Lab Manual
Greedy Algorithms

1. Activity Selection Problem

Input: A set of activities S = {a_1,…, a_n}
Each activity has start time and a finish time
$a_i=(s_i, f_i)$
Two activities are compatible if and only if their time does not overlap
Output: a maximum-size subset of mutually compatible activities

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14

- $\{a_3, a_9, a_{11}\}$ can be completed
- But so can $\{a_1, a_4, a_8, a_{11}\}$ which is a larger set
- But it is not unique, consider $\{a_2, a_4, a_9, a_{11}\}$

Pseudocode

```
GREEDY-ACTIVITY-SELECTOR(s, f)
n ← length[s]
A ← {a1}
i ← 1
for m ← 2 to n
     do if s_m ≥ f_i
           then A ← A ∪ {a_m}
                i ← m
return A
```

2. Knapsack Problem

- There are *n* different items in a store
- Item *i* :
    - weighs w_i pounds
    - worth $\$v_i$
- A thief breaks in
- Can carry up to *W* pounds in his knapsack
- What should he take to maximize the value?

0-1 Knapsack Problem:

- The items cannot be divided
- Thief must take entire item or leave it behind
- Greedy strategy does not work for the 0-1 knapsack problem

Fractional Knapsack Problem:

- Thief can take partial items
- For instance, items are liquids or powders
- Solvable with a greedy algorithm

Pseudocode

```
Greedy-fractional-knapsack (w, v, W)

    FOR i =1 to n
        do x[i] =0
    weight = 0
    while weight < W
        do i = best remaining item
            IF weight + w[i] ≤ W
                then x[i] = 1
                    weight = weight + w[i]
                else
                    x[i] = (w - weight) / w[i]
                    weight = W
    return x
```

3. Counting Money

- Suppose you want to count out a certain amount of money, using the fewest possible bills and coins
- Greedy algorithm to do this would be:
  At each step, take the largest possible bill or coin that does not overshoot
  Example: To make \$6.39, you can choose:

    - a \$5 bill
    - a \$1 bill, to make \$6
    - 25¢ coin, to make \$6.25
    - A 10¢ coin, to make \$6.35
    - four 1¢ coins, to make \$6.39

For US money, the greedy algorithm always gives the optimum solution

Implement this money counting problem using Bangladeshi monetary system.

4. Connect n ropes with minimum cost

- There are given n ropes of different lengths, we need to connect these ropes into one rope. The cost to connect two ropes is equal to sum of their lengths.
- We need to connect the ropes with minimum cost.