Second Round Theoretical Tasks

Swiss Olympiad in Informatics

March 3, 2018

Instructions

- Open your exam only after you have been told so. The examination starts for everyone at the same time and lasts 5 hours.
- With the exception of watches (no smart watches), no electronical devices are allowed on your table. Switch off your mobile phone.
- Start each task on a a separate sheet of paper and write your name on every sheet. Number your sheets and sort them before handing them in.
- Do not use pencil and do not write with red.
- Write legible.
- Have a look at each subtask, even when you couldn't solve the previous one. Some subtasks are independent and and can be solved without the previous subtasks.

Grading

The solutions will be graded according to similar criteria as the first theoretical round. The most important criteria are correctness and asymptotic run time. The quality of the description and the arguments asserting the correctness will also be taken into account.

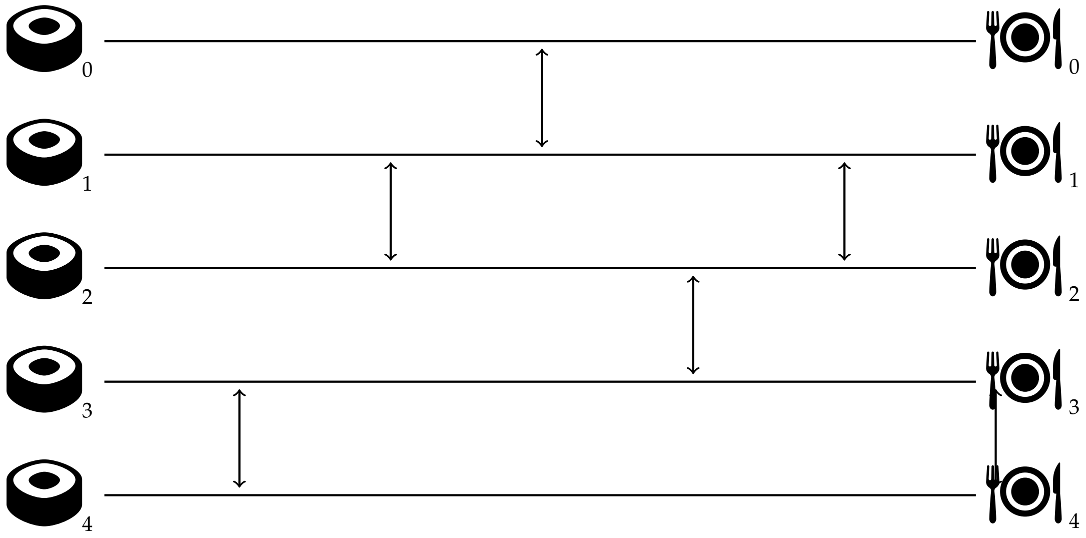
To describe an algorithm, you should structure your solution as according to the following guideline:

1. Describe the idea for an algorithm that solves the problem. The description should be understandable without looking at its source code.
2. Argue about the correctness of the approach.
3. Indicate asymptotic running time and memory usage.
4. Write an implementation in pseudo code. This part should contain the source code of the most important parts of the algorithm in something that ressembles some programming language. You can skip simple parts like input, output and can use mathematical expressions.

If some part of your solution can be used for multiple subtasks, it suffices to write him down only once and refer to it from other parts.

Sushi Belts

In the Takano restaurant there are b sushi conveyor belts that are used to distribute sushi to customers. The belts are straight and run from west to east. They are numbered 0 to $b-1$ from north to south.



On the west end of each belt there is a chef preparing a sushi meal. Each chef has his own signature dish – the meal prepared by chef at belt i is considered to be distinct from meals prepared by chefs at other belts.

On the east end of each belt there is a seat for a guest. As seats are quite far apart, the guest siting at the end of belt i can only consume a meal that arrived on belt i.

Soon this setup became very boring for customers – regardless of where they sat, they were only able to taste a single meal. Hence the restaurant employed w waiters who can move the meals between adjacent belts.

The waiters are numbered from 0 to $w-1$. For each j, waiter j is stationed at the x-coordinate x_j between two neighbouring belts y_j and y_{j+1}. This waiter can take any meal that arrives on belt y_j and place it onto belt y_{j+1} or vice versa. You may assume that all waiters have distinct x-cordinates and hence they do not interfere. However there might be multiple waiters moving meals between the same two belts.

When a customer places an order, all waiters coordinate their efforts to deliver the meal of their choice (if possible) by moving it between adjacent belts.

Subtask 1: Serving all meals to all customers (10 Points)

For $b = 10$ sushi conveyor belts find any placement of 47 or fewer waiters such that each meal can be delivered to each seat in the restaurant. Explain why your way of placing the waiters works.

Subtask 2: Minimum number of waiters (30 Points)

Takano would like to employ as few waiters as possible, while still being able to serve each meal to each seat in the restaurant. For an arbitrary $b \geq 2$, what is the minimum number of waiters needed (as a function of b) and how should these waiters be placed?

For the proof that your solution does indeed use the smallest possible number of waiters, you will get 20 points.

