This assignment is **due on October 29** and should be submitted on Gradescope. All submitted work must be *done individually* without consulting someone else's solutions in accordance with the University's "Academic Dishonesty and Plagiarism" policies.

As a first step go to the last page and read the section: Advice on how to do the assignment.

Problem 1. (15 points) We are participating in a running event for charity and we are being sponsored for every kilometre we run. The organisation of the event has decided to give every runner the freedom to pick their own course within a given area of the city, as long as they start from a given starting position s. This is given to us as a simple connected undirected weighted graph $G = (V, E)$, where all weights w are distinct positive integers. Unfortunately, we get bored easily, so we want to visit every location (vertex) at most once. To help the charity the most, we aim to find the longest running course in G that starts from s and visits every vertex at most once.

We are given two algorithms to compute the longest running course. Algorithm PICKHIGHESTDEGREE starts from s and repeatedly follows the edge to the unvisited vertex of highest degree (if multiple vertices have the same degree, it picks the one that is reached by following the longest edge). Algorithm PICKLONGESTEDGE starts from s and repeatedly follows the longest edge to an unvisited vertex.

```
1:  function PickHighestDegree(G, w, s)
2:      result ← 0
3:      course ← ∅
4:      current ← s
5:      while current has an unvisited neighbour do
6:          next ← neighbour of current of highest degree (longest edge as tie-
    breaker)
7:          Append next to course
8:          result ← result + w(current, next)
9:          current ← next
10:     return result, course
```

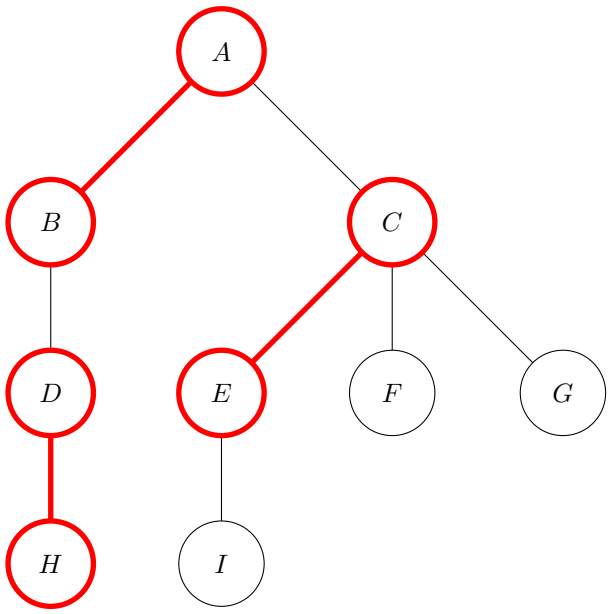
```
1:  function PickLongestEdge(G, w, s)
2:      result ← 0
3:      course ← ∅
4:      current ← s
5:      while current has an unvisited neighbour do
6:          next ← neighbour of current that maximizes w(current, next)
7:          Append next to course
8:          result ← result + w(current, next)
9:          current ← next
10:     return result, course
```

a) Show that PICKHIGHESTDEGREE doesn't always return the longest course by giving a counterexample.

b) Argue whether PICKLONGESTEDGE always returns the longest course by either arguing its correctness (if you think it does) or by providing a counterexample (if you think it doesn't).

Problem 2. (30 points) Given a tree T, an *upward path* is defined as a path where every edge is traversed from child to parent. An upward path has length k if it consists of exactly k edges.

Example:



In the example above, $k = 1$. There are three upward paths shown in red: $[B, A]$, $[E, C]$, and $[H, D]$. Note that this isn't the largest set of upward paths of length 1 possible in this tree.

Your task is to design an algorithm that takes a tree T and an integer k as input and returns the largest set of upward paths of length k in T such that no two paths share a vertex. For full marks your algorithm needs to run in $O(n \log n)$ time.

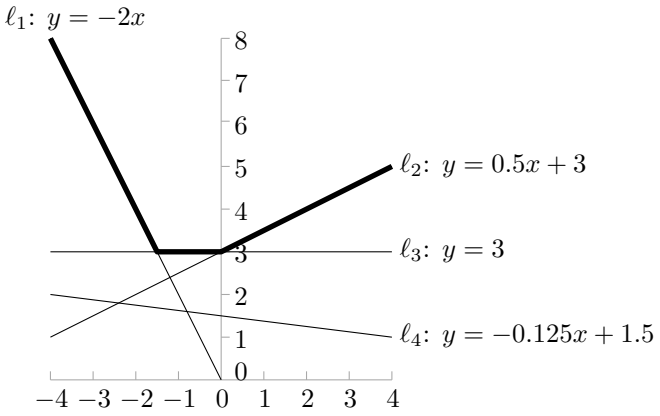
a) Design an algorithm that solves the problem.

b) Briefly argue the correctness of your algorithm.

c) Analyse the running time of your algorithm.

Problem 3. (30 points) The innovative company Fishbowls-R-Us allows their customers to design their own fishbowls by providing n lines in 2D, representing the slopes of different sections of the bottom half of the fishbowl. Each line is given to you as an equation $y = ax + b$. Unfortunately, their 3D printers don't take lines as input, so you are asked to convert the lines into a single shape: the fishbowl. Intuitively speaking, if we take the union of all lines and pour water from $(0, \infty)$ the fishbowl is formed by the line segments that touch the water. More formally, the fishbowl consists of the highest line segment for any given x-coordinate. Note that this implies that not all lines are necessarily part of the fishbowl, for example, if a customer specifies a line that is always below some other line. For simplicity, you can assume that we're interested in the fishbowl in the printing range $[left, right]$ and that no two lines have the same y-coordinate at these boundaries.

Example:



In the example above, there are four lines. Assuming the printing range is $[-4, 4]$, the fishbowl is formed by the three thick line segments and can be output using the intersection points of the lines and the left and right printing boundary: $[(-4, 8), (-1.5, 3), (0, 3), (4, 5)]$. Line ℓ_4 is not part of the fishbowl.

Your task is to design a divide and conquer algorithm that returns the fishbowl of n lines given in an array. For full marks your algorithm needs to run in $O(n \log n)$ time.

a) Design an algorithm that solves the problem.

b) Briefly argue the correctness of your algorithm.

c) Analyse the running time of your algorithm.

Advice on how to do the assignment

- Assignments should be typed and submitted as pdf (no handwriting).
- Start by typing your student ID at the top of the first page of your submission. Do **not** type your name.
- Submit only your answers to the questions. Do **not** copy the questions.
- When designing an algorithm or data structure, it might help you (and us) if you briefly describe your general idea, and after that you might want to develop and elaborate on details. If we don't see/understand your general idea, we cannot give you points for it.
- Be careful with giving multiple or alternative answers. If you give multiple answers, then we will give you marks only for "your worst answer", as this indicates how well you understood the question.
- Some of the questions are very easy (with the help of the lecture notes or book). You can use the material presented in the lecture or book without proving it. You do not need to write more than necessary (see comment above).
- When giving answers to questions, always prove/explain/motivate your answers.
- When giving an algorithm as an answer, the algorithm does not have to be given as (pseudo-)code.
- If you do give (pseudo-)code, then you still have to explain your code and your ideas in plain English.
- Unless otherwise stated, we always ask about worst-case analysis, worst case running times etc.
- As done in the lecture, and as it is typical for an algorithms course, we are interested in the most efficient algorithms and data structures.
- If you use further resources (books, scientific papers, the internet,...) to formulate your answers, then add references to your sources.
- If you refer to a result in a scientific paper or on the web you need to explain the results to show that you understand the results, and how it was proven.