DESIGN AND ANALYSIS OF ALGORITHMS

QUESTION BANK

UNIT1: INTRODUCTION

OBJECTIVE: Algorithms play the central role in both the science and the practice of computing. There are compelling reasons to study algorithms. This Unit is broadly divided into four sections. The first section deals with the notion of algorithm. The second deals with algorithmic problem solving. Several issues related to the design of and analysis of algorithms is discussed. The third section is devoted to a few problem types that have proven to be particularly important to the study of algorithms and their applications. Finally the fourth section contains a review of fundamental data structures.

1. What is an algorithm? 2
2. Explain the Euclid’s algorithm for computing GCD. 5
3. Explain the consecutive integer checking algorithm for computing GCD. 5
4. Explain the Sieve’s algorithm for generating the prime numbers. 5
5. Explain the sorting and searching problem. 10
6. With help of a neat flow diagram, explain the Algorithm design and analysis process. Discuss the various stages of algorithm design and analysis process using flow chart. 10
7. What is string processing problem? 5
8. Give a detailed description of an algorithm for transforming a free tree into a tree rooted at a given vertex of the free tree. 10
9. What are combinatorial problems? 5
10. What are Geometric and numerical problems? 5
11. Describe the standard algorithm for finding the binary representation of a positive decimal integer in
    a. English
    b. in a pseudo code 10
12. What are the ways in which you can classify algorithms? 5
13. Compare the Euclid’s algorithm and consecutive integer algorithm for computing GCD. 10
14. Explain the various linear data structures. 10
15. Define a graph and the terminologies used in graphs. 10
16. Explain the various graph representations. 10
17. Define tree and the terminologies used in trees. 10
18. What are sets and dictionaries? 10
19. Discuss the sequence of steps one goes through in designing and analyzing an algorithm. 20
20. a. Let A be the adjacency matrix of undirected graph. Explain what property of the matrix indicates that
    i. Graph is complete.  ii. Graph has a loop, i.e. a edge connecting a vertex to itself.
    iii. Graph has an isolated vertex, i.e., a vertex with no edges incident upon it. 10
21. Answer the same questions (Q. No 20) for the adjacency linked list representation. 10
22. Explain important fundamental problem types of different category 10
23. Explain how priority Queue can be implemented as unsorted array 06
24. What are rooted trees explain and differentiate the same with ordered trees 20
25. Explain the concept of multi set, bags, dictionaries 10
26. When do you say an algorithm is in place and why? 02
27. Define the following: 1. Algorithmic Algorithm, sequential algorithm, Parallel Algorithm, ADT, pseudo code, flow chart, Time efficiency, Space efficiency
28. Find GCD(60,24) by applying Euclid’s formula. Estimate the number of times computation is done in Euclid’s method and in an algorithm based on checking consecutive integers from min(m,n) down to gcd(m,n).

FUNDAMENTALS OF THE ANALYSIS OF ALGORITHM EFFICIENCY

OBJECTIVE: This chapter deals with the analysis of algorithms. It starts with the general framework of analyzing algorithm efficiency. The three notations $O$, $Ω$, and $q$ are introduced. These notations have become the language for discussing algorithm efficiency. The general framework outlined earlier is then applied to analyzing the efficiency of recursive and non-recursive algorithms.

29. Discuss the time efficiency and space efficiency of an algorithm. Explain methods for representing them.
30. How does one measure an input’s size and running time?
31. Show how operation counts and step count methods are used to determine the time complexity of a program. Derive operation counts for best, worst and average cases.
32. What is amortized efficiency?
33. What are homogenous recurrences?
34. Explain the various asymptotic notations with examples. Give examples for each.
35. What is the general plan for analyzing efficiency of non-recursive algorithms?
36. What is order of growth? Explain the same by considering $n=10^*10, 10$ for $\log n, n\log n, 2^n$.
37. Write an algorithm finding the max element in an array of numbers. Analyze its efficiency.
38. Write an algorithm to check whether all elements in the array are distinct (element uniqueness problem). Analyze its efficiency.
39. Analyze the efficiency of the matrix multiplication problem.
40. What is the general plan for analyzing efficiency of recursive algorithms? Suggest a recursive algorithm to find factorial of a number. Derive its efficiency.
41. Analyze the recursive program for tower of Hanoi problem.
42. Analyze the recursive program for computing the factorial of an arbitrary number.
43. Analyze the recursive program to find the sum of first $n$ cubes.
44. Write an algorithm for computing Fibonacci series. Analyze the algorithm.
45. Write the bubble sort algorithm and show that the worst case efficiency is quadratic.
46. If $T_1(n)=O(f(n))$ and $T_2(n)=O(g(n))$, then show that $T_1(n)+T_2(n)=O(\max(f(n),g(n)))$
47. Solve the recurrence relations: $T(n) = \begin{cases} aT(n/c)+bn & \text{for } n>1 \\ b & \text{for } n=1 \end{cases}$

