CSCI 3104 Spring 2022
Instructors: Profs. Chen and Layer

Problem Set 1

Due Date ..... January 25, 2022
Name ..... **Mia Nguyen**
Student ID **109861272**
Collaborators ..... **List Your Collaborators Here**

Contents

1 Instructions

- The solutions **should be typed**, using proper mathematical notation. We cannot accept hand-written solutions. Here's a short intro to LaTeX.
- You should submit your work through the **class Canvas page** only. Please submit one PDF file, compiled using this LaTeX template.

- You may not need a full page for your solutions; pagebreaks are there to help Gradescope automatically find where each problem is. Even if you do not attempt every problem, please submit this document with no fewer pages than the blank template (or Gradescope has issues with it).
- You are welcome and encouraged to collaborate with your classmates, as well as consult outside resources. You must **cite your sources in this document. Copying from any source is an Honor Code violation. Furthermore, all submissions must be in your own words and reflect your understanding of the material.** If there is any confusion about this policy, it is your responsibility to clarify before the due date.
- Posting to **any** service including, but not limited to Chegg, Reddit, StackExchange, etc., for help on an assignment is a violation of the Honor Code.
- You **must** virtually sign the Honor Code (see Section 2). Failure to do so will result in your assignment not being graded.

2 Honor Code (Make Sure to Virtually Sign)

Problem 1.
- My submission is in my own words and reflects my understanding of the material.
- Any collaborations and external sources have been clearly cited in this document.
- I have not posted to external services including, but not limited to Chegg, Reddit, StackExchange, etc.
- I have neither copied nor provided others solutions they can copy.

Agreed (signature here). I agree to the above. Mia Nguyen □

3 Standard 1- Proof by Induction

3.1 Problem 2

Problem 2. A student is trying to prove by induction that $3^n < n!$ for $n \geq 7$.

Student's Proof. The proof is by induction on $n \geq 7$.

- **Base Case:** When $n = 7$, we have that:

$$\begin{aligned} 3^7 &= 2187 \\ &< 5040 \\ &= 7! \end{aligned}$$

- **Inductive Hypothesis:** Now suppose that for all $k \geq 7$ we have that $3^k < k!$.
- **Inductive Step:** We now consider the $k+1$ case. We have that $3^{k+1} < (k+1)!$. It follows that $3^k < k!$. The result follows by induction.

□

There are two errors in this proof.

(a) The Inductive Hypothesis is not correct. Write an explanation to the student explaining why their Inductive Hypothesis is not correct. [**Note:** You are being asked to explain why the Inductive Hypothesis is wrong, and **not** to rewrite a corrected Inductive Hypothesis.]

Answer. The inductive hypothesis is incorrect because it assumes that the statement holds for **all** $k \geq 7$ which assumes the claim is true and uses the claim to prove itself. □

(b) The Inductive Step is not correct. Write an explanation to the student explaining why their Inductive Step is not correct. [**Note:** You are being asked to explain why the Inductive Step is wrong, and **not** to rewrite a corrected Inductive Step.]

Answer. The inductive step is incorrect because the student just kind of restated the inductive hypothesis and did not do any real math or algebra to prove that the inductive hypothesis would hold for the $k+1$ case. □

3.2 Problem 3

Problem 3. Consider the sequence T_n, $n \geq 1$ defined by the following recurrence: $T_1 = T_2 = T_3 = 1$ and $T_n = T_{n-1} + T_{n-2} + T_{n-3}$ for $n \geq 4$.

Prove by induction that $T_n < 2^n$ for all $n \geq 1$.

Proof. I will use strong induction to show that $T_n < 2^n$ for all $n \geq 1$.

Base Case:
I will check to see if $T_n < 2^n$ for all $n \geq 1$ holds true for $n = 1$, $n = 2$, $n = 3$.
when $n = 1$: $T_1 = 1 < 2^1 = 2$
when $n = 2$: $T_2 = 1 < 2^2 = 4$
when $n = 3$: $T_3 = 1 < 2^3 = 8$
$T_n < 2^n$ is true for $n = 1, 2, 3$.

