Cosc341 - Assignment 3

Dan Skaf, Elijah Passmore, Tom Berg

May 23, 2021

1 Longest Path problem

The **Longest path problem** is the problem for finding the maximum length Simple-path in a graph, $G(V, E)$ with $|V| = n \; : \; n \in \mathbb{N}$. The graph is undirected and all edges weights are assumed to be 1.

1.1 Simple-path

Suppose in our graph, $G(V, E)$, we want the Simple-path between two vertices s and t. The Simple-path is a path between any two nodes in the graph that does not go over the same node twice, and for two consecutive nodes there is a path between them.

As such a simple path between two nodes $s, t \in V$ is the sequence of vertices (v_1,v_2,v_3,....,v_k) that satisfy these conditions:

- $s \; = v_1$ and $t \; = \; v_k$
- Each consecutive nodes (v_i, v_{i+1}) there is an edge $e \; = (v_i, v_{i+1}) \; \in E$
- No node appears more than once in the sequence.

1.2 Longest-simple path and Decision version

The figure below shows the longest simple path, in blue, of a graph.

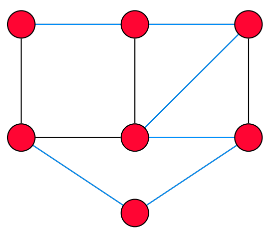

Figure 1: Longest simple path of a graph

In the decision version of the Longest path problem, yes or no version; we input a graph, G, and an integer k and outputs a yes or no if a simplest path of at least length, k, exists in the graph.

1.3 NP

A problem is in NP if we can verify the solution in polynomial time. A corresponding certificate can be produced from the decision version of the Longest path problem. If we can verify this certificate in polynomial time then Longest path problem is in NP. The certificate is a path at least of length k with no duplicate nodes.
To check that the path has no duplicate nodes we can iterate over each node in the path for every node, $O(k^2)$ complexity.
To check the length of the path we just need to count each node and check it is greater than or equal to k, this operation has a $O(k)$ complexity.
In some implementations like most programming languages getting the length of a list is constant complexity $O(1)$, in most cases and the worst case we can check a certificate with $O(n^2)$ complexity, therefore the Longest path problem is in NP.

2 NP-complete

If a problem, X, is in NP-complete and X polynomial reduces to another problem Y in NP, then Y is also in NP-complete.
We have shown that the Longest path problem is NP, all thats left is to show that another problem in NP-complete polynomial reduces to the Longest path problem.

2.1 Hamiltonian path

A Hamiltonian path problem - Determining if a Hamiltonian path:
A path which visits every node in a graph exactly once
exists in a graph, is a known NP-complete problem. It stands to show that the Hamiltonian path problem polynomial reduces to the Longest path problem.

2.2 Reduction to Longest Path Problem

With any instance of the Hamiltonian path problem, an undirected graph of size n, we can instead ask if a Longest path, of size $k = n - 1$, exists. When we set k to $n - 1$ we are asserting that the Longest path problem is equal to a Hamiltonian path.

2.3 Correctness

To prove our reduction we must show:
Correct decision version of Hamiltonian path $\iff$
Correct decision version of Longest Path
That is a instance of the Hamiltonian path problem exits if and only if a Longest path, of $k = n - 1$, exists. We will prove this in both directions from the definitions.

2.3.1 A Longest path exists

- If a Longest Path of size $k = n - 1$ exists, there is a simple path that passes each node exactly once.
- This simple path is a Hamiltonian path, as such if a Longest path does not exist there cannot be a Hamiltonian path.

2.3.2 A Hamiltonian path exists

- If a Hamiltonian path exists, there is a simple path of length $n - 1$
- This is the maximum possible length of any simple path in the graph, as such it is a Longest Path of size $k = n - 1$.
- If a Hamiltonian path does not exist it is impossible for a Longest path of $n - 1$ to exist. Illustrated by a disconnected graph in $fig.2$

Figure 2: Graph without a Hamiltonian path, Longest path on right.

Therefore we have proved our reduction is correct.