

Contents

Chapter Wise all semester Final & Mid Ques Solved!

Note: ২ ঘন্টায় যতটুকু পারার টাই করছি। বাট **Chapter 6, 21,22,23,32** বেশি **important** আমার কাছে মনে হয়।

Chapter 1 & 3

18-19 mid1

1(a) . What do you mean by datatype? Is Array a data structure? Justify your answer.

A datatype is a classification of data that determines the type of operations that can be performed on it. In programming, data types are used to define variables, functions, and parameters in a program.

An array is a data structure that stores a collection of elements of the same data type in contiguous memory locations. As such, it is not a data type but a data structure that uses a specific data type to store its elements.

Arrays are a fundamental data structure in programming and are used to store and manipulate collections of data efficiently. They allow for quick access to individual elements and support operations such as sorting and searching.

Arrays also support dynamic resizing, making them versatile for handling collections of varying sizes. They are commonly used in algorithms, data processing, and database management.

In summary, while arrays are not a data type, they are a crucial data structure that relies on a specific data type to store its elements.

18-19 mid1

1(b) .What do you mean by random-access machine?

A random-access machine (RAM) is a theoretical model of computation that performs calculations on a stored program. It is a type of computer architecture that allows for constant-time access to any memory location, irrespective of the memory's physical location.

The RAM model assumes that every operation takes the same amount of time, and its performance is typically measured in terms of the number of basic operations performed.

The RAM model is used to analyze the time and space complexity of algorithms, and it is a widely used theoretical framework for studying computation.

17-18 final

1(a). Write down the definitions of algorithms as technology.

An algorithm is a set of instructions or rules that a computer program follows to accomplish a task. It is a sequence of steps that a computer executes to solve a problem.

Algorithms are considered a technology because they enable the creation of software applications that can automate complex tasks. They provide a systematic and efficient

approach to solving problems, which is critical for developing software systems that can handle large volumes of data.

Algorithms can be designed to perform a wide range of functions, from sorting and searching data to processing images and speech. They are a key component of artificial intelligence and machine learning, which rely on complex algorithms to analyze and learn from data.

Algorithms are also used in a variety of industries, including finance, healthcare, and transportation, to optimize resource usage, improve efficiency, and reduce costs.

In summary, algorithms are a technology that enables the development of powerful software applications that can automate complex tasks, analyze data, and make intelligent decisions.

16-17 final

1(a). Why are data structures and algorithms so important in Computer science?

They provide a way to organize and manipulate data efficiently.

They help optimize resource usage, such as memory and processing power.

They are fundamental building blocks for developing complex software systems.

They enable software engineers to solve complex problems in a systematic and structured way.

They improve the performance and scalability of software systems.

They are widely used in industries such as finance, healthcare, and transportation.

They are a critical component of coding interviews for tech jobs.

Mastery of data structures and algorithms is a hallmark of a competent software engineer.

8(a). Define algorithms and Data structures? As a CSE student, why should you have good knowledge on algorithms?

An algorithm is a set of instructions or rules that a computer program follows to accomplish a task. It is a sequence of steps that a computer executes to solve a problem. A data structure, on the other hand, is a way of organizing and storing data in a computer program, so it can be accessed and used efficiently.


As a computer science engineering (CSE) student, having good knowledge of algorithms is essential because algorithms are the building blocks of software systems. They enable you to develop efficient and effective solutions to complex problems and help optimize resource usage, such as memory and processing power.

Data structures, on the other hand, provide a way to organize and manipulate data efficiently, which is critical for developing complex software systems. Understanding different data structures and their operations can help you choose the right one for a given problem and develop software that performs optimally.

Overall, good knowledge of algorithms and data structures is essential for CSE students as it is critical to developing high-quality software and solving complex problems efficiently. It is a fundamental component of computer science and is widely used in industry and research.

Chapter 4

For a better understanding of TOH & LCS You can see the videos which I provide below:

For TOH :  Tower of Hanoi Problem - Made Easy

For LCS : [LCS - Longest Common Subsequence - Algorithms \(Bangla Tutorial\)](#)

Big mod er link:

<https://imranshabijabi.wordpress.com/2012/11/24/%E0%A6%AC%E0%A6%BF%E0%A6%97-%E0%A6%AE%E0%A7%8B%E0%A6%A1-%E0%A6%85%E0%A7%8D%E0%A6%AF%E0%A6%BE%E0%A6%B2%E0%A6%97%E0%A7%8B%E0%A6%B0%E0%A6%BF%E0%A6%A6%E0%A6%AEbig-mod-algorithm/>

Ques and Ans part:

Q: Define divide and Conquer. Solve the problem $5^{61} \% 9$ using divide and conquer technique.

Ans: Divide and Conquer is recursively breaks down a problem into two or more sub-problems of the same or related type.

Solved the $5^{61} \% 9$:-

We know,

$$\begin{aligned} a^p \% q &= (a \cdot a^{p-1}) \% q \\ &= ((a \% q) * (a^{p-1} \% q)) \% q \end{aligned}$$

So,

Big-Mod

$\text{BIG-MOD}(\text{base}, \text{power}, \text{mod})$

if $(\text{power} == 0)$

return 1

else if $(\text{power} \% 2 == 1)$

$P1 = \text{base} \% \text{mod}$

$P2 = \text{BIG-MOD}(\text{base}, \text{power}-1, \text{mod})$

return $(P1 * P2) \% \text{mod}$

else if $(\text{power} \% 2 == 0)$

$P1 = \text{BIG-MOD}(\text{base}, \text{power}/2, \text{mod})$

return $(P1 * P1) \% \text{mod}$

This is the algorithm to solve modular problems
using divide and conquer with the procedure.

22



* বিগত বহু গুণিত ৮৫ নাম গুলো আছে:-

$5^{67} \% 7$, $7^{87} \% 9$, $9^{19} \% 7$, $3^{86} \% 7$, $3^{29} \% 9$.

সকলোয় সিফটনাম সব খাতি রপ্ত করি ও বকল স্ম.

LCS
Topic

☐ What do you mean dynamic Programming?

Write down the program to find out the longest common subsequence on the sequences:-

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

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

Solⁿ: Dynamic Programming: Dynamic Programming like the divide and conquer method, solves problems by combining combining the solutions to subproblems.

Simulation

$X = \langle B, A, B, C, B, D, A, B \rangle$

$Y = \langle B, D, C, A, B, A \rangle$

		B	D	C	A	B	A
		0	0	0	0	0	0
A		0 → 0	0 → 0	0 → 0	1 → 1	1 → 1	1
B		0	1 → 1	1 → 1	1 → 1	2 → 2	2
C		0	1	1 → 1	2 → 2	2 → 2	2
B		0	1	1	2 → 2	3 → 3	3
D		0	1	2 → 2	2 → 2	3 → 3	3
A		0	1	2	2 → 2	3 → 3	4
B		0	1	2	2	3 → 3	4 → 4

a	
a	

match

	c
b	

not

match

b or c

[bigger]

B D A B

Q: Write down the recursive formula for TOH for 3 disks of

the "Tower of Hanoi" problems.

Solⁿ: The tower of hanoi consists of three rods and a number of disks of different parameters.

If there are n disk. We write:-

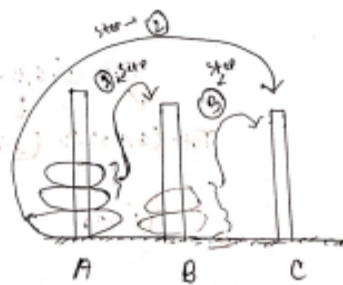
TOH (n, A, B, C)

{
if $n > 0$

TOH ($n-1, A, C, B$)

print ($A \rightarrow C$)

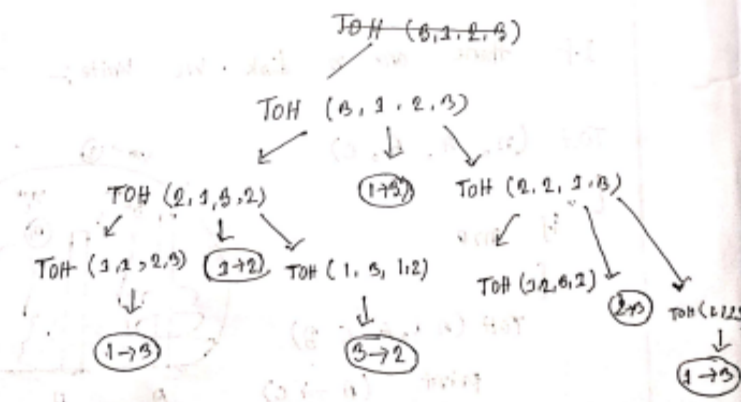
TOH ($n-1, B, A, C$)



From algorithm, we write the TOH in 3 step:

- ① Move $n-1$ disk from A to B using C.
- ② Move a disk from A to C.
- ③ Move $n-1$ disk from B to C using A.

For 3 disks:-



So, the moving path:-

1 → 3, 1 → 2, 3 → 2, 1 → 3, 2 → 1, 2 → 3, 3 → 1

Scanned with CamScanner

Here, is 7 moves for 3 disk:-

$$2^3 - 1 = 7.$$


Note: - যেকোনো n টি ডিস্কের জন্য $2^n - 1$ টি মোবাইল প্রয়োজন।
 2 disks দিলে মোবাইল 3 টি হবে।
 3 disks দিলে মোবাইল 7 টি হবে।
 4 disks দিলে মোবাইল 15 টি হবে।
 5 disks দিলে মোবাইল 31 টি হবে।
 6 disks দিলে মোবাইল 63 টি হবে।
 7 disks দিলে মোবাইল 127 টি হবে।
 8 disks দিলে মোবাইল 255 টি হবে।
 9 disks দিলে মোবাইল 511 টি হবে।
 10 disks দিলে মোবাইল 1023 টি হবে।

AVAILABLE AT:

Onebyzero Edu - Organized Learning, Smooth Career

The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

Chapter 6

To learn Heap Sort:  2.6.3 Heap - Heap Sort - Heapify - Priority Queues

18-19 mid-1

2(b) . Write down the algorithm of Heap sort with example. [4]

```
HEAPSORT(A)
1  BUILD-MAX-HEAP(A)
2  for i = A.length downto 2
3      exchange A[1] with A[i]
4      A.heap-size = A.heap-size - 1
5      MAX-HEAPIFY(A, 1)
```

Example .###17-18 final Data structure 2(c) as same

18-19 mid-1

2(c). Write pseudo code for the procedures HEAP_DECREASE_KEY that implements a min priority queue with a min-heap

5. (CLRS 6.5-3) Write pseudocode for the procedures HEAP-EXTRACT-MIN, HEAP-DECREASE-KEY and HEAP-INSERT that implement a min-priority queue with a min-heap.

