Problema dello Zaino (Knapsack Problem)

I. Introduzione

Il **problema dello zaino** è un problema di ottimizzazione che consiste nel massimizzare il profitto tratto dalla vendita di n oggetti, ognuno di loro capace di generare un profitto p_i e con un costo di acquisto (che dobbiamo sostenere noi) c_i.

Il problema sta quindi nel generare il massimo profitto dalla vendita di questi oggetti, avendo un budget limitato pari a B in fase di acquisto.

Avremo quindi **3 vettori**: uno per gli **indici** dei vari oggetti, un vettore dei **profitti** ed uno dei **costi**.

Formalmente diciamo che si vogliono trovare n valori $x_i, x_2, \ldots, x_n$ tali che:

- $x_n \in \{0,1\} \ \ (1 \leq i \leq n)$ → questo non sarà altro che il vettore di output, nel quale ci sono da *1* a *n* elementi (con *n* numero degli elementi dati nel problema); questo vettore conterrà solo valori 1 (se l'elemento in quello specifico indice viene incluso nello zaino) oppure 0 (se l'elemento con indice i non viene incluso).
- $\sum_{i=1} P_i X_i \text{ sia massima}$ → i profitti devono essere massimizzati
- $\sum_{i=1} C_i X_i \leq B$ → la somma dei costi deve essere inferiore al budget

II. Soluzione con Tecnica Greedy

Per risolvere questo problema con **tecnica greedy** si procede come segue:

Si definisce una funzione **zainoGreedy** che ha come parametri: **numero di oggetti** n che va da 0 a $n-1$, vettore P dei **profitti**, vettore C dei **costi** e **budget** B.

Il risultato viene restituito in un vettore X che contiene valori 0/1 dove l'*i-esimo* componente vale 1 se l'*i-esimo* oggetto è stato inserito nello zaino e 0 altrimenti.

La funzione ricava poi il rapporto **costo/profitto** P_i/C_i per ogni elemento e ordina i vettori in ordine decrescente in base al valore calcolato.

Dopo aver ordinato i gli oggetti, l'algoritmo procede con la scansione del vettore ed aggiunge un oggetto allo zaino solo se c'è ancora budget sufficiente.

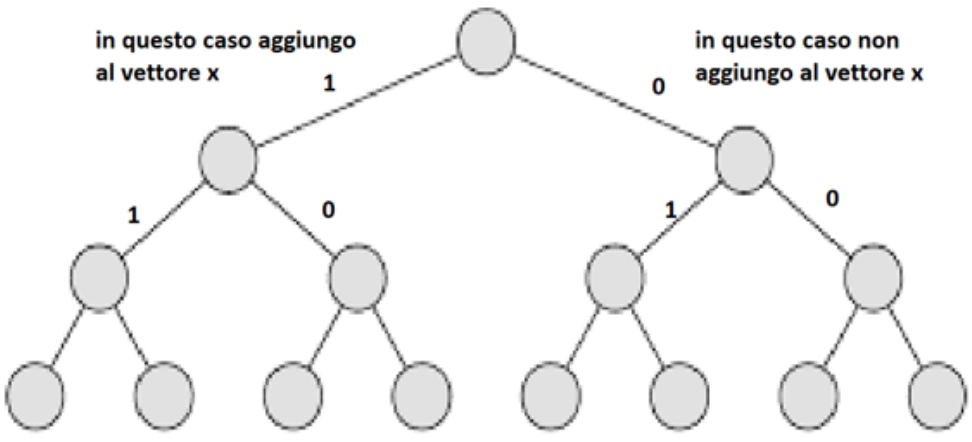
Si nota l'utilizzo delle tecnica greedy dal fatto che la decisione viene presa volta per volta, oggetto per oggetto, senza mai tornare indietro.

II. Soluzione con Backtracking

Per risolvere questo problema con il **backtracking** si va a creare un albero di ricerca.

Se abbiamo n oggetti, si può costruire ogni elemento dello spazio di ricerca in n stadi: all'*i-esimo stadio* si decide se aggiungere o meno l'*i-esimo* elemento al vettore X.
Di conseguenza anche l'albero di ricerca avrà profondità n.

Le soluzioni parziali saranno rappresentate dal vettore X che conterrà valori 0/1 dove l'*i-esimo* componente vale 1 se l'*i-esimo* oggetto è stato inserito nello zaino e 0 altrimenti.



In base all'arco che si decide di seguire il valore viene inserito nel vettore o meno

Inoltre, per **valutare le soluzioni parziali** abbiamo bisogno di 2 funzioni:

- una funzione che restituisce $\texttt{true}$ se il **costo della soluzione parziale supera il budget**

$$\sum_{k=1}^{i} C_k X_k > B$$

- una funzione che restituisce true se il **profitto ottenibile dalla soluzione parziale** è minore dell'**ottimo corrente**

$$\sum_{i=1}^{l} P_i X_i + \sum_{i+1}^{n} P_i < P_{ott}$$

In entrambi i casi la soluzione va scartata, in quanto non rispetta un vincolo oppure non massimizza il profitto.