Lecture 7

All-Pairs Shortest Paths II

Supplemental reading in CLRS: Section 25.3

7.1 Johnson's Algorithm

The Floyd–Warshall algorithm runs in $\Theta(V^3)$ time. Recall that, if all edge weights are nonnegative, then repeated application of Dijkstra's algorithm using a Fibonacci heap gives the all-pairs shortest paths in $\Theta(V^2 \lg V + VE)$ time. If G is not dense (i.e., if $|E|$ is not on the order of $|V|^2$), then Dijkstra's algorithm asymptotically outperforms the Floyd–Warshall algorithm. So our goal for the first half of this lecture will be to reduce the problem of finding all-pairs shortest paths on an arbitrary graph to that of finding all-pairs shortest paths on a graph with nonnegative edge weights.

7.1.1 Reweighting

How might we "eliminate" negative edge weights? The naïve way would be to add some constant N to all the edge weights, where N is chosen large enough to make all the edge weights nonnegative. Unfortunately, this transformation does not preserve shortest paths. Given a path $p : \langle u = v_0, v_1, \ldots, v_k = v \rangle$ from u to v with total weight $w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$, the weight of p in the new graph would be $w(p) + kN$. Thus, a very light path in the original graph might not be so attractive in the new graph, if it contained lots of edges (see Figure 7.1).

We will need to do something smarter; the change in weight must vary from edge to edge. Our goal is to assign to each edge (u, v) a new weight $\widehat{w}(u, v)$, such that the new weight function $\widehat{w}$ has the following two properties:

1. For each pair of vertices $u, v \in V$ and each path $p : u \rightsquigarrow v$, the old weight $w(p)$ is minimal if and only if the new weight $\widehat{w}(p)$ is minimal (among paths from u to v).
2. For all edges (u, v), the new weight $\widehat{w}(u, v)$ is nonnegative.

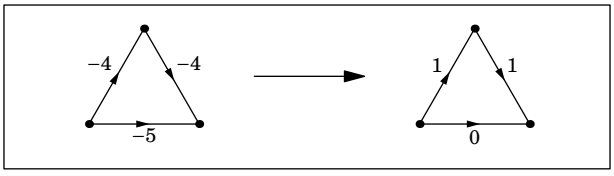

Figure 7.1. Adding a constant to the weight of every edge does not preserve shortest paths.

A clever, simple solution to this problem is as follows. Let $h : V \to \mathbb{R}$ be a scalar-valued function on the vertices. Assign

$$\widehat{w}(u,v) = w(u,v) + h(v) - h(u).$$

This weighting has the property that, for a path $p : \langle u = v_0,\ v_1, \ldots, v_k = v\rangle$, the new weight is

$$\begin{aligned}
\widehat{w}(p) &= \sum_{i=1}^{k} \widehat{w}(v_{i-1}, v_i) \\
&= \sum_{i=1}^{k} w(v_{i-1}, v_i) + h(v_i) - h(v_{i-1}) \\
&= w(p) + \sum_{i=1}^{k} h(v_i) - h(v_{i-1}) \\
&= w(p) + h(v_k) - h(v_0) \\
&= w(p) + h(v) - h(u),
\end{aligned}$$

by telescoping. Thus, the amount by which p changes depends only on the endpoints u and v, not on the intermediate vertices. Consequently, a path $p : u \rightsquigarrow v$ is minimal with respect to $\widehat{w}$ if and only if it is minimal with respect to w. Also, if $u = v$, then $\widehat{w}(p) = w(p)$, so the weight of cycles is unchanged. Thus, negative-weight cycles are preserved, so the distance from u to v with respect to $\widehat{w}$ is $-\infty$ if and only if the distance from u to v with respect to w is $-\infty$.

Our task now is to choose an appropriate function h so that the edge wieghts $\widehat{w}$ become nonnegative. There is a magical answer to this question: Let s be a new vertex, and construct an augmented graph $G' = (V', E')$ with $V' = V \cup \{s\}$. Take E' to consist of all the edges in E, plus a directed edge from s to each vertex in V (see Figure 7.2). Then let

$$h(v) = -\delta(s,v),$$

where $\delta(s,v)$ is the weight of the shortest path from s to v. Note that Proposition 6.1 implies the "triangle inequality"

$$\delta(s,v) \le \delta(s,u) + \delta(u,v)$$

(with equality if one of the shortest paths from s to v passes through u). Since $\delta(u,v) \le w(u,v)$, we have

$$\delta(s,v) \le \delta(s,u) + w(u,v).$$

Hence

$$\begin{aligned}
\widehat{w}(u,v) &= w(u,v) + h(v) - h(u) \\
&= w(u,v) - \delta(s,v) + \delta(s,u) \\
&\ge w(u,v) - \big(\delta(s,u) + w(u,v)\big) + \delta(s,u) \\
&= 0.
\end{aligned}$$

Applying the weights $\widehat{w}$ to G, we obtain the following algorithm, due to Johnson:

