KnapSack Problem

BruteForce approach

idea

check all the possible subsets of existing rocks

Timecomplexity

choose a subset => O(2^n)

check a subset => O(n)

merge => O(C(n, 1)*1 + C(n, 2)*2 + ... + C(n, n)*n!)

DynamicProgramming approach

idea

if dp[n][W] means best answer for n rocks and maximum W TotalWeight:

```
dp[i][j] = max (dp[i-1][j], dp[i-1][j-w[i]] + value[i]) if w[i] <= k

else dp[i][j] = dp[i-1][j]
```

Timecomplexity

we need 2 loops to update our dp table with O(nk)

In [2]:
```
from IPython.display import clear_output
import time
```

In [22]:
```
def KnapSack (Weights, Values, TotalWeight):

    #init
    n = len(Weights)
    k = len(Weights)
    dp = [[0 for i in range(TotalWeight+1)] for j in range (n+1)]

    #update dp
    for i in range(1, n):
        for j in range(TotalWeight+1):
            clear_output()

            if (Weights[i] <= j):
                dp[i][j] = max(dp[i-1][j], dp[i-1][j-Weights[i]] + Values[i])
            else:
                dp[i][j] = dp[i-1][j]

            #print the dp table
            for x in range(n):
                print(dp[x])
            time.sleep(1)

    #extract chosen rocks
    rocks = []
    w = TotalWeight
    i = n
    res = dp[n-1][w]
    while i > 0 and res > 0:
        if res == dp[i-1][w]:
            i -= 1
            continue
        else:
            rocks.append((i, Weights[i], Values[i]))
            res -= Values[i]
            w -= Weights[i]
            i -= 1

    return (dp[n-1][TotalWeight], rocks)
```

In [20]:
```
KnapSack([0, 1, 1, 10, 14, 17], [0, 4 ,5, 12, 2, 25], 20)
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4]
[0, 5, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9]
[0, 5, 9, 9, 9, 9, 9, 9, 9, 9, 12, 17, 21, 21, 21, 21, 21, 21, 21, 21, 21]
[0, 5, 9, 9, 9, 9, 9, 9, 9, 9, 12, 17, 21, 21, 21, 21, 21, 21, 21, 21, 21]
[0, 5, 9, 9, 9, 9, 9, 9, 9, 9, 12, 17, 21, 21, 21, 21, 21, 25, 30, 34, 34]
```

Out[20]:
```
(34, [(5, 17, 25), (2, 1, 5), (1, 1, 4)])
```

In [ ]:
```

```