Data Structure – Solved GATE Questions

UNIT – 1

1) **Consider the following pseudo code. What is the total number of multiplications to be performed? (2015)**

```
D = 2
fori = 1 to n do
for j = i to n do
for k = j + 1 to n do
    D = D * 3
```

(A) Half of the product of the 3 consecutive integers.
(B) One-third of the product of the 3 consecutive integers.
(C) One-sixth of the product of the 3 consecutive integers.
(D) None of the above.

Answer (C)

The statement "D = D * 3" is executed **n*(n+1)*(n-1)/6** times.
Let us see how.

For i = 1, the multiplication statement is executed (n-1) + (n-2) + .. 2 + 1 times.
For i = 2, the statement is executed (n-2) + (n-3) + .. 2 + 1 times
………………………..
……………………….
Fori = n-1, the statement is executed once.
For i = n, the statement is not executed at all

So overall the statement is executed following times
[(n-1) + (n-2) + .. 2 + 1] + [(n-2) + (n-3) + .. 2 + 1] + … + 1 + 0

The above series can be written as
S = [n*(n-1)/2 + (n-1)*(n-2)/2 + ….. + 1]
The sum of above series can be obtained by trick of subtraction the series from standard Series S1 = $n^2 + (n-1)^2 + .. 1^2$. The sum of this standard series is n*(n+1)*(2n+1)/6

S1 – 2S = n + (n-1) + … 1 = n*(n+1)/2
2S = n*(n+1)*(2n+1)/6 – n*(n+1)/2
S = n*(n+1)*(n-1)/6

2) **Let A be a square matrix of size n x n. Consider the following program. What is the expected output?(2014)**

```
C = 100
fori = 1 to n do
    for j = 1 to n do
    {
    Temp = A[i][j] + C
```

```
             A[i][j] = A[j][i]
            A[j][i] = Temp - C
             }
     fori = 1 to n do
            for j = 1 to n do
            Output(A[i][j]);
```

(A) ThematrixAitself
(B) TransposeofmatrixA
(C) Adding 100 to the upper diagonal elements and subtracting 100 from diagonal elements of A
(D) None of the above

Answer:A

If we take look at the inner statements of first loops, we can notice that the statements swap A[i][j] and A[j][i] for all i and j.Since the loop runs for all elements, every element A[l][m] would be swapped twice, once for i = l and j = m and then for i = m and j = l. Swapping twice means the matrix doesn't change.

3. What is the return value of f(p, p) if the value of p is initialized to 5 before the call? Note that the first parameter is passed by reference, whereas the second parameter is passed by value.(2014)

```
int f(int&x, int c)
 {
c  = c - 1;
if (c == 0) return 1;
     x = x + 1;
return f(x, c) * x;
}
```

(A)3024(B)6561 (C)55440(D) 161051

Answer(B)
Since c is passed by value and x is passed by reference, all functions will have same copy of x, but different copies of c.

f(5, 5) = f(x, 4)*x = f(x, 3)*x*x = f(x, 2)*x*x*x = f(x, 1)*x*x*x*x = 1*x*x*x*x = x^4

Since x is incremented in every function call, it becomes 9 after f(x, 2) call. So the value of expression x^4 becomes 9^4 which is 6561.

UNIT – II

4. **Consider the following C function in which size is the number of elements in the array E:The value returned by the function MyX is the(2014)**

```
intMyX(int*E, unsigned intsize)
{
   intY = 0;
   intZ;
   inti, j, k;

   for(i = 0; i< size; i++)
      Y = Y + E[i];

   for(i = 0; i< size; i++)
      for(j = i; j < size; j++)
      {
         Z = 0;
         for(k = i; k <= j; k++)
            Z = Z + E[k];
         if(Z > Y)
            Y = Z;
      }
   returnY;
}
```

(A) maximum possible sum of elements in any sub-array of array E.

(B) maximum element in any sub-array of array E.