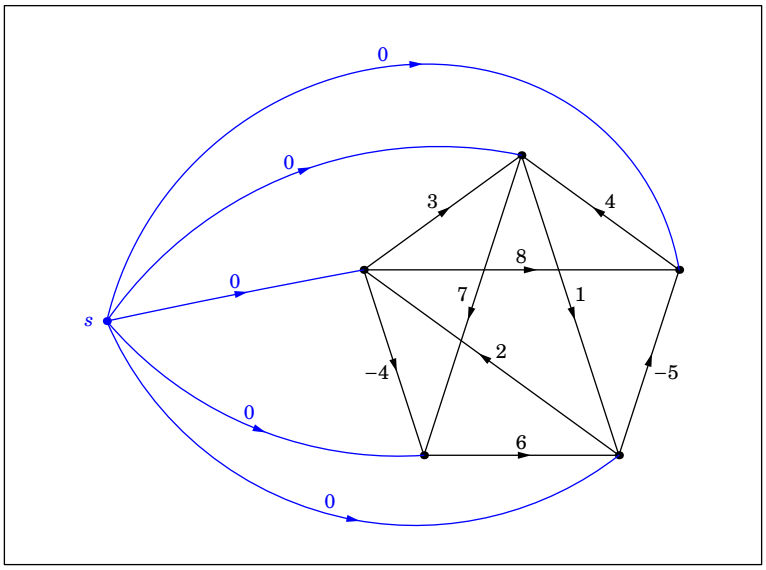

Figure 7.2. The augmented graph G' in Johnson's algorithm, consisting of G together with a special vertex s from which there emanates an edge of weight 0 to each other vertex.

Algorithm: JOHNSON(G)

1 Construct the augmented graph G', where $G'.V = G.V \cup \{s\}$ and $G'.E = G.E \cup \{(s,v)$ with $G'.w(s,v) = 0$, for each $v \in G.V\}$
2 **if** BELLMAN–FORD(G', s) = FALSE **then**
3 ▷ if BELLMAN–FORD complains about a negative-weight cycle
4 **error** "The input graph contians a negative-weight cycle."
5 **else**
6 **for each** vertex $v \in G.V$ **do**
7 Set $h(v) \leftarrow -\delta(s,v)$, where $\delta(s,v)$ is as computed by BELLMAN–FORD
8 **for each** edge $(u,v) \in G.E$ **do**
9 $\widehat{w}(u,v) \leftarrow G.w(u,v) + h(v) - h(u)$
10 Let $D = \big(d_{uv}\big)$ and $\Pi = \big(\pi_{uv}\big)$ be new $n \times n$ matrices
11 **for each** vertex $u \in G.V$ **do**
12 Run DIJKSTRA($G, \widehat{w}, u$) to compute $\widehat{\delta}(u,v)$ for all $v \in G.V$
13 **for each** vertex $v \in G.V$ **do**
14 ▷ correct the shortest path weights
15 $d_{uv} \leftarrow \widehat{\delta}(u,v) - \big(h(v) - h(u)\big)$
16 $\pi_{uv} \leftarrow$ the predecessor of v as computed by DIJKSTRA($G, \widehat{w}, u$) above
17 **return** D

The only steps in this algorithm that take more than $O(E)$ time are the call to BELLMAN–FORD on line 2 (which takes $\Theta(VE)$ time) and the $|V|$ calls to DIJKSTRA on lines 11–16 (each of which takes $\Theta\big(V \lg V + E\big)$ time using a Fibonacci heap). Thus, the total running time of JOHNSON is $\Theta\big(V^2 \lg V + VE\big)$.

7.2 Linear Programming

Linear programming is a very general class of problem that arises frequently in diverse fields. The goal is to optimize a linear objective function subject to linear constraints (equalities and inequalities). That is, we want to maximize or minimize the function

$$f(x_1, x_2, \ldots, x_n) = a_1 x_1 + a_2 x_2 + \cdots + a_n x_n = \sum_{j=1}^{n} a_j x_j$$

subject to the constraints

$$g^{(i)}(x_1, \ldots, x_n) = b_1^{(i)} x_1 + b_2^{(i)} x_2 + \cdots + b_n^{(i)} x_n = \sum_{j=1}^{n} b_j^{(i)} x_j = p^{(i)}$$

and

$$h^{(i)}(x_1, \ldots, x_n) = c_1^{(i)} x_1 + c_2^{(i)} x_2 + \cdots + c_n^{(i)} x_n = \sum_{j=1}^{n} c_j^{(i)} x_j \leq q^{(i)}.$$

In vector form, we want to maximize or minimize

$$\mathbf{a}^T \mathbf{x}$$

subject to the constraints

$$B\mathbf{x} = \mathbf{p} \quad \text{and} \quad C\mathbf{x} \leq \mathbf{q}$$

(meaning that the ith entry of $C\mathbf{x}$ is less than or equal to the ith entry of $\mathbf{q}$ for every i). Here B and C are matrices with n columns (and any number of rows).

