Alikhan and studying

- Author — Alikhan Zimanov;
- Developers — Alikhan Zimanov, Fedor Popov, Taimas Korganbayev, Batyr Sardarbekov.

This problem is about calculating the sum of k-th largest distance from a point on an edge to some subset of vertices and summing it up over all integer points on that edge. Let's call this subset of vertices as important vertices. Let's also denote the shortest distance from u to v by $d(u, v)$. It's pretty straightforward to precalculate all pairwise shortest distances $d(u, v)$ by starting n Dijkstra's algorithms from each of the vertex in $O(n(n + m) \log m)$ time, which is fast enough with the problem's constraints.

Group 1 $(S = 1)$ — 8 points

In this test group, exactly one vertex is important (let's call it A), thus $k = 1$ and we need to calculate the sum of distances from all points on the edge (u, v, w) to this important vertex A.

Let's understand how the shortest path from the point x on the edge (u, v, w) to the vertex A might look like. Denote $d(x, u) = \alpha$, then $d(x, v) = w - \alpha$. Obviously,

$$\begin{aligned} d(x, A) &= \min(d(x, u) + d(u, A), d(x, v) + d(v, A)) \\ &= \min(\alpha + d(u, A), w - \alpha + d(v, A)). \end{aligned}$$

If we draw this function on a coordinate plane with x-axis being the value of α (on the interval $[0, w]$) and the y-axis being the value of $d(x, A)$, then we should get a piecewise linear function, consisting of a line segment with slope equal to 1 and a line segment following it with slope equal to -1.

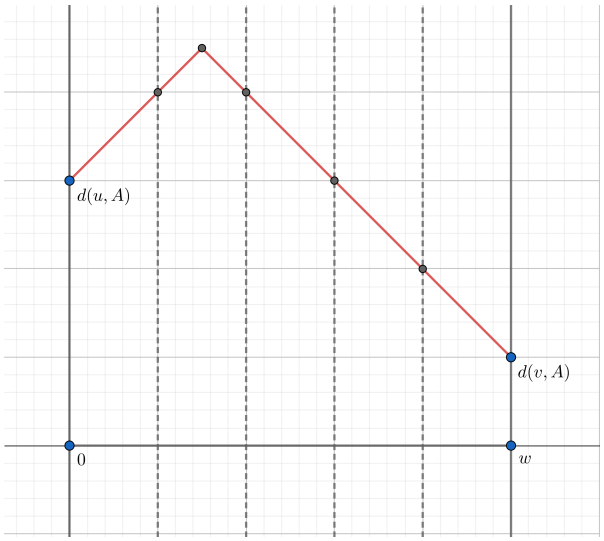

Рис. 1: Plot of the function $f(\alpha) = d(x, A)$, where x is the point on (u, v, w) such that $d(u, x) = \alpha$. The answer is the sum of the values of this piecewise linear function over integer points.

Note that the peak of this function is a point x, such that $d(x, u) + d(u, A) = d(x, v) + d(v, A)$, or, equivalently,

$$\alpha = \frac{d(v, A) + w - d(u, A)}{2}.$$

Hence, we can easily obtain all of the corner points of this piecewise linear function in $O(1)$ time by simply computing the formula stated above and the only thing left to do is to find the required sum of distances. This can be easily done by considering each straight line segment independently and carefully handling double counting. For a single line segment $(L_x, L_y) \to (R_x, R_y)$ we can round the left endpoint up and the right endpoint down. Then, if the line is ascending (which means its slope is 1), the answer will look like:

$$\sum_{x=L_x}^{R_x} (L_y + x - L_x) = L_y \cdot (R_x - L_x + 1) + \sum_{x=0}^{R_x - L_x} x = L_y \cdot (R_x - L_x + 1) + \frac{(R_x - L_x)(R_x - L_x + 1)}{2}.$$

The same formula can be obtained if the line is descending.

To sum up, for each query, we need to find the peak of the mentioned above piecewise linear function, divide it into two straight line segments, calculate answers on each of them, add them up, and subtrack double counting part if needed. This works in $O(n(n + m) \log m + q)$, since all of the computations are done in $O(1)$.

