Second Round Theoretical

Tasks

Swiss Olympiad in Informatics

March 9, 2019

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.

Cheese Recommendation

Mouse Stofl is doing an apprenticeship with cheese grandmaster JK. Unfortunatly JK is very secretive when it comes to how to choose two types of cheese that go well together. Mouse Stofl knows that each kind of cheese i has two characteristics a_i, b_i which gradmaster JK determines with his excellent sense of smell. Mouse Stofl thinks that he has cracked the formula on how to determine how well two types of cheese go together:

$$\frac{|a_i - a_j| \cdot b_i \cdot b_j}{\max(|b_i|, |b_j|)} \tag{1}$$

In this formula, $|x|$ stands for the absolute value of x (x if $x \geq 0$ or $-x$ if $x < 0$) and $\max(x, y)$ stands for the maximum of x and y (x if $x \geq y$ and y if $x < y$).

We call the result of this formula the recommendation score. The higher this value, the better cheese i and j go together. You can also combine a type of cheese with itself. In that case the recommendation score goes to 0 ($|a_i - a_i| = 0$). There may also be types of cheese that do not fit together at all and have a negative recommendation value.

Mouse Stofl would like to impress cheese grandmaster JK and would like you to write a program that determines the highest possible recommendation score for him.

Formal Description We are given the number of different types of cheese n ($2 \leq n$). For each type of cheese are given two integers a_i, b_i ($1 \leq i \leq n$). Find the maximal value of $\frac{|a_i - a_j| \cdot b_i \cdot b_j}{\max(|b_i|, |b_j|)}$.

Subtask 1: Make a recommendation (10 points)

Today there are 9 kinds of cheese in the shop. Which two of them should Stofl combine such that the recommendation score is as large as possible?

i	0	1	2	3	4	5	6	7	8
$\mathbf{a_i}$	1	-1	5	-2	3	5	-6	4	0
$\mathbf{b_i}$	5	1	2	4	7	-3	-2	3	4

Subtask 2: Select two types of cheese (70 points)

On the next day there are so many different types of cheese that Stofl can not perform the calculation by hand anymore. However, he notices that all the cheese types have a positive value for b ($b_i \geq 0$).

Design an algorithm that given n and the values a_i and b_i finds the maximal recommendation score.

Subtask 3: Negative characteristics (20 points)

This time there are also types of cheese with a negative b_i characteristic. Assuming you know an efficient solution for subtask 2, how can you modify the algorithm such that it works in the general case?

Note: Even if you were not able to solve subtask 2, you can assume that there is an efficient algorithm for the problem of only positive b_i.

Pomegranates

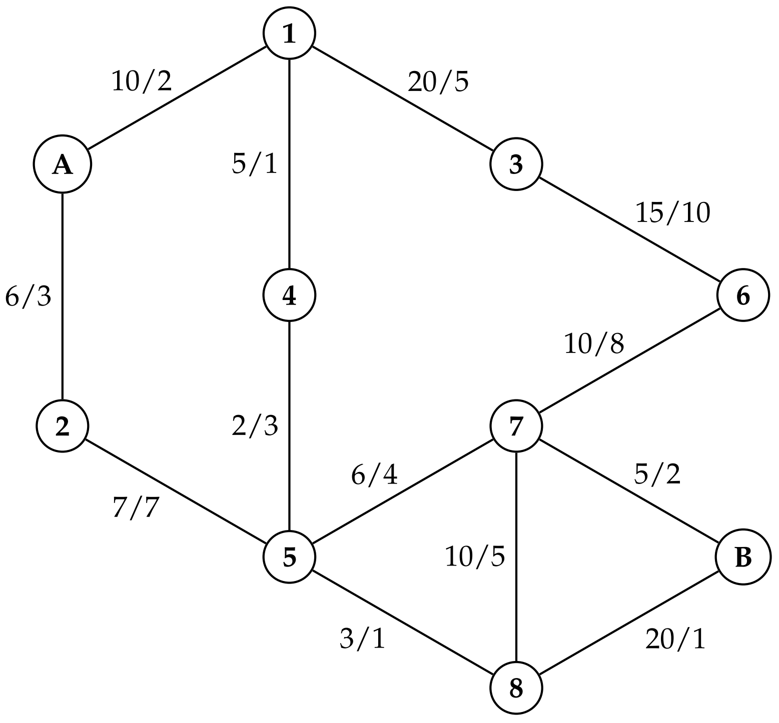
Mouse Ada is the organisator of the MOI (Mouse Olympiad in Informatics) in Baku. One of her functions is to deliver as many pomegranates as possible as snacks from the warehouse to the contest place. To deliver the pomegranates within the time T from the warehouse to the contest place, Mouse Ada uses a truck that can fit any amount of pomegranates. The problem is that every road in Baku has a weight limit, which the truck is not allowed to exceed to be able to drive on. And as the pomegranates weigh something, she can't take arbitrarily many with her . . .

