Institut für Informatik
Prof. Dr. Heiko Röglin
Joshua Könen,
Sarah Sturm,
Aaron Weinmann

Lab: Efficient Algorithms
for selected Problems
Summer 2025

Problem Set 4

Tasks that are marked with * are optional for groups with only two members and will not count into their score.

1 Maximum Tolerance Class

You are given a directed graph $G = (V, E)$ on the set of vertices $V = \{0, \dots, n-1\}$. Let $v_1, v_2 \in V$. We say that $v_1 \sim v_2$ if there exists a path from v_1 to v_2 or a path from v_2 to v_1.

$\sim$ is a *tolerance relation* as it is *reflexive* and *symmetric*. A *tolerance class* is an inclusionwise maximal set $W \subseteq V$ such that for all $w_1, w_2 \in W$ it is true that $w_1 \sim w_2$.

Compute the maximal cardinality of a tolerance class.

Input: The first line contains the number of vertices n and the number of edges m. The following m lines describe the edges. You can assume that $n \leq 50000$.

Output: Output the maximal cardinality of a tolerance class.

Sample Input:

```
5 5
0 1
1 2
2 3
3 1
4 3
```

Sample Output:

```
4
```

2 Social Nerds

This years field trip for the computer science students is a hiking trip. There are far too many students to hike with one big group, so the students decide to form several smaller groups. This is where the problem emerges, every student has a different idea of what number is a perfect group size. Fortunately, the students are very understanding and willing to compromise, so every student $i \in n$ gives an interval $[a_i, b_i]$ of group sizes they could join.

For a given set of students (and their individual intervals), is it possible to form hiking groups such that the size of each group lies within the interval of each of the groups members and no student is left behind?

Input: The number of students n followed by n lines with the upper and lower bound of the interval a_i and b_i. You can assume that $a_i \leq b_i \leq n$.

Output: If it is possible to partition the students into groups $(S_1, S_2, \dots, S_k)$ such that $|S_j| \in [a_i, b_i]$ for all $i \in S_j$, output `possible`, otherwise, output `impossible`

Sample Input:

```
5
2 2
3 4
2 4
3 5
2 5
```

Sample Output:

```
possible
```

3 Tyrion*

During his time as hand of King Joffrey, Tyrion Lannister met Varys, the Master of Whisperers, in the gardens of the palace. 'Tyrion', Varys said, 'I have made a list of all the houses, major and minor, and sorted them according to whispers and rumors, into who is friend and foe of House Lannister. Would you like to know those you are truly on your side?' Tyrion shrugged, 'of course', he says, 'but I think I have a fairly good idea myself. 'And he named a few of which he was very certain. 'Ha!', Varys said, 'not bad, but not true either. I reveal this to you: The number of houses that you named and that are true friends of house lannister, this number, it is odd.' Tyrion was not amused, but then made up his mind, naming a different set of houses. Again, Varys would not tell him which were right, but only that the number of friends in the list was this time even. And so they continued for quite some time, until Varys remarked: 'But now, my dear friend, you must have gotten it. You should now my true list of true friends.' Tyrion nodded, and while the listening bystanders were utterly confused, the two men parted ways knowing that they were in absolute agreement about the secret list.

Input: Concise notes of the conversation. It begins with the number of houses in consideration, a number n, in the first line. Then n lines follow. Each is of the following form: First, houses are named, separated by spaces. Then the last word in the line is either 'even' or 'odd'. You may assume that the n lines are enough to tell Tyrion the friendship information for all houses.

Output: Output the number of friends of house Lannister.

Sample Input:

```
4
Baratheon Lannet Stark Targaryan odd
Baratheon Lannet Targaryan odd
Baratheon Stark Targaryan even
Stark Targaryan even
```

Sample Output:

```
1
```

4 Brackets and Question Marks

Task: You are given a sequence that contains three symbols: (,), and ?. The task is to figure out if you can replace the question marks by brackets such that a legal bracket term is generated. A bracket term is legal if at any point in the sequence of (and), the number of (up to that point is greater or equal to the number of) up to that point, and additionally, the total number of (and) is equal.

For this task, you have a computer at your disposal, but it is stone-aged and has very little main memory. It is even overburdened with storing the sequence of brackets and question marks.

Note: Solve this task in C++.

Input: A line containing a sequence of (,) and ?.

