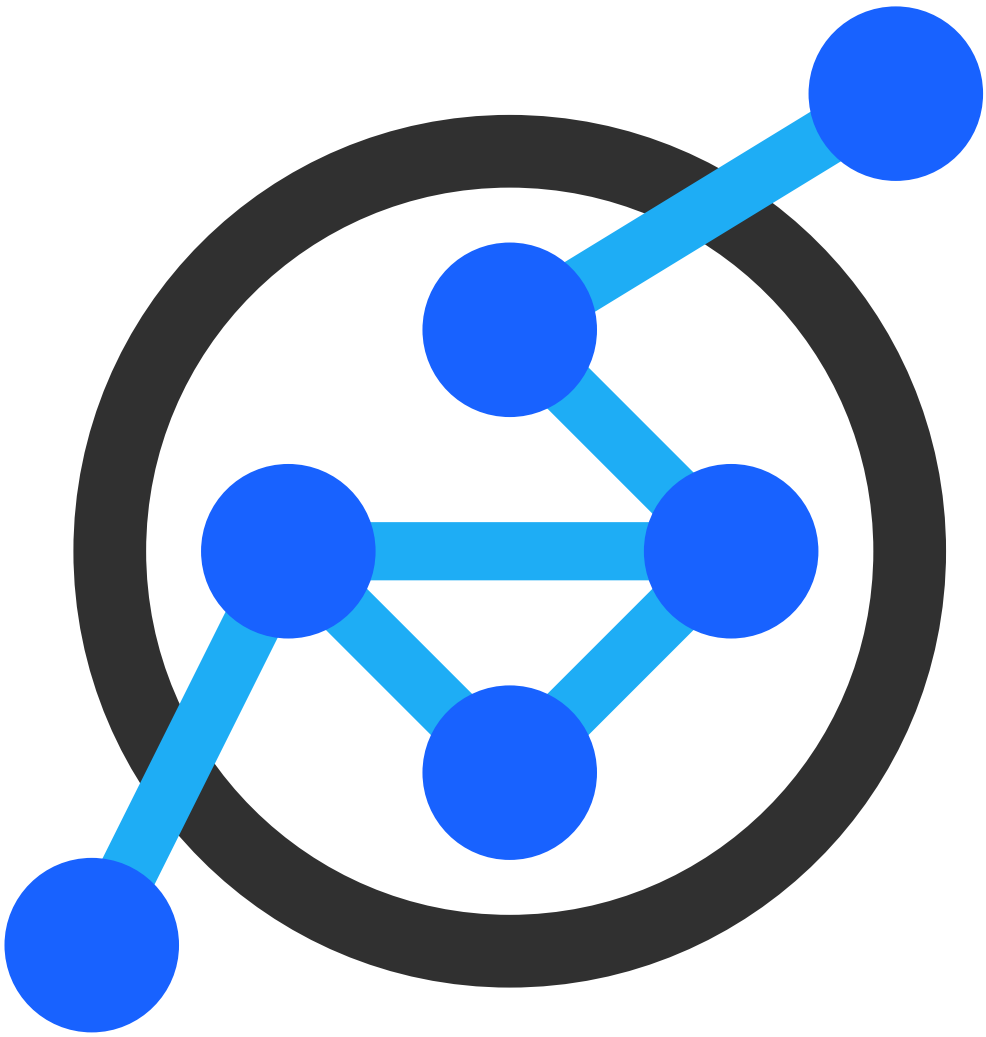
Second Round Theoretical

Tasks



Swiss Olympiad in Informatics

March 6–7, 2021

Instructions

- You have 5h and 30min to solve the tasks. Make sure you reserve enough time for submitting. We recommend reserving 30min for scanning and upload, and if you finish early you may continue working on the tasks and resubmit again later.
- The contest is "Open Book": you may use the internet, books, old solutions etc.
- If you have questions during the round, ask us via email (`info@soi.ch`). We won't give out any hints though, so try to formulate clear yes/no questions.
- Your solution must be fully hand-written (pen and paper).
- Write legibly.
- Your solution must be self-contained: you can't reference any known algorithms, blog articles or papers except for the SOI wiki, sections "First Round" and "Second Round".
- You should upload one PDF per task to the grader, with good enough resolution so we can read everything. A few megabytes per task should suffice (the hard limit is 100 MiB).
- We only look at the last submission of each task, so you may resubmit later on.
- You can see a preview of your submission if you press "Detail", to ensure you uploaded the right file.
- In case you have technical problems either with your scanner or the grader, you can prove to us that you finished in time by handing in a low-resolution photo. Send it to Johannes Kapfhammer over email, Discord or Signal/WhatsApp (contact information redacted for this public task archive).

