

# Data Structures and Algorithms

**Prepared by: Mohamed Ayman**

Algorithm Engineer at Valeo

Deep Learning Researcher and Teaching Assistant  
at The American University in Cairo (AUC)

spring 2020



[sw.eng.MohamedAyman@gmail.com](mailto:sw.eng.MohamedAyman@gmail.com)



[linkedin.com/in/cs-MohamedAyman](https://linkedin.com/in/cs-MohamedAyman)



[github.com/cs-MohamedAyman](https://github.com/cs-MohamedAyman)



[codeforces.com/profile/Mohamed\\_Ayman](https://codeforces.com/profile/Mohamed_Ayman)



# Mohamed Ayman

## Experience



- Valeo
  - Deep Learning Researcher
  - Algorithm Software Engineer



- The American University in Cairo (AUC)
  - Research Assistant
  - Teaching Assistant



- ICPC - International Collegiate Programming Contest
  - Coach at ACPC Africa and Arab Collegiate Programming Contest



Cairo University

## Education

- MSc in Deep Learning, Cairo University
- BSc in Computer Science, Cairo University



# Data Structures and Algorithms Training



# Lecture Agenda

We will discuss in this lecture  
the following topics

- 1- Data Structures and Algorithms Features
  - 2- Data Structures and Algorithms Content
  - 3- Practice on Online Judges
  - 4- Programming Competitions
  - 5- Tutorials and References
  - 6- Online Courses
-



Let's  
**STARTUP**

# Lecture Agenda

---



**Section 1: Data Structures and Algorithms Features**

Section 2: Data Structures and Algorithms Content

Section 3: Practice on Online Judges

Section 4: Programming Competitions

Section 5: Tutorials and References

Section 6: Online Courses



# Data Structures and Algorithms Features

---



- **Data structure is a way to store and organize data** in order to support efficient insertions, queries, searches, updates, and deletions. Although a data structure in itself does not solve the given programming problem, the algorithm operating on it does, using the most efficient data structure for the given problem may be a difference between passing or exceeding the problem's time limit. There are many ways to organize the same data and sometimes one way is better than the other on different context.
- **Algorithms is the current term of choice for a problem-solving procedure**, algorithm, is commonly used nowadays for the set of rules a machine (and especially a computer) follows to achieve a particular goal. It does not always apply to computer-mediated activity, however. Algorithm is often paired with words specifying the activity for which a set of rules have been designed.
- **Characteristics of Data Structures and Algorithms:**
  - 1 - Correctness: Data structures and Algorithms implementation should implement its interface correctly.
  - 2 - Time Complexity: Running time or the execution time of operations must be as small as possible.
  - 3 - Space Complexity: Memory usage of a data structure operation should be as little as possible.

# Data Structures and Algorithms Features

---



➤ **Why we need data structures and algorithms?** there are several advantages of using them, few of them are as follows:

1. Data Organization: We need a proper way of organizing the data so that it can be accessed efficiently when we need that particular data. DS provides different ways of data organization so we have options to store the data in different data structures based on the requirement.

2. Efficiency: The main reason we organize the data is to improve the efficiency. We can store the data in arrays then why do we need linked lists and other data structures? because when we need to perform several operations such as add, delete, update and search on arrays, it takes more time in arrays than some of the other data structures. So the fact that we are interested in other data structures is because of the efficiency.

➤ **Time Complexity:** It is a way to represent the amount of time required by the program to run till its completion. It's generally a good practice to try to keep the time required minimum, so that our algorithm completes its execution in the minimum time possible. We will study about Time Complexity in details in later sections.

➤ **Space Complexity:** It's the amount of memory space required by the algorithm, during the course of its execution. Space complexity must be taken seriously for multi-user systems and in situations where limited memory is available.



# Data Structures and Algorithms Features

---



# Data Structures and Algorithms Features

---



# Lecture Agenda

---



✓ Section 1: Data Structures and Algorithms Features

**Section 2: Data Structures and Algorithms Content**

Section 3: Practice on Online Judges

Section 4: Programming Competitions

Section 5: Tutorials and References

Section 6: Online Courses



# Data Structures Content

---



## Part 1: Linear Data Structures

Lecture 1: Complexity Analysis & Recursion

Lecture 2: Array

Lecture 3: Linked List

Lecture 4: Stack

Lecture 5: Queue

Lecture 6: Deque

Lecture 7: Built-in Linear Data Structures

## Part 2: Non-Linear Data Structures

Lecture 8: Binary Tree

Lecture 9: Binary Search Tree

Lecture 10: Self Balancing Binary Search Tree

Lecture 11: Binary Heap Tree

Lecture 12: Hash Table

Lecture 13: Graph

Lecture 14: Built-in Non Linear Data Structures

# Data Structures Content

---



## Part 3: Advanced Data Structures

Lecture 15: Disjoint Set

Lecture 16: Skip List

Lecture 17: Trie

Lecture 18: Segment Tree

Lecture 19: Binary Indexed Tree (Fenwick Tree)

Lecture 20: Treap (Randomized Binary Search Tree)

## Part 4: Advanced Data Structures

Lecture 21: K-Dimensional Tree

Lecture 22: Sparse Table

Lecture 23: B/B+ Tree

Lecture 24: Suffix Array

Lecture 25: Suffix Tree

Lecture 26: Advanced Trees

# Hands-on Projects & Assignments & Practices

---



## Data Structures Projects (4 Projects)

- |  |   |
|--|---|
| Project 1: Mathematical Equations Calculator | (linear data structures application)              |
| Project 2: Mobile Contacts Indexing          | (linear & non-linear data structures application) |
| Project 3: Big Families                      | (non-linear data structures application)          |
| Project 4: University Friends                | (non-linear data structures application)          |

## Data Structures Assignments (10 Assignment)

- After each lecture we have an assignment (Implementing & Testing the Data Structures on each Lecture)

Data Structures Practices (30+ Practice Problems) on each Lecture.



# Algorithms Content

---



## Part 1: Basic Algorithms

### Lecture 1: Analysis of Algorithms

- Analysis Methods in Time & Space Complexity
- Master theorem - Substitution method
- Recursion tree method

### Lecture 2, 3: Sorting Algorithms

- |                  |                       |
|------------------|-----------------------|
| - Selection Sort | - Insertion Sort      |
| - Bubble Sort    | - Shell Sort          |
| - Merge Sort     | - Quick Sort          |
| - Heap Sort      | - Count Sort          |
| - Bitonic Sort   | - Radix Sort          |
| - Bucket Sort    | - Pigeonhole Sort     |
| - Tim Sort       | - Cartesian Tree Sort |

### Lecture 4, 5: Searching Algorithms

- |                      |                        |
|----------------------|------------------------|
| - Linear Search      | - Binary Search        |
| - Ternary Search     | - Jump Search          |
| - Exponential Search | - Sublist Search       |
| - Fibonacci Search   | - Interpolation Search |

### Lecture 6: Divide and Conquer Algorithms

- |   |                            |
|---|----------------------------|
| - Binary Search                               | - Merge & Quick Sort       |
| - Fast Power                                  | - Closest Pair of Points   |
| - Count Inversions                            | - Multiply Two Polynomials |
| - Strassen's Matrix Multiplication            |                            |
| - Karatsuba Algorithm for Fast Multiplication |                            |

# Algorithms Content

---



## Part 2: Graph Algorithms

### Lecture 7, 8, 9, 10: Graph Algorithms

- Graph Traversal
- Topological Sort
- Connectivity
- Lowest Common Ancestor
- Single source shortest paths
- All pairs shortest paths
- Dijkstra
- Spanning trees
- Minimum Spanning Tree
- Matching
- Cycles
- Backtracking
- Maximum Flow
- Floyd Warshall
- Bellman Ford
- Kirchhoff Theorem
- Prim & Kruskal

### Lecture 11, 12: Greedy Algorithms

- Standard Greedy Algorithms
- Greedy Algorithms in Graph
- Greedy Algorithms in Arrays
- Greedy Algorithms in Operating Systems



