IOI 2024 Solutions: Problem Nile

Problem Information

Problem Author: Pikatan Arya Bramajati (Indonesia)

Problem Preparation: Arshia Dadras, Hazem Issa

Editorial: Agustín Santiago Gutiérrez, Pikatan Arya Bramajati

Subtasks hinting key observations

Preliminary observation

Since we have to transport all N artifacts and for each of them pay either $A[i]$ or $B[i]$, we can define a positive cost $C[i] = A[i] - B[i] > 0$ associated with transporting artifact i as a single artifact in its own boat instead of sharing a boat with another artifact. More precisely: Let S be the sum of all the $B[i]$, then if X is the set of artifacts that will be transported as a single artifact in its own boat, the total cost of transportation is $S + \sum_{i \in X} C[i]$. From now on, we will focus on the problem of pairing artifacts into boats in order to minimize the extra cost $\sum_{i \in X} C[i]$ associated with unpaired artifacts.

Subtask 1: $Q \leq 5; N \leq 2000; W[i] = 1$

In this subtask, it is always possible to put any two desired artifacts in the same boat, as $|W[p] - W[q]| = 0 \leq E[j]$ for any j.

Thus, if N is even, the extra cost is 0 as we can pair all artifacts. Otherwise, we must leave exactly one artifact unmatched, which we can choose at will and so it is optimal to choose an artifact i such that $C[i]$ is minimum.

Subtask 2: $Q \leq 5; W_i = i + 1$

For each query, notice that all that we care about is whether $E_j = 1$ or $E_j > 1$.

This is because if $E_j > 1$, we can match all artifacts if N is even, and we can leave any artifact we want as the single unmatched one if N is odd, thus giving exactly the same answer as in subtask 1. The proof is to simply match greedily the two smallest-index artifacts still unmatched that we want to match: notice that this will at most skip one artifact (the specific one we want to skip if N is odd) and so indexes of matched artifacts differ by at most 2, implying $|W[p] - W[q]| \leq 2 \leq E[j]$.

If $E_j = 1$, we can still match all artifacts for even N, but for odd N it might not be possible to freely choose any single arbitrary artifact to leave unmatched. We still want to leave exactly one artifact unmatched: the number of unmatched artifacts will be necessarily odd, and if we had at least 2 unmatched artifacts, we could choose two of them i_1, i_2 such that all artifacts in between are matched (that is, artifact k is matched for all $i_1 < k < i_2$) and change the situation by instead matching i_1 with i_1+1, i_1+2 with i_1+3, …, i_2-1 with i_2, thus strictly improving the solution. Note that there are i_2-i_1-1 artifacts in between in this case, which must be an even number as otherwise those i_2-i_1-1 artifacts would be impossible to match, since they cannot be matched with other artifacts lower than i_1 or greater than i_2 because $E_j = 1$.

Thus, when $E_j = 1$ the whole problem is to identify which artifacts it is actually possible to leave as the single unmatched element, and from those choose the artifact i with smallest $C[i]$. By exactly the same reasoning as in the previous paragraph, exactly the even values of i are possible, so we choose the smallest $C[i]$ from those.

Subtask 3: $Q \leq 5; A_i = 2; B_i = 1$

In this subtask, $C[i] = 1$ for all i. Thus we only need to minimize the number of unpaired artifacts.

Assume that we sort artifacts by $W[i]$, so that now $W[i] \leq W[i+1]$ holds for all i. If we have $W[i+1] - W[i] > E[j]$, then no artifact i or lower can be matched with any artifact $i+1$ or higher, and we can split the problem into two completely independent subproblems. If we do it for all such i, we are left with subproblems inside each of which $W[i+1] - W[i] \leq E[j]$ holds, and so by the same reasoning as in subtask 2 the minimum number of unpaired artifacts for each such subproblem is either 1 or 0, depending only on whether the number of artifacts in the subproblem is even or odd.

Full solution description

The explained solutions to the previous subtasks contain some of the key observations about the problem's structure that are required for a full solution. We will now put them all together efficiently.

