ZIO 2024 Editorials

Problem 1:

Consider the array A. It contains some negative integers, positive integers and zeros. Particularly, we do not care about zeros since they do not affect the sum $A_1 + A_2 + \cdots + A_N$.

It is helpful to see that the subset of all negative integers will be the minimum value in S, i.e. the first value in it. If this is not true, there exists a negative integer that is not included in the subset.

Similarly, the subset of all positive integers will be the maximum value in S.

Now the sum of the elements in A is the sum of negative and positive integers in it. This is nothing but $S_1 + S_{2^N}$.

Note 1: In subpart (a), the sum of all elements in S is 0. It is not hard to see that the sum of elements in A should also be 0.

Note 2: In subpart (b), an easier version of the above explanation is given, where $S_1 = 0$. This means there are no negative integers, and hence, the answer is simply S_{2^N}.

Problem 2:

At the end, H needs to be a consecutive sequence in ascending order. This is a bit tricky. It is helpful to assign $X_i = H_i - i$, because of which, now we want all elements in X to be the same.

Since we need all elements in X to be the same, we can arrange X in any way now. From now onwards, we will assume X is sorted in ascending order.

By exchange argument, we can prove the minimum number of operations to make all elements in X same is equal to $\sum_{i=1}^{N} |X_i - \mathrm{med}|$, where med is the median of X.

Note: In subpart (b), the optimal value of med causes some of the values of H (for example H_1) to be negative. Since this is not allowed, $H_1 = 1$ is optimal in this case.

Problem 3:

Since we want to maximise the sum of distances between all pairs of nodes (i, j), we would like to put the heaviest weight on the edge that is used the most.

Note that in the resulting graph (i.e. a tree), every edge disconnects the nodes into two separate components. Thus, virtually removing an edge E_i disconnects the nodes to two separate components X and Y.

Two nodes in X can still reach each other, even after removing E_i. Similarly, two nodes in Y can still reach each other. But, a node $x \in X$ can no longer reach $y \in Y$. However, it is given that you can reach any node y from any node x. Thus, you must use E_i to each y from x.

This holds true for any 2 nodes x and y such that $x \in X$ and $y \in Y$. Hence, the number of times the edge E_i is used is $|X| \cdot |Y|$, where $|X|$ denotes the number of nodes in X.

Calculate this value for each edge, and accordingly assign the weights, with the heaviest one going to the edge with the maximum value and so on.

Let V denote the value of each edge. Sort V and W in descending order.
The answer will be $\sum_{i=1}^{N-1} V_i \cdot W_i$.

Problem 4:

In general, a subset S is good if the most frequent colour is at most $\lceil \frac{|S|}{2} \rceil$, where $|S|$ is the size of S.

It is easier to calculate the subsets that are not good (in other words, bad). The most frequent colour in that subset is at least $\lceil \frac{|S|}{2} \rceil + 1$. At most one such colour exists for any S, and hence, we will not overcount.

Let's fix the most frequent colour to be c. Let f be the number of occurrences of c in the array given. Since a bad subset should have a size of at least 2, the least number of occurrences of this colour in a bad subset should be at least 2.

We first iterate i from 2 to f, where i denotes the frequency of c in the bad set. The number of ways to choose i from f balls is $\binom{f}{i} = \frac{f!}{i! \cdot (f-i)!}$. Now, the number of balls with colours other than c can be at most $i - 2$, to keep the condition of a bad set satisfied.

Thus, we iterate j from 0 to $i - 2$, where j denotes the frequency of balls without the colour c. The number of ways to choose j from $n - f$ (since there are $n - f$ balls that are not coloured c) is $\binom{n-f}{j}$.

Now, we subtract this from all the possible subsets, which is 2^N.

To sum up the explanation, let there be K colours, with the ith of these having a frequency F_i.
The answer is $2^N - \sum_{color=1}^{K} \sum_{i=2}^{F_{color}} \sum_{j=0}^{i-2} \binom{F_{color}}{i} \cdot \binom{N-F_{color}}{j}$

Note 1: This summation is quadratic in N, but it can be sped up to linear (if you already have $\binom{N}{r}$ calculated) by maintaining prefix sums over $\binom{N-F_{colour}}{j}$.

Note 2: We do not need to pre-calculate all $\binom{n}{r}$ values, which would take n^2 time using Pascals Triangle. Rather we can find $\binom{n}{i+1}$ by multiplying and dividing appropriate terms from $\binom{n}{i}$.

Note 3: If 2 colours have equal frequencies, calculating the number of bad subsets for one of them is enough.