Solution:

```
HEAP-MINIMUM(A)
    return A[1]
```

```
HEAP-EXTRACT-MIN(A)
    if heap-size[A] < 1
        then error "heap underflow"
    min ← A[1]
    A[1] ← A[heap-size[A]]
    heap-size[A] ← heap-size[A] - 1
    MIN-HEAPIFY(A,1)
    return min
```

```
HEAP-DECREASE-KEY(A,i,key)
    if key > A[i]
        then error "new key is larger than current key"
    A[i] ← key
    while i > 1 and A[parent(i)] > A[i]
        do exchange A[i] ↔ A[parent(i)]
        i ← parent(i)
```

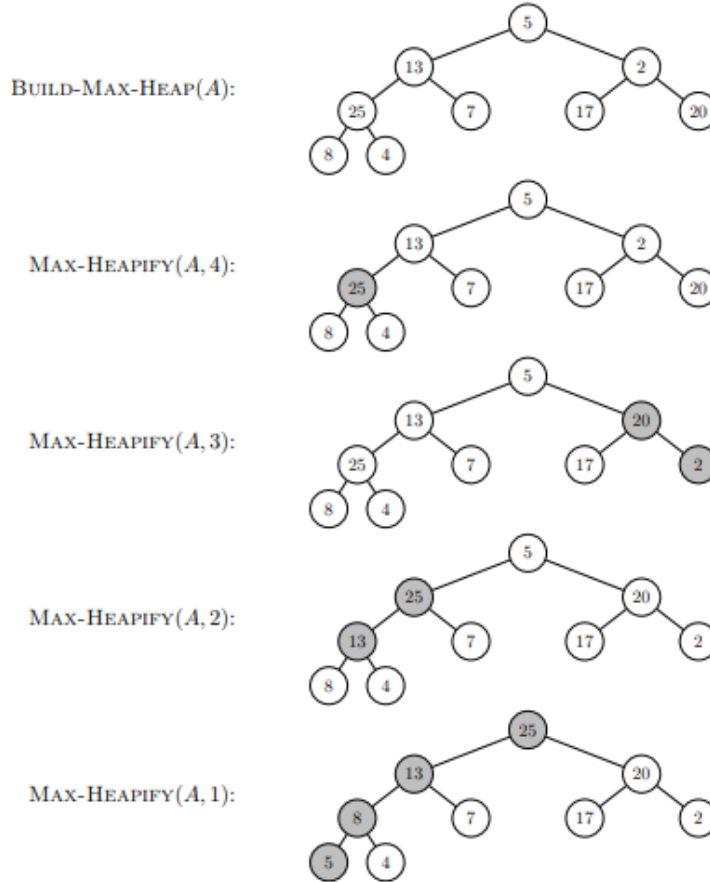
```
MIN-HEAP-INSERT(A,key)
    heap-size[A] ← heap-size[A] + 1
    A[heap-size[A]] ← +inf
    HEAP-DECREASE-KEY(A,heap-size[A],key)
```

17-18 final

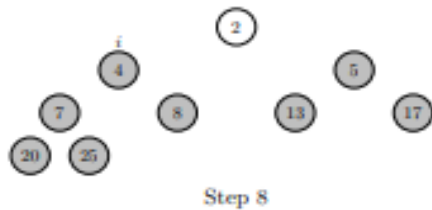
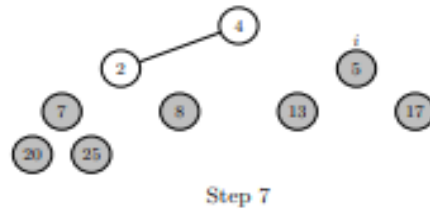
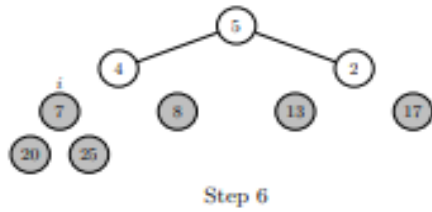
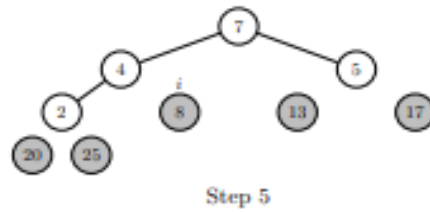
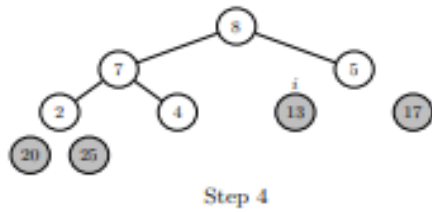
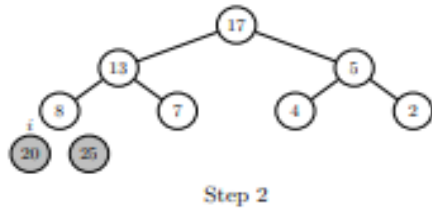
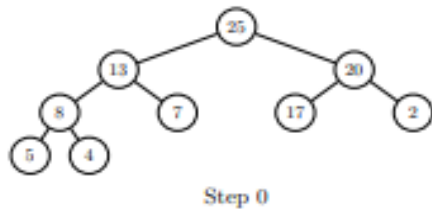
2(c). Write down the operation of HEAPSORT on the array $A = \langle 5, 13, 2, 25, 7, 17, 20, 8, 4 \rangle$

6.4-1 (p.160) Using Figure 6.4 as a model, illustrate the operation of HEAPSORT on the array $A = \langle 5, 13, 2, 25, 7, 17, 20, 8, 4 \rangle$.

Since $A.length = 9$, the command $\text{MAX-HEAPIFY}(A, i)$ is called for $i = 4, 3, 2, 1$. The action of BUILD-MAX-HEAP is as follows (these first few diagrams are not required for a correct answer), with the nodes exchanged at each step shaded:



Now we follow Figure 6.4 (these diagrams are required for a correct answer):



This gives a final sorted array $A = \langle 2, 4, 5, 7, 8, 13, 17, 20, 25 \rangle$.

16-17 final

1(c). What do you mean by priority queue? Write down the operations of the priority queue? Write down the program to max heapify of an array?

A priority queue is a type of data structure that stores a collection of elements with associated priorities. Each element in the queue is assigned a priority, and the queue maintains the order of the elements based on their priority.

The operations of a priority queue are:

Insertion: Add an element to the queue with its associated priority.

Deletion: Remove the element from the queue with the highest priority.

Peek: Get the element with the highest priority without removing it from the queue.

Change Priority: Modify the priority of an element already in the queue.

Merge: Combine two priority queues into a single queue while maintaining the order of the elements based on their priorities.

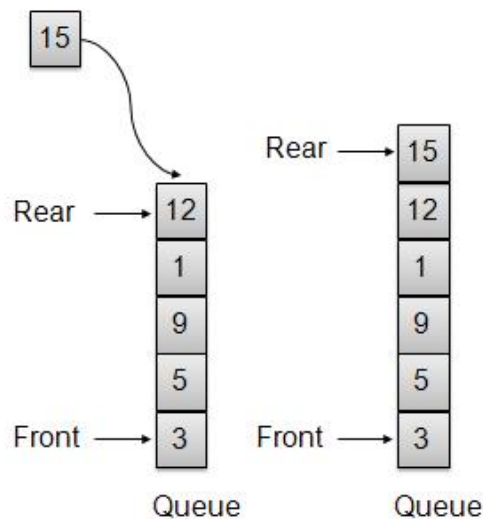
Priority queues are commonly used in algorithms such as Dijkstra's shortest path algorithm and Huffman coding. They are also used in computer networking, operating systems, and other applications where scheduling or resource allocation is required.

Operations

- A max-priority queue provides the following operations:
 - **insert:** add an element to the priority queue.
 - **maxElement:** return the largest element in the priority queue.
 - **removeMaxElement:** remove the largest element from the priority queue.

Insert / Enqueue Operation

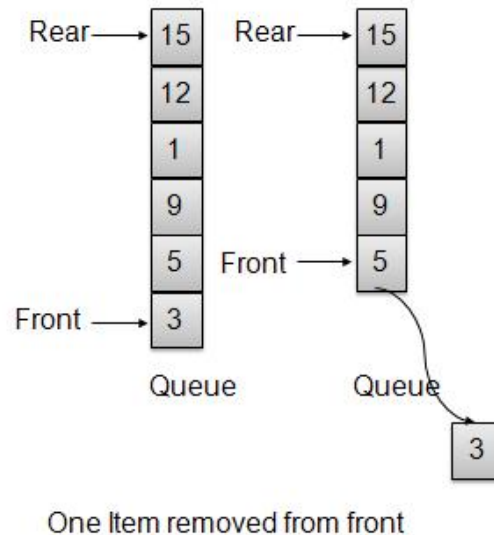
Whenever an element is inserted into queue, priority queue inserts the item according to its order. Here we're assuming that data with high value has low priority.



One item inserted at rear end

Remove / Dequeue Operation

Whenever an element is to be removed from queue, queue get the element using item count. Once element is removed. Item count is reduced by one.



