

Introduction to advanced data structures and algorithms

CSEN 1038

German University in Cairo

Outline

- 1 Motivation
- 2 Problem Solving Methodology
- 3 Administrivia
- 4 Lab Zero

Why study advanced data structures and algorithms?

■ Tasks

- set-set intersection
- sorting and retrieval

■ Problems

- deleting database records with dependency constraints
- Conflict resolution among versions (e.g. version control, distributed database eventual consistency)

Why study advanced data structures and algorithms?

■ Tasks

- set-set intersection
- sorting and retrieval

■ Problems

- deleting database records with dependency constraints
- Conflict resolution among versions (e.g. version control, distributed database eventual consistency)

Why study advanced data structures and algorithms?

A competitive programming based course



UNIVERSITÄT
DES
SAARLANDES

Problem Solving Methodology



Strategies

Tactics

Tools

Adopted from *The Art and Craft of Problem Solving*

Strategies

- Read the problem carefully
- Visualize
- Draw observations, Make Conclusions
- Get your hands dirty
- Build and trust your intuition
- Argue for your solution
- Stuck? Explore other ways

Strategies

- Read the problem carefully
- Visualize
- Draw observations, Make Conclusions
- Get your hands dirty
- Build and trust your intuition
- Argue for your solution
- Stuck? Explore other ways

Strategies

- Read the problem carefully
- Visualize
- Draw observations, Make Conclusions
- Get your hands dirty
- Build and trust your intuition
- Argue for your solution
- Stuck? Explore other ways

Strategies

- Read the problem carefully
- Visualize
- Draw observations, Make Conclusions
- Get your hands dirty
- Build and trust your intuition
- Argue for your solution
- Stuck? Explore other ways

Strategies

- Read the problem carefully
- Visualize
- Draw observations, Make Conclusions
- Get your hands dirty
- Build and trust your intuition
- Argue for your solution
- Stuck? Explore other ways

Strategies

- Read the problem carefully
- Visualize
- Draw observations, Make Conclusions
- Get your hands dirty
- Build and trust your intuition
- Argue for your solution
- Stuck? Explore other ways

Strategies

- Read the problem carefully
- Visualize
- Draw observations, Make Conclusions
- Get your hands dirty
- Build and trust your intuition
- Argue for your solution
- Stuck? Explore other ways

Tactics

Methods applicable to different settings

- Cyclic dependencies or recursive relationships
 - Graph modeling
- Range queries
 - Precomputation or dedicated data structures
- Optimization problems
 - Invariants or monotonicity

Tactics

Methods applicable to different settings

- Cyclic dependencies or recursive relationships
 - Graph modeling
- Range queries
 - Precomputation or dedicated data structures
- Optimization problems
 - Invariants or monotonicity

Tactics

Methods applicable to different settings

- Cyclic dependencies or recursive relationships
 - Graph modeling
- Range queries
 - Precomputation or dedicated data structures
- Optimization problems
 - Invariants or monotonicity

Tactics

Methods applicable to different settings

- Cyclic dependencies or recursive relationships
 - Graph modeling
- Range queries
 - Precomputation or dedicated data structures
- Optimization problems
 - Invariants or monotonicity

Tools

Focused techniques and tricks for specific situations.

What is this course about?



What is this course about?

■ What it is

- practical
- implementation oriented

■ What it is not

- theoretically formal
- heavily rigorous

What is this course about?

- What it is
 - practical
 - implementation oriented
- What it is not
 - theoretically formal
 - heavily rigorous

What is this course about?

- What it is
 - practical
 - implementation oriented
- What it is not
 - theoretically formal
 - heavily rigorous

What is this course about?

- What it is
 - practical
 - implementation oriented
- What it is not
 - theoretically formal
 - heavily rigorous

What is this course about?

- What it is
 - practical
 - implementation oriented
- What it is not
 - theoretically formal
 - heavily rigorous

What is this course about?

- What it is
 - practical
 - implementation oriented
- What it is not
 - theoretically formal
 - heavily rigorous

Course Material

All course material will be uploaded here
<https://github.com/AhmadHoseiny/ADSA-2025>

Course Resources

- <https://cp-algorithms.com/>
- **Introduction to Algorithms, 3rd Edition (CLRS)** by Cormen, Leiserson, Rivest, and Stein.

Course Assessment

- All assessments will be in the form of Codeforces contests and problems
- Lab and Home assignments are weekly based
- Attendance of lectures is highly recommended

Lab Assignments	20%
Home Assignments	50%
Project	30%

Course outline

Week	Topic
1	Introduction
2	Persistent Data Structures and Disjoint Sets Union
3	Matrix Exponentiation
4	Segment Trees with Lazy Propagation
5	Binary Search, Backpropagation and Meet in the Middle
6	String Processing Algorithms
7	Fast Fourier Transform
8	Strongly Connected Components and 2SAT Problem
9	Lowest Common Ancestor
10	Advanced Dynamic Programming
11	Max Flow Algorithms
12	Joker

Lab Zero

Lab Zero

Codeforces

A. Easy Problem

time limit per test: 1 second

memory limit per test: 256 megabytes

Cube is given an integer n . She wants to know how many ordered pairs of positive integers (a, b) there are such that $a = n - b$. Since Cube is not very good at math, please help her!

Input

The first line contains an integer t ($1 \leq t \leq 99$) — the number of test cases.

The only line of each test case contains an integer n ($2 \leq n \leq 100$).

Output

For each test case, output the number of ordered pairs (a, b) on a new line.

Example

input

Copy

3
2
4
6

output

Copy

1
3
5

Java Refresher - Sorting

```
Car[] arr = new Car[5];
Comparator<Car> c = (Car a, Car b) -> {
    return a.price - b.price;
};
Arrays.sort(arr, c);
// Collections.sort for ArrayList, LinkedList, etc.
```

Remark

Time complexity of the sorting is $O(n \log n)$

Java Refresher - HashSets

```
HashSet<Integer> hs = new HashSet<>();  
hs.add(e:1);  
hs.add(e:2);  
if(hs.contains(o:1)){  
    // do something  
}  
hs.remove(o:1);
```

Remark

Time complexity of these methods is roughly $O(1)$

Java Refresher - HashMaps

```
HashMap<Integer, Integer> hm = new HashMap<>();  
hm.put(key:1, value:2);  
hm.put(key:2, value:3);  
if(hm.containsKey(key:1)){  
    System.out.println(hm.get(key:1));  
}  
hm.remove(key:1);
```

Remark

Time complexity of these methods is roughly $O(1)$

Graphs Refresher - Representation

- Adjacency Matrix
- Adjacency List

Graphs Refresher - Representation

- Adjacency Matrix
- Adjacency List

Graphs Refresher - DFS

```
static ArrayList<Integer>[] adjL;  
static boolean [] visited;  
public static void dfs(int node){  
    visited[node] = true;  
    for(int child : adjL[node]){  
        if(!visited[child]){  
            dfs(child);  
        }  
    }  
}
```

Remark

Time complexity of DFS $O(n + m)$

Codeforces

Let's ace some problems !