Tina Borundia

Practical 3

Aim: Perform Fractional Knapsack for the given scenario.

Problem Definition: Suppose you are a transport dealer and want to load a truck with different types of boxes. Assume there are 50 types of boxes (Box-1 to Box-50), which weigh different and that the truck has a maximum capacity (truckSize). Each box has a profit value associated with it. It is the commission that the transporter will receive after transporting the box. You can choose any box to put on the truck as long as the number of boxes does not exceed truckSize. ¶

In [4]:
```
import time
```

In [6]:
```
### Knapsack Problem : minimum weight
def knapsack(y,capacity):
    start=time.perf_counter()
    x=sorted(y,key=lambda i:i[0])
    result=[]
    obj=[]
    profit=0
    for i in range(len(x)):
        if x[i][0]<=capacity and capacity>0:
            capacity-=x[i][0]
            profit+=x[i][1]
            result.append('C')
            obj.append(x[i])

        elif capacity>0:
            profit+=x[i][1]*(capacity/x[i][0])
            cap=0
            break
    end=time.perf_counter()
    timei=end-start
    print("Time taken : ",timei)
    return profit
```

In [7]:
```
w=[7, 1, 30, 22, 80, 94, 11, 81, 70, 64, 59, 18, 13, 36, 3, 8, 15, 42, 9, 17,
42, 47, 52,32, 26, 48, 55, 6, 29, 84, 2, 4, 18, 56, 7, 29, 93, 44, 71, 3, 86,
66, 31, 65, 37, 79,20, 65, 52, 13]
p=[ 360, 83, 59, 130, 431, 67, 230, 52, 93, 125, 670, 892, 600, 38, 48, 147, 7
8, 256,63, 17, 120, 164, 432, 35, 92, 110, 22, 42, 50, 323, 514, 28, 87, 73, 7
8, 15, 26,78, 210, 36, 85, 189, 274, 43, 33, 10, 19, 389, 276, 312 ]
```

In [9]:
```
l1=[]
for i in range(len(w)):
    temp=[]
    temp.append(w[i])
    temp.append(p[i])
    temp.append(p[i]/w[i])
    l1.append(temp)
print(l1)
```

```
[[7, 360, 51.42857142857143], [1, 83, 83.0], [30, 59, 1.9666666666666666], [2
2, 130, 5.909090909090909], [80, 431, 5.3875], [94, 67, 0.7127659574468085],
[11, 230, 20.90909090909091], [81, 52, 0.6419753086419753], [70, 93, 1.328571
4285714285], [64, 125, 1.953125], [59, 670, 11.35593220338983], [18, 892, 49.
55555555555556], [13, 600, 46.15384615384615], [36, 38, 1.0555555555555556],
[3, 48, 16.0], [8, 147, 18.375], [15, 78, 5.2], [42, 256, 6.095238095238095],
[9, 63, 7.0], [17, 17, 1.0], [42, 120, 2.857142857142857], [47, 164, 3.489361
7021276597], [52, 432, 8.307692307692308], [32, 35, 1.09375], [26, 92, 3.5384
615384615383], [48, 110, 2.2916666666666665], [55, 22, 0.4], [6, 42, 7.0], [2
9, 50, 1.7241379310344827], [84, 323, 3.8452380952380953], [2, 514, 257.0],
[4, 28, 7.0], [18, 87, 4.833333333333333], [56, 73, 1.3035714285714286], [7,
78, 11.142857142857142], [29, 15, 0.5172413793103449], [93, 26, 0.27956989247
311825], [44, 78, 1.7727272727272727], [71, 210, 2.9577464788732395], [3, 36,
12.0], [86, 85, 0.9883720930232558], [66, 189, 2.8636363636363638], [31, 274,
8.838709677419354], [65, 43, 0.6615384615384615], [37, 33, 0.891891891891891
9], [79, 10, 0.12658227848101267], [20, 19, 0.95], [65, 389, 5.98461538461538
5], [52, 276, 5.307692307692307], [13, 312, 24.0]]
```

In [10]:
```
x=knapsack(l1,850)
print(x)
```

```
Time taken :  6.600000051548705e-05
5845.375
```

In [11]:
```
### Maximum Profit
def max_profit(y,capacity):
    start=time.perf_counter()
    x=sorted(y,key=lambda i:i[1],reverse=True)
    result=[]
    obj=[]
    profit=0
    for i in range(len(x)):
        if x[i][0]<=capacity and capacity>0:
            capacity-=x[i][0]
            profit+=x[i][1]
            result.append('C')
            obj.append(x[i])

        elif capacity>0:
            profit+=x[i][1]*(capacity/x[i][0])
            cap=0
            break
    end=time.perf_counter()
    timeb=end-start
    print("Time taken : ",timeb)
    return profit
```

In [12]:
```
x=max_profit(l1,850)
print(x)
```

```
Time taken :  0.00015779999739606865
7046.291666666667
```

In [13]:
```
### maximum profit/weight
def ratio(y,capacity):
    start=time.perf_counter()
    x=sorted(y,key=lambda i:i[2],reverse=True)
    result=[]
    obj=[]
    profit=0
    for i in range(len(x)):
        if x[i][0]<=capacity and capacity>0:
            capacity-=x[i][0]
            profit+=x[i][1]
            result.append('C')
            obj.append(x[i])

        elif capacity>0:
            profit+=x[i][1]*(capacity/x[i][0])
            cap=0
            break
    end=time.perf_counter()
    timec=end-start
    print("Time taken : ",timec)
    return profit
```

In [14]:
```
x=ratio(l1,850)
print(x)
```

```
Time taken :  6.24999993306119e-05
7476.857142857143
```

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

In [ ]:
```
import numpy as np
xpoints=np.array(['weight','profit','ratio'])
ypoints=np.array([6.600000051548705e-05, 0.00015779999739606865,6.249999933061
19e-05])
plt.plot(xpoints,ypoints)
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x6ffffef458d0>]
```

In [ ]:
```

```