SYED AMMAL ENGINEERING COLLEGE

(An ISO 9001: 2008 Certified Institution)
Dr. E.M. Abdullah Campus, Ramanathapuram – 623 502

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(Accredited by NBA)

DESIGN OF ALGORITHMS & ANALYSIS QUESTIONS

1. Explain what is Quick Sort algorithm?

   Quick Sort algorithm has the ability to sort list or queries quickly. It is based on the principle of partition exchange sort or Divide and conquer. This type of algorithm occupies less space, and it segregates the list into three main parts Elements less than the Pivot element Pivot element Elements greater than the Pivot element

2. Explain what is an algorithm in computing?

   An algorithm is a well-defined computational procedure that take some value as input and generate some value as output. In simple words, it's a sequence of computational steps that converts input into the output.

3. Explain what is time complexity of Algorithm?

   Time complexity of an algorithm indicates the total time needed by the program to run to completion. It is usually expressed by using the big O notation.

4. Mention what are the types of Notation used for Time Complexity?

   The types of Notations used for Time Complexity includes Big Oh: It indicates "fewer than or the same as" <expression>iterations Big Omega: It indicates "more than or same as" <expression>iterations Big Theta: It indicates "the same as"<expression>iterations Little Oh: It indicates "fewer than" <expression>iterations Little Omega: It indicates "more than" <expression>iterations

5. Explain how binary search works?

   In binary search, we compare the key with the item in the middle position of the array. If the key is less than the item searched then it must lie in the lower half of the array, if the key is greater than the item searched than it should be in upper half of the array.

Department of CSE

6. Explain whether it is possible to use binary search for linked lists?

   Since random access is not acceptable in linked list, it is impossible to reach the middle element of O(1) time. Thus, binary search is not possible for linked list.

7. Explain what is heap sort?

   Heap-sort can be defined as a comparison based sorting algorithm. It divides its input into the unsorted and sorted region, until it shrinks the unsorted region by eliminating the smallest element and moving that to the sorted region.

8. Explain what is Skip list?

   Skip list the method for data structuring, where it allows the algorithm to search, delete and insert elements in a symbol table or dictionary. In a skip list, each element is represented by a node. The search function returns the content of the value related to key. The insert operation associates a specified key with a new value, while the delete function deletes the specified key

9. Explain what is Space complexity of insertion sort algorithm?

   Insertion sort is an in-place sorting algorithm which means that it requires no extra or little. storage. For insertion sort, it requires only single list elements to be stored out-side the initial data, making the space-complexity 0(1)..

10. Explain what a "Hash Algorithm" is and what are they used for?

    "Hash Algorithm" is a hash function that takes a string of any length and decreases it to a unique fixed length string. It is used for password validity, message & data integrity and for many other cryptographic systems.

Department of CSE

11. Explain how to find whether the linked list has a loop?

To know whether the linked list has a loop, we will take two pointer approach. If we maintain two pointers, and we increase one pointer after processing two nodes and other after processing every node, we are likely to encounter a situation where both the pointer will be pointing to the same node. This will only occur if linked list has a loop.

12. Explain what is the difference between best case scenario and worst case scenario of an algorithm?

Best case scenario: Best case scenario for an algorithm is explained as the arrangement of data for which the algorithm performs best. For example, we take a binary search, for which the best case scenario would be if the target value is at the very center of the data you are searching. The best case time complexity would be 0 (1)

Worst case scenario: It is referred for the worst set of input for a given algorithm. For example quicksort, which can perform worst if you select the largest or smallest element of a sublist for the pivot value. It will cause quicksort to degenerate to O (n2)

13. Explain what is Radix Sort algorithm?

Radix sort puts the element in order by comparing the digits of the numbers. It is one of the linear sorting algorithms for integers.

14. Mention what are the three laws of recursion algorithm?

All recursive algorithm must follow three laws It should have a base case A recursive algorithm must call itself A recursive algorithm must change its state and move towards the base case

Department of CSE

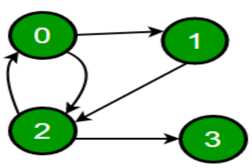
15. Explain what is bubble sort algorithm?