Group 2 ($n, q \le 100, L = 1$) — 8 points

In this test group, the constraints for n and q are pretty small and all of the edges have length of exactly 1.

For this subtask, we only need to consider two cases: $x = u$ and $x = v$. For each of them, we need to sort all of the important vertices by their distance to the corresponding vertex u or v and obtain the k'th largest distance.

Total asymptotics will be $O(n(n+m)\log m + qn\log n)$.

Group 3 ($n, L \le 100, q \le 10^3$) — 12 points

In this test group, the constraint for n is still small, but edge lengths can now be up to 100 (but it's still small enough) and the number of queries is small $q \le 10^3$.

Such constraints lead to a slow solution which iterates over all possible x points and finds k'th distance to important points directly as in the previous subtask. Recall that $d(x, A) = min(\alpha + d(u, A), w - \alpha + d(v, A))$, so for each point x we can calculate all distances from that point to important points and get the k'th largest distance by explicitly sorting these values.

Total asymptotics will be $O(n(n+m)\log m + qLn\log n)$.

Group 4 ($m = n - 1, u_i = i, v_i = i + 1, k_j = 1$) — 10 points

In this test group, our graph is a bamboo (with no constraints on its size) $1 \to 2 \to \ldots \to n$ and $k_j = 1$, which means that we are interested in finding only the farthest vertex from x among important vertices.

Let's consider an edge $(i, i+1, w)$ and fix a point x on it. To find the farthest important vertex from point x, we, obviously, need to consider only the leftmost important vertex among $1, 2, \ldots i$ and the rightmost important vertex among $i{+}1, i{+}2, \ldots, n$. Since they don't depend on the choice of x, we can simply find them by storing all important vertices in a set and get the minimum and maximum value from that set (or even just by iterating over all vertices from 1 to n, since overall constraints allow that). After finding these leftmost and rightmost important vertices l and r, we again end up with a piecewise linear function:

$$\begin{aligned} f(\alpha) &= \max(d(x,i) + d(i,l), d(x,i+1) + d(i+1,r)) \\ &= \max(\alpha + d(i,l), w - \alpha + d(i+1,r)). \end{aligned}$$

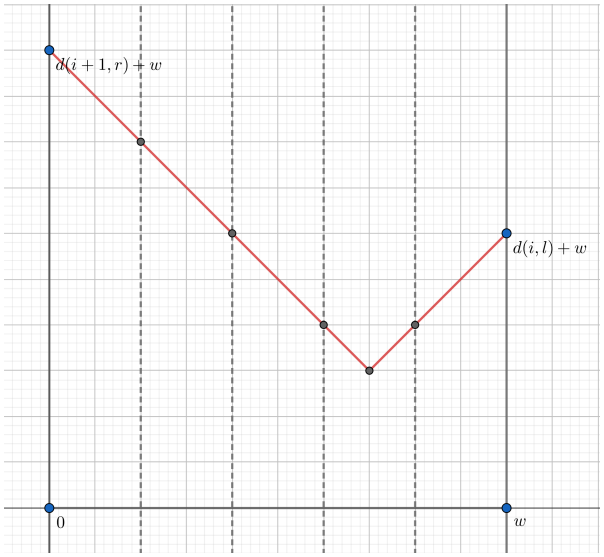

Рис. 2: Plot of the function $f(\alpha) = d(x, A)$, where x is the point on $(i, i{+}1, w)$ such that $d(i, x) = \alpha$, l and r are the leftmost and rightmost important vertices repsectively. The answer is the sum of the values of this piecewise linear function over integer points.

Recall that almost the same construction appeared in the first subtask and can be solved in exactly the same way, except for finding the middle corner point.

Total asymptotics for this test group will be $O(n(n+m)\log m + q\log n)$.

Group 5 ($m = n - 1, k_j = 1$) — 15 points

In this test group, our graph is a tree (with no constraints on its size) and $k_j = 1$, which means that we are interested in finding only the farthest vertex from x among important vertices.