# Algorithms Content

---



## Part 3: Mathematical Algorithms

### Lecture 13, 14: Mathematical Algorithms

- Greatest Common Divisor (GCD)
- Latest Common Multiple (LCM)
- Prime Factorization and Divisors
- Chinese Remainder Theorem
- Sieve Algorithm
- Modular Arithmetic
- Euler Totient Function
- Number Theory
- nCr Computations
- Series

### Lecture 15, 16: Geometric Algorithms

- Lines
- Polygon
- Circle
- Quickhull
- Triangle
- Rectangle
- Square
- Convex Hull
- Quadrilateral
- 3D Objects
- Plane Sweep
- Voronoi diagrams
- Delaunay triangulations

### Lecture 17, 18, 19, 20: Computer Graphics Algorithms

- Line Generation Algorithm
- Circle Generation Algorithm
- Polygon Filling Algorithm
- Viewing & Clipping Algorithm
- 2D Transformation
- 3D Transformation
- Projection from 3D to 2D
- Computer Graphics Curves
- Computer Graphics Surfaces
- Visible Surface Detection
- Computer Graphics Fractals

# Algorithms Content

---



## Part 4: Dynamic Programming

### Lecture 21: Bitwise Algorithms

- Bit Manipulation
- Bitmasks Algorithm
- Bit Stuffing in Computer Networks
- Error Detection in Computer Networks

### Lecture 22, 23, 24: Dynamic Programming

- Overlapping Sub-problems Property
- Optimal Sub-structure Property
- Tabulation vs Memoization
- Bitmasking & Dynamic Programming

### Lecture 25, 26: Randomized Algorithms

- Randomized Quick Sort
- Monte Carlo Algorithms
- Las Vegas Algorithms
- Atlantic City Algorithms
- Computational Complexity

## Part 5: String Algorithms

### Lecture 27: String Algorithms

- Anagram
- Palindrome
- Binary String
- Subsequence
- Pattern Searching

### Lecture 30, 31, 32: Pattern Searching Algorithms

- Naïve Pattern Searching
- KMP Algorithm
- Rabin-Karp Algorithm
- Finite Automata
- Boyer Moore Algorithm
- Z Algorithm
- Aho-Corasick Algorithm
- Kasai's Algorithm
- Anagram Substring Search
- Pattern Searching using a Trie of all Suffixes

### Lecture 28, 29: String Compression Algorithms

- Lempel-Ziv Compression (LZ77 & LZ78)
- Lempel-Ziv-Markov Chain Algorithm (LZMA)
- Lempel-Ziv-Oberhumer (LZO)
- Lempel-Ziv-Storer-Szymanski (LZSS)
- Lempel-Ziv-Welch (LZW)
- Lempel-Ziv Finite State Entropy (LZFSE)
- Standard Huffman Coding Algorithm
- Modified Huffman Coding Algorithm
- Adaptive Huffman Coding Algorithm
- Arithmetic Coding (Float & Binary)

# Hands-on Projects & Assignments & Practices

---



## Algorithms Projects (6 Projects)

Project 1: Dictionary Sorting Simulator

(sorting application)

Project 2: Dictionary Searching Simulator

(searching application)

Project 3: Advanced Mathematical Calculator

(mathematical application)

Project 4: Advanced Geometric Simulator

(geometric application)

Project 5: Advanced Computer Graphics Generator

(computer graphics application)

Project 6: Advanced Computer Graphics Simulator

(computer graphics application)

## Algorithms Assignments (8 Assignment)

- After each lecture we have an assignment (Implementing & Testing the Algorithms on each Lecture)

Algorithms Practices (30+ Practice Problems) on each Lecture.

# Lecture Agenda

---



✓ Section 1: Data Structures and Algorithms Features

✓ Section 2: Data Structures and Algorithms Content

**Section 3: Practice on Online Judges**

Section 4: Programming Competitions

Section 5: Tutorials and References

Section 6: Online Courses



# Practice on Online Judges

---



[codeforces.com](https://codeforces.com)



[hackerearth.com](https://hackerearth.com)



[hackerrank.com](https://hackerrank.com)



[topcoder.com](https://topcoder.com)



[atcoder.jp](https://atcoder.jp)



[onlinejudge.org](https://onlinejudge.org)

# Codeforces Online Judge



Codeforces is a website that hosts competitive programming contests. It is maintained by a group of competitive programmers from ITMO University led by Mikhail Mirzayanov.



# Codeforces Online Judge



Mohamed Ayman | [Logout](#)

[HOME](#) [TOP](#) [CONTESTS](#) [GYM](#) [PROBLEMSET](#) [GROUPS](#) [RATING](#) [API](#) [HELP](#) [CALENDAR](#)

[MAIN](#) [ACMSGURU](#) | [PROBLEMS](#) [SUBMIT](#) [STATUS](#) [STANDINGS](#) [CUSTOM TEST](#)

#	Name				
<a href="#">1294F</a>	<a href="#">Three Paths on a Tree</a>	dfs and similar, dp, trees			2100 <a href="#">x1488</a>
<a href="#">1294E</a>	<a href="#">Obtain a Permutation</a>	greedy, implementation, math			2000 <a href="#">x1886</a>
<a href="#">1294D</a>	<a href="#">MEX maximizing</a>	data structures, math			1600 <a href="#">x4272</a>
<a href="#">1294C</a>	<a href="#">Product of Three Numbers</a>	greedy, math, number theory			1300 <a href="#">x8603</a>
<a href="#">1294B</a>	<a href="#">Collecting Packages</a>	implementation, sortings			1200 <a href="#">x9042</a>
<a href="#">1294A</a>	<a href="#">Collecting Coins</a>	math			900 <a href="#">x12769</a>
<a href="#">1293B</a>	<a href="#">JOE is on TV!</a>	combinatorics, greedy, math			1000 <a href="#">x9324</a>
<a href="#">1293A</a>	<a href="#">ConneR and the A.R.C. Markland-N</a>	binary search, brute force, implementation			1100 <a href="#">x8075</a>
<a href="#">1292F</a>	<a href="#">Nora's Toy Boxes</a>	bitmasks, combinatorics, dp			3400 <a href="#">x24</a>

## → Pay attention

**Before contest**  
[Educational Codeforces Round 81](#)  
(Rated for Div. 2)  
3 days

Like 137 people like this. Be the first of your friends.

## → Filter Problems

Difficulty:  —   
[Add tag](#)

## → Settings

☒ Show tags for unsolved problems





# Register in New Contest



Register The Contest from [Register now >>](#) link



[Enter](#) | [Register](#)

[HOME](#) [TOP](#) [CONTESTS](#) [GYM](#) [PROBLEMSET](#) [GROUPS](#) [RATING](#) [API](#) [HELP](#) [CALENDAR](#)

Current or upcoming contests					
Name	Writers	Start	Length		
Microsoft Q# Coding Contest - Winter 2019 <a href="#">Enter &gt;&gt;</a>	Nickolas	<a href="#">Mar/01/2019 19:00UTC+2</a>	3:00:00	<a href="#">Current standings</a> Running 47:16:49	<a href="#">Register &gt;&gt;</a> <a href="#">x4882</a> Until closing 47:16:49
Codeforces Round #543 (Div. 1, based on Technocup 2019 Final Round)		<a href="#">Mar/03/2019 17:35UTC+2</a>	02:00	Before start 21:51:49	<a href="#">Register &gt;&gt;</a> <a href="#">x93</a> Until closing 21:46:49 *has extra registration
Codeforces Round #543 (Div. 2, based on Technocup 2019 Final Round)		<a href="#">Mar/03/2019 17:35UTC+2</a>	02:00	Before start 21:51:49	<a href="#">Register &gt;&gt;</a> <a href="#">x667</a> Until closing 21:46:49 *has extra registration

## → Pay attention

Before contest  
[Codeforces Round #543 \(Div. 1, based on Technocup 2019 Final Round\)](#)  
21:51:49  
[Register now >>](#)  
\*has extra registration



