

Simple Problem

Problem author and developer: Alexey Vasilyev

First, let us compute an auxiliary array $closest_{v,b}$ — the distance from vertex v to the nearest vertex whose number contains the set bit b ($0 \leq b < k$). We solve this independently for each bit. When we want to compute $closest$ for some particular bit, we run a multi-source BFS from all vertices that contain this set bit. Alternatively, this can be done using subtree DP. The complexity of this part is $O(nk)$.

Suppose we know the answer set A , and fix in it the vertices at maximum distance from each other. Let these vertices be a and b . Call the path between these vertices the diameter, and let the length of this diameter be the number of edges in it (denote the length by d).

Let us consider two cases:

1. Suppose the diameter length is even. Let vertex m — be the middle of the diameter. Then all vertices from A are at distance at most $\frac{d}{2}$ from m (if this is not the case, then it can be shown that we chose the wrong diameter). This means that if we take into the answer set all vertices at distance at most $\frac{d}{2}$ from m , it will not make things worse for us (that is, the set will not decrease, and the cost of this set will remain the same).

We process this case as follows. Fix a vertex m . Now we want to find the minimum x such that if we take into the answer set all vertices at distance at most x from m , then they will have the OR value we need. It is claimed that x is the maximum value of $closest_{m,b}$ over all bits b . Thus, we find this x and relax the answer with $2x$ (because x is $\frac{d}{2}$).

2. Suppose the diameter length is odd. Then in the middle there is not a vertex, but an edge. Let the vertices of the middle edge be m_1 and m_2 . Then, similarly to the previous item, we say that all vertices of the set A are at distance at most $\frac{d-1}{2}$ from vertex m_1 or from vertex m_2 . And if we take into the answer set all vertices at distance at most $\frac{d-1}{2}$ from m_1 or m_2 , it will not make things worse for us.

Therefore, similarly to the previous item, we iterate over the edge m_1, m_2 . Next, we want to find the minimum x such that if we take into the answer set all vertices at distance at most x from m_1 or m_2 , then they will have the OR value we need. It is claimed that x is the maximum value of $\min(closest_{m_1,b}, closest_{m_2,b})$. We find this x and relax the answer with $2x + 1$ (because x is $\frac{d-1}{2}$).

One can also avoid thinking about the second case separately as follows: add a dummy vertex with value 0 in the middle of each edge and solve the problem on this new tree. After this modification of the tree, the center of the answer diameter will always be a vertex, so one only needs to think about the first case.

The complexity of the second part is $O(nk)$, because we considered $O(n)$ candidates for the center of the diameter, and for each of them computed x in $O(k)$.