**Fractional Knapsack Problem**

Given a set of items, each with a weight and a value, determine a subset of items to include in a collection so that the total weight is less than or equal to a given limit (i.e., Knapsack size) and the total value is as large as possible considering you can take any item in less or equal to available quantity.

Sample Dataset:

N = 5, Knapsack Size = 60 kg

Items weights: (w1, w2, w3, w4, w5) = (5, 10, 15, 22, 25)

Items Benefits: (b1, b2, b3, b4, b5) = (30, 40, 45, 77, 90)

**Solution**

Maximum benefits: 230 units

By taking these items: ( I1 , I2 , I5 , (20 out of 22) of I4)

**Activity Selection problem**

You are given **n** activities with their start and finish times. Select the maximum number of activities that can be performed by a single person, assuming that a person can only work on a single activity at a time. [An activity can be started immediately after ending of another activity]

Start Finish Activity Name

1 2 a2

3 4 a3

0 6 a4

5 7 a5

5 9 a1

8 9 a6

**Solution**

Number of activities can be selected: 4

Activities are: (1,2), (3,4), (5,7), (8,9)

**Job Sequencing Problem**

Given an array of jobs where every job has a deadline and associated profit if the job is finished before the deadline. It is also given that every job takes a single unit of time, so the minimum possible deadline for any job is 1. How to maximize total profit if only one job can be scheduled at a time.

Input: Five Jobs with following

deadlines and profits

JobID Deadline Profit

a 2 100

b 1 19

c 2 27

d 1 25

e 3 15

Output: Following is maximum profit sequence of jobs

c, a, e

**Solving assignment with distinct difficulty**

At the start of the semester you are given n homework assignments **{a1,...,an}**. You can do the assignments in any order, but you must turn in **1** assignment per week over the **n** weeks of the semester. Assignment **ai** has a difficulty **di** (assume the difficulty values are distinct). If you turn in **ai** on week **j**, you get **di x (n - j)** points.

Considering above scenario, design and implement a greedy algorithm to maximize the number of points

**Sample Input**

N=3

Assignment ID = {A1, A2, A3}

Assignment Difficulty = {5, 3, 2}

**Sample Output**

Maximum Point = 13

Assignment Submission Order = A1, A2, A3