Help Mouse Ada to find a path shorter than T such that she can fill the truck with the largest amoutn of pomegranates.

Formal Description Given is a graph where every edge has two positive numbers w and t. Find the maximal weight W such that there exists a path from A to B whose length (the sum of t of the edges) is $\leq T$ and for every edge on the path we have $w \geq W$.

It is guaranteed that the graph is connected and that there always is a path from A to B in time at most T.

Subtask 1: Solving an example (10 Punkte)



Which is the maximal weight such that the truck can reach the contest place from the warehouse in $T = 18$ time and doesn't exceed any weight limit? The warehouse is marked with A, the contest place with B. On each edge there are two numbers, w/t, the left one is the weight limit and the right one is the time, the truck needs. Explain shortly why there is no better solution.

Subtask 2: Infinite time (30 points)

We assume that it doesn't matter when the pomegranates arrive on the contest place, therefore is $T = \infty$. The weight limits for the roads are positive integers.

Subtask 3: Two weights (20 points)

The time limit is back. But now each road has a weight limit of 1 or 2 that means $1 \leq w \leq 2$.

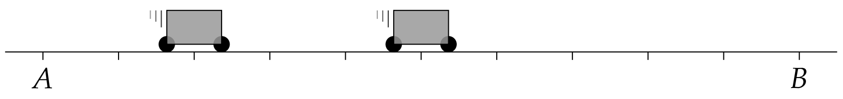
Subtask 4: General case (40 points)

The time limit remains and the weight limits are now positive integers again.

Mine Carts

As a kid, mouse Benjamin loved to play in his sandbox. Recently, he became head of a ore mine in Azerbaijan. Part of his new job is to schedule the mine carts along a track to bring ores out of the mine. Completely overwhelmed by this new position, he asks you for help.

There are two stations, one near the mining area, station A, and one at the outside, station B. Both stations are connected by a single track. The track can only be operated in one direction at a time, as otherwise the carts would collide. One trip on the track takes d seconds (regardless of the direction $A \to B$ or $B \to A$). Mouse Benjamin may send a new mine cart along the track at either station each second.



Initially, there are n mine carts at station A and 0 mine carts at station B. The mining crew mines new units of ore at times $t_0, t_1, \ldots, t_{n-1}$, given in increasing order. Once mined, a unit is ready to be sent from station A to station B on a mine cart. A mine cart can carry at most one unit per trip. The work day of mouse Benjamin is finished once all ores have been sent to B and all mine carts are at station A again.

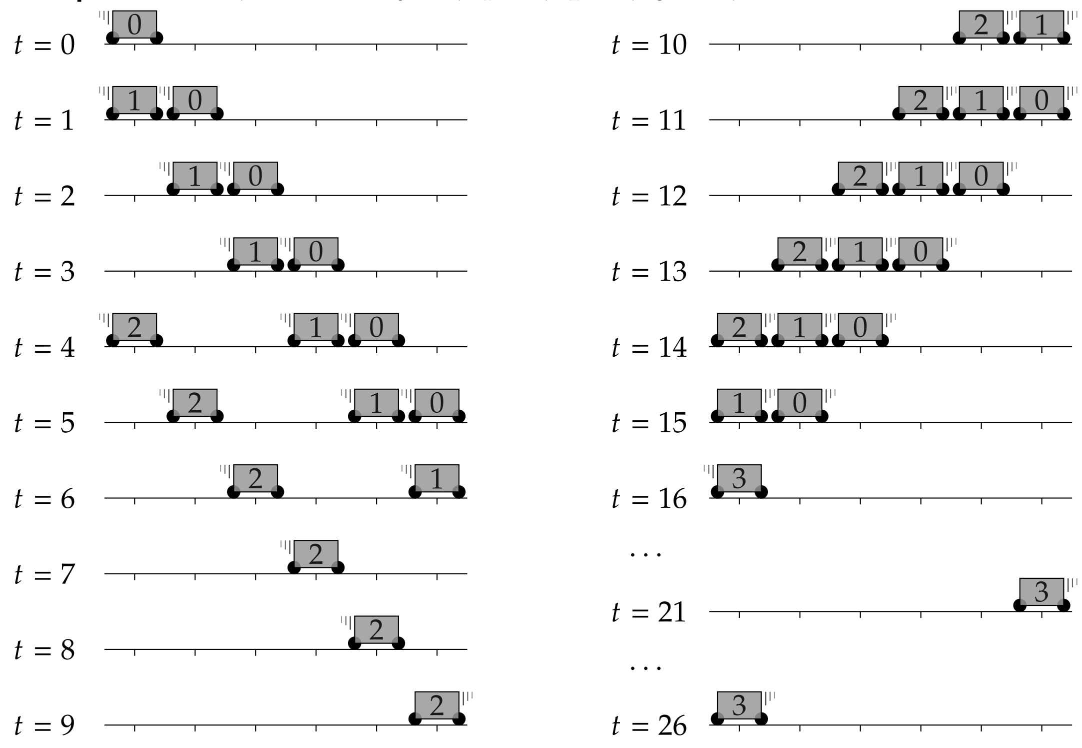
The problem of mouse Benjamin is to find out the minimal time T such that there is a way to schedule all mine carts to transport all ores to station B and bring the mine carts back to station A.

