CSCI 3104 Spring 2022
Instructors: Profs. Chen and Layer

Problem Set 1

Due Date ..... January 25, 2022
Name ..... **Chengming Li**
Student ID **109251991**
Collaborators ..... **N/A**

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,Chengming Li** □

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. Proved by induction is trying to show the property holds for all $k \geq 7$. Thus, this student cannot assume this property holds for all k. It is pointless when the student already assumed the property holds for all $k \geq 7$. However, this student can assume the property holds for some fixed k number in the inductive hypothesis, and prove the $k + 1$ case in the inductive step to prove the property. □

(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 not correct because the direction of this proof is opposite to what it supposed to be. As, in the inductive step, we should prove the k+1 case based on the Inductive hypothesis. However, in this student's proof, the student is trying to prove the Inductive Hypothesis, as known as the k case, based on the assumption of k+1 case.
In the other words, this student shouldn't assume $3^{k+1} < (k+1)!$, and to prove Inductive Hypothesis,$3^k < k!$, since $3^{k+1} < (k + 1)!$ is what the student need to prove in the Inductive Step and ,$3^k < k!$ is what he could use to finish the proof in the Inductive Step. □

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.

Base Case: When $n = 1$, we have that:

$$\begin{aligned} T_1 &= 1 \\ &< 2 \\ &= 2^1 \end{aligned}$$

When $n = 2$, we have that:

$$\begin{aligned} T_2 &= 1 \\ &< 4 \\ &= 2^2 \end{aligned}$$

When $n = 3$, we have that:

$$\begin{aligned} T_3 &= 1 \\ &< 8 \\ &= 2^3 \end{aligned}$$

When $n = 4$, we have that:

$$\begin{aligned} T_4 &= T_3 + T_2 + T1 \\ &= 1 + 1 + 1 \\ &< 16 \\ &= 2^4 \end{aligned}$$

Inductive Hypothesis: For a given fixed $n \geq 1$, and suppose that $T_k < 2^k$ holds for all $k \leq n$

Inductive Step: We will use inductive hypothesis to show that $T_{k+1} < 2^{k+1}$ is also true, and so to finish the proof. We have:

$$\begin{aligned} T_{k+1} &= T_k + T_{k-1} + T_{k-2} && (1) \\ &\overset{IH}{<} 2^k + 2^{k-1} + 2^{k-2} && (2) \\ &< 2^k + \frac{1}{2} * 2^k + \frac{1}{4} * 2^k && (3) \\ &< (1 + \frac{1}{2} + \frac{1}{4}) * 2^k && (4) \\ &< 1.75 * 2^k && (5) \\ &< 2^{k+1} && (6) \\ &= 2 * 2^k && (7) \end{aligned}$$

Here, proof finished. And the result follows by applying the Inductive Hypothesis at the (2) to obtain $T_{k+1} < 2^{k+1}$. The result follows by induction. □

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.

Base Case: We should consider the base case of $d = 0$. Note from the definition of balanced ternary tree of depth 0, $\mathcal{T}(0)$ consists of a single vertex, or a single leaf node. Thus, $\mathcal{T}(0) has 3^0 = 1$ leaf node. And base case proof finished.

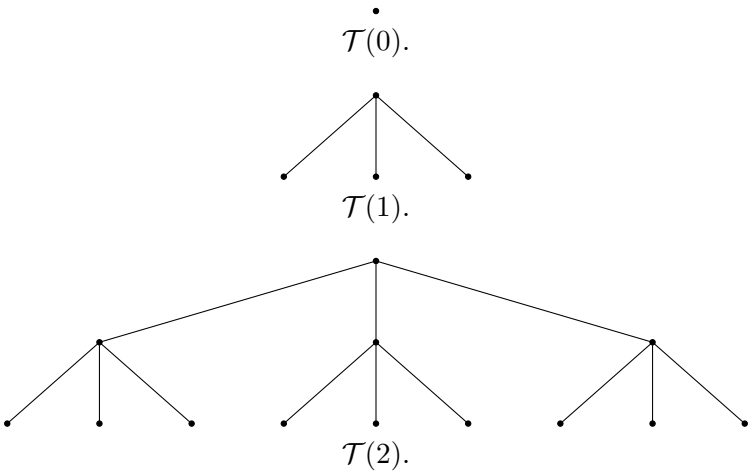
Inductive Hypothesis: For a given fixed $d \geq 0$, and suppose that $\mathcal{T}(d)$ has 3^d leaf nodes.

