Page No.
Date

College Name :

DELHI GLOBAL INSTITUTE OF TECHNOLOGY

Name : BAZGHA RAZI

Course Code : PCC-CSE-203 G

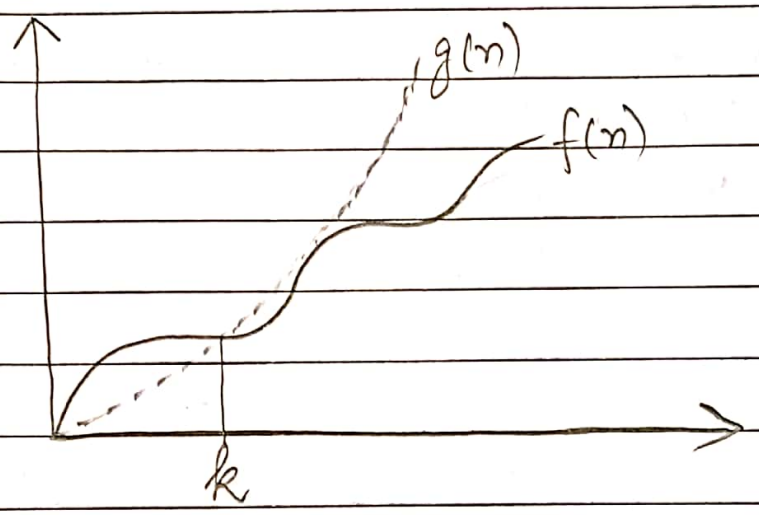
Subject : Data Structure & Algorithm

Session : 2019-2023

Page No.
Date

Ans 1 b) Big "O" Notation

Big Oh is a characteristic scheme that measures properties of algorithm complexity performance or memory requirements.
It defines the upper bound of any algorithm i.e., your algorithm can't take more time than this time.
It denotes the maximum time taken by an algorithm or the worst-case time complexity of an algorithm.



For example, for a function f(n)

$O(f(n)) = \{ g(n) : \text{there exists } c>0 \text{ and } n_0 \text{ such that } f(n) \leq c \cdot g(n) \text{ for all } n > n_0 \}$

Date

Big O Notation has following limitations:

i) It contains no effort to improve the programming methodology. It doesn't improve the efficiency of the program but it helps to analyze and calculate the efficiency of the program.

ii) It does not exhibit the potential of the constants.

iii) There are numerous algorithms are the way too difficult to analyze mathematically.

iv) It tells us how the algorithm grows with the size of the problem and not the efficiency related to it.

v) It only gives sensible comparisons of algorithms in different complexity classes when n is large.

Ans 1e) Dynamic Memory Allocation

The process of allocating memory at the time of execution is called dynamic memory allocation. User has the flexibility to allocate memory according to his/her needs.

The memory comes from the static part of the data segment. Programs may request memory and may also return previously dynamically allocated memory. Memory may be returned whenever it is no longer needed.

Dynamically allocated space usually placed in a program segment known as the heap.

Heap is the segment of memory where dynamic memory allocation takes place.

Unlike stack where memory is allocated or deallocated in a defined order, heap is an area of memory where memory is allocated or deallocated without any order or randomly.

Pointers play an important role in dynamic memory allocation. Allocated memory can only be accessed through pointers.

Advantages of allocating memory dynamically:

- When we do not know how much amount of memory would be needed for the program beforehand.
- When we want data structures without any upper limit of memory space.
- Dynamically created lists insertions and deletions can be done very easily just by the manipulation of addresses whereas in case of statically allocated memory insertions and deletion leads to more movements and wastage of memory.

Page No.
Date

ans 26) Strings are defined as an array of character. Strings in C are stored as null character, '\0', terminated character array i.e., the length of a string is the number of characters it contains one more to store the null character. Common string operations include finding lengths, copying, searching, replacing and counting the occurences of specific characters and words.

