In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

Dominik Matracki 408558

Problem plecakowy 0-1 KP

Mamy plecak o maksymalnej pojemnosci B oraz zbior N elementow. Kazdy element charakteryzuje sie zyskiem oraz waga. Dyskretny problem plecakowy:

- Maksymalizuj zysk $\sum_{j=1}^{n} c_j x_j$
- Przy ograniczeniach wagi $B \sum_{j=1}^{n} a_j x_j \leq B, x_j \in 0, 1$

Przyklad problemu plecakowego

$x_1 + 3x_2 + 2x_3 + 2x_4 - > max$

$x_1 + 4x_2 + 3x_3 + 3x_4 \leq 7$

In [3]:
```
from typing import List

def maxArg(array: List[int]):
    if len(array) == 0:
        return

    maximum = -np.inf
    arg = None
    for i, el in enumerate(array):
        if el > maximum:
            arg = i
            maximum = el
    return arg, maximum


def binaryKP(weights, prices, capacity):
    n = len(weights)
    table = [[0 for _ in range(capacity+2)] for _ in range(n+1)]

    # Tworze tabele zyskow
    for i in range(1, n+1):
        for w in range(1, capacity+2):
            if 0 < w-weights[i-1] < capacity + 1:
                table[i][w] = max(table[i-1][w], table[i-1][w-weights[i-1]] + prices[i-1
                continue
            table[i][w] = table[i-1][w]

    # Find max in last row
    arg, maximum = maxArg(table[-1])
    print("Tablica: \n", np.array(table))
    print("Maksymalna wartosc zysku: ", maximum)

    # Ukladam sciezke
    currentPrice = maximum
    i = n
    result = [0 for _ in range(n)]
    while i > 0:
```

```
        if table[i][arg] != table[i-1][arg]:
            result[i-1] = 1
            currentPrice -= p[i-1]
            arg = table[i-1].index(currentPrice)
        i -= 1
    return result
```

In [4]:
```
w = [1, 4, 3, 3, 1, 2, 2, 5, 3, 5]
p = [1, 3, 2, 2, 9, 3, 4, 4, 2, 4]
binaryKP(w, p, 7)
```

```
Tablica:
 [[ 0  0  0  0  0  0  0  0  0]
 [ 0  0  1  1  1  1  1  1  1]
 [ 0  0  1  1  1  3  4  4  4]
 [ 0  0  1  1  2  3  4  4  5]
 [ 0  0  1  1  2  3  4  4  5]
 [ 0  0  9 10 10 11 12 13 13]
 [ 0  0  9 10 12 13 13 14 15]
 [ 0  0  9 10 13 14 16 17 17]
 [ 0  0  9 10 13 14 16 17 17]
 [ 0  0  9 10 13 14 16 17 17]
 [ 0  0  9 10 13 14 16 17 17]]
Maksymalna wartosc zysku:  17
```

Out[4]:
```
[1, 0, 0, 0, 1, 1, 1, 0, 0, 0]
```

Zadanie 3

Jakie zalozenia musza byc spelnione dla wag oraz zyskow? Co stanie sie jesli te zalozenia nie spelnimy?

Wagi oraz zyski musza byc dodatnie, poniwaz gdyby wagi byly ujemne to mozliwe by bylo dodawanie wszystkich przedmiotow oraz natychmiastowe uzyskanie najlepszego zysku. Natomiast gdyby zyski byly ujemne to nieoptymlane byloby dodawanie jakiegokolwiek przemiotu.

Jaka jest zlozonosc obliczeniowa algorytmu?

Zlozonosc obliczeniowa algorytmu wynosi $O(B \cdot n)$, gdzie n - liczba przedmiotow, B - pojemnosc.