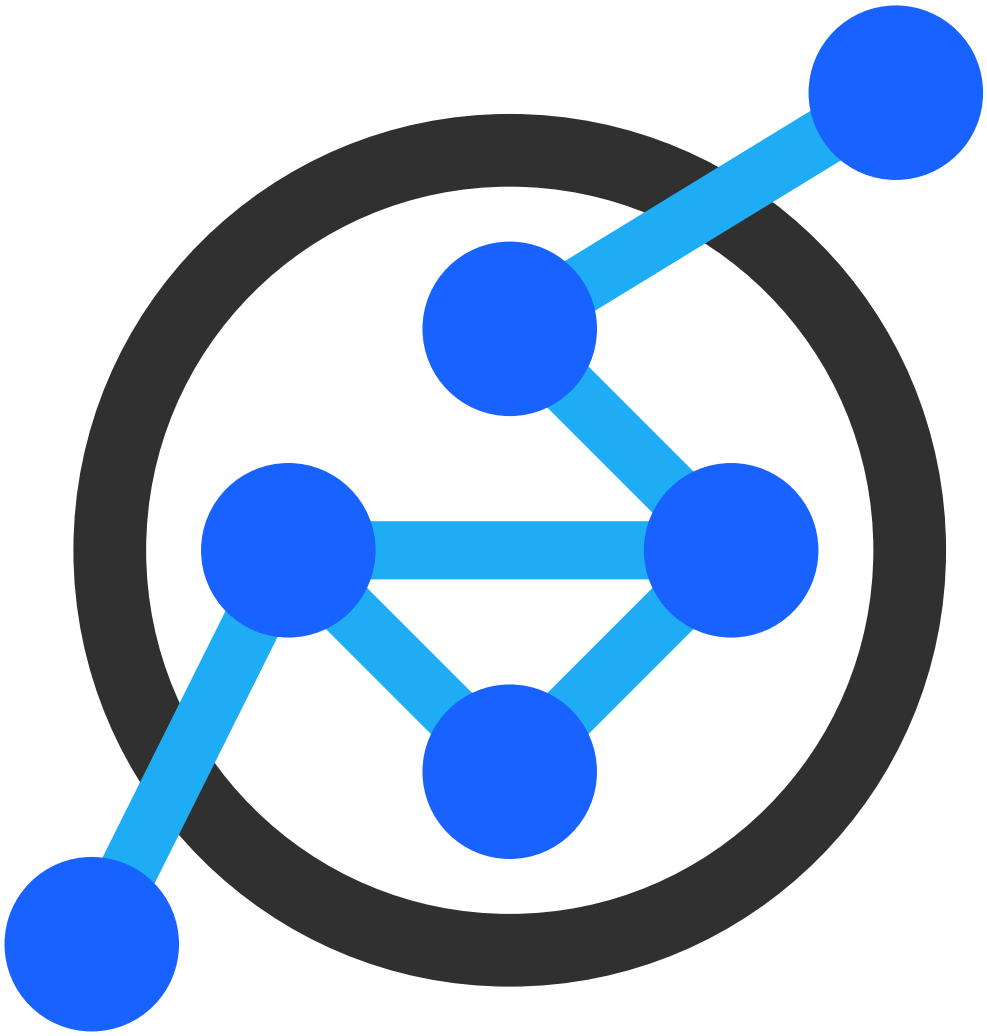
Second Round Theoretical

Tasks



Swiss Olympiad in Informatics

March 4–7, 2022

Instructions

- You have 5h 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 Joël Huber (contact information redacted for public task archive).

Tasks

Note the order of the tasks: The first task is intended to be the easiest task. The other three tasks are in a random order. We recommend you to read all of the problems and think about all of them.

For every task, unless otherwise specified, you can assume that the input is given as *read-only* variables and arrays, which do *not* count towards the memory usage of your solution.

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 and justify the 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.

Consensus

For Mouse Stofl, the Sbrinz Camp always is the highlight of the year. 8 days of nothing but algorithms, coding and extra hard cheese. How awesome!

Alas, for Stofl the camp is way too short. So he went each of the n leaders and asked them what their optimal number of days for the camp would be. As expected, Stofl got lots of different opinions. They told that if Stofl could make them reach a consensus, meaning they all agree on the same number of days, they would change the camp to that duration. "Challenge accepted", thought Stofl.

You are given n numbers $a_0, a_1, \ldots, a_{n-1}$, where a_i is the preferred number of days for the i-th leader.

