Elementary Maths for GMT - 2019 - Homework Exam 2

This homework exam consists of 2 questions. You can earn a total of 100 points. You may discuss questions and approaches on a high level with your fellow students, but must write down your own solutions. The first 10 points can be obtained by following these rules:

- Hand in your solutions on paper.
- Write clearly and legibly, or use a type setting system (such as \LaTeX).
- Write your name and student number on each sheet you hand in, and use page numbers.

Make sure you understand the questions, and be concise in your answers. An explanation of how you obtained your answer is not mandatory unless specified in the question; however, a short derivation may be helpful in case of arithmetic mistakes.

**Question 1**

One annoying problem I often have is that the various cables of my various electronic devices automatically end up, after a while, in a knot. This is especially annoying for cables which are closed loops, because then it is sometimes impossible to undo these knots.

A knot is a closed loop in $\mathbb{R}^3$; that is, it is a continuous injective function from $S^1$ to $\mathbb{R}^3$. A diagram of a knot is the projection of a knot to $\mathbb{R}^2$. We can see a diagram as a drawing of the knot where each crossing is annotated with the information which strand of the knot passes over and which passes under. For example, the following drawing is a knot diagram:

Two knots diagrams are equivalent if they are drawings of the same knot; that is, if there exists a knot which can be rearranged in space and then projected to both diagrams. Equivalent diagrams can be transformed to each other using Reidemeister moves:

\[
\begin{align*}
(1) & \quad \equiv \\
(2) & \quad \equiv \\
(3) & \quad \equiv 
\end{align*}
\]

(a) Draw a sequence of knot diagrams, making one Reidemeister move between diagrams, to show the knot drawn above is equivalent to the unknot: a loop without any crossings.

(b) Draw a knot diagram which is equivalent to the unknot, where the only possible Reidemeister moves are of type (3).

An alternating knot diagram is a knot diagram which has the property that when we traverse the cycle, the crossings are alternatingly going over and under each other. For example:

(c) Is is possible that an alternating knot diagram is equivalent to the unknot? If yes, draw an example; if no, give a proof.

(d) Given a closed curve in $\mathbb{R}^2$, is it always possible to assign over/under information so that it becomes an alternating knot diagram? If no, draw an example; if yes, give a proof.
Question 2

I like to climb trees, but I don’t like my neighbour, who also likes to climb trees. Therefore, I want to build a fence between our gardens which blocks visibility from any tree in my garden to any tree in my neighbour’s garden. I do not like to spend more money than necessary, so I want the fence to be as low as possible. Also, I live in $\mathbb{R}^2$.

I model my problem as follows. Imagine an axis system such that the $x$-axis is the ground, and the $y$-axis is the border between my garden and my neighbour’s garden. Let $P \subset \mathbb{R}^2$ be a set of points which represent the tops of the trees; all trees have positive height and do not grow exactly at the border, that is, if $(x, y) \in P$ then $x \neq 0$ and $y > 0$.

For a pair of points $p, q \in P$, let $h(p, q)$ be the $y$-coordinate of the intersection between the line through $p$ and $q$ and the $y$-axis. Then, in order to block visibility between points $p$ and $q$ on opposite sides of the border, the fence needs to have at least height $h(p, q)$.

(a) Give pseudocode for an algorithm which computes $h(p, q)$.

(b) Give pseudocode for an algorithm below which takes as input three points $p, q,$ and $r$, and which tests whether $p$ lies on or below the line through $q$ and $r$.

Now, in order to find the optimal height of the fence, I just need to find the pair of trees which requires the highest fence. Consider the following algorithms.

```plaintext
function fence
input: $P \subset \mathbb{R}^2$: a set of points
output: ($l, r$): a pair of points
$L = R = \emptyset$;
$l = r = (0, 0)$;
for $p \in P$ do
    if not below ($p, l, r$) then
        if $p.x < 0$ then $l = p$; $r = \text{scan}(l, R)$;
        else $r = p$; $l = \text{scan}(r, L)$;
    end
    if $p.x < 0$ then $L = L \cup \{p\}$;
    else $R = R \cup \{p\}$;
end
return ($l, r$);
```

```plaintext
function scan
input: $p \in \mathbb{R}^2$: a point; $Q \subset \mathbb{R}^2$: a set of points
output: $r$: a point
$r = (0, 0)$;
$m = 0$;
for $q \in Q$ do
    if $h(p, q) > y$ then
        $m = h(p, q)$;
        $r = q$;
    end
end
return $r$;
```

(c) Execute $\text{fence}$ on the point set $P = \{(-3, 1), (2, 1), (-1, 2), (1, 4)\}$. Draw the intermediate states after each step of the for loop.

Now, let $P$ be a general set of $n$ points.

(d) What is the worst-case time complexity of $\text{scan}$, as a function of $n$ using $O()$ notation?

(e) What is the worst-case time complexity of $\text{fence}$?

(f) What is the best-case time complexity of $\text{fence}$?

The running time of $\text{fence}$ does not only depend on the input set, but also on the order in which the input elements are. Sometimes the worst-case running time can be avoided by randomizing the input order.

(g) Assume the order of points in $P$ is random. What is the expected running time of $\text{scan}$?

To calculate the expected running time of $\text{fence}$, we need to know the probability that we execute $\text{scan}$. A powerful technique for this is known as backwards analysis.

(h) Consider the $i$th pass through the for-loop of $\text{fence}$. What is the probability that, after this pass, either $l$ or $r$ was just computed using $\text{scan}$?

(i) What is the expected running time of $\text{fence}$?