Homework 17 sample solution

Due 11/09/16

November 3, 2016

The *rook-path problem* accepts an array of n 2D points *data* and returns the length of the shortest *rook-path* from $data[1]$ to $data[n]$, where a *rook-path* is a sequence of moves that start on a point in *data* and move to another point in *data* that is horizontal or vertical.

For example, if $data = \{(0,\ 0), (10,\ 0), (10,\ 1), (0,\ 2), (1,\ 2), (1,\ 1)\}$, the shortest rook-path from $(0,\ 0)$ to $(1,\ 1)$ would have length 4: (0, 0) – (0, 2) – (1, 2) – (1, 1). There is another rook-path of length 20 from (0, 0) to (1, 1), but length 4 is the shortest rook-path. Note that the rook can't move to (0, 1) or (1, 0) from (0, 0), because these points are not in *data*.

Describe an efficient algorithm to compute the shortest rook-path in a given array of points.

```
Input: $data$: set of 2D points
Input: $n$: number of points in $data$
Output: length of shortest rook-path in $data$ from $data[1]$ to $data[n]$
Algorithm: RookPath
$G = \text{WeightedAdjListGraph}(n)$
for $i = 1$ to $n - 1$ do
    for $j = i + 1$ to $n$ do
        if $data[i].x = data[j].x$ then
            $G.\text{AddEdge}(i,\ j,\ |data[i].y - data[j].y|)$
        else if $data[i].y = data[j].y$ then
            $G.\text{AddEdge}(i,\ j,\ |data[i].x - data[j].x|)$
        end
    end
end
return Dijkstra($G$)
```

More clever answers: you can accelerate the graph construction by hashing the points according to their x and y coordinates and adding edges when you detect equal values. Alternatively, you could also sort the points according to x and y coordinates, though you would need to track the start and goal points when sorting.