(C) sum of the maximum elements in all possible sub-arrays of array E

(D) the sum of all the elements in the array E.

Answer:(A)

Explanation: The function does followingY is used to store maximum sum seen so far and Z is used to store current sum1) Initialize Y as sum of all elements 2) For every element, calculate sum of all subarrays starting with arr[i]. Store the current sum in Z. If Z is greater than Y, then update Y.

5. **The procedure given below is required to find and replace certain characters inside an input character string supplied in array A. The characters to be replaced are supplied in array oldc, while their respective replacement characters are supplied in**

array newc. Array A has a fixed length of five characters, while arrays oldc and newc contain three characters each. However, the procedure is flawed(2013)

```
voidfind_and_replace(char *A, char *oldc, char *newc)
{
for (inti = 0; i< 5; i++)
for (int j = 0; j < 3; j++)
if (A[i] == oldc[j]) A[i] = newc[j];
}
```

The procedure is tested with the following four test cases

(1) oldc = "abc", newc = "dab"

(2) oldc = "cde", newc = "bcd"

(3) oldc = "bca", newc = "cda"

(4) oldc = "abc", newc = "bac"

The tester now tests the program on all input strings of length five consisting of characters ‘a’, ‘b’, ‘c’, ‘d’ and ‘e’ with duplicates allowed. If the tester carries out this testing with the four test cases given above, how many test cases will be able to capture the flaw? (2012)

(A) Only one

(B) Only two

(C) Only three

(D) All four

Answer (B)

The test cases 3 and 4 are the only cases that capture the flaw. The code doesn't work properly when an old character is replaced by a new character and the new character is again replaced by another new character. This doesn't happen in test cases (1) and (2), it happens only in cases (3) and (4).

6. **If array A is made to hold the string “abcde”, which of the above four test cases will be successful in exposing the flaw in this procedure? (2012)**

(A) None

(B) 2 only

(C) 3 and 4 only

(D) 4 only

Answer (C)

```
#include <stdio.h>
#include <string.h>

void find_and_replace(char *A, char *oldc, char *newc) {
   for (inti = 0; i< 5; i++)
      for (int j = 0; j < 3; j++)
         if (A[i] == oldc[j]) A[i] = newc[j];
}

int main()
{
   char *oldc1 = "abc", *newc1 = "dab";
   char *oldc2 = "cde", *newc2 = "bcd";
   char *oldc3 = "bca", *newc3 = "cda";
   char *oldc4 = "abc", *newc4 = "bac";

   char test[] =  "abcde";

   printf("Test 2\n");
   printf("%s\n", test);
   find_and_replace(test, oldc2, newc2);
   printf ("%s\n", test);

   printf("\nTest 3\n");
   strcpy(test, "abcde");
   printf("%s\n", test);
   find_and_replace(test, oldc3, newc3);
   printf ("%s\n", test);

   printf("\nTest 4\n");
   strcpy(test, "abcde");
   printf("%s\n", test);
   find_and_replace(test, oldc4, newc4);
   printf ("%s\n", test);
}
```

Output:

```
Test 2
abcde
abbcd
```

Test 3
abcde
addde

Test 4
abcde
aacde

7. **Suppose implementation supports an instruction REVERSE, which reverses the order of elements on the stack, in addition to the PUSH and POP instructions. Which one of the following statements is TRUE with respect to this modified stack?(2013)**

(A) A queue cannot be implemented using this stack.

(B) A queue can be implemented where ENQUEUE takes a single instruction and DEQUEUE takes a sequence of two instructions.

(C) A queue can be implemented where ENQUEUE takes a sequence of three instructions and DEQUEUE takes a single instruction.

(D) A queue can be implemented where both ENQUEUE and DEQUEUE take a single instruction each.

Answer:(C)

ToDEQUEUEanitem,simplyPOP.