****code:**

<https://www.geeksforgeeks.org/building-heap-from-array/>

// C++ program for building Heap from Array

```
#include <iostream>
```

```
using namespace std;
```

```
// To heapify a subtree rooted with node i which is
```

```
// an index in arr[]. N is size of heap
```

```
void heapify(int arr[], int n, int i)
```

```
{
```

```
    int largest = i; // Initialize largest as root
```

```
    int l = 2 * i + 1; // left = 2*i + 1
```

```
    int r = 2 * i + 2; // right = 2*i + 2
```

```

// If left child is larger than root
if (l < n && arr[l] > arr[largest])
    largest = l;

// If right child is larger than largest so far
if (r < n && arr[r] > arr[largest])
    largest = r;

// If largest is not root
if (largest != i) {
    swap(arr[i], arr[largest]);

    // Recursively heapify the affected sub-tree
    heapify(arr, n, largest);
}
}

```

```

// Function to build a Max-Heap from the given array
void buildHeap(int arr[], int n)
{
    // Index of last non-leaf node
    int startIdx = (n / 2) - 1;

    // Perform reverse level order traversal
    // from last non-leaf node and heapify
    // each node
    for (int i = startIdx; i >= 0; i--) {
        heapify(arr, n, i);
    }
}

```

```

// A utility function to print the array
// representation of Heap
void printHeap(int arr[], int n)
{
    cout << "Array representation of Heap is:\n";
}

```



```

    for (int i = 0; i < n; ++i)
        cout << arr[i] << " ";
    cout << "\n";
}

// Driver Code
int main()
{
    // Binary Tree Representation
    // of input array
    //
    //           1
    //        /   \
    //       3     5
    //    / \   / \
    //   4  6 13 10
    //  /\ /\
    // 9 8 15 17
    int arr[] = { 1, 3, 5, 4, 6, 13, 10, 9, 8, 15, 17 };

    int n = sizeof(arr) / sizeof(arr[0]);

    buildHeap(arr, n);

    printHeap(arr, n);
    // Final Heap:
    //
    //           17
    //        /   \
    //       15    13
    //    / \   / \
    //   9  6 5 10
    //  /\ /\
    // 4 8 3 1

    return 0;
}

```

16-17 final 2(c) er ii & 17-18 final 2(c)

17-18 final algorithm 2(c):

Write down the procedure of HEAP-MINIMUM and HEAP-EXTRACT-MIN

Sol: Same as 18-19 mid-1 er 2(c)

17-18 final algorithm 2(d):

A heap is an advanced tree-based data structure used primarily for sorting and implementing priority queues. They are complete binary trees

Same as 17-18 final 2(c)

Chapter 7 & 8

Quick sort: [▶ 2.8.1 QuickSort Algorithm](#)

Counting Sort: [▶ Counting Sort | Easiest explanation with example](#)

Chapter 10

2016-17

[1.c] What do u mean by priority queue? Write down the operation of of the priority queue.[.....]

3

Solution:

The **priority queue in the data structure** is an extension of the “normal” queue. It is an abstract data type that contains a group of items. It is like the “normal” queue except that the dequeuing elements follow a priority order. The priority order dequeues those items first that have the highest priority.

It is an abstract data type that provides a way to maintain the dataset. The “normal” queue follows a pattern of first-in-first-out. It dequeues elements in the same order followed at the time of insertion operation. However, the element order in a priority queue depends on the element’s priority in that queue. The priority queue moves the highest priority elements at the beginning of the priority queue and the lowest priority elements at the back of the priority queue.

Operation:

A priority queue is of two types:

- Ascending Order Priority Queue
- Descending Order Priority Queue

Ascending Order Priority Queue

An ascending order priority queue gives the highest priority to the lower number in that queue. For example, you have six numbers in the priority queue that are 4, 8, 12, 45, 35, 20. Firstly, you will arrange these numbers in ascending order. The new list is as follows: 4, 8, 12, 20, 35, 45. In this list, 4 is the smallest number. Hence, the ascending order priority queue treats number 4 as the highest priority.

4	8	12	20	35	45
---	---	----	----	----	----

In the above table, 4 has the highest priority, and 45 has the lowest priority.

Descending Order Priority Queue

A descending order priority queue gives the highest priority to the highest number in that queue. For example, you have six numbers in the priority queue that are 4, 8, 12, 45, 35, 20. Firstly, you will arrange these numbers in ascending order. The new list is as

follows: 45, 35, 20, 12, 8, 4. In this list, 45 is the highest number. Hence, the descending order priority queue treats number 45 as the highest priority.

45	35	20	12	8	4
----	----	----	----	---	---

In the above table, 4 has the lowest priority, and 45 has the highest priority.

[3.a] Describe a situation where storing items in an array is clearly better than storing items on a linked list.

Solution:

When elements must be retrieved by their position.

[For more info]

Disadvantages of using linked lists over arrays-

1. The main disadvantage of linked list over array is *access time to individual elements*. Array is random-access, which means it takes $O(1)$ to access any element in the array. Linked list takes $O(n)$ for access to an element in the list in the worst case.
2. Another advantage of arrays in access time is special locality in memory. Arrays are defined as contiguous blocks of memory, and so any element will be physically near its neighbours. This greatly benefits from modern CPU caching methods.
3. In linked list, Random access is not allowed. We have to access elements sequentially starting from the first node. So we cannot do binary search with linked lists.
4. Extra memory space for a pointer is required with each element of the list. Hence Linked lists waste memory in terms of extra reference points.

b) write down the output of this code with THREE pushes ONE pop

```
intStack s = new intStack();
```

```
s.push(1);
```

```
s.push(2);
```

```
s.push(3);
```

```
s.push(12);
```

```
s.push(14);
```

```
print(s.pop())
```

C. write down the process of ENQUEUE and DEQUEUE using an array.

Solution:

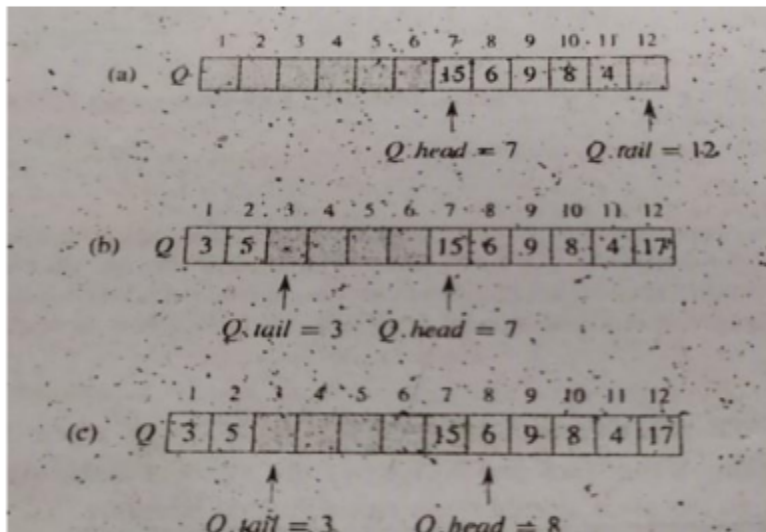


Figure 10.2 A queue implemented using an array $Q[1..12]$. Queue elements appear only in the lightly shaded positions. (a) The queue has 5 elements, in locations $Q[7..11]$. (b) The configuration of the queue after the calls $ENQUEUE(Q, 17)$, $ENQUEUE(Q, 3)$, and $ENQUEUE(Q, 5)$. (c) The configuration of the queue after the call $DEQUEUE(Q)$ returns the key value 15 formerly at the head of the queue. The new head has key 6.

d. Draw a picture of the sequence 13,4,8,19,5,11 stored as a doubly linked list using the multiple array representation. Do the same for the single array representation



2017-18

3

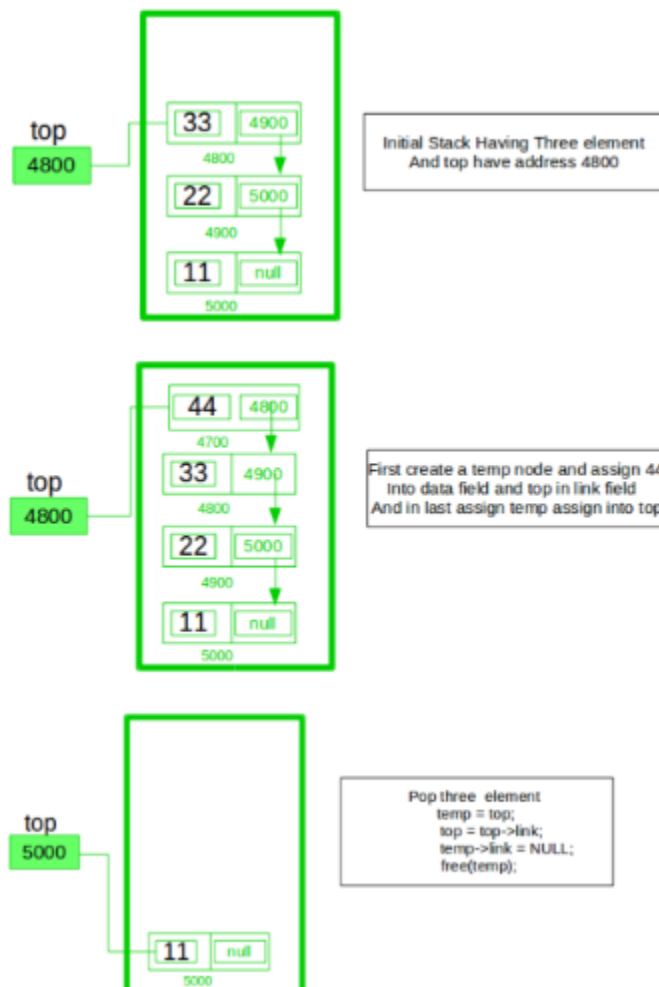
c) Define Stack and Queue. Given the following numbers 10, 4, 6, 8.9 and 3, write

- What would be the value to the stack, if three times push, one time pop, one time push and one time pop. Where push and pop operation refers to insert and pull an element to the stack respectively.
- What would be the value to the queue, if two times enqueue, one time dequeue, three times enqueue and four times dequeue.
- Write down the operations for making empty stack to the numbers.

d) Define linked list. Implement a stack using a singly linked list L The operations PUSH and POP should still take $O(1)$ time.