Inductive Hypothesis:
$T_n < 2^n$ for $n = k, k-1, k-2...$

Inductive Step:
If we assume $T_n < 2^n$ is true for $n = 1, 2, 3...k$ then $T_{n+1} = T_k + T_{k-1} + T_{k-2}$

$$\begin{aligned} T_{n+1} &= T_k + T_{k-1} + T_{k-2} \\ &= 2^{k+1}(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} \\ &= 2^{k+1}(\frac{7}{8}) \end{aligned}$$

$$T_{n+1} = T_k + T_{k-1} + T_{k-2} = 2^{k+1}(\frac{7}{8})$$

$$2^{k+1}(\frac{7}{8}) < 2^{k+1}$$

Thus, by strong induction $T_n < 2^n$ is true for all $n \geq 1$

□

3.3 Problem 4

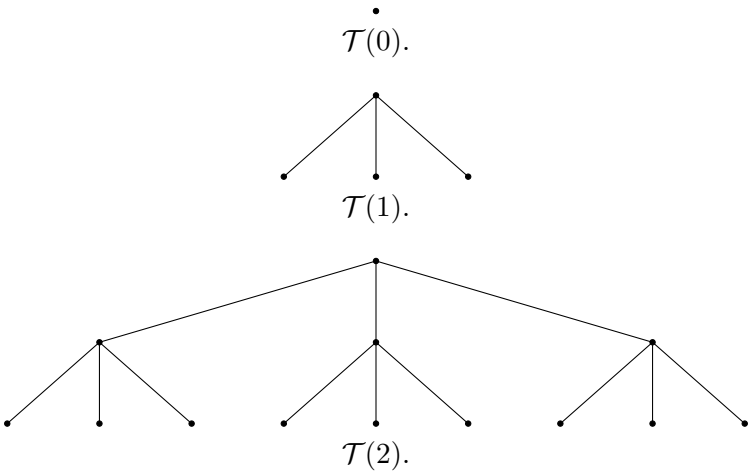
Problem 4. The complete, balanced ternary tree of depth d, denoted $\mathcal{T}(d)$, is defined as follows.

- $\mathcal{T}(0)$ consists of a single vertex.
- For $d > 0$, $\mathcal{T}(d)$ is obtained by starting with a single vertex and setting each of its three children to be copies of $\mathcal{T}(d-1)$.

Prove by induction that $\mathcal{T}(d)$ has 3^d leaf nodes. To help clarify the definition of $\mathcal{T}(d)$, illustrations of $\mathcal{T}(0), \mathcal{T}(1)$, and $\mathcal{T}(2)$ are on the next page. [**Note:** $\mathcal{T}(d)$ is a tree and **not** the number of leaves on the tree. Avoid writing $\mathcal{T}(d) = 3^d$, as these data types are incomparable: a tree is not a number.]

Proof. □

Example 1. We have the following:



$\mathcal{T}(0)$.

$\mathcal{T}(1)$.

$\mathcal{T}(2)$.

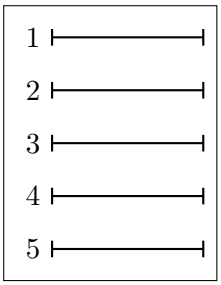
4 Standard 2- Examples Where Greedy Algorithms Fail

4.1 Problem 5

Problem 5. Recall the Interval Scheduling problem, where we take as input a set of intervals $\mathcal{I}$. The goal is to find a maximum-sized set $S \subseteq \mathcal{I}$, where no two intervals in S intersect. Consider the greedy algorithm where we place all of the intervals of $\mathcal{I}$ into a priority queue, ordered earliest start time to latest start time. We then construct a set S by adding intervals to S as we poll them from the priority queue, provided the element we polled does not intersect with any interval already in S.

Provide an example with at least 5 intervals where this algorithm fails to yield a maximum-sized set of pairwise non-overlapping intervals. Clearly specify both the set S that the algorithm constructs, as well a larger set of pairwise non-overlapping intervals.

You may explicitly specify the intervals by their start and end times (such as in the examples from class) or by drawing them. **If you draw them, please make it very clear whether two intervals overlap.** You are welcome to hand-draw and embed an image, provided it is legible and we do not have to rotate our screens to grade your work. Your justification should still be typed. If you would prefer to draw the intervals using LaTeX, we have provided sample code below.



