07/08/23

DATE : / /
PAGE NO. : 1

Algorithms and Problem Solving
Assignment 1.

Q1 What are algorithms? Explain algorithm as technology with example.

A (I) The algorithm is defined as a collection of unambiguous instructions occurring in some specific sequence and such an algorithm should produce output for given set of input in finite amount of time.

(II) It is not the complete program or code; it is just a solution (logic) of a problem, which can be represented either as an informal description using a Flowchart or Pseudocode.

(III) Properties of Algorithm

(i) Non ambiguity
Each step in an algorithm should be non-ambiguous. Each instruction should be clear and precise. The instructions should not denote any conflicting meaning.

(ii) Range of input
The range of input should be specified to prevent the algorithm from going into an infinite state.

(iii) Multiplicity
The same algorithm can be represented in several different ways. In other words, we define the algorithms. It can be in simple English sentences or in the form of pseudocode.

(iv) Speed
The algorithm is written using some specific ideas to ensure that the algorithm is efficient and delivers output as fast as it can.

(v) Finiteness

The algorithm should be finite in nature i.e. it should terminate at some point in time.

(IV) Algorithms as Technology

(I) Algorithms are used as a tool to utilize reasonable amount of execution time and memory space.

(II) Different algorithms can be used to solve the exact same problem; they would vary in execution time.

(III) The execution depends upon two factors -

(1) Selection of efficient algorithm

(2) Fast hardware upon which the algorithm is to be executed.

(IV) Algorithms are core or fundamental for the following technologies:

- Advanced computer architectures
- Object oriented systems
- Fast networking applications that can be wired or wireless
- Applications that use Graphical User Interface (GUI)
- Integrated web technologies.

Q2 What is iterative algorithm in design and analysis of algorithms?

A (E)

(I) Iteration refers to repeating a certain number of steps continuously until a particular condition is met successfully.

(II) Iterations play a very important role in performing the same instructions and for repeating the same line of codes over one another.

(III) Sometimes iteration is worthless and creates a deadlock when the condition falls in the an infinite loop.

(IV) Iterative algorithm has at least one iterative component or loop. Hence the part of algorithmic statements will be executed for n number of times

(V) Even small amount of time spent on execution of loop will directly affect the efficiency of overall algorithm.

(VI) The design issues of the iterative algorithms are :
(i) Iteration using the loop control structure.
(ii) Improving the efficiency of the algorithm.
(iii) Estimation of time and space requirements.
(iv) Expressing the complexities using order notation.
(v) Applying different algorithmic strategies.

(VII) For any iterative loop, we specify :
(i) The initial condition that is set to be TRUE before the beginning of the loop
(ii) The invariant relation that must hold before, during, and after each iteration of the loop.
(iii) The terminating condition that specifies the condition for which the loop must terminate.

Q3 Compare the following : ① Divide and Conquer and Dynamic Programming
② Greedy method and Dynamic Programming

A ①

Divide and Conquer Method	Dynamic Programming
(i) It has 3 steps at each level : ① Divide the problem into a number of subproblems. ② Conquer the subproblems by solving them recursively ③ Combine the solution to the subsolution for the subproblems.	(i) It has 4 steps ① Characterize the structure of optimal solutions ② Recursively define the values of optimal solutions. ③ Compute the value of the optimal solutions ④ Construct an optimal solution from the computed information

	Divide and Conquer Method	Dynamic Programming
(ii)	It is recursive.	It is non recursive.
(iii)	It does more work on subproblems and hence has more time consumption.	It solves subproblems only one and then stores in the table.
(iv)	It is a top-down approach.	It is a bottom up approach.
(v)	Subproblems are indipendent.	Subproblems are interdipendent.
(vi)	Example: Merge sort, binary search, etc.	Example: matrix chain multiplication, 0/1 Knapsack problem, etc.

(2)	Greedy Method	Dynamic Programming
(i)	We make the choice based on current outlook of the problem and then solve the sub-problems arising after the choice is made.	We choose at each step but the choice may depend on the solution to sub-problems.
(ii)	More efficient than divide and conquer and dynamic programming.	Less efficient than greedy.
(iii)	Iterative in nature.	Recursive in nature.
(iv)	Optimal solution is NOT guaranteed.	Optimal solution is guaranteed using the Principle of Optimality.