Syntax : char str_name [size];

Various operation performed in strings :

- strcat : This is a function appends a source string to the end of the destination string. This returns a pointer to the description string or NULL pointer on error.

Example → char * strcat (char * dst, const char* src);

- strncat : It appends at most N characters from the source string to the end of the destination string. It returns a pointer to the destination string.

Example → char *strncat (char* dst, const char *src, size N)

Page No.
Date

- strcpy : It copies a string, including the null character terminator from the source string to the destination. It returns a pointer to the destination string.

Example → char *strcpy (char*dst, const char *src);

- strncpy : It is similar to strcpy, but it allows the number of characters to be copied to be specified.

Example → char* strncpy (char* dst, const char* size_tlen);

- strlen: It returns the length of a string, not counting the null character at the end. It reurns the character count of the string, without terminator.

Example → size_t strlen (const char * str);

- strcmp : Two strings are compared by using strcmp. If the first string is greater than the second it returns a number greater than zero. If the second string is greater, it returns a number less than zero. If the strings are equal than it return 0.

Example → int strcmp (const char * first, const char * second);

- strncmp: This compares the first N characters of each string. If the first string is greater than the second, it returns a number greater than the second, it returns a number greater than zero. If the second string is greater, it returns the number less than zero. If the strings are equal, it returns 0.

Example → int strncmp(const char* first, const char* second, size_t N);

- memset: It is useful to initialize a string to all nulls, or to any character.

Example → void* memset (const void *dst, int c, size_t N);

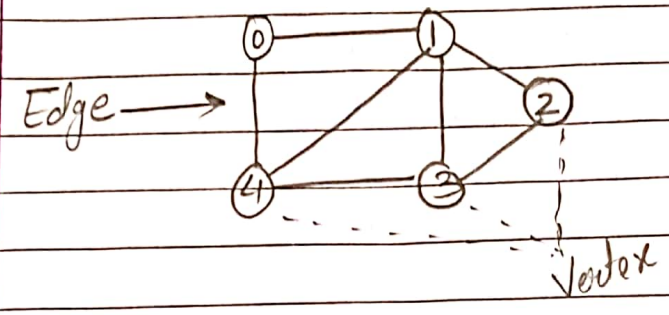
- strchr: Returns a pointer to the first occurrence of character ch in string s1.

Syntax: strchr (s1, ch);

- strstr: Returns a pointer to the first occurrence of string s2 in string s1.

Syntax: strstr (s1, s2);

Ans 35) The graph is a non-linear data structure consisting of nodes and edges. The nodes are sometimes also refferred as vertices and the edges are lines that connect any two nodes in the graph.



Graph Traversal

It is a technique used for searching vertex in a graph. It is also used to decide the order of vertices is visited in the search process. It finds the edges to be used in the search process without creating loops.

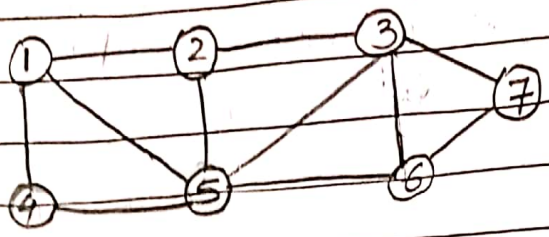
There are two graph traversal techniques are as follows:

1) BFT (Breadth First Traversal)

It produces a spanning tree as final result. Spanning tree is basically a graph without loops.

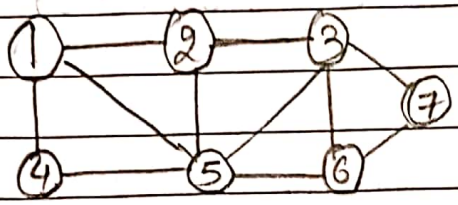
Page No.
Date

Example:



This is the example of graph to perform breadth first traversal.

