Elementary Maths for GMT - 2016 - Homework Exam 3
SOLUTIONS

This homework exam consists of 4 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.

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 [20 points]
Consider the following impartial two-player game. On the table, there are three gold coins and three silver coins. On their turn, a player can choose to do one of the following things:

- Take a single silver coin.
- Take a gold coin and put down 3 silver coins (from an infinite bank).

Whoever takes the last coin wins the game.

(a) Assuming both players play optimally, who wins this game?
(b) Give a general formula for who wins the game starting with \(x\) silver and \(y\) gold coins.

This game is boring, nobody wants to play it. Let’s add a third possible move:

- Take 4 silver coins and put down a gold coin (from an infinite bank).

(c) Assuming both players play optimally, who wins the game now?

Solution

(a) From any state \((x, y)\) we can reach the states \((x - 1, y)\) and \((x + 3, y - 1)\), assuming the resulting numbers are non-negative. We simply fill in a table of possible game states, and see that the state \((3, 3)\) evaluates to true: the first player wins.

(b) The table suggest a pattern: \(w(x, y)\) is true if and only if \(x\) is odd. We can prove this by checking the formula:

\[ w(x, y) = \neg w(x - 1, y) \lor \neg w(x + 3, y - 1). \]

(c) Now, from any state \((x, y)\) we can reach three states: \((x - 1, y)\), \((x + 3, y - 1)\) and \((x - 4, y + 1)\) (still assuming the resulting numbers are non-negative). We fill in a table of possible game states again, and see that this time, the state \((3, 3)\) evaluates to false: the second player wins.
**Question 2**

Consider the following transformation of \( \mathbb{R}^2 \):

- a rotation by 45° around the point \((2, 2)\); followed by
- an inversion in the circle of radius 1 centered at the point \((1, 1)\); followed by
- a reflection in the line through the points \((2, 2)\) and \((2 + \frac{2}{3}\sqrt{3}, 0)\).

(a) Apply this transformation to the picture and sketch the result.

(b) What are the precise coordinates of the resulting concave vertex?

(c) Is this transformation rigid? Linear? Affine? Möbius?

(d) Give a formula for this transformation.

**Solution**

(a) See picture.

(b) The concave vertex starts at \((2, 2)\). We can observe how it changes through each transformation.

- \((2, 2) \mapsto (2, 2)\)
- \((2, 2) \mapsto (1\frac{1}{2}, 1\frac{1}{2})\)
- \((1\frac{1}{2}, 1\frac{1}{2}) \mapsto (2\frac{1}{4} + \frac{1}{4}\sqrt{3}, 1\frac{3}{4} + \frac{1}{4}\sqrt{3})\)

The coordinates are \((2\frac{1}{4} + \frac{1}{4}\sqrt{3}, 1\frac{3}{4} + \frac{1}{4}\sqrt{3})\).

(c) This is a Möbius transformation.

(d) We first find formulas for the individual steps of the transformation:

- \((x, y) \mapsto (2 + \sqrt{2}x - \sqrt{2}y, 2 - 4\sqrt{2} + \sqrt{2}x + \sqrt{2}y)\)
- \((x, y) \mapsto (1 + \frac{x - 2x + y^2 - 2y^2 + 1}{x^2 - 2x + y^2 - 2y^2 + 2}, 1 + \frac{y - 1}{x^2 - 2x + y^2 - 2y^2 + 2})\)
- \((x, y) \mapsto (3 + \sqrt{3} - \frac{1}{2}x - \frac{1}{2}\sqrt{3}y, 1 + \sqrt{3} - \frac{1}{2}\sqrt{3}x + \frac{1}{2}y)\)

**Question 3**

Three kids are in different houses and can choose to either stay inside and read a book, or go outside to play on the seesaw. If a kid chooses to stay inside, they receive 2 units of happiness, independent of what the other kids do. If a kid chooses to go outside but they are the only one, they cannot play and become very sad. They receive 0 units of happiness. If two kids choose to go outside, they can play together and they both receive 3 units of happiness. If all three kids choose to go outside, they have to take turns and receive only 1 unit of happiness each.

(a) Model the situation above as a three-player simultaneous game.

(b) Does this game have any pure equilibria? List them all.

(c) Does this game have any mixed equilibria? List them all.

**Solution**

(a) We can make a 2 by 2 by 2 table to model this game.

