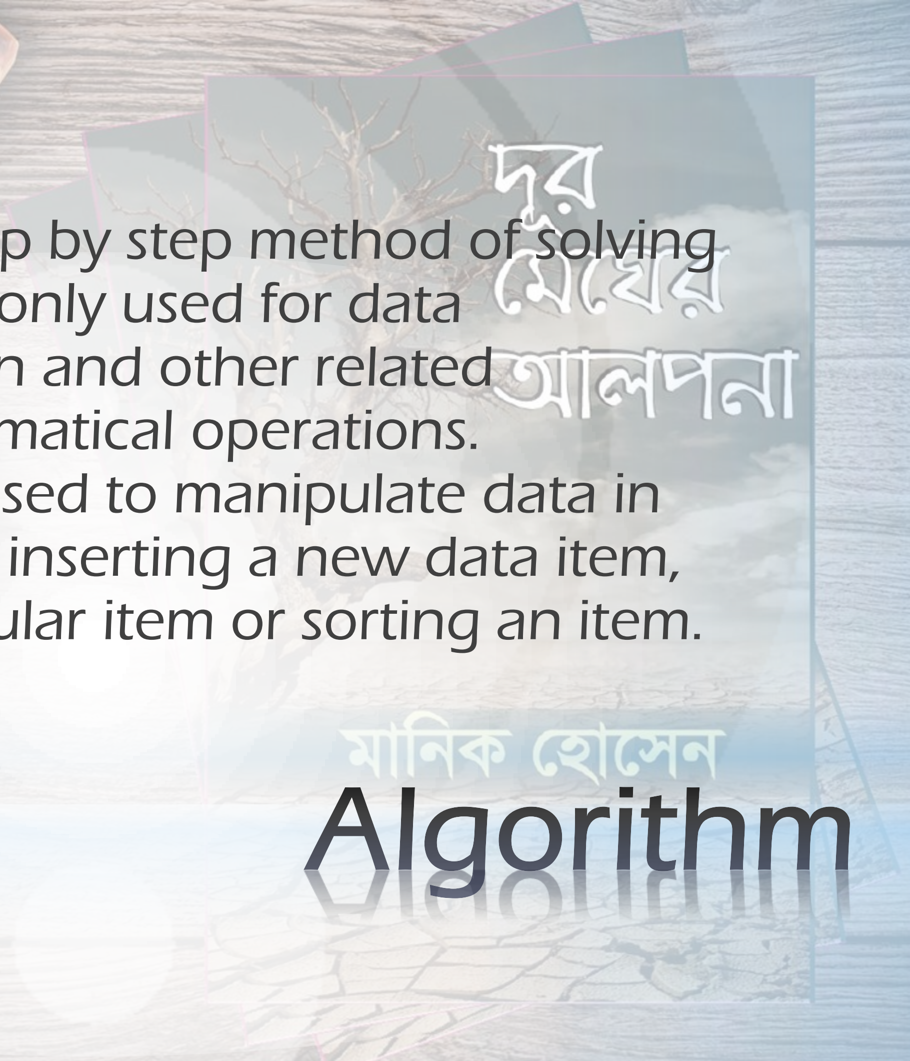
Analysis of Algorithm

Supported By “দূর মেঘের আলপনা”

Adapted By Manik Hosen

Algorithm

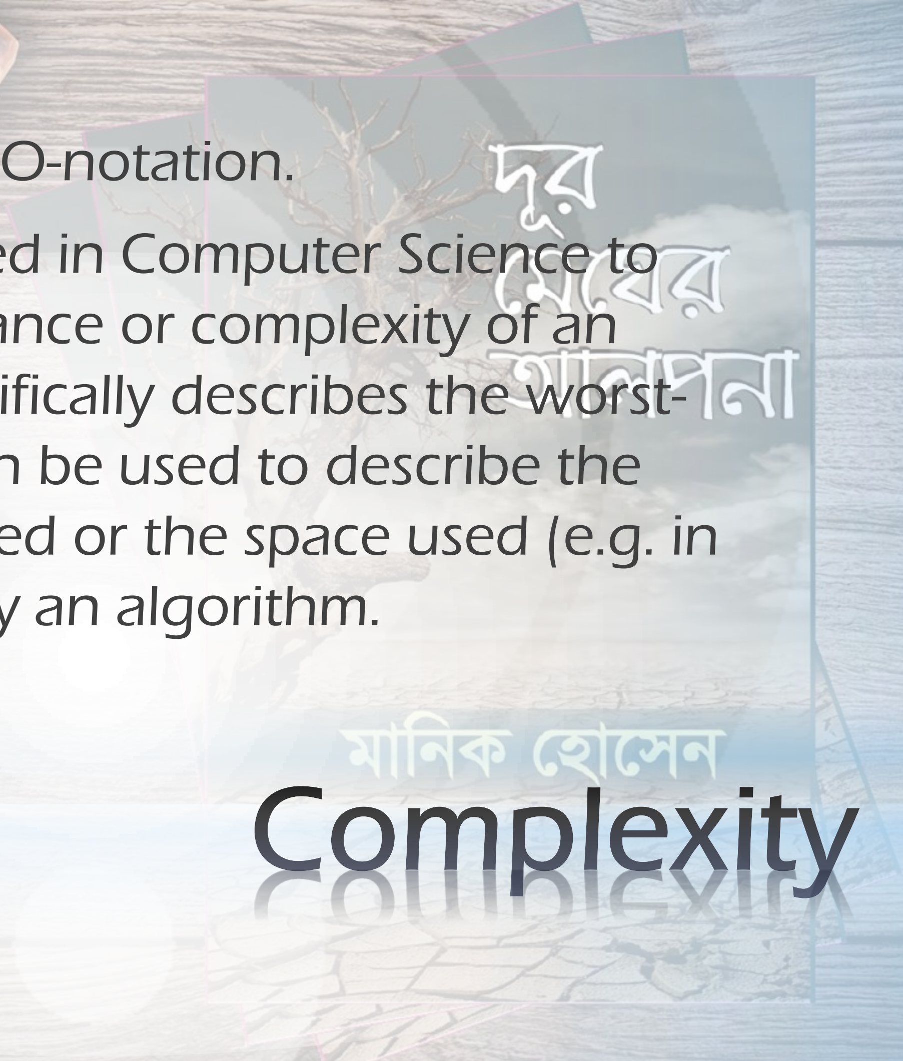
- Define algorithm.
- An algorithm is a step by step method of solving a problem. It is commonly used for data processing, calculation and other related computer and mathematical operations. An algorithm is also used to manipulate data in various ways, such as inserting a new data item, searching for a particular item or sorting an item.