Bubble sort algorithm is also referred as sinking sort. In this type of sorting, the list to be sorted out compares the pair of adjacent items. If they are organized in the wrong order, it will swap the values and arrange them in the correct order

16. Define Transitive Closure of a Graph using DFS

Given a directed graph, find out if a vertex v is reachable from another vertex u for all vertex pairs (u, v) in the given graph. Here reachable mean that there is a path from vertex u to v. The reach-ability matrix is called transitive closure of a graph.

For example, consider below graph



Transitive closure of above graphs is

1 1 1 1

1 1 1 1

1 1 1 1

0 0 0 1

Department of CSE

17. List out Applications of Depth First Search

- For an unweighted graph, DFS traversal of the graph produces the minimum spanning tree and all pair shortest path tree.
- **Detecting cycle in a graph**
- **.Path Finding**.
- **To test if a graph is** bipartite.
- Topological Sorting
- **Finding Strongly Connected Components of a graph**
- **Solving puzzles with only one solution**,

18. What is recurrence for worst case of QuickSort and what is the time complexity in Worst case?

Recurrence is T(n) = T(n-1) + O(n) and time complexity is O(n^2)

19. Suppose we have a O(n) time algorithm that finds median of an unsorted array. Now consider a QuickSort implementation where we first find median using the above algorithm, then use median as pivot. What will be the worst case time complexity of this modified QuickSort.

O(nLogn)

20. Given an unsorted array. The array has this property that every element in array is at most k distance from its position in sorted array where k is a positive integer smaller than size of array. Which sorting algorithm can be easily modified for sorting this array and what is the obtainable time complexity?

Heap Sort with time complexity O(nLogk)

21. Which of the following is not true about comparison based sorting algorithms?

Heap Sort is not a comparison based sorting algorithm.

22. What is time complexity of fun()?

```
int fun(int n)
{
  int count = 0;
  for (int i = n; i > 0; i /= 2)
    for (int j = 0; j < i; j++)
      count += 1;
  return count;
}
```

O(n)

Department of CSE

23. What is the time complexity of fun()?

```
int fun(int n)
{
  int count = 0;
  for (int i = 0; i < n; i++)
    for (int j = i; j > 0; j--)
      count = count + 1;
  return count;
}
```

Theta (n^2)

24. The recurrence relation capturing the optimal time of the Tower of Hanoi problem with n discs is.

T(n) = 2T(n – 1) + 1

25 . Which of the following algorithms is NOT a divide & conquer algorithm by nature?

Heap Sort

Department of CSE

26 . Consider the polynomial p(x) = a0 + a1x + a2x^2 +a3x^3, where ai != 0, for all i. The minimum number of multiplications needed to evaluate p on an input x is:

3

27. Maximum Subarray Sum problem is to find the subarray with maximum sum. For Example ,given an array {12, -13, -5, 25, -20, 30, 10}, the maximum subarray sum is 45. The naive solution for this problem is to calculate sum of all subarrays starting with every element and return the maximum of all. We can solve this using Divide and Conquer, what will be the worst case time complexity using Divide and Conquer.

O(nLogn)

28 . Consider a situation where you don't have function to calculate power (pow() function in C) and you need to calculate x^n where x can be any number and n is a positive integer. What can be the best possible time complexity of your power function?

O(Logn)

29. The secant method is used to find the root of an equation $f(x) = 0$. It is started from two distinct estimates x_a and x_b for the root. It is an iterative procedure involving linear interpolation to a root. The iteration stops if $f(x_b)$ is very small and then x_b is the solution. The procedure is given below. Observe that there is an expression which is missing and is marked by? Which is the suitable expression that is to be put in place of? So that it follows all steps of the secant method?

