Fundamental Algorithms and Data Structures

Algorithm Questions

- Many technical interview questions are designed to assess your ability to design efficient solutions to problems.
- Often, these questions hinge on your ability to choose the right algorithms and data structures for the job.
- You will often be asked to analyze your solution's time and space complexity.
- CS161 is a *great* way to prepare for these questions. Today, we'll give a brief review.

A Refresher: Big-O Notation

Big-O Notation

- Big-O notation is a way of describing the scalability of an algorithm on large inputs.
- Some common runtimes:
  - O(1): Constant time
  - O(log n): Logarithmic time
  - O(n): Linear time
  - O(n log n): “Linearithmic” time
  - O(n^2): Quadratic time
  - O(n^3): Cubic time
  - O(a^n): Exponential time
  - O(n!): Factorial time (eek!)

Determining Time Complexity

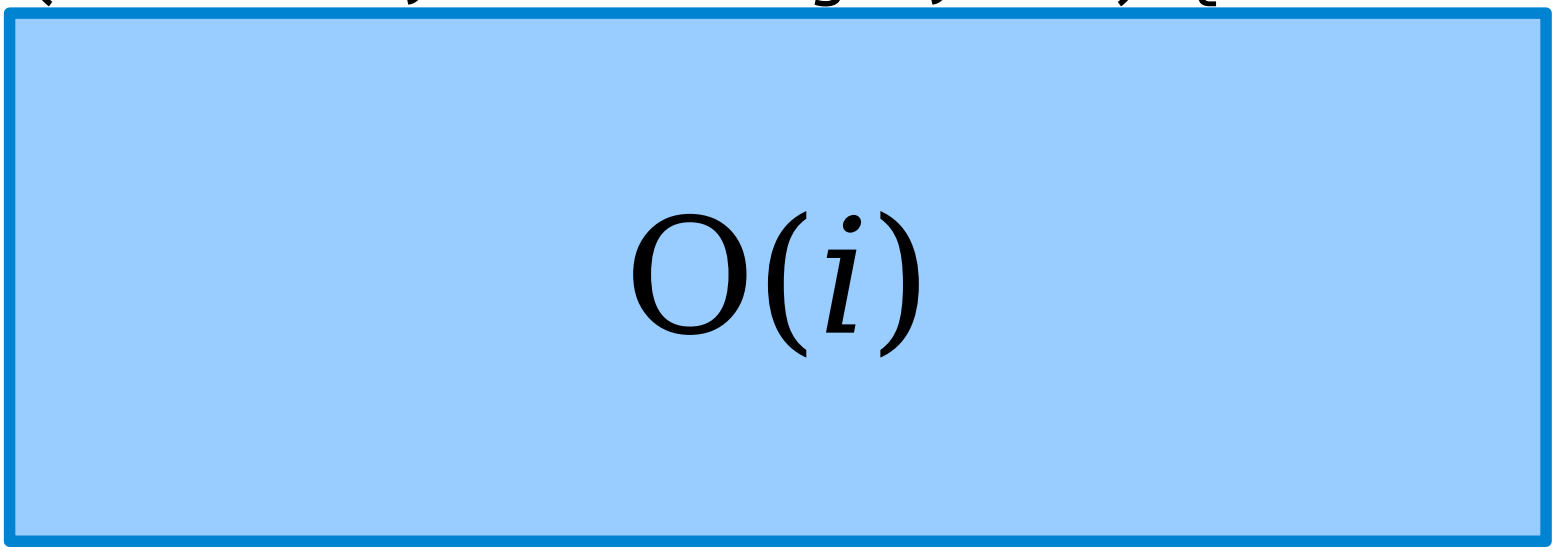
- There are a few standard ways to determine time complexity:
  - Count up how much work the algorithm does on one iteration, then multiply it by the total number of iterations.
  - Find some operation that the algorithm performs, count up how many times it performs it, and multiply by the cost.
- Old CS106B/X materials would be a great way to study for this if you haven't taken CS161.

```
private void selectionSort(int[] v) {
    for (int i = 0; i < v.length; i++) {
        int minIndex = i;
        for (int j = i + 1; j < v.length; j++) {
            if (v[j] < v[minIndex]) {
                minIndex = j;
            }
        }
        swap(v, i, minIndex);
    }
}
```

```
private void selectionSort(int[] v) {
    for (int i = 0; i < v.length; i++) {
        int minIndex = i;
        for (int j = i + 1; j < v.length; j++) {
            if (v[j] < v[minIndex]) {
                minIndex = j;
            }
        }
        swap(v, i, minIndex);
    }
}
```

```
private void selectionSort(int[] v) {
    for (int i = 0; i < v.length; i++) {

    }
}
```



