School of Information Technology, Deakin University

SIT320 — Advanced Algorithms

Module Six — Graphs II

Overview

ChatGPT Link

GitHub Link

Task 0: Overview

Key Points (Lecture notes)

- **Shortest Path Problem -** If there are no weight we can just comment about how many edges were traversed. Usually used to find the sum of the weights of the shortest path from node A to node B.
  - **Single Source Shortest Path -** Find the shortest path from a **single source** to all other vertices in the graph.
    - **Dijkstra's Algorithm - Greedy algorithm** that finds the shortest path from a **single source to all other vertices** in the graph. Does not work when there is negative weights in the graph. I used a simple list to store the distances from the source to each vertex.
    - **Bellman-Ford Algorithm - Dynamic programming** algorithm that finds the shortest path from a single source to all other vertices in the graph. Loops through all the edges in the graph and relaxes them. **Relaxation** is the process of updating the distance of a vertex from the source vertex. The analogy is when a spring is relaxed it is at its shortest distance. A second pass is done to **detect negative cycles**. If a negative cycle is detected then the algorithm will return a message that there is a negative cycle in the graph. Slower than Djikstra's algorithm and does not use a labelling mechanism for which vertices have been visited. O(m*n) where m and n are the number of edges and vertices respectively.
  - **All Pairs Shortest Path -** Find the shortest path between **all pairs** of vertices in the graph.
    - **Floyd-Warshall Algorithm -** Dynamic programming algorithm that finds the shortest path between all pairs of vertices in the graph. Uses an **inefficient** triple nested loop. Can handle negative weights and cycles, but **may produce the wrong output** for the latter.
- **Interaction with the Lecturer/Peers**

    - Trying to activgely participate in the chat
    - Sometimes I would ask questions and sometimes I would answer questions. Not always the best asnwers though!
    - I want to answer more questions in the future, but I feel the pressure of answering on the spot =)
    - Thanks to Adrian for his input. His explanations were very clear and concise and helped me understand the material better.

Task 1: Turning Dijkstra's into a Bellman-Ford

Key Points

- The challenge of turning Djikstra's into a Bellman-Ford is that Djikstra's is a greedy algorithm and Bellman-Ford is a dynamic programming algorithm
- The first step was to understand what problem Bellman-Ford was trying to solve
    - Bellman-Ford solves the problem of a graph with negative weights and cycles
- The second step was to understand how Bellman-Ford works
    - Bellman-Ford works by relaxing all the edges in the graph n-1 times where n is the number of vertices in the graph **Djikstra Vs Bellman-Ford**
- The main difference between Djikstra's and Bellman-Ford is that Djikstra's uses a **Greedy** design whereas Bellman-Ford uses a **Dynamic Programming** design **Test Cases**
- The main thing to test for was graphs with negative weights and cycles **Algorithm Analysis**
- The algorithm is O(m*n) where m and n are the number of edges and vertices respectively
- This is worse than Djikstra's which is O((n+m)log(n)) where n and m are the number of vertices and edges respectively

Task 2: Floyd-Warshall Algorithm - The most adaptable but inefficient of the shortest path algorithms

Key Points

- The Floyd-Warshall algorithm is a dynamic programming algorithm that finds the shortest path between all pairs of vertices in the graph
- The key point here is **ALL PAIRS** of vertices
- All pairs means that the algorithm uses a systematic approach to find the shortest path between all pairs of vertices **Floyd-Warshall Algorithm**
- the fact that Floyd-Warshall find all pairs means that it is a **triple nested loop**
- This **negatively** impacts the efficiency of the algorithm **Handling Negative Weights and Cycles**

- Negative weights and cycles are **no problem** for Floyd-Warshall
- The algorithm will still produce the correct output **Algorithm Implementation**
- The algorithm is implemented using a triple nested loop
- The first loop is used to iterate through the vertices
- The second loop is used to iterate through the rows of the adjacency matrix
- The third loop is used to iterate through the columns of the adjacency matrix **Test Cases**
- I testing with a simple graph that had no negative weights or cycles
- I then tested the algorithm with a graph with negative weights and cycles **Algorithm Analysis**
- The running time of the algorithm is **O(n^3)** where n is the number of vertices in the graph
- This is **not good** for large graphs
- A real life usage of the algorithm is in the **routing of packets in a network**

Readings

- **Algorithms Illuminated Part 2 & 3 - Tim Roughgarden**
  - Chapter 9: Dijkstra's Shortest-Path Algorithm
  - Chapter 13: Introduciton to Greedy Algorithms
  - Chapter 16: Introduction to Dynamic Programming
  - Chapter 18: Shortest Paths Revisited
- **Introduction to Algorithms - CLRS**
  - Chapter 15: Dynamic Programming
  - Chapter 16: Greedy Algorithms
  - Chapter 24: Single-Source Shortest Paths
  - Chapter 25: All-Pairs Shortest Paths