48. Consider the following algorithm:

```
ALGORITHM Mystery(n)
//input: A nonnegative integer n
S = 0
for i = 1 to n do
    S = S + i*i
return S
```

i) What does this algorithm compute?
ii) What is its basic operation?
iii) How many times is the basic operation executed?

iv) What is the efficiency class of this algorithm?

49. Explain all the mathematical notations used for the analysis of an algorithm

50. Order the following functions according to their order of growth (from lowest to highest) \((n-2)!\), \(5\log_{10}(n+10)^{10}\), \(2^{2n}\), \(\log_{2}n\), \(\sqrt{n}\), \(3^{n}\).

51. Solve the following recurrence relations: 
   - \(x(n) = 3x(n-1)\) for \(n > 1\), \(x(1) = 4\) and 
   - \(x(n) = x(n/2) + n\) for \(n > 1\), \(x(1) = 1\), \(n = 2^k\). 

52. Explain in brief the basic asymptotic efficiency classes.

53. Explain the method of comparing the order of the growth of two functions using limits. Compare order of growth of following functions:
   - i) \(\log n\) and \(\sqrt{n}\)
   - ii) \((\log 2 n)^2\) and \(\log 2 n\) square

54. Write the algorithm for sequential search and find its best and worst case efficiency

55. Order the following functions according to their order of growth (from lowest to highest) \(\frac{1}{2n(n-1)}\), \(\text{npow2}\), \(\log n\), \(\text{npow1/2}\), \(n!\) and \(2\text{pown}\) respectively

56. Write a note on basic efficiency classes

57. Define Big-oh, Omega and Theta notation. Give examples for each.

58. Explain the analysis framework of algorithms. Explain the worst case, best case and average case efficiencies, with an algorithm

59. If \(t_1(n) \in O(g_1(n))\) and \(t_2(n) \in O(g_2(n))\), prove that \(t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})\).

60. Suggest a general plan for analyzing the efficiency of non-recursive algorithms. Apply the plan to analyze the following algorithms:
   - i. Algorithm to check uniqueness of elements.
   - ii. Algorithm to multiply matrices

61. Suggest a general plan for analyzing the efficiency of recursive algorithms. Apply the plan to analyze the efficiency of following algorithms:
   - i. Recursive algorithm for Tower of Hanoi problem
   - ii. Recursive algorithm to find the factorial of \(N\).

**BRUTE FORCE**

OBJECTIVE: Brute force is a straightforward approach to solving a problem. This unit discusses the various brute force algorithms like sorting, searching etc. This unit also discusses exhaustive search, which is a brute approach to combinatorial problems like knapsack, traveling salesman, assignment problem.

62. What is meant by a brute force method? Example sequential search algorithm with an example. Analyze its efficiency.


64. Outline an exhaustive search algorithm for the TSP. What is the efficiency class of this algorithm? Illustrate with an example.

65. Explain the selection sort algorithm. Analyze its efficiency.

66. Write a bubble sort algorithm and show that the worst case efficiency is quadratic.

67. Write an algorithm for sequential search. Determine its worst case, best case & average case efficiency.

68. Write an algorithm for selection sort.
   - i) Is selection sort stable?
ii) Write the property which distinguishes select sort positively from other sorting algorithms. 10
iii) Is it possible to implement selection sort for linked list with the same efficiency as the array version. 69.

69. Explain the brute force method for algorithm analysis and design. Explain the brute force string matching algorithm with an example. Give its efficiencies. 10
70. Explain the knapsack and assignment problem wrt the brute force method 10
71. What is the Hungarian method? Explain 05
72. Find the no of character comparisons that will be made by the straight forward string matching for the pattern ABABC in the following text: BAABABABCCA 10
73. Find the no of comparisons required to search for 6 in the given sequence of numbers: 10, 19, 7, 9, 6, 15 05
74. Solve the Knapsack for the following data 12

UNIT II: DIVIDE AND CONQUER

OBJECTIVE: Divide and conquer approach divides the problem into smaller sub problems. The solutions of sub problems are combined to get the solution. This unit discusses the divide and conquer algorithms like Binary search, Merge sort and Quick sort. This analyzes the efficiency of divide and conquer approaches.