Initialize: x_a, x_b, ε, N // ε = convergence indicator

$f_b = f(x_b)$ i = 0

while (i < N and $|f_b| > ε$) do

i = i + 1 // update counter

x_t = ? // missing expression for

// intermediate value

$x_a = x_b$ // reset x_a

$x_b = x_t$ // reset x_b

$f_b = f(x_b)$ // function value at new x_b

end while

if $|f_b| > ε$

then // loop is terminated with i = N

write "Non-convergence"

else

write "return x_b"

end if

$$\mathbf{x_a - (x_b - x_a)\, f_a / (f_b - f(x_a))}$$

Department of CSE

30. Suppose you are provided with the following function declaration in the C programming language.

int partition (int a[], int n);

The function treats the first element of a[] as a pivot, and rearranges the array so that all elements less than or equal to the pivot is in the left part of the array, and all elements greater than the pivot is in the right part. In addition, it moves the pivot so that the pivot is the last element of the left part. The return value is the number of elements in the left part. The following partially given function in the C programming language is used to find the kth smallest element in an array a[ ] of size n using the partition function. We assume $k \leq n$

```
int kth_smallest (int a[], int n, int k)
{
   int left_end = partition (a, n);
   if (left_end+1==k)
   {
       return a [left_end];
   }
   if (left_end+1 > k)
   {
      return kth_smallest (____________________);
   }
   else
   {
      return kth_smallest (____________________);
    }
}
```

(a, left_end, k) and (a+left_end+1, n–left_end–1, k–left_end–1)

Department of CSE

31. What is the value of following recurrence.

T(n) = T(n/4) + T(n/2) + cn^2

T(1) = c

T(0) = 0

Where c is a positive constant

O(n^2)

32. Consider the following program fragment for reversing the digits in a given integer to obtain a new integer. Let n = D1D2…Dm

```
int n, rev;
rev = 0;
while (n > 0)
{
  rev = rev*10 + n%10;
  n = n/10;
}
```

n = D1D2…Dm-i and rev = DmDm-1…Dm-i+1

Department of CSE

33. What is the best time complexity of bubble sort?

N

34. What is the worst case time complexity of insertion sort where position of the data to be inserted is calculated using binary search?

N^2

35**.** The tightest lower bound on the number of comparisons, in the worst case, for comparison-based sorting is of the order of

NlogN

36. In a modified merge sort, the input array is splitted at a position one-third of the length(N) of the array. What is the worst case time complexity of this merge sort?

N(logN base 3/2)

37. What is the time complexity of the below function?

```
void fun(int n, int arr[])
{
   int i = 0, j = 0;
   for(; i < n; ++i)
      while(j < n && arr[i] < arr[j])
         j++;
}
```

O(n)

38. In a competition, four different functions are observed. All the functions use a single for loop and within the for loop, same set of statements are executed. Consider the following for loops:

A) for(i = 0; i < n; i++)

B) for(i = 0; i < n; i += 2)

C) for(i = 1; i < n; i *= 2)

D) for(i = n; i > -1; i /= 2)

If **n** is the size of input(positive), which function is most efficient (if the task to be performed is not an issue)?

for(i = 1; i < n; i *= 2)

39. The following statement is valid. log(n!) = θ(n log n).

TRUE

Department of CSE

40. What does it mean when we say that an algorithm X is asymptotically more efficient than Y?

X will be a better choice for all inputs except small inputs

41. What is the time complexity of Floyd–Warshall algorithm to calculate all pair shortest path in a graph with **n** vertices?

Theta(n^3)

42. A list of n string, each of length n, is sorted into lexicographic order using the merge-sort algorithm. The worst case running time of this computation is

O(n^2log n)

43. In quick sort, for sorting n elements, the (n/4)th smallest element is selected as pivot using an O(n) time algorithm. What is the worst case time complexity of the quick sort?

theta(nlogn)

44. In the following C function, let n >= m.

```
int gcd(n,m)
{
  if (n%m ==0) return m;
  n = n%m;
  return gcd(m, n);
}
```

How many recursive calls are made by this function?

theta(logn)

45. Consider the following function

```
int unknown(int n) {
   int i, j, k = 0;
   for (i  = n/2; i <= n; i++)
       for (j = 2; j <= n; j = j * 2)
           k = k + n/2;
   return k;
}
```

Theta(n^2log n)

Department of CSE

46. What is running time of an algorithm is given by the reccurence t(n)=t(n-1)+n,t(1)=1.

n

47. Which data structure is used in the implementation of topological sorting?

Directed acyclic graph

48. Which one of the following is the tightest upper bound that represents the number of swaps required to sort n numbers using selection sort?