DATE : / /
PAGE NO. : 5

	Greedy Method		Dynamic Programming
(v)	Extra memory is not required	(v)	More memory is required to store the subproblems for later use.
(vi)	Example: Fractional Knapsack problem, Job sequence, etc	(vi)	Example: 0/1 Knapsack, Matrix chain multiplication, etc.

Q4 What is the need for the correctness of an algorithm?

A We need to check the correctness of algorithm because:

(I) To ensure that the algorithm is developed to correctly satisfy the requirements.

(II) To ensure that the solution for the given problem is valid.

(III) To check whether the given problem has finite or infinite solutions.

(IV) To ensure the efficient execution of the solution.

(V) To ensure that all the cases in the problem statements are covered.

Basic steps in algorithmic correctness

(I) Identification of the properties of input data. These properties of data are preconditions

(II) Identification of the properties which must be satisfied by the output data. These properties are called postconditions.

(III) Proving that starting from preconditions and executing each step specified in the algorithms, one obtains the postconditions.

Q5 Explain Algorithm Design Techniques.

A (I) Designing an efficient algorithm to solve a computational problem is a skill that needs good algorithmic thinking.

(II) A good algorithm should possess 5 major characteristics:

(1) Input
(2) Output
(3) Definiteness
(4) Finiteness
(5) Effectiveness.

(III) Basic Steps to Design an Algorithm:

(I) Understand and analyse the problem to be solved.
(II) Name the algorithm properly.
(III) Select the appropriate algorithmic strategy to solve the problem by analysing the characteristic nature of the problem.
~~(II)~~ (IV) Identify the legitimate inputs of an algorithm.
~~(III)~~ (V) Identify the expected outputs of an algorithm.
~~(IV)~~ (VI) Decide the suitable data structure to define inputs and to present the output.
(VII) Describe a finite set of well ordered unambiguous instructions to produce the expected output.

(IV) Some popular design approaches:

(I) Divide and Conquer
It involves dividing the problem into sub problems, recursively solving them, and then recombining them for the final answer.

(II) Greedy Method
At each step, a decision is made to choose the local optimum without thinking about the future consequences.

(iii) Dynamic Programming
It is similar to divide and conquer, the difference being that whenever we have recursive function calls with the same result, instead of calling them again, we try to store the result in a data structure.

(iv) Linear Programming
There are inequalities in terms of input and maximizing or minimizing some linear functions of inputs is carried out.

(v) Backtracking
While exploring an option if a point is reached that doesn't lead to the solution, the program control backtracks one step and starts exploring the next option. In this way, the program explores all possible courses and finds the route that leads to the solution.

(vi) Branch and Bound.
In this approach, the entire solution space is represented in the form of a state space tree. As the ~~prob~~ program progresses, each state combination is explored and the previous solution is replaced by a new one if the newer solution turns out to be more optimal.

Q6 Explain the classification of Time Complexities.

A

	Name	Order	Description	Example
(i)	Constant	1	As input size grows, we get larger running time	Scanning array elements.
(ii)	Linear	n	The running time of algorithm depends on the input size n	Performing sequential search operations.

DATE : / /
PAGE NO. :

	Name	Order	Description	Example
(iii)	Logarithmic	$\log n$	The algorithm does not consider all its inputs; the problem is divided into smaller parts on each iteration	Performing binary search
(iv)	nlogn	$n \log n$	Some instance of input is considered for the list of size n	Performing sequential search
(v)	Quadratic	n^2	When the algorithm has two nested loops, then this efficiency occurs	Scanning matrix elements.
(vi)	Cubic	n^3	When the algorithm has 3 nested loops, then this efficiency occurs	Performing matrix multiplication
(vii)	Exponential	a^n	When the algorithm has a very fast rate of growth, then this efficiency occurs	Generating all subsets of n elements
(viii)	Factorial	$n!$	When an algorithm is computing all the permutations then this efficiency occurs.	Generating all permutations.

For reference purposes only. Not liable for any misuse or misinterpretation.

We're interested in providing notes and assignments for free because college is more than just about submissions! :D

Thank you for all your support!