Complexity

❑ Explain why we use O-notation.

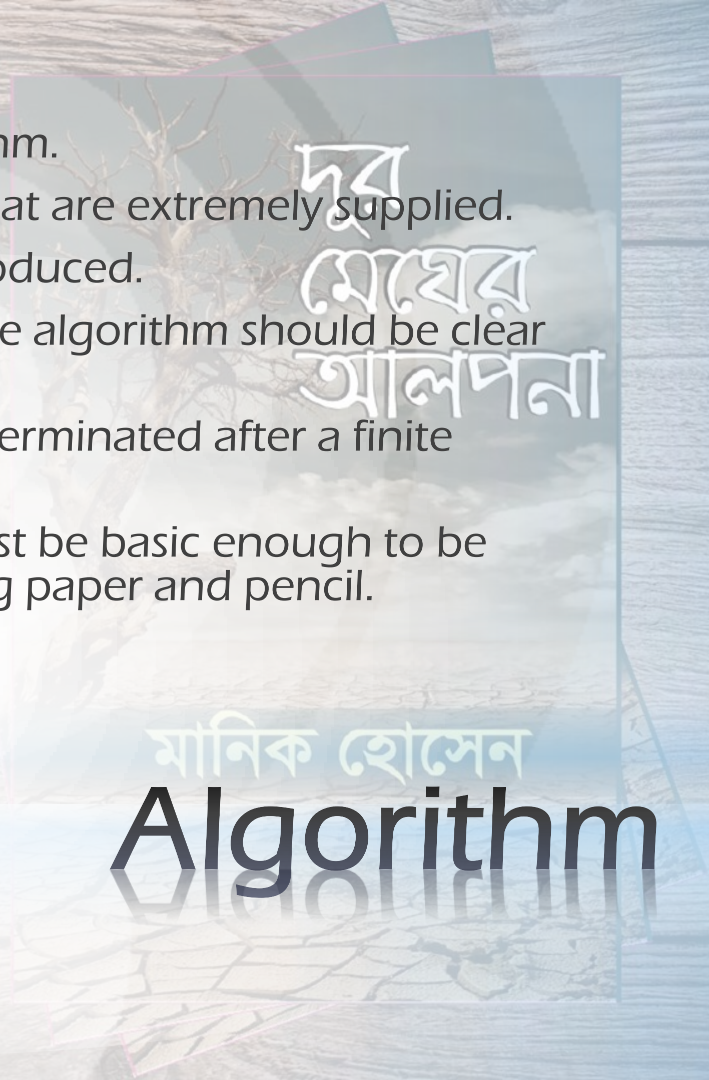
❑ Big O notation is used in Computer Science to describe the performance or complexity of an algorithm. Big O specifically describes the worst-case scenario, and can be used to describe the execution time required or the space used (e.g. in memory or on disk) by an algorithm.



Algorithm

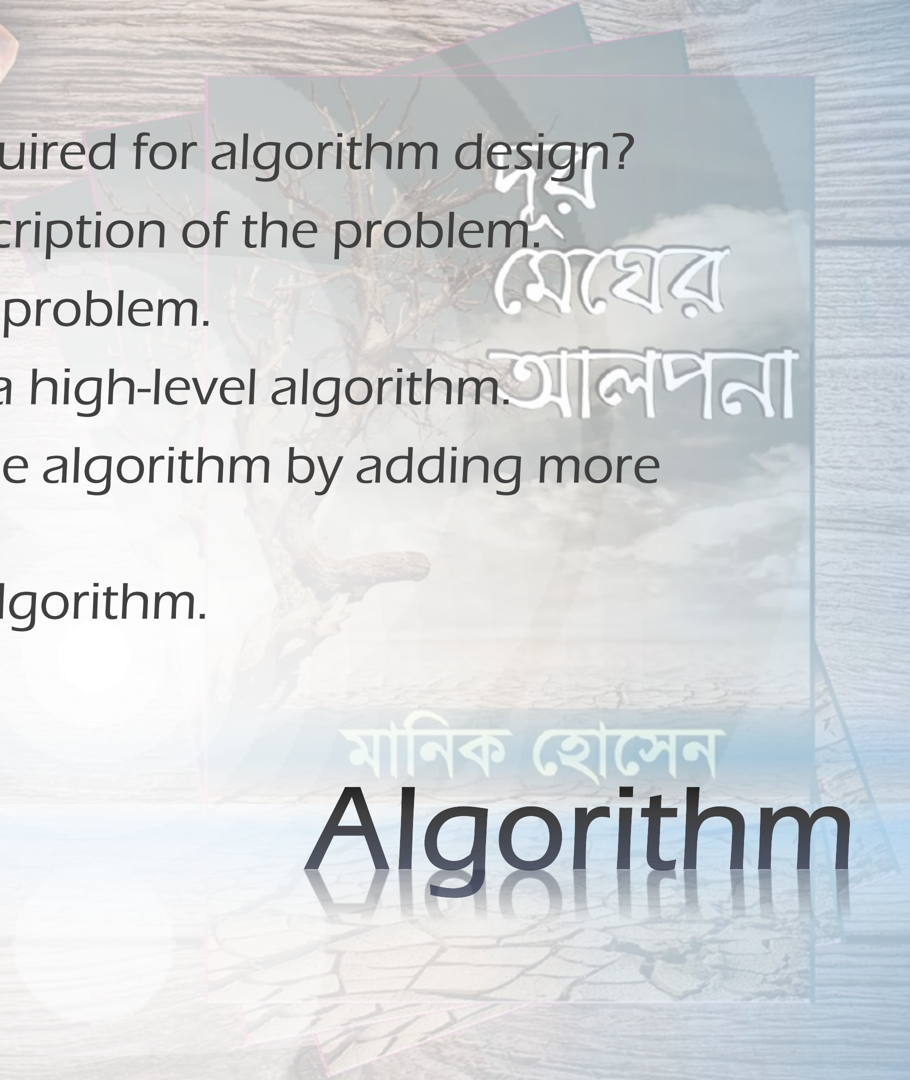
❑Write down the properties of algorithm.