<table>
<thead>
<tr>
<th></th>
<th>P2</th>
<th>P3</th>
<th>payoff</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>I</td>
<td>I</td>
<td>(2, 2, 2)</td>
</tr>
<tr>
<td>I</td>
<td>I</td>
<td>O</td>
<td>(2, 2, 0)</td>
</tr>
<tr>
<td>I</td>
<td>O</td>
<td>I</td>
<td>(2, 0, 2)</td>
</tr>
<tr>
<td>I</td>
<td>O</td>
<td>O</td>
<td>(2, 3, 3)</td>
</tr>
<tr>
<td>O</td>
<td>I</td>
<td>I</td>
<td>(0, 2, 2)</td>
</tr>
<tr>
<td>O</td>
<td>I</td>
<td>O</td>
<td>(3, 2, 3)</td>
</tr>
<tr>
<td>O</td>
<td>O</td>
<td>I</td>
<td>(3, 3, 2)</td>
</tr>
<tr>
<td>O</td>
<td>O</td>
<td>O</td>
<td>(1, 1, 1)</td>
</tr>
</tbody>
</table>
(b) There are four equilibria: (I,I,I), (I,O,O), (O,I,O) and (O,O,I).

(c) Expected payoff for staying inside is 2 (since it does not depend on the choices of the others). Expected payoff for going out is \( p^2 \cdot 0 + p(1 - p) \cdot 3 + p(1 - p) \cdot 3 + (1 - p)^2 \cdot 1 = -5p^2 + 4p + 1 \). If we are in equilibrium, these payoffs must be equal. That is, 
\[
5p^2 - 4p + 1 = 0
\]
This equation has no real-valued solutions, so there is no mixed equilibrium.

**Question 4**

A kD-tree is a data structure that stores points in \( \mathbb{R}^k \). It can be built by the following algorithm:

```
function plantTree
input: P: a set of points in \( \mathbb{R}^k \), B: a bounding box of P, d: a number between 1 and k
if |P| = 1, P = \{p\} then
    new leaf v;
    v.point = p;
    v(bbox = B);
    return v;
end
else
    x = median (P, d);
    \( P_1 = \{p \in P \mid p_d < x\} \);
    \( P_2 = \{p \in P \mid p_d > x\} \);
    new node v;
    v.leftchild = plantTree (P_1, B \cap \{p \in \mathbb{R}^k \mid p_d \leq x\}, d \mod k + 1);
    v.rightchild = plantTree (P_2, B \cap \{p \in \mathbb{R}^k \mid p_d \geq x\}, d \mod k + 1);
    v.count = |P|;
    v.bbox = B;
    return v;
end
```

Here the function “median” takes a set of \( n \) points and an index \( d \), and returns a number \( x \) such that \( \floor{\frac{n}{2}} \) points are smaller than \( x \) in the \( d \)th dimension, and \( \ceil{\frac{n}{2}} \) points are larger than \( x \) in the \( d \)th dimension. It is possible to implement “median” to run in linear time.

(a) Draw the result of the construction algorithm on the following point set.

(b) Analyse the worst-case running time of this construction algorithm.

kD-trees can be used to perform so-called range counting queries, which return how many points are contained in a given range.

```
function rangeCount
input: v: the root of a kD-tree, R: a k-dimensional query range
if v is a leaf then
    return if v.point \in R then 1 else 0;
end
else if v.bbox \subseteq R then
    return v.count;
end
else if v.bbox \cap R = \emptyset then
    return 0;
end
else
    return rangeCount (v.leftchild, R) + rangeCount (v.rightchild, R);
end
```
(c) Does this algorithm give the correct result? Argue why or why not.
(d) Analyse the worst-case running time of this query algorithm when $k = 2$ and $R$ is an axis-aligned rectangle.

Solution

(a) Draw the result of the construction algorithm on the following point set.
(b) We can see this by looking at how much time each operation takes: constant time for the first if statement, linear time to compute the median and split the point set, and two recursive calls to the same algorithm, each on a point set that is half the size of the original one. If $T(n)$ is the worst-case running time of our algorithm on a set of $n$ points, this leads to the following recursion:

$$T(n) = O(n) + 2T\left(\frac{n}{2}\right).$$

Solving this recursion, we find that $T(n) \in O(n \log n)$.

(c) Yes, it works. The algorithm takes a query range $R$ and a node of the $k$D-tree, and counts all points that are stored in the subtree at this node and lie inside $R$: if $R$ contains the bounding box of the tree, then clearly all points in the tree are in $R$; if $R$ is disjoint from the bounding box of the tree, then clearly no points in the tree are in $R$; otherwise, we don’t know yet and have to look deeper in the tree.

(d) To find the worst-case running time, we use a geometric argument. We observe that the query algorithm only visits those nodes of the tree that have a bounding box that is not completely inside $R$, and also not completely outside $R$. That is, we only visit nodes of the tree whose bounding boxes are intersected by the boundary of $R$.

How many boxes can intersect the boundary of $R$? Since $R$ is a rectangle, the boundary consists of four straight axis-aligned line segments. Because we alternatingly split in the $x$- and $y$-direction, one line can intersect at most $O(\sqrt{n})$ boxes. Therefore, the query algorithm visits at most $O(\sqrt{n})$ nodes of the $k$D-tree.

Since the algorithm takes constant time per visited node or leaf, the total running time of the algorithm is $O(\sqrt{n})$ in the worst case.