Let's consider an edge (u, v, w) and fix a point x on it. To find the farthest important vertex from x, we, obviously, need to only consider the farthest important vertex from u in the subtree of u and the farthest important vertex from v in the subtree of v, since the farthest path from x passes either through u, or through v to the correspoing farthest vertex in the subtree. To accomplish this, we can iterate over all important vertices A and check in which of the two subtrees they are (if $d(u, A) + w = d(v, A)$, A lies in the subtree of u and vice versa) and update the farthest vertex in that subtree correspondingly. After finding these two vertices, the solution goes exactly the same as in the previous subtask 4.

Total asymptotics will be $O(n(n + m) \log m + qn)$.

Group 6 ($k_j = 1$) — 24 points

In this test group, for each point x we only need to consider the farthest important vertex.

Let's consider an edge (u, v, w) and a point x on it. Recall from Figure 1 how the distance from point x to an important vertex A looks like when we move x from u to v. It is a piecewise linear function that starts as a line with slope 1 and ends as a line with slope -1. Let's denote by $A_1, \ldots, A_l$ all of the important vertices. If we draw such graph for each A_i, we would need to consider only the points on the upper bound of all of these distances $d(x, A_i)$.

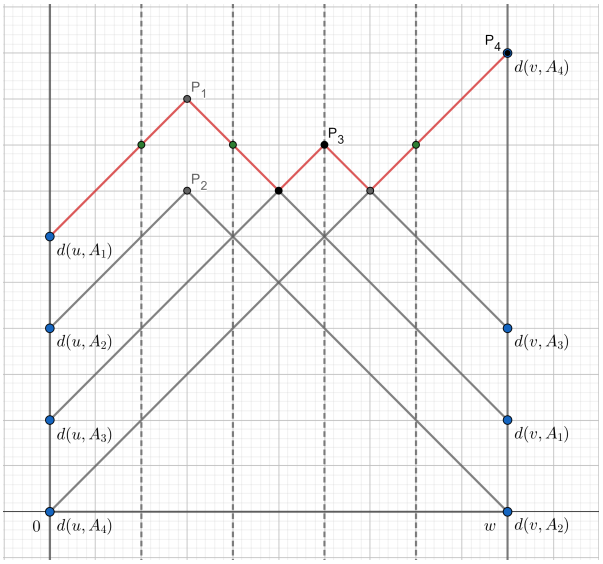

Рис. 3: Plot of the distances $d(x, A_i)$ for all important vertices $A_1, \ldots, A_l$. The upper bound is highlighted in red. The answer is the sum of the values of the upper bound piecewise linear function over integer points.

Note that if there are two vertices A_i and A_j such that $d(u, A_i) \geq d(u, A_j)$ and $d(v, A_i) \geq d(v, A_j)$, then the graph of $d(x, A_i)$ is always superior over the graph of $d(x, A_j)$, thus, we can leave A_j out of the consideration. Let's order important points so that $d(u, A_1) > \ldots > d(u, A_l)$. Then, by the remark above, we get $d(v, A_1) < \ldots < d(v, A_l)$. It's easy to see that each of these remaining vertices will contribute some part in the resulting upper bound. Moreover, their parts will be ordered in exactly the same order $(A_1, \ldots, A_l)$.

To obtain these vertices and their corresponding order, we need to precalculate for each vertex v a list of all vertices, sorted by their distance to v in descending order. This can be done in $O(n^2 \log n)$ time. If we have such a list, then, to obtain all superior points, we can simply iterate over the precalculated list for the vertex u and maintain a stack of vertices as it is done in the algorithm for finding convex hull of a set of points on plane.

To get all of the corner points of the upper bound, we need to calculate the peak graphs for each A_i and the intersections of graphs A_i and A_{i+1}, totally being done in $O(l) = O(n)$ time.

Finally, after computing all of the corner points of the upper bound of the target function, we can do the same thing as in the subtask 1 (calculate the answer independently on each line subsegment and handle double counting).

Total asymptotics will be $O(n(n + m) \log m + n^2 \log n + qn)$.

Group 7 (no additional constraints) — 23 points

In this test group, no additional constraints are given, so we need to solve the full problem.

Let's consider an edge (u, v, w) and a point x on it. Let's say that $(A_1, \ldots, A_l)$ are all important vertices.