Answer. 5 intervals where the algorithm fails can be:
$[1, 100]$
$[2, 3]$
$[4, 5]$
$[6, 7]$
$[8, 9]$
The set S that the algorithm constructs would just contain [1,100] because it would be ordered first in the queue and none of the other intervals would be added because they overlap with [1,100].
The larger set of pairwise non-overlapping intervals would be [2,3], [4,5], [6,7], [8,9]. □

4.2 Problem 6

Problem 6. Consider now the Weighted Interval Scheduling problem, where each interval i is specified by

$$([\text{start}_i, \text{end}_i], \text{weight}_i).$$

Here, the weight is an assigned value that is independent of the length $\text{end}_i - \text{start}_i$. Here, you may assume $\text{weight}_i > 0$. We seek a set S of pairwise non-overlapping intervals that maximizes $\sum_{i \in S} \text{weight}_i$. That is, rather than maximizing the number of intervals, we are seeking to maximize the sum of the weights.

Consider a greedy algorithm which works identically as in Problem 5. Draw an example with at least 5 appointments where this algorithm fails. Show the order in which the algorithm selects the intervals, and also show a subset with larger weight of non-overlapping intervals than the subset output by the greedy algorithm. The same comments apply here as for Problem 5 in terms of level of explanation.

Answer. 5 appointments where the algorithm fails can be:
$([1, 5], 100)$
$([2, 10], 200)$
$([3, 50], 300)$
$([4, 25], 40)$
$([5, 40], 20)$

Here, the algorithm would just choose the interval with the greatest weight. The set S the algorithm would generate would just contain $([3, 50], 300)$. The subset with the larger weight of non-overlapping intervals would be ([1,5], 100),([2,10], 200),([3,50], 300) and would have a total weight of 600, rather than the total weight of 300 generated by the algorithm.

□

5 Standard 3- Exchange Arguments

5.1 Problem 7

Problem 7. Recall the Making Change problem, where we have an infinite supply of pennies (worth 1 cent), nickels (worth 5 cents), dimes (worth 10 cents), and quarters (worth 25 cents). We take as input an integer $n \geq 0$. The goal is to make change for n using the fewest number of coins possible.

Prove that in an optimal solution, we use at most 2 dimes.

Proof. In an optimal solution, we use at most 2 dimes because 3 dimes can always be replaced with a quarter and a nickel. □

5.2 Problem 8

Problem 8. Consider the Interval Projection problem, which is defined as follows.

- Instance: Let $\mathcal{I}$ be a set of intervals on the real line.
- Solution: A minimum sized set S of points on the real line, such that for every interval $[s, f] \in \mathcal{I}$, there exists a point $x \in S$ where x is in the interval $[s, f]$. We call S a *projection set.*

Do the following.

5.2.1 Problem 8(a)

(a) Find a minimum sized projection set S for the following set of intervals:

$$\mathcal{I} = \{[0, 1], [0.5, 1], [0.9, 1.5], [1.2, 2], [1.7, 2.3]\}.$$

Answer. □

5.2.2 Problem 8(b)

(b) Fix a set of intervals $\mathcal{I}$, and let S be a projection set. Prove that there exists a projection set S' such that (i) $|S'| = |S|$, and (ii) where every point $x \in S'$ is the right end-point of some interval $[s, f] \in \mathcal{I}$.

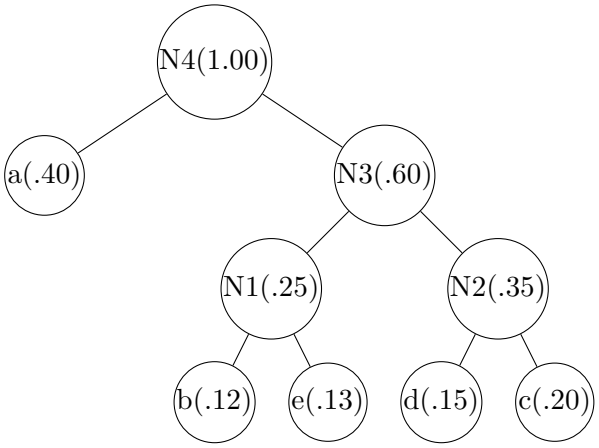
Proof. □

6 Standard 4- Huffman coding

6.1 Problem 9

Problem 9. Given an alphabet of five symbols: a, b, c, d and e, with frequencies 0.4, 0.12, 0.2, 0.15, and 0.13 respectively, work out the Huffman codes for the symbols. You need to first show the optimal binary tree you construct, and then write down the corresponding codes.

Answer. **Huffman Tree:**



Huffman Codes:

a = 0
b = 100
c = 111
d = 110
e = 101

□