Inductive Step: We will use the Inductive Hypothesis,as stated above, to show that $\mathcal{T}(d+1)$ **has** 3^{d+1} **leaf nodes.** By the definitions, we can tell $\mathcal{T}(d+1)$ is obtained by starting with a single vertex and setting each of its three children to be copies of $\mathcal{T}(d)$. Thus, **its three children have the property of** $\mathcal{T}(d)$**.** By the **Inductive Hypothesis**, **each child has** 3^d **leaf nodes.** So, the total leaf nodes of $\mathcal{T}(d+1)$ is $3 * 3^d = 3^{d+1}$, since each single vertex has three children.
So, we have $\mathcal{T}(d+1)$ has:

$$\begin{aligned} \mathcal{T}(d+1) &= 3 * 3^d \\ &= 3^{d+1} \end{aligned}$$

The result follows by induction □

Example 1. We have the following:



$\mathcal{T}(0)$.

$\mathcal{T}(1)$.

$\mathcal{T}(2)$.

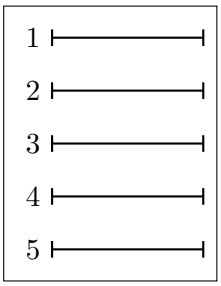
4 Standard 2- Examples Where Greedy Algorithms Fail

4.1 Problem 5

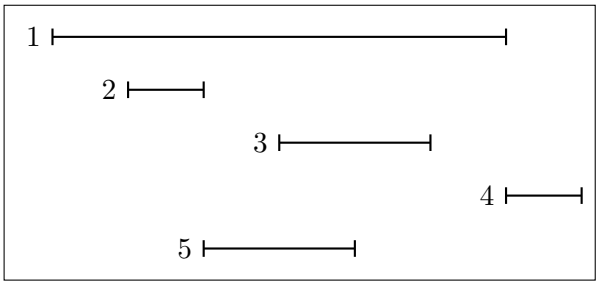
Problem 5. Recall the Interval Scheduling problem, where we take as input a set of intervals $\mathcal{I}$. The goal is to find a maximum-sized set $S \subseteq \mathcal{I}$, where no two intervals in S intersect. Consider the greedy algorithm where we place all of the intervals of $\mathcal{I}$ into 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.



- As the intervals described above, the priority queue Q, ordered earliest start time to latest start time, is [1,2,5,3,4]. And the solution set S is initialized to $\emptyset$.
- And based on this greed algorithm, interval 1 will be polled from the priority queue. Since solution set $S = \emptyset$, the interval 1 will be put into the Solution Set as $S : \{1\}$.
- We next poll 2,5,3,4 from the priority queue in sequence. As they all overlap with 1, we discard 2, 5, 3, 4. So, the solution set S is $S : \{1\}$, which fails to yield a maximum-sized set of pairwise non-overlapping intervals.
- We are supposed to get the maximum-sized set of pairwise non-overlapping intervals as $S : \{2, 3, 4\}$ if the priority queue is ordered earliest end time to latest end time, [2, 5, 3, 1, 4]
- This algorithm fails to yield a maximum-sized set of pairwise non-overlapping intervals due to the wrong sorting method, ordered earliest start time to latest start time, used in the priority queue. And because of this incorrect priority queue Q, this algorithm fails to yield a maximum-sized solution set S.

