facce, il valore 6 è ottenibile in tre modi diversi non contando le possibili permutazioni: $1 + 1 + 4$, $1 + 2 + 3$, $2 + 2 + 2$.

Permutazioni: contatele oppure no, ma siatene consapevoli...

Sfilatino alla Nutella

Nella sagra di Hateville, è stato deciso di realizzare lo sfilatino alla Nutella più grande del mondo, lungo L centimetri. Ora si tratta di realizzare un altro record: il maggior numero di persone servite con lo stesso sfilatino. Alla sagra sono presenti n persone, dove la persona i-esima chiede un *segmento* di sfilatino lungo $V[i]$ centimetri; secondo il regolamento, ogni richiesta va servita esattamente, ovvero se la persona i verrà servita, riceverà il segmento richiesto. Tutte le lunghezze sono intere. Scrivere un algoritmo che restituisca il numero massimo di persone che possono essere servite con lo sfilatino. Non è necessario utilizzare tutto lo sfilatino.
Oltre a calcolare la complessità dell'algoritmo proposto, discutere anche la correttezza, menzionando la tecnica scelta e specificando bene perché tale tecnica può essere applicata in questo caso.

Ottimizza la somma

Supponete di avere in input un vettore di n interi positivi distinti $V[1 \dots n]$ e un valore W. Scrivere un algoritmo che:

1. restituisca il massimo valore $X = \sum_{i=1}^{n} x[i]V[i]$ tale che $X \leq W$ e ogni $x[i]$ è un intero non negativo;
2. stampi il vettore x.

Ad esempio, per $V[\,] = \{18, 3, 21, 9, 12, 24\}$ e $W = 17$, una possibile soluzione ottima è $x[\,] = \{0, 2, 0, 1, 0, 0\}$ da cui deriva $X = 15$.
Discutere correttezza e complessità.

Cubi

Siano dati n cubi. Su ogni faccia del cubo è presente una lettera dell'alfabeto. Ogni cubo può essere descritto da una stringa di 6 caratteri (e.g. "ABCDEF"), che rappresenta le 6 facce del cubo.
Sia data una parola costituita da $t \leq n$ caratteri; il vostro compito è descrivere un algoritmo che sia in grado di dire se è possibile comporre la parola utilizzando t degli n cubi, scegliendo *una* faccia per ognuno di essi. Calcolare la complessità dell'algoritmo risultante.

Ad esempio, supponete di avere i seguenti cubi: "ABCDEF", "GHIJKL", "AABBCC", "ISTUVW" e di voler comporre la parola IDEA.

Ballo di fine anno

Una scuola vuole organizzare un ballo di fine anno. Ci sono n maschi e m femmine. Ogni coppia di studenti (composta da un ragazzo ed una ragazza che intendono danzare insieme) ha dovuto registrarsi (altrimenti non avrebbero potuto danzare insieme). I regolamenti della scuola impongono che ogni data coppia non possa danzare insieme più di 3 volte. In più, ogni studente non può danzare più di 10 volte in totale. Potete assumere che il ballo duri abbastanza a lungo da permettere a tutti di completare le proprie danze, se le registrazioni lo permettono.

Trovare un algoritmo che, dato in input l'insieme dei maschi e delle femmine e l'insieme delle registrazioni, massimizzi il numero di danze in totale.

Discutere la complessità dell'algoritmo proposto.

Spoiler alert!

D20

Utilizziamo la programmazione dinamica per il calcolo e calcoliamo il numero di modi $T[x, i]$ con cui è possibile ottenere un valore x con i primi i dadi:

$$T[x,i] = \begin{cases} \sum_{j=1}^{F[i]} T[x-j, i-1] & i > 0 \wedge x > 0 \\ 1 & x = 0 \wedge i = 0 \\ 0 & \text{altrimenti} \end{cases}$$

D20

Il problema di questa versione è che conta tutte le possibili permutazioni; per ovviare a questo, è possibile aggiungere un terzo parametro m che indica il valore minimo del dado che può essere considerato, che deve essere più alto o uguale dei valori già ottenuti:

$$T[x,i,m] = \begin{cases} \sum_{j=m}^{F[i]} T[x-j, i-1, j] & i > 0 \wedge x > 0 \\ 1 & x = 0 \wedge i = 0 \\ 0 & \text{altrimenti} \end{cases}$$