1. **Input**: There are more quantities that are extremely supplied.
2. **Output**: At least one quantity is produced.
3. **Definiteness**: Each instruction of the algorithm should be clear and unambiguous.
4. **Finiteness**: The process should be terminated after a finite number of steps.
5. **Effectiveness:** Every instruction must be basic enough to be carried out theoretically or by using paper and pencil.



❑ What are the steps required for algorithm design?

1. Design: Obtain a description of the problem.
2. Analyze: Analyze the problem.
3. Implement: Develop a high-level algorithm.
4. Experiment: Refine the algorithm by adding more detail.
5. Review: Review the algorithm.



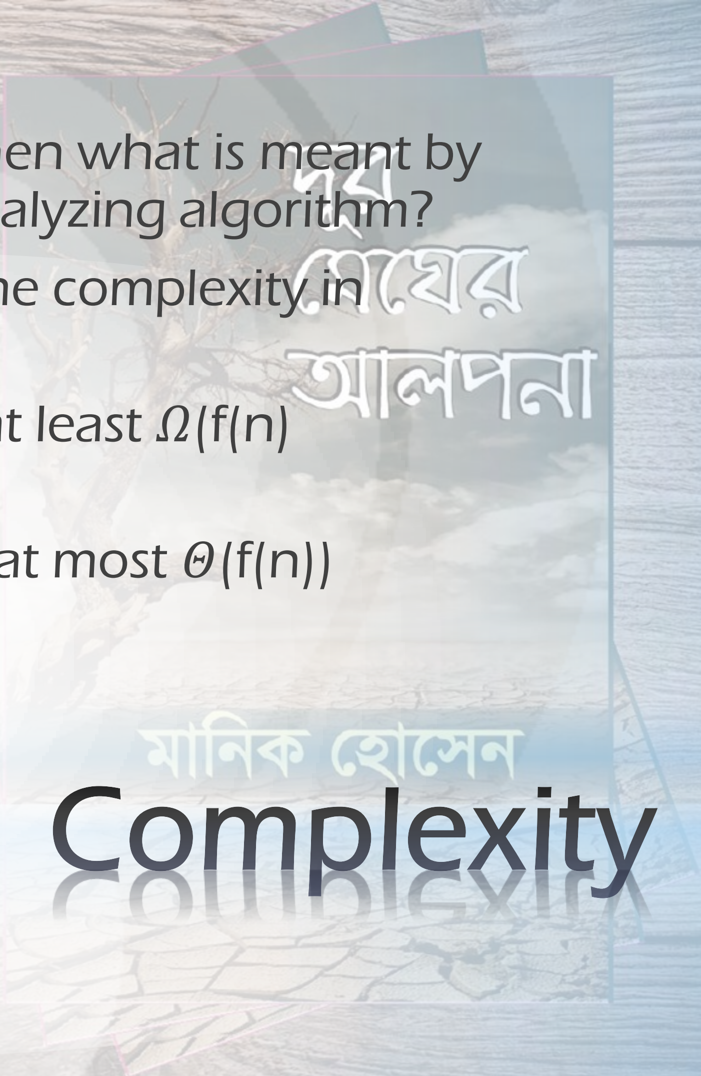
Algorithm

Complexity

- ❑What is best case, worst case and average case complexity?
- ❑Worst Case Complexity: The worst-case complexity usually denoted by Θ, measures the resources (e.g. running time, memory) an algorithm requires in the worst-case.
- ❑Best Case Complexity: The best-case complexity usually denoted by Ω, measures the resources (e.g. running time, memory) an algorithm requires in the worst-case.
- ❑Average Case Complexity: The average-case complexity usually denoted by O, measures the resources (e.g. running time, memory) an algorithm requires in the worst-case.

Complexity

- If f is a function of data size n then what is meant by O(f(n)), Ω(f(n)) and Θ(f(n)) for analyzing algorithm?
- O(f(n)) means the function's time complexity in average case.
- Ω(f(n) means the function has at least Ω(f(n) complexity, that is best case.
- Θ(f(n)) means the function has at most Θ(f(n)) complexity, that is worst case.



- What is the complexity of Quick sort, Merge sort, Heap sort, Bubble sort, Insertion sort, Selection sort, Counting sort and Binary search?

Algorithm	Data structure	Time Complexity		
		Best Case	Average Case	Worst Case
Quick sort	Array	O(n log(n))	O(n log(n))	O(n^2)
Merge sort	Array	O(n log(n))	O(n log(n))	O(n log(n))
Heap sort	Array	O(n)	O(n log(n))	O(n log(n))
Bubble sort	Array	O(n)	O(n^2)	O(n^2)
Insertion sort	Array	O(n)	O(n^2)	O(n^2)
Selection sort	Array	O(n^2)	O(n^2)	O(n^2)
Counting sort	Array	O(n+k)	O(n+k)	O(n+k)
Binary Search	Array	O(1)	O(log n)	O(log n)

Complexity

Insertion Sort

Show worst case running time of insertion sort in terms of cost and times.