Stofl can *mediate* between exactly k leaders (where k is a constant given in the input) by inviting them to a discussion round and do the following:

- Vote whether the camp should last $\geq x$ days.
  At least half of the leaders of the discussion round (i.e. $\geq k/2$) need to agree to this vote.
- Vote whether the camp should last $\leq x$ days.
  At least half of the leaders of the discussion round (i.e. $\geq k/2$) need to agree to this vote.
- If both votes passed, the leaders of the discussion round will change their opinion towards x as the optimal number of days for the camp.

Stofl can mediate as often as he wishes, as long as it finishes in a finite amount of time.

What is the maximal number of camp days for which Stofl can reach a consensus?

Example Input: $k = 3$, $a = [3, 1, 4, 1, 5]$.

- In the first round of mediation, Stofl could invite leaders 0, 1 and 3 and have them agree on $x = 1$. The opinions will then be $a = [1, 1, 4, 1, 5]$.
- In the second round of mediation, Stofl could invite leaders 0, 2 and 4 and have them agree on $x = 4$. The opinions will then be $a = [4, 1, 4, 1, 4]$.
- In the third round of mediation, Stofl could invite leaders 0, 1 and 2 and have them agree on $x = 4$. The opinions will then be $a = [4, 4, 4, 1, 4]$.
- In the fourth round of mediation, Stofl could invite leaders 0, 1 and 3 (as he did in the first round) and have them agree on $x = 4$. The opinions will then be $a = [4, 4, 4, 4, 4]$.

Thus their consensus will be 4. For this specific input, this is also the largest possible number.

Example Input: $k = 2$, $a = [3, 1]$. Stofl could get consensus for either 1, 2 or 3.

Subtask 1: Solve the example (10 Points)

For the following input, what is the largest consensus Stofl can produce? $a = [1, 6, 1, 8, 0, 3, 3]$ Solve it for *a)* $k = 3$ and *b)* $k = 2$.

Write down for each round of mediation which leaders you invite, what number they agree on and how the curent opinions will look like. You do not need to argue why your number is optimal.

Subtask 2: Design an Algorithm (90 Points)

Help Stofl find an algorithm that solves the problem for any $2 \leq k \leq n$ and $n \geq 2$. Your algorithm only needs to print the highest number of days (and not the different rounds of mediation).

Prove why your algorithm is correct.

Indicate the running time and space usage in terms of n and k. Try to optimize the running time of the algorithm before the memory usage. Note that $k \leq n$, meaning an algorithm with a running time in $O(n^7 k)$ is considered faster than an algorithm with a running time in $O(n^8)$, but an algorithm with a running time in $O(nk^2)$ is considered slower than one with a running time in $O(n^2)$ (e.g. in a case where $n = k$ holds).

You can get partial points for solving the following subproblems:

a) $k = 3$ – mediation always happens between 3 people (30 points).

b) $a_i = 8$ or $a_i = 9$ for all i – there are at most two different opinions (30 points).

If you solve the full problem optimally, you can of course skip those subproblems; however you may be able to score more points if you find a faster solution for the subproblems.

Scientific Conference

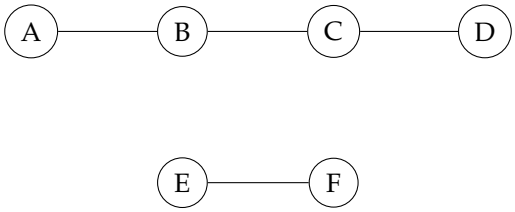
Mouse Binna is organizing a scientific conference. There are n mouse scientists that she could invite. She also knows that m pairs of scientists have published a paper together.

For each mouse scientist that is invited to the conference their *popularity* at the conference is defined as the number of other mouse scientists that were invited to the conference and that have published a paper together with them.

Mouse Binna knows that if there exists an invited scientist such that their popularity plus one (we add one to account for the scientist themselves) is at least half of the total number of invited scientists, then the conference small talk will focus too much on their work. That would be detrimental to the forming of new ideas, so she must not allow that to happen.

Formally speaking, if Mouse Binna invites k mouse scientists $a_1, a_2, \ldots, a_k$ to the conference, then for each i we must have $1 + \#\{j : 1 \leq j \leq k, j \neq i, \{a_i, a_j\} \in E\} < \frac{k}{2}$, where E is the set of pairs of scientists that have published a paper together. Note that those pairs are unordered, so $\{a_i, a_j\} \in E$ means the same as $\{a_j, a_i\} \in E$.

Consider the following example, where circles represent scientists and scientists that have published a paper together are connected with a line:



Inviting all those scientists to a conference would not work, because the scientists B and C would have popularity 2, so the constraint above for B and C would look like $1 + 2 < \frac{6}{2}$, which is false. However, we could invite the scientists A, C, D, E and F. Then the popularity of the scientists C, D, E and F will be 1, and the popularity of the scientist A will be 0, and both $1 + 1 < \frac{5}{2}$ and $1 + 0 < \frac{5}{2}$ are true.

