# COURSE HANDOUT

| | |
|---|---|
| Course Code | ACSC13 |
| Course Name | Design and Analysis of Algorithms |
| Class / Semester | IV SEM |
| Section | A-SECTION |
| Name of the Department | CSE-CYBER SECURITY |
| Employee ID | IARE11023 |
| Employee Name | Dr K RAJENDRA PRASAD |
| Topic Covered | Job sequencing with deadlines |
| Course Outcome/s | Compare Identify suitable problem solving techniques for a given problem and finding optimized solutions using Greedy Method |
| Handout Number | 27 |
| Date | |

## Content about topic covered: Job Sequencing with Deadlines

## Job Sequencing with Deadlines

We are given a set of n jobs. Associated with job i is an integer deadline $d_i \geq 0$ and a profit $P_i > 0$. For any job i the profit $P_i$ is earned if and only if the job is completed by its deadline. To complete the job, one has to process the job on a machine for one unit of time. Only one machine is available for processing jobs.

A feasible solution for this problem is a subset J of jobs such that each job in this subset can be completed by its deadline. The value of a feasible solution J is the sum of the profits of the jobs in J, or

$$\sum_{i \in J} P_i$$

An optimal solution is feasible solution with maximum value.

E.g.: Let n = 4, $(P_1,P_2,P_3,P_4) = (100,10,15,27)$ and $(d_1,d_2,d_3,d_4)= (2,1,2,1)$
The feasible solutions and their values are:

| S.No. | Feasible solution | Processing sequence | value |
|---|---|---|---|
| 1 | (1,2) | 2,1 | 110 |
| 2 | (1,3) | 1,3 or 3,1 | 115 |
| 3 | (1,4) | 4,1 | 127 |
| 4 | (2,3) | 2,3 | 25 |
| 5 | (3,4) | 4,3 | 42 |
| 6 | (1) | 1 | 100 |
| 7 | (2) | 2 | 10 |
| 8 | (3) | 3 | 15 |
| 9 | (4) | 4 | 27 |

Solution 3 is optimal. In this solution, only Jobs 1 and 4 are processed and the value is 127. These jobs must be processed the order job 4 followed by job 1.

Here the objective function is $\sum_{i \in J} P_i$

To include the next job that increases $\sum_{i \in J}(P_i)$ subject to the constraint that J is a feasible solution.
Consider the jobs in descending order of $P_i$.

| | |
|---|---|
| J = ∅ | $\sum_{i \in J} P_i$ |
| J= {1} | feasible |
| J= {1,4} | feasible |
| J= {1,3,4} | not feasible |
| J= {1,2,4} | not feasible |

So J= {1,4}

Value = 127.

Eg: Let n = 5, $(P_1,P_2,P_3,P_4,P_5) = (20,15,10,5,1)$ and $(d_1,d_2,d_3,d_4,d_5)= (2,2,1,3,3)$

| J | Assigned slots | Job considered | Action | Profit |
|---|---|---|---|---|
| ∅ | None | 1 | Assign to [1,2] | 0 |
| {1} | [1,2] | 2 | Assign to [0,1] | 20 |
| {1,2} | [0,1],[1,2] | 3 | Cannot fit. Reject | 35 |
| {1,2} | [0,1],[1,2] | 4 | Assign to [2,3] | 35 |
| {1,2,4} | [0,1],[1,2],[2,3] | 5 | reject | 40 |

Optimal solution is J={1,2,4} with a profit of 40.

Greedy algorithm for sequencing unit time jobs with deadlines and profits:

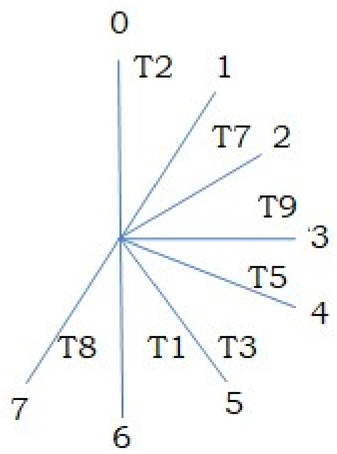
```
Algorithm JS(d, j, n)
// d[i] ≥ 1, 1 ≤ i ≤ n are the deadlines, n ≥ 1. The jobs
// are ordered such that p[1] ≥ p[2] ≥ ··· ≥ p[n]. J[i]
// is the ith job in the optimal solution, 1 ≤ i ≤ k.
// Also, at termination d[J[i]] ≤ d[J[i + 1]], 1 ≤ i < k.
{
    d[0] := J[0] := 0; // Initialize.
    J[1] := 1; // Include job 1.
    k := 1;
    for i := 2 to n do
    {
        // Consider jobs in nonincreasing order of p[i]. Find
        // position for i and check feasibility of insertion.
        r := k;
        while ((d[J[r]] > d[i]) and (d[J[r]] ≠ r)) do r := r − 1;
        if ((d[J[r]] ≤ d[i]) and (d[i] > r)) then
        {
            // Insert i into J[ ].
            for q := k to (r + 1) step −1 do J[q + 1] := J[q];
            J[r + 1] := i; k := k + 1;
        }
    }
    return k;
}
```