Grading

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

You can always refer to some content of the SOI Wiki or 2H. You don't need to explain why e.g. Dijkstra works, but you should argue why and how it can be applied. In the case of Dijkstra, you should clearly state on which graph you run it, and note that the edge weights are not negative.

The algorithm should be described in enough detail that it is easy to convert the description into a program. Also write down which data structures you would choose. Usually it is best to just write a short pseudocode.

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

1. Describe the idea for an algorithm that solves the problem.
2. Give pseudocode or explain how one would implement the algorithm.
3. Argue about the correctness of the approach.
4. Indicate asymptotic running time and memory usage.

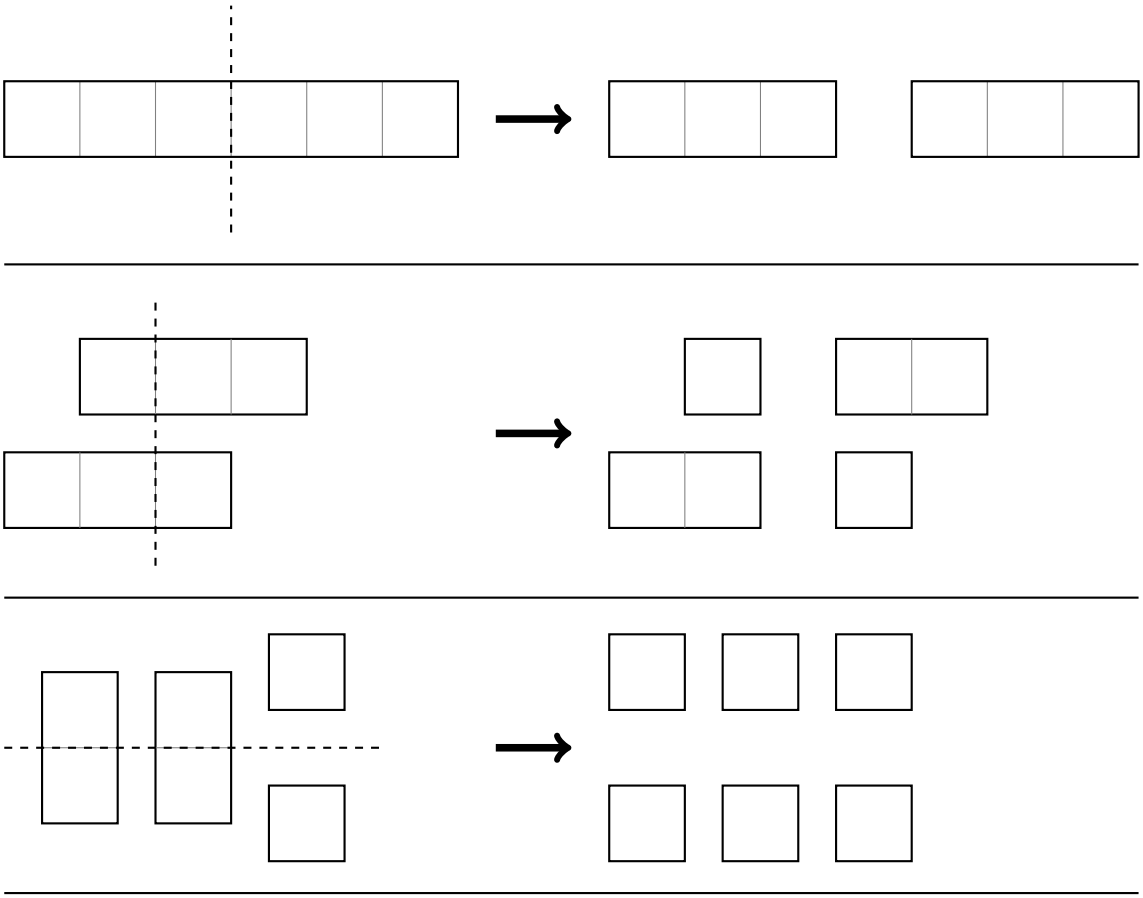
If some part of your solution can be used for multiple subtasks, it suffices to write it down only once and refer to it from other parts. Note, however, that a general algorithm might be able to solve the previous subtasks, but not necessarily optimally.

