**Experiment No.:-4**

**Write a program to solve a 0-1 Knapsack problem using dynamic programming or branch and bound strategy.**

Source Code:-

|  |
| --- |
| **def** knapsack\_01(n, values, weights, W):  dp **=** [[0] **\*** (W**+**1) **for** \_ **in** range(n**+**1)]  **for** i **in** range(n**+**1):  **for** w **in** range(W**+**1):  **if** i **==** 0 **or** w **==** 0: dp[i][w] **=** 0  **elif** weights[i**-**1] **<=** w:  dp[i][w] **=** max(dp[i**-**1][w], dp[i**-**1][w**-**weights[i**-**1]] **+** values[i**-**1])  **else**:  dp[i][w] **=** dp[i**-**1][w]  selected\_items **=** [] i, w **=** n, W **while** i **>** 0 **and** w **>** 0:  **if** dp[i][w] **!=** dp[i**-**1][w]: selected\_items**.**append(i**-**1) w **-=** weights[i**-**1]  i **-=** 1  **return** dp[n][W], selected\_items    *# Take input from the user*  n **=** int(input("Enter the number of items: "))  values **=** list(map(int, input("Enter the values of the items separated by space: ")**.**spl weights **=** list(map(int, input("Enter the weights of the items separated by space: ")**.** W **=** int(input("Enter the maximum capacity of the knapsack: "))  max\_value, selected\_items **=** knapsack\_01(n, values, weights, W) print("Maximum value:", max\_value) print("Selected items:", selected\_items) |

In [1]:

s

Enter the number of items: 4

Enter the values of the items separated by space: 3 4 5 6 Enter the weights of the items separated by space: 2 3 4 6

Enter the maximum capacity of the knapsack: 5

Maximum value: 7

Selected items: [1, 0]

In [ ]:

**Correct code :**def knapsack\_01(n, values, weights, W):

# Initialize the DP table with 0s

dp = [[0] \* (W + 1) for \_ in range(n + 1)]

# Build the DP table

for i in range(n + 1):

for w in range(W + 1):

if i == 0 or w == 0:

dp[i][w] = 0

elif weights[i - 1] <= w:

dp[i][w] = max(dp[i - 1][w], dp[i - 1][w - weights[i - 1]] + values[i - 1])

else:

dp[i][w] = dp[i - 1][w]

# Backtrack to find the items that were selected

selected\_items = []

i, w = n, W

while i > 0 and w > 0:

if dp[i][w] != dp[i - 1][w]:

selected\_items.append(i - 1) # Item index

w -= weights[i - 1]

i -= 1

# Return the maximum value and the selected items

return dp[n][W], selected\_items

# Take input from the user

n = int(input("Enter the number of items: "))

values = list(map(int, input("Enter the values of the items separated by space: ").split()))

weights = list(map(int, input("Enter the weights of the items separated by space: ").split()))

W = int(input("Enter the maximum capacity of the knapsack: "))

# Get the maximum value and selected items

max\_value, selected\_items = knapsack\_01(n, values, weights, W)

# Output the results

print("Maximum value:", max\_value)

print("Selected items:", selected\_items)