Let's draw graphs for each of them as we did in the previous subtask, but in this case we cannot delete inferior vertices since they still can contribute to the answer.

Notice that if we move x from u to v, k'th maximum value moves continuously as a piecewise linear function. It can be proven by considering how this value changes at the points of intersection of some of the graphs and at the peaks of these graphs.

At first glance, it might be very difficult to track this piecewise linear function directly from u to v. To simplify it, we partition the interval $[0, w]$ using the x-coordinates of the peaks for each A_i and do a scanline over the resulting subintervals for x.

Let's look at how the graphs look like on a single scanline subinterval. We must have t lines with slope 1 and $n - t$ lines with slope -1 for some t. Obviously, we need at most k highest lines with slope 1 and at most k highest lines with slope -1, since lower lines will never contribute to the k-th maximum value. We can obtain these lines by maintaining a set of lines with slope 1 and a set of lines with slope -1 during our scanline in $O(n \log n)$ time and get k highest lines from each set in $O(k+\log n)$ time for each subinterval (iterate over a set from its end). By doing so, we obtain a vector of at most k lines with slope 1 and a vector of at most k lines with slope -1, both of them sorted in descending order.

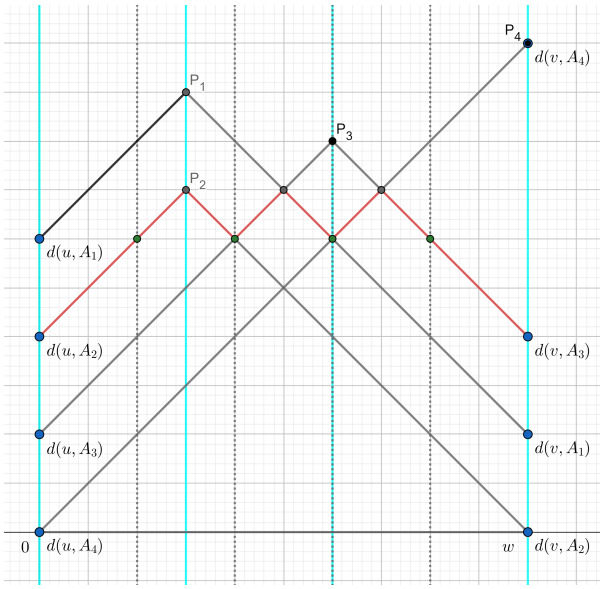

Рис. 4: Plot of the distances $d(x, A_i)$ for all important vertices $A_1, \ldots, A_l$. The k'th distance function for $k = 2$ is highlighted in red, scanline partition events are highlighted in blue. The answer is the sum of the values of the red piecewise linear function over integer points.

Now we'll show that we can completely construct the k-th maximum value function on this subinterval in $O(k)$ time.

First, we need to find its leftmost point. It appears to be the k-th maximum value among the intersections of lines with slope 1 and -1 with the left partition line of the scanline. It can be found in $O(k)$ time using merge sort, since we store these lines in vectors in descending order. Further in the solution, we'll use 0-based indexing of these vectors. Assume that k-th maximum starts on a descending j-th line (see Figure 1 on the second subinterval with $j = 1$). It's easy to see that the first ascending line it is going to intersect will be $i = k - j - 1$ ascending line. It's also easy to see that the order of lines for k-th maximum during the movement of x from left to right will alternate between ascending and descending lines with increasing i and descreasing j. The same can be said if our starting line is ascending. Thus, coupled with the fact that we currently have at most $2k$ lines, all of the corner points of the piecewise linear function of k-th maximum value for this subinterval can be constructed in $O(k)$ time.

After finding exact form of the k-th maximum value function on each subinterval, we can simply merge them and obtain the function over the entire interval $[0, w]$. After that, by doing the same thing as in the previous subtask, we can calculate the final answer. Moreover, since each line can contribute to the k-th maximum function at most once, this function will contain at most $2n$ line segments (but it's irrelevant for our solution).

Total asymptotics will be $O(n(n + m) \log m + qn(k + \log n))$.

Model solution

Author's model solution can be found in the following link:

`https://gist.github.com/dxtvzw/3d6604238d1f267cef28b88ce9f93849`