Mouse Binna does not care how many scientists are invited to the conference, as long as there is at least one: $k > 0$. Can you help her figure out which scientists she should invite, or determine that organizing such a conference is impossible? In case there are multiple sets of scientists that would satisfy the above constraint, any such set will do.

From her previous experience organizing scientific conferences, Mouse Binna knows that it's easier to find such set of scientists when the number of pairs of scientists that have published a paper together is small. Therefore in this problem we will reward solutions that only work for cases where the number m is relatively small, and solutions that achieve better runtime complexity when m is relatively small. Please see the subtask list below for more details.

In each of the subtasks 2-4, you need to describe the algorithm, write down *pseudocode*, the *runtime complexity*, *memory usage* of your algorithm and explain its *correctness*.

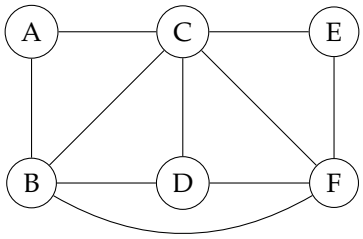
In order to achieve the full score for each of subtasks 2-4, your solution needs to be correct and have good runtime complexity and memory usage for the cases where m satisfies the bounds from that subtask. However, partial progress and suboptimal solutions in each subtask will also be rewarded.

You are free to propose solutions that solve all or multiple subtasks at the same time, or to propose different solutions for different subtasks.

In all subtasks, you may assume $n \geq 5$.

Subtask 1: Solve an example (10 points)

For the following set of scientists, where circles represent scientists and scientists that have published a paper together are connected with a line or a curve, can you find any set of scientists that can be invited to a conference?



To avoid misunderstandings, here is the same set of pairs of scientists that have published a paper together in text form:

$$\{\{A,B\},\{A,C\},\{B,C\},\{B,D\},\{C,D\},\{C,E\},\{C,F\},\{D,F\},\{B,F\},\{E,F\}\}$$

You do not need to prove your answer, just writing down any correct set is enough.

Subtask 2: Very few papers (20 points)

The number of pairs of mouse scientists that have published a paper together is very small: $m < \frac{(n-5)^2}{10}$.

Subtask 3: Few papers (30 points)

The number of pairs of mouse scientists that have published a paper together is relatively small: $\frac{(n-5)^2}{10} \leq m < \frac{(n-5)^2}{4}$.

Subtask 4: Many papers (40 points)

The number of pairs of mouse scientists that have published a paper together is relatively big: $m \geq \frac{(n-5)^2}{4}$.

Oneway Portals

Mouse Stofl recently got elected as minister of transport in the Alliance of Independent Systems (*the Alliance* for short).

The Alliance consists of n planets and planet Apodemus is the capital. Each planet was inhabited by exactly 1 mouse. Under the project name "all portals lead to Apodemus", Stofl's predecessor launched an initiative to relocate all mice to the capital, in which he built $n-1$ one-way portals between planets such that it is possible to reach Apodemus from all planets (either directly or indirectly by going through other planets). Ultimately, the previous minister of transport was not reelected because his mission failed; while some planets are now uninhabited, others still contain exactly 1 mouse that refused to relocate to planet Apodemus.

To encourage the remaining mice to relocate, mouse Stofl, the new minister of transport, wants to build up to k additional one-way portals. The goal is to minimize the average number of portals needed to travel to planet Apodemus, taking into account each planet that is still inhabited by a single mouse. Stofl knows the configuration of the $n-1$ one-way portals and knows which planets are inhabited, but does not know what the best way to add k new portals would be. Can you help him?

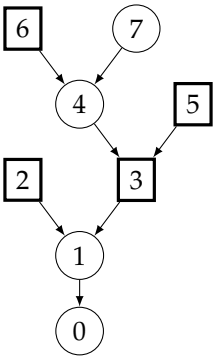
More formally, we can represent the Alliance as a rooted directed unweighted tree with n vertices, where all the edges are directed towards the root. The root of this tree is planet Apodemus. A subset S of the vertices is still inhabited by a single mouse on each vertex (where Apodemus is not in S). You can add up to k directed unweighted edges. Let $f(v)$ be the length of the shortest path from v to the root after adding these edges and let

$$A = \frac{\sum_{v \in S} f(v)}{|S|}$$

be the average number of portals for S using those edges. Find the minimal achievable value of A.

