Solution for Buses

The Buses task is clearly a graph-theoretic task that is asking for the shortest (or, in this case, cheapest) path, given some constraints. In such tasks, it is usually the best approach to divide the solution into two conceptual steps: first build the graph, then run the appropriate algorithm.

In our task, there were multiple ways how to model the graph. The more natural one is to treat every grid point as a vertex. However, in such a graph it is hard to represent the bus lines: note that once we pay the fee for a bus, we would get to traverse multiple edges in such a graph. Also, it might be possible to get to a given spot in two ways, one of them cheaper, the other shorter, and one can not easily decide which of those ways will be used in the optimal path. Both of these issues show that the naive representation might not be the most suitable one. To find a better representation, we will first analyze the problem to see what we really care about.

Clearly, we will only use each bus line at most once. And obviously, if there is a solution, there is an optimal solution with the following property: Whenever we walk from a bus line X to a bus line Y, we take the shortest path available. (Take any valid cheapest solution. Write down the sequence of bus lines used. If we now construct a new path that uses the same bus lines in the same order, and always uses shortest walks, it will have the same cost and at most the same distance walked.)

Let $D_{X,Y}$ be the shortest distance one has to walk to get from (some spot on) the bus line X to (some spot on) the bus line Y. In our solution, we will start by computing the values $D_{X,Y}$ for each pair of lines X,Y. We will now explain a simple way how to compute these values.

Clearly, each bus line can be seen as a union of axis-parallel segments. If X and Y are two lines, then the distance $D_{X,Y}$ can be computed as the minimum of all distances between a segment P in X and a segment Q in Y. Computing the distance of two segments is a simpler task. If they intersect, the distance is zero. If not, the distance is either the distance between some of their endpoints, or the distance from the endpoint of one segment to its orthogonal projection on the other segment. Using this observation, one can compute the distance between two segments in O(1). Vector and scalar products are a good tool to implement this function in a nice, human readable way. If each of the bus lines has at most N segments, one can compute a single value $D_{X,Y}$ in $O(N^2)$, and thus all the shortest distances in $O(R^2N^2)$.

Having computed the values $D_{X,Y}$, it is easily seen that the exact topology of the bus lines does not concern us any more. And in this way we just found a more suitable graph representation: the vertices of the graph will correspond to the bus lines, the edges to walks between them. Each edge will have an associated length (the length of the walk) and a cost (the cost of buying a ticket for the bus line at its end). For edges entering the goal the cost will obviously be zero.

Now imagine that we would like to implement a complete search that examines all valid paths in our graph. At any moment we would need to know not only the vertex we are at (the bus line we are currently using), but also the total distance we walked so far.

Using this observation, we can finally formulate our task as a shortest path problem. Consider a graph with (R+2) times (D+1) vertices, where each vertex represents a bus line and the distance we had to walk to reach it. From each vertex, we have at most R+1 edges that correspond to walking to other lines and updating the total distance walked. For example, if the walking distance between line X and line Y is 3, then for each valid d we will have an edge from [X,d] to [Y,d+3].

In such a graph, we want to find the cheapest path from the vertex [A,0] to any vertex [B,*]. To do this, we can use Dijkstra's algorithm. If implemented with a priority queue that works in logarithmic time, the time complexity of searching for the path is $O(R^2D \log RD)$. The total time complexity of our solution is therefore $O(R^2N^2 + R^2D \log RD)$.

Both parts of this solution can still be improved. For the first part, sweeping can be used for a more effective way of computing the distances between the bus lines, reducing the time complexity for the first part to $O(R^2N \log N)$. For the second part, one can note that the graph is almost acyclic (acyclic except for cycles created by intersecting bus routes), and thus we can compute cheapest paths simply in $O(R^2D)$. Neither improvement was necessary to obtain a full score.