# Register in Previous Contest



## You Can Compete in Previous Contests

### Contest history

#### Past contests

Name	Writers	Start	Length		
Codeforces Round #542 [Alex Lopashev Thanks-Round] (Div. 1) <a href="#">Enter »</a> <a href="#">Virtual participation »</a>	<a href="#">top34051</a> <a href="#">zoomswk</a>	Feb/24/2019 17:35 <sup>UTC+2</sup>	02:00	<a href="#">Final standings</a>	<a href="#">x793</a>
Codeforces Round #542 [Alex Lopashev Thanks-Round] (Div. 2) <a href="#">Enter »</a> <a href="#">Virtual participation »</a>	<a href="#">MikeMirzayanov</a> <a href="#">top34051</a> <a href="#">zoomswk</a>	Feb/24/2019 17:35 <sup>UTC+2</sup>	02:00	<a href="#">Final standings</a>	<a href="#">x7221</a>
Codeforces Round #541 (Div. 2) <a href="#">Enter »</a> <a href="#">Virtual participation »</a>	<a href="#">MikeMirzayanov</a> <a href="#">Sehnsucht</a> <a href="#">Sender</a> <a href="#">V--gLaSsH0ldEr593--V</a> <a href="#">VFeafanov</a> <a href="#">_kun_</a> <a href="#">ch_egor</a> <a href="#">grphil</a> <a href="#">voidmax</a>	Feb/23/2019 12:20 <sup>UTC+2</sup>	02:00	<a href="#">Final standings</a>	<a href="#">x8568</a>



# AtCoder - Online Judge



Home Contest Ranking

English Mohamed\_Ayman

## Contest

### Permanent Contests

#### Contest Name

• practice contest

### Upcoming Contests

#### Start Time

#### Contest Name

5/30(Sat) 14:00

• NOMURA Programming Competition 2020

### Recent Contests

#### Start Time

#### Contest Name

4/26(Sun) 14:00

• AtCoder Beginner Contest 164

4/19(Sun) 14:00

• AtCoder Beginner Contest 163

4/12(Sun) 14:00

• AtCoder Beginner Contest 162

4/4(Sat) 14:00

• AtCoder Beginner Contest 161

[Detail]

## Information

[How to get an account / participate in contests?](#)

[AtCoder's Contest Format](#)

[AtCoder's Testcases - AtCoder's Rating System - AtCoder Race Ranking](#)

### AtCoder Beginner Contest 164 Announcement

posted: about 17 hours ago

We will hold AtCoder Beginner Contest 164.

- Contest URL: <https://atcoder.jp/contests/abc164>
- Start Time: <http://www.timeanddate.com/worldclock/fixedtime.html?iso=20200426T2100&p1=248>
- Duration: TBD (around 2 hours)
- Number of Tasks: 6
- Writer: [kyopro\\_friends](#), [Kmcode](#), [latte0119](#), [tozangezan](#), [ynymxiaolongbao](#), [wo01](#)
- Rated range: ~ 1999

The point values will be 100-200-300-400-500-600.


We are looking forward to your participation!

last update: about 17 hours ago



# AtCoder - Online Judge



 AtCoder Beginner Contest 164

English Mohamed\_Ayman (Guest)

Contest Duration: 2020-04-26(Sun) 14:00 ~ 2020-04-26(Sun) 15:40 (local time) (100 minutes) [Back to Home](#)

[Top](#) [Tasks](#) [Clarifications](#) [Submit](#) [Results](#) [Standings](#) [Virtual Standings](#) [Custom Test](#) [Editorial](#) [Discuss](#)

## AtCoder Beginner Contest 164

Virtual Participation

Can Participate: All Rated Range: ~ 1999 Penalty: 5 minutes

### Contest Information

- Duration: 100 minutes
- Rated Range: 0 - 1999

### Point Values

Task	Score
A	100
B	200
C	300
D	400
E	500
F	600



# AtCoder - Online Judge



## Contest Rules

This contest is full-feedback (solutions are judged during the contest).

When you solve a problem, you get a score assigned to it. Competitors are ranked first by total scores, then by penalties. The penalties are computed as (the time you spend to get your current score) + (5 minutes) \* (the number of incorrect attempts).

## Useful Links

- [AtCoder top page](#)
- [How to participate](#)
- [Practice contest](#)



# HackerEarth - Online Judge



A screenshot of the HackerEarth homepage. The background is a dark blue with a subtle pattern of people's faces. At the top left is the 'hackerearth' logo. To its right are links for 'For Developers', 'For Businesses' (with a dropdown arrow), 'Login', and a 'Sign up' button. On the far right, there is a vertical green bar with a red '1' and the text 'LIVE EVENTS'. The main content area features the text 'Home to 3M+ talented developers. Trusted by 1000+ top enterprises.' followed by a paragraph: 'From supporting developers who shape technology to helping enterprises innovate at scale and hire tech talent, HackerEarth bridges the gap between tomorrow's tech powered growth and today's workforce.' Below this is a blue button that says 'Join our community'. In the bottom right corner, there is a network diagram with nodes and lines, and a circular icon with a question mark.



# HackerEarth - Online Judge



Signup and get free access to 100+ Tutorials and Practice Problems

Start Now

LIVE EVENTS

## Programming Tutorials and Practice Problems

### RECOMMENDED



#### Start online programming

Familiarize with online programming in just 7 easy steps



#### Code Monk

Be better at programming, one step at a time

### ALL TRACKS



#### Basic Programming

Input/Output, Complexity Analysis, Implementation, etc.

THIS WEEK'S LEADER



Mayank Chaudhary

POINTS

782.9



#### Data Structures

Arrays, Stacks, Queues, etc.

THIS WEEK'S LEADER



Suraj Jha

POINTS

406.1



#### Algorithms

Searching, Sorting, Greedy Algorithms, etc.

THIS WEEK'S LEADER



DARK\_SHADOW

POINTS

500.3



#### Math

Number Theory, Combinatorics, Geometry

THIS WEEK'S LEADER



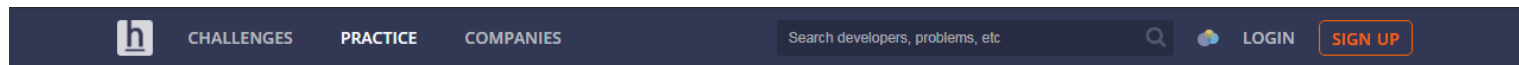
Satyam Kumar

POINTS

330.0



# HackerEarth - Online Judge



Signup and get free access to 100+ Tutorials and Practice Problems [Start Now](#)

LIVE EVENTS

All Tracks > Basic Programming > Input/Output > Basics of Input/Output



## Basic Programming

Solve any problem to achieve a rank  
[View Leaderboard](#)

Input/Output

Complexity Analysis

Implementation

Operators

Bit Manipulation

Recursion

## Basics of Input/Output

[TUTORIAL](#) [PROBLEMS](#)

The very first step of getting started with online judge is to understand:

- How to read input data?
- How to output the answer?

At HackerEarth, input data is read from [standard input](#) stream (STDIN) and results are printed to [standard output](#) stream (STDOUT). Most of the questions will deal with either integers or strings.

An example C code to read an integer from STDIN and printing it out to STDOUT is shown below.

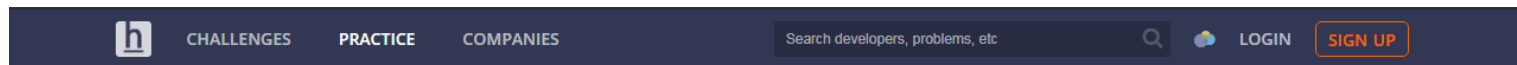
```
#include <stdio.h>
int main()
{
    int n;
    scanf("%d",&n); //read input integer from STDIN
    printf("%d",n); //print output integer to STDOUT
    return 0;
}
```

Sample code snippet to read integer for all other languages are given in the code editor below.





