Prima elementare

A mia figlia (prima elementare) è stato chiesto di disegnare tutte le possibili sequenze composte da tre pallini rossi e due pallini gialli.

1. Scrivere un algoritmo che stampa tutte le possibili stringhe composte da n caratteri R e da m caratteri G, per un totale di $n + m$ caratteri.
2. Scrivere un algoritmo che conta tutte queste possibile stringhe – ovviamente senza generarle tutte e poi contandole.

Complessità correttezza blah blah

Sciatori

Siano dati n sciatori di altezza $p_1, \ldots, p_n$, e n paia di sci di lunghezza $s_1, \ldots, s_n$. Il problema è assegnare ad ogni sciatore un paio di sci, in modo da minimizzare la differenza totale fra le altezza degli sciatori e la lunghezza degli sci; ovvero, se allo sciatore i è assegnato il paio di sci $h(i)$, minimizzare la seguente quantità:

$$\sum_{i=1}^{n} |p_i - s_{h(i)}|$$

Si consideri il seguente algoritmo greedy. Si individui la coppia (sciatore, sci) con la minima differenza. Si assegni allo sciatore questo paio di sci. Si ripete con gli sciatori restanti fino a quando non si è terminato.
Provare la correttezza di questo algoritmo o trovare un controesempio.

Cammini indipendenti

Dato un grafo orientato $G = (V, E)$ e due vertici u, v contenuti in V, trovare il numero totale di cammini "edge-independent", ovvero in cui un arco può comparire al massimo in un cammino.

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.

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

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

Spoiler alert!

Prima elementare

stampaCombinazioni(**char**[ ] V, **int** i, **int** n, **int** m)

if $n == 0$ **and** $m == 0$ **then**
    print V

if $n > 0$ **then**
    $V[i] =$ "R"
    stampaCombinazioni$(V, i+1, n-1, m)$

if $m > 0$ **then**
    $V[i] =$ "G"
    stampaCombinazioni$(V, i+1, n, m-1)$

Prima elementare

```
calcolaCombinazioniRic(int n, int m)
if n == 0 or m == 0 then
    return 1
else
    return calcolaCombinazioniRic(n − 1, m) +
      calcolaCombinazioniRic(n, m − 1)
```

Prima elementare

calcolaCombinazioni(**int** n, **int** m)

int[ ][ ] $M =$ **new int**$[0 \ldots n][0 \ldots m]$
for $i = 0$ **to** n **do**
  $M[i][0] = 1$
for $j = 0$ **to** m **do**
  $M[0][j] = 1$
for $i = 1$ **to** n **do**
  for $j = 1$ **to** n **do**
    $M[i][j] = M[i-1][j] + M[i][j-1]$
return $M[n][m]$

Sciatori

Si consideri il seguente input: $p = \{5, 10\}$, $s = \{9, 14\}$. Secondo l'algoritmo, associamo $p[2]$ ad $s[1]$ e $p[1]$ ad $s[2]$ (ovvero $h(1) = 2$, $h(1) = 2$. La differenza totale è $|10 - 9| + |14 - 5| = 10$. Se l'associazione fosse $h(1) = 1$ e $h(2) = 2$, la differenza totale sarebbe: $|9 - 5| + |10 - 14| = 8$. Quindi l'algoritmo proposto non è corretto.

Cammini indipendenti

Si costruisce una funzione di capacità c tale per cui

- $c(x, y) = 1$ se $(x, y) \in E$
- $c(x, y) = 0$ se $(x, y) \notin E$.

Si consideri il massimo flusso fra u e v; questo valore corrisponde al numero totale di cammini indipendenti, in quanto essendo la capacità di tutti gli archi pari ad 1, ogni cammino aumentante avrà valore di flusso 1 e tutti i suoi archi saranno distinti dagli altri cammini aumentanti.

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).

---

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

---

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

Somma di quadrati

Sia $Q(n)$ il minimo numero di quadrati necessari per esprimere n. Ovviamente il caso base è $Q(0) = 0$; per quanto riguarda la parte ricorsiva, consideriamo tutti i valori t tali per cui $1 \leq t^2 \leq n$; consideriamo quindi tutti i sottoproblemi $Q(n - t^2)$ e scegliamo fra questi quello che richiede il minor numero di quadrati, aggiungendo 1 per il quadrato considerato:

$$Q(n) = \begin{cases} 0 & n = 0 \\ \min_{1 \leq t \leq \lfloor \sqrt{n} \rfloor} \{T(n - t^2) + 1\} & n > 1 \end{cases}$$

Somma di quadrati – Programmazione dinamica

squareSum(**int** n)

int[ ] $Q =$ **new int**$[1 \ldots n]$
$Q[0] = 0$
$Q[1] = 1$
for $i = 2$ **to** n **do**
  $Q[i] = \infty$
  for $t = 1$ **to** $\lfloor \sqrt{i} \rfloor$ **do**
    $Q[i] = \mathsf{min}(Q[i], Q[i - t^2] + 1)$
return $Q[n]$

Complessità: $\Theta(n\sqrt{n})$. E' interessante notare che il numero di quadrati necessari è sempre inferiore o uguale a 4, come è stato dimostrato da Lagrange (1798).

Somma di quadrati – Backtrack

Per risolvere la seconda parte del problema, dobbiamo invece utilizzare la tecnica del backtrack. Una versione semplice per risolvere il problema potrebbe scegliere tutti i possibili valori di t tali per cui $1 \leq t^2 \leq n$, e provare ricorsivamente tutte le possibilità:

---

printSquare(**int** n, **int** [ ] S, **int** i)

```
if n == 0 then
    print S[1...i]
else
    for t = 1 to ⌊√n⌋ do
        s[i] = t
        printSquare(n − t², S, i + 1)
```

Somma di quadrati – Backtrack

Il vettore S delle scelte ha dimensione n (somma di tutti 1); l'indice i memorizza la posizione attuale nel vettore. La complessità è superpolinomiale. Questo algoritmo, tuttavia, stampa anche tutte le permutazioni dei quadrati; è accettabile per il compito, ma una soluzione migliore è la seguente

---

printSquare(**int** n, **int** [ ] S, **int** i, **int** max)

```
if n == 0 then
    print S[1 ... i]
else
    for t = 1 to ⌊min(√n, max)⌋ do
        s[i] ← t
        printSquare(n − t², S, i + 1, t)
```