Determine the efficiency of divide and conquer algorithms. 05

1. Explain and Analyze the merge sort algorithm. 12
2. How quick sort can be improved? 05
3. Explain the binary searching algorithm in detail, with an example. Show the worst case efficiency of binary search is in \( \Theta(\log n) \) 10
4. Write an algorithm for leaf count. 05
5. Explain the divide and conquer strategy with examples. 10
6. Apply quick sort to sort the list E, X, A, M, P, L, E in alphabetical order. Draw the tree of the recursive calls made. 12
   i. Write the best case input for the quick sort
   ii. Find the best case time efficiency for the quick sort
   iii. Are the arrays made up of all equal elements the worst case input, the best case input or neither?
7. Write a pseudo code for divide & conquer algorithm for finding the position of the largest element in an array of numbers. 10
8. Write a quick sort algorithm and derive the worst case and average case complexity class of this algorithm. 10
9. Discuss the efficiency of quick sort algorithm. 10
10. Explain how the merge sort can be viewed as a recursive application of the Divide and conquer methodology. Suggest a pseudo code for merge sort and analyze its complexities. Trace its application to the following data set: 9, 4, 3, 8, 6, 2, 1, 5, 7. 10
11. Explain a recursive algorithm to compute the height of a binary tree. 10
12. Explain the multiplication of large integers using divide and conquer. 10
13. Explain Strassen’s matrix multiplication. Evaluate it’s efficiency. 10
14. Propose a divide and conquer strategy-based procedure to procedure to search a key in a set of n elements. Demonstrate the process to 18 in 6, 8, 15, 18, 22, 23. 8
15. Using divide-and-conquer approach, write an algorithm for Quicksort and derive the best, worst, and average complexity of the same. 10
16. Write Binary Search algorithm and determine its time complexity. 10
17. Design and analyze the algorithm for tiling the defective chess board using triominoes.  

UNIT III : GREEDY TECHNIQUE

OBJECTIVE: The greedy technique approach suggests constructing a solution through a sequence of steps, each expanding a partially constructed solution obtained so far, until a complete solution to the problem is reached. This chapter discusses algorithms that use this technique. Some of them are Prims and Kruskal’s algorithm to find minimum spanning tree, Dijkstra’s algorithm for the shortest path problem, Huffman trees and their applications, and Huffman codes.

1. Explain Greedy Technique.  
2. Write Prim’s Algorithm to construct a minimum spanning tree.  
3. Explain Kruskal’s Algorithm With an example.  
4. Explain Dijkstra’s Algorithm With an example.  
5. Write a note on Huffman Trees and Huffman code.  
6. Construct a Huffman code for the following data:  
   
<table>
<thead>
<tr>
<th>Character</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probability</td>
<td>0.4</td>
<td>0.1</td>
<td>0.2</td>
<td>0.15</td>
<td>0.15</td>
</tr>
</tbody>
</table>

7. Write Dijkstra’s algorithm for single source shortest path? What modifications are required to the above algorithm so that it calculates all pair shortest paths?  
8. What are decision trees? Explain the concept of decision trees for sorting algorithms with an example.  
9. Write and explain Dijkstra’s algorithm for single source shortest distance problem. Also derive the time complexity of it.  
10. i) Write Prim’s algorithm to find the minimum cost spanning tree.  
    ii) Using Prim’s algorithm, determine minimum cost spanning tree for the weighted graph shown below.

![Weighted Graph](image)

11. i) Write Kruskal’s algorithm to find the minimum cost spanning tree.  
    ii) Using Kruskal’s algorithm, determine minimum cost spanning tree for the weighted graph shown in the previous question.  
13. Explain how find and union operations of Kruskal’s algorithm is implemented.
UNIT IV: DYNAMIC PROGRAMMING

OBJECTIVE: Dynamic programming is a technique for solving problems with overlapping sub problems. These sub problems arise from a recurrence relation relating a solution to a given problem with solutions to its smaller sub problems of the same type. Rather than solving overlapping sub problems again and again, dynamic programming suggests solving each of the smaller sub problems only once and recording the results in a table from which we can then obtain a solution to the original problem. Some of the algorithms discussed here are computing a binomial coefficient, Warshall’s algorithm, Flyod’s algorithm, Knapsack problem.

