Sciatori

Siano dati n sciatori di altezza $p[1], \ldots, p[n]$ ed 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.

Sfilatino alla Nutella

Nella sagra di Hateville, è stato realizzato lo sfilatino alla Nutella più lungo 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.

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.

Quadrato binario

Sia A una matrice $n \times n$ di valori booleani 0/1. Scrivere un algoritmo che prenda in input la matrice A e la sua dimensione n, e restituisca la dimensione del più grande quadrato composto da valori 1 contenuto nella matrice. Ad esempio, nella matrice seguente, i quadrati di dimensione massima sono grandi 4×4 (ve ne sono due, di cui uno evidenziato in rosso).

```
1 0 1 0 1 0 0
1 0 1 1 1 1 0
0 1 1 1 1 1 0
0 0 1 1 1 1 0
1 1 1 1 1 1 0
1 1 1 1 1 1 0
```

Spoiler alert!

Sciatori (Esercizio 1.6 in 14-greedy.pdf)

Si consideri il seguente input: $p = [5, 10]$, $s = [9, 14]$.

- Secondo l'algoritmo, associamo:
  - $p[2] = 10$ con $s[1] = 9$ $(h(2) = 1)$
  - $p[1] = 5$ con $s[2] = 14$ $(h(1) = 2)$

  La differenza totale è $|9 - 10| + |14 - 5| = 10$.
- Se l'associazione fosse:
  - $p[2] = 10$ con $s[1] = 14$ $(h(2) = 2)$
  - $p[1] = 5$ con $s[2] = 9$ $(h(1) = 1)$

  La differenza totale è: $|9 - 5| + |14 - 10| = 8$.

Quindi l'algoritmo proposto non è corretto.

Sfilatino alla Nutella (Compito 31/05/13)

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 maxSegments(**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 $i - 1$

Sfilatino alla Nutella (Compito 31/05/13)

Sottostruttura ottima

- Sia S una soluzione ottima, $x \in S$ un elemento di S
- $S - \{x\}$ è una soluzione ottima per il sottoproblema $L - V[x]$ che non consideri il segmento x
- Se così non fosse, esisterebbe una soluzione S' tale che $|S'| > |S - \{x\}|$.
- Allora si potrebbe ottenere una soluzione $S' \cup \{x\}$ di cardinalità superiore a S, assurdo

Sfilatino alla Nutella (Compito 31/05/13)

Proprietà greedy

- Sia S una soluzione ottima che non contiene il segmento più corto m
- Sia m' il segmento più corto in S
- La soluzione $S - \{m'\} \cup \{m\}$ ha la stessa cardinalita di S ed quindi è ottima

High Line (Compito 29/06/17)

- Il problema può essere risolto tramite un algoritmo greedy
- Ogni irrigatore definisce un intervallo $[\ D[i] - R[i], D[i] + R[i]\ ]$
- Ordiniamo gli intervalli per estremo di inizio
- La variabile *last* contiene la posizione dell'ultimo punto innaffiato
- All'inizio, $last = 0$, a significare che il prossimo innaffiatore deve coprire questa posizione
- Fra tutti gli irrigatori che raggiungono *last*, si prende quello con estremo di fine più alto
- Si memorizza questo estremo di fine in *last*
- Il problema si riduce al sottoproblema $[last, L]$, dove tutti gli intervalli precedenti non devono più essere considerati perché hanno estremo superiore minore di *last*.

High Line (Compito 29/06/17)

int sprinkles(**int**[ ] D, **int**[ ] R, **int** n)

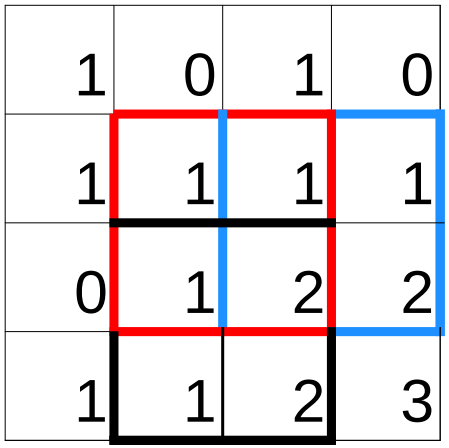
{ ordina D, R per $D[i] - R[i]$ crescenti }
int $last = 0$ % Ultimo estremo intervallo da intersecare
int $count = 0$ % Numero innaffiatori da restituire
int $i = 0$
while $i \leq n$ **and** $0 \leq last \leq L$ **do**
    int $max = -1$
    while $i \leq n$ **and** $D[i] - R[i] \leq last$ **do**
        $max = \max(D[i] + R[i], max)$
        $i = i + 1$
    $last = max$
    $count = count + 1$
return iif($last \geq L$, $count$, -1)

High Line (Compito 29/06/17)

Il costo dell'algoritmo è lineare nel numero di intervalli, ma l'ordinamento richiede $O(n \log n)$.

Quadrato binario (Compito 10/09/12)

1	0	1	0
1	1	1	1
0	1	1	1
1	1	1	1



Quadrato binario

$DP[i][j]$ contiene la dimensione del più grande quadrato composto da soli 1 il cui angolo in basso a destra sia nella posizione (i, j)

$$DP[i][j] = \begin{cases} 0 & A[i][j] = 0 \\ 1 & A[i][j] = 1 \wedge (i = 1 \vee j = 1) \\ \min\{DP[i-1][j], & \\ \quad DP[i-1][j-1], & \\ \quad DP[i][j-1]\} + 1 & \text{altrimenti} \end{cases}$$

Quadrato binario

```
int maxSquare(boolean[][] A, int n)
int[][] DP = new int[1...n][1...n]
for i = 1 to n do
    DP[i][1] = A[i][1]
    DP[1][i] = A[1][i]
for i = 2 to n do
    for j = 2 to n do
        if A[i][j] == 0 then
            DP[i][j] = 0
        else
            DP[i][j] = min(DP[i-1][j], DP[i-1][j-1], DP[i][j-1]) + 1
return max(DP, n)                    % Return max value in matrix, Θ(n^2)
```