# Data Structures and Algorithms Homework 2

Due Wednesday Sept 11; Joseph Sepich (jps6444)

## 1 Problem 1 Sorted Array

Given a sorted arrayAofn(possibly negative) distinct integers, you want to find out whether there is anindexifor whichA[i] =i. Give a divide-and-conquer algorithm that runs in timeO(logn). Provide only themain idea and the runtime analysis.

There are two very important parts to this problem. The first important part of the problem is the fact tha the array is sorted. The second important part is knowing that you are looking for a integer contained in the array that is the same value as the index. From here we can use a binary search like algorithm. If we look at the middle integer and the statement holds, A[i] == i, then we can return true. If the middle index is larger than the value contained in it, you know that none of the values below that index will work, because the index has to be larger than it. If the middle index is smaller than the value contained in the array at the position, then you know that none of the value above that index will work, because the index would have to be smaller then it. These two conditions also rely on the fact that A only contains integer values and not doubles or floating point values.

Knowing these three conditions you can recursively call one subproblem of size n/2 with constant time at each step eventually reaching a number that fulfills the requirement or a problem with size 1, that doesn't fulfill the requirement where you return false. (Accessing a value at an index is constant time). Using the master theorem we know this running time would be O(log(n)), since a is 1, b is 2, and d is 0.

Collaborators: None

# 2 Problem 2 Squaring vs Multiplying: Matrices

The square of a matrix A is its product with itself, AA.

## 2.1 Part a

Show that five multiplications are sufficient to compute the square of a 2 x 2 matrix.

### 2.2 Part b

### 2.3 Part c

## 3 Problem 3

# 4 Problem 4

# 5 Problem 5

## 5.1 Part a

Run DFS at node A. Run through nodes alphabetically.

List the nodes in the order you visit them.

1. A
2. B
3. D
4. E
5. G
6. F
7. C
8. H
9. I

## 5.2 Part b

List each node with its pre- and post-number. THe numbering starts from 1 and ends at 18.

Using the list from before should make this easy. We know that all intervals should be disjoint or contained within another.

1. A: [1, 12]
2. B: [2, 11]
3. D: [3, 6]
4. E: [4, 5]
5. G: [7, 10]
6. F: [8, 9]
7. C: [13, 18]
8. H: [14, 17]
9. I: [15, 16]

### 5.3 Part c

1. Edge [A, B]: Tree
2. Edge [B, D]: Tree
3. Edge [D, E]: Tree
4. Edge [B, G]: Tree
5. Edge [G, F]: Tree
6. Edge [C, H]: Tree
7. Edge [H, I]: Tree
8. Edge [A, E]: Forward
9. Edge [E, D]: Back
10. Edge [G, D]: Cross
11. Edge [C, I]: Forward

## 5.4 Part d

Big O notation means worst case scenario, so we want to prove that a graph with |V| vertices has at most |V|^2 edges. You could look at a complete graph to prove this claim, since a complete graph will have the most possible edges. A complete graph in this case would mean that an adjacency matrix would have all 1's except in the diagonal, so each vertex would be connected to every other vertex, besides itself. If you total these 1 values up in an n by n matrix, you can get rid of the diagonal, which would have n to have n^2 - n total edges, but half the matrix is also stating the same information (reflexive). So you would also half this value to get $\frac{n^2-n}{2} = \frac{n(n-1)}{2}$ total edges possible in a complete graph. Since a complete graph would be the worst case considering the number of edges this implies $\frac{n(n-1)}{2} = O(n^2)$.

### 5.5 Part e

1. Let's start with the most basic graph, a single node. With this kind of graph you would have 0 edges, so the degree would be zero. The sum of all degrees are even.
2. If you add a second node with any number of edges between it, then you have 2 (the number of nodes) times the number of edges, which would always be an even sum.
3. If you carry this on you would always have an even degree number, because each vertex is connected to a side of each edge. This inherently requires the sum to be even. (Each edge has two vertices connected to it).