Readings

- [Lecture Notes Chapter 20: Dijkstra’s Algorithm]

Review: Single Source Shortest Path (SSSP) Problem

In the Single Source Shortest Path (SSSP) Problem, we are given a graph $G = (V, E)$, and we want to find a shortest path from a given source vertex $s \in V$ to each vertex $v \in V$. From Recitation 6, we proved that BFS solves this problem for unweighted graphs. However, for weighted graphs, we need something a little more robust. Thus, Dijkstra’s algorithm is an algorithm that solves this problem for graphs with non-negative edge weights; some relevant definitions are as follows:

**Shortest Path:** A path from vertex $s$ to vertex $t$ with the property that no other such path has a lower total edge weight.

**Negative Weight Cycle:** A cycle with weights that sum to a negative number.

Review: Dijkstra’s Algorithm

Dijkstra’s algorithm finds the shortest path between two given vertices in a weighted graph, assuming that the graph’s edge weights are **non-negative**. The running time of the algorithm is $O(|E| \log |V| + |V| \log |V|)$ when the graph is implemented using adjacency lists. The pseudocode for the algorithm is given below:

```plaintext
Dijkstra(G, s)
    for each v ∈ V do
        dist[v] = ∞
        parent[v] = NIL
    dist[s] = 0
    S = ∅
    Q = min-priority queue on all vertices, keyed by dist value
    while Q is not empty do
        u = Extract-Min(Q)
        S = S ∪ {u}
        for each v ∈ Adj[u] do
            if v ∈ Q and dist[v] > dist[u] + w(u, v) then
                dist[v] = dist[u] + w(u, v)
                parent[v] = u
```

---

1
Runtime Analysis

The running time of Dijkstra’s algorithm has two terms: $|E| \log |V|$ and $|V| \log |V|$. We first consider the $|V| \log |V|$ term: each EXTRACT-MIN operation takes $O(\log |V|)$ time, and it is called $|V|$ times because there are $|V|$ vertices.

The $|E| \log |V|$ term has to do with the relaxation step of Dijkstra’s algorithm. Each edge examined may result in a relaxation of the neighboring node in the heap — a DECREASE-KEY operation that takes $O(\log |V|)$ time. The number of vertices examined in line 8 above is bounded by the total degree of all vertices, as each vertex is added and popped exactly once from the min-heap. This value is $2|E|$ by the Handshaking Lemma, so in the worst-case we have $2|E|$ decrease-key operations, for a total of $O(|E| \log |V|)$.

This analysis works for easily proving our runtime, but we can actually do a better analysis. Each edge $(u, v)$ can only cause one relaxation, not 2 as the Handshaking Lemma suggests. This is because $(u, v)$ is explored only when node $u$ is popped from the min-heap. This means that when $(u, v)$ is explored from node $v$, we know node $u$ has already been removed, so its key cannot be decreased. Hence, the $O(|E| \log |V|)$ term comes from the $O(\log |V|)$ cost of a DECREASE-KEY operation, which is called at most $|E|$ times overall.

Greedy Algorithms

A greedy algorithm is an algorithm that always makes the locally optimal choice — the best available choice at that moment—in order to find the best globally optimal solution. Greedy algorithms do not always yield optimal solutions, but for many problems they do. Note that Dijkstra’s algorithm solves the “Single Source Shortest Path” problem by following this paradigm — it uses a priority queue structure that always yields the node with the shortest distance from the source node when polled. Consider the set $S$ of vertices in Dijkstra’s whose final shortest-path weights from the source have already been determined. At each step in the algorithm, since we ultimately want to find the shortest path from our source, we use our priority queue to make the locally optimal choice by adding the node with the current shortest distance from the source node to the set $S$.

Problems

Problem 1

Find the shortest paths from A:

![Diagram of a graph with vertices A, B, C, D, E and edges with weights 1, 2, 3, 4, 7, 8, 2, 3, 7, 8]  

Problem 2

Does Dijkstra’s algorithm work with negative weights? Why or why not?

Problem 3: True or False

1. Dijkstra’s algorithm will not terminate when run on a graph with negative edge weights.
2. If we double the weights of all edges, then Dijkstra’s algorithm produces the same shortest path.

3. If we square the weights of all edges, then Dijkstra’s algorithm produces the same shortest path.

**Problem 4**

(Adapted from CLRS 24.3-6) We are given a directed graph $G = (V, E)$ on which each edge $(u, v) \in E$ has an associated value $r(u, v)$, which is a real number in the range $0 < r(u, v) \leq 1$ that represents the reliability of a communication channel from $u$ to $v$. We thus interpret $r(u, v)$ as the probability that the channel from $u$ to $v$ will not fail, and we assume that these probabilities are independent. Design an efficient algorithm to find the most reliable path between two given vertices.