To ENQUEUE an item, we can do following 3 operations

1)REVERSE

2)PUSH

3) REVERSE

8. **A priority queue is implemented as a Max-Heap. Initially, it has 5 elements. The level-order traversal of the heap is: 10, 8, 5, 3, 2. Two new elements 1 and 7 are inserted into the heap in that order. The level-order traversal of the heap after the insertion of the elements is(2014)**

(A) 10, 8, 7, 3, 2, 1, 5

(B) 10, 8, 7, 2, 3, 1, 5

(C) 10, 8, 7, 1, 2, 3, 5

(D) 10, 8, 7, 5, 3, 2, 1

Answer: (A)

Initially heap has 10, 8, 5, 3, 2

```
10
/  \
8    5
/ \
3   2
```

After insertion of 1

```
10
/   \
8     5
/ \   /
3   2 1
```

No need to heapify as 5 is greater than 1.

After insertion of 7

```
10
/   \
8     5
/ \   / \
3   2 1   7
```

Heapify 5 as 7 is greater than 5

```
10
/   \
8     7
/ \   / \
3   2 1   5
```

No need to heapify any further as 10 is greater than 7

9 What is the worst case time complexity of a sequence of n MultiDequeue() operations on an initially empty queue? (2013)

(A)$\Theta(n)$

(B)$\Theta(n+k)$

(C)$\Theta(nk)$

(D) $\Theta(n^2)$

Answer(A)

Since the queue is empty initially, the condition of while loop never becomes true. So the time complexity is $\Theta(n)$

10)Suppose a circular queue of capacity (n – 1) elements is implemented with an array of n elements. Assume that the insertion and deletion operation are carried out using REAR and FRONT as array index variables, respectively. Initially, REAR = FRONT = 0. The conditions to detect queue full and queue empty are(2012)

(A) Full: (REAR+1) mod n == FRONT, empty: REAR == FRONT

(B) Full: (REAR+1) mod n == FRONT, empty: (FRONT+1) mod n == REAR

(C) Full: REAR == FRONT, empty: (REAR+1) mod n == FRONT

(D) Full: (FRONT+1) mod n == REAR, empty: REAR == FRONT

Answer (A)

11. Consider a hash table with 9 slots. The hash function is h(k) = k mod 9. The collisions are resolved by chaining. The following 9 keys are inserted in the order: 5, 28, 19, 15, 20, 33, 12, 17, 10. The maximum, minimum, and average chain lengths in the hash table, respectively, are(2015)

(A)3,0,and1

(B)3,3,and3

(C)4,0,and1

(D) 3, 0, and 2

Answer:(A)

Following are values of hash function for all keys

5 --> 5

28 --> 1

19 -->1 [Chained with 28]

15 --> 6

20 --> 2

33 -->6 [Chained with 15]

12 --> 3

17 --> 8

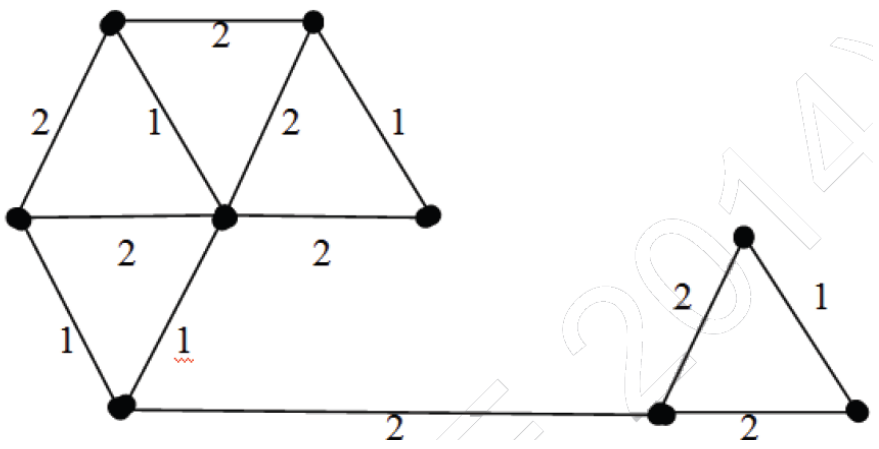
10 --> 1 [Chained with 28 and 19]

