Board Games — Solution

Translation:

Partition the vertices of a graph into the minimum number of subgraphs such that:

1. Each subgraph is a contiguous range $[l_i, r_i]$
2. Each subgraph is connected

Turns out that the following solution works:

> If graph is connected, we are done. If not, there exists some i such that i and $i + 1$ are in different connected components. Solve recursively for subgraphs $[1, i]$ and $[i + 1, N]$.

However, the naive solution is quadratic. Here’s a sketch on how to optimize.

Consider DSU (initially on the whole graph). First, find some splitting point i in $\tilde{O}(\min\{i, N - i\})$ (if it exists). Then, erase the prefix/suffix of vertices (whichever one is shorter). Solve the bigger (”heavy”) half recursively. Then solve the smaller (”light”) half from scratch. From HLD, it’s easy to see that this amortizes to $O(N \log N)$.

However, simple DSU doesn’t allow us to erase prefix/suffix as we please. Let’s fix that.

Consider the fixed midpoint $m = M$ in the list of $2M$ edges sorted by origin vertex. Consider the left and the right half independently. Partition each half into a union of segments (”blocks”) of lengths 2^k, using the binary representation (e.g. if the right half has length $14 = 8 + 4 + 2$, split it into blocks $[m + 1, m + 8], [m + 9, m + 12], [m + 13, m + 14]$). **Block lengths should decrease as you move away from the midpoint.**

Insert the edge blocks into a Rollback DSU in **decreasing** order of their lengths. **We will interweave segments coming from left and right halves**. This way, when erasing some prefix/suffix of edges of length l you can do that by undoing the whole state up until the block of the first removed edge, and then redoing the remaining edges, making sure to keep the “decreasing interweaved powers of two” invariant.

It turns out that such operations amortize the total cost to $O(M \log M)$ add-edge and undo operations. To see this, consider the potential function of some state of the data structure as $\pi(S) = \sum_{x \in S} \log_2(\operatorname{len}(\operatorname{block}(x)))$. Each erasing up until some x does $O(\operatorname{len}(\operatorname{block}(x))$ work to undo and redo, but then potential afterwards gets decremented by at least $\operatorname{len}(\operatorname{block}(x))$, because $\operatorname{block}(x)$ gets “splitted” into smaller blocks with less potential.

This proves a bound of $O(M \log M \log N)$ add/undo operations over the whole process; however, with very rigorous analysis of how potential from an erased prefix/suffix of length l transfers to a new structure in the subsequent recursive calls one can prove a tighter $O(M \log M)$ bound.

Note: The midpoint m remains the same after such a mutation, and it might not be the middle anymore; however, if the prefix/suffix ever intersects m, discard everything and rebuild from scratch (”heavy” segment is now half the original size, so should be okay; also, potential decreases).

In a nutshell, the “data structure” $\mathcal{D}$ supports two operations:

- Splitting a DS of size n into two DS of size k and $n - k$ (from the left or from the right, depending on the side of k wrt the midpoint), which takes $O(\operatorname{block}(k))$ work and decreases the global potential by at least $\operatorname{block}(k)$.
- Rebalancing a completely unbalanced DS of size n, setting the midpoint in the middle $\frac{n}{2}$ position, which takes $O(n)$ work and decreases the global potential by at least $n - 1$.

Total complexity is $O(M \log M \log N + N \log^2 N)$.

($O(M \log M)$ union/undo operations + $O(N \log N)$ find-set operations)