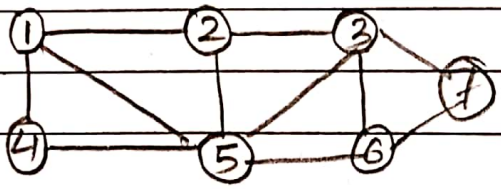
1) Select vertex 1 as starting point i.e., visit 1.
Insert 1 into the queue



Queue

1						

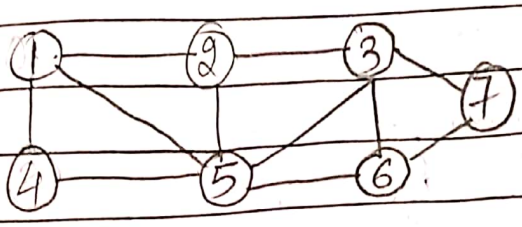
2) Visit all adjacent vertices of 1 which are not visited i.e., 2, 4, 5.
Insert newly visited vertices into the Queue and delete 1 from the queue.



Queue

4	5	2			

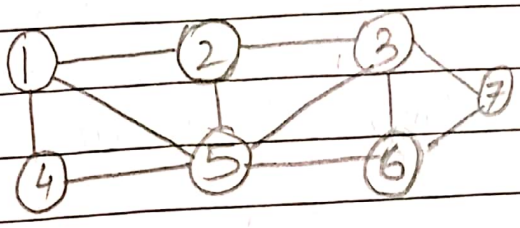
3) Vist all adjacent vertices of 4 which are not visited. So after visting there are no vertex So Delete 4 from the Queue.



Queue

		5	2		

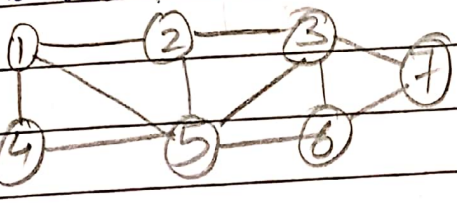
4) Visit all adjacent vertices of 5 which are not visited i.e., 3 and 6.
Insert newly visited vertices into Queue and delete 5 from the Queue.



Queue.

			2	3	6	

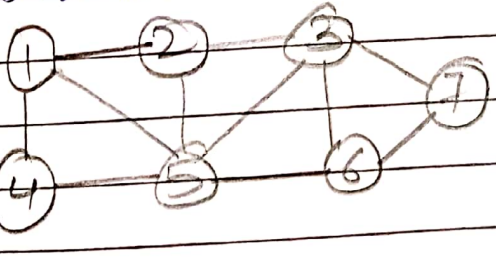
5) Visit all adjacent vertices of 3 which are not visited. So after visiting, there ~~are~~ is no vertex. Delete 3 from the Queue.



Queue

				3	6	

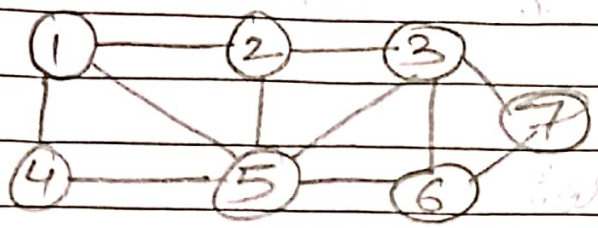
6) Visit all adjacent vertices of ~~3~~ which are not visited. So after visiting, there is vertex 7 which is not visited yet. Insert newly visited vertex into the Queue and delet 3 from que



Queue

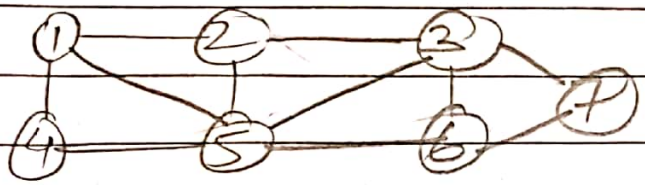
					6	7

7) Visit all adjacent vertices of 6 which are not visited. So after visiting, there is no vertex. So, Delete 6 from the Queue.



