Graph

Subtasks 1-4

These subtasks essentially are for the solutions that have the correct idea (see Subtask 5), but with slower implementation.

Subtask 5

According to the last example in the task statement it is clear that the graph may actually be a multigraph, i.e., may contain more than one edge between a pair of vertices, loops, or isolated vertices.

Reading the input carefully allows us to get rid of extraordinary situations:

- Loops define the values of the vertices (0.5 or 1 depending on the color of the edge).
- For the multi-edges it should be checked that all edges are of the same color. If not, there is no solution.
- For each isolated vertex a 0 should be assigned.

Since there may be several connected components, the task should be solved for each component separately. Minimum sums of absolute values found for separate components will sum up to the minimum sum for the whole graph.

Let's investigate a solution for a single connected component.

For each vertex we will assign a tuple (a, b) which corresponds to the value in the form $ax + b$ where $a \in \{-1, 0, 1\}$ and b — any real number. Any vertex may be already processed or not yet processed. Value $a = 0$ means that for the particular vertex the exact value is already known (b). If $a \neq 0$, it means that we still do not know the exact value for the particular vertex — it still contains a variable part.

We start with an arbitrary vertex v_0 and assign $(1, 0)$ to it. This means that the vertex is processed while the exact value is still unknown (we can denote it by a variable x). Now let's process all other vertices by DFS going from already processed vertices via edges.

Let's see what are the possible cases if an already processed vertex v is connected with a vertex u.

We first calculate (a'_u, b'_u), the values for u that follow from values for v and the color of the edge that connects them. Namely, $a'_u = -a_v$ and $b'_u = 1 - b_v$ if the edge is black or $b'_u = 2 - b_v$ if the edge is red.

Then we check whether u is already processed. If not, we assign $(a_u, b_u) = (a'_u, b'_u)$ and proceed with DFS. If, however, the vertex is already processed (we have found a cycle), there are several cases:

- If $a_u = a'_u$ and $b_u = b'_u$, there is no additional information and we proceed with DFS.
- If $a_u = a'_u$ and $b_u \neq b'_u$, there is contradiction and we can stop DFS.

- Otherwise we have $a_u a'_u = -1$, in which case it is possible to establish the value of x: $x_{\text{val}} = (b_v - b_u)/(a_u - a_v)$. Now that we know x_{val} we need to recalculate the values for all the already processed vertices by replacing (a_w, b_w) with $(0, a_w x_{\text{val}} + b_w)$.

It can be seen that during the DFS we may need to replace the values for already processed vertices only once, and after that all further processed vertices will have exact values ($a_v = 0$).

If at the end we have $a_v = 0$ for all vertices, this is the only valid solution.

Otherwise ($|a_v| = 1$) we need to find the value of x giving the overall minimum for the sum of absolute values. Lets take all values of b_v (for $a_v = -1$) and $-b_v$ (for $a_v = 1$) and find the median of them. It can be proved that this is the value of x resulting in the overall minimum (proof left as an exercise). To find the median of N numbers, you can use various algorithms:

- You can sort the numbers and take the middle one ($O(N \log N)$).
- You can use the quickselect algorithm (expected running time $O(N)$, $O(N^2)$ worst-case).
- You can use the median-of-medians algorithm ($O(N)$ worst-case).

Complexity: $O(N + M)$ / $O(N \log N + M)$.