Chapter 55

Introduction

Given a graph where edges are labeled with weights (or distances) and a source vertex, what is the shortest path between the source and some other vertex? Problems requiring us to answer such queries are broadly known as ***shortest-paths problems***.

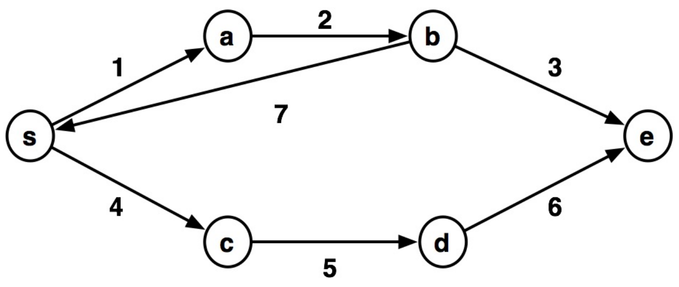
In this chapter, we define several flavors of shortest-path problems and in the next three chapters describe three algorithms for solving them: Dijkstra's , Bellman-Ford's , and Johnson's algorithms. Dijkstra's algorithm is more efficient but it is mostly sequential and it works only for graphs where edge weights are non-negative. Bellman-Ford's algorithm is a good parallel algorithm and works for all graphs but performs significantly more work. Dijkstra's and Bellman-Ford's algorithms both find shortest paths from a single source, Johnson's algorithm extends this to find shortest paths between all pairs of vertices. It is also highly parallel.

1 Path Weights

Consider a weighted graph $G = (V, E, w)$, where V is the set of vertices, E is the set of edges, and $w : E \to \mathbb{R}$ is a function mapping each edge to a real number, or a weight. The graph can either be directed or undirected. For convenience we define $w(u, v) = \infty$ if $(u, v) \notin E$.

Definition 55.1 (Path Weight)**.** Given a weighted graph, we define the ***weight of the path*** in the graph as the sum of the weights of the edges along that path.

Example 55.1. In the following graph the weight of the path $\langle\, s, a, b, e \,\rangle$ is 6. The weight of the path $\langle\, s, a, b, s \,\rangle$ is 10.

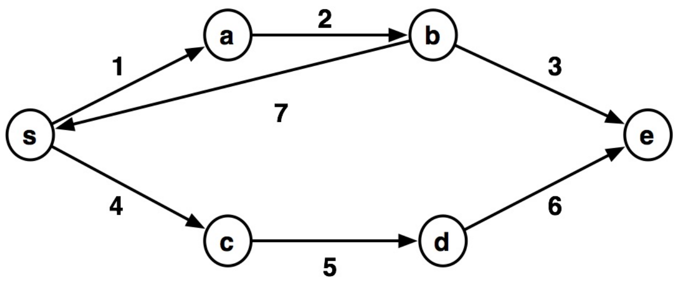


Definition 55.2 (Shortest Paths and Distance)**.** For a weighted graph $G(V, E, w)$ a ***shortest weighted path***, or ***shortest path*** from vertex u to vertex v is a path from u to v with minimal weight. In other words, a shortest path is the path with the smallest weight among all paths from u to v. Note that there could be multiple paths with equal weight; if so they are all shortest paths from u to v.

We define the ***distance*** from u to v, written $\delta_G(u, v)$, as the weight of a shortest path from u to v. If there is no path from u to v, then the distance is infinity, i.e., $\delta_G(u, v) = \infty$.

Note. Historically, the term "shortest path" is used as a brief form for "shortest weighted path" even though the term "shortest path" is inconsistent with the use of the term "weight." Some authors use the term "length" instead of weight; this approach has the disadvantage that length are usually thought to be non-negative.

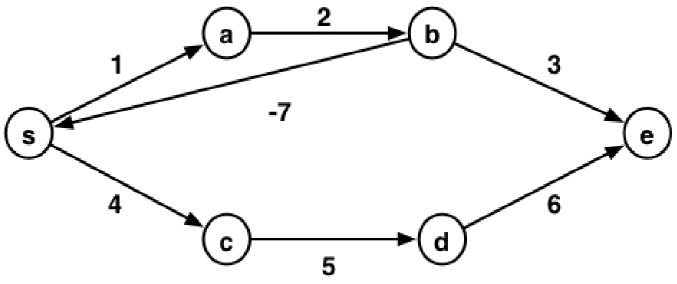
Example 55.2. In the following graph, the shortest path from s to e is $\langle s, a, b, e\rangle$ with weight 6.



Negative Edge Weights. Negative edge weights have significant influence on shortest paths, because they could cause the weight of paths to become non-monotonic: we can decrease the weight of a path by adding a negative-weight edge to the path. Furthermore, negative-weight edges allow for cycles with negative total weight. Such cycles in turn can lead to shortest paths with total weight $-\infty$.

For these reasons, we will need to be careful about negative edge weights when computing shortest paths. As we will discover, even if there are no negative weight cycles, negative edge weights make finding shortest paths more difficult.