Lasercut

Mouse Binna recently received a laser cutter. With it, she would like to cut a $n \times m$ bar of chocolate into $n \cdot m$ squares of size 1×1 each. The laser cutter works as follows: Mouse Binna can line up one or more pieces of chocolate in the plane. She may move them around and/or rotate them. Then the laser cutter moves along a straight line, cutting all pieces it moves over. (In other words, in a single step, you can do a straight-line cut on arbitrarily many pieces.) What's the minimum number of cuts Mouse Binna needs to do?

Formally, you are given a $n \times m$ rectangle and must find the minimum number of operations to split it into 1×1 squares. In a single operation, you may first orient each piece, this includes moving and/or rotating each piece. Then you choose a line and every piece that intersects this line is split into two pieces along the line. This concludes a single operation.

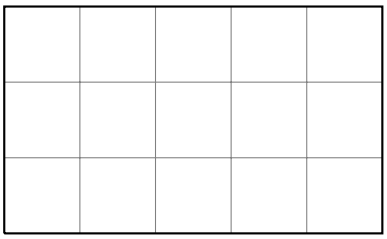
Splitting a 1×6 Bar with 3 Cuts



- The dashed line shows where we will place the cut.
- From each row to the next the pieces are rearranged.

The total number of cuts for this 1×6 bar was 3, and it can be shown that this is optimal.

Subtask 1: A 3×5 bar (10 points)



Give the minimal number of cuts Mouse Binna needs to cut this 3×5 bar of chocolate into 15 single pieces.

Subtask 2: A long thin bar (35 points)

For a $1 \times m$ bar of chocolate, give an algorithm that computes the minimal number of cuts. Prove its correctness and state its asymptotic running time and space usage.

Subtask 3: General case (55 points)

Give an algorithm that computes the minimal number of cuts for an $n \cdot m$ bar of chocolate. Prove its correctness and state its asymptotic running time and space usage.

Plane seating

There is a plane with $R \cdot C$ seats, organised in R rows and C columns. Each seat can be assigned unique coordinates (r, c) (where $1 \leq r \leq R$, $1 \leq c \leq C$). We say that two seats are neighbors if they are directly adjacent in the same row, or if one is in front of the other. Formally, two seats (r_i, c_i) and (r_j, c_j) are neighbors if and only if $|r_i - r_j| + |c_i - c_j| = 1$.

Stofl's big family, consisting of $R \cdot C$ mice, numbered conveniently 1 through $R \cdot C$, are organising a trip to Singapore. Since this is a big family, some mice might not be on good terms with each other.

In particular, you are given two functions: `likes(m)` which returns the set of mice mouse m likes ($1 \leq m \leq R \cdot C$) and `num(m)` which returns the size of `likes(m)` (i.e. `num(m)` $= |$`likes(m)`$|$). This relation is guaranteed to be symmetric: if mouse a likes mouse b then mouse b also likes mouse a, i.e. $a \in$ `likes(b)` $\iff$ $b \in$ `likes(a)`. All other pairs dislike each other.

You want to help Stofl make the trip a success. Help Stofl find a seating arrangement, such that the following conditions are satisfied:

- Each mouse sits in their own seat.
- All pairs of mice who like each other are seated as neighbors.
- No pair of mice who dislike each other are seated as neighbors.

If there are multiple seating arrangements that fulfill the above criteria, you may find any of them. Call the function `assign(m, r, c)` to assign mouse m to seat (r, c). If there are no such arrangements, report that there are none by calling the function `impossible()` (this will ignore any previous `assign(m, r, c)` calls).

You can assume for the analysis that `num`, `assign` and `impossible` run in constant time and in constant space and that `likes` has a runtime and space usage of $O($`num(m)`$)$.

Subtask 1: Concrete example (20 points)

Let $R = 3$ and $C = 3$ and let the $N = 12$ pairs of mice be as follows:

$$\{1,2\}, \{5,6\}, \{9,7\}, \{8,9\}, \{1,3\}, \{3,4\}, \{8,1\}, \{4,5\}, \{6,2\}, \{4,2\}, \{6,7\}, \{9,2\}.$$

Can mice be seated in the plane in accordance with the conditions above?

Subtask 2: A plane with a single seat in a row (30 points)

In this subtask, assume that $C = 1$ (i.e. there is only one seat in a row) but there are no further constraints on R (i.e. we have an arbitrary number of rows in the plane).