**Example:**

| job | d | P |
|---|---|---|
| T1 | 7 | 15 |
| T2 | 2 | 20 |
| T3 | 5 | 30 |
| T4 | 3 | 18 |
| T5 | 4 | 18 |
| T6 | 5 | 10 |
| T7 | 2 | 23 |
| T8 | 7 | 16 |
| T9 | 3 | 25 |

After arranging them in decreasing order of the profit

| Job | d | P |
| --- | --- | --- |
| T3 | 5 | 30 |
| T9 | 3 | 25 |
| T7 | 2 | 23 |
| T2 | 2 | 20 |
| T4, T5 | 3, 4 | 18,18 |
| T8 | 7 | 16 |
| T1 | 7 | 15 |
| T6 | 5 | 10 |




Total Profit = 20+23+25+18+30+15+16 = 147.

# COURSE HANDOUT

| | |
|---|---|
| Course Code | ACSC13 |
| Course Name | Design and Analysis of Algorithms |
| Class / Semester | IV SEM |
| Section | A-SECTION |
| Name of the Department | CSE-CYBER SECURITY |
| Employee ID | IARE11023 |
| Employee Name | Dr K RAJENDRA PRASAD |
| Topic Covered | knapsack problem |
| Course Outcome/s | Compare Identify suitable problem solving techniques for a given problem and finding optimized solutions using Greedy Method |
| Handout Number | 28 |
| Date | |

## Content about topic covered: Knapsack

We are given with n objects and a knapsack or bag. Object i has a weight $w_i$ and the knapsack has a capacity m. If a function $x_i$, $0 \leq x_i \leq 1$ of object i is placed in to the knapsack, then a profit of $P_i x_i$ is earned. The objective is to obtain a filling of the knapsack that maximizes the total profit earned.

Since the knapsack capacity is m, we require the total weight of the chosen objects to be atmost m.

So the problem can be stated as

$$\text{Maximize } \sum_{1 \leq i \leq n} P_i x_i \quad \cdots \ (1)$$

$$\text{Subjected to } \sum_{1 \leq i \leq n} W_i x_i \leq m \cdots \ (2)$$

$$\text{And } 0 \leq x_i \leq 1,\ 1 \leq i \leq n \quad \cdots \ (3)$$

The profits and weights are +ve numbers.

A feasible solution is any set $(x_1, x_2, x_3, \ldots, x_n)$ satisfies (2) and (3). An optimal solution is a feasible solution for which (1) is maximized.

```
Algorithm GreedyKnapsack(m,n)
// P[1:n] and W[1:n] contain profits and weights respectively of the n objects
//ordered such that (P[i]) / W[i]) ≥ (P[i + 1]) / W[i + 1]) . m is the knapsack size
//and x[1:n] is the solution vector.
{
        for i := 1 to n do
                X[i] := 0.0            //Initialize x
        U := m;
        for i:= 1 to n do
        {
                If (W[i] > U) then
                        Break;
                X[i] := 1.0;
                U:=U-W[i];
        }
        if ( i ≤ n) then
                X[i] := U/W[i];
}
```

Eg: $n = 7$, $m = 15$

$(P_1, P_2, P_3, P_4, P_5, P_6, P_7) = (10, 5, 15, 7, 6, 18, 3)$

$(W_1, W_2, W_3, W_4, W_5, W_6, W_7) = (2, 3, 5, 7, 1, 4, 1)$

$P_1/W_1 = 10/2 = 5$

$P_2/W_2 = 5/3 = 1.66$

$P_3/W_3 = 15/5 = 3$

$P_4/W_4 = 7/7 = 1$

$P_5/W_5 = 6/1 = 6$

$P_6/W_6 = 18/4 = 4.5$

$P_7/W_7 = 3/1 = 3$

Arranging them $(P[i]) / W[i]) \geq (P[i+1]) / W[i+1])$ order

$P_5/W_5 > P_1/W_1 > P_6/W_6 > P_3/W_3 \geq P_7/W_7 > P_2/W_2 > P_4/W_4$

(6) (5) (4.5) (3) (3) (1.66) (1)

$x_5 = 1$, $x_1 = 1$, $x_6 = 1$, $x_3 = 1$, $x_7 = 1$, $x_2 = 2/3$, $x_4 = 0$.

P= 6+10+18+15+3+3.33 = 55.33

All optimal solutions will fill the knapsack exactly.

The above statement is true because we can always increase the contribution of some object i by a fractional amount until the total weight is exactly m.