KNAPSACK FRAZIONARIO: ho "polvere di oggetti"

L = limite dello zaino

Voglio $S \subseteq O$ che massimizzi il valore degli O_i presi.

*** POSSIBILE SOLUZIONE:**

- Ordiniamo per valore
- prendo in base al valore finché ho spazio
- per l'ultimo oggetto, prendo la frazione che riempie lo zaino.

→ CONTROESEMPIO:

$L = 20$

$O = O_1 \quad O_2 \quad O_3$

$V = 10 \ ; \ 9 \ ; \ 8$

$W = 20 \ ; \ 8 \ ; \ 5$

* se prendo il primo interamente:

$S = \{O_1\} \quad W(S) = 20 \quad V(S) = 10$

* se non lo prendo interamente

$S = \{O_2, O_3, O_1\} \quad W(S) = 13 \quad V(S) = 17$

in più → ho ancora $L = 7$ kg.

*** Quindi:** la soluzione di prima <u>non</u> mi restituisce l'ottimo.

*** EFFETUIAMO UNA MODIFICA:**

- Calcolo $R_i = V_i / W_i$ per ogni O_i.
- Ordino in base R_i.
- Seguo l'algoritmo di prima.

→ CON L'ESEMPIO DI PRIMA:

$R = 1/2 \ ; \ 1/8 \ ; \ 8/5$

Allora $S = \{O_3, O_2,$ + una frazione $O_1\}$ → giusto!

*** É CORRETTO ORA?**

→ Dimostrazione matematica! (non sto quì a spiegarlo, é giusto).

QUESTO TIPO DI ALGORITMO É DETTO **GREEDY**.

GREEDY:

- **Struttura generale**:

  - <calcola n parametri> → $O(n)$
  - Ordino per parametro → $\Theta(n \log n)$
  - $S = \emptyset$

  for $i = 1$ to n: → $O(n)$

  if I_i "può essere aggiunto":

  $S = S \cup \{I_i\}$

  Retur(S); → $\Theta(1)$

! * tutto ció che é dentro < > é opzionale, ed il suo $T(n)$ é quello che si presenta "generalmente".

// Passa per ogni oggetto dell'input, e
// vedi se puó essere aggiunto.

* In generale, un Greedy ha $O(n \log n)$ → $\simeq$ "il tempo che ci mettiamo ad ordinare i valori."

LA DIFFICOLTÁ IN GENERE NEI GREEDY → é la dimostrazione (farla sempre).

ESERCIZIO

Immaginiamo di star facendo un viaggio, volendo minimizzare il numero di soste per fare benzina.

Scrivere l'algoritmo Greedy, dimostrarlo e discuterne la complessità.

$km = 0 \longrightarrow N$
Autonomia auto r
Ho n aree di servizio $\rightarrow$ $k[1 \dots n]$

Assumiamo:
- che la $d(k_i, k_{i+1}) < r$
- che le stazioni siano ad un $0 < km < N$.

```
S = ∅ ;  A = r - A[i]                    Θ(1)

// riordiniamo in base al nostro
// itinerario.

k' = ordina (k)                          Θ(n log n)

// Per ogni stazione, mi chiedo
for i = 1 to n                           Θ(n)
    // Posso raggiungere la successiva?
    if A ≥ (k'[i+1] - k[i])
        A = A - k'[i+1] - k[i] ;    // Sì
    else
        S = S ∪ {k'[i]} ;           // No
        A = r - k'[i+1] ;

Return S;                                Θ(1)
```

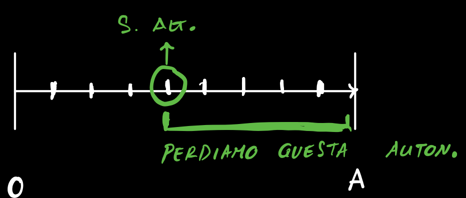
$T(n) = \Theta(n \log n)$

$S(n) = |k|$

$\downarrow$

non ho strutture dati in più.

* Manca la dimostrazione (o il controesempio):



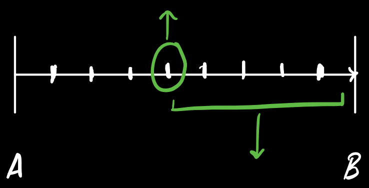
* Posso (soluzione alternativa)

- Fermarmi comunque ad A
- Fermarmi in una tra 0 ed A

} non posso scegliere dopo A, a causa dell'autonomia prima sosta che faccio

AL MAX

Ci fermiamo qui perché oltre non riusciamo



Autonomia che avevamo perso prima.

* Posso →

- prendere una stazione prima della fine dell'autonomia
  ↳ PERDO AUTONOMIA (ULTERIORE)
- fermarmi dove finisce l'autonomia
  ↳ PROPAGO L'AUTONOMIA PERSA AL PUNTO A.

In ogni caso alla fine →

- mi devo fermare UNA O PIÙ VOLTE in più rispetto a $|S|$.
- oppure esattamente $|S|$ fermate (uguale)

↓

NON POSSO MIGLIORARE ULT. $|S|$

ESERCIZIO (problema di copertura)

Dato $x_1 \dots x_n \in \mathbb{R}$, determinare il più piccolo insieme di intervalli CHIUSI di lunghezza unitaria, che vada a coprire tutti i punti in esame.

Esempio con input:

$X = \{ x_1, x_2 \dots x_n \}$

lunghezza intervalli chiusi $= 1$

7,2 4,5 5,1 3,4 8,3 7,8 2,4

→ 2,4 3,4 4,5 5,1 7,2 7,8 8,3

numero minimo di intervalli : 4

$T(n) = \Theta(n \log n)$

```
S = ∅
Y = ordina(X);  last = [Y_1 ; Y_1 + 1]
for i = 1 to n
    if Y[i] ∉ last
        last = [Y[i], Y[i] + 1]
        S = S ∪ last
Return S.
```

la dimostrazione → é identica al problema dei km.