(A) O(logn**) (B) O(n)** (C)O(nlogn)(D) O(n2)

49. Which one of the following is the tightest upper bound that represents the time complexity of inserting an object into a binary search tree of n nodes?

(A) O(1) (B) O(logn) (C) **O(n) (**D)O(nlog n)

50. What is the time complexity of Bellman-Ford single-source shortest path algorithm on a complete graph of n vertices?

Θ(n3)

51. Consider the following two functions. What are time complexities of the functions?

```
int fun1(int n)
{
   if (n <= 1) return n;
   return 2*fun1(n-1);
}
int fun2(int n)
{
   if (n <= 1) return n;
   return fun2(n-1) + fun2(n-1);
}
```

O(n) for fun1() and O(2^n) for fun2()

Department of CSE

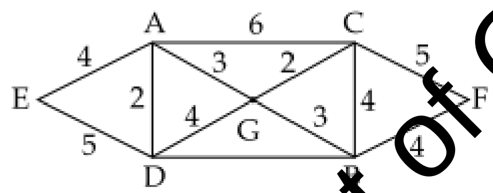
52. n quick sort, for sorting n elements, the (n/4)th smallest element is selected as pivot using an O(n) time algorithm. What is the worst case time complexity of the quick sort?

Theta(nLogn)

53. You have an array of n elements. Suppose you implement quicksort by always choosing the central element of the array as the pivot. Then the tightest upper bound for the worst case performance is

$O(n^2)$

54.



Consider the graph given below : Use Kruskal's algorithm to find a minimal spanning tree for the graph. The List of the edges of the tree in the order in which they are choosen is?

(1) AD, AE, AG, GC, GB, BF

(2) GC, GB, BF, GA, AD, AE

(3) GC, AD, GB, GA, BF, AE

(4) AD, AG, GC, AE, GB, BF

ANS: BD, GC, AD, BG, AE, BF

Department of CSE

55. The following numbers are inserted into an empty binary search tree in the given order: 10, 1, 3, 5, 15, 12, 16 What is the height of the binary search tree ?

3

56. Consider the following C-function:

```
double foo (int n)
{
   int i;
```

```
    double sum;
    if (n = = 0) return 1.0;
    else
    {
        sum = 0.0;
        for (i = 0; i < n; i++)
            sum += foo (i);
        return sum;
    }
}
```

The space complexity of the above function is:

O(n)

57. In the following table, the left column contains the names of standard graph algorithms and the right column contains the time complexities of the algorithms. Match each algorithm with its time complexity.

1. Bellman-Ford algorithm 2. Kruskal's algorithm 3. Floyd-Warshall algorithm 4. Topological sorting	A : O (m log n) B : O (n^3) C : O (nm) D : O (n + m)

1 → C, 2 → A, 3 → B, 4 → D

58. Let T(n) be a function defined by the recurrence $T(n) = 2T(n/2) + \sqrt{n}$ for $n \geq 2$ and T(1) = 1 Which of the following statements is TRUE?

T(n) = θ(n)

59. The worst case running times of Insertion sort, Merge sort and Quick sort, respectively, are:

$\Theta(n^2)$, $\Theta(n \log n)$ and $\Theta(n^2)$

Department of CSE

60. Assume that the algorithms considered here sort the input sequences in ascending order. If the input is already in ascending order, which of the following are TRUE ?