Subtask 1: Solve the example (10 Points)

Stofl first wants you to solve an example to check whether you are able to help him. In this example, $n = 8$, $k = 2$, the inhabited planets are $S = \{2, 3, 5, 6\}$ and Apodemus is vertex 0.



What is the minimal value of A and how would you place the portals to achieve it? You do not need to justify your solution.

Subtask 2: Expensive Portals, Poor Government (20 Points)

The portals are very expensive, and the Alliance can only afford one new portal. Thus in this subtask, $k = 1$.

Describe an algorithm that solves this problem and argue about its correctness. Your algorithm only needs to compute A. You may assume that the input is given in some convenient form (e.g. Apodemus is vertex 0 and the tree is given as adjacency list).

Analyze its running time and memory usage in terms of n.

Subtask 3: The Alliance (70 Points)

In this subtask, please help Stofl to solve the general problem.

Describe an algorithm that solves this problem and argue about its correctness. As before, your algorithm only needs to compute A and you may assume that the input is given in some convenient form.

Analyze its running time and memory usage in terms of n and k. Try to optimize the running time of the algorithm before the memory usage.

Note that $k \leq n$, meaning an algorithm with a running time in $O(n^7 k)$ is considered faster than an algorithm with a running time in $O(n^8)$, but an algorithm with a running time in $O(nk^2)$ is considered slower than one with a running time in $O(n^2)$ (e.g. in a case where $n = k$ holds).

Remark: In case you can not find an algorithm, significant progress towards a solution will be awarded partial points.

Particle detector

Mouse Binna wants to do a particle physics experiment at the Large Hadron Collider, which is located deep below the ground near Geneva. For this purpose, she has built a particle detector, consisting of a ring of sensors. There are N sensors numbered $0, 1, \ldots, N-1$. Sensors 0 and 1 are adjacent, as well as 1 and 2, $\ldots$, $N-2$ and $N-1$, $N-1$ and 0.

Unfortunately, some of the sensors may have a certain manufacturing defect, which is not easy to detect, and Binna doesn't know which sensors are broken (it could even be that all or none are broken). The sensors have a special mode which allows to detect if an adjacent sensor is broken. If an adjacent sensor is broken, it reports a 1, otherwise a 0. However, if both adjacent sensors are broken, they cancel each other out and the fault cannot be detected. Furthermore, if the sensor with which the brokenness is detected is itself broken, it inverts the result.

Essentially, this means that a good sensor will report 0 if both neighbors have the same brokenness (either both are good or both are broken), and 1 otherwise. Similarly, a broken sensor will report 1 if both neighbors have the same brokenness, and 0 otherwise.

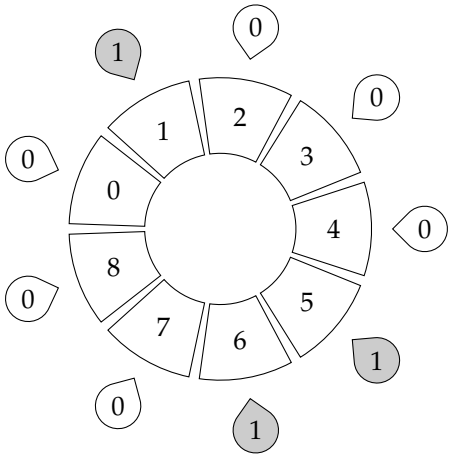
Mouse Binna has collected the reports from all the sensors. She now wants to know how many broken sensors there can be at most. It could also be that there does not even exist a configuration of broken and good sensors which would result in the reported values; this indicates that something else is wrong with the particle detector.

Formal Description Given is the number of sensors $N \geq 3$, as well as the report $r_i \in \{0, 1\}$ for each sensor $i \in \{0, \ldots, N-1\}$. Output the maximum possible number of broken sensors, or -1 if there is no configuration of sensors which would result in the given reports.

Subtask 1: A toy detector (15 points)

Mouse Binna has built a small detector, which looks like this. The numbers in the bubbles are the reports of the sensors.

Can you figure out how many broken sensors there can be at most?



Subtask 2: All sensors report 1 (10 points)

All sensors of the detector report 1.

Calculate the highest possible number of broken sensors for any given N, for the case where $r_i = 1$ for all i. It is enough to write an expression, you don't need to describe an algorithm. Show why your answer is correct.

Subtask 3: All sensors report 0 (15 points)

All sensors of the detector report 0. Does this mean that everything is fine?

Calculate the highest possible number of broken sensors for any given N, for the case where $r_i = 0$ for all i. It is enough to write an expression, you don't need to describe an algorithm. Show why your answer is correct.

Subtask 4: General case (60 points)

Now, there are no longer any restrictions. Describe your algorithm.