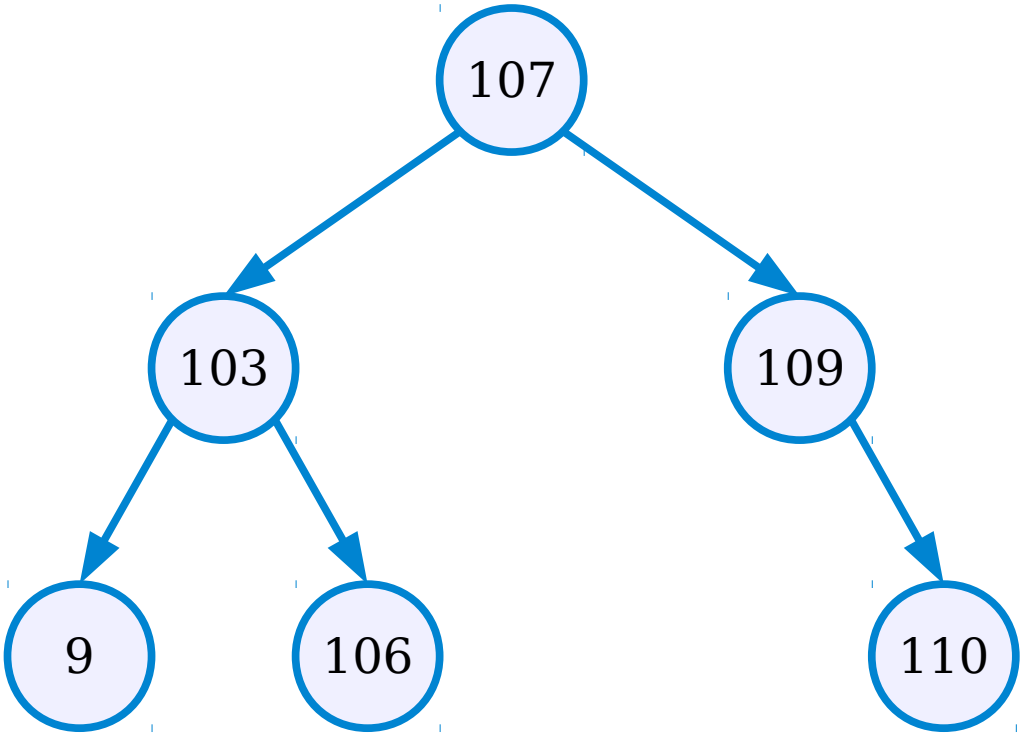
Useful Intuitions

- You'd be amazed how many times these come up:

  $1 + 2 + 3 + 4 + 5 + \ldots + n = \Theta(n^2)$

  $n + n/2 + n/4 + n/8 + \ldots + 1 = \Theta(n)$

- Some quantities and facts are useful to know in a big-O sense:
  - Number of times you can cut something in half before you get down to 1: $\Theta(\log n)$.
  - Number of k-tuples you can form in an array: $\Theta(n^k)$.
  - Number of subsets of n elements: $\Theta(2^n)$.
  - Number of permutations of n elements: $\Theta(n!)$.




```
int countNodesIn(BSTNode* root) {
    if (root == nullptr) return 0;

    return 1 + countNodesIn(root->left)
             + countNodesIn(root->right);
}
```

Fundamental Algorithms and Data Structures

What You Should Know

- You should be familiar with the following data structures and know the big-O runtimes of their main operations:
  - Dynamic array
  - Singly-linked list
  - Doubly-linked list
  - Balanced BST
  - Hash table
  - Trie
  - Stack
  - Queue
  - Binary Heap
  - Bitvector
- You should also know the relative tradeoffs between these structures.

What You Should Know

You should be familiar with the following data structures and know the big-O runtimes of their main operations:

- Dynamic array
- Singly-linked list
- Doubly-linked list
- Balanced BST
- Hash table
- Trie
- Stack
- Queue
- Binary Heap
- Bitvector

You should also know the relative tradeoffs between these structures.

What You Should Know

- You should be familiar with the following algorithms and their runtimes:
  - Binary search
  - Some O(n log n) sort
  - Depth-first search
  - Breadth-first search
  - Dijkstra's algorithm
- You should feel comfortable coding or pseudocoding these algorithms on a board.

Nice Things to Know

- It would be *nice* to know
  - at least one way to implement a hash table.
  - how to perform operations on a BST.
  - at least one extra data structure of your choice. *(Cynthia recommend Bloom filters. Keith recommends order statistic trees.)*
  - how a binary heap works.
  - counting sort and radix sort.
  - at least one fancy algorithm of your choice. *(We recommend Floyd-Warshall.)*

Practice Problems