Islamic University of Technology (IUT)

Organization of Islamic Cooperation (OIC)

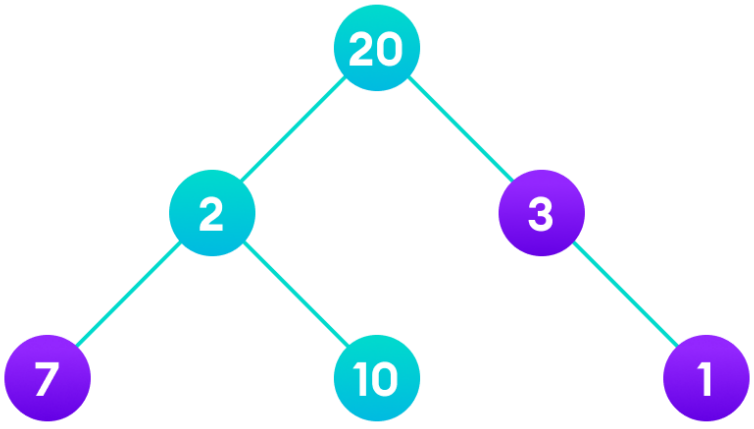
Department of Electrical and Electronic Engineering (EEE)

EEE 4416 **Lab – 09**

Greedy Algorithm

A greedy algorithm is an approach for solving a problem by selecting the *'best option' 'available at the moment'*. It doesn't worry whether the current best result will bring the overall optimal result. It always goes for the local best choice to produce the global best result. Therefore, the obtained solution may not be optimal. However, it reaches the solution much quicker than the brute force algorithm.

- Let's look at an example –



Here, you want to find the best way (maximum gain) to traverse through the nodes. Your starting point or root node has a weight of 20. From this position, you have only two ways to go. And the greedy choice is - always go for the local best. So, you move to the node with weight 3. Then, you have only one other node to go. This way, the total weight you gained is 24. However, if you had

Prepared by
ASIF NEWAZ
Lecturer, Department of EEE, IUT

gone the other way, you could have gained a much higher weight of 32. Since, the greedy algorithm only works in the top-down approach and never reverses its decision, it failed to provide the optimal solution.

20+2 = 22 --- --- --- 22+10= 32

20+3 = 23 --- --- --- 23+1 = 24

In practice, when the search space is very large, greedy choice is usually capable of providing a sub-optimal solution with much shorter time complexity, making it attractive for many applications.

- ❖ Let's look at another simplified example –

Say, you have 10 random integers. You want to find the maximum summation of any 5 integers.

Number set = [3, 1, 10, 4, 8, 5, 6, 3, 2, 5]

The brute-force approach to solve the problem will be to obtain all the combinations of 5 numbers and get their summation. The maximum of those summations will be the answer.

As you can see, in this approach, you have to search through hundreds of possible combinations to find the optimal combination. Clearly, this is not the appropriate approach.

The better way to solve the problem will be to use the greedy algorithm. Here, you make a greedy choice that you always choose the maximum number from the set. This way, the solution would look like this -

10 -> 8 -> 6 -> 5> 5 ➔ 34

You can achieve this very easily by sorting your array and taking the first five elements.

- ✓ When applying greedy algorithm, you have to make a greedy choice. Depending on the scenario, you have to identify which choice is the optimal one.

Prepared by
ASIF NEWAZ
Lecturer, Department of EEE, IUT

There are many practical problems where greedy algorithm is applied. Some popular applications of greedy algorithm include –

i. Job sequencing problem
ii. Fractional knapsack problem
iii. Minimum spanning tree (MST)
iv. Huffman Coding
v. Sequential feature selection (SFS)

We will look at a couple of examples here.

Money Change Problem

The task is to find the minimum number of coins/notes required to make up a given amount of money.

In our currency, we have notes of 1000, 500, 200, 100, 50, 20, 10, and 5. Say, in the ATM booth, you want to withdraw a certain amount of money. You have to write a program such that the ATM machine can provide that desired sum with the minimum number of notes.

For instance, if you want to withdraw 1200 tk – the atm machine can give you twelve 100 tk notes or six 200 tk notes or several other ways possible. The optimal way would be to provide one 1000 tk note along with one 200 tk note.

You have to make a greedy choice to solve the problem – always choose the highest valued note as many as possible.

Test Case - 01

- Input: 1200
- Output: 2 ➔ (1000, 200)

Test Case - 02

- Input: 4600
- Output: 6 ➔ (4*1000, 500, 100)

Test Case - 03

- Input: 1900
- Output: 4 ➔ (1000, 500, 2*200)

Prepared by
ASIF NEWAZ
Lecturer, Department of EEE, IUT

- The greedy way to solve the problem works for canonical coin system. However, in arbitrary coin system, the approach would fail. For instance,

  Coins = [1, 2, 4, 5]
  Desired sum = 8
  Greedy choice = 3 (5+2+1)
  Optimal solution = 2 (4+4)

- In scenarios like this, you need to consider another popular algorithm – "Dynamic Programming".

Job Sequencing Problem

You are given a list of jobs and their associated profits along with their deadline. Each job takes a single day to complete. Your task is to schedule the jobs in such a way that maximizes the profit.

Example - 01

Jobs	a	b	c	d
Deadline	4	1	1	1
Profit	20	10	40	30

Optimal solution: c, a.

Example - 02

Jobs	a	b	c	d	e
Deadline	2	1	2	1	3
Profit	100	19	27	25	15

Prepared by
ASIF NEWAZ
Lecturer, Department of EEE, IUT

Optimal solution: c, a, e

Example - 03

Jobs	a	b	c	d	e	f
Deadline	5	3	3	2	4	2
Profit	200	180	190	300	120	100

Optimal solution: b, d, c, e, a

In order to solve problems like these, you need to come up with an algorithm that fulfils the objective.

Objective – maximize the profit.

To realize the objective, you need to focus on the 'profit' parameter; not the deadline. You need to complete the jobs in such a way that profit can be maximized. It need not consider how many tasks you can complete.

For instance, in example – 2, for both jobs 'b' and 'd', day – 1 is the deadline. But profit from those jobs is small. Job 'c' provides comparatively higher profit than 'b' and 'd' but lower than 'a'. Job 'a' provides the highest profit and it can be completed by day 2. If you take this job on the first day, then you lose the opportunity to undertake the other jobs that can only be completed on day 1 and provide some profits.

So, you can see, you need to consider several associated parameters and come up with an algorithm that takes care of all these contradictory factors.

In scenarios like these, you can make what is called a 'greedy choice'.

Approach

I. Create an empty set (array) with a length = maximum deadline. This set represents the jobs that we can complete. We can do maximum 3 jobs here.

out			

Prepared by
ASIF NEWAZ
Lecturer, Department of EEE, IUT

II. Sort the given matrix based on the profit in descending order.

Jobs	a	c	d	b	e
Deadline	2	2	1	1	3
Profit	100	27	25	19	15

III. Take one job at a time from the sorted matrix sequentially and place the jobs on the 'out' array. The jobs have to be placed on the 'last' possible deadline.

For instance, job 'a' can be done on day 1 or day 2. You are making the choice to do it on the last day (greedy choice), keeping the first day available for another job (that will offer lower profit than a).

out		a	

out	c	a	

out	c	a	e

This is the way you will obtain the maximum profit. You can try all other possible combinations – and you will see, this is the optimal one.

Class work

Solve example 3 in a similar way.

Prepared by
ASIF NEWAZ
Lecturer, Department of EEE, IUT