Analysis of Algorithms

Homework 4

Abraham Murciano

December 21, 2020

1 Dijkstra's Algorithm with negative weights

Part A

Figure 1 shows a graph with negative weights such that if we apply Dijkstra's algorithm to find the shortest path between vertices S and D, it will return the wrong path.

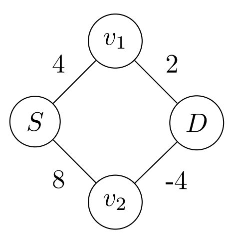

Figure 1: Graph for which Dijkstra doesn't work

Starting off, we assign the unvisited vertices v_1 and v_2 with the distances 4 and 8 respectively, marking S as visited. Then we take the unvisited vertex with the smallest distance, v_1, and check its neighbours, namely D. We assign it the distance 6 and mark v_1 as visited. Now that our destination vertex is the unvisited vertex with the shortest distance, the algorithm would claim that it has finished, with the shortest path going through v_1 with a distance of 6.

However, in reality the shortest path goes through v_2 and has a total distance of $8 - 4 = 4$. This path was not considered by the algorithm because the path to the intermediate vertex v_2 has a larger distance than the path it found first.

Part B

If we take the example graph in figure 1 and modify it so that the edges are directed (away from S or towards D), then that would form a directed acyclic graph for which Dijkstra's algorithm would not work for a similar reason to that of part A.

2 Floyd-Warshall with Negative Cycles

We are to add pseudocode to the Floyd-Warshall algorithm which checks for negative cycles. First, let us take a look at the algorithm.

```
function FloydWarshall(V, E)
    for (u, v) ∈ V × V do                    ▷ Initialise all distances to infinity
        D_{u,v} := ∞
    for (u, v) ∈ E do                        ▷ Apply distances of each edge
        D_{u,v} := Weight(u, v)
    for v ∈ V do                             ▷ Set distance to itself to zero
        D_{v,v} := 0
    for k ∈ V do        ▷ k is a possible intermediate vertex between all (u, v)
        for u ∈ V do
            for v ∈ V do
                D_{u,v} := Min(D_{u,k} + D_{k,v}, D_{u,v})     ▷ Seek shorter path via k
    return D
```

Suppose there exists at least one negative cycle (one such that the weights of its edges sum up to a negative number) in a graph $G = (V, E)$. Now suppose $a, b \in V$ are two distinct vertices within the most negative cycle. At the start of the algorithm, $D_{a,a} = 0$. At some later point in the algorithm, the variables u and v will be referring to a and the variable k will be referring to b. When this occurs, $D_{a,a} = 0$ will be compared to $D_{a,b} + D_{b,a}$. However, we can be certain that $D_{a,b} + D_{b,a} < 0$, because a path $(a, \ldots, b, \ldots, a)$ forms a negative cycle. And thus $D_{a,a}$ will be assigned a negative value.

Therefore, if G contains negative cycles, when the algorithm concludes, it will tell us that $\exists v \in V, D_{v,v} < 0$. So in order to check if there are negative cycles, we can modify the algorithm by calling the following function before returning. If this function returns true, then we know there is a negative cycle and can proceed to throw an exception.

```
function CheckNegativeCycles(V, E)
    for v ∈ V do
        if D_{v,v} < 0 then return true
    return false
```