Note: Be sure to handle line breaks correctly; each line may have a unix-style (`'\n'`) or a windows-style (`'\r\n'`) linebreak, or (if it is the last line) no linebreak at all.

Output: Output 1 if it is possible to replace all ? such that a legal bracket term is the result, and 0 otherwise.

Sample Input 1:

```
((??))
```

Sample Output 1:

```
1
```

Sample Input 2:

```
((?))
```

Sample Output 2:

```
0
```

5 Express Delivery

Task: You join a company that makes deliveries using smart drones. Each station has a different type of drone with a specific speed and battery capacity. The supply of that type of drone at that station is practically infinite.

Your company does deliveries from one station to another and your task is to find the fastest possible delivery paths. After a drone leaves a station, it can travel until its battery capacity is depleted. At this point, it has to be inside another station to hand over the delivery to another drone. The drones cannot be recharged.

You know the distances between stations, the speeds and capacities as well as a list of possible delivery requests. For each possible request, compute the fastest possible delivery time.

Input: The input has the following form:

- The first line contains two integers: n, the number of stations, and q, the number of deliveries.
- The next n lines contain two integers each: c_i is the capacity in kilometers and s_i is the speed in kilometers per hour of drones at station i.
- The next n lines contain n integers each. The j-th integer on the i-th line gives the distance $d_{i,j}$ from station i to station j in kilometers. The distance -1 indicates that station j cannot be reached directly from station i.
- The next q lines contain two integers each: u_k, the start, and v_k, the end station of delivery request k.

Output: Compute the time y_i in hours required for each request i rounded to three places. The output is a single line $y_1\ y_2\ \ldots\ y_q$ given as floating point numbers.

Sample Input 1:

```
3 1
2 3
2 4
4 4
-1 1 -1
-1 -1 1
-1 -1 -1
1 3
```

Sample Output 1:

```
0.583
```

Sample Input 2:

```
5
4 3
30 60
10 1000
12 5
20 1
-1 10 -1 31
10 -1 10 -1
-1 -1 -1 10
15 6 -1 -1
2 4
3 1
3 2
```

Sample Output 2:

```
0.510 8.010 8.000
```

6 Ornithology

In a research project, you have been studying the flying behavior of a flock of brent geese[1]. Your team has equipped s geese with little sensors that give you their position at discrete points in time (a series of three-dimensional points). The sensors are not completely reliable. You see that some paths have much less points than others, but what you know for sure is that for each goose, you receive at most n points and that those points are in the correct order. We call such an ordered series $P_i = p_{i1}, \dots, p_{im_i}$ of $m_i \leq n$ points a *goose path*. The points are from $\mathbb{R}^3$, but have integer coordinates.

Now you want to compute a representative flight path of the flock, i.e., you want to compute a goose path $R = r_1, \dots, r_n$ that roughly summarizes the paths of all geese in the flock. R does not have to be one of the actually observed goose paths (but it can be).

Now it is not obvious how to even judge how well $R = r_1, \dots, r_n$ fits a different goose path $P = p_1, \dots, p_m$. In order to define this, you map R to P, more precisely, you map the vertices of R to vertices of P. Indeed you compute an *optimal* mapping $M = (i_1, j_1), \dots, (i_t, j_t)$ under the following conditions:

1. The first and the last vertex of R are mapped to the first and last vertex of P, respectively:
$$i_1, j_1 = 1, \; i_t = n, \; j_t = m.$$
2. As we progress along the path, the mapping is neither allowed to move “backwards” nor to skip vertices:
$$\forall \, (i_u, j_u) \in M : \; i_{u+1} - i_u \in \{0, 1\} \text{ and } j_{u+1} - j_u \in \{0, 1\}.$$
3. The mapping does not contain redundancies, i.e. going from one pair of vertices to the next, at least one vertex of the pair changes.
$$\forall \, (i_u, j_u) \in M : \; (i_{u+1} - i_u) + (j_{u+1} - j_u) \geq 1.$$

You claim that $d(R, P) := \min_{\text{all legal} M} \max_{(i_u, j_u) \in M} ||r_{i_u} - p_{j_u}||$ is a nice distance measure for goose paths: One can imagine $d(R, P)$ as the maximum distance of two birds that (in a discretized way) fly along R and P, following an optimal mapping of the two paths.

[1]That’s birds.

This measure looks quite complicated, but at least you have shown that it actually defines a metric on the space of all goose paths[2].

