D4. Graphs

Problem 1.

It makes sense to use Breadth-First Search, since in an unweighted graph it finds the shortest route of k friends. We want the shortest route to make the message more "impactful". The time complexity of Breadth-First Search is O(V+E) where V is number of vertices and E the number of edges.

Problem 2.

If we set strong connections as having a cost of 0, and weak connections as having a cost of 1, running the Dijkstra algorithm will give us the path of the least weak connections.

Problem 3.

```
Int nrOfPathsTo(int v) {
        int counter = 0;
        int distance = distTo[v];
        for(int v = 0; v < G.V(); v++){
                if(distTo[v] == distance){
                counter++;
                }
        }
        Return counter;
}
```

Problem 4.

Kruskal uses relative weight to determine the graph. By decreasing all edges by 1, the relative weight still stays the same.

Problem 5.

A single line of vertices connected each by only one edge (visually similar to a linked list), where each edge gets decreasingly less weight starting from the 0-1 edge with the highest weight.

Here it is visually with “[x]” representing vertices and “=y=” representing edges where x is the name of the vertex and y the weight of the edge:

[0]=5=[1]=4=[2]=3=[3]=2=[4]=1=[5]

Kruskal gives: 4 5, 3 4, 2 3, 1 2, 0 1 while Prim gives the opposite: 0 1, 1 2, 2 3, 3 4, 4 5.