I. Quicksort runs in $\Theta(n^2)$ time

II. Bubblesort runs in $\Theta(n^2)$ time

III. Mergesort runs in $\Theta(n)$ time

IV. Insertion sort runs in $\Theta(n)$ time

I and IV only

61. A problem in NP is NP-complete if

The 3-SAT problem can be reduced to it in polynomial time

62. The characters a to h have the set of frequencies based on the first 8 Fibonacci numbers as follows

a : 1, b : 1, c : 2, d : 3, e : 5, f : 8, g : 13, h : 21/

A Huffman code is used to represent the characters. What is the sequence of characters corresponding to the following code? 110111100111010

fdheg

63. What is the size of the smallest MIS(Maximal Independent Set) of a chain of nine nodes?

3

64. Arrange the following functions in increasing asymptotic order:

A. $n^{1/3}$

B. e^n

C. $n^{7/4}$

D. $n \log^9 n$

E. 1.0000001^n

A, D, C, E, B

Department of CSE

65. If we use Radix Sort to sort n integers in the range ($n^{k/2}$,n^k], for some k>0 which is independent of n, the time taken would be?

Θ(nlogn)

66. The auxiliary space of insertion sort is O(1), what does O(1) mean ?

It means the amount of extra memory Insertion Sort consumes doesn't depend on the input. The algorithm should use the same amount of memory for all inputs.

67. The time complexity of computing the transitive closure of binary relation on a set of n elements is known to be

$O(n^3)$

68**.** Match the following:

codes:

List - I	List - II
(a) The 8-Queen's problem	(i)Dynamic programming
(b) Single-Source shortest paths	(ii) Divide and conquer
(c)STRASSEN's Matrix multiplication	(iii)Greedy approach
(d)Optimal binary search trees	(iv)Backtracking

Ans: (iv) (iii) (ii) (i)

68. Floyd-Warshall algorithm utilizes __________ to solve the all-pairs shortest paths problem on a directed graph in __________ time.

Dynamic programming, θ (V^3)

Department of CSE

69. To determine the efficiency of an algorithm the time factor is measured by:

Counting number of key operations

70. Let T(n) be defined by T(1) = 10 and T(n + 1) = 2n + T(n) and for all integers n ≥ 1 . Which of the following represents the order of growth of T(n) as a function of

O(n^2)

71. An all-pairs shortest-paths problem is efficiently solved using:

Floyd-Warshall algorithm

72. The travelling salesman problem can be solved in:

Exponential time using dynamic programming algorithm or branch-and-bound algorithm

73. Which of the following is asymptotically smaller?

lg(lg*n)

74. Let f(n) and g(n) be asymptotically non-negative functions. Which of the following is correct?

θ (f (n) + g(n)) = max (f (n), g(n))

75. Which design matric is used to measure the compactness of the program in terms of lines of code?

Conciseness

76. Which of the following standard algorithms is not Dynamic Programming based.

Prim's Minimum Spanning Tree

77. We use dynamic programming approach when

The solution has optimal substructure

78. A sub-sequence of a given sequence is just the given sequence with some elements (possibly none or all) left out. We are given two sequences X[m] and Y[n] of lengths m and n respectively, with indexes of X and Y starting from 0. We wish to find the length of the longest common sub-sequence(LCS) of X[m] and Y[n] as l(m,n), where an incomplete recursive definition for the function l(i,j) to compute the length of The LCS of X[m] and Y[n] is given below:

Department of CSE

l(i,j) = 0, if either i=0 or j=0

= expr1, if i,j > 0 and X[i-1] **=** Y[j-1]

= expr2, if i,j > 0 and X[i-1] **!=** Y[j-1]

expr2 ≡ max(l(i-1, j), l(i, j-1))

79. Consider the following two sequences :

X = < B, C, D, C, A, B, C >, and

Y = < C, A, D, B, C, B >

The length of longest common subsequence of X and Y is :

4

80. The following paradigm can be used to find the solution of the problem in minimum time: Given a set of non-negative integer, and a value K, determine if there is a subset of the given set with sum equal to K:

Dynamic Programming

81. Consider the problem of searching an element x in an array 'arr[]' of size n. The problem can be solved in O(Logn) time if. 1) Array is sorted 2) Array is sorted and rotated by k. k is given to you and k <= n 3) Array is sorted and rotated by k. k is NOT given to you and k <= n 4) Array is not sorted

1, 2 and 3 only

82. **What are the arguments present in pattern matching algorithms?**

These are the following arguments which are present in pattern matching Algorithms.

