CS1200: Intro. to Algorithms and their Limitations Anshu & Vadhan

Section 0: Warm Up: Recursion, Binary Trees, and Induction

Harvard SEAS - Fall 2024 *Sept. 6, 2024*

1 Introduction

Welcome to CS 1200! Sections will be held every week of the semester to review lecture material and help you with that week's problem set. Feel free to change which section you go to in different weeks to fit your schedule best.

In this Section 0, we will review the material covered in Lecture 1: Sorting Algorithms and Computational Problems, and get some familiarity with Pset 0.

Before diving into algorithms, we'll review a set of ideas and techniques that will be useful when writing programs and analyzing different algorithms. Some of these concepts may be familiar or unfamiliar; feel free to go through the different problems at the end of each section to get comfortable with these ideas. Don't hesitate to post on Ed if you have any questions or reach out to teaching fellows during office hours!

2 Mathematical notation

1. $\mathbb{Z}$: the integers. $\mathbb{N} = \{0, 1, 2, \ldots\}$: the naturals. $\mathbb{Q}$: the rationals. $\mathbb{R}$: the reals.
2. $\in$: belongs to. For example, $3 \in \mathbb{Z}$.
3. $\forall$: for all. $\exists$: there exists.
4. Iff: if and only if. I.e., both implication directions hold.
5. $\{\ldots\}$: a set. For example, $4 \in \{1, 2, 3, 4, 5\}$.
6. $\cap$: intersection of sets. $\cup$: union of sets.
7. $\emptyset$: empty set.
8. $n!$: n factorial. $n! = 1 \cdot 2 \cdot \ldots \cdot n$.
9. $[a, b]$: the interval from a to b.
10. $[n] = \{0, \ldots, n-1\}$.
11. $A[i]$: i-th item of array A (0-indexing). For example, if $A = (6, 2, 7, 8, 3)$, then $A[3] = 8$.

3 Recursion

Recursion is a technique that allows us to break down a problem into one or more subproblems. This is motivated by the idea that larger problem can use previously solved instances of the same problem to build its solution. A frequent occurrence of recursion is when working with tree data structures; recurrence algorithms can also be used to solve games like Tower of Hanoi or math formulas like Euclidean Algorithm.

Question 3.1. Given a string s, write a recursive algorithm to determine whether s is a palindrome or not.

Definition 3.2 (Tree)**.** A tree is a commonly used data structure to simulate a hierarchical tree structure. It can be defined starting with a root node and then recursively defining its children nodes who are trees themselves. We've defined a tree class below, with three attributes.

```
class Tree:
      children: Tree []
      key: int
      temp: int
```

3.1 Problems

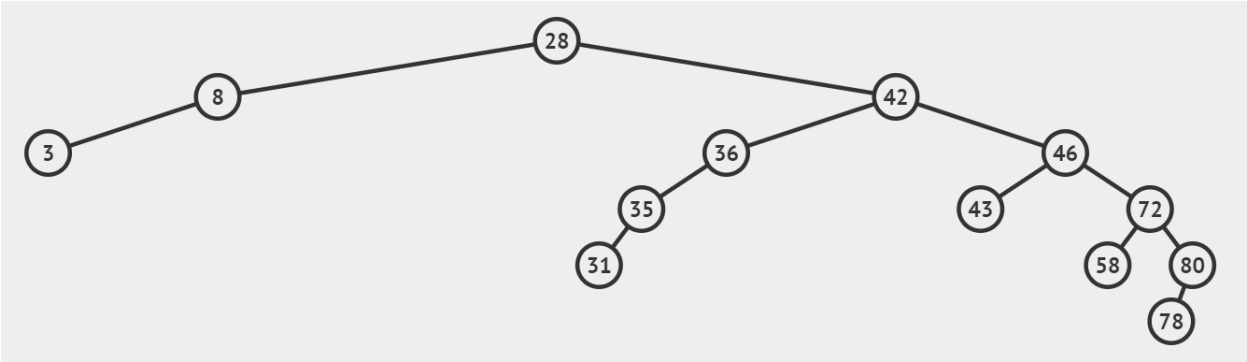
1. Write a function `populateTemp` that takes in a Tree T, and populates the temp attribute of every node in the tree with the number of immediate children the node has.