Formal description Given integers $t_0, t_1, \ldots, t_{n-1}$ with $t_0 < t_1 < \cdots < t_{n-1}$ and an integer $d > 0$, find the minimal T such that there are integers $a_0, a_1, \ldots, a_{n-1}$ (the times when you send the i-th mine cart from A to B) and $b_0, b_1, \ldots, b_{n-1}$ (the times when you send the i-th mine cart from B to A) satisfying

- $b_i + d \leq T$ for all i (you are done at time T),
- $a_i \geq t_i$ (can send a unit of ore only after it has been mined),
- $b_i \geq a_i$ (can send mine cart back to A only after it has arrived at B),
- $a_i \neq a_j$ and $b_i \neq b_j$ for $i \neq j$ (can't start two carts at the same time) and
- $|a_i - b_j| \geq d$ for all i and j (can't use the track in both directions at the same time).

Analysis An algorithm for this problem is given the two integers n and d, as well as the list of starting times $t_0, \ldots, t_{n-1}$, with $t_0 < \cdots < t_{n-1}$. You should give the running time and space usage depending on n and d. Both parameters are independent from each other, so $\Theta(n + d)$ is worse than $\Theta(n)$ and $\Theta(d)$. Similarly, $\Theta(\min(n, d))$ is better than $\Theta(n)$ and $\Theta(d)$. The values $t_0, \ldots, t_{n-1}$ do not count for the memory usage. You may read them without additional memory usage, but you may not modify them in-place.

Example If $n = 4, d = 5$ and $t_0 = 0, t_1 = 1, t_2 = 4, t_3 = 16$, the answer is 26:



The starting times of the carts are $a = [0, 1, 4, 21]$ and $b = [11, 10, 9, 26]$.

Subtask 1: Busting Stofl's Algorithm (15 Points)

Mouse Stofl came up with the following algorithm to solve this problem:

Algorithm: Stofl's Heuristic for Mine Carts

```
input  : $n, d$ and the array $t[0], \dots, t[n-1]$
output : Some upper bound on $T$, but not a minimal one
$w = 0$                                   // the number of mine carts waiting in B
for $i = 0$ to $n - 2$ do
    $x = t[i+1] - t[i] - 2 \cdot d$       // extra time if we would do a roundtrip
    if $x \geq 0$ then                    // if we have time for a full roundtrip
        $w = \max(0, w - x)$              // do one and send back extra carts
    else                                  // otherwise just send the ore to B
        $w = w + 1$                       // and add it to the waiting mine carts
return $t[n-1] + 2 \cdot d + w$           // roundtrip for last ore followed all waiting carts
```

In this subtask you have to:

- Indicate the running time and memory usage of Stofl's algorithm. You *do not* need to justify your answer (5 points).

- Find a counterexample where Stofl's algorithm doesn't give an optimal solution. State the output of Stofl's algorithm (no justification needed) and how the optimal solution would look like (10 points).

Subtask 2: Optimal Algorithm (85 Points)

Now it's your turn! Find an algorithm that solves the problem optimally.

Light Show

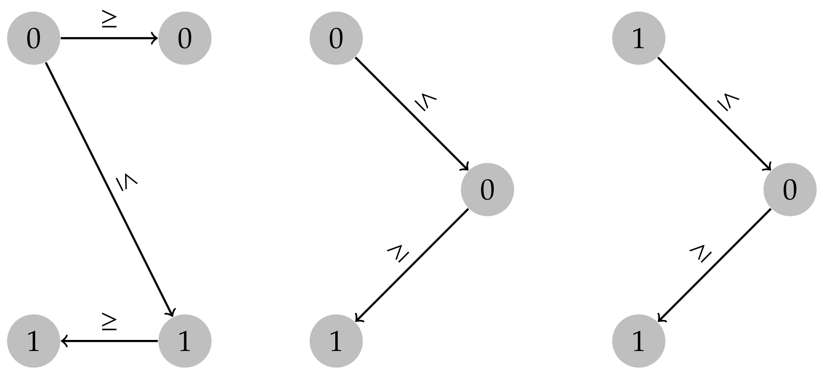
Mouse Stofl wants to put on a nice light show in front of his house. He's bought some garlands with lights on them. The garlands come in a weird shape: they do not just form a long strip but a sort of tree of light bulbs and cables.