Solution:

Implement a [stack](#) using single linked list concept. all the single [linked list](#) operations perform based on Stack operations LIFO(last in first out) and with the help of that knowledge we are going to implement a stack using single linked list. using single linked lists so how to implement here it is linked list means what we are storing the information in the form of nodes and we need to follow the stack rules and we need to implement using single linked list nodes so what are the rules we need to follow in the implementation of a stack a simple rule that is last in first out and all the operations we should perform so with the help of a top variable only with the help of top variables are how to insert the elements let's see



A stack can be easily implemented through the linked list. In stack Implementation, a stack contains a top pointer, which is "head" of the stack where pushing and popping items happens at the head of the list. first node have null in link field and second node link have first node address in link field and so on and last node address in "top" pointer.

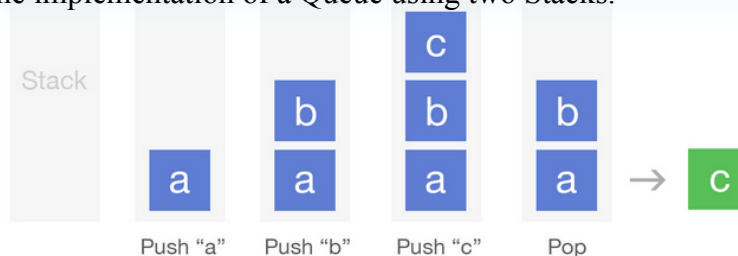
The main advantage of using linked list over an arrays is that it is possible to implements a stack that can shrink or grow as much as needed. In using array will put a restriction to the maximum

capacity of the array which can lead to stack overflow. Here each new node will be dynamically allocate, so overflow is not possible.

Stack Operations:

1. [push\(\)](#) : Insert the element into linked list nothing but which is the top node of Stack.
2. [pop\(\)](#) : Return top element from the Stack and move the top pointer to the second node of linked list or Stack.

Write down the implementation of a Queue using two Stacks.



Algorithm for queue using two stacks

For example: Suppose we push "a", "b", "c" to a stack. If we are trying to implement a queue and we call the **dequeue** method 3 times, we actually want the elements to come out in the order: "a", "b", "c", which is in the opposite order they would come out if we popped from the stack. So, basically, we need to access the elements in the reverse order that they exist in the stack. The following algorithm will implement a queue using two stacks.

(1) When calling the **enqueue** method, simply **push** the elements into the stack 1.

(2) If the **dequeue** method is called, push all the elements from stack 1 into stack 2, which reverses the order of the elements. Now pop from stack 2.

Activate W
Go to Settings

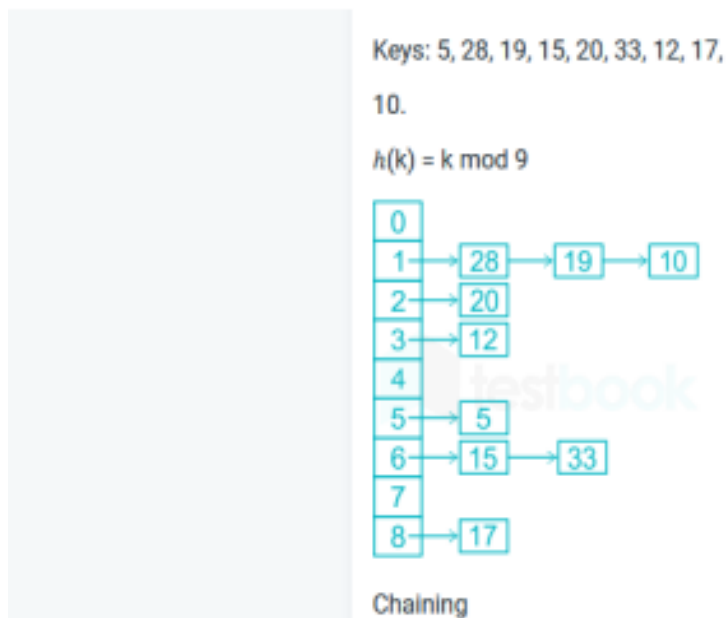
Solution steps

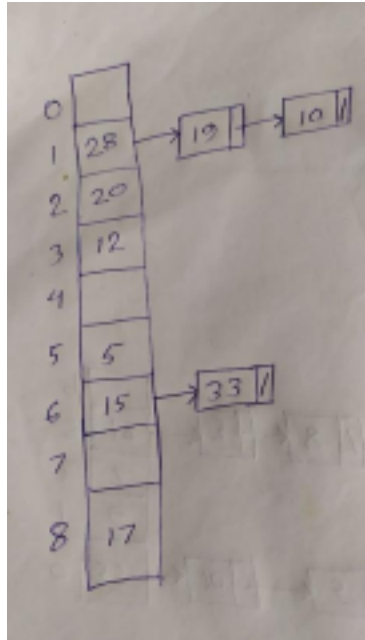
1. Take two stacks S1 and S2.
2. If S1 is empty, insert directly into S1.
3. If S1 is not empty, push all the elements from S1 to S2.
4. Push the element to be inserted into S1.
5. Push everything from S2 back to S1.

Chapter 11

2017-18 . 4.

i) Demonstrate what happens when we insert the keys 5, 28, 19, 15, 20, 33, 12, 17, 10 into a hash table with collisions resolved by chaining. Let the table have 9 slots, and let the hash function be $h(k) = k \bmod 9$.

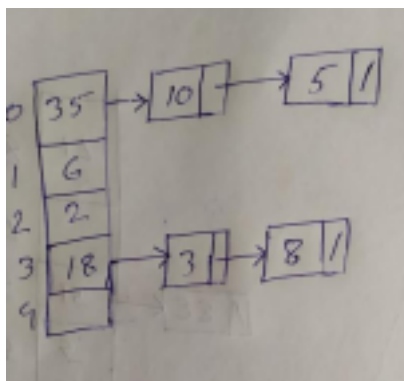




2016-17 .

4.

i) Take an initially empty hash table with five slots, with hash function $h(x) = x \% 5$ and with collisions resolved by chaining. Draw a sketch of what happens when inserting the following sequence of keys into it: 35, 2, 18, 6, 3, 10, 8, 5



ii) repeat part (a) but with the following changes; the hash table with ten slots, the function is $h(x) = x \% 10$ and collision resolved by linear probing.

0	10
1	
2	2
3	3
4	
5	35
6	6
7	5
8	18
9	8

Video link [8.1 Hashing Techniques to Resolve Collision| Separate Chaining and Linear Pro...](#)

Chapter 15

Same as Chapter 4

For a better understanding of TOH & LCS You can see the videos which I provide below:

AVAILABLE AT:

Onebyzero Edu - Organized Learning, Smooth Career
The Comprehensive Academic Study Platform for University Students in Bangladesh (www.onebyzeroedu.com)

For LCS : [LCS - Longest Common Subsequence - Algorithms \(Bangla Tutorial\)](#)

I Can't understand MCM (Matrix Chain Multiplication). Though it's not an important topic you should learn it!

I'm also unable to understand the LCS algorithm due to a lack of my time

Chapter 21 & 22 (V.V.I)

#নিচের প্রশ্ন সম্ভের সাথে টপিক মিলিয়ে ইউটিউবে ভিডিও দেখলেই বুঝা যাবে।

Topological sort using DFS :

[Topological sort using DFS | with starting time and finishing time | bangla](#)

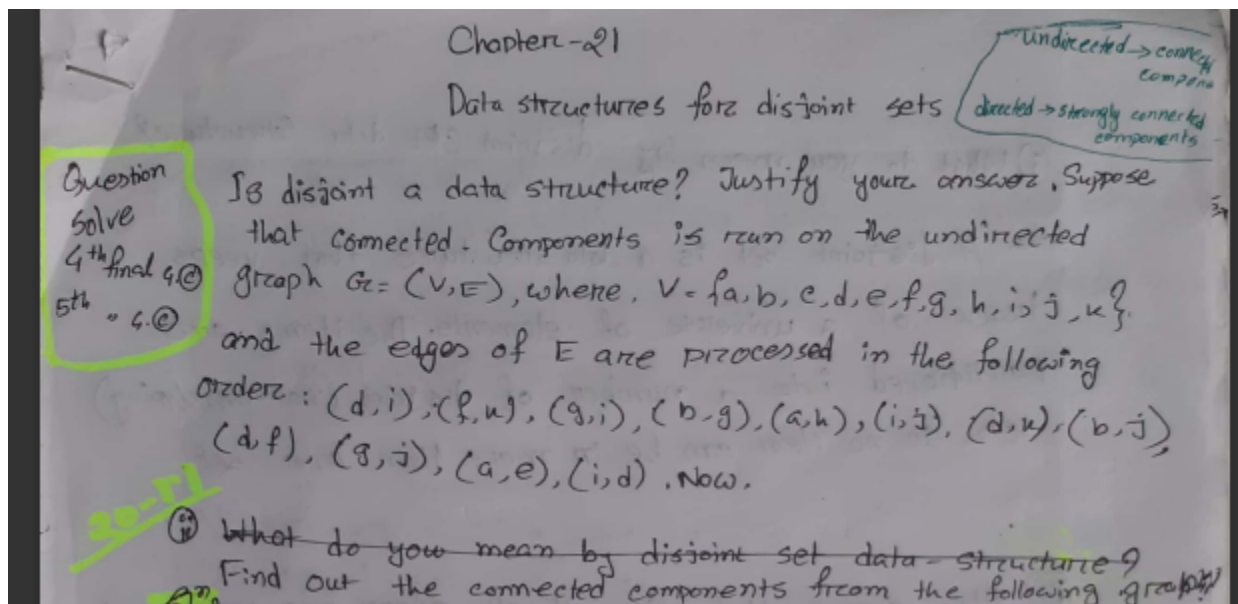
Strongly Connected Component :

