DYNAMIC PROGRAMMING:

1. MEMOIZATION(BACKTRACKING)

* RECURSION
* LESS USED

1. TABULATION

* ITERATION
* MOSTLY USED

Code:

def calc\_max(p,w,c,n):

if n==0 or c==0:

return 0

if w[n-1]>c :

return calc\_max(p,w,c,n-1)

else:

return max(p[n-1]+calc\_max(p,w,c-w[n-1],n-1),calc\_max(p, w, c, n-1))

p=[5,10,15,7,8,9,4]

w=[1,3,5,4,1,3,2]

c=15

n=len(p)

dp=[[-1 for i in range (c+1)] for j in range (n+1)]

print(dp)

print('maximum profit is',calc\_max(p, w, c, n))

#print(calc\_max(p, w, c, n))

Output:

[[-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1], [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1], [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1], [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1], [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1], [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1], [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1], [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]]

maximum profit is 51