The maximum chain length is 3. The keys 28, 19 and 10 go to same slot 1, and form a chain of length 3.The minimum chain length 0, there are empty slots (0, 4 and 7). Average chain length is (0 + 3 + 1 + 1 + 0 + 1 + 2 + 0 + 1)/9 = 1

UNIT – III

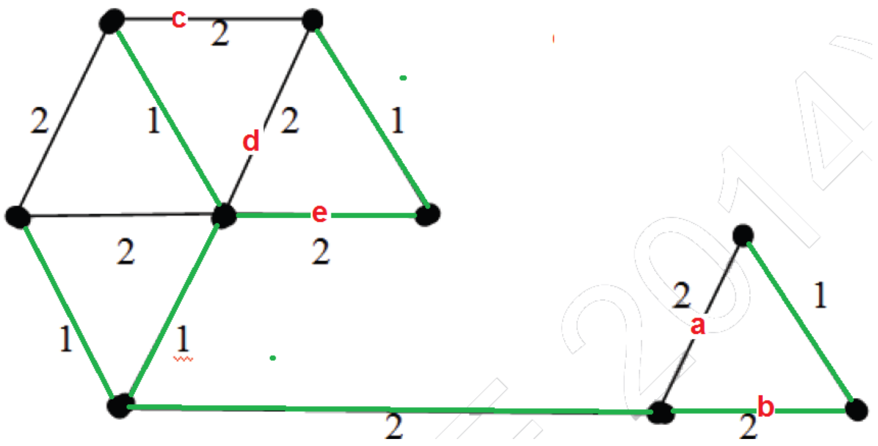
TREE

12. The number of distinct minimum spanning trees for the weighted graph below is ____(2014)

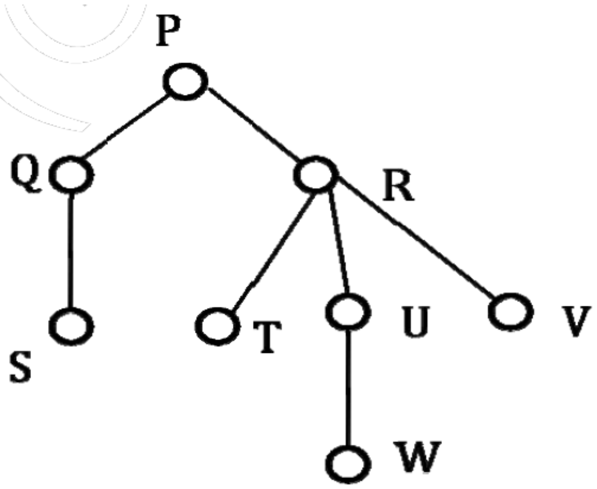


Answer:6

Highlighted (in green) are the edges picked to make a MST. In the right side of MST, we could either pick edge 'a' or 'b'. In the left side, we could either pick 'c' or 'd' or 'e' in MST.There are 2 options for one edge to be picked and 3 options for another edge to be picked. Therefore, total 2*3 possible MSTs.



13)Consider the following rooted tree with the vertex P labeled as root (2013)



The order in which the nodes are visited during in-order traversal is

(A)SQPTRWUV

(B)SQPTURWV

(C)SQPTWUVR

(D) SQPTRUWV

Answer:(A)

The only confusion in this question is, there are 3 children of R. So when should R appear – after U or after R? There are two possibilities: SQPTRWUV and SQPTWURV.

Only 1st possibility is present as an option A, the IInd possibility is not there. Therefore option A is the right answer.

14)**Consider the tree arcs of a BFS traversal from a source node W in an unweighted, connected, undirected graph. The tree T formed by the tree arcs is a data structure for computing. (2012)**

(A) the shortest path between every pair of vertices.