[Connected and Strongly Connected Components in a Graph #Graph Series #24](#)

[Strongly Connected Component](#)

Types of edges :

[6.3 Types of Edges in DFS | Edge Classification | Data Structures and Algorithms](#)



51st:

the connected components from the following graph:

Edge Processed	Collection of disjoint set									
initial set	{a}	{b}	{c}	{d}	{e}	{f}	{g}	{h}	{i}	{j}
(d,i)	{a}	{b}	{c}	{d,i}	{e}	{f}	{g}	{h}	{i}	{j}
(f,u)	{a}	{b}	{c}	{d,i}	{e}	{f,u}	{g}	{h}	{i}	{j}
(g,i)	{a}	{b}	{c}	{d,i,g}	{e}	{f,u}	{g}	{h}	{i}	{j}
(b,g)	{a}	{b,d,i,g}	{c}	{e}	{f,u}	{g}	{h}	{i}	{j}	{k}
(a,h)	{a,h}	{b,d,i,g}	{c}	{e}	{f,u}	{g}	{h}	{i}	{j}	{k}
(i,j)	{a,h}	{b,d,i,g,j}	{c}	{e}	{f,u}	{g}	{h}	{i}	{j}	{k}
(d,u)	{a,h}	{b,d,i,g,j,f,u}	{c}	{e}						
(b,j)	{a,h}	{b,d,i,g,j,f,u}	{c}	{e}						
(d,f)	{a,h}	{b,d,i,g,j,f,u}	{c}	{e}						
(g,j)	{a,h}	{b,d,i,g,j,f,u}	{c}	{e}						
(a,e)	{a,h,e}	{b,d,i,g,j,f,u}	{c}							
(i,d)	{a,h,e}	{b,d,i,g,j,f,u}	{c}							

So, the connected components that we left with {a,h,e}, {b,d,i,g,j,f,u} and {c}

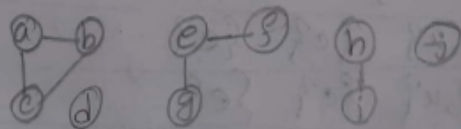
① What do you mean by disjoint set data structure?

⇒ A disjoint set is a data structure that keeps track of a universe of elements. The items are partitioned into a number of disjoint (non-overlapping) sets, so no item can be in more than one set.

② Also prove that after all edges are processed by Connected-Components, two vertices are in the same connected component if & only if they are in the same set.

⇒ If two vertices are not in the same set, meaning that no edge connecting the set of one vertex with the other. So two vertices are in the same connected component if and only if

they are in the same set.



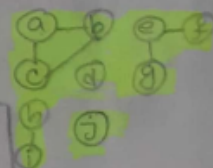
Here are 5 connected components, because $\{a, b, c, d\}$, $\{e, f\}$, $\{g\}$, $\{h\}$ and $\{i\}$ are 5 components. Here, if d was connected with b then there would be an edge that connected b and d and a, b, c are connected component because these vertices are connected by 3 edges.

During the execution of Connected-Components on the or undirected graph $G = (V, E)$ with k connected components, how many times is Find-set called? How many times is Union-set called? Express answer in terms of $|V|, |E|$ and k .

⇒ Find set is called twice on line 4, this is run once per edge in the graph, so, we have that find set is run $2|E|$ times. Since, we start with $|V|$ sets, at the end only have k , and each call to UNION reduces the number of sets by one, we have that we have to make $|V| - k$ calls to UNION.

Like find
4. (2.11)

Disjoint set collection using graph



Edge processed	collection of disjoint sets
initial sets	$\{a\} \{b\} \{c\} \{d\} \{e\} \{f\} \{g\} \{h\} \{i\} \{j\}$
(a,b)	$\{a,b\} \{c\} \{d\} \{e\} \{f\} \{g\} \{h\} \{i\} \{j\}$
(a,c)	$\{a,b,c\} \{d\} \{e\} \{f\} \{g\} \{h\} \{i\} \{j\}$
(b,c)	$\{a,b,c\} \{d\} \{e\} \{f\} \{g\} \{h\} \{i\} \{j\}$
(e,f)	$\{a,b,c\} \{d\} \{e,f\} \{g\} \{h\} \{i\} \{j\}$
(e,g)	$\{a,b,c\} \{d\} \{e,f,g\} \{h\} \{i\} \{j\}$
(h,i)	$\{a,b,c\} \{d\} \{e,f,g\} \{h,i\} \{j\}$

So the connected components that we left, $\{a,b,c\} \{d\} \{e,f,g\} \{h,i\} \{j\}$

5th Final DS

4. (d) What is the running time of BFS if we represent its input graph by an adjacency matrix and modify the algorithm to handle this form of input?

⇒ If we use an adjacency matrix for each vertex u and we dequeue we'll have to examine all vertices v to decide whether or not v is adjacent to u . This makes the for-loop of line 12 $O(V)$. In a connected graph we enqueue every vertex of the graph, so the worst case runtime becomes $O(V^2)$.

Chapter - 22

Representation of graph

(i) Adjacency list

(ii) " matrix

using in sparse graph is

graph (few number of edges in vertices)

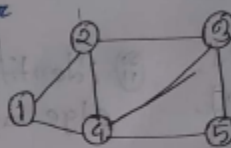
Dense graph (every node is connected to each other)

Adjacency matrix

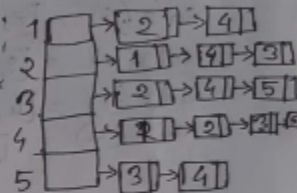
(Undirected graph)

	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	1	0
3	0	1	0	0	1
4	1	1	1	0	1
5	0	0	1	1	0

vertex number
→ $n \times n$ size
undirected graph
অন্য node কে সাধারণ
অন্য node
directed graph
1 → 2 হল ডায়াল

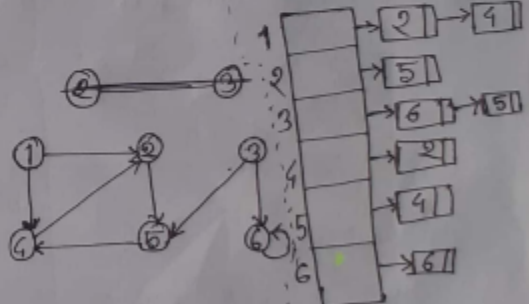


Adjacency list



Directed graph

	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	0	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1



Space complexity :- $O(n^2)$

$O(n+2e)$

Q. of 5th final_DS \rightarrow 5B
5th final_Algo \rightarrow 7C

Consider the following adjacent list for directed graph.

$\text{adj}(y) = [x]$

$\text{adj}(x) = [z]$

$\text{adj}(z) = [y, w]$

$\text{adj}(w) = [x]$

$\text{adj}(s) = [z, u]$

$\text{adj}(v) = [s, w]$

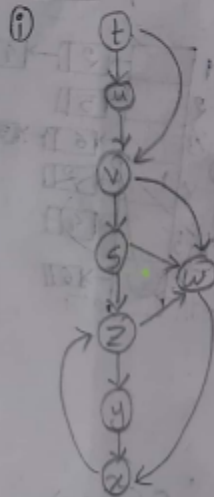
$\text{adj}(t) = [u, v]$

$\text{adj}(u) = [v]$

(i) Construct a graph from the above information and sort the node using topological sort.

(ii) Identify the tree edge, back edge, cross edge and forward edge for the graph.

Solution :-



Topological sort

Topological sort can not be done

because x, z, w and x, y, z are connected in cycle. As topological sort is performed in Directed acyclic graph that's why for this graph topological sorting is impossible.

TYPES of edges in DFS

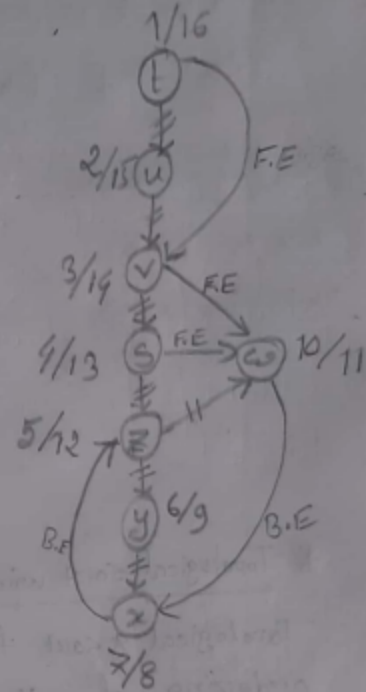
⑩ Identifying the tree edge, back edge and cross edge

Tree edges :- $(t,u), (u,v), (v,s), (s,z), (z,y), (y,x), (z,w)$

Forward edges :- $(t,v), (v,w), (s,w)$

Back edges :- $(w,x), (x,z)$

Cross edges :- No cross edge



Define types of edges

Define types of edges

Tree edges :- Are those edges which are members of DFS travels.

Forward edges :- $e_i(x,y)$ where y appears after x and there is a path from x to y .

Back edges :- $e_i(x,y)$ where y appears before x and there is a path from y to x .

Cross edge :- $e_i(x,y)$ where there is no path from y to x .

(16-17) Final Algo:- Classify and examine edges

Tree:- (a,b), (b,g), (f,c), (f,g),
(f,e), (c,d), (d,h), (h,d)

Forward:- (g,h)
Back:- (e,b)
Cross:- (c,g)

Topological Sort \rightarrow graph কে অব্যাহত directed graph
 \rightarrow এবং কোনো cycle থাকে না

* Topological Sort using DFS:-

Topological sort for directed acyclic graph (DAG) is a linear ordering of vertices such that for every directed edge uv , vertex u comes before v in the ordering.

