Greedy Algorithm

In [4]:

```
def greedyByProfit(p,w,m) :
    sortByProfit(p,w)
    sumpixi = 0
    for i in range(0,len(p))  :
        if m <= 0 :
            break;
        elif m >= w[i] :
            sumpixi = sumpixi + (p[i] * 1)
        else :
            sumpixi = sumpixi + (p[i]*(m/w[i]))

        m -= w[i]
    print("suboptimal solution when greedy by profit :",sumpixi)

def greedyByWeight(p,w,m) :
    sortByWeight(p,w)
    sumpixi = 0
    for i in range(0,len(w))  :
        if m <= 0 :
            break;
        elif m >= w[i] :
            sumpixi = sumpixi + (p[i] * 1)
        else :
            sumpixi = sumpixi + (p[i]*(m/w[i]))
        m -= w[i]
    print("suboptimal solution when greedy by weight :",sumpixi)

def greedyByProfitPerUnit(p,w,m) :
    ratio = []
    for i in range(0,len(p)) :
        ratio.append(p[i]/w[i])

    sortByRatio(ratio,p,w)
    sumpixi = 0
    for i in range(0,len(w))  :
        if m <= 0 :
            break;
        elif m >= w[i] :
            sumpixi = sumpixi + (p[i] * 1)
        else :
            sumpixi = sumpixi + (p[i]*(m/w[i]))
        m -= w[i]
    print("optimal solution when maximum profit is per unit capacity :",sumpixi)



def sortByProfit(p,w): # Decreasing order

    for i in range(1, len(p)):

        key = p[i]
        temp = w[i]
        j = i-1
        while j >=0 and key > p[j] :
                p[j+1] = p[j]
                w[j+1] = w[j]
                j -= 1
        p[j+1] = key
        w[j+1] = temp

def sortByWeight(p,w): # Increasing order

    for i in range(1, len(w)):

        key = w[i]
        temp = p[i]
        j = i-1
        while j >=0 and key < w[j] :
                w[j+1] = w[j]
                p[j+1] = p[j]
                j -= 1
        w[j+1] = key
        p[j+1] = temp

def sortByRatio(ratio,p,w): # dencreasing order

    for i in range(1, len(ratio)):

        key = ratio[i]
        temp1 = p[i]
        temp2 = w[i]
        j = i-1
        while j >=0 and key > ratio[j] :
                ratio[j+1] = ratio[j]
                p[j+1] = p[j]
                w[j+1] = w[j]
                j -= 1
        ratio[j+1] = key
        p[j+1] = temp1
        w[j+1] = temp2


if __name__ == '__main__':
    p = [int(item) for item in input("Enter profit : ").split()]
    w = [int(item) for item in input("Enter weight : ").split()]
    m = int(input("Enter maximum objects can be choosen (m) :"))
    greedyByProfit(p,w,m)
    greedyByWeight(p,w,m)
    greedyByProfitPerUnit(p,w,m)
```

```
Enter profit : 12 9 6 11 7 5 8
Enter weight : 6 3 2 5 7 5 9
Enter maximum objects can be choosen (m) :15
suboptimal solution when greedy by profit : 32.888888888888886
suboptimal solution when greedy by weight : 31
optimal solution when maximum profit is per unit capacity : 36.0
```

Job Sequencing with Deadlines problem using Greedy design strategy

In [26]:

```
def jobSequencing(j,d,p) :
    sortByProfit(p,d,j)
    profit = p[0]  # 1st job is selected
    r = 1  # 0 to 1 slot is assigned
    #n = 0
    print("Scheduled jobs :")
    print(j[0], " is selected and assigned slot is [",r-1 ,r,"]")
    for i in range(1,len(j)) :
        if d[i] > r :
            profit += p[i]
            r += 1
            print(j[i], " is selected and assigned slot is [",r-1 ,r,"]")

            #n+=1


            #k.insert(i , 1)
    print("profit=",profit)

def sortByProfit(p,d,jobs): # Decreasing order

    for i in range(1, len(jobs)):

        key = p[i]
        temp = d[i]
        temp2 = jobs[i]
        j = i-1
        while j >=0 and key > p[j] :
                p[j+1] = p[j]
                d[j+1] = d[j]
                jobs[j+1] = jobs[j]
                j -= 1
        p[j+1] = key
        d[j+1] = temp
        jobs[j+1] = temp2


if __name__ == '__main__':
    job = [ji for ji in input("Enter Jobs: ").split()]
    d = [int(di) for di in input("Enter Deadline: ").split()]
    p = [int(pi) for pi in input("Enter Profit: ").split()]
    jobSequencing(job,d,p)
```

```
Enter Jobs: j1 j2 j3 j4 j5
Enter Deadline: 2 2 1 3 3
Enter Profit: 20 18 10 5 1
Scheduled jobs :
j1  is selected and assigned slot is [ 0 1 ]
j2  is selected and assigned slot is [ 1 2 ]
j4  is selected and assigned slot is [ 2 3 ]
profit= 43
```