(B) the shortest path from W to every vertex in the graph.

(C) the shortest paths from W to only those nodes that are leaves of T.

(D) the longest path in the graph

Answer: (B)

BFS always produces shortest paths from source to all other vertices in an unweighted graph. The reason is simple, in BFS, we first explore all vertices which are 1 edge away from source, then we explore all vertices which are 2 edges away from the source and so on. This property of BFS makes it useful in many algorithms like Edmonds–Karp algorithm.

15)**Consider a rooted Binary tree represented using pointers. The best upper bound on the time required to determine the number of subtrees having having exactly 4 nodes O(n^a Logn ^{b}). Then the value of a + 10b is ________ (2013)**

Answer:1

Explanation: We can find the subtree with 4 nodes in O(n) time. Following can be a simple approach.

1) Traverse the tree in bottom up manner and find size of subtree rooted with current node

2) If size becomes 4, then print the current node.

```
intprint4Subtree(structNode *root)
{
   if(root == NULL)
     return0;
   intl =  print4Subtree(root->left);
   intr =   print4Subtree(root->right);
   if((l + r + 1) == 4)
      printf("%d ", root->data);
```

```
        return(l + r + 1);
    }
```

16. The preorder traversal sequence of a binary search tree is 30, 20, 10, 15, 25, 23, 39, 35, 42. Which one of the following is the postorder traversal sequence of the same tree? (2012)

(A) 10, 20, 15, 23, 25, 35, 42, 39, 30
(B) 15, 10, 25, 23, 20, 42, 35, 39, 30
(C) 15, 20, 10, 23, 25, 42, 35, 39, 30
(D) 15, 10, 23, 25, 20, 35, 42, 39, 30

Answer (D)

The following is the constructed tree

```
        30
      /    \
     20     39
    / \    / \
  10   25 35  42
   \   /
   15 23
```

17. **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? (2011)**

(A)O(1)
(B)O(logn)
(C)O(n)
(D) O(n log n)

Answer (C)

The worst case occurs for a skewed tree. In a skewed tree, when a new node is inserted as a child of bottommost node, the time for insertion requires traversal of all node. For example, consider the following tree and the case when something smaller than 70 is inserted.

```
        100
        /
       90
      /
     80
    /
   70
```

18. The worst case running time to search for an element in a balanced in a binary search tree with n2^n elements is(2013)

(A) Θ(n log n)

(B) Θ($n2^n$)

(C) Θ(n)

(D) Θ(log n)

Answer (C)

Time taken to search an element is Θ(h) where h is the height of Binary Search Tree (BST). The growth of height of a balanced BST is logerthimic in terms of number of nodes. So the worst case time to search an element would be Θ(Log(n*2^n)) which is Θ(Log(n) + Log(2^n)) Which is Θ(Log(n) + n) which can be written as Θ(n).

19) The height of a tree is defined as the number of edges on the longest path in the tree. The function shown in the pseudocode below is invoked as height (root) to compute the height of a binary tree rooted at the tree pointer root.(2015)

```
int height (treeptr n)
{ if (n== NULL) return -1;
  if (n → left == NULL)
       if (n → right ==NULL) return 0;
       else return BI;                    // Box 1
else {h1 = height (n → left);
       if (n → right == NULL) return (1 + h1);
       else {h2 = height (n → right);
              return B2 ;                 // Box 2
              }
       }
}
```

The appropriate expression for the two boxes B1 and B2 are

(A) B1 : (1 + height(n->right)), B2 : (1 + max(h1,h2))

(B) B1 : (height(n->right)), B2 : (1 + max(h1,h2))

(C) B1 : height(n->right), B2 : max(h1,h2)

(D) B1 : (1 + height(n->right)), B2 : max(h1,h2)

Answer (A)

The box B1 gets exected when left subtree of n is NULL and right sbtree is not NULL. In this case, height of n will be height of right subtree plus one. The box B2 gets executed when both left and right sbtrees of n are not NULL. In this case, height of n will be max of heights of left and right sbtrees of n plus 1.

