Task: TOW
Tower (author: Justas Klimas)

BOI TORUŃ 2025

BOI 2025, Day 1: Analysis.

First, we try to split the polygon into two parts such that the size of each part is at least $n/3$ (and we know which vertex belongs to which part). To this end, we choose a random pair of vertices u and v. Then, we query about all the triples u, v, w, for $w \in V \setminus \{u, v\}$. This gives us the information whether w is closer to u or v. If the query returns the pair u, v as the answer to the query u, v, w then we discard the pair u, v and repeat the process. Otherwise, one part consists of all vertices that are closer to u than to v, while the other part contains all the remaining vertices. It can be proven that after these steps the size of each part is at least $n/3$. It is straightforward to calculate that the expected number of repetitions is constant.

We now assume that we have a contiguous part of the polygon of size $\leq 2n/3$, and want to recursively recover the order of the vertices there. This can be done similarly as above. We choose a random pair of vertices u and v, and then query about all the triples u, v, w, for $w \in V \setminus \{u, v\}$. If the query returns the pair u, v as the answer to the query u, v, w then we discard the pair u, v and repeat. Otherwise, one part consists of all vertices w that are closer to u than to v, while the other part contains all the remaining vertices. Now we don't check their sizes but instead simply recurse. The expected number of repetitions is still constant (for example, denoting the length of the current part by ℓ, it is enough if u is among the first $\ell/3$ vertices while v is among the last $\ell/3$ vertices, which happens with probability at least $1/9$), and further with constant probability each part consists of at least a constant fraction of the vertices (for example, assuming that u is already among the first $\ell/3$ vertices while v is among the last $\ell/3$ vertices, each part consists of at least $\ell/6$ vertices with probability at least $1/4$). Thus, with constant probability we recurse on two instances smaller by a constant fraction, and otherwise (in the worst case) we repeat the random choice. By standard argument (as in randomised QuickSort) the expected total number of queries is $\mathcal{O}(n \log n)$.

There is one remaining detail: after returning from the recursive calls we need to reverse one or both of the parts in order to make sure that the vertices in the middle are adjacent.

If we have two sequences $x_1, \ldots, x_n$ and $y_1, \ldots, y_m$ and we want to check whether their concatenation $x_1, \ldots, x_n, y_1, \ldots, y_m$ is a valid sequence of vertices in the polygon, we query about the triple x_{n-1}, x_n, y_1. We know that x_{n-1} and x_n are adjacent in the polygon. Therefore, if we get x_n, y_1 in the answer, we know that these two are adjacent as well, so the sequence $x_1, \ldots, x_n, y_1, \ldots, y_m$ is a valid sequence of vertices in the polygon. Otherwise, we know that the sequence is invalid. We can repeat this process for all 4 combinations of the two parts and find the one that is valid.

Overall, we the expected total number of queries is $\mathcal{O}(n \log n)$, the solution needs to be implemented carefully as to keep the constant reasonably small.