His idea is to show all of the possible combinations of lights turned off and on at exactly one point during the show, and to transition between combinations by switching on or off only one light bulb at a time.

Unfortunately, he bought the cheapest garlands available from some shady store and they all have a big miswiring problem: for any two light bulbs directly connected by a cable, exactly one of the two can only function if the other one is turned on. That makes a lot of combinations impossible. This disappoints Stofl quite a lot, but he's decided to go through with his idea, only skipping the impossible combinations. However, that makes finding a suitable sequence of combinations quite difficult and that's the reason why Stofl asked you to devise an algorithm to do that for him.

Formal description Given is a graph G with n vertices and m undirected edges. It is guaranteed that this graph is acyclic, i.e. there is no cycle in it. The graph is thus a collection of trees. Vertices in this graph take a value of either 0 or 1 (depending on the state of the corresponding light bulb). Edges are labeled (directionally) with either '$\leq$' or '$\geq$': if an edge from vertex a to vertex b is labeled with '$\leq$', then it is never allowed to have the value of a be 1 and the value of b be 0 (similarly, if such an edge is labeled with '$\geq$', then it is not allowed to have the value of a be 0 and the value of b be 1).

Any mapping of the vertices of G to $\{0, 1\}$ is a valid combination if and only if all the conditions depending on the labels of the edges are satisfied. For example, the two graphs on the left below are in a valid combination, whereas the rightmost one is not. Your task is to develop an algorithm to compute a working sequence of all and only the valid combinations such that any two consecutive combinations differ in the value that they give to exactly one vertex.



To that effect, you should implement the following two functions:

1. *start*: takes as input the graph, initializes any data structure that you need and returns a starting assignment of values to vertices.
2. *next*: returns the index of the next vertex whose value should be switched or −1 if you've reached the end of the sequence.

If e is an edge, you can use *e.a* and *e.b* to know which two vertices are bound by e. *e.d* is '≤' if a must be '≤' than b and '≥' if a must be '≥' than b.

Subtask 1: Solve the example (8 points)

Solve the problem for the graph below by giving a list with the initial state of each vertex as well as a sequence of vertices that you want to switch so that you go through each valid combination exactly once.

$$\begin{tikzcd}
 & B & & \\
A \arrow[ur, "\leq"] \arrow[r, "\geq"] \arrow[dr, "\geq"] & C & E \arrow[r, "\leq"] & F \\
 & D & &
\end{tikzcd}$$

Subtask 2: Combine garlands (24 points)

First, Stofl would like you to help him produce a show with some garlands for which he has already found a solution: you just need to combine them! In this subtask, you can assume to be given *start* and *next* functions for each individual tree in the graph. You're given the graph as a list of these trees L. You may use instructions like *L[i].start()* to use the i-th tree's *start* and *next* function. Your task is to combine the garlands; in other words, you should implement the functions *start* and *next* for the whole graph. You may assume that the runtime of the functions for trees are $O(x)$ for *start* on a tree of x vertices and $O(1)$ for *next*.

Subtask 3: Individual garlands (68 points)

In this subtask, you're working on additional individual garlands to improve the show. That's all you need, since you can already combine them for Stofl with your algorithm from Subtask 2! You can therefore assume that G is connected, i. e. you're working on a tree. You should make functioning *start* and *next* functions for that tree. You are allowed to assume the following properties for this tree in exchange of a point deduction:

Sort of graph	Max. points
Path (maximal degree 2) with edges from i to $i+1$ with label '≥'	8
Binary tree (maximal degree 3) of which you're given the root r. Edges are labeled such that the parent is always '≥' its child.	24
Tree of which you're given the root r. Edges are labeled such that the parent is always '≥' its child.	34
Tree with arbitrary labels	68

Grading In the two last subtasks, the main focus is the correctness of your algorithm. Your algorithm does not need to be optimal to score some or even most of the points. Here is a list of runtimes for which you can aim and the percentages of the total points that you may be awarded for each of them. In this table, P stands for the number of valid combinations and poly(n) for a polynomial in n.

Runtimes	Fraction of the points
start in $O(P \cdot \text{poly}(n))$, *next* in constant time	25%
start in $O(\text{poly}(n))$, *next* in $O(n)$	50%
start in $O(n)$, *next* in constant time in the average case	75%
start in $O(n)$, *next* in constant time in the worst case	100%