ALGORITHMIC THINKING

Presented by Royce Arockiasamy

01

EVENT OVERVIEW

Overview of the event and event guide

02

CONTENT

Overview of all the content in the event

03

RESOURCES

Resources for beginners, intermediates, and advanced competitors

SECTION 01

Event Overview

BRIEF DESCRIPTION

- Competitors are tested on their general knowledge of **algorithms** and ability to develop and **write algorithms in pseudocode** that solves tasks efficiently. **Coding languages can be used** but will not result in the addition of extra points. Pseudocodes can be written in plain english as long as the graders understand the “gist” of what you are trying to say

- Competitors will compete **individually** in this event

- Competitors will take this test with Multiple Choice and Long Response Questions. Long Response Questions involve writing algorithms to solve tasks

- Competitors are allowed to prepare a binder (online or physical) to assist them in this event

- Competitors are graded based on their accuracy on the test. If a tie occurs, the following tiebreakers are used in the order listed:
    - Predetermined tiebreaker questions
    - Fastest time complexity for tasks
    - Time

TEST BREAK UP

50%

MULTIPLE CHOICE

Questions on algorithms, time complexities, etc

50%

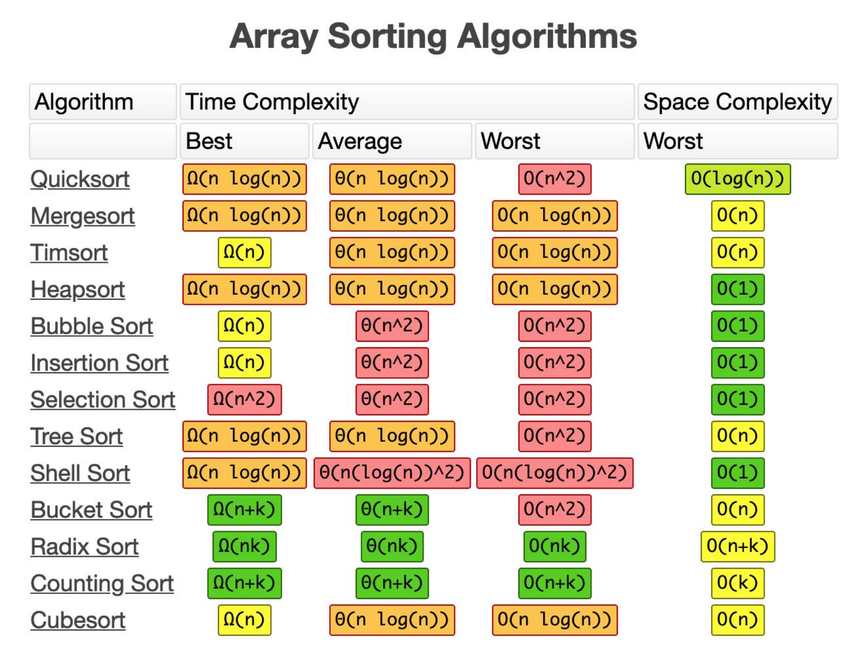
LONG RESPONSE

Solving specific tasks using algorithms

SECTION 02

Content

TIME & SPACE COMPLEXITY NOTATIONS

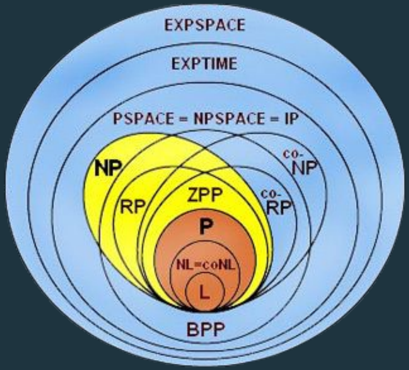


Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)

COMPLEXITY CLASSES

- Complexity classes help group problems based on how much time and space is required to solve problems and verify solutions
  - Having a solid grasp on Big-O notation is necessary
- Which of these might show up on the test?
  - APX
  - EXPSPACE & EXPTIME
  - PSPACE
  - NP
  - RP



ALGORITHMIC REDUCTIONS

- Definition of a Reduction
  - Problem A is reducible, or more technically Turing reducible, to problem B, denoted A ≤ B if there a main program M to solve problem A that lacks only a procedure to solve problem B.
- Assume that A is some new problem that you would like to develop an algorithm for, and that B is some problem that you already know an algorithm for. Then showing A ≤ B will give you an algorithm for problem A.

SEARCH ALGORITHMS

- Linear Search
- Binary Search
- Jump Search
- Interpolation Search
- Exponential Search
- Sublist Search
- Fibonacci Search
- Ubiquitous Binary Search

SORTING ALGORITHMS

- Selection Sort
- Bubble Sort
- Insertion Sort
- Merge Sort
- Radix Sort
- TimSort
- Cycle Sort
- Gnome Sort

PATHFINDING ALGORITHMS

- DFS
- BFS
- Floodfill
- A*
- D*
- Yen’s Algorithm
- Dijkstra
- Minimum Spanning Tree

SECTION 03

Resources

RESOURCES

- USACO Guide
  - Contains algorithms and exercises for specific algorithms. Levels are divided into bronze, silver, gold, and platinum. Platinum is the highest level of difficulty on this site.
- Algorithms by Jeff E
  - Textbook on logic and algorithms. Includes many of the search and sorting algorithms covered in the test.
- Sorting Algorithms - GeeksforGeeks
  - Contains hyperlinks to explanations and code for all of the sorting algorithms covered in the test
- Search Algorithms - GeeksforGeeks
  - Contains hyperlinks to explanations and code for all the search algorithms covered in the test

GOOD LUCK!