Graphs II: Introduction

The shortest path of two nodes in a graph depends on whether the edges are weighted or not.

If the edges are not weighted, the shortest path between two nodes is calculated by counting the number of edges between them.

Weighted graphs are represented by: w(u,v), where w is the total weight between node u and node v.

The shortest path in a weighted graph is calculated by finding the minimum cost of path (sum of weights along a path).

Graphs II: Dijkstra's Algorithm

Dijkstra's algorithm helps us find the shortest path between two nodes on a weighted graph. It is also called the Single-Source Shortest Path Algorithm. In this algorithm, each node stores a table which stores the shortest paths to get to the rest of the nodes in the graph.

The tables not only store the length of the shortest path to each node, but it also maps the nodes traversed in the shortest path.

Dijkstra cannot handle negative weights.

When initializing the Dijkstra's algorithm, the distance to the source node is set to 0 and all the other nodes are set to infinity as they are unvisited. The algorithm then explores the node with the smallest known distance neighbouring the source node, the algorithm then updates that node's distance and marks it as visited. The algorithm will continue to traverse through the nodes by choosing the neighbours with the smallest cost of path until it reaches its destination.

Graphs II: Bellman-Ford

The Bellman-Ford algorithm is similar to the Dijkstra algorithm. However, unlike the Dijkstra algorithm, Bellman-Ford can handle negative weights at the expense of being slower than the Dijkstra algorithm.

Unlike Dijkstra's algorithm which visits each node once, Bellman-Ford algorithm iterates over all of the nodes multiple times, finding shorter paths for each node in each iteration.

In Bellman-Ford, the nodes do not have 'explored' statuses.

Graphs II: Running Time Analysis

The time complexity of Dijkstra's algorithm varies depending on the data structure it's implemented on:

Arrays: $O(n^2)$

RB Trees: $O((n+m)\log(n))$

Heap: $O(n\log(n)) + m)$

The time complexity of Bellman-Ford algorithm is O(mxn), where we analyse each edge m, at least n-1 times where n is the number of nodes.

Graphs II: Dynamic Programming First Look

Dynamic Programming is an algorithm design paradigm used to solve optimization problems.

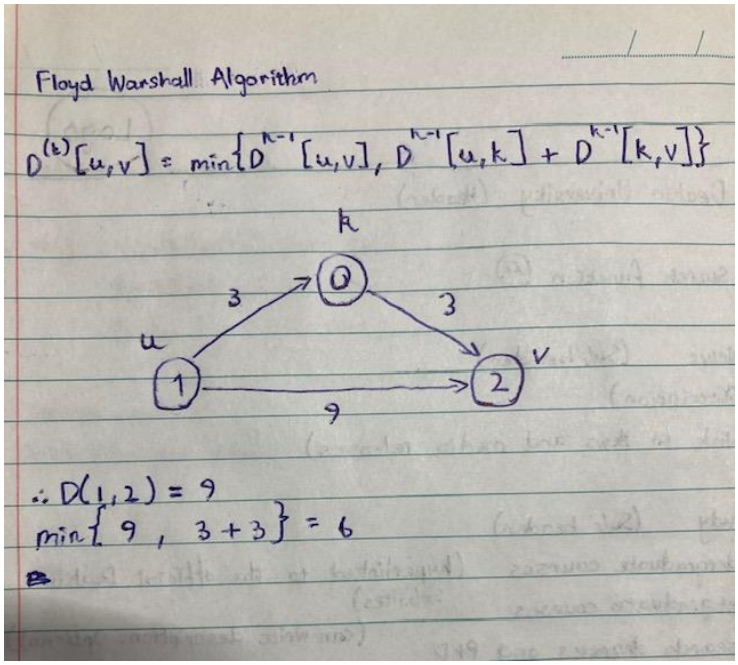
Dynamic Programming is used when the problem has an optimal sub-structure and overlapping sub-problems. Optimal sub-structure means that a big problem can be broken into sub-problems.

Graphs II: Floyd-Warshall

The time complexity of Floyd-Warshall is O(n^3).

Unlike the previous algorithms which solved single source shortest path problems. The Floyd-Warshall algorithm finds the shortest path between all pairs of vertices. These are found by using a table and iterating the formula: $D^{(k)}[u,v] = \min\{D^{(k-1)}[u,v],\ D^{(k-1)}[u,k] + D^{(k-1)}[k,v]\}$

Where $D^{(k-1)}[u,v]$ is the cost of the shortest path between nodes u and v, and $D^{(k-1)}[u,k] + D^{(k-1)}[k,v]$ is the cost of the shortest path from u to k, and then from k to v.



https://www.youtube.com/watch?v=NdBHw5mqIZE