1. What is dynamic programming? Design an algorithm to solve the 0/1 knapsack problem using Dynamic programming.  

2. Write Warshall’s algorithm and apply it to compute transitive closure for the directed graph with the adjacency matrix shown below:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

3. Solve the following TSP using Dynamic programming which is represented as cost adjacency matrix of a directed graph.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>10</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>0</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>13</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>8</td>
<td>9</td>
<td>0</td>
</tr>
</tbody>
</table>

4. Define transitive closure of a directed graph.

5. Write and explain an algorithm to solve the non crossing subset of nets problem using dynamic programming.

6. What is the efficiency of binomial coefficient algorithm?

7. Compare divide and conquer strategy with dynamic programming.

8. What is the efficiency of Warshall’s algorithm?

9. What is the time and space efficiency of the algorithm for knapsack problem?

10. Explain the algorithm to find binomial coefficient.

11. Explain the Warshall’s algorithm with an example.

12. Explain the dynamic programming with Floyd’s algorithm in detail. Apply Floyd’s all pairs shortest problem. For the digraph given below.

Department of ISE
UNIT V: DECREASE AND CONQUER & SPACE TIME TRADEOFFS

DECREASE AND CONQUER

OBJECTIVE: Decrease and conquer technique is based on exploiting the relationship between a solution to a given instance of a problem and a solution to a smaller instance of the same problem. Once such relationship is established, it can be exploited either top down or bottom up. This part discusses the three major variations of decrease and conquers namely a) decrease by a constant b) decrease by constant factor c) variable size decrease and also the examples associated with them.

1. Suggest pseudo codes for i) depth first search and ii) Breadth first search Illustrate with examples. 12

2. Define a) digraph b) directed acyclic graph. 05

3. Write an algorithm to topologically sort an digraph using DFS. Prove the correctness and find time efficiency. 10

4. What is efficiency of DFS based algorithm for topological sorting? 05

5. Explain the decrease and conquer strategy and its variations. 10

6. i) Write an algorithm for Insertion sort. 10
   ii) Derive Best case and worst case efficiency for the same algorithm
   iii) Show how Insertion sort algorithm arranges the following members in increasing order. 61 28 9 85 34

7. Explain the concept of decrease and conquer methodology, indicating the three major variations of the same. 08

8. Explain the efficiency of DFS and BFS Algorithm. 10

9. What are the different applications of DFS and BFS? 05

10. i) Write an algorithm to traverse the graph using BFS. 10
    ii) Traverse the following using BFS and construct corresponding BFS tree.
11. i) Write an algorithm to traverse the graph using DFS
   ii) Traverse the graph of the previous question using DFS and construct corresponding DFS tree.

12. Apply the DFS based algorithm to solve the topological problem for the following graph.
OBJECTIVE: space and time tradeoffs in algorithm design are a well known issue for both theoreticians and practitioners of computing. This unit discusses an approach to problem solving called input enhancement and examines the algorithms like counting method for sorting, Boyer Moore algorithm for string matching. It also discusses the other approach called presorting and this approach is illustrated by hashing and indexing with B-trees.

13. Explain the Horspool’s string matching algorithm for a text that comprises English letters and spaces (denoted by underscore) with a pattern BARBER. Explain all the cases of Horspool algorithm and give its efficiency. 10
14. What is distributed counting and pre structuring approach? 04
15. What is input enhancement approach? What algorithms are based on this approach? 05
16. What is hashing? Explain the open addressing method of hashing to insert the text “A FOOL AND HIS MONY ARE SOON PARTED” in a hash table and delete the word “SOON” from the i/p data [Hash table size = 13] 10
17. Explain the comparison counting algorithm. What is the efficiency of this algorithm? 10
18. Explain the distributed counting algorithm. What is the efficiency of this algorithm? 10
19. Sort the following list in alphabetical order by distributing counting algorithm b, c, d, c, b, a, a, b. 10
20. Explain the algorithm for computing the shift table entries. 10
21. Explain the Boyer Moore algorithm. 10
22. Apply horspool’s algorithm to search for the pattern BAOBAB in the BESS-KNEW-ABOUT-BAOBABS. 10
23. Explain the open hashing and closed hashing method. 20
24. How many character comparisons will be Boyer-Moore algorithm make in searching for each of the following patterns in the binary text of 1000 zeros?
   a. 00001  b. 10000  c. 01010  6
25. Consider the problem of searching for genes in DNA sequence using horspool’s algorithm, A DNA sequence is represented by a text on the alphabet {A,C,G,T} & the gene or the gene segment is the pattern. 20
   a) construct the shift table for the following gene segment of your chromosome 10 TCCTATTCTT
   b) Apply Horspool’s algorithm to locate the pattern in the following DNA sequence
      TTATAGATCTCTGTTTATAGATCTCCTATTCT
26. For the input 30, 20, 56, 75, 31, 19 and the hash function h(k) = k mod 11 20
   a) Construct the open hash table.
   b) Find the largest and the average number of comparisons in a successful search in this table.
27. Write Horspool’s algorithm. Apply Horspool algorithm to search for the pattern BAOBAB in the text BESS_KNEW_ABOUT_BAOBABA. 10