# HackerEarth - Online Judge



Signup and get free access to 100+ Tutorials and Practice Problems [Start Now](#)

LIVE EVENTS

[All Tracks](#) > [Math](#) > [Number Theory](#) > Basic Number Theory-1



Math

Solve any problem to achieve a rank  
[View Leaderboard](#)

Number Theory

Combinatorics

Geometry

## Basic Number Theory-1

[TUTORIAL](#) [PROBLEMS](#)

### Introduction

This article discusses topics that are frequently used to solve programming problems based on math. It includes the following topics:

1. Modular arithmetic
2. Modular exponentiation
3. Greatest Common Divisor (GCD)
4. Extended Euclidean algorithm
5. Modular multiplicative inverse

### 1. Modular arithmetic

When one number is divided by another, the modulo operation finds the remainder. It is denoted by the % symbol.

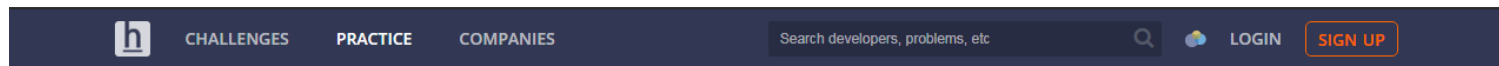
#### Example

Assume that you have two numbers 5 and 2.  $5\%2$  is 1 because when 5 is divided by 2, the remainder is 1.

#### Properties



# HackerEarth - Online Judge



Signup and get free access to 100+ Tutorials and Practice Problems [Start Now](#)

LIVE EVENTS

All Tracks > Data Structures > Arrays > 1-D



## Data Structures

Solve any problem to achieve a rank  
[View Leaderboard](#)

Arrays ▾

Stacks ▾

Queues ▾

Hash Tables ▾

Linked List ▾

Trees ▾

Advanced Data Structures ▾

Disjoint Data Structures ▾

## 1-D

[TUTORIAL](#) [PROBLEMS](#)

An array is a sequential collection of elements of same data type and stores data elements in a continuous memory location. The elements of an array are accessed by using an index. The index of an array of size  $N$  can range from  $0$  to  $N - 1$ . For example, if your array size is  $5$ , then your index will range from  $0$  to  $4$  ( $5-1$ ). Each element of an array can be accessed by using `arr[index]`.

Consider following array. The size of this array is  $5$ . If you want to access  $12$ , then you can access it by using `arr[ 1 ]` i.e.  $12$ .

<i>arr</i>	4	12	7	15	9
<i>index</i>	0	1	2	3	4

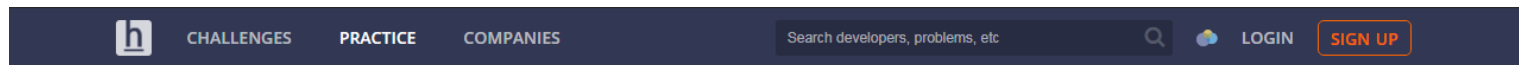
### Array declaration

Declaring an array is language-specific.

For example, in C/C++, to declare an array, you must specify, the following:



# HackerEarth - Online Judge



Signup and get free access to 100+ Tutorials and Practice Problems [Start Now](#)

LIVE EVENTS

All Tracks > Algorithms > Searching > Linear Search



## Algorithms

Solve any problem to achieve a rank  
[View Leaderboard](#)

Searching

Sorting

Greedy Algorithms

Graphs

String Algorithms

Dynamic Programming

## Linear Search

[TUTORIAL](#) [PROBLEMS](#)

Linear search is used on a collections of items. It relies on the technique of traversing a list from start to end by exploring properties of all the elements that are found on the way.

For example, consider an array of integers of size  $N$ . You should find and print the position of all the elements with value  $x$ . Here, the linear search is based on the idea of matching each element from the beginning of the list to the end of the list with the integer  $x$ , and then printing the position of the element if the condition is 'True'.

Implementation:

The pseudo code for this example is as follows :

```
for(start to end of array)
{
    if (current_element equals to 5)
    {
        print (current_index);
    }
}
```

For example, consider the following image:



# HackerRank - Online Judge

Virtual Event | 11/08 | Learn how to master the art and science of skill assessments | Live [streamed from San Francisco](#)

**HackerRank**

[Products](#) [Customers](#) [Resources](#) [Research](#) [Blog](#) [About Us](#)

[Login](#)

[Sign Up](#)

Join over **5 million developers**.  
Practice coding, prepare for interviews, and get hired.

[Sign Up & Code](#)

Hiring Talent? [Learn more](#)



UBER

vmware



# HackerRank - Online Judge



Practice

## Dashboard

### Explore HackerRank Skills

#### PROBLEM SOLVING

**Algorithms**

**Data Structures**

**Mathematics**

#### LANGUAGE PROFICIENCY

**C**

**C++**

**Java**

**Python**

**Ruby**

**Linux Shell**

**Functional Programming**

#### SPECIALIZED SKILLS

**Artificial Intelligence**

**SQL**

**Databases**

**Distributed Systems**

**Regex**

**Security**



# HackerRank - Online Judge



Practice > Functional Programming

## Functional Programming

<b>Solve Me First FP</b> <small>Easy, Max Score: 3, Success Rate: 98.79%,</small>	<a href="#">Solve Challenge</a>	<b>STATUS</b> <input type="checkbox"/> Solved <input type="checkbox"/> Unsolved  <b>DIFFICULTY</b> <input type="checkbox"/> Easy <input type="checkbox"/> Medium <input type="checkbox"/> Hard  <b>SUBDOMAINS</b> <input type="checkbox"/> Introduction <input type="checkbox"/> Recursion <input type="checkbox"/> Functional Structures <input type="checkbox"/> Memoization and DP <input type="checkbox"/> Persistent Structures <input type="checkbox"/> Ad Hoc <input type="checkbox"/> Parsers <input type="checkbox"/> Interpreter and Compilers
<b>Hello World</b> <small>Easy, Max Score: 5, Success Rate: 95.91%,</small>	<a href="#">Solve Challenge</a>	
<b>Hello World N Times</b> <small>Easy, Max Score: 5, Success Rate: 96.48%,</small>	<a href="#">Solve Challenge</a>	
<b>List Replication</b> <small>Easy, Max Score: 10, Success Rate: 97.88%,</small>	<a href="#">Solve Challenge</a>	
<b>Filter Array</b> <small>Easy, Max Score: 10, Success Rate: 99.26%,</small>	<a href="#">Solve Challenge</a>	



# HackerRank - Online Judge



Practice > Mathematics

## Mathematics

### Find the Point

Easy, Max Score: 5, Success Rate: 90.97%,

[Solve Challenge](#)

### Maximum Draws

Easy, Max Score: 5, Success Rate: 96.46%,

[Solve Challenge](#)

### Handshake

Easy, Max Score: 10, Success Rate: 94.09%,

[Solve Challenge](#)

### Minimum Height Triangle

Easy, Max Score: 10, Success Rate: 92.15%,

[Solve Challenge](#)

### Army Game

Easy, Max Score: 10, Success Rate: 85.78%,

[Solve Challenge](#)

#### STATUS

- ☐ Solved
- ☐ Unsolved

#### DIFFICULTY

- ☐ Easy
- ☐ Medium
- ☐ Hard


#### SUBDOMAINS

- ☐ Fundamentals
- ☐ Number Theory
- ☐ Combinatorics
- ☐ Algebra
- ☐ Geometry
- ☐ Probability
- ☐ Linear Algebra Foundations



# HackerRank - Online Judge



 [PRACTICE](#) [COMPETE](#) [JOBS](#) [LEADERBOARD](#)

Q Search [Log In](#) [Sign Up](#)

