Algorithms and Data Structures - 1

Introduction. Big-O notation. Binary Search

Plan

- General information
  - Contacts
  - Course program
  - Importance of that course
  - Sources
  - Game rules
- Introduction. Big-O notation. Binary Search

Contacts

- Islam Alekberov - tg: @artrimbaud
- Ramazan Kozhagulov - tg: @KozhagulovR

Course program - 1

- Unit 1: Introduction. Algorithms vocabulary
    - Introduction. Big-O notation. Master Theorem. Binary Search
    - Linked Lists. Stack implementation using a linked list
- Unit 2: Sorting
    - Sorting. Lower bound for comparisons in the sort. Insertion sort. Bubble Sort. Time complexity & space complexity
    - Quick Sort
    - Merge Sort
    - Binary Heap. Sift Up, Sift Down, Insert, GetMin, ExtractMin, DecreaseKey. Heap Sort.

Course program - 2

- Unit 3: Binary Trees
    - Binary Search Trees. Insert & Delete & BST Sort
    - Balanced Binary search Trees. AVL Tree. Height of AVL Tree on n nodes.
- Unit 4: Hashing
    - Hash Table Chaining. Insert & Delete & Search
    - Hash Table Open Addressing. Insert & Delete & Search
    - Bloom Filter. Insert & Search. Applications. Time complexity & space complexity

Importance of that course

-

Sources

- Thomas H. Cormen. Charles E. Leiserson. Ronald L. Rivest. Clifford Stein. Introduction to Algorithms. Third Edition. The MIT Press.
- google.com

Game rules

- Contests
- Language - C++
- GitHub page
- Code review
- Quizzes after sections

Introduction.

- Problem
- Algorithm
- Correctness
- Efficiency
- Model of computation
- Data structure
- Running time analysis

Big-O notation

- Big-O
- Big-Omega
- Big-Theta

Binary search

- What is Binary Search?
- Why is it important?

Step-by-step Binary Search Algorithm

1. Compare x with the middle element. If x matches with the middle element, we return the mid index.
2. Else If x is greater than the mid element, then x can only lie in the right half subarray after the mid element.
3. So we recur for the right half.
4. Else (x is smaller) recur for the left half.

Binary search example

Find 34 in sorted array:

1. [3, 7, 13, 19, <u>23</u>, 34, 37, 50, 79]
2. 23 < 34
3. [34, 37, <u>50</u>, 79]
4. 34 < 50
5. [34, 37]
6. 34 < 37
7. [34]

Performance of binary search

- Best Case : O(1)
- Worst case: at each step, the remaining search space is divided in half, that's O(log n)