□

4.2 Problem 6

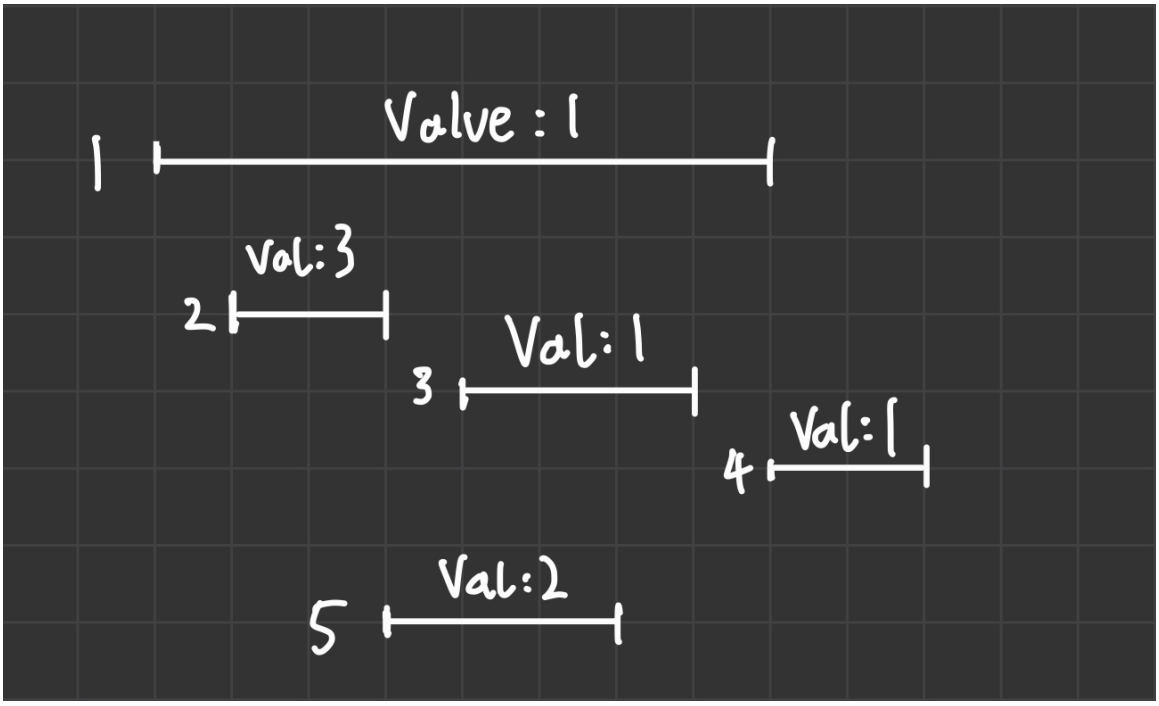
Problem 6. Consider now the Weighted Interval Scheduling problem, where each interval i is specified by

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

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.



- As the intervals described above, the priority queue Q, ordered earliest start time to latest start time, is [1,2,5,3,4]. And the solution set S is initialized to $\emptyset$.
- And based on the greed algorithm described in Q5, interval 1 with weight as 1 will be polled from the priority queue. Since solution set $S = \emptyset$, the interval 1 will be put into the Solution Set as $S : \{1\}$.
- We next poll interval 2 with weight 3, interval 5 with weight 2, interval 3 with weight 1 and interval 4 with weight 1 from the priority queue in sequence. As they all overlap with interval 1, we discard 2, 5, 3, 4. So, the solution set S is $S : \{1\}$ with $weight = 1$, which fails to yield a maximum-weighted set of pairwise non-overlapping intervals.
- We are supposed to get the maximum-weighted set of pairwise non-overlapping intervals as $S : \{2, 3, 4\}$ with $weight = 5$ if the priority queue is ordered earliest end time to latest end time, [2, 5, 3, 1, 4]
- This algorithm fails to yield a maximum-weighted set of pairwise non-overlapping intervals due to the wrong sorting method, ordered earliest start time to latest start time, used in the priority queue. And because of this incorrect priority queue Q, this algorithm fails to yield a maximum-weighted solution subset S.

□

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. I will prove this statement by contradiction and based on the exchange rule of Making Change algorithm.