Practice > Data Structures

## Data Structures

<b>Arrays - DS</b> <small>Easy, Max Score: 10, Success Rate: 93.96%</small>	<a href="#">Solve Challenge</a>
<b>2D Array - DS</b> <small>Easy, Max Score: 15, Success Rate: 90.70%</small>	<a href="#">Solve Challenge</a>
<b>Dynamic Array</b> <small>Easy, Max Score: 15, Success Rate: 83.13%</small>	<a href="#">Solve Challenge</a>
<b>Left Rotation</b> <small>Easy, Max Score: 20, Success Rate: 87.15%</small>	<a href="#">Solve Challenge</a>
<b>Sparse Arrays</b> <small>Medium, Max Score: 25, Success Rate: 96.68%</small>	<a href="#">Solve Challenge</a>
<b>Array Manipulation</b> <small>Hard, Max Score: 60, Success Rate: 51.22%</small>	<a href="#">Solve Challenge</a>

**STATUS**  
☐ Solved  
☐ Unsolved

**DIFFICULTY**  
☐ Easy  
☐ Medium  
☐ Hard

**SUBDOMAINS**  
☐ Arrays  
☐ Linked Lists  
☐ Trees  
☐ Balanced Trees  
☐ Stacks  
☐ Queues  
☐ Heap  
☐ Disjoint Set  
☐ Multiple Choice  
☐ Trie  
☐ Advanced





# HackerRank - Online Judge



Practice > Algorithms

## Algorithms

<b>Solve Me First</b> <small>Easy, Max Score: 1, Success Rate: 98.14%,</small>	<a href="#">Solve Challenge</a>
<b>Simple Array Sum</b> <small>Easy, Max Score: 10, Success Rate: 94.53%,</small>	<a href="#">Solve Challenge</a>
<b>Compare the Triplets</b> <small>Easy, Max Score: 10, Success Rate: 94.01%,</small>	<a href="#">Solve Challenge</a>
<b>A Very Big Sum</b> <small>Easy, Max Score: 10, Success Rate: 98.61%,</small>	<a href="#">Solve Challenge</a>
<b>Diagonal Difference</b> <small>Easy, Max Score: 10, Success Rate: 95.86%,</small>	<a href="#">Solve Challenge</a>
<b>Plus Minus</b> <small>Easy, Max Score: 10, Success Rate: 98.12%,</small>	<a href="#">Solve Challenge</a>
<b>Staircase</b> <small>Easy, Max Score: 10, Success Rate: 98.12%,</small>	<a href="#">Solve Challenge</a>

STATUS

☐ Solved

☐ Unsolved

DIFFICULTY

☐ Easy

☐ Medium

☐ Hard

SUBDOMAINS

☐ Warmup

☐ Implementation

☐ Strings

☐ Sorting

☐ Search

☐ Graph Theory

☐ Greedy

☐ Dynamic Programming

☐ Constructive Algorithms

☐ Bit Manipulation

☐ Recursion

☐ Game Theory

☐ NP Complete

☐ Debugging



# Lecture Agenda

---



✓ Section 1: Data Structures and Algorithms Features

✓ Section 2: Data Structures and Algorithms Content

✓ Section 3: Practice on Online Judges

**Section 4: Programming Competitions**

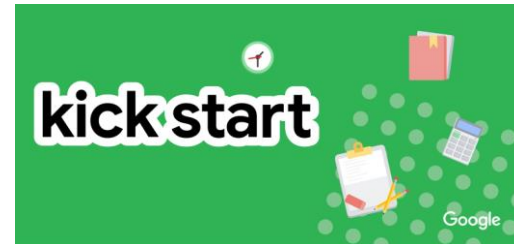
Section 5: Tutorials and References

Section 6: Online Courses



# Programming Competitions

---



# Google Competitions

---



**code jam**



**hash code**



**kick start**

# Google Competitions - Code Jam



**code jam** **WHAT LANGUAGE**  
**print "hello, world!"** **DO YOU SPEAK?**

# Google Competitions - Code Jam

---



- Code Jam - Practice Session March
- Code Jam - Qualification Round March
- Code Jam - Round 1A April
- Code Jam - Round 1B April
- Code Jam - Round 1C May
- Code Jam - Round 2 May
- Code Jam - Round 3 June
- Code Jam - World Finals August

**code jam**  
print "hello, world!"

# Google Competitions - Kick Start



Time to take your coding  
skills to the next level!

Register now

**kick start**



Google

# Google Competitions - Kick Start

---



- Kick Start - Round A March
- Kick Start - Round B April
- Kick Start - Round C May
- Kick Start - Round D July
- Kick Start - Round E August
- Kick Start - Round F September
- Kick Start - Round G October
- Kick Start - Round H November

The 'kick start' logo is displayed on a green rectangular background. The words 'kick' and 'start' are written in a bold, white, lowercase sans-serif font. The 'kick' part has a slight shadow or outline effect, making it stand out from the green background.



# Google Competitions - Hash Code

---



# Google Competitions - Hash Code

---



- Hash Code - Hub registration opens November
- Hash Code - Individual registration opens January
- Hash Code - Registration closes February
- Hash Code - Online qualification round February
- Hash Code - Results announced March
- Hash Code - Final round April



# Facebook Hacker Cup Competition

---



- Facebook Hacker Cup - Qualification round June
- Facebook Hacker Cup - Round 1 June
- Facebook Hacker Cup - Round 2 July
- Facebook Hacker Cup - Round 3 August
- Facebook Hacker Cup - Onsite Final September



# ICPC - International College Programming Contest

---



- Qualification Round in Universities September
- ECPC Egyptian College Programming Contest October
- ACPC Arab College Programming Contest January
- ICPC International College Programming Contest May



# Lecture Agenda

---



✓ Section 1: Data Structures and Algorithms Features

✓ Section 2: Data Structures and Algorithms Content

✓ Section 3: Practice on Online Judges

✓ Section 4: Programming Competitions

**Section 5: Tutorials and References**

Section 6: Online Courses

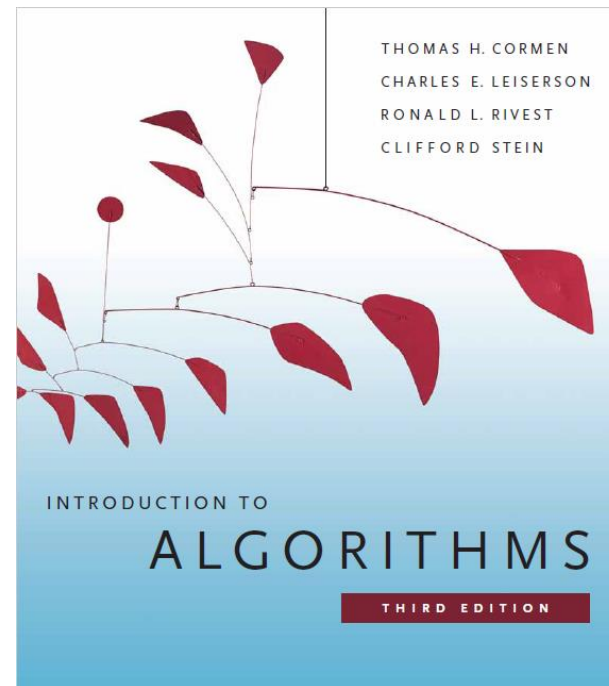


# Introduction to Algorithms Thomas H. Cormen

---



- Foundations [005 - 145]
- Sorting and Order Statistics [145 - 220]
- Data Structures [230 - 350]
- Advanced Design and Analysis Techniques [355 - 460]
- Advanced Data Structures [480 - 575]
- Graph Algorithms [585 - 750]
- Selected Topics [770 - 1130]

