Homework 6

Due 04/18/25

1. Develop an *upper bound* for the complexity of the ExpandedNeighborhood algorithm below, assuming that G is represented using an adjacency list and H is represented using an adjacency matrix. Justify your bound.

```
Input: G = (V, E): graph to analyze
Input: n, m: order and size of G
Output: H: graph with n vertices where the neighbors of each vertex
        are those of distance one or two in G
Algorithm: ExpandedNeighborhood
H = Graph(n)
for v ∈ V do
    for u ∈ N_G(v) do
        H.AddEdge(v, u)
        for w ∈ N_G(u) do
            H.AddEdge(v, w)
        end
    end
end
return H
```

2. A *tree* is a connected, acyclic graph (i.e., no cycles). Describe an algorithm to determine whether a given undirected graph with n vertices and m edges is a tree in $O(n + m)$ time.

   Hint: be careful when detecting cycles—it's easy to "detect" cycles that aren't actually cycles. You may wish to simulate your algorithm on a two- or three-vertex tree to make sure it is correct.

3. 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.