First, let's sort the artifacts in increasing order of $W[i]$. We will always refer to artifact indexes with respect to this order. Next, let's denote a configuration of pairings as an array of pairs $[(L_1, R_1), (L_2, R_2), \ldots, (L_k, R_k)]$ such that $L_i < R_i$ and $L_i < L_{i+1}$.

Notice that any configuration where there exists a pair of indices (i, j) such that $L_i < L_j < R_j < R_i$ can always be changed into pairing artifact L_i with L_j, and pairing artifact R_j with R_i, for the same total cost. Similarly, any configuration where there exists a pair of (i, j) such that $L_i < L_j < R_i < R_j$ can always be changed into pairing artifact L_i with L_j and pairing artifact R_i with R_j. Therefore, from now on we will just consider configurations with $R_i < L_{i+1}$. Notice that any such configuration where there exists (L_i, R_i) such that $R_i - L_i \geq 3$ is never optimal, because we could instead pair artifacts L_i with $L_i + 1$ and pair artifact $R_i - 1$ with R_i, to get a lower total cost. Therefore, we only consider configurations with $R_i - L_i \leq 2$.

To do this, we can consider all $O(N)$ pairs of artifacts $(i, i+1)$ and $(i, i+2)$. We sort those pairs by increasing W difference (that is, for pair (a, b) we consider the difference $W[b] - W[a]$).

Also, we sort the queries array E in non decreasing order. Then, a two-pointer algorithm can be used: we iterate the queries, while simultaneously iterating the array of sorted pairs, and at each step we add to the current set of available pairs, all extra pairs satisfying $W[b] - W[a] \leq E[j]$.

We maintain connected components of artifacts using a Disjoint Set Union data structure. Initially, each artifact is its own connected component. Note that connected components are always ranges of consecutive artifacts, so that it is always possible to match artifacts i and $i+1$ belonging to the same component.

When adding a pair $(i, i+1)$, we connect artifacts i and $i+1$ into the same connected component. At any moment, it is possible and optimal to leave 0 unmatched artifacts in even-sized components, and exactly one unmatched artifact in each odd-sized component.

An artifact in an odd-sized component can be the single unmatched artifact, if its index inside the component is even, that is, if its index has the same parity as the minimum index in that connected component. Furthermore, adding a pair $(i, i+2)$ adds artifact $i+1$ as an additional valid candidate for the single unmatched element in its connected component. For each connected component with odd size, we just need to find the minimum $C[i]$ among all candidates.

We can maintain the current total cost, and the candidates summary information for each component, throughout the whole process to answer each query. Candidates summary information for each component consists of the component's size, the minimum index of an artifact in the component, the minimum odd indexed $C[i]$, the minimum even indexed $C[i]$, and the minimum $(i, i+2)$-enabled $C[i]$, all of which can be updated in constant time when merging two components or enabling a pair $(i, i+2)$.

The time complexity of the solution is $O(N \log N + Q \log Q)$

A similar complexity can be achieved using segment tree, and using the same key ideas of decomposing into components: each component will be a range, so that the component's relevant information can be quickly computed by means of the segment tree data structure.

Subtask 4: $Q \leq 5; N \leq 2000$

Get the observation about $R_i < L_{i+1}$. Then, use dynamic programming. Define $dp[i]$ as the minimum total cost for all artifacts from 0 to i. When calculating $dp[i]$, we can try all possible ways to pair artifact i with any artifact $j (j < i)$ having $W[i] - W[j] \leq D$. This does not require the observation about $j - i \leq 2$.

Subtask 5: $Q \leq 5$

Use the same dynamic programming approach as in subtask 4, but only try pairing artifact i with $i-1$ or $i-2$.

Subtask 6: $A_i = 2; B_i = 1$

In this subtask, $C[i] = 1$ for all i. Thus we only need to minimize the number of unpaired artifacts.

We can use the same idea as in the full solution, but the solution for each component is much easier as it is simply 1 unmatched artifact if the component size is odd, or 0 unmatched artifacts if it is even. Thus, component size parity is the only information required from the DSU data structure.