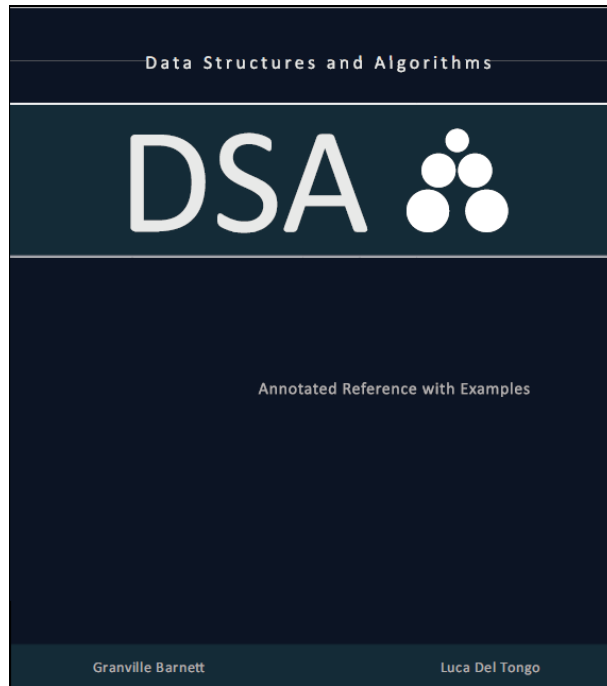


Introduction to Algorithms Thomas H. Cormen

# Data Structures and Algorithms Annotated Reference



- Introduction [01 - 10]
- Linked Lists [10 - 20]
- Binary Search Tree [20 - 30]
- Heap [30 - 40]
- Sets [40 - 50]
- Queues [50 - 55]
- AVL Tree [55 - 60]
- Sorting [60 - 70]
- Numeric [70 - 75]
- Searching [75 - 80]
- Strings [80 - 85]

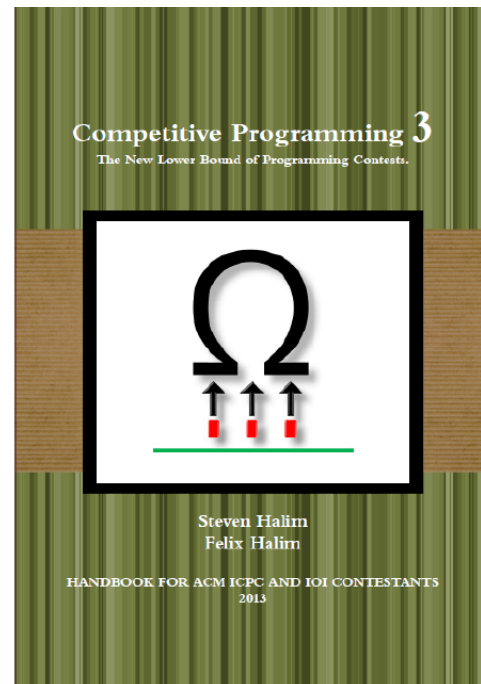


Data Structures and Algorithms Annotated Reference

# Competitive Programming 3 Steven Halim



- Introduction [001 - 030]
- Data Structures and Libraries [030 - 070]
- Problem Solving Paradigms [070 - 120]
- Graph [120 - 190]
- Mathematics [190 - 230]
- String Processing [230 - 270]
- (Computational) Geometry [270 - 300]
- More Advanced Topics [300 - 330]
- Rare Topics [330 - 390]



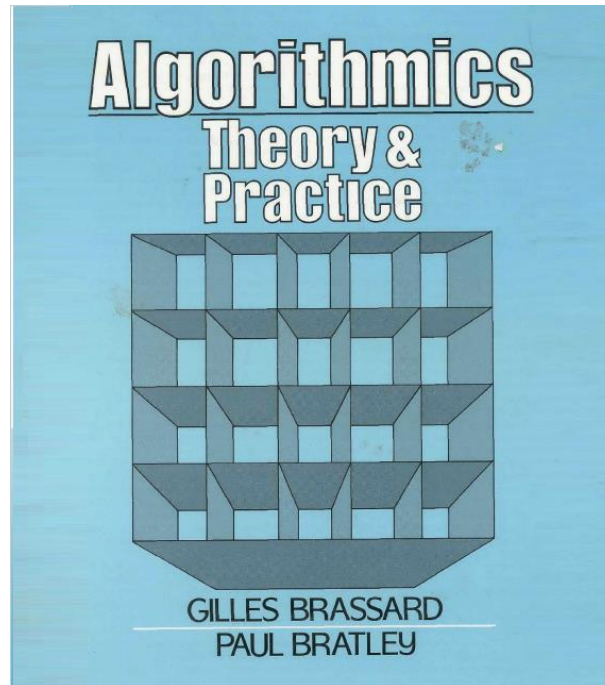
Competitive Programming 3 Steven Halim



# Fundamental of Algorithmics Gilles Brassard



- Preliminaries [001 - 035]
- Analyzing the Efficiency of Algorithms [035 - 080]
- Greedy Algorithms [080 - 105]
- Divide and Conquer [105 - 140]
- Dynamic Programming [140 - 170]
- Exploring Graphs [170 - 205]
- Preconditioning and Pre-computation [205 - 225]
- Probabilistic Algorithms [225 - 275]
- Transformations of the Domain [275 - 290]
- Introduction to Complexity [290 - 335]



Fundamental of Algorithmics Gilles Brassard and Paul Bartley

# Analysis of Algorithms An Active Learning Approach

---



- Analysis Basics [001 - 040]
- Searching and Selection Algorithms [040 - 055]
- Sorting Algorithms [060 - 100]
- Numeric Algorithms [105 - 120]
- Matching Algorithms [120 - 140]
- Graph Algorithms [145 - 175]
- Parallel Algorithms [175 - 210]
- Nondeterministic Algorithms [210 - 230]
- Other Algorithmic Techniques [230 - 260]

## **Analysis of Algorithms: An Active Learning Approach**

*Jeffrey J. McConnell*

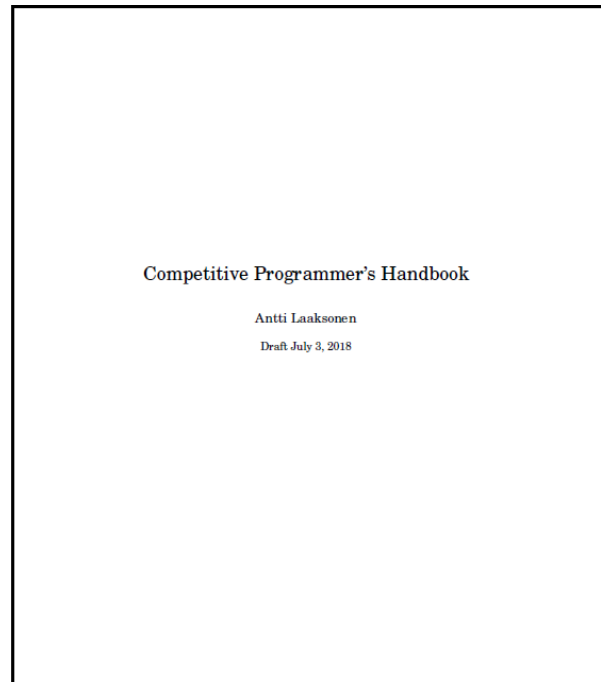
JONES AND BARTLETT PUBLISHERS

Analysis of Algorithms An Active Learning Approach

# Competitive Programmer's Handbook



- **Basic techniques** [001 - 105]
  - Time complexity
  - Sorting
  - Complete search
  - Greedy algorithms
  - Dynamic programming
  - Amortized analysis
  - Range queries
  - Bit manipulation
- **Graph algorithms** [105 - 195]
  - Graph traversal
  - Shortest paths
  - Spanning trees
  - Directed graphs
  - Tree queries
  - Paths and circuits
  - Tree algorithms
  - Strong connectivity
  - Flows and cuts
- **Advanced topics** [195 - 275]
  - Number theory
  - Combinatorics
  - Game theory
  - String algorithms
  - Segment trees revisited
  - Geometry
  - Matrices
  - Square root algorithms
  - Sweep line algorithms