5th Final DS

5. (a)

or,

$\therefore 1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 4$

$\therefore 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$

4th final_DS

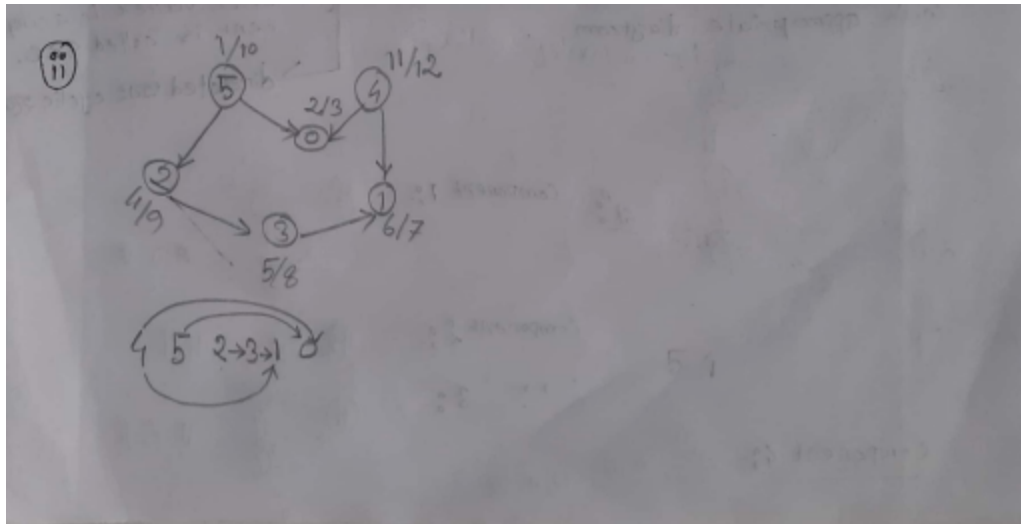
Q. ⑥ A Topological sort or topological ordering of a directed graph is a linear ordering of its vertices such that for every directed edge u and v from vertex u to vertex v , u comes before v in the ordering.

5th final
Algo

① Why must a graph with a topological sort be acyclic?
3. ② Topological order is possible iff the graph has no cycles. Justify?

sm:

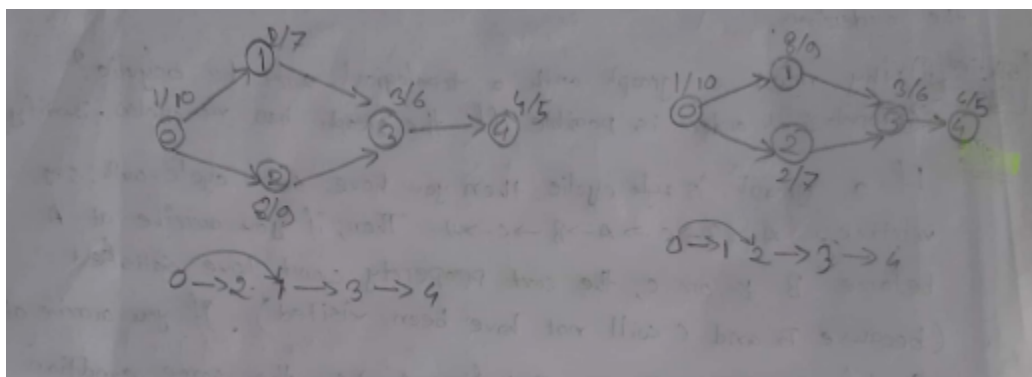
If a graph is cyclic, then you have some cycle with, say, vertices $A \rightarrow B \rightarrow C \rightarrow A \rightarrow B \rightarrow C \rightarrow A \dots$. Then, if you arrive at A before B & or C , the sort property can't be satisfied (because B and C will not have been visited). If you arrive at C before A and B or B before A and C the same condition is applicable. With a cyclic ^{directed} graph or which point in the cycle first arrived is no matter, there is no possible way to satisfy the topological sort.



(14-15) Final algo:-

7. (b) A topological sort or topological ordering of a directed graph is a linear ordering of its vertices such that for every edge u and v from vertex u to vertex v , u comes before v in ordering.

(i) find out the topological sort

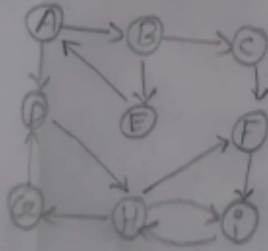


6th Final DS

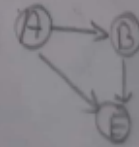
Strongly connected component

① Find out the cycles from the graph using strongly connected component with appropriate diagram

→ if we can reach from every vertex to every other vertex in a component is called SCC.
→ directed graph cyclic graph

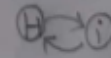


SCC:- Component 1:-



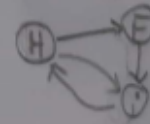
A, B, E

Component 2:-



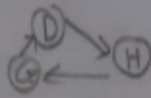
H, I

Component 3:-



H, F, I

Component 4:-

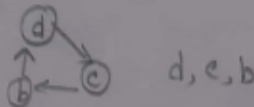


D, H, G

5th final algo:-

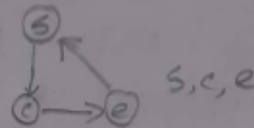
Q. b. ii) Find out strongly connected component

Component-1:-

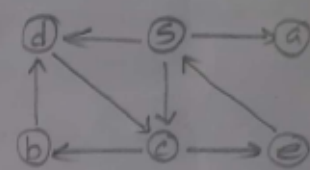


d, c, b

Component-2:-



s, c, e



5th final DS → 5 ©
6th final DS → 6 © ii.

Professorz Bacon claims that the algorithm for Strongly connected components would be simpler if it used the original (instead of the transpose) graph in the second depth-first search and scanned the vertices in order of increasing finishing times. Does the simpler algorithm always produce correct result?

Chapter 23

প্রশ্ন সলভ না বুঝাটাই স্বাভাবিক। তাই মেইন বই বা এই টপিকের ভিডিও দেখে প্রশ্ন সলভ করলেই ইজিলি বুঝতে পারার কথা।

5th batch (2nd Sem)

2nd mid

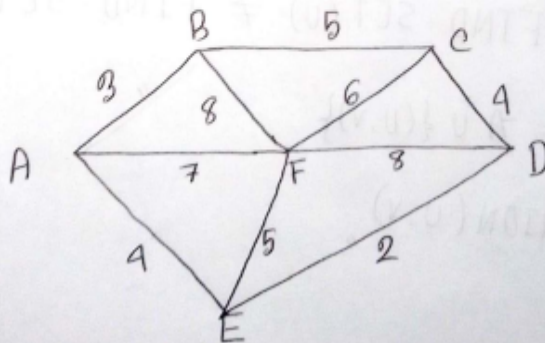
Kruskal's algorithm:

1. Graph থেকে loop, multiple edge delete করতে হবে।
2. নতুন graph কোন cycle ছাড়া যাবে না।
3. নতুন graph vertex হবে প্রস্তুত দেওয়া graph এর vertex অক্ষান এবং edge হবে $(\text{vertex}-1)$ ।

Question:

Question:

1) (d) Find the minimum weight for the minimum spanning tree and highlighted the selected nodes and edges.

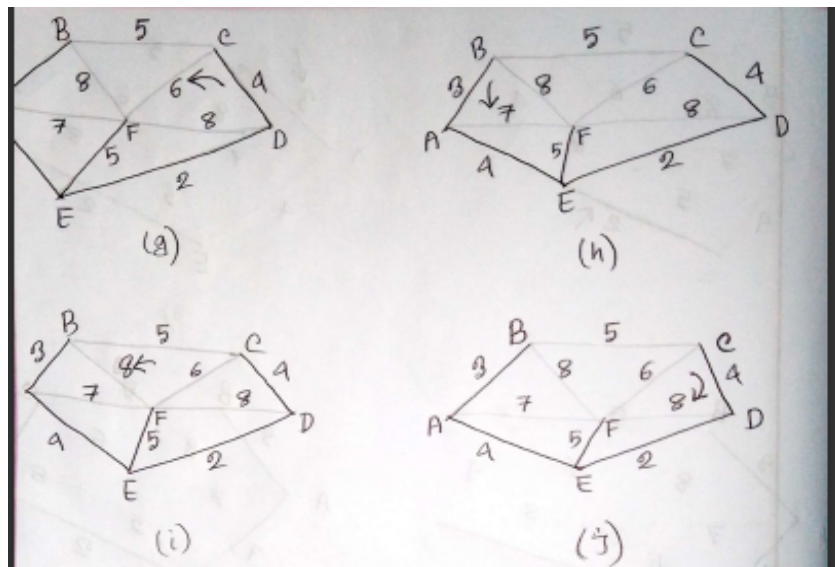
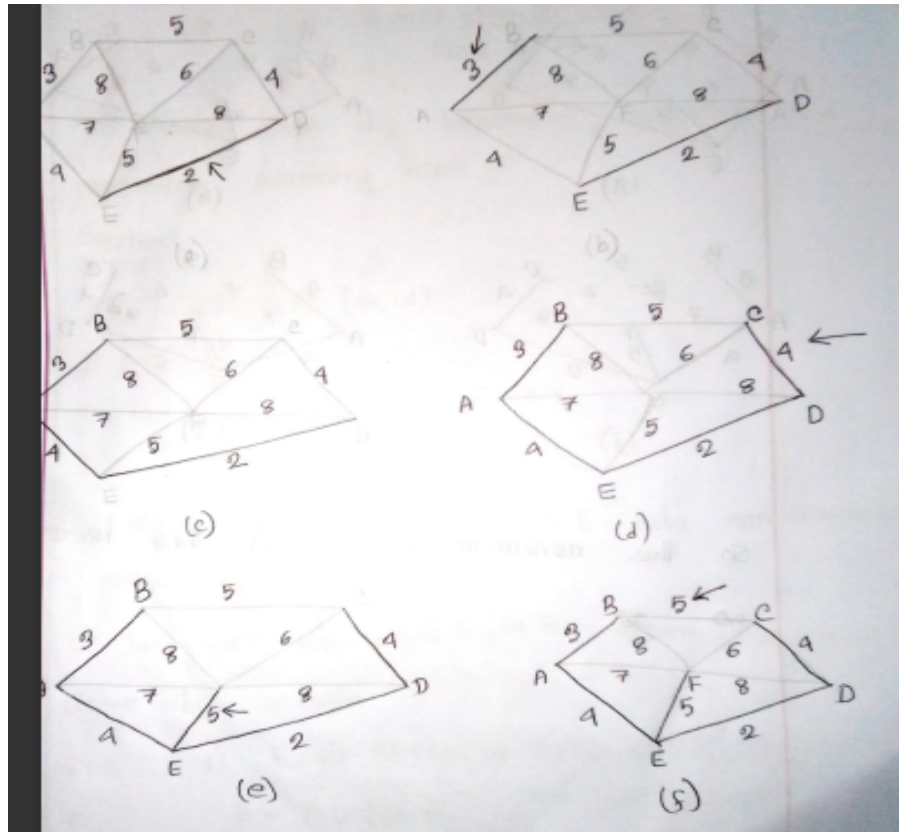