UNIT VI: LIMITATIONS OF ALGORITHMIC POWER AND COPING WITH THEM

LIMITATIONS OF ALGORITHMIC POWER
OBJECTIVE: This unit deals with the fair assessment of the algorithms as problem solving tool. It starts with the methods for obtaining lower bounds i.e estimates on a minimum amount of work needed
to solve a problem. The next section deals with decision trees. This technique allows us to establish lower bounds on efficiency of comparison based algorithms for sorting and for searching in sorted arrays. This unit also discusses about P NP and NP Complete problems.

1. Explain briefly the limits of algorithm power. 05
2. What are information theoretic arguments? 05
3. Obtain a decision tree for finding minimum of 3 numbers. 05
4. What do you mean by lower bound of an algorithm? How do you obtain a lower bound of an algorithm? 10
5. Explain the adversary method for establishing lower bounds. 10
6. Find a trivial lower-bound class for each of the following problems and indicate, if you can, whether this bound is tight.
   a. Finding the largest element in an array
   b. Checking completeness of a graph represented by its adjacency matrix.
   c. Generating all the subsets of a n-element set
   d. Determining whether n given real numbers are all distinct
7. How is problem reduction method used in finding the limits of algorithm? 10
8. How are decision trees used in judging the performance of an algorithm? 10
9. Obtain a decision tree for selection sort. 10
10. Obtain a decision tree for searching a sorted array. 10
11. What are P and NP and NP complete problems? Give examples. What is meant by reducibility? 10
12. Show that the partition problem is polynomially reducible to the decision version of the Knapsack problem. 10

UNIT VII : COPING WITH LIMITATIONS OF ALGORITHMIC POWER

OBJECTIVE: Many problems are different to solve algorithmically. This chapter outlines several ways of dealings with such difficult problems. It introduces two algorithm design techniques back tracking and branch & bound. Algorithms that solve problems approximately but fast are also discussed here. Solutions to knapsack and TSP are also dealt here.

1. a. What is Backtracking? 10
   b. Draw the state – space tree for 4-queens problem.
   c. Write algorithms to check whether kth queen can be placed successfully and to place all N queens on the chessboard.
2. Write an algorithm for sum of subset problem using backtracking. Also solve the following instance of sum of subset problem : S =\{1,5,2,7\} with d = 8. 10
3. Solve the following Assignment problem using Branch and Bound algorithm. 10

<table>
<thead>
<tr>
<th>Persons</th>
<th>Job1</th>
<th>Job2</th>
<th>Job3</th>
<th>Job4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10</td>
<td>3</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>5</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>C</td>
<td>6</td>
<td>9</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>D</td>
<td>8</td>
<td>7</td>
<td>10</td>
<td>5</td>
</tr>
</tbody>
</table>

Department of ISE
4. Apply Branch and Bound algorithm to solve the travelling salesman problem for the graph with a cost adjacency matrix is as follows.

\[
\begin{array}{c|ccccc}
  & A & B & C & D & E \\
\hline
A & 0 & 3 & 1 & 5 & 8 \\
B & 3 & 0 & 6 & 7 & 9 \\
C & 1 & 6 & 0 & 4 & 2 \\
D & 5 & 7 & 4 & 0 & 3 \\
E & 8 & 9 & 2 & 3 & 0 \\
\end{array}
\]

5. What are NP hard problems? Write short notes on the procedures of the following approximation algorithms to solve TSP using suitable examples.
   a) Nearest Neighbor algorithm
   b) Twice-around-the-tree algorithm

6. Explain the knapsack problem using branch & bound technique with an example.
7. Explain the traveling salesman problem using branch & bound and backtracking.

UNIT VIII : PRAM ALGORITHMS

OBJECTIVE: To give the basic understanding on the efficiency and complexity analysis of parallel algorithms.

1. Explain briefly the computational model of PRAM algorithms.
2. Explain Deterministic list ranking and Randomized list ranking.
3. Explain the Prefix computational problem.
4. Explain the Graph problems in detail