Example 55.3 (Negative Weights and Cycles)**.** In the graph below, the cycle $\langle\, s, a, b, s \,\rangle$ has a weight of -4 and the cycle $\langle\, s, a, b, s, a, b, s \,\rangle$ has a weight of -8. This means that the shortest path from s to e has distance $-\infty$.



2 Shortest Path Problems

Shortest path problems come in several flavors, such as single-pair, single-source, and all-pairs problems.

Problem 55.1 (Single-Pair Shortest Paths)**.** Given a weighted graph G, a source vertex s, and a destination vertex t, the ***single-pair*** shortest path problem is to find the shortest weighted path from s to t in G.

Problem 55.2 (Single-Source Shortest Paths (SSSP))**.** Given a weighted graph $G = (V, E, w)$ and a source vertex s, the ***single-source shortest path (SSSP) problem*** is to find a shortest weighted path from s to every other vertex in V.

Problem 55.3 (All-Pairs Shortest Paths)**.** Given a weighted graph, the ***all-pairs shortest path problem*** is to find the shortest paths between all pairs of vertices in the graph.

Problem 55.4 (SSSP$^+$)**.** Consider a variant of the SSSP problem, where all the weights on the edges are non-negative (i.e. $w : E \to \mathbb{R}^+$). We refer to this as the **SSSP$^+$** ***problem***.

Note. Shortest-path problems typically require finding only one of the possibly many shortest paths between two vertices considered. In some cases, we only care about the distance $\delta_G(u, v)$ but not the path itself.

3 The Sub-Paths Property

Definition 55.3 (Sub-Path)**.** A ***sub-path*** of a path is itself a path that is contained within the path.

Example 55.4. If $p = \langle\, a, b, c, e \,\rangle$ is a path in a graph, then the following are some valid subpaths of p:

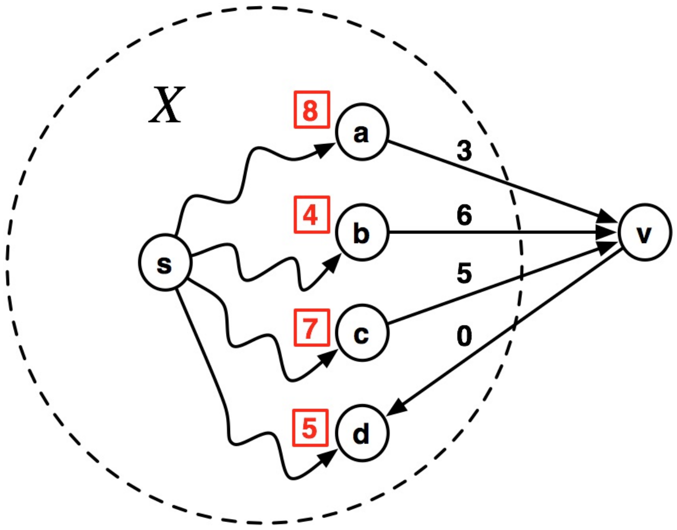
- $\langle\, a, b, c, e \,\rangle$,

- $\langle a, b, c \rangle$,
- $\langle b, c \rangle$, and
- $\langle b \rangle$.

Definition 55.4 (Sub-Paths Property)**.** The ***sub-paths property*** states a basic fact about shortest paths: any sub-path of a shortest path is itself a shortest path. The sub-paths property makes it possible to construct shortest paths from smaller shortest paths. It is used in all efficient shortest paths algorithms we know about.

Example 55.5 (Subpaths property)**.** If a shortest path from Pittsburgh to San Francisco goes through Chicago, then that shortest path includes the shortest path from Pittsburgh to Chicago, and from Chicago to San Francisco.

Applying the Sub-Paths Property. To see how the sub-paths property can help find shortest paths, consider the graph G shown below.



Suppose that an oracle has told us the shortest paths from s to all vertices except for the vertex v, shown in red squares. We want to find the shortest path to v.

By inspecting the graph, we know that the shortest path to v goes through either one of a, b, or c. Furthermore, by the sub-paths property, we know that the shortest path to v consists of the shortest path to one of a, b, or c, and the edge to v. Thus, all we have to do is to find the vertex u among the in-neighbors of v that minimizes the distance to v, i.e., $\delta_G(s, u)$ plus the additional edge weight to get to v. The weight of the shortest path to v is therefore

$$\min\left(\delta_G(s, a) + 3, \delta_G(s, b) + 6, \delta_G(s, c) + 5\right).$$

The shortest path therefore goes through the vertex b with distance 10, which minimizes the weight. We will use this observation in both Dijkstra's and Bellman-Ford's algorithm.

Exercise 55.5. Prove that the sub-paths property is correct.

Solution. We prove the property by using proof by contradiction. Consider a shortest path p and suppose that it has a sub-path q that is not itself a shortest path between its end-points. Replace now the sub-path q with a shortest path between the same end points. This results in a path p' that is shorter than p. But p is a shortest path to start with, so a contradiction.