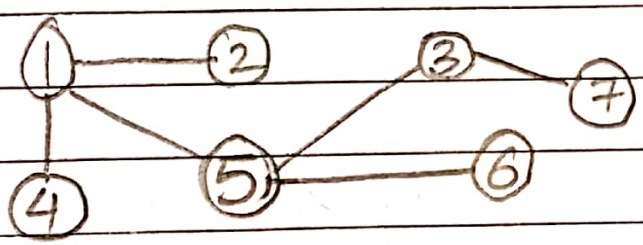
Queue

8) Visit all adjacent vertices of 7 which are not visited. So, after visiting there is no vertex. So, Delete 7 from the queue.



Queue

Queue become empty. So stop the breadth first traversal. Find result is as follows;

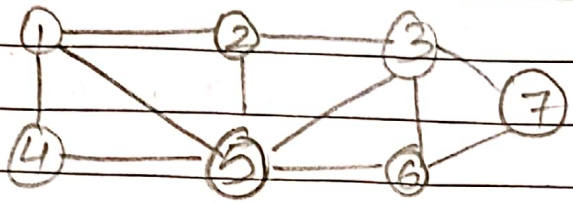


Page No.
Date

2) DFT (Depth First Traversal)

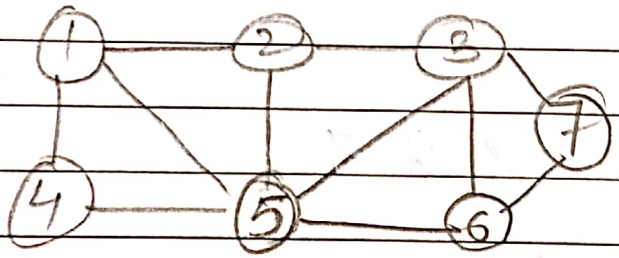
It produces a spanning tree as final result.
Spanning tree is graph without loops.
It use stack to perform the traversal.

Example:



This is the example of graph to perform depth first traversal.

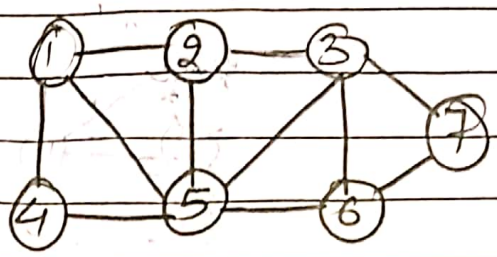
1) Select the vertex 1 as starting point i.e., visit 1
Push 1 on to the stack.

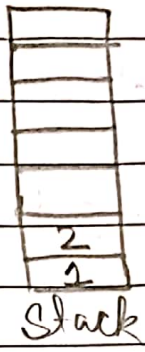


Stack

Page No.
Date

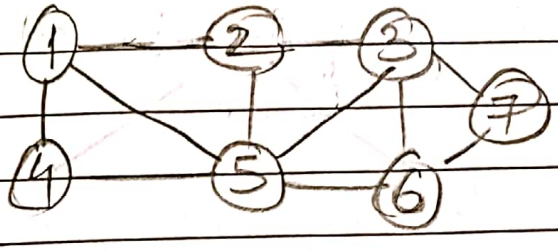
2. Visit any adjacent vertex of 1 which is not visited i.e., 2.
Push newly visited vertex 2 into the stack

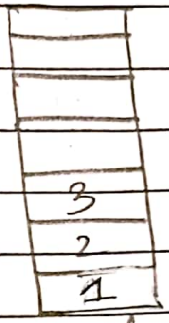




Stack

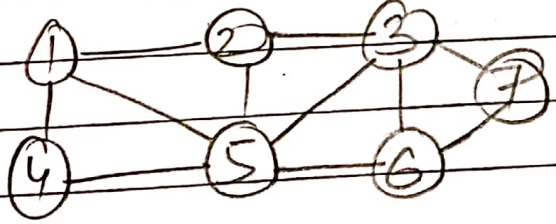
3. Visit any adjacent vertex of 2 which is not visited i.e., 3
Push 3 on to the stack.





