1

Complexity, Sorting, Searching

CSC 340

April 30, 2016

2

Overview

- Complexity Analysis
- Sorting Algorithms
  - Selection Sort
  - Insertion Sort
- Search Algorithms
- Linked Lists Operations

3

Complexity Analysis

- Focus on loops
  - Assume it takes a constant amount of time to execute primitive operations (e.g. assignment and addition)
  - Identify the total number of times each line in a for loop is executed
  - Total run time is the sum of run time consumed by each line of code
  - The run time is a polynomial function of the upper bounds of the indexing variables

For Loops

- How many times will the statements in the following loops be executed?

```
for( int i = 0; i < n; i++ )
  sum += i;
```

4

4

n times

For Loops

- How many times will the statements in the following loops be executed?

```
for( int i = 0; i < n; i++ )
  for( int j = 0; j < n; j++ )
    sum += i * j;
```

5

5

n squared times

For Loops

- How many times will the statements in the following loops be executed?

```
for( int i = 0; i < n; i++ )
  for( int j = i; j < n; j++ )
    sum += i * j;
```

6

6

(n-1) + (n-2) + … + 2 + 1 + 0

Sn = n(a1 + an)/2 = n(n-1+0)/2

n(n - 1)/2 == O(n^2)

7

Overview

- Complexity Analysis
- Sorting Algorithms
  - Selection Sort
  - Insertion Sort
- Search Algorithms
- Linked List Operations

8

Selection Sort

- Goal is to place the elements of an array in ascending order, in place in the array
- Start at beginning, **select** the next smallest number in remainder of array, then swap
- https://gist.github.com/jrob8577/363da37c19f7594ca0f0

9

Insertion Sort

- Partition the array into two sections: sorted and unsorted
- The sorted partition starts with size 0, and unsorted partition is the entire array
- Insert the next item in the unsorted partition into the correct position in the sorted partition
- https://gist.github.com/jrob8577/7044b898fbf9332550f4

Overview

- Complexity Analysis
- Sorting Algorithms
  - Selection Sort
  - Insertion Sort
- Search Algorithms
- Linked List Operations

10

10

Brute Force Search

- How many operations to find an item in list?
- Is there any way to optimize?

11

11

O(n)

Binary Search

- Assumes we have a sorted list (there's a potential trade off here - what does it cost to sort)
- Compare the item in the middle to the search key
  - If same, item found!
  - If greater, item may be found in lower half of array
  - If smaller, item may be found in upper half of array
- https://gist.github.com/jrob8577/aee060fa98f6f2f68a0d

12

12

Costs nothing if you maintain a list in sorted order…

Overview

- Complexity Analysis
- Sorting Algorithms
  - Selection Sort
  - Insertion Sort
- Search Algorithms
- Linked List Operations

13

13

Linked List Operations

- add - add to beginning of list
- remove - remove item from list
- size - count items in list

14

14

add - O(1) - constant time
remove - O(n) - (Average case - we need to search through all items in list to remove appropriate item)
size - O(n) - we have to count all items