Kick Start 2022 - Coding Practice with Kick Start Session #2

Analysis: Parcels

View problem and solution walkthrough video

Test Set 1

For Test Set 1, we are able to check all possible locations for the new delivery office in order to find the one that minimizes delivery time. We will do this in two stages. First, we will compute the delivery time for each square given the existing delivery offices. Second, we will try all possible locations for the new office. The precomputation in the first part will allow us to find the delivery time in the second part more efficiently.

For the first stage, we compute the delivery time of a square by iterating over the entire grid and finding the minimum manhattan distance to a square that has a delivery office. This has a time complexity of $O(\mathbf{RC})$ per square for a total time complexity of $O((\mathbf{RC})^2)$.

For the second stage, we iterate over all possible locations for the new delivery office and for each location, search the entire grid for the square with the maximum new delivery time. The new delivery time is the minimum of the delivery time computed in the first part and the manhattan distance to the new delivery office. This also has a time complexity of $O(\mathbf{RC})$ per delivery location for a total time complexity of $O((\mathbf{RC})^2)$, which is sufficient for Test Set 1.

Alternatively, we could skip the first stage if we use a faster way of computing the delivery time for each square such as breadth-first search. See the next section for more details.

Test Set 2

We can use a similar approach for Test Set 2, however we will need a more efficient way to compute the maximum delivery time for a new delivery location. We will do this by solving the following subproblem: given a value of $\mathbf{K}$, can we add a new delivery office so that the maximum delivery time is at most $\mathbf{K}$? The solution to this subproblem will be similar to Test Set 1, however having a target value of $\mathbf{K}$ allows us to identify exactly which squares need to be serviced by the new delivery office. We can use this difference to create a faster solution.

Note that if the answer to the original problem is $\mathbf{K}$, then the answer to our subproblem will be `'No'` for values in the range [1, $\mathbf{K}$-1] and `'Yes'` for values in the range [$\mathbf{K}$, `infinity`]. For these kinds of subproblems, we can use binary search: if a given value works, then it is an upper bound for the answer; otherwise it's a strict lower bound for the answer. Hence, once we have a solution for the subproblem, we can use binary search to solve the original problem. This is a common technique to transform optimization problems into decision problems.

First, we efficiently compute the existing delivery time of every square by inverting the problem: instead of finding the shortest distance to a delivery office for each square, we find the shortest distance to each square from a delivery office. This can be done using a multiple-source, breadth-first search starting at all of the delivery offices. A multiple-source BFS is the same as a regular BFS except that you use multiple starting locations instead of one. This search visits each square at most once, which gives us a time complexity of $O(\mathbf{RC})$.

Second, we identify all of the squares which have a delivery time greater than $\mathbf{K}$ and then determine if there exists a location that is within a distance of $\mathbf{K}$ to each of these squares. In order to do this efficiently, we note that the manhattan distance has an equivalent formula:

dist(($\mathbf{x_1}$, $\mathbf{y_1}$), ($\mathbf{x_2}$, $\mathbf{y_2}$)) = max(abs($\mathbf{x_1}$ + $\mathbf{y_1}$ - ($\mathbf{x_2}$ + $\mathbf{y_2}$)), abs($\mathbf{x_1}$ - $\mathbf{y_1}$ - ($\mathbf{x_2}$ - $\mathbf{y_2}$)))

This formula is based on the fact that for any point, the set of points within a manhattan distance of $\mathbf{K}$ form a square rotated by 45 degrees. The benefit of this formula is that if we fix ($\mathbf{x_2}$, $\mathbf{y_2}$), the distance will be maximized when $\mathbf{x_1}$ + $\mathbf{y_1}$ and $\mathbf{x_1}$ - $\mathbf{y_1}$ are either maximized or minimized.

Hence, we can compute the maximum and minimum values of both $\mathbf{x_1}$ + $\mathbf{y_1}$ and $\mathbf{x_1}$ - $\mathbf{y_1}$ for all squares with a delivery time greater than $\mathbf{K}$. Then, we can try all possible locations for the new delivery office and check if the maximum distance from the location to a square with a current delivery time greater than $\mathbf{K}$ is at most $\mathbf{K}$ in constant time. Hence, we can check if the answer is at most $\mathbf{K}$ with a time complexity of $O(\mathbf{RC})$.

With the binary search, the time complexity becomes $O(\mathbf{RC}\log(\mathbf{R}+\mathbf{C}))$, which is sufficient for the test set. There is a way to improve this to $O(\mathbf{RC})$ time by computing the min/max values mentioned above for all possible $\mathbf{K}$ in a single pass over the grid and then using casework to determine if a viable new delivery office location exists for each $\mathbf{K}$, but this optimization is unnecessary.