Solⁿ Solution:

To find the minimum weight of minimum spanning tree, we use kruskal algorithm's.

MST- kruskal (G, w)

1. $A = \emptyset$
2. for each vertex $v \in G.V$
3. Make-SET(v)
4. sort the edge of $G.E$ into non-decreasing order by weight w
5. for each edge $(u, v) \in G.E$ taken non-decreasing order by weight
6. if $\text{FIND-SET}(u) \neq \text{FIND-SET}(v)$
7. $A = A \cup \{(u, v)\}$
8. UNION(u, v)
9. return A



So the minimum weight of this tree is 18.

Q. Write down the Kruskal's algorithm for minimum spanning tree.

Solution:

MST- KRUSKAL (G, w)

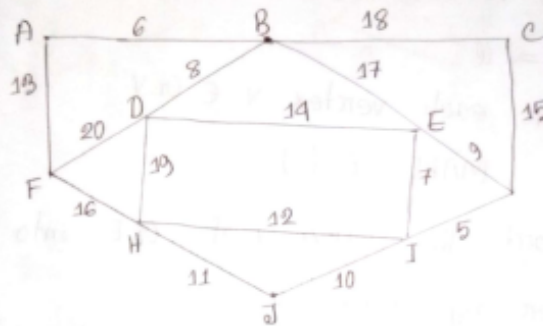
1. $A = \emptyset$
2. Sort each vertex $v \in G.V$
3. MAKE-SET(v)
4. sort the edges e of $G.E$ into non-decreasing order by weight w
5. for each edge $(u, v) \in G.E$ taken non-decreasing order by weight.
 6. if FIND-SET(u) \neq FIND-SET(v)
 7. $A = A \cup \{(u, v)\}$
 8. UNION(u, v)
 9. return A

5th batch (4th sem)

Mid 2 + Final

71

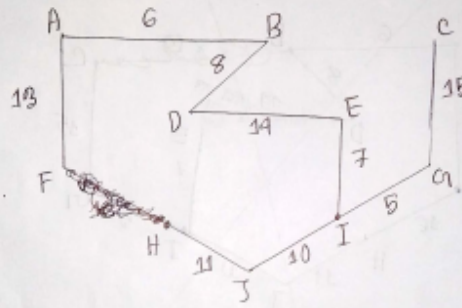
7b) The following network has 10 vertices A, B, ..., J. The numbers represent the distances in miles between pairs of vertices.



- Use Kruskal's algorithm to find the minimum spanning tree for the network and also draw the spanning tree.
- Find out the weight of the spanning tree.

Solution:

i. প্রথমে kruskal's algorithm টা নিখাত হবে
যা উপরে ৫th final (2nd sem) question এর
solution আছে.



ii. The weight of the spanning tree
is 91.

4th batch (3rd sem)

Final

Q1) A minimum spanning tree (MST) or minimum weight spanning tree is a subset of the edges of a connected edge-weighted (un)directed graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight.

i) Describe in detail both Prim's and Kruskal's algorithms for finding a minimum cost spanning tree of an undirected graph with edges labelled with positive costs and explain why they are correct.

ii) Compare the relative merits of the two algorithms.

Solutions

(i)

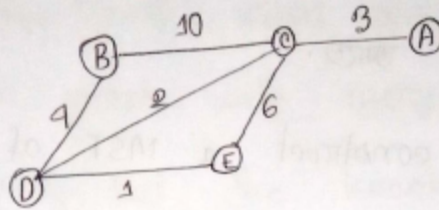
Prim's algorithm starts with the single node and explore all the adjacent nodes with all the connecting edges at every step. The edges with the minimal weights causing no cycles in the graph got selected.

MST-PRIM(G, w, π)

1. for each $v \in G.V$
2. $v.key = \infty$
3. $v.\pi = NIL$
4. $\pi.key = 0$
5. $Q = G.V$
6. while $Q \neq \emptyset$
7. $u = \text{EXTRACT-MIN}(Q)$
8. for each $v \in G.Adj[u]$
9. if $v \in Q$ and $w(u, v) < v.key$
10. ~~v~~ $v.\pi = u$

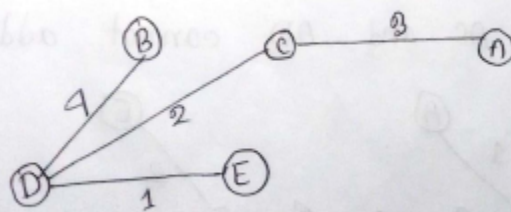
11. $v.key = w(u, v)$

For example construct MST for the following graph.



choose a starting vertex B. choose the edge with the minimum weight among all i.e. BD and Add to the MST. Add the adjacent vertices of D i.e. C and E. DE and CD edge are add to MST because they are minimum weighted edge. Then choose CA. We cannot choose CE because it would cause cycle in the graph.

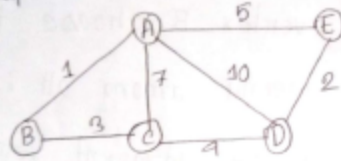
it would cause cycle in the graph.



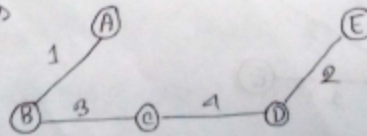
Kruskal's algorithm's:

এখানে আমরা Kruskal's algorithm
টা বিবর্তে হবে যা উপরে ৫th batch (৫th sem)
question দেওয়া আছে।

For example construct a MST of the following
graph.



1st add AB to the MST because this edge
is minimum. Then add DE and BC and also
CD. The next step is to add AE but we
cannot add it as it will cause a cycle.
Similarly AC and AD cannot add. The final
MST is



(ii)

Prim's algorithm is significantly faster in the limit when we have got a really dense graph with many more edges than vertices. But the kruskal performs better in typical situations (sparse graphs) because it uses simpler data structures. The kruskal algorithm is better to use regarding the easier implementation and the best control over the resulting MST.

3rd batch (4th sem)

Final

Q/ a. b \rightarrow same question + answer 5th
batch (4th sem) final + mid 2 eq. a. b.

Q) What is prim's algorithm? Identify its
difference with kruskal's algorithm.

Solution:

Prim's algorithm:

Prim's algorithm is used to
find the minimum spanning tree from a
graph. Prim's algorithm finds the subset of
edges that includes every vertex of the
graph such that the sum of the weights
of the edges can be minimized.

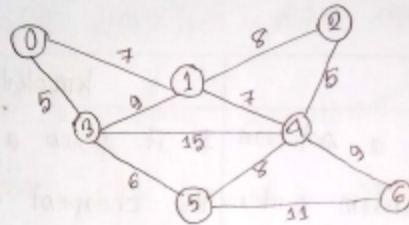
একসঙ্গে prim's algorithm টা বিচারে হবে. সব answer
উপরে 4th batch final question হওয়া আছে.

The difference between prim's and kruskal's
algorithm:

Prim's	Kruskal's
1. It grows a solution from a random vertex by adding the cheapest vertex to the existing tree.	1. It grows a solution from the cheapest edge by adding the next cheapest edge to the existing tree.
2. It is faster for dense graphs.	2. It is faster for sparse graph.
3. Time complexity is $O(V^2)$	3. Time complexity is $O(E \log V)$
4. It traverse one node more than one time to get minimum distance.	4. It traverse one node only once.

6th batch (and some final)

Q16) Define safe edge. Write down the execution of Prim's algorithm on the following graph.



Solution:

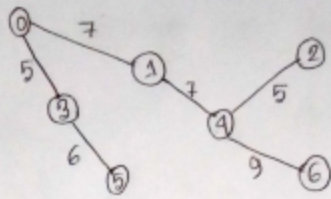
Safe edge:

An edge that may be added to A without violating the invariant that A is a subset of some minimum spanning tree.

MST-PRIM(G, w, π)

1. for each $u \in G.V$
2. $u.key = \infty$
3. $u.\pi = NIL$
4. $\pi.key = 0$
5. $Q = G.V$
6. while $Q \neq \emptyset$
7. $u = \text{EXTRACT-MIN}(Q)$
8. for each $v \in G.Adj(u)$
9. if $v \in Q$ and $w(u, v) < v.key$
10. $v.\pi = u$
11. $v.key = w(u, v)$

Apply the prim's algorithm for the follow graph?