1) Subject,

2) Pattern

3) Cursor

4) MATCH_STR

5) REPLACE_STR

6) REPLACE_FLAG

Department of CSE

83. **How to find median of a BST?**

Find the no. of elements on the left side. If it is n-1 the root is the median. If it is more than n-1, then it has already been found in the left subtree. Else it should be in the right subtree

84. What is Diffie-Hellman?

It is a method by which a key can be securely shared by two users without any actual exchange.

85. What is the goal of the shortest distance algorithm?

The goal is completely fill the distance array so that for each vertex v, the value of distance[v] is the weight of the shortest path from start to v.

86. Explain the depth of recursion?

This is another recursion procedure which is the number of times the procedure is called recursively in the process of enlarging a given argument or arguments. Usually this quantity is not obvious except in the case of extremely simple recursive functions, such as FACTORIAL (N), for which the depth is N.

87. Which are the sorting algorithms categories?

Sorting algorithms can be divided into five categories:

a) insertion sorts

b) exchange sorts

c) selection sorts

d) merge sorts

e) distribution sorts

88.Define a brute-force algorithm. Give a short example.

A brute force algorithm is a type of algorithm that proceeds in a simple and obvious way, but requires a huge number of steps to complete. As an example, if you want to find out the factors of a given number N, using this sort of algorithm will require to get one by one all the possible number combinations.

Department of CSE

89. What is a greedy algorithm? Give examples of problems solved using greedy algorithms.

A greedy algorithm is any algorithm that makes the local optimal choice at each stage with the hope of finding the global optimum. A classical problem which can be solved using a greedy strategy is the traveling salesman problem. Another problems that can be solved using greedy algorithms are the graph coloring problem and all the NP-complete problems.

90. What is a backtracking algorithm? Provide several examples.

It is an algorithm that considers systematically all possible outcomes for each decision. Examples of backtracking algorithms are the eight queens problem or generating permutations of a given sequence.

91. What is the difference between a backtracking algorithm and a brute-force one?

Due to the fact that a backtracking algorithm takes all the possible outcomes for a decision, it is similar from this point of view with the brute force algorithm. The difference consists in the fact that sometimes a backtracking algorithm can detect that an exhaustive search is unnecessary and, therefore, it can perform much better.

92. Describe divide and conquer paradigm.

When a problem is solved using a divide and conquer algorithm, it is subdivided into one or more subproblems which are all similar to the original problem in such a way that each of the subproblems can be solved independently. In the end, the solutions to the subproblems are combined in order to obtain the solution to the original problem.

93. Describe on short an insertion sorting algorithm.

An algorithm that sorts by insertion takes the initial, unsorted sequence and computes a series of sorted sequences using the following rules:

a) the first sequence in the series is the empty sequence

b) given a sequence S(i) in the series, for 0<=i<="" p="">

Department of CSE

94. Which are the advantages provided by insertion sort?

Insertion sort provides several advantages:

a) simple implementation

b) efficient for small data sets

c) adaptive - efficient for data sets that are already substantially sorted: the time complexity is O(n + d), where d is the number of inversions

d) more efficient in practice than most other simple quadratic, i.e. O(n2) algorithms such as selection sort or bubble sort; the best case (nearly sorted input) is O(n)

e) stable - does not change the relative order of elements with equal keys

f) in-place - only requires a constant amount O(1) of additional memory space

g) online - can sort a list as it receives it.