Algorithm	Cost	Times
for j←2 to n	c1	n
do key←A[j]	c2	n-1
Insert A[j] into the sorted sequence A[1 . . j -1]	0	n-1
i←j-1	c4	n-1
while i>0 and A[i]>key	c5	$\sum_{j=2}^{n} t_j$
do A[i+]←A[i]	c6	$\sum_{j=2}^{n} (t_j - 1)$
i←i-1	c7	$\sum_{j=2}^{n} (t_j - 1)$
A[i+1]←key	c8	n-1

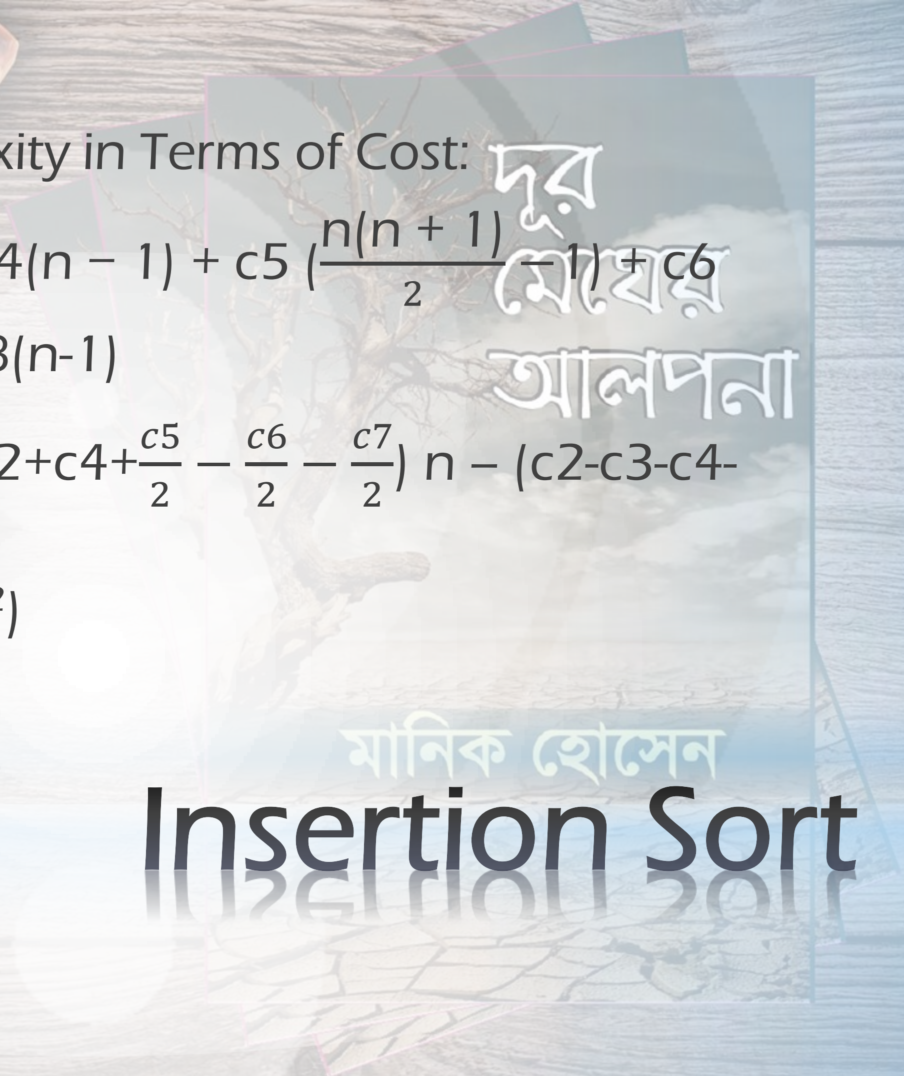
Insertion Sort

Worst Case Time Complexity in Terms of Cost:

$$C(n) = c1n + c2(n-1) + c4(n-1) + c5\left(\frac{n(n+1)}{2} - 1\right) + c6\,\frac{n(n-1)}{2} + c9\,\frac{n(n-1)}{2} + c8(n\text{-}1)$$

$$C(n) = \frac{c5+c6+c7}{2}\,n^2 + \left(c1+c2+c4+\frac{c5}{2} - \frac{c6}{2} - \frac{c7}{2}\right) n - (c2\text{-}c3\text{-}c4\text{-}c8)$$

$$C(n) = an^2 + bn + c = \Theta(n^2)$$



Worst Case Time Complexity in Terms of Times:

$$T(n) = n + (n - 1) + (n - 1) + (\frac{n(n + 1)}{2} - 1) + \frac{n(n - 1)}{2}$$
$$+ \frac{n(n - 1)}{2} + (n-1)$$

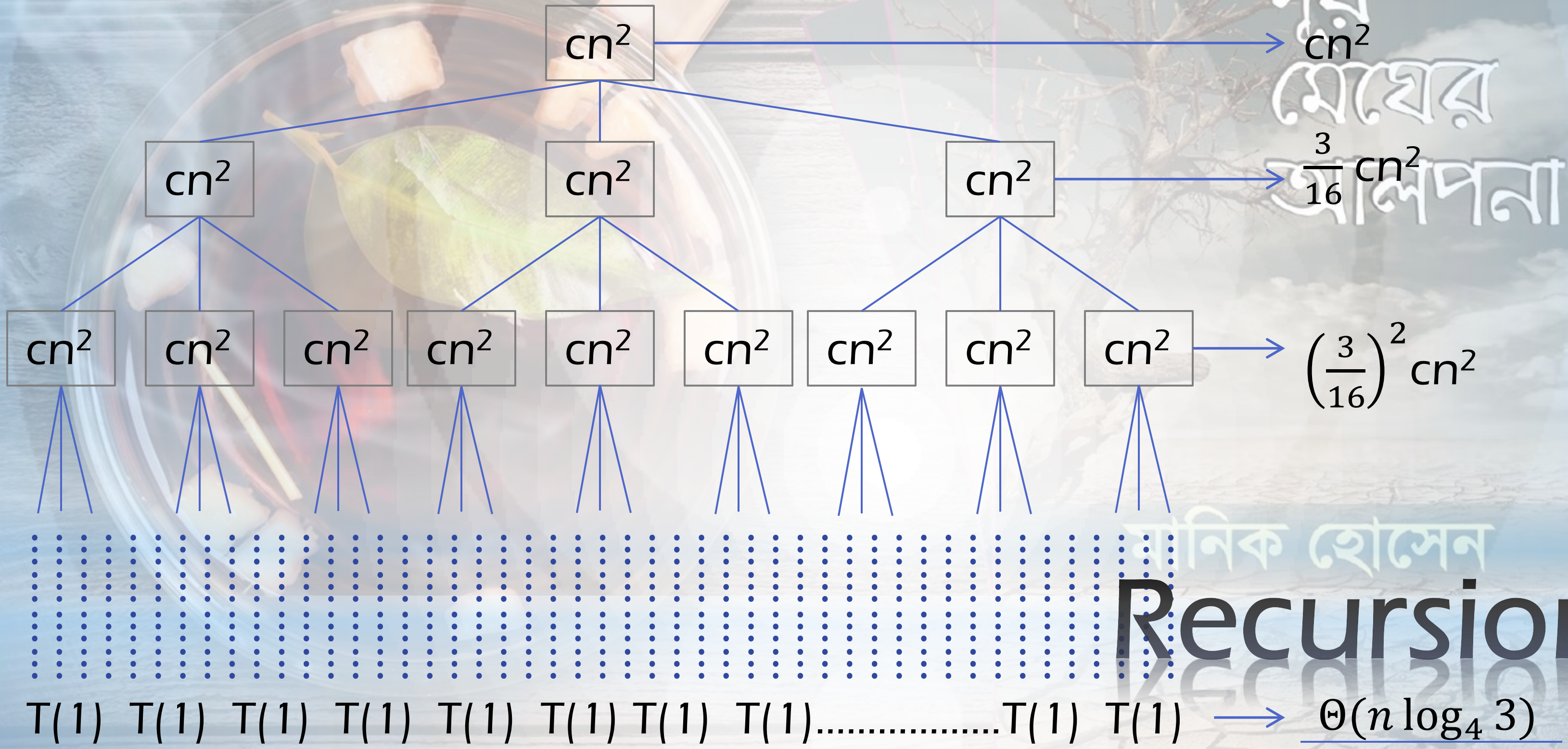
$$T(n) = \frac{1+1+1}{2} n^2 + (1+1+1+\frac{1}{2} - \frac{1}{2} - \frac{1}{2}) n - (1+1+1+1)$$

