IOI 2024 Solutions: Problem Sphinx

Problem Information

Problem Author: Joshua Lau (Australia)

Problem Preparation: Bartosz Kostka, Mohamed Bakry

Editorial: Bartosz Kostka

Problem Statement

Given a graph with N vertices and M edges, where each vertex has a colour, our goal is to determine the colours by conducting recolouring experiments. A recolouring experiment alters the colours of some vertices and returns the number of monochromatic components in the resulting graph.

Identifying Monochromatic Components

We initially focus on identifying the monochromatic components within the graph without explicitly determining their actual colours (which will give us 50 points).

We construct the monochromatic spanning forest of the graph, where every tree will be a monochromatic component. This can be done inductively. For each new vertex, we perform a binary search to find the appropriate monochromatic component it should be added to. To identify the right component for this new vertex, we temporarily colour all unprocessed vertices to colour N. Now, we should verify whether the new vertex belongs to any already processed component. Then, we can use binary search on the already determined components. We can narrow the binary search by colouring all the components not in the search range to colour N.

Colour Discovery

We can now concentrate on discovering the colours within the pre-determined monochromatic components.

Let us initially examine how to identify colours in the case of a path graph (subtask 3). For a given colour X, we can determine the number of vertices with even and odd indices with this colour separately. Let us first focus on the even vertices. We can modify all the odd vertices to colour N while leaving all the even vertices unchanged. This will give us N monochromatic components. Now let us notice that if we instead assign colour X to each odd vertex, then each even vertex with colour X will decrease the number of monochromatic components by 2 (or 1 if the vertex is at the end of the path), thereby revealing the number of even vertices having colour X. We can replicate this process to determine the count of odd vertices of colour X.

Moreover, we can extend this method to find out the number of vertices with colour X within a specific range by colouring the prefix and suffix outside this range with colour N. Consequently, we can use a binary search to locate the positions of vertices with colour X (first we find the vertex with the smallest index, then find the second smallest, and so on).

Full Solution

Now, let us combine these two approaches.

In the first step, we discovered all monochromatic components, so we can collapse them into single vertices. Then we can build a spanning tree on the collapsed graph, and transform the problem into one where we have two independent sets A and B, with vertices of odd / even depth in this spanning tree. By setting all vertices in A to colour X while leaving all the vertices in B unchanged, whenever a vertex in B also has colour X, it decreases the number of monochromatic components.

Query Count

Finding monochromatic components requires 1 query for each new vertex (to find out if it belongs to any of the already processed monochromatic component) and $\log_2(n)$ queries to identify each new edge of the spanning forest, resulting in a total of $n + (n-1)\log_2(n)$ queries, since there are at most $n-1$ edges in the spanning forest.

Determining the colours requires 2 queries per colour (one for A and one for B) and $\log_2(n)$ queries to identify each colour within the collapsed graph.

Note that every edge we find in the colour-connected spanning forest to form the collapsed graph removes a vertex from the collapsed graph, hence the total number of queries is at most $3n + nlog_2(n)$, which for $n = 250$ is less than $2,750$, required by the problem.