2. Write a function `elementExists` that takes in a Tree T and integer target, and returns true if there exists a node in T where T.temp = target and false otherwise.

3.2 Binary trees

Question 3.3. As defined in Pset 0, what additional properties does a *Binary* Tree satisfy?

Consider the following Binary Tree `T`:



The number in each node represents the corresponding key.

Question 3.4. Consider the following questions:

a) What is `T.root.key`?

b) What is `T.root.left.key`?

c) What is `T.root.right.key`?

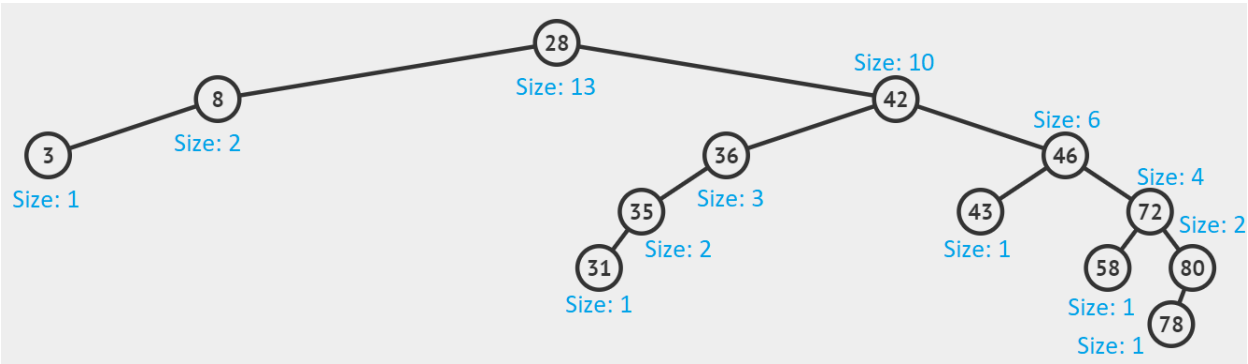
d) What vertices are leaves?

e) Who are the descendants of the vertex with key 72?

f) What is the distance from the vertex with key 42 to the vertex with key 58?

g) What is the size of `T`? What is the height of `T`?

h) Write down the size attribute for each of the vertices in `T`.



Thinking about how you filled out this tree should help you with Problem 1. a) in Pset 1.

4 Induction

Induction is a powerful tool that gives us the ability to prove a general statement by building on top of smaller base cases.

4.1 The Principle of Induction

Let $P(n)$ be the predicate, which we want to prove true for all $n \geq a$ for some integer a. If:

(i) **Base Case:** $P(a)$ is true.

(ii) **Inductive Hypothesis:** Assume that the statement $P(k)$ for $k = n - 1$.

(iii) **Inductive Step:** Prove that the statement $P(k)$ is also true for $k = n$.

Then, by the principle of mathematical induction, $P(n)$ holds true for all $n \geq a$

Tip: For best practices, be sure to explicitly label base case(s) and inductive steps. Don't forget to invoke induction at the end of the proof!

4.2 Strong Induction

Strong induction is a variant of induction, which uses a stronger assumption for the inductive step (ii). Rather than assuming that $P(k)$ is true for just $k = n - 1$, we must assume that $P(k)$ is true for all $k < n$.

In application of recurrences that rely on two or more preceding cases, strong induction can be useful for proofs.

4.3 Example

Problem: Prove that $n! \leq n^n$ for all $n \in \mathbb{N}$

Solution: We will prove that $n! \leq n^n$ for all $n \geq 1$ by mathematical induction.
Base Case: For $n = 1$, $1! = 1$ and $1^1 = 1$. Therefore, $n! \leq n^n$ holds true for n = 1.
Inductive Hypothesis: Suppose for some $n - 1$, $(n-1)! \leq (n-1)^{n-1}$ is true.
Inductive Step: For n:

$$\begin{aligned} n! &= n \cdot (n-1)! \\ &\leq n \cdot (n-1)^{n-1} && \text{(By inductive hypothesis)} \\ &\leq n \cdot n^{n-1} && \text{(Given } n > 0\text{)} \\ &\leq n^n \end{aligned}$$

By mathematical induction, we've proven the statement.

