**Knapsack Problem**

**Description:**

It’s a full generic program that takes number of rows as input and then generate these rows with random weights and significances and then after a binary selection, the program generates the best result.

**Code:**

from random import \*

item=[]

weight=[]

percent=[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

significance=[]

range\_num=int(input("Enter the number of rows for comparison : "))

total\_list=[]

res=[]

final\_list=[]

def table\_values():

print("\n item",item,"\n")

print("\n weight",weight,"\n")

print("\n significance",significance,"\n")

def initializer():

for i in range(range\_num):#here

item.append(i)

weight.append(randint(1, 12))

significance.append(percent[(randint(0, 10-1))])

#p()

###checking####

#m=weight[0]

#for i in range(1,len(weight)):

# if(m>weight[i]):

# m=weight[i]

##print(m)

#p()

#m=0

#for i in range(len(weight)):

# if(weight[i]==12):

# if(m<significance[i]):

# m=significance[i]

# weight[i]=0

# significance[i]=0

#print(m)

#i=-1

#while(True):

# i=i+1

# try:

# if(weight[i]==0):

# weight.pop(i)

# significance.pop(i)

# except:

# print(len(weight),"weight")

# print(len(significance),"significancwe")

# break

#p()

def random(n):

l=[]

# Python3 implementation of the

# above approach

# Function to print the output

def custom():

l1=[]

index=0

for i in range(0,len(total\_list),range\_num):#here

for j in range(0,range\_num):#here

l1.append(total\_list[index])

index=index+1

final\_list.append(l1)

l1=[]

print(final\_list)

def printTheArray(arr, n):

for i in range(0, n):

print(arr[i], end = " ")

l.append(arr[i])

print(end = "\t\tPROCESSING")

print()

# Function to generate all binary strings

def generateAllBinaryStrings(n, arr, i):

if i == n:

printTheArray(arr, n)

return

# First assign "0" at ith position

# and try for all other permutations

# for remaining positions

arr[i] = 0

generateAllBinaryStrings(n, arr, i + 1)

# And then assign "1" at ith position

# and try for all other permutations

# for remaining positions

arr[i] = 1

generateAllBinaryStrings(n, arr, i + 1)

# Driver Code

#n = 4

arr = [None] \* n

# Print all binary strings

generateAllBinaryStrings(n, arr, 0)

global total\_list

total\_list=l

custom()

return(l)

#print(l)

#print(len(l))

def result(l):

#######breaking list in final ##########

final=[]

l1=[]

index=0

for i in range(0,len(l),range\_num):#here

for j in range(0,range\_num):#here

l1.append(l[index])

index=index+1

final.append(l1)

l1=[]

#print(len(final))

########max###############

print("\n\nRESULT\n\n")

best=0

max\_weights=[]

max\_significance=[]

while(best!=range\_num):

index=0

max1=-1

x=0

row=0

for i in range(len(final)):

k=0

for j in range(range\_num):#here

x+=1

if(final[i][j]==1):

k+=weight[j]

if((k>max1)and(k<=20)):#max weight

max1=k

index=i

row=int(x/range\_num)#here

#print("max is:",max1)

max\_weights.append(max1)

#print("row is:",row," bcz combinations are : ", end="")

#for i in range(row-1,row):

# print(final[i])

#print(index)

#####significance######

k=0

for j in range(range\_num):#here

if(final[row-1][j]==1):

#print("true")

k+=significance[j]

#print("significance",k)

max\_significance.append(k)

res.append(row-1)

print(res)

best+=1

###rejecting already printed rows#####

for i in range(range\_num):

final[index][i]=0

#print(max\_weights)

#print(max\_significance)

best=0

index=0

print("item\t\tweight\t\tsignificances\t combinations")

for i in range(range\_num):

print(i+1,"\t\t",max\_weights[i],"\t\t",max\_significance[i],"\t\t",final\_list[res[i]])

if(best<max\_significance[i]):

best=max\_significance[i]

index=i

print("\nBEST IS : ")

print("item\t\tweight\t\tsignificances\t combinations")

print(index+1,"\t\t",max\_weights[index],"\t\t",max\_significance[index],"\t\t",final\_list[res[index]])

####way of procedure####

#l=random(len(weight))

#result(l)

initializer()

table\_values()

result(random(len(weight)))

print(res)

print(final\_list)

**Output:**