UNIT IV

20. You have an array of n elements. Suppose you implement quick sort by always choosing the central element of the array as the pivot. Then the tightest upper bound for the worst case,performance is(2015)

(A)O(n^2)

(B)O(nLogn)

(C)Θ(nLogn)

(D) O(n^3)

Answer:(A)

The middle element may always be an extreme element (minimum or maximum) in sorted order, therefore time complexity in worst case becomes O(n^2)

21. Let P be a QuickSort Program to sort numbers in ascending order using the first element as pivot. Let t1 and t2 be the number of comparisons made by P for the inputs {1, 2, 3, 4, 5} and {4, 1, 5, 3, 2} respectively. Which one of the following holds?(2013)

(A)t1=5

(B)t1<t2

(C)t1>t2

(D) t1 = t2

Answer:(C)

Explanation: When first element or last element is chosen as pivot, Quick Sort's worst case occurs for the sorted arrays.In every step of quick sort, numbers are divided as per the following recurrence.T(n) = T(n-1) + O(n)

22. The number of elements that can be sorted in Θ(logn) time using heap sort is (2012)

(A) Θ(1)

(B) Θ(sqrt(logn))

(C) Θ(Log n/(Log Log n))

(d) Θ(Log n)

Answer (C)

Time complexity of Heap Sort is Θ(mLogm) for m input elements. For m = Θ(Log n/(Log Log n)), the value of Θ(m * Logm) will be Θ([Log n/(Log Log n)] * [Log (Log n/(Log Log n))]) which will be Θ([Log n/(Log Log n)] * [Log Log n - Log LogLog n]) which is Θ(Log n)

23. Which one of the following is the tightest upper bound that represents the number of swaps required to sort n numbers using selection sort? (2014)

(A)O(logn)

(B)O(n)

(C)O(nlogn)

(D) O(n^2)

Answer(B)

Selection sort requires only O(n) swaps

24. **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(2011)**

(A)O(nlogn)

(B)O(n^2 logn)

(C)O(n^2+logn)

(D) O(n^2)

Answer(B)

The recurrence tree for merge sort will have height Logn. And O(n^2) work will be done at each level of the recurrence tree (Each level involves n comparisons and a comparison takes O(n) time in worst case). So time complexity of this Merge Sort will be O(n^2 log n).

UNIT V

25. Let G be a graph with n vertices and m edges. What is the tightest upper bound on the running time on Depth First Search of G? Assume that the graph is represented using adjacency matrix.(2014)

(A)O(n)

(B)O(m+n)

(C)O(n^2)

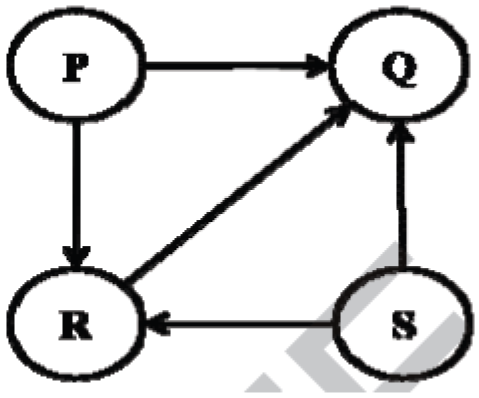
(D) O(mn)

Answer:(C)

Explanation: Depth First Search of a Tree takes O(m+n) time when the graph is represented using adjacency list.

In adjacency matrix representation, graph is represented as an "n x n" matrix. To do DFS, for every vertex, we traverse the row corresponding to that vertex to find all adjacent vertices (In adjacency list representation we traverse only the adjacent vertices of the vertex). Therefore time complexity becomes O(n^2).