Your task is to compute a 2-approximation to the optimal representative R. The optimal representative R is a goose path for which the maximum of $d(R, P)$ is minimized, where $d(R, P)$ itself stems from an optimal matching between R and P. We call the optimum distance d^*:

$$d^* = \min_R \max_{j \in \{1,\dots,s\}} d(R, P_j)$$

Input: The first line contains s, the second line contains n. The following s lines define the goose paths. Each starts with an integer $m_i \leq n$, the number of points for the ith goose path. This number is followed by the $3 \cdot m_i$ integers which are the ordered m_i points of the ith goose path.

Output: Compute a number that is at least d^* and at most $2d^*$. To avoid rounding problems, **output the square of the number**, i.e., output X^2 with $d^* \leq X \leq 2d^*$.

Sample Input:

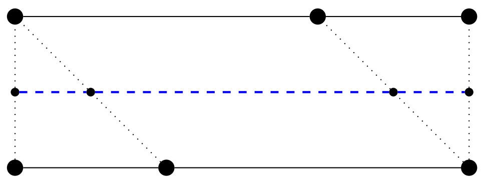
```
2
4
3 0 0 8 2 0 8 6 0 8
3 0 2 8 4 2 8 6 2 8
```

Sample Output:

```
2
```

In this example, $\sqrt{2}$ is the optimum value. Thus, outputting 2 is allowed and optimal. However, solutions up to $2 \cdot \sqrt{2}$ would be permitted, i.e., outputs up to $(2\sqrt{2})^2 = 8$ are accepted.

Here is a picture illustrating the optimum representative path (blue, dashed) and the two matchings with the two input paths (the picture ignores the third coordinate):



7 Pizzeria Advertisement 2

The grand opening of your pizza place was a disaster. You tried to save money on the prints and heavy rainfalls destroyed all of your non-waterproof posters. You do not want to give up on your pizza place and decide to try advertising again. This time however, you have learned from your past mistakes and decide to go for higher quality prints. Since you are evidently low on budget, all you can afford are 6 posters. Is there a way to distribute the 6 posters on road intersections in such a way that everyone that travels along a road will see at least one advertisement?

Input: Your home town is given as a graph where roads are represented by edges and intersections are represented by nodes. The first line of the input contains two numbers, the number of nodes n, and the number of edges m. This is followed by m lines containing two integers $a, b \in \{0, \dots, n-1\}$. Each such pair describes an edge in the graph.

Output: Output `possible` if it is possible to distribute the posters such that everyone that travels along a road will see at least one advertisement, else output `impossible`

[2]This is not completely true because two paths can have distance 0 even though their vertices are not identical (if they repeat different points). However, this is unimportant for the exercise.

Sample Input:

```
7 7
0 1
1 2
2 3
3 4
4 5
5 6
6 0
```

Sample Output:

```
possible
```

8 The Fast and the Furious II*

Task: After your last race in Manhattan you appeared in the list of the most wanted people of the FBI. So you decide that it is better to escape from the United States and go to South America. Unfortunately, you also have to leave your loved Lamborghini behind.

The street racing competitions here are similar to the ones in Manhattan: You and your opponent are given some destinations, must visit them all and the one who returns to the start first wins. But there are also some differences: The street network here is different than in Manhattan, usually you have to visit fewer destinations, and your new car is not faster than your opponent's car.

Remark: It is recommended to solve the problem in `C++` or `Java`.

Input: The first line contains three integers n, m and d. n is the number of crossroads, which are numbered from 0 to $n-1$, m is the number of streets, and d the number of destinations (including the starting point). Then, m lines follow, which describe the streets that connect the crossroads: Each line contains three values a_i, b_i and w_i. a_i and b_i are the two crossroads that are connected by this street, and w_i denotes its length. Finally, in a last line the d destinations are described. Here, the first destination is start (and finish) of the race.

You can assume that $1 \leq n \leq 50000$, $d \leq 20$, and all values are integers. Moreover, you can drive on the streets in both directions.

Output: Output the length of a shortest tour beginning and finishing in the start and visiting all destinations.

Sample Input:

```
9 12 3
0 1 2
0 3 5
1 2 4
1 4 2
2 5 2
3 4 5
3 6 1
4 5 4
4 7 4
5 8 2
6 7 3
7 8 5
0 2 8
```

Sample Output:

```
20
```