D20

---

dice(**integer**[] F, **integer** i, **integer** x, **integer** m, **integer**[][][] T)

---

if $x = 0$ **and** $i = 0$ **then** **return** 1

if $x == 0$ **or** $i == 0$ **then** **return** 0

if $T[x, i, m] == \perp$ **then**
  $T[x, i, m] = 0$
  for $j = m$ **to** $F[i]$ **do**
    $T[x, i, m] = T[x, i, m] + \mathsf{dice}(F, i - 1, X - j, j, T)$

return $T[x, i, m]$

Il costo è pari a $O(nXM^2)$, dove M è il dado con il maggior numero di facce. Questo perchè ci sono nXM celle da riempire, ognuna delle quali viene riempita con costo $O(M)$.

Sfilatino alla Nutella

Il problema può essere risolto con tecnica greedy. Ordiniamo i segmenti per lunghezza crescente. Procediamo quindi a tagliare prima il segmento più corto, poi quello successivo e così via finché possibile (cioè fino a quando la lunghezza residua ci consente di ottenere un altro segmento).

---

integer maxSegmenti(**integer**[] V, **integer** n, **integer** L)

---

sort(V, n)
integer $i = 1$
while $i \leq n$ **and** $L \geq V[i]$ **do**
    $L = L - V[i]$
    $i = i + 1$
return $i - 1$

Ottimizza la somma

$$X[i,w] = \begin{cases} -\infty & i \geq 0 \wedge w < 0 \\ 0 & i = 0 \vee w = 0 \\ \max\{X[i-1,w], X[i,w-V[i]] + V[i]\} & \text{altrimenti} \end{cases}$$

Ottimizza la somma

integer sum(**integer**[ ] V, **integer** i, **integer** w, **integer**[ ][ ] X)

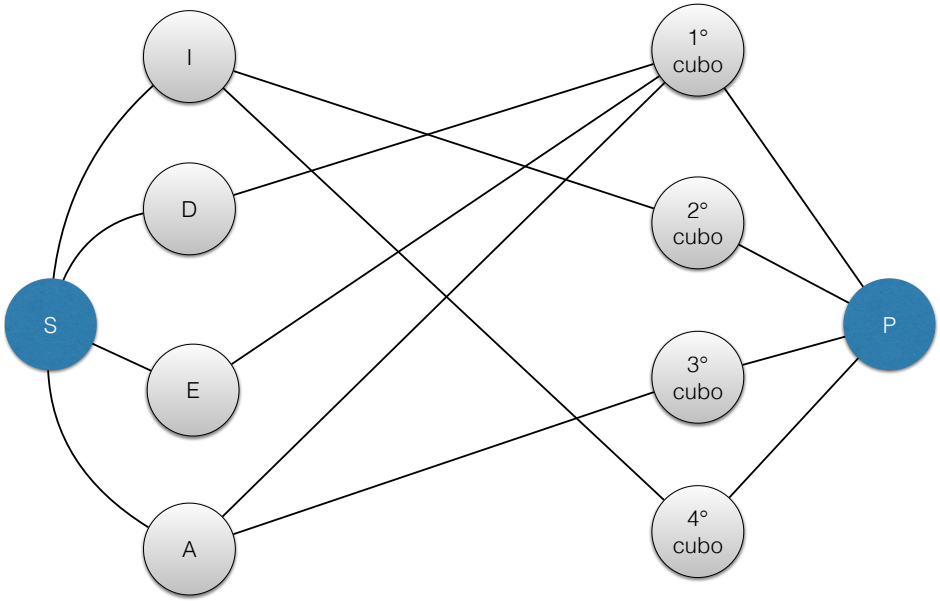
if $w < 0$ **then**
  return $-\infty$

if $i = 0$ **or** $w = 0$ **then**
  return 0

if $X[i, w] = \perp$ **then**
  $X[i, w] \leftarrow \max\{\mathsf{sum}(X, i-1, w), \mathsf{sum}(X, i, w - V[i]) + V[i]\}$

return $X[i, w]$

Cubi



Ballo di fine anno