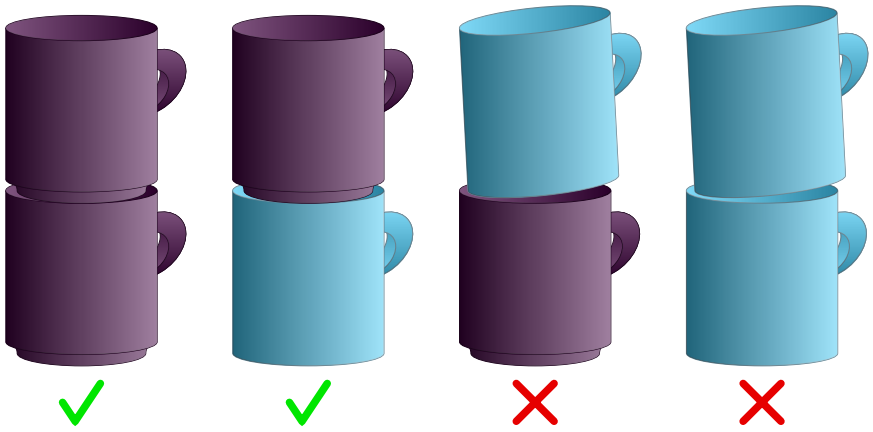
Subtask 3: General case (50 points)

In this subtask, there are no constraints on R or C.

Mug stacking

Mouse Stofl has $2n$ mugs in his cupboard. The mugs are arranged from left to right on a line and are numbered from 1 to $2n$. As mouse Stofl is running out of space horizontally, he would like to stack these mugs.

Some of the mugs have a perfectly flat base, while others curve towards the center. Flat mugs can't be stacked on other mugs, as the resulting stack would be very unstable. Curved mugs on the other hand can be stacked on other mugs just fine. In the picture below, the two stacks on the left are allowed, while the two stacks on the right are not.



Mouse Stofl wants to stack these mugs into n stacks of height 2. To stack mug i onto mug j, mouse Stofl needs $|i - j|$ seconds. Find the minimal time Stofl needs to stack his mugs.

For the analysis you can assume to have a function `type(i)`, which returns the type of mug i ($1 \leq i \leq 2n$) in constant time and space.

Subtask 1: Solve an example (10 points)

For the following initial arrangement of mugs, how can Stofl stack the cups in the minimal time possible? You do not have to prove your answer.

Subtask 2: Perfect pairing (40 points)

Exactly half of the cups have a flat base. Develop an algorithm that finds the minimal time. Prove its correctness and state its asymptotic running time and space usage.

Subtask 3: Stofl's cupboard (50 points)

Stofl's cupboard does not necessarily satisfy the additional constraint of the previous subtask. However, it is guaranteed that at least n cups have a curved base. Develop an algorithm that finds the minimal time for the general case. Prove its correctness and state its asymptotic running time and space usage.

Astrophysics

Dr. Mouse Binna is a famous experimental astrophysicist. Her general research focus is applied general relativity. The details of her latest research cannot be understood by anyone but the most advanced researchers specializing in her field, but it has been popularized as follows: The goal of Mouse Binna is to find black holes and white holes.

Intuitively, a black hole is very *attractive*: starting from any point in the galaxy that is not in a black hole, there exists a geodesic terminating in the black hole, but it is impossible to exit the black hole and reach any point outside of it.[1]

The dual of a black hole is a white hole. Intuitively, a white hole is very *repulsive*: there exists a geodesic originating in the white hole traveling to any point in the galaxy not in a white hole, but there is no way to enter the white hole once you have left it.[2]

It is popularly understood that Mouse Binna is considering a set of $n \geq 2$ space-time locations within a galaxy where there is at most one black hole and at most one white hole.

Using a complicated experimental-theoretical method (involving *science*), Mouse Binna can determine for two locations a and b whether it is easily possible to reach location b when starting in location a.[3]

Important note: Mouse Binna's notion of *simple reachability* is based on astrophysics above any level that can be popularly understood. A location is simply reachable from itself, but that is the only rule on which you can rely. In particular, if you know simple reachability for some pairs of locations, this does not tell you *anything* about simple reachability for any other pair of distinct locations! (In particular, this relation is not transitive: it is possible that we can simply reach b from a as well as c from b, but not c from a.)

Inspired by her method, Mouse Binna has invented the notions of *generalized* black holes and white holes (but she endearingly refers to them as simple black holes and white holes nonetheless):

A black hole is a set of locations that is simply reachable from anywhere, but cannot be simply exited. I.e.:

- If x is inside the black hole, then for *any* location y, x is simply reachable from y.
- If x is inside the black hole, but y is *not* inside the black hole, then y is *not* simply reachable from x.