$$T(n) = \frac{3}{2}n^2 + \frac{5}{2}n - 4c = \Theta(n^2)$$

Insertion Sort

□ Show how a recursion tree provide a good guess for recurrence T(n)=3T(n/4)+cn^2.

□ Recursion tree for given recurrence:



cn^2

cn^2 cn^2 cn^2

cn^2 cn^2 cn^2 cn^2 cn^2 cn^2 cn^2 cn^2 cn^2

T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1)..................T(1) T(1)

cn^2

$\frac{3}{16}cn^2$

$\left(\frac{3}{16}\right)^2 cn^2$

$\Theta(n\log_4 3)$

Total: O(n^2)

□ From the tree we can say, recursion gives good guess.

Recursion

Dynamic Programming

❑Compare between Divide-and-Conquer method and Dynamic Programming.

❑Ans: The primary difference is that the sub problems of the divide-and-conquer approach are independent, while in dynamic programming they interact. Also, dynamic programming solves problems in a “bottom-up” manner as opposed to divide-and-conquer’s “top-down” approach.

Huffman Coding

- Consider the array A that contains the frequency of the charecters in a file. What will be the minimum file size if we apply fixed length coding? Compute Huffman Coding and compute file size after Huffman Coding.

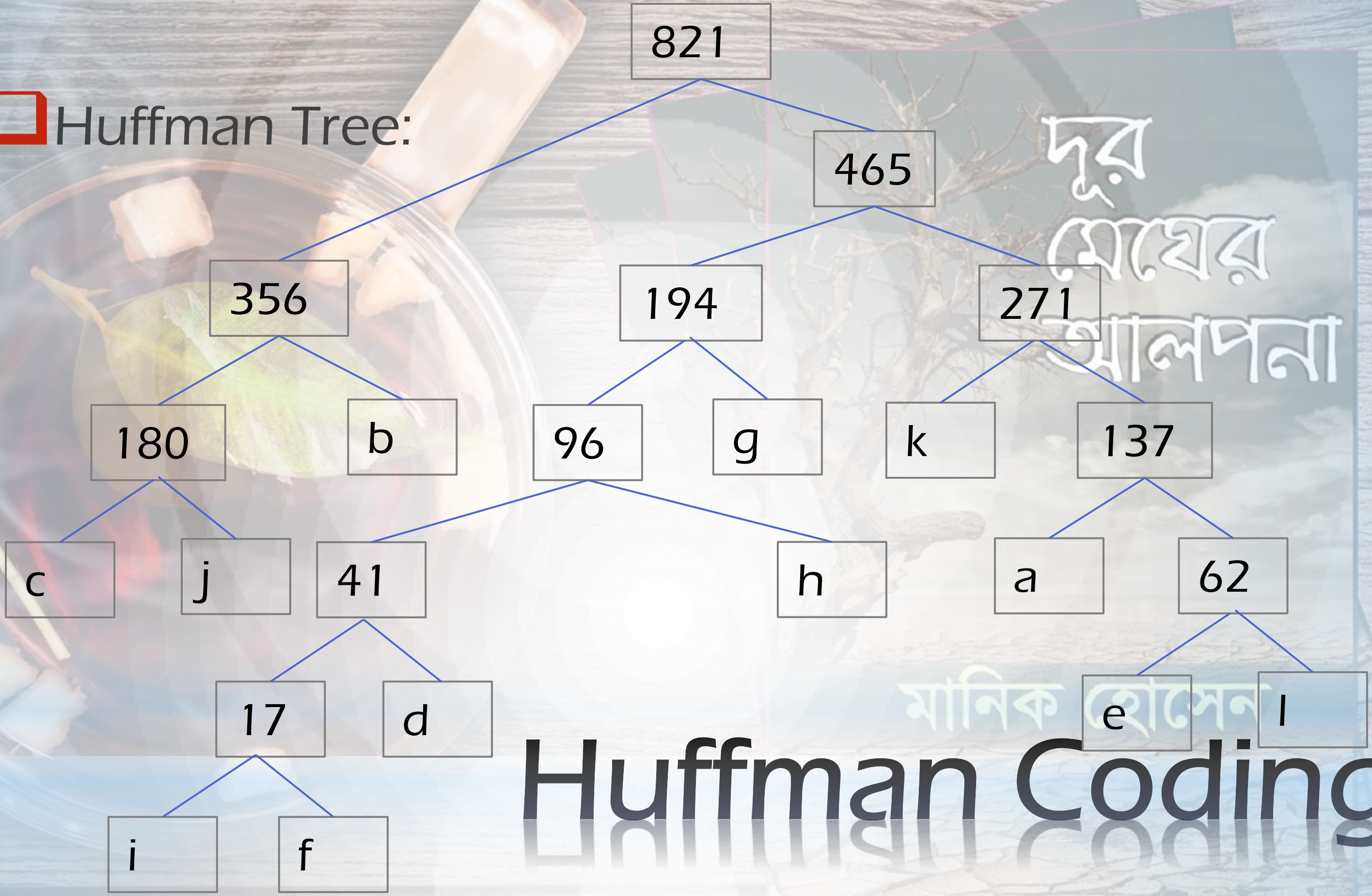
Charecter	Fmrequency(A)
a	75
b	176
c	89
d	24
e	33
f	5
g	98
h	55
i	12
j	91
k	134
l	29