95. **What is Huffman coding?**

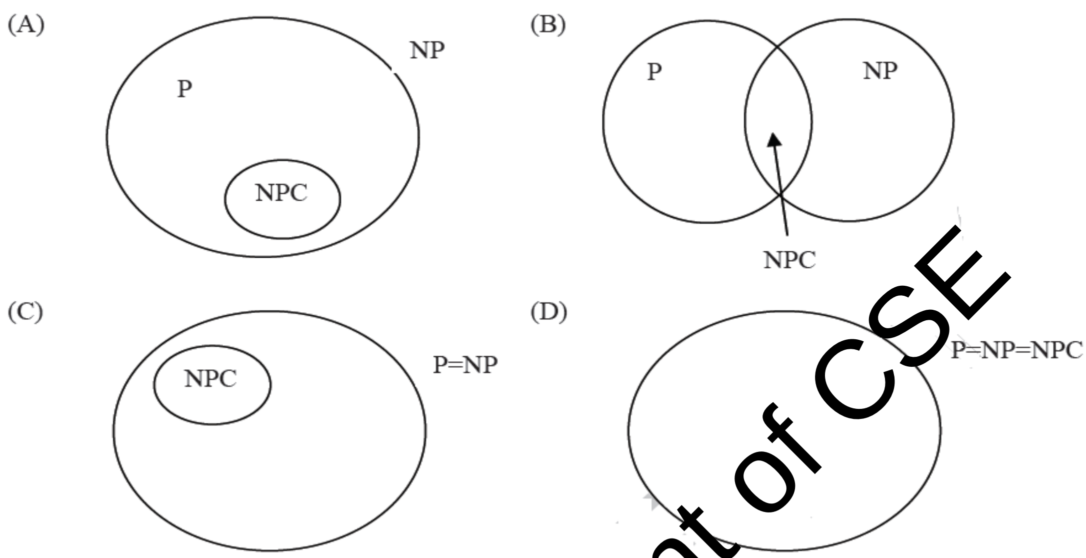
In computer science and information theory, Huffman coding is an entropy encoding algorithm used for lossless data compression. The term refers to the use of a variable-length code table for encoding a source symbol (such as a character in a file) where the variable-length code table has been derived in a particular way based on the estimated probability of occurrence for each possible value of the source symbol.

96. Let S be an NP-complete problem and Q and R be two other problems not known to be in NP. Q is polynomial time reducible to S and S is polynomial-time reducible to R. Which one of the following statements is true?

R is NP-hard

Department of CSE

97. Suppose a polynomial time algorithm is discovered that correctly computes the largest clique in a given graph. In this scenario, which one of the following represents the correct Venn diagram of the complexity classes P, NP and NP Complete (NPC)?



Answer: D

98. Consider the decision problem 2CNFSAT defined as follows:

{ Φ | Φ is a satisfiable propositional formula in CNF with at most two literals per clause }

For example, $\Phi = (x_1 \vee x_2) \wedge (x_1 \vee \bar{x}_3) \wedge (x_2 \vee x_4)$ is a Boolean formula and it is in *2CNFSAT*.

The decision problem *2CNFSAT* is

A NP-Complete.

B solvable in polynomial time by reduction to directed graph reachability.

C solvable in constant time since any input instance is satisfiable.

D NP-hard, but not NP-complete.

Answer: NP-Complete

Department of CSE

99. Let $SHAM_3$ be the problem of finding a Hamiltonian cycle in a graph G = (V,E) with V divisible by 3 and $DHAM_3$ be the problem of determining if a Hamiltonian cycle exists in such graphs. Which one of the following is true?

A. Both $DHAM_3$ and $SHAM_3$ are NP-hard

B $SHAM_3$ is NP-hard, but $DHAM_3$ is not

C $DHAM_3$ is NP-hard, but $SHAM_3$ is not

D. Neither $DHAM_3$ nor $SHAM_3$ is NP-hard

Answer: Both $DHAM_3$ and $SHAM_3$ are NP-hard

Department of CSE

100. Which of the following statements are TRUE?

1. The problem of determining whether there exists a cycle in an undirected graph is in P.
2. The problem of determining whether there exists a cycle in an undirected graph is in NP.
3. If a problem A is NP-Complete, there exists a non-deterministic polynomial time algorithm to solve A.

Answer: 1, 2 and 3