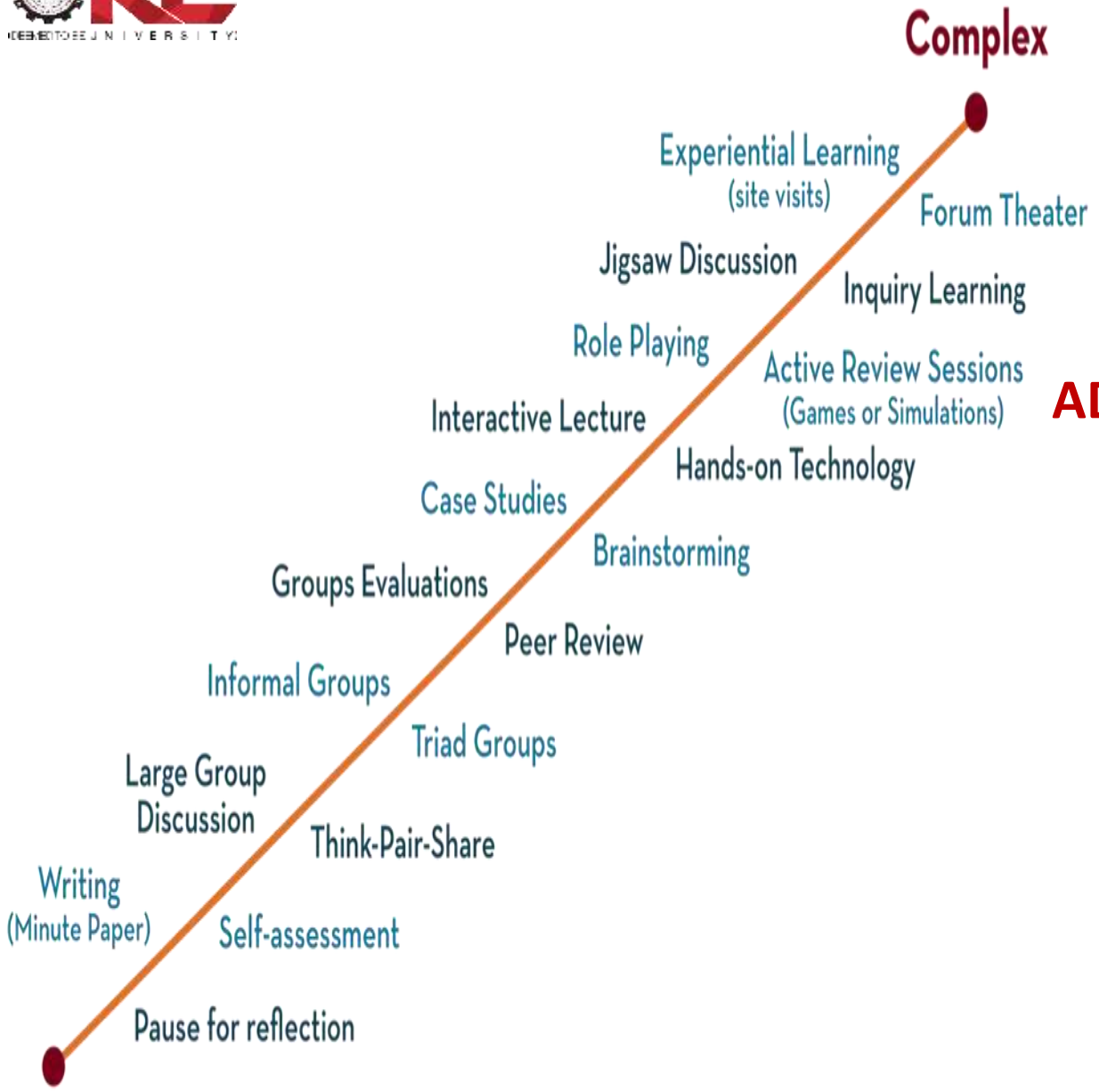
Advanced Algorithms & Data Structures



Department of CSE

ADVANCED ALGORITHMS AND DATA STRUCTURES
23CS03HF

Topic:

Greedy Method

AIM OF THE SESSION

To familiarize students with the concept of Greedy Method

INSTRUCTIONAL OBJECTIVES

This Session is designed to:

1.Demonstrate :- Greedy Method.

2.Describe :- The optimization problem, Types in Greedy strategy.

LEARNING OUTCOMES

At the end of this session, you should be able to:

1. Define :- Greedy Method.
2. Describe :- The optimization problem, Types in Greedy strategy
3. Summarize:- Comparision between Divide and Conquer and Greedy Method.

Introduction to Optimization problem

Optimization problems are those for which the objective is to Maximize or Minimize some values.

For example,

- Finding the minimum number of colors needed to color a given graph.
- Finding the shortest path between two vertices in a graph

Greedy Method

- Greedy method solves problems by making the best choice that seems best at the particular moment.

- In some real-world games we need to think about the future before taking a decision and some games require a decision that is best at that moment.

  Example : In chess game, every time we make a decision we need to think about the future consequences as well, where as, in Tennis or volleyball game, we make a decision which looks good at that situation.

- In computers, to solve some kind of problems, we need an algorithm that need to take a decision which looks good at that situation and there comes our greedy algorithm.

- Decisions are made based on Greedy Choice Property

Greedy Approach

- Greedy algorithm obtains an Optimal solution by making a sequence of decisions.
- Every greedy based problems will be given with a set of inputs and a set of constraints.
- Our objective is to find a solution vector which satisfies a set of constraints.
- Decisions are made one by one in some order.
- Each decision is made using a greedy-choice property or greedy criterion.

