Homework 6

Sam Daitzman // DSA Spring 2020

Exercise 1: Academitis

This question builds on problem 2 on Homework 4, in which you designed an algorithm to find a list of all possible patients starting with patient zero.

a) O(n) Time Algorithm

In this question, you will complete the runtime analysis for your algorithm. Suppose that you are given a list of all students at Olin, and for every class at Olin, you are given a list of students in that class. Further, suppose that at most 25 students are in a class and every student takes at most 5 classes (so these are constants that do not depend on the number of students at Olin). Update your algorithm from HW4 to pre-process these class lists so that your algorithm now runs in O(n) time, where n is the number of students at Olin (**Hint**: you may want to use hash maps).

b) Proof by Contradiction

Previously, you argued why every student added to the list was a potential patient and why the returned list contains all such students. This question will formalize that last part using a proof by contradiction. That is, suppose that a student caught academitis but was not added to the list. Prove that this leads to a contradiction.

Exercise 2: Merge Sort Inductive Proof

In class, we argued why Merge Sort always returned a fully sorted list. Formal-ize that argument using a proof by induction. Your proof should contain your inductive hypothesis, a base case, and an induction step.

Exercise 3: Union-Find Variants

Union-Find is an abstract data type and supports the following operations

(a) `make_set(i)`: Creates a set with a single node i.
(b) `union(A,B)`: Merges two sets A and B.
(c) `find(i)`: Given a node i, returns the head element of i's set.

a) Union-Find with Doubly Linked Lists

Explain how to construct a Union-Find DS using doubly linked lists with the following runtimes:

- `make_set`: $O(1)$.
- `union(A,B)`: $O(min(nA, nB))$, where nA is the number of elements in A and nB is the number of elements in B.
- `find(i)`: $O(1)$.

b) Union-Find with Tree Structure

Explain how to construct a Union-Find DS using a tree of elements with the following runtimes:

- `make_set`: $O(1)$.
- `union(A,B)`: $O(1)$.
- `find(i)`: $O(log(n))$.