Stack.

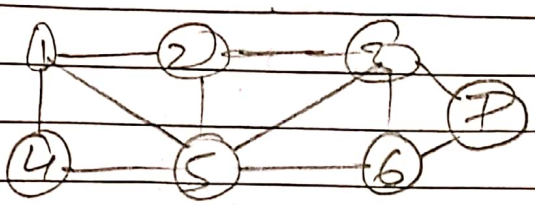
4. Visit any adjacent vertex of 5 which is not visited i.e., 5.
Push 5 on to the stack



Stack.

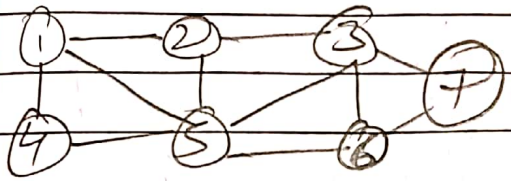
Page No.
Date

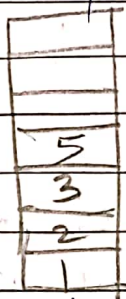
5. Visit any adjacent vertex of 5 which is not visited i.e., 4.
Push 4 on to the stack.



stack

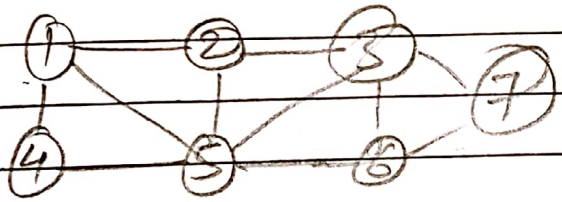
6. There is no new vertex to be visited from 4.
So use back track and pop 4 from the stack.

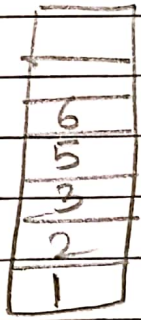




stack

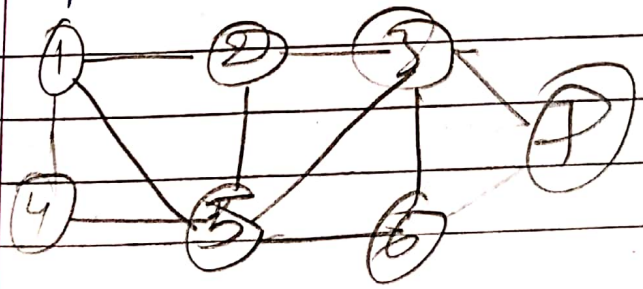
7. Visit any adjacent vertex of 5 which is not visited i.e., 6.
Push 6 on to the stack.

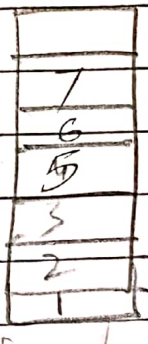




stack

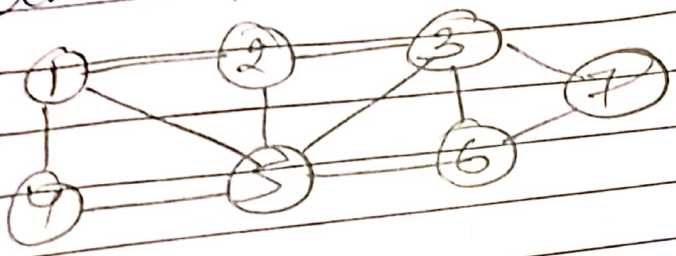
8. Visit any adjacent vertex of 6 which is not visited i.e., 7.
Push 7 onto the stack.





stack.

9. There is no new vertex to be visited from 6. So use back track and pop every vertex from the stack one by one i.e., first pop 7, then 6, then 5, the 3, and so on.



So, stack become empty. So stop depth first traversal.
Result of depth first traversal is following spanning tree.