- Fixed Length Coding:

Charcter	a	b	c	d	e	f	g	h	i	j	k	l
Code	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011

We see 4 bits per character. Hence the file size is: 1000x4=4000bits.

- Huffman Tree:



Huffman Coding

❑ Codeword after Huffman Coding:

Charcter	C	J	b	i	f	d	h	g	k	a	e	l
Code	000	001	01	100000	100001	10001	1001	101	110	1110	11110	11111

❑ The file size is now: 1000x 1000000 = 10^9 bits

CT

- ❑You have to set a problem for a competitive programming contest. Write a problem description so that it can be solved using DFS algorithm. Prepare sample input output and judge input output. Judge input output should contain at least one critical input output. Which one is critical output and why, Explain.

DFS

CT

- ❑ Problem:
- ❑ You are given a graph with *N* vertices and *M* edges. Master parent is the vertex which has no parent but may have 0 or more children. In any connected component of the graph, vertex with the lowest value in that component serves as the master parent.
  A vertex is called happy if it has more children than its parent. Count the number of happy vertices in the given graph. The graph has no cycles or self loops.
- ❑ **Input Format**:

  First line consists of two space separated integers denoting *N* and *M* and the following *M* lines consist of two space separated integers *X* and *Y* denoting there is an edge between vertices *X* and *Y*.
- ❑ **Output Format**:

  Print the number of happy vertices in the graph.

See in HackerEarth

DFS

CT

□ Sample Input:

```
4 3
1 2
2 3
2 4
```

□ Sample Output:

```
1
```

See in HackerEarth

DFS

CT

□ Judge Input:

```
4 3
1 2
2 3
2 4
```

□ Judge Output:

```
1
```

□ Judge Input:

```
7 6
1 2
2 3
2 4
4 5
4 6
4 7
```

□ Judge Output:

```
2
```

□ The Second one is critical output because it contains too much nodes.

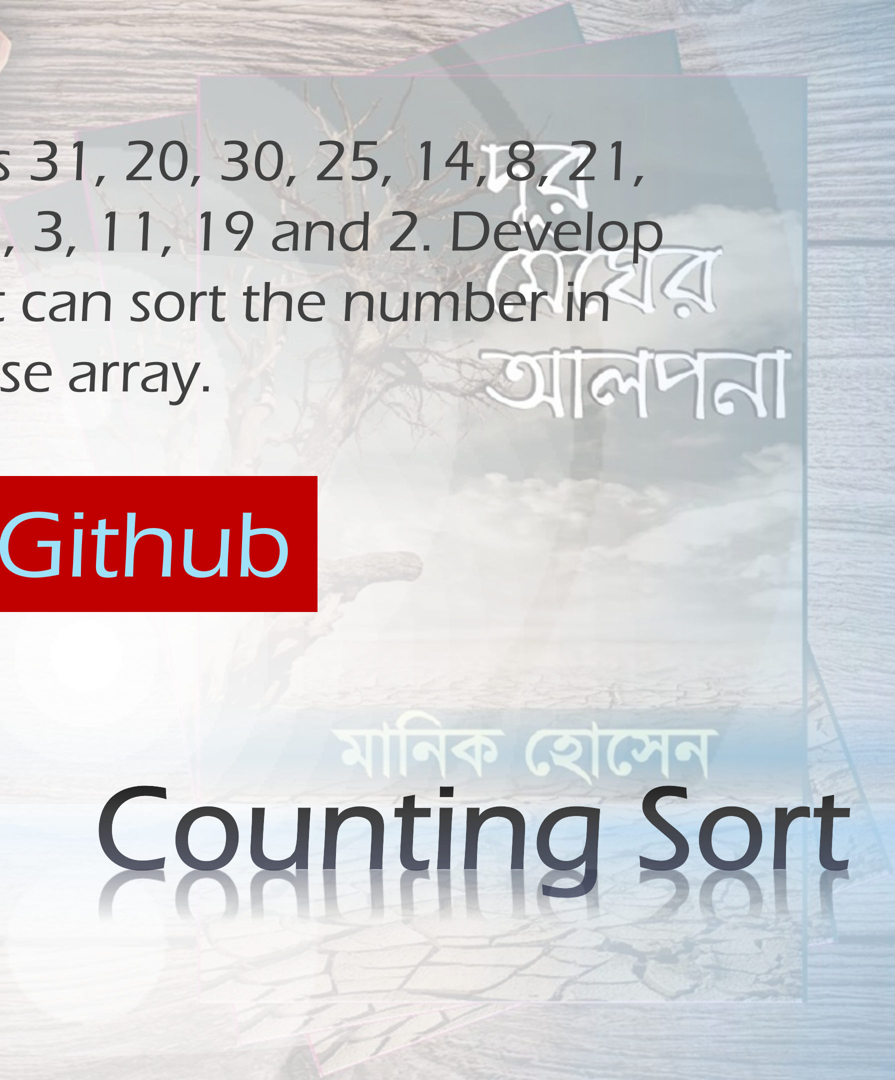
See in HackerEarth

DFS

CT

- □ Given some numbers 31, 20, 30, 25, 14, 8, 21, 9, 26, 12, 28, 14, 7, 27, 3, 11, 19 and 2. Develop an algorithm so that it can sort the number in O(n) time. You can’t use array.
- □ Code:

See in Github



Counting Sort

CT

- In terms of time complexity and space complexity, which input representation is suitable for Breadth-First-Search algorithm and why, Explain.
- Ans:
- The adjacency list representation is suitable.
- The complexity of BFS implemented using an Adjacency Matrix will be $O(|V|^2)$. And that when implemented by an Adjacency List is $O(|V| + |E|)$.
- This is mainly because every time we want to find what are the edges adjacent to a given vertex 'U', we would have to traverse the whole array AdjacencyMatrix[U], which is off course of length V.

