Code Jam to I/O for Women 2017 - Code Jam to I/O for Women

Analysis: Where Ya Gonna Call?

Where Ya Gonna Call?: Analysis

We can model the input as a weighted undirected graph. The question to be answered is similar to finding the graph's radius. In fact, we are asked to minimize the input graph's radius by possibly adding a single node inside an existing edge. This view is not only a succinct and accurate way to describe the problem, but also a first step towards a solution.

Small dataset

One solution for the Small relies on a simple fact: the result is always an integer multiple of 1/2, and if the optimal location is inside an edge, it is always at a point that is at a distance from one of the incident nodes that is an integer multiple of 1/2. This property is not hard to prove: if the location is in a building, all distances to it are integers. If not, let L be a location and B be a farthest building at distance D. Let B' be the farthest building among those whose minimum path approaches L from the other side of the edge as B's minimum path, with distance D'. If there were no buildings whose minimal path approaches L from the other side, L would not be an optimal location, since we could move L towards B decreasing all distances. If D' < D, we could move the location slightly towards B and decrease the overall minimum, so D' ≤ D, which together with D' ≥ D by definition implies D' = D. Notice that the fractional parts of both D and D' only depend on where L is located on the edge, because the rest of each distance comes from a distance between buildings, which is an integer. Therefore, since the fractional parts of D and D' are equal, the fractional parts of the edge on both sides of L are equal, and thus an integer multiple of 1/2.

With this property in mind, and the limitation that the maximum edge length is 2 in the Small dataset, there are only 3 positions on each edge that can contain an optimal location (at distances 0.5, 1, and 1.5 from one end). For each of these, we can find the farthest building. To do that, we can use Dijkstra's algorithm. We also have to consider the radius of the original graph (which represents choosing a location in a building), which we can do by running the same algorithm starting at each building. Then, we just take the minimum farthest distance from all of those options. The running time of this solution is $O(M\mathbf{B}^4)$ where M is the maximum length of an edge. This is because we need to run Dijkstraâ€™s algorithm 2M-1 times per edge for up to $O(\mathbf{B}^2)$ edges and once per node for $O(\mathbf{B})$ nodes (a total of $O(M\mathbf{B}^2)$ times), and each run takes $O(\mathbf{B}^2)$ time.

Large dataset

The approach outlined for the Small dataset does not work for the Large, because the edges can be really long and thus the number of locations to try is too large, even when restricted to integer multiples of 1/2.

What we can do to simplify the problem is to use binary search. That is, write an algorithm to determine whether there is a location with farthest distance D or less. The statement is clearly false for some interval [0, X) of values for D and true for [X, infinity), so we can simply binary search for X.

The simplification that we obtain is that we can now check for a fixed distance. We start by finding the distance between all pairs of buildings. We can either run Dijkstra's algorithm once per building as mentioned above, or use something simpler for all pairs like Floyd-Warshall's.

We can then iterate over each edge to see whether there is a viable location within it. For a fixed edge, we iterate over each building and see where in the edge a location could be at a distance D or less from that building. To go to a point inside edge (I,J) from building K, there are two options: go from K to I and then move inside the edge, or go from K to J and then move inside the edge. Since we now know the distance from K to I and J, we can calculate the interval of positions within the edge that can be reached with distance D from each end I and J. Then, if those intervals cover the entire edge (by overlapping or because one of them is big enough), then any location inside (I,J) is reachable from K with a distance D or less. Otherwise, there is some interval of unreachable locations. We record such intervals for each building and then check whether the union of all those intervals covers all of (I,J). If it does, no location inside (I,J) is viable. Otherwise, there is at least one that is.

To check if a given set of intervals fully covers another interval A there is a greedy algorithm: sort the intervals by starting point and process them while keeping a current covered upper bound U, which is initialized to the lower bound of A. While U is less than the upper bound of A, for each interval, if its lower bound L is greater than U, then there is an uncovered interval (U, L) and we are done. Otherwise, if the upper bound H of the current interval is greater than U, set U := H. If the iteration finishes due to U becoming larger than A's upper bound, all of A is covered. Otherwise we either found a hole or there is one between the last value of U and the upper bound of A.

For this solution we run Floyd-Warshall's which takes $O(\mathbf{B}^3)$ time and then run a procedure for each of $O(\mathbf{B}^2)$ edges. This procedure iterates all **B** nodes within a binary search, which takes O(log **B**M) to converge (remember M is the maximum length of an edge, so **B**M is an upper bound on the output), which makes it take time $O(\mathbf{B}^3 \log \mathbf{B}M)$ overall.

Notice that since we are binary searching for X, precision is not an issue, as a bad decision due to precision would only give a slightly larger or slightly smaller result. Additionally, we could use the property of the result being a multiple of 1/2 to do all calculations on integers by doubling all edges in the input and dividing by 2 at the very end.

A number of people tried to use ternary search to solve this problem. Ternary search assumes a convex or concave function, and the function in this case (the farthest distance for each point within an edge) is neither. The distance from each point to a single fixed building is indeed a concave function. However, the maximum of many concave functions is not a concave nor a convex function. Some ternary search implementations may suceed in the Small because of the really small number of critical points, which may all be tried even under the flawed assumption.