4.4 Exercises

1. Prove that for all $n \geq 1$, $6^n \bmod 5 \equiv 1$ *(Hint: write as a equation in terms of n)*

2. Let F_n be the n-th term of the Fibonacci sequence. Prove that for all $n \geq 1$, $F_1 + F_3 + \cdots + F_{2n-1} = F_{2n}$

3. Using strong induction, prove that all natural numbers can be written as a sum of distinct powers of two, such that $n = 2^{a_1} + 2^{a_2} + ... + 2^{a_k}$, where $0 \leq a_1 < a_2 < ... < a_k$.

4. Prove that for all natural numbers n, a connected graph with n nodes will have at least $n - 1$ edges. *(Hint: start by removing any vertex and proceed with strong induction on the component(s) that remain)*

5. Refer to the MergeSort in Lecture 1. Assume that Merge correctly merges two sorted arrays.

Prove that MergeSort will correctly sort a list of any size. (For this problem, please consult the definition of `Merge` in the readings, as indicated in the Lecture 1 notes.)

Before proving the correctness of MergeSort, we advise that you run through an example.

5 Proof by Contradiction

Question 5.1. Show that $\sqrt{2}$ is irrational.

6 Asymptotic Analysis

Asymptotic notation is often used in computer science to analyze the time complexity of algorithms. Asymptotic notation analyzes how the runtime of the algorithm changes as the input size becomes arbitrarily large.

Definition 6.1 (Big-O)**.** Given functions f and g, $f = O(g)$ if there exists a real number c and N such that $\forall n \geq N$, $f(n) \leq c \cdot g(n)$.

Definition 6.2 (Little-o)**.** Given functions f and g, $f = o(g)$ if for all real numbers $c > 0$ there exists a real number N such that $\forall n \geq N$, $f(n) \leq c \cdot g(n)$.

Another equivalent definition of $f = o(g)$ is $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.

6.1 Example

Problem: Show that $f(n) = 3n^4 + 2n^2 + n$ is in $O(n^4)$.
Solution: It suffices to show that there exists a c and N such that for all $n \geq N$, $f(n) \leq c \cdot n^4$. Let $c = 6$ and $N = 1$. For all $n \geq 1$,

$$n \leq n^4$$

$$2n^2 \leq 2n^4$$

Therefore, $3n^4 + 2n^2 + n \leq 6n^4$ for $n \geq 1$. It follows that $f(n) = O(n^4)$.

6.2 Problems

1. Prove (by example) that there exists a function f_1 such that $f_1(x^2) \neq O(f_1)$.

2. Give a proof or a counterexample that for all functions f_1, $f_1(x^2) \neq O(f_1)$.

3. Given the following functions

   (a) $f_1 = |\sin(x)|$
   (b) $f_2 = |x \sin(x)|$
   (c) $f_3 = x \log(x)$
   (d) $f_4 = x^2$
   (e) $f_6 = x^x$
   (f) $f_7 = 2^x$

   Order the functions by time complexity. For instance, $f_a = O(f_b), f_b = O(f_c), \ldots f_y = O(f_z)$.

7 Programming: Getting Set Up

7.1 Github Repo Instructions

1. Set up Git: https://docs.github.com/en/get-started/quickstart/set-up-gitsetting-up-git.
2. Run the following line in command line:
   `git clone https://github.com/Harvard-CS-120/cs120.git cs120`.

7.2 Python Interpreter Installation

An interpreter is a program that convert user-written code into an intermediate language. A Python interpreter is needed before running Python programs. To check if you already have Python installed and what version, run the following line in command line: `python --version`.

- Windows: https://www.python.org/downloads/.
- MacOS: `brew install python3`.

7.3 Visual Studio Code Set Up

1. Install Visual Studio Code: https://code.visualstudio.com/.
2. From VSCode, install the Python extension from Visual Studio Market Place.
3. From VSCode, open the Command Palette (Ctrl + Shift + P) and begin typing **Python: Select Interpreter**.
4. Select the installed python interpreter (Python 3.x.x).

More detailed instructions can be found here. If you are having issues with local set up, you may use an online IDE (eg. Replit, Google Collab, onlinegdb, etc.).

7.4 Python Learning Resources

- Codeacademy.
- CS50 Lecture on Python.