- Assumption: Assume that in an optimal solution S with $size = |S|$, we use more than 2 dimes. And in the solution S, it contains more than 2 dimes.
- Since the number of dimes is greater than 2, according the rule of Making Change, we may exchange every 2 dimes with another 1 nickels for 1 quarter.
- As stated above, the new size of solution $|S'| = |S| - 2(dimes) - 1(nickle) + 1(quarter) = |S| - 2 < |S|$, or $|S'| = |S| - 3(dimes) + 1(quarter) + 1(nickle) = |S| - 1 < |S|$, because we can exchange every two dimes with another 1 nickels for a quarter . We observe that the new solution S' use fewer numbers of coins than the optimal solution S in the assumption, which contradicts the assumption I made above.
- Thus, in an optimal solution, we couldn't use more than 2 dimes. In the other words, in an optimal solution, we use at most 2 dimes.
- Example, assume n = 47 cents. Based on the assumption I made above, we use more than 2 dime to make change, then we get $47 = 10 + 10 + 10 + 10 + 5 + 1 + 1$. However, based on the rule of exchange, choosing as many quarter as possible, etc, we get a new combination of change 47 cent $= 25 + 10 + 10 + 1 + 1$. Apparently, the second combination(5 coins) is more optimal than the first combination(7 coins). And the second solution satisfy the property that use at most 2 dimes to get an optimal solution.

Proof finished, In an optimal solution, we use at most 2 dimes. □

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 (i) 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*.

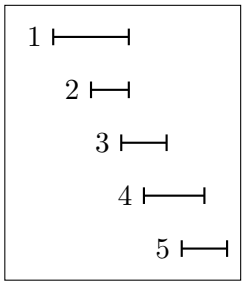
Do the following.

5.2.1 Problem 7(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.



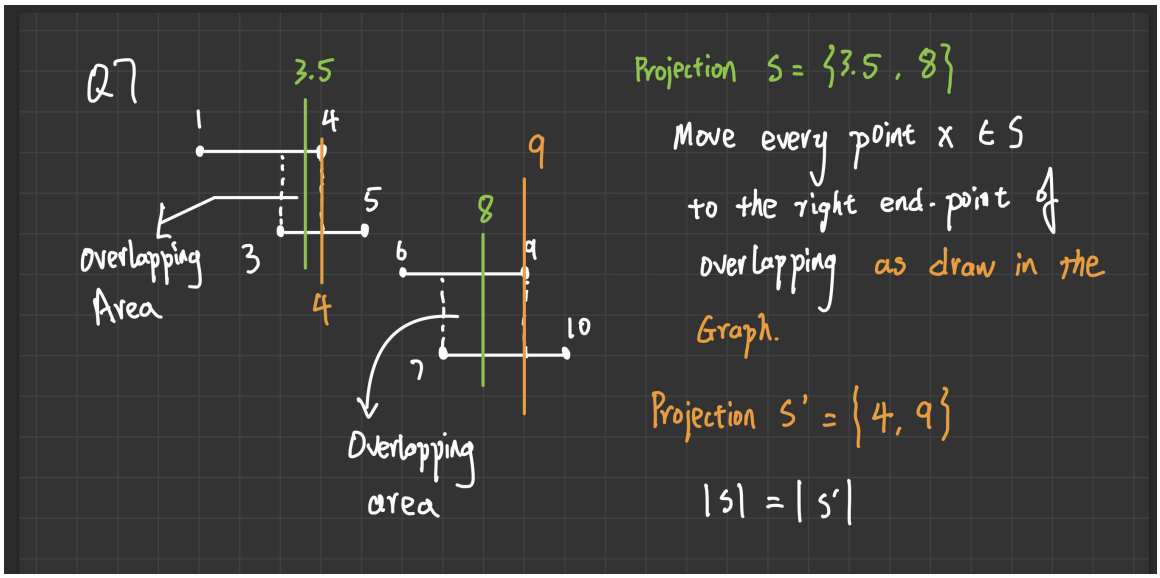
$S = \{1, 2\}$
Point 1 will cover the first 3 intervals
Point 2 will cover the last 2 intervals. □

