Question 1. (20 marks) We can find the distance between two vertices \( s \) and \( t \) in an undirected graph \( G \) in \( O(n + m) \)-time by building a BFS starting at \( s \). In some cases we can do better by starting a BFS from both \( s \) and \( t \) simultaneously. This is what the following algorithm does. Each vertex will have a “status,” which can have integer values from \(-2\) to \(2\). A status of \(0\) means the vertex is undiscovered (not yet visited), \(-1\) or \(1\) means discovered (discovered from \( s \), respectively \( t \)), and \(-2\) or \(2\) means finished (examined from \( s \), respectively \( t \)). The algorithm terminates when a vertex \( u \) wants to examine a vertex \( v \) of opposite sign (the two BFS meet) by returning the distance from \( s \) to \( t \).

TWO-SIDED-BFS(\( G, s, t \)):

```
if \( s = t \) then return 0
for each \( v \in V[G] - \{s, t\} \) do // initialization
    status[v] ← 0
    d[v] ← ∞
    π[v] ← nil
status[s] ← −1; d[s] ← 0; π[s] ← nil
status[t] ← +1; d[t] ← 0; π[t] ← nil
Q ← \{s, t\} // initialize queue
while Q ≠ ∅ do // main loop
    u ← DEQUEUE(Q)
    for each \( v \in \text{Adj}[u] \) do
        if status[v] = 0 then
            status[v] ← status[u]
            d[v] ← d[u] + 1
            π[v] ← u
            ENQUEUE(Q, v)
        else if status[v] · status[u] < 0 then
            return d[u] + d[v] + 1
    status[u] ← 2 · status[u]
return ∞
```

a. (5 marks) Prove that the algorithm returns the correct distance and that it does so in \( O(n + m) \) time.

b. (5 marks) Modify this algorithm so that it returns a shortest path from \( s \) to \( t \) in \( O(n + m) \) time.

c. (5 marks) For \( k \geq 1 \), let graph \( T_k \) be the complete (rooted) binary tree on \( 2^k − 1 \) vertices. Explain why the worst-case running time for determining the distance between two vertices in \( T_k \) using BFS is \( \Theta(n) = \Theta(2^k) \). Explain why \( \Theta(n) \) is the worst-case running time even if we ignore the initialization steps and count only the main loop of the BFS algorithm.
d. (5 marks) Show that when TWO-SIDED-BFS is used to find the distance between the root \( s \) and a leaf \( t \) of \( T_k \), the time to execute the while loop is \( \Theta(\sqrt{n}) \).

e. (no marks) [This question will not be graded. It is given as an interesting problem you may want to think about.] Show that the worst-case time to execute the while loop is \( \Theta(n^{2/3}) \) when TWO-SIDED-BFS\((T_k, s, t)\) is called for arbitrary vertices \( s \) and \( t \).

Question 2. (20 marks) A bipartite graph \( G = (V, E) \) is a graph where \( V \) can be partitioned into two sets \( V_1 \) and \( V_2 \) (one of which may be empty) such that there are no edges between any two nodes in \( V_1 \) or between any two nodes in \( V_2 \) (that is, all edges of \( G \) have one endpoint in each set). There are many fast algorithms on bipartite graphs, so it is important to be able to recognize whether a given graph is bipartite.

a. (8 marks) Prove that a graph is bipartite if and only if it does not contain an odd cycle. Recall that an odd cycle is a simple cycle that contains an odd number of vertices.

b. (12 marks) Show how to modify DFS so that, given any graph \( G \), it proves in \( \Theta(n + m) \) time whether the graph is bipartite or not. If the graph is bipartite, your algorithm should return two sets \( V_1 \) and \( V_2 \) as described above, but if the graph is not bipartite, your algorithm should return an odd cycle in \( G \).

Question 3. (20 marks) A unicyclic graph is a connected graph with exactly one simple cycle. Give an efficient algorithm that, given a connected undirected non-tree graph \( G = (V, E) \) (represented by its adjacency lists) and a weight function \( w : E \to \mathbb{R} \), finds a unicyclic spanning subgraph of \( G \) of minimum weight. Prove that your algorithm is correct and analyze its worst case time complexity as a function of \( n = |V| \) and \( m = |E| \). (A portion of your mark will be devoted to efficiency, with more marks awarded to more efficient algorithms.)

Question 4. (40 marks) This question is a programming assignment. To see its description follow the link given in the “Assignments” section of the course web page.