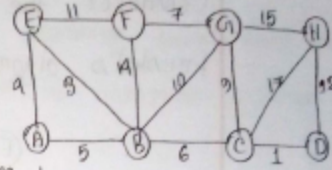


Choose a starting vertex 0. Choose the edge with the minimum weight among all 0 i.e. 03 and add to the MST. Add the adjacency vertices of 3 i.e. 1, 5. Add 35 to the MST and this edge is minimum weight. Then choose 01 and 14 and add them to the MST. Add the adjacency vertices of 4 add 42 to the MST. The next step is to add 21 but we cannot add it to the MST because it will cause cycle. ~~Similarly~~ Similarly we cannot add 45 and 13. Now add 46 to the MST. This is the final minimum spanning tree.

4th batch (2nd semr final)

2(b) Consider weighted graph above.

i. Run prim's algorithm starting from vertex A. Write the edge in the order which they are added in the minimum spanning tree.

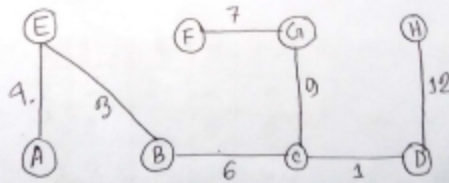


ii. Run kruskal's algorithm starting from vertex A. Write the edge in the order which they are added in the minimum spanning tree.

Solution:

i. প্রথমে prim's algorithm টা নিইতে হবে। যা

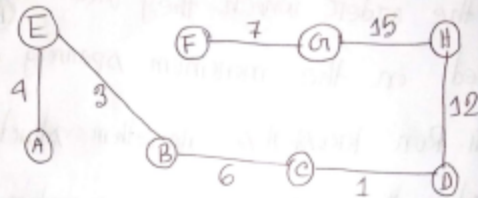
5th batch 2nd semr final question এর answer দেওয়া আছে



So the number of vertex is 8 and edge is 7.

ii. প্রথমে kruskal's algorithm টা বিস্মৃত হ

Construct the minimum spanning tree using kruskal's algorithm



This is final MST. Here the total vertex is 8 and the number of edge is 7.

2(a) Show that the minimum spanning tree is unique but the second-best minimum spanning tree need not be unique and give an algorithm to compute second best minimum spanning tree.

Solution:

To see that the second best minimum spanning tree is not be unique, we consider the following example graph on four vertices. Suppose the vertices are $\{a, b, c, d\}$ and the edge weights are as follows:

	a	b	c	d
a	-	1	4	3
b	1	-	5	2
c	4	5	-	6
d	3	2	6	-

Then the minimum spanning tree has weight 7, but

there are two spanning trees of second best weight 8.

We provide here an algorithm that takes time $O(V^2)$ and leave open if there exists a linear time solution, that is a $O(E+V)$ time solution. First we find a minimum spanning tree in time $O(E+V \lg(V))$, which is in $O(V^2)$. Then using the algorithm, we find the double array \max . Then we take a running minimum

over all pairs of vertices u, v of the value of $w(u, v) - \max[u, v]$. If there is no edge between u and v , we think of the weight being infinite. Then sort the pair that resulted in the minimum value of this difference, we add in that edge and remove from the minimum spanning tree

an edge that is \neq in the path from u to v that has weight $\max[u, v]$

Chapter 32

Naive string matching:

▶ Naive String Matching Algorithm in Hindi with Solved Examples - Algorithm Design Analy...

Finite automata: ai video tar 24 min theke dekbi

▶ DAA96: Algorithm for String Matching with Finite Automata | Finite Automata String Matching

Aita dekbi table ta kivabe create kre tar jonno

▶ String Pattern Matching with Finite Automata||Example-1||Design and analysis of algo...

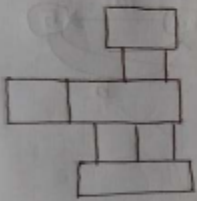
Compute Pi function: ▶ kmp algorithm prefix function shortcut method | kmp algorithm in daa

String Matching

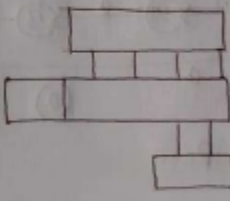
5th batch \rightarrow 2nd semi (Mid - 2 + final), 4th semi (Mid + final) +
3rd batch - final

Question: Suppose that x, y and z are strings such that $x \sqsubset z$. Prove that if $|x| \leq |y|$ then $x \sqsubset y$ if $|x| \geq |y|$ then $y \sqsubset x$ and if $|x| = |y|$ then $x = y$

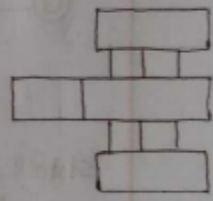
Solution:



(a)



(b)



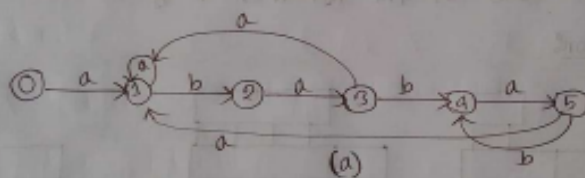
(c)

We suppose that $x \sqsubset z$ and $y \sqsubset z$.

(a) if $|x| \leq |y|$ then $x \sqsubset y$ (b) if $|x| \geq |y|$ then $y \sqsubset x$
(c) if $|x| = |y|$ then $x = y$

(b) For the following text $T = abababababa$ and pattern $P = ababa$. Construct the string matching automaton and find out the number of hits does the following string matching algorithms encounter to the text.

Solution:



State	Input		P
	a	b	
0	1	0	a
1	1	2	b
2	3	0	a
3	1	4	b
4	5	0	a
5	1	4	

(b)

$$m = P.length$$

FINITE - AUTOMATON - MATCHER (T, δ , m)

1. $n = T.length$
2. $q = 0$
3. for $i = 1$ to n
4. $q = \delta(q, T[i])$
5. if $q = m$
6. print "Pattern occurs with shift $i-m$ "

Here $m = 5$

$i \rightarrow$	1	2	3	4	5	6	7	8	9	10	11
$T[i]$	a	b	a	b	a	b	a	b	a	b	a
state $T(i)$	1	2	3	4	5	4	5	4	5	4	5

The sequence of states for T is 0, 1, 2, 3, 4, 5, 4, 5, 4, 5, 4, 5 and so finds 3 occurrences of the pattern at $s = 1$, $s = 3$ and $s = 7$.

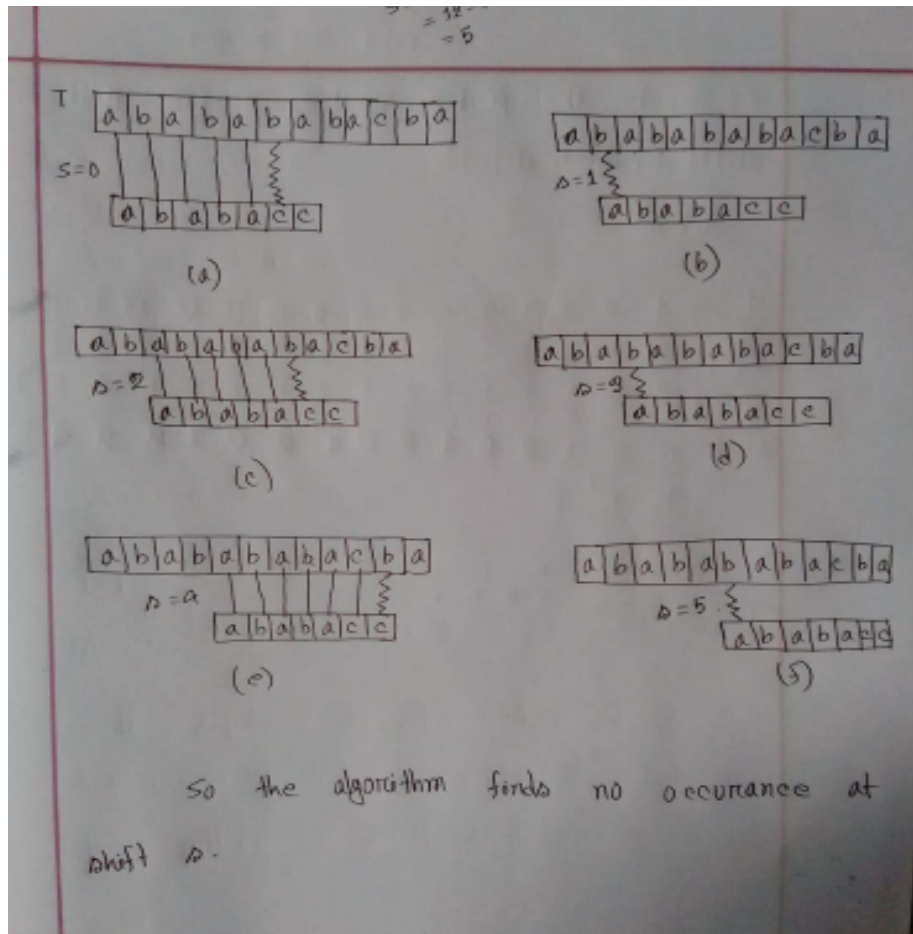
Q. a) Write down the number of hits does the following string matching algorithms encounter in the text $T = ababababacba$, pattern $P = ababacc$ and also write down their procedure.

- i. Naive String Matching
- ii. Finite State Automata

Solution:

① NAIVE STRING MATCHER(T, P)

1. $n = T.length$
2. $m = P.length$
3. For $s = 0$ to $n - m$
4. if $P[1 \dots m] = T[s+1 \dots s+m]$
5. print "Pattern occurs with shift s "



4th batch (3rd sem)

* Compute the prefix function π for the pattern ababbababbababbabbb.

Solution:

$i \rightarrow$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
$T[i]$	a	b	a	b	b	a	b	b	a	b	b	a	b	a	b	b	a	b	b
$\pi[i]$	0	0	1	2	0	1	2	0	1	2	0	1	2	3	4	5	6	7	8