Subtask 3: Suboptimal waiters (60 Points)

The waiters in Takano were unable to solve the previous two subtasks. Hence, they are just standing at some arbitrary positions and they are trying to do their best with delivering the meals to their customers.

You are given b, w and the numbers x_j and y_j that describe the placement of waiters as described above. Let m_i be the number of different meals the waiters can deliver to the seat at the end of belt i. Find an algorithm that computes the numbers $m_0, \dots, m_{b-1}$.

For the proof that your solution for the previous subtask is correct, you will get 20 points.

Osaka Tube

Mouse Stofl has just arrived in Osaka and is about to buy a SOI-Card (Smart Osaka Inter Card) for the local transport network. The city consists of n underground stations which are connected through m lines. A line operates without stop between two stations in both directions. It takes a given amount of time to travel along a line and this time is independent of the direction. Due to the perfect design of the stations, the time needed to change the lines can be neglected. It is also possible to reach each station from any other station, possible via multiple intermediate stations.

The price for a journey is determined as follows: each station is assigned a zone (in total, there are z zones). Travelling in the zone i costs p_i Yen. The price of a journey through multiple stations is computed as the sum of the prices of all the zones along the way (note that each zone is only counted once). As an example, consider a journey through the zones 0, 0, 3, 0, 4, 4, 3. The total price is then $p_0 + p_3 + p_4$.

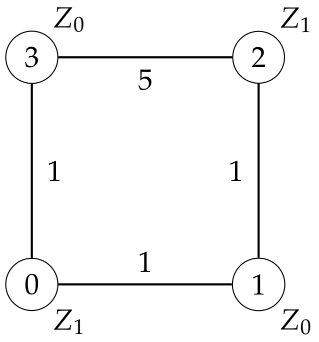
A *cheapest* route is a route with the minimum price.

When travelling using the SOI-Card, the following method is used to charge for travel: you check in at a station, travel arbitrarily, and finally check out at a station. The price which is charged is then the price of a cheapest route between the check-in and check-out stations.

Mouse Stofl likes the pricing, but wonders if it is consistent. A network is *consistent* if there are no two stations such that each shortest route between them (in terms of travel time) is strictly more expensive than any cheapest route.

Since the transport network of Osaka is pretty complicated, Mouse Stofl asks you for your help.

An example of a network is in the following figure. There are two zones ($z = 2$) which are put into parentheses in the figure. The travel times are given next to the edges. The zones cost $p_0 = 1$ Yen and $p_1 = 2$ Yen. The only shortest route from 3 to 2 is the connection $3 \rightarrow 0 \rightarrow 1 \rightarrow 2$. A cheapest route from 3 to 2 is the direct connection $3 \rightarrow 2$ as well as the route $3 \rightarrow 0 \rightarrow 1 \rightarrow 2$.



Subtask 1: Find a counterexample (10 points)

Design a transport network and a division of the stations into zones, such that the pricing is *not* consistent.

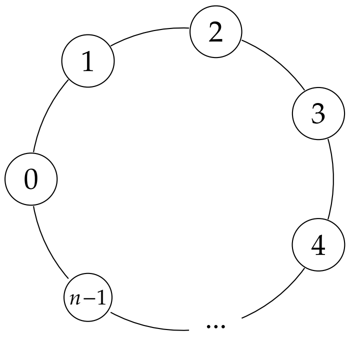
The network should be given as an undirected graph. The travel time along each edge should be marked directly in the graph (note that the travel time is independent of the direction). In addition, each node should be labeled with its assigned zone. Finally, the prices for the individual zones should be listed in a table (since one only has to pay for a zone once, you should label the nodes with the zones rather than the prices).

Subtask 2: Circular network (30 points)

Let us now assume that the transport network is a cycle. This means that the station i is connected to the station $i + 1$, for $0 \le i < n - 1$, and the station $n - 1$ is connected to the station 0.

Design an algorithm with the best possible runtime and space usage which decides whether a given network is consistent.

A circular network is drawn in the following figure.



Subtask 3: Transport network of Osaka (60 points)

Let us now analyze the actual transport network in Osaka. It is not necessarily a cycle, but features only a few zones (in comparison to n).

Design an algorithm which decides whether such a given network is consistent. Optimize running time and space usage with respect to n and m as first priority, and with respect to z as a second priority.

Dandyshow

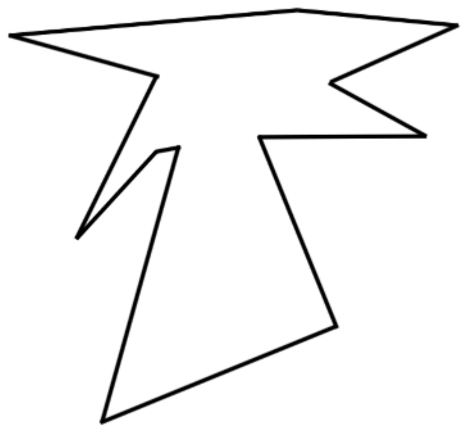
Mouse Stofl is truly famous in his universe and does rightfully hold the title of a space dandy. In one of his innumerable adventures he figured out, that the universe he lives in is a 2-dimensional polygon (At least he cannot permeate through the walls of this universe, of which he crafted a map). He became rich and famous in his universe and now only has a single wish: To be remembered. In order to not be forgotten, he plans to create the biggest disco party in his entire universe! However Stofl wants the disco party to happen at a place, where everyone in the universe can directly see him. Help Stofl find such a place in his universe.