5.2.2 Problem 7(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.

- We know the Project set S exists for a fixed set of intervals $\mathcal{I}$, which has cardinality $|S|$. such that for every interval $[s, f] \in \mathcal{I}$, there exists a point $x \in S$ where x is in the interval $[s, f]$. And every point is in the overlapping area of some interval.
- Then, we can create a new projection set S' based on moving every point in their overlapping areas. But, there are two cases we need to consider: 1. the point is already the right-end point of some interval if the overlapping areas of some interval is a single point. 2. The point exists in an overlapping areas of some interval, like midpoint of an overlapping areas.
- Considering the first case, we don't have to move that point to the right end-point of some interval $[s, f] \in \mathcal{I}$, because it already was.
- Considering the second case, we could move that point to the right end-point of some interval as long as that point still stay in the overlapping area of some interval $[s, f] \in \mathcal{I}$
- (i) Since we didn't add or delete any points in the projection set S when we move every point in the projection set, we could assume the cardinality$|S'| = |S|$.
- (ii) Moreover, as the second case stated above, every point $x \in S'$ is the right end-point of some interval $[s, f] \in \mathcal{I}$ as long as we move every point to the right end-point of the overlapping interval, which correspond to the right end-point of some interval $[s, f] \in \mathcal{I}$, and every point still stay in the overlapping areas.
- The movement of each point still follow the rule of $Projectionset S$ as long as every point $x \in S$ where x is in the interval $[s, f]$
- Then, the new projection set S' can be made by the movement of every point $x \in S$. And, new projection set S' satisfy (i) $|S'| = |S|$, and (ii) where every point $x \in S'$ is the right end-point of some interval $[s, f] \in \mathcal{I}$.
- Example can be seen below.



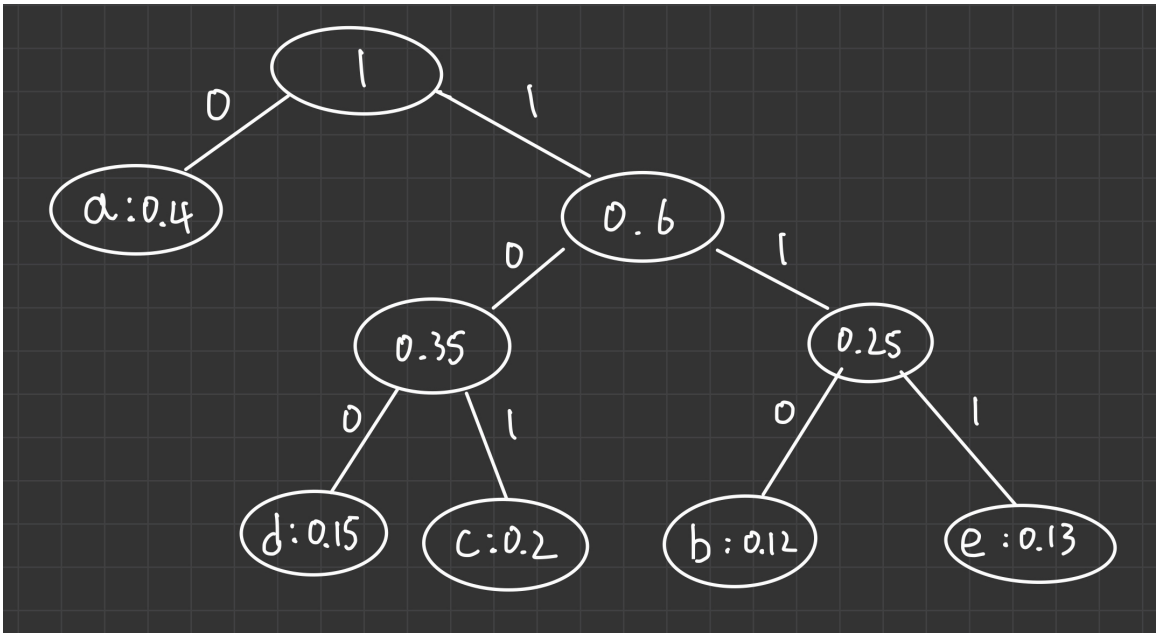
□

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.



- a: 0
- b: 110
- c: 101
- d: 100
- e: 111

□