BFS

CT

- □ N points of a polygon are given. How can you efficiently find out, if a polygon is convex or non-convex?
- □ Ans: Consider each set of three points along the polygon. If every angle is 180 degrees or less you have a convex polygon. When you figure out each angle, also keep a running total of (180 - angle). For a convex polygon, this will total 360.
- □ This test runs in O(n) time.

See in StackOverFlow

Convex Hull

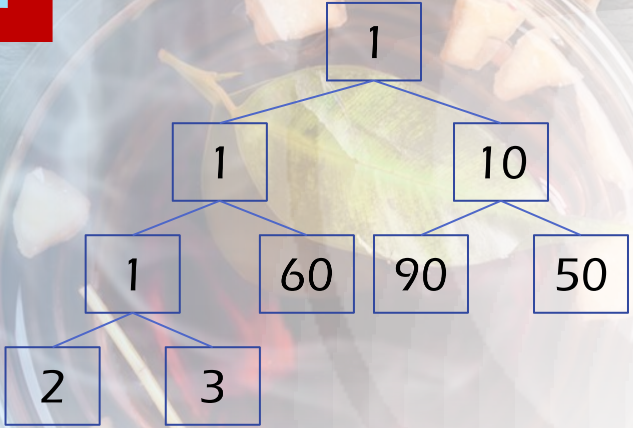
CT

- Given a set of numbers. Perform greatest common divisor (GCD) operation to any range of the numbers. Numbers are {2, 3, 60, 90, 50} and queries are given by index ranges {(1,3}, (2,4), (0,2)}. Represent the numbers using an efficient data structure and perform the queries.

Segment Tree

CT

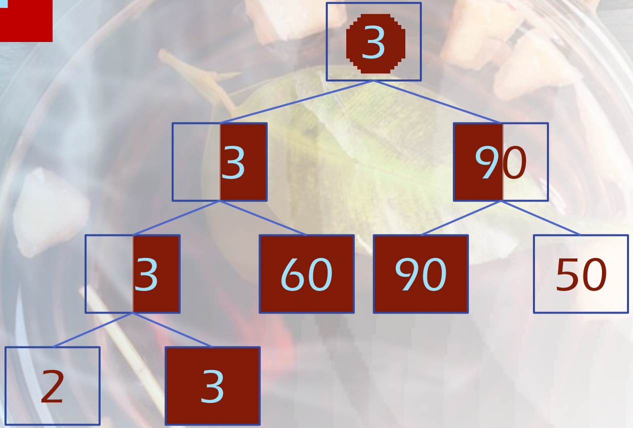
❑ Representation Using Tree:

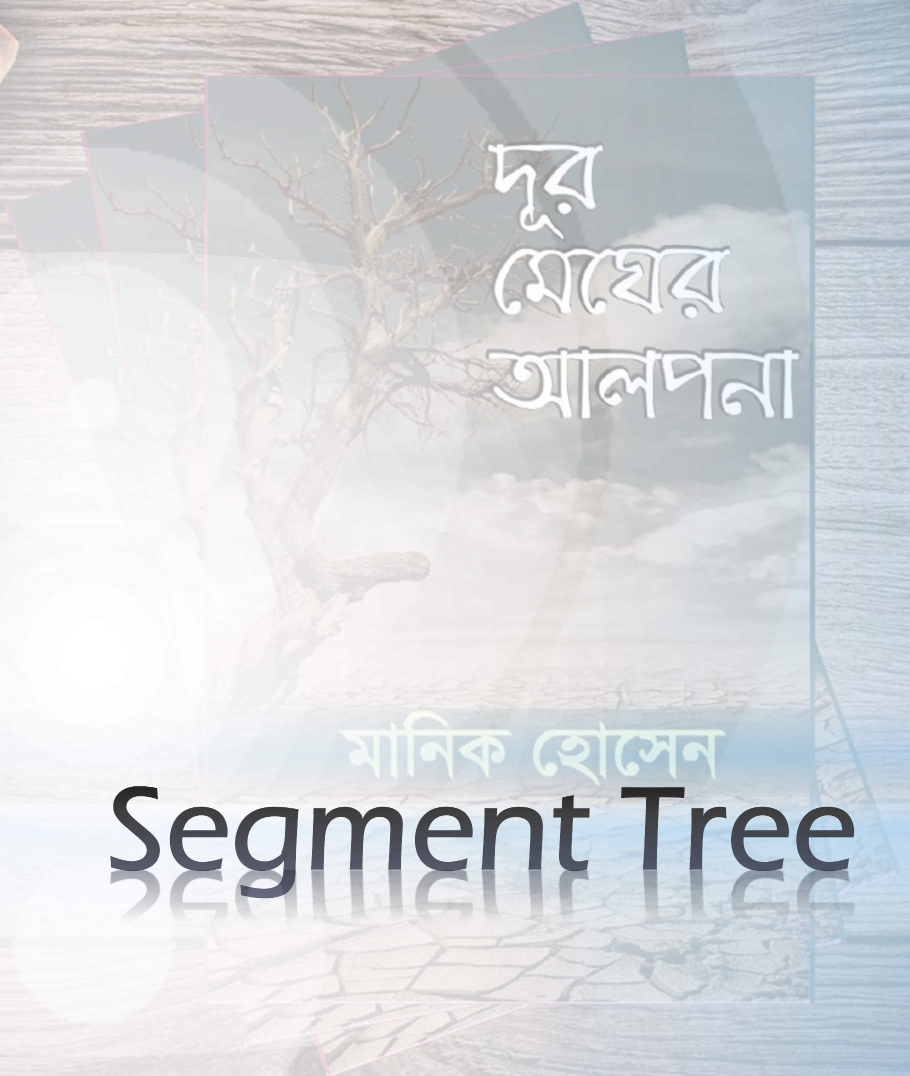


Segment Tree

CT

- Query in (1,3):

