Bikes vs Cars

EGOI 2023

Problem author: Nils Gustafsson.

Solution:

We can start by writing down what the numbers $C_{i,j}$ and $B_{i,j}$ mean in a more concise way:

$$C_{i,j} = \max_p(\min_b(W - b))$$

$$B_{i,j} = \max_p(\min_b(b))$$

Here, the `max` is taken over all paths from i to j, and the `min` is over all bike lane widths of edges on the path.

Note that if we let $A_{i,j} = W - C_{i,j}$, then we get rid of the parameter W, because

$$A_{i,j} = \min_p(\max_b(b))$$

So now we only focus on the bike lane width, and our goal is to construct a graph that satisfies all the `minmax`- and `maxmin`-constraints imposed by the numbers $A_{i,j}$ and $B_{i,j}$.

Subtask 1 and 2:

Here it is helpful to make the following observation:

Observation 1: The minimum edge weight in the graph is $min(A_{i,j})$, and the maximum edge weight is $max(B_{i,j})$.

In this subtask, this implies that if $B_{i,j} < A_{i,j}$, then it is impossible.

Otherwise, we can add edges of weight $A_{i,j}$ and $B_{i,j}$ that connect all pairs of vertices.

Subtask 3 ($N \leq 40$)

To solve the case when N is small, we need to find a construction that always works, but uses too many edges.

Observation 2: If there is a valid solution, then the following construction will also work: disregard all pairs of vertices where $A_{i,j} > B_{i,j}$. For all other pairs of vertices, add edges of weight $A_{i,j}$ and $B_{i,j}$.

Here is some motivation why this works:

First, if we have an edge between i and j of weight b, then $A_{i,j} \leq b$ and $B_{i,j} \geq b$, which means that $A_{i,j} \leq B_{i,j}$. So we can never have an edge between two vertices if $A_{i,j} > B_{i,j}$. This means that if a graph constructed as above is disconnected, then there is no solution.

Second, a path in the construction that minimizes the maximum weight between i and j will only use edges $A_{x,y}$ that we added, since they are always smaller than the $B_{x,y}$-edges. But it could happen that the minimum maximum weight ends up smaller than $A_{i,j}$, if there is a path $i, a_1, a_2, \ldots, j$ such that

$$\max(A_{i,a_1}, A_{a_1,a_2}, \ldots) < A_{i,j}$$

However, if such a path existed, then it could also be used in any other construction, so in this case the answer should be `NO` anyway.

Third, similar arguments can be made about the $B_{i,j}$-edges.

So all we have to do to get this subtask is to construct the graph above, and check that it is a valid solution. This check can be done by running an algorithm similar to Floyd-Warshall's, or by using minimum spanning trees.

Subtask 5 (all $B_{i,j}$ are the same)

Let B be the value of all $B_{i,j}$. Remember from subtask 1 that the maximum weight in the graph is the maximum value of $B_{i,j}$, which is B. So in this subtask, we must have that $B \geq \max(A_{i,j})$, otherwise there is no solution. But if this is the case, then we can connect all vertices with edges of weight B and then forget about the `maxmin` constraints.

After that, we only have to focus on the numbers $A_{i,j}$.

Observation 3: If we have a graph that satisfies all $A_{i,j}$-constraints, then the minimum spanning tree of that graph will still satisfy all the $A_{i,j}$-constraints.

This fact is a rather standard trick when it comes to minimum spanning trees. To see why it is true, assume that there exists a path from i to j such that the maximum weight is smaller than the maximum weight of the path along the minimum spanning tree. Then we could remove the largest weight edge on the path along the tree, and replace it with an edge of smaller weight. This would create a smaller spanning tree, which is a contradiction.

So, to solve the subtask, find a minimum spanning tree of the complete graph whose edge weights are $A_{i,j}$, and check that it is a valid solution. There are several algorithms to efficiently find a minimum spanning tree, like Prim's or Kruskal's.

Full score

To get full score, we will put together the things we learned in the previous subtasks.

First, take the construction from Subtask 3. Like we saw in Subtask 5, we can take a minimum spanning tree of this graph, and it will still satisfy the `minmax`-constraints. Similarly, we can take the maximum spanning tree to satisfy the `maxmin`-constraints. Furthermore, if we take the union of these two trees, then we get a solution, if one exists.

To see why this works, note that the minimum spanning tree will only use the $A_{i,j}$-edges added in Subtask 3, and the maximum spanning tree will only use the $B_{i,j}$-edges. Also, a path between i and j that minimizes the maximum weight will only use the edges from the MST, like we saw in Subtask 5, and this value is exactly $A_{i,j}$ like we want (and similarly for $B_{i,j}$).

Summary of how to get 100 points: Create a graph with edges of weight $A_{i,j}$ and $B_{i,j}$ between every pair of vertices such that $A_{i,j} \leq B_{i,j}$, take the union of the minimum and maximum spanning tree, and finally check that this is a valid solution.