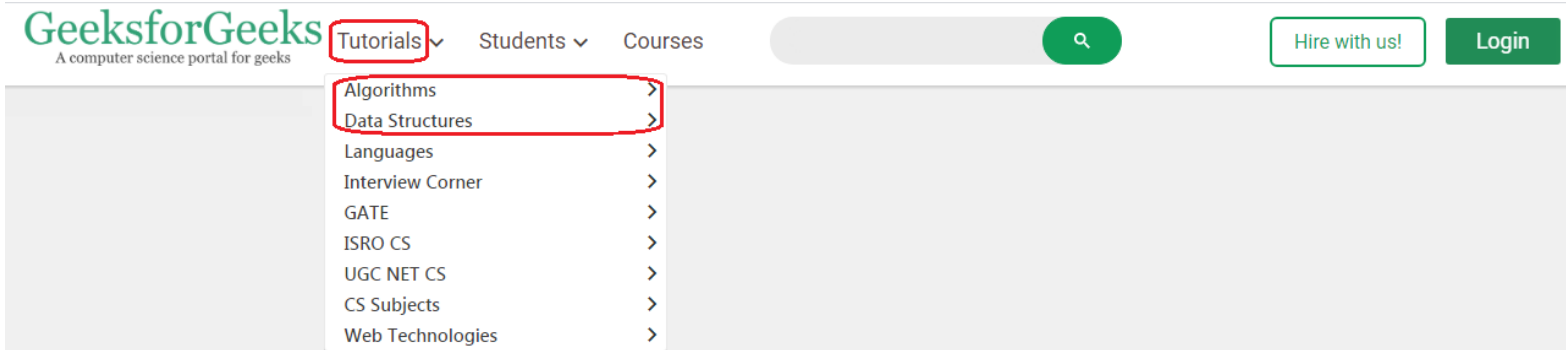


Competitive Programmer's Handbook



A computer science portal for geeks

[geeksforgeeks.org](https://www.geeksforgeeks.org)



# Lecture Agenda

---



- ✓ Section 1: Data Structures and Algorithms Features
- ✓ Section 2: Data Structures and Algorithms Content
- ✓ Section 3: Practice on Online Judges
- ✓ Section 4: Programming Competitions
- ✓ Section 5: Tutorials and References



## Section 6: Online Courses

# Data Structures & Algorithms Specializations



- Accelerated Computer Science Fundamentals Specialization (3 Courses)  
by University of Illinois at Urbana-Champaign [coursera.org/specializations/cs-fundamentals](https://coursera.org/specializations/cs-fundamentals)

## Course : Object-Oriented Data Structures in C++

Week 1: Orientation; Writing a C++ Program  
Week 2: Understanding the C++ Memory Model  
Week 3: Developing C++ Classes  
Week 4: Engineering C++ Software Solutions

## Course : Ordered Data Structures

Week 1: Orientation; Linear Structures  
Week 2: Introduction to Tree Structures  
Week 3: Advanced Tree Structures  
Week 4: Heap Structures

## Course : Unordered Data Structures

Week 1: Orientation; Hashing  
Week 2: Disjoint Sets  
Week 3: Graph Data Structures  
Week 4: Graph Algorithms

# Data Structures & Algorithms Specializations



- Algorithms Specialization (4 Courses) by Stanford University [coursera.org/specializations/algorithms](https://coursera.org/specializations/algorithms)



## Course : Divide and Conquer, Sorting and Searching, and Randomized Algorithms

Week 1: Introduction, big-oh notation and asymptotic analysis

Week 2: Divide and conquer basics, the master method for analyzing  
divide and conquer algorithms

Week 3: The QuickSort algorithm and its analysis, probability review

Week 4: Linear-time selection, graphs, cuts,  
and the contraction algorithm



## Course : Graph Search, Shortest Paths, and Data Structures

Week 1: Breadth-first and depth-first search,  
computing strong components, applications

Week 2: Dijkstra's shortest-path algorithm

Week 3: Heaps, balanced binary search trees

Week 4: Hashing, bloom filters



- Algorithms Specialization (4 Courses) by Stanford University [coursera.org/specializations/algorithms](https://coursera.org/specializations/algorithms)



## Course : Greedy Algorithms, Minimum Spanning Trees, and Dynamic Programming

Week 1: Two motivating applications, selected review, introduction to greedy algorithms, a scheduling application, Prim's MST algorithm

Week 2: Kruskal's MST algorithm and applications to clustering, advanced union-find

Week 3: Huffman codes, introduction to dynamic programming

Week 4: Advanced dynamic programming: the knapsack problem, sequence alignment, and optimal binary search trees



## Course : Simulation, Algorithm Analysis, and Pointers

Week 1: The Bellman-Ford algorithm, all-pairs shortest paths

Week 2: NP-complete problems and exact algorithms for them

Week 3: Approximation algorithms for NP-complete problems

Week 4: Local search algorithms for NP-complete problems, the wider world of algorithms



# Data Structures & Algorithms Specializations



- Data Structures and Algorithms Specialization (6 Courses)  
by University of California San Diego & National Research University Higher School of Economics  
[coursera.org/specializations/data-structures-algorithms](https://coursera.org/specializations/data-structures-algorithms)



## Course : Algorithmic Toolbox

- Week 1: Programming Challenges
- Week 2: Algorithmic Warm-up
- Week 3: Greedy Algorithms
- Week 4: Divide-and-Conquer
- Week 5: Dynamic Programming 1
- Week 6: Dynamic Programming 2



## Course : Data Structures

- Week 1: Basic Data Structures
- Week 2: Dynamic Arrays and Amortized Analysis
- Week 3: Priority Queues and Disjoint Sets
- Week 4: Hash Tables
- Week 5: Binary Search Trees
- Week 6: Binary Search Trees 2

# Data Structures & Algorithms Specializations



- Data Structures and Algorithms Specialization (6 Courses)  
by University of California San Diego & National Research University Higher School of Economics  
[coursera.org/specializations/data-structures-algorithms](https://coursera.org/specializations/data-structures-algorithms)



## Course : Algorithms on Graphs

- Week 1: Decomposition of Graphs 1
- Week 2: Decomposition of Graphs 2
- Week 3: Paths in Graphs 1
- Week 4: Paths in Graphs 2
- Week 5: Minimum Spanning Trees
- Week 6: Advanced Shortest Paths Project



## Course : Algorithms on Strings

- Week 1: Suffix Trees
- Week 2: Burrows-Wheeler Transform and Suffix Arrays
- Week 3: Knuth-Morris-Pratt Algorithm
- Week 4: Constructing Suffix Arrays and Suffix Trees

# Data Structures & Algorithms Specializations



- Data Structures and Algorithms Specialization (6 Courses)  
by University of California San Diego & National Research University Higher School of Economics  
[coursera.org/specializations/data-structures-algorithms](https://coursera.org/specializations/data-structures-algorithms)



## Course : Advanced Algorithms and Complexity

- Week 1: Flows in Networks
- Week 2: Linear Programming
- Week 3: NP-complete Problems
- Week 4: Coping with NP-completeness
- Week 5: Streaming Algorithms



## Course : Genome Assembly Programming Challenge

- Week 1: The 2011 European E. coli Outbreak
- Week 2: Assembling Genomes Using de Bruijn Graphs
- Week 3: Genome Assembly Faces Real Sequencing Data

# Data Structures & Algorithms Specializations



- Algorithms, Part I by Princeton University  
[coursera.org/learn/algorithms-part1](https://coursera.org/learn/algorithms-part1)

- Algorithms, Part II by Princeton University  
[coursera.org/learn/algorithms-part2](https://coursera.org/learn/algorithms-part2)



## Course : Algorithms, Part I

- Week 1: Course Introduction
  - Union-Find
  - Analysis of Algorithms
- Week 2: Stacks and Queues
  - Elementary Sorts
- Week 3: Merge sort
  - Quick sort
- Week 4: Priority Queues
  - Elementary Symbol Tables
- Week 5: Balanced Search Trees
  - Geometric Applications of BSTs
- Week 6: Hash Tables
  - Symbol Table Applications



## Course : Algorithms, Part II

- Week 1: Introduction
  - Undirected Graphs
  - Directed Graphs
- Week 2: Minimum Spanning Trees
  - Shortest Paths
- Week 3: Maximum Flow and Minimum Cut
  - Radix Sorts
- Week 4: Tries
  - Substring Search
- Week 5: Regular Expressions
  - Data Compression
- Week 6: Reductions
  - Linear Programming (optional)
  - Intractability