Formal description You are given a 2-dimensional simple polygon (where simple means, that the edges of the polygon do not intersect with themselves and the edges only touch in the edge points of the polygon). Find a point in this polygon, such that every other point in the polygon can be connected through a straight-line, without intersecting/touching the edges of the polygon.

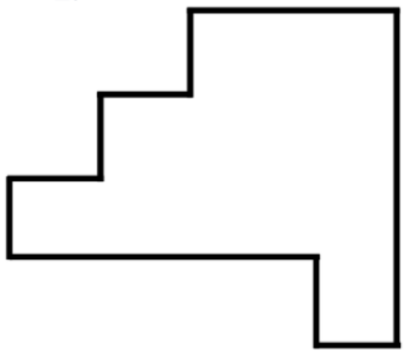
Subtask 1: Solve the task with some given universes (10 points)

Mark all places, where Dandy could possibly situate his disco party.

1.



2.



Subtask 2: Solve the task with rectilinear universes (35 points)

Construct an algorithm, which either finds a point in the universe where Space Dandy Stofl can situate his disco party or outputs "Too bad", if there is no such point.

The polygon is given as a list of x and y coordinates indicating points of the polygon in clock-wise direction.

Additionally Stofl figured out, that each of the edges of the polygon are either horizontal or vertical (in other words all the angles between the edges are either 90° or 270° degrees). We assume that the universe is not twisted in any way.

Subtask 3: Solve the general case (55 points)

Solve the general case, where the polygon can be any simply polygon.

Gas supply

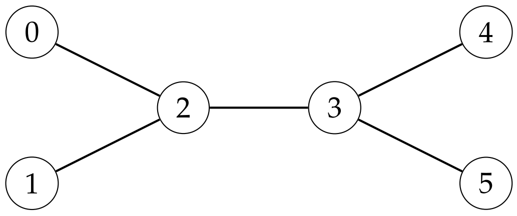
Mouse Stofl switched careers and is now responsible for the gas supply in Tsukaba. The gas network consists out of nodes and pipes (undirected edges), that connect the nodes. The network is connected and does not contain any cycles. In graph theory such a graph is commonly referred to as a tree. Additionally each network has at most 3 pipes running out of the node.

Saidly mouse Stofl already has a problem at his first day of work of work. The gas demand increases erratically (jumpy), which can only mean, that one of the nodes has a leak. He does however not know which node is leaking. In order to figure this out, he develops a plan: In the night no gas will be consumed, which will lead to positive pressures in the pipes, so that all the gas flows in the direction of the leak. Mouse Stofl can read at every pipe in which direction the gas flows, but reading the direction takes a lot of effort and time. He now wants to know how many pipes he needs to check maximally in order to determine, where the leak is located.

Note: You should optimize for the case, that the number of nodes are very large. For example it is better to have $\sqrt{N} + 100$ queries vs. $2 \cdot \sqrt{n} + 2$ queries. Terms other then the most dominant term will be ignored for grading, for example a solution with $n^{\frac{1}{2}} + 6n^{\frac{1}{4}} + 9$ queries will score the same as a solution with $\sqrt{n} - 42$ queries. If you are unsure, just write down the whole formula.

Formal description Given a tree T with max degree 3, find a node L in this tree. You can gain information using the query (A, B), where A and B are two connected nodes and it will return either A, if L is closer to A or B if L is closer to B.

Subtask 1: Solve an example (10 points)

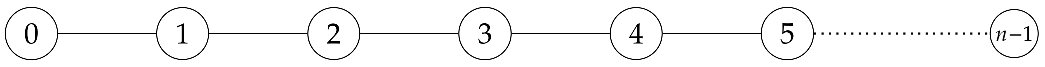


What is the minimal number of queries you have to use in order to find the leak for sure? Explain shortly, why these queries do not overlap.

Subtask 2: A simple network (30 points)

Let's first look at the simpler case, where a network consists solely from nodes that have at most 2 pipes (edges) connected to them. You can assume that the node i is connected with node $i + 1$ (for $0 \leq i < n - 1$). A network would therefor look like this:

You should find the minimal number of queries that are necessairy in order to find a



leak. Proof, that it is always possible to find the leak with the number of queries you came up with. Also develop an algorithm for Stofl, that does the queries for him and determine the runtime of your algorithm (in other words provide a constructive proof).

Subtask 3: The general case (60 points)

Now do the same thing for the general case (3 or less pipes connected to each edge). Proof that your minimal number of queries is sufficient to always find the leak and provide an algorithm that does the queries for Stofl and supply its runtime.