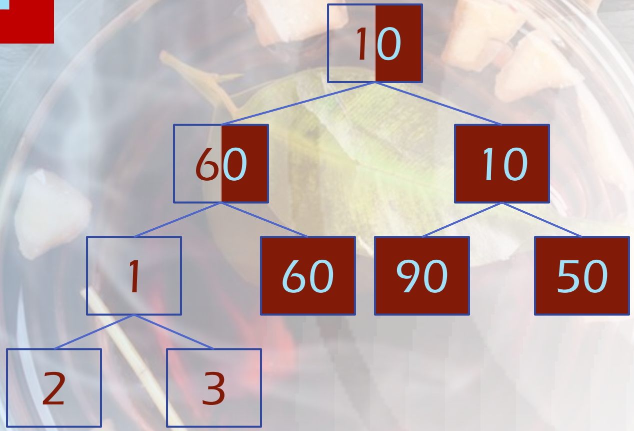




Segment Tree

CT

- Query in (2,4):



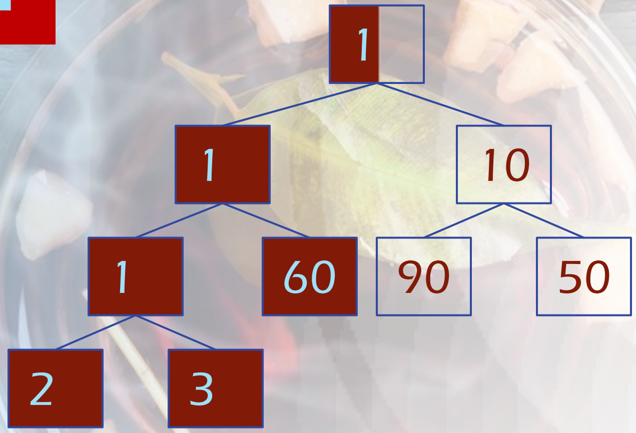
দূর মেঘের আলপনা

মানিক হোসেন

Segment Tree

CT

□ Query in (2,4):



Segment Tree

CT

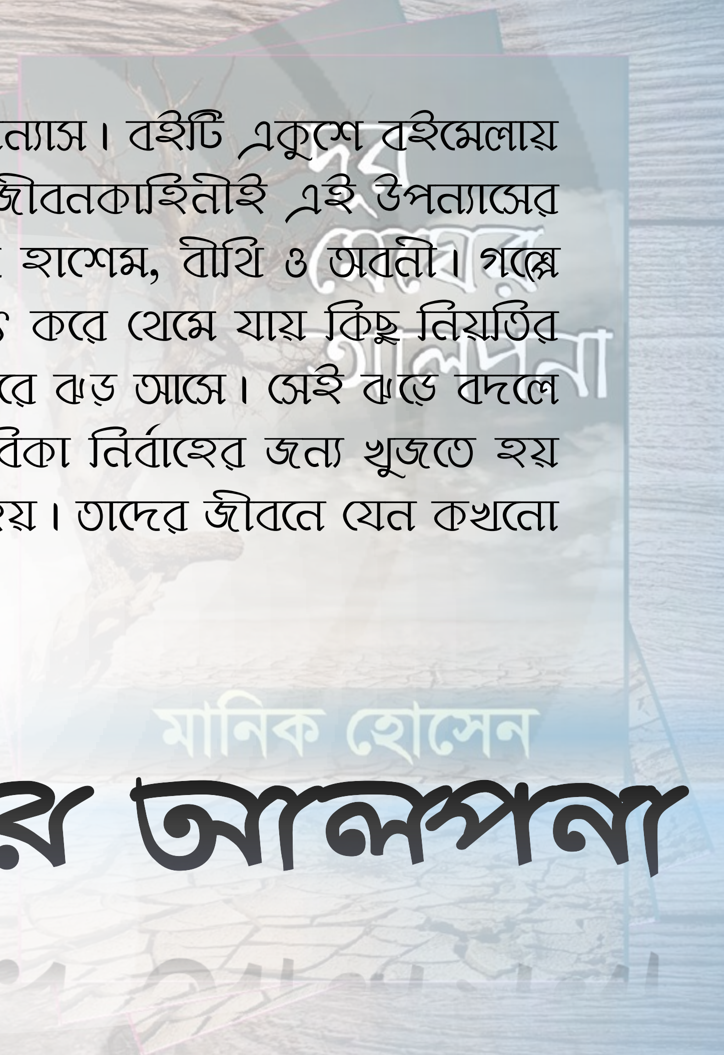
- How can you find the largest circle that can fit inside a concave polygon?
- Ans: Binary search for largest radius R for a circle:
- At each iteration, for a given radius r, push each edge E, "inward" by R, to get E'. For each edge E', define half-plane H as the set of all points "inside" the the polygon (using E' as the boundary). Now, compute the intersection of all these half-planes E', which could be done in O(n) time. If the intersection is non-empty, then if you draw a circle with radius r using any point in the intersection as the center, it will be inside the given polygon.

See in Github

See in StackOverFlow

Convex Hull

দূর মেঘের আলপনা আমার লেখা প্রথম উপন্যাস। বইটি একুশে বইমেলায় প্রকাশ হচ্ছে। গ্রামের মানুষের প্রতিদিনের জীবনকাহিনীই এই উপন্যাসের পটভূমি। উপন্যাসের কেন্দ্রীয় চরিত্রে রয়েছে হাশেম, বীথি ও অবনী। গল্পে গল্পে চলে তাদের জীবন। গল্পের মাঝে হঠ্যা করে থেমে যায় কিছু নিয়তির কারণে। এমনই একটি কারণে বীথির সংসারে ঝড় আসে। সেই ঝড়ে বদলে যায় অনেক কিছুই। তাদের প্রতিদিনের জীবিকা নির্বাহের জন্য খুজতে হয় বিভিন্ন পথ। তার জন্য অনেক দূরেও যেতে হয়। তাদের জীবনে যেন কখনো বৃষ্টি ঝরে না।



দূর মেঘের আলপনা