- Geometric Algorithms by EIT Digital [coursera.org/learn/geometric-algorithms](https://coursera.org/learn/geometric-algorithms)



## Course : Geometric Algorithms

Week 1: Plane Sweep Algorithms

Week 2: Voronoi diagrams and Delaunay triangulations

Week 3: Orthogonal range searching

- Approximation Algorithms by EIT Digital [coursera.org/learn/approximation-algorithms](https://coursera.org/learn/approximation-algorithms)



## Course : Approximation Algorithms

Week 1: Point inclusion in a polygon

Week 2: Convex hulls

Week 3: Intersections

Week 4: Polygon triangulation

Week 5: Orthogonal range search

# Data Structures & Algorithms Specializations



- Analysis of Algorithms by Princeton University  
[coursera.org/learn/analysis-of-algorithms](https://coursera.org/learn/analysis-of-algorithms)



## Course : Analysis of Algorithms

Week 1: Analysis of Algorithms  
Week 2: Recurrences  
Week 3: Generating Functions  
Week 4: Asymptotics  
Week 5: Analytic Combinatorics  
Week 6: Trees  
Week 7: Permutations  
Week 8: Strings and Tries  
Week 9: Words and Mappings

- Computational Geometry  
by Saint Petersburg State University  
[coursera.org/learn/computational-geometry](https://coursera.org/learn/computational-geometry)



## Course : Computational Geometry

Week 1: Point inclusion in a polygon  
Week 2: Convex hulls  
Week 3: Intersections  
Week 4: Polygon triangulation  
Week 5: Orthogonal range search

# Data Structures and Algorithms Playlists



- **Playlist:** Arrays | Data Structures & Algorithms  
[youtube.com/playlist?list=PLqM7aIHxFySEQDk2MDfbwEdjd2svVJH9p](https://youtube.com/playlist?list=PLqM7aIHxFySEQDk2MDfbwEdjd2svVJH9p)  
[100 videos] [5 min] **Channel:** GeeksforGeeks
- **Playlist:** Linked List | Data Structures & Algorithms  
[youtube.com/playlist?list=PLqM7aIHxFySH41ZxznPNj2pAYPOI8ITe7](https://youtube.com/playlist?list=PLqM7aIHxFySH41ZxznPNj2pAYPOI8ITe7)  
[60 videos] [5 min] **Channel:** GeeksforGeeks
- **Playlist:** Stack | Data Structures & Algorithms  
[youtube.com/playlist?list=PLqM7aIHxFySF7Lap-wi5qlaD80EBx9RMV](https://youtube.com/playlist?list=PLqM7aIHxFySF7Lap-wi5qlaD80EBx9RMV)  
[20 videos] [5 min] **Channel:** GeeksforGeeks
- **Playlist:** Queue | Data Structures & Algorithms  
[youtube.com/playlist?list=PLqM7aIHxFySG6wgjVeEat ouTii0IBQ6D](https://youtube.com/playlist?list=PLqM7aIHxFySG6wgjVeEat ouTii0IBQ6D)  
[10 videos] [5 min] **Channel:** GeeksforGeeks
- **Playlist:** Graph | Data Structures & Algorithms  
[youtube.com/playlist?list=PLqM7aIHxFySEaZgcg7uRYJFBnYMLti-nh](https://youtube.com/playlist?list=PLqM7aIHxFySEaZgcg7uRYJFBnYMLti-nh)  
[30 videos] [5 min] **Channel:** GeeksforGeeks
- **Playlist:** Trees | Data Structures & Algorithms  
[youtube.com/playlist?list=PLqM7aIHxFySHCXD7r1J0ky9Zg\\_GBB1dbk](https://youtube.com/playlist?list=PLqM7aIHxFySHCXD7r1J0ky9Zg_GBB1dbk)  
[200 videos] [5 min] **Channel:** GeeksforGeeks
- **Playlist:** Matrix | Data Structures & Algorithms  
[youtube.com/playlist?list=PLqM7aIHxFySGNyLyr8A2CBEbUUEC38f](https://youtube.com/playlist?list=PLqM7aIHxFySGNyLyr8A2CBEbUUEC38f)  
[10 videos] [10 min] **Channel:** GeeksforGeeks
- **Playlist:** Hashing | Data Structures & Algorithms  
[youtube.com/playlist?list=PLqM7aIHxFySGwXaessYMemAnlTqlZdZVE](https://youtube.com/playlist?list=PLqM7aIHxFySGwXaessYMemAnlTqlZdZVE)  
[10 videos] [5 min] **Channel:** GeeksforGeeks

# Data Structures and Algorithms Playlists



- **Playlist:** Data Structures [90 videos] [10 min] **Channel:** RobEdwardsSDSU  
[youtube.com/playlist?list=PLpPXw4zFa0uKKhaSz87lowJnOTzh9tiBk](https://youtube.com/playlist?list=PLpPXw4zFa0uKKhaSz87lowJnOTzh9tiBk)
- **Playlist:** Data Structure(ETCS - 209) - IP University Syllabus [60 videos] [10 min] **Channel:** Easy Engineering Classes  
[youtube.com/playlist?list=PLV8vITYIdSna11Vc54-abg33JtVZiiMfg](https://youtube.com/playlist?list=PLV8vITYIdSna11Vc54-abg33JtVZiiMfg)
- **Playlist:** Data Structures and Algorithms [70 videos] [10 min] **Channel:** Gate Instructors  
[youtube.com/playlist?list=PLXVjII7-2kRkrlwIVmSTF236m3z9sRCr8](https://youtube.com/playlist?list=PLXVjII7-2kRkrlwIVmSTF236m3z9sRCr8)
- **Playlist:** Data Structures [40 videos] [15 min] **Channel:** mycodeschool  
[youtube.com/playlist?list=PL2aWCzGMAwI3WJlcBbtYTwIQSs0Ta6P](https://youtube.com/playlist?list=PL2aWCzGMAwI3WJlcBbtYTwIQSs0Ta6P)
- **Playlist:** Algorithms and Data structures [15 videos] [30 min] **Channel:** Gate Lectures  
[youtube.com/playlist?list=PLEbnTDJUrIeHYw\\_sfBOJ6gk5pie0yP-0](https://youtube.com/playlist?list=PLEbnTDJUrIeHYw_sfBOJ6gk5pie0yP-0)
- **Playlist:** Design and Analysis of Algorithms [55 videos] [15 min] **Channel:** Computer Science and Engineering  
[youtube.com/playlist?list=PLJ5C\\_6qdAvBE5VcLlv1xIFMRpGu3BQneh](https://youtube.com/playlist?list=PLJ5C_6qdAvBE5VcLlv1xIFMRpGu3BQneh)
- **Playlist:** Design and Analysis of Algorithms, Spring 2015 [35 videos] [80 min] **Channel:** MIT OpenCourseWare  
[youtube.com/playlist?list=PLUI4u3cNGP6317WaSNfmCvGymZucw3oGp](https://youtube.com/playlist?list=PLUI4u3cNGP6317WaSNfmCvGymZucw3oGp)
- **Playlist:** Introduction to Algorithms, Fall 2011 [45 videos] [50 min] **Channel:** MIT OpenCourseWare  
[youtube.com/playlist?list=PLUI4u3cNGP610q3tWYp6V\\_F-5jb5L2iHb](https://youtube.com/playlist?list=PLUI4u3cNGP610q3tWYp6V_F-5jb5L2iHb)



# Lecture Agenda

---



- ✓ Section 1: Data Structures and Algorithms Features
- ✓ Section 2: Data Structures and Algorithms Content
- ✓ Section 3: Practice on Online Judges
- ✓ Section 4: Programming Competitions
- ✓ Section 5: Tutorials and References
- ✓ Section 6: Online Courses





DO  
MORE.