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.

Scheduling

Siano dati n job da sottomettere ad un processore; il job i-esimo ($1 \leq i \leq n$):

- ha una deadline positiva intera $D[i]$
- un guadagno positivo intero $G[i]$
- un tempo di esecuzione pari a 1 (uguale per tutti)

Se il job i è eseguito entro l'istante $D[i]$ darà un guadagno $G[i]$, altrimenti il guadagno è 0. Trovare una sequenza di esecuzione che massimizzi il guadagno.

---

maxgain(**int**[ ] D, **int**[ ] G, **int** n)

```
{ ordina i vettori D, G per guadagno decrescente }
int t = 1
for i = 1 to n do
    if t ≤ D[i] then                % Job i può essere eseguito al tempo t
        print i
        t = t + 1
```

- Provare che l'algoritmo proposto non è corretto

Spoiler alert!

High Line

- 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
- si consideri il metro 0 (all'inizio memorizzato in $last = 0$)
- almeno un irrigatore deve raggiungere questa posizione, altrimenti si restituisce -1
- fra tutti gli irrigatori che raggiungono il metro 0, si prende quello con estremo di fine più alto (memorizzato in $last$)
- a questo punto, il problema si riduce al sottoproblema $[last, L]$, dove tutti gli intervalli che contenevano il $last$ precedente non devono più essere considerati perché hanno estremo di fine inferiore al nuovo $last$

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

High Line

```
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 ≤ n and 0 ≤ last ≤ L do
    int max = −1
    while i ≤ n and D[i] − R[i] ≤ last do
        max = max(D[i] + R[i], max)
        i = i + 1
    last = max
    count = count + 1
return iif(last ≥ L, count, -1)
```

Scheduling

1. Si considerino due job, uno con guadagno 2 e deadline 2 e uno con guadagno 1 e deadline 1. Eseguendo prima il primo job, come da algoritmo, si può eseguire solo quello e il guadagno è 2; eseguendo invece prima il secondo e poi il primo, si ottiene un guadagno di 3. Questo dimostra che l'algoritmo greedy non è corretto.