26. Consider the directed graph given below. Which one of the following is TRUE?(2013)



(A) The graph doesn't have any topological ordering
(B) Both PQRS and SRPQ are topological ordering
(C) Both PSRQ and SPRQ are topological ordering
(D) PSRQ is the only topological ordering
Answer:(C)
Explanation: The graph doesn't contain any cycle, so there exist topological ordering. P and S must appear before R and Q because there are edges from P to R and Q, and from S to R and Q.

27. Which of the following statements is/are TRUE for an undirected graph?(2012)

P: Number of odd degree vertices is evenQ: Sum of degrees of all vertices is even
A)POnly
B)QOnly
C)BothPandQ
D) Neither P nor Q
Answer(C)
Q is true: Since the graph is undirected, every edge increases the sum of degrees by 2. P is true: If we consider sum of degrees and subtract all even degrees, we get an even number (because Q is true). So total number of odd degree vertices must be even.

28. **Consider an undirected random graph of eight vertices. The probability that there is an edge between a pair of vertices is 1/2. What is the expected number of unordered cycles of length three? (2011)**

(A)1/8

(B)1

(C)7

(D) 8

Answer(C)

A cycle of length 3 can be formed with 3 vertices. There can be total 8C3 ways to pick 3 vertices from 8. The probability that there is an edge between two vertices is 1/2. So expected number of unordered cycles of length 3 = (8C3)*(1/2)^3 = 7

29. **Which of the following statements are TRUE?** (2013)

(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.

(A)1,2and3

(B)1and2only

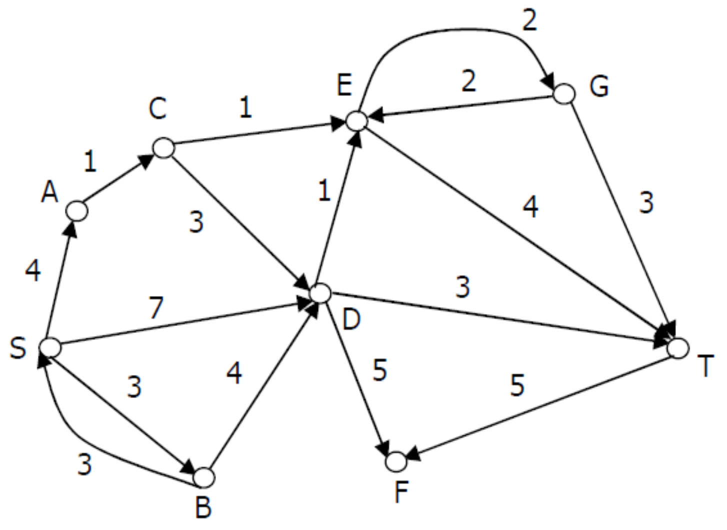
(C)2and3only

(D) 1 and 3 only

Answer(A)

- **1** is true because cycle detection can be done in polynomial time using DFS .
- **2** is true because P is a subset of NP.
- **3** is true because NP complete is also a subset of NP and NP means **N**on-deterministic **P**olynomial time solution exists.

30. Consider the directed graph shown in the figure below. There are multiple shortest paths between vertices S and T. Which one will be reported by Dijstra's shortest path algorithm? Assume that, in any iteration, the shortest path to a vertex v is updated only when a strictly shorter path to v is discovered.(2014)



(A) SDT

(B) SBDT

(C) SACDT

(D) SACET

Answer (D)

31. What is the time complexity of Bellman-Ford single-source shortest path algorithm on a complete graph of n vertices? (2012)

(A) $\Theta(n^2)$

(B) $\Theta(n^2 \text{ Logn})$

(C) $\Theta(n^3)$

(D) $\Theta(n^3 \text{ Logn})$

Answer (C).

Time complexity of Bellman-Ford algorithm is $\Theta(VE)$ where V is number of vertices and E is number edges. If the graph is complete, the value of E becomes $\Theta(V^2)$. So overall time complexity becomes $\Theta(V^3)$.