A white hole is a set of locations from which we can simply reach anywhere else, but that we cannot simply enter. I.e.:

- If x is inside the white hole, then for *any* location y, y is simply reachable from x.
- If x is inside the white hole, but y is *not* inside the white hole, then x is *not* simply reachable from y.

As checking for simple reachability is expensive, Mouse Binna would like to do it the smallest possible number of times.

Subtask 1: A very small galaxy (10 points)

To test her method, Mouse Binna already has exhaustively probed reachability for a small set of $n = 7$ locations within a small galaxy.

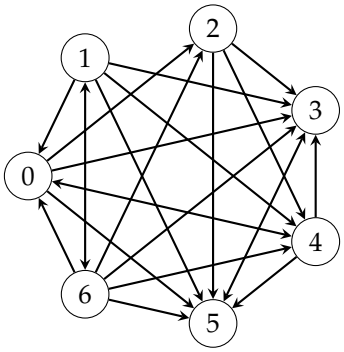
[1] As Mouse Binna likes to point out, this is because once you have entered the black hole, the outside is a point in time rather than a location in space; exiting the black hole would amount to travel backwards in time.

[2] Mouse Binna says that this is because the interior of the white hole is in the past. For example, the big bang is a white hole.

[3] Though Mouse Binna prefers to say: There is an admissible geodesic with low enough proper time starting in a and ending in b within a Riemannian space-time manifold with an appropriate metric tensor, although for most of our purposes, chicken are spherical, we are ignoring the evolution of the system according to the Einstein field equations, etc. [more details omitted in the interest of brevity].

There is an arrow from location a to location b if and only if location b is simply reachable from location a.

There is an arrow $a \to b$ if and only if location b is simply reachable from location a; as well as an arrow in both directions $a \leftrightarrow b$ if and only if a is simply reachable from b and b is simply reachable from a.



Find the set of all locations that are inside the black hole and the set of all locations that are inside the white hole.

Subtask 2: A small black hole (15 points)

In this subtask, there is no white hole. Furthermore, it is guaranteed that there is a black hole and exactly one of the given locations is inside the black hole. I.e., your algorithm does *not* have to check whether there is a black hole or whether it has really size 1. It can assume that the black hole exists and that it consists of precisely one location.

You can ask Mouse Binna to measure (unidirectional) reachability for locations a to b. I.e., Mouse Binna will tell you whether location b is simply reachable from location a. You should find the single location that is inside the black hole using a small number of measurements.

Note that in this task, we ignore lower-order terms, but the leading constant matters. For example, a solution that uses $4 \cdot n^2 + 3 \cdot n$ measurements is equivalent to a solution that uses $4 \cdot n^2 + 2 \cdot n$ measurements, but a solution that uses $3 \cdot n + 100$ measurements is strictly better than a solution that uses $4 \cdot n$ measurements. (The two quadratic solutions are both worse than any linear-time solution, no matter the leading constant.)

Note that in this task (as for other tasks), we expect you to show that your algorithms are correct and efficient, i.e., you have to argue why they will always find the correct sets of locations and you also have to determine some upper bound on the number of measurements that they use.

On the other hand, you do *not* have to prove that your solutions are optimal, but solutions that use fewer measurements will score more points.

Subtask 3: Small black hole and white hole (20 points)

In this subtask, it is guaranteed that there exist both a black hole and a white hole. Furthermore, exactly one of the given locations is inside the black hole and exactly one other location is inside the white hole.

I.e., your algorithm does *not* have to check whether there is a black hole or a white hole and it does not have to check whether they really have size 1. It can assume that both types of holes exist and that each of them consists of precisely one location.

Find both the location of the black hole and the location of the white hole, with a small total number of measurements.

Again, you do not have to prove optimality, but try to make the leading term as small as possible.

Subtask 4: A large black hole (25 points)

In this subtask, there is no white hole. It is guaranteed that there exists a black hole and at least one of the given locations is inside the black hole. Find the set of all locations that are inside the black hole, with a small number of measurements.

Like before, you do not have to prove optimality, but try to make the leading term as small as possible.

Subtask 5: Large black hole and white hole (30 points)

In this subtask, it is guaranteed that there exist both a black hole and a white hole. At least one of the given locations is inside the black hole and at least one of the given locations is inside the white hole. The black hole and the white hole do not have any locations in common. Find the set of locations that are inside the black hole, as well as the set of locations that are inside the white hole. Use a small total number of measurements.

You still do not have to prove optimality, but try to make the leading term as small as possible. Solutions that use $4 \cdot n + O(1)$ measurements can score up to 18 points for this subtask.