Greedy Choice Property

- Global Optimal Solutions will be made with local optimal (Greedy) choices
- In Greedy algorithm, the best choice will be selected at the moment to solve the sub problem that remains
- The choice of the greedy may depend on the previous choices made but it cannot depend on future choices or on the solutions to the sub problems

- A feasible solution is any subset of original input that satisfies a given set of constraints.
- A objective function is an input for which a feasible solution is to be obtained that either maximizes or minimizes.
- An optimal solution is a feasible solution which maximizes or minimizes the objective function.
- Irrevocable - Once the choices are made in a particular step , it can not be changed in subsequent steps.

Algorithm For Greedy Method Control Abstraction

```
Greedy(a,n)            // a[1:n] contains the n inputs
{
    solution= φ    // Initialize solution
     for i=1 to n do
     {
          x := Select(a);
          if  Feasible(solution , x) then
               solution=Union(solution , x)
     }
   return solution;
}
```

Counting Coins Problem :

This problem is to count to a desired value by choosing the least possible coins and the greedy approach forces the algorithm to pick the largest possible coin. If we are provided coins of **₹ 1, 2, 5 and 10** and we are asked to count **₹18** then the greedy procedure will be,

1 – Select one ₹ 10 coin, the remaining count is 8
2 – Then select one ₹ 5 coin, the remaining count is 3
3 – Then select one ₹ 2 coin, the remaining count is 1
4 – And finally, the selection of one ₹ 1 coins solves the problem

•Though, it seems to be working fine, for this count we need to pick only 4 coins. But if we **slightly change** the problem then the same approach may not be able to produce the same optimum result.
•For the currency system, where we have coins of **1, 7, 10** value, counting coins for value **18** will be absolutely optimum but **for count like 15**, it may **use more coins than necessary.** For example, the greedy approach will use 10 + 1 + 1 + 1 + 1 + 1, total 6 coins. Whereas the same problem could be solved by using only 3 coins (7 + 7 + 1)

•Hence, we may conclude that the greedy approach picks an immediate optimized solution and may fail where global optimization is a major concern.

Greedy Strategy

Subset Paradigm:

- The selection procedure itself based on some optimization measures

***Example*:**

- Knapsack problem
- job sequencing with deadlines

Order Paradigm:

- The selection procedure does not depends on optimization measures. But, decisions by considering the inputs some particular order.

***Example*:**

- Minimum Spanning tree (Prim's and Kruskal's method))

Types of Greedy Problems

- **Subset Paradigm:** To solve a problem (or possibly find the optimal/best solution), greedy approach generate subset by selecting one or more available choices.

  Example:

  - Knapsack problem
  - Job sequencing with deadlines

- **Order Paradigm:** greedy approach generate some arrangement/order to get the best solution

  Example:

  •Single source shortest path problem

Three Important Activities

1. **Selection:** Selection of solution from a[] and removes it
2. **Feasibility:** Feasible(solution,x) is a Boolean function to determine whether x can be included into the solution vector
3. **Optimality:** From the set of feasible solutions, the particular solution which minimizes or maximizes the given objective function

Applications of Greedy Method

- Knapsack Problem
- Job Sequencing with deadlines
- Minimum cost spanning tree
- Huffman Codes
- Single source shortest path problem

Divide and conquer	Greedy Algorithm
Divide and conquer is used to find the solution, it does not aim for the optimal solution.	A greedy algorithm is optimization technique. It tries to find an optimal solution from the set of feasible solutions
DC approach divides the problem into small sub problems, each sub problem is solved independently and solutions of the smaller problems are combined to find the solution of the large problem.	In greedy approach, the optimal solution is obtained from a set of feasible solutions.
Sub problems are independent, so DC might solve same sub problem multiple time.	Greedy algorithm does not consider the previously solved instance again, thus it avoids the re-computation.
DC approach is recursive in nature, so it is slower and inefficient.	Greedy algorithms are iterative in nature and hence faster.
Divide and conquer algorithms mostly runs in polynomial time	Greedy algorithms also run in polynomial time but takes less time than Divide and conquer
Example: Merge sort, Quick sort	Example: Knapsack problem, Job scheduling problem

SUMMARY

- **Selection**: Make decisions based on the best available option at each step without reconsidering previous choices.

- **Locality**: Optimize locally with the hope of finding a global optimum.

- **Efficiency**: Efficient for problems where making locally optimal choices leads to an optimal solution overall, but may not guarantee the best solution in all cases.

SELF-ASSESSMENT QUESTIONS

What type of problems are well-suited for the greedy method?

(a) Problems with complex constraints

(b) Problems requiring global optimum solutions

(c) Problems where local optimization leads to a feasible solution

(d) Problems involving exhaustive search techniques

Which step is crucial in the greedy method to ensure that each chosen option contributes to a feasible solution?

(a) Sorting choices in ascending order

(b) Evaluating all possible subsets

(c) Considering constraints and requirements

(d) Using dynamic programming techniques

TERMINAL QUESTIONS

1. Describe the main ideas behind greedy algorithms.
2. List out the various types in Greedy strategy.
3. Analyze the time complexity of fractional knapsack problem

REFERENCES FOR FURTHER LEARNING OF THE SESSION

Reference Books :

1. Introduction to Algorithms, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein., 3rd, 2009, The MIT Press.
2 Algorithm Design Manual, Steven S. Skiena., 2nd, 2008, Springer.
3 Data Structures and Algorithms in Python, Michael T. Goodrich, Roberto Tamassia, and Michael H. Goldwasser., 2nd, 2013, Wiley.
4 The Art of Computer Programming, Donald E. Knuth, 3rd, 1997, Addison-Wesley Professiona.

MOOCS :

1. https://www.coursera.org/specializations/algorithms?=
2.https://www.coursera.org/learn/dynamic-programming-greedy-algorithms#modules

THANK YOU