7.2.1 Application: Bacterial Growth

One of the many applications of linear programming is the problem of maximizing bacterial growth subject to biochemical constraints. Consider the transport pathways of a bacterial cell, which we model as a graph embedded in three-dimensional space (see Figure 7.3). Let x_i be the flux through edge i.

- At steady-state, the net flux through each vertex of the graph is zero, as wastes are not building up along the cell pathways. This amounts to a linear constraint at each vertex, e.g., $x_1 - x_2 - x_3 = 0$ in the picture.
- Each edge has a minimum (namely zero) and a maximum rate at which molecules can pass through. This amounts to a linear constraint $0 \leq x_i \leq v_{\max}^{(i)}$ at each edge.
- The objective function is the sum of fluxes contributing to biomass production (e.g., synthesis of amino acids and proteins).

The availability of efficient solutions to linear programming problems has been valuable in the engineering of new bacterial strains with improved capabilities, e.g., to produce biofuels.

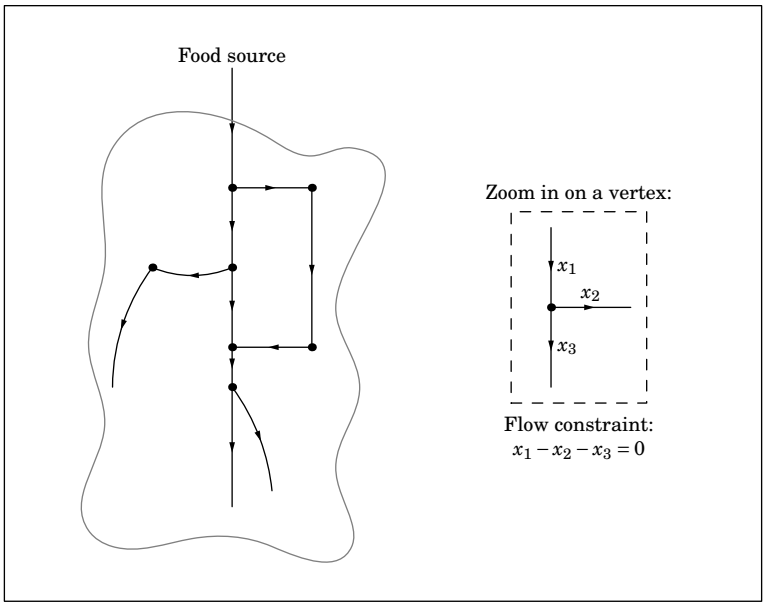

Figure 7.3. Bacterial cell, with a graph representing its transport pathways. At steady-state, the net flux through each vertex is zero.

7.2.2 Difference Constraints as Graphs

It turns out that shortest path problems are related to some forms of linear programming problems. Consider a system of linear constraints, each having the form

$$x_j - x_i \leq b_k,$$

in other words, a system of the form

$$A\mathbf{x} \leq \mathbf{b},$$

where A is an $m \times n$ matrix, $\mathbf{x}$ is an n-dimensional vector and $\mathbf{b}$ is an m-dimensional vector, and where each row of A contains one 1, one -1, and all other entries are zero. As a concrete example, let's consider the system

$$\begin{pmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \leq \begin{pmatrix} 0 \\ -1 \\ 1 \\ 5 \\ -3 \end{pmatrix}.$$

This is equalivalent to the inequalities

$$\left\{ \begin{array}{l} x_1 - x_2 \leq 0 \\ x_1 - x_4 \leq -1 \\ x_2 - x_4 \leq 1 \\ x_3 - x_1 \leq 5 \\ x_4 - x_3 \leq -3 \end{array} \right\}.$$

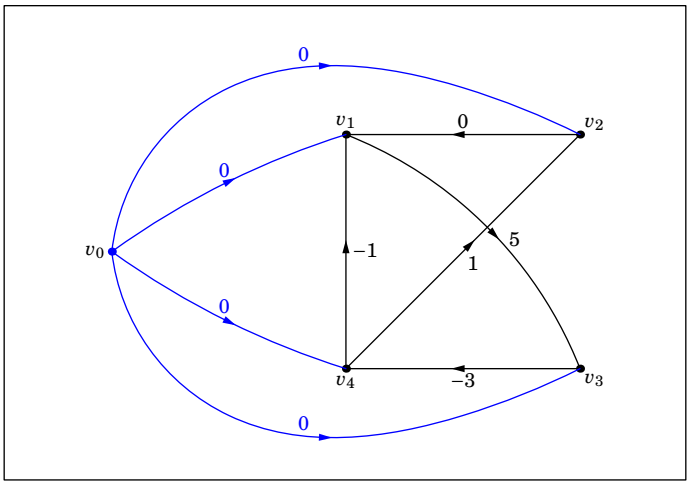

Figure 7.4. Constraint graph for the system of difference constraints $A\mathbf{x} \le \mathbf{b}$, where A and $\mathbf{b}$ are as in our example.

Two plausible solutions to this system are

$$\mathbf{x} = \begin{pmatrix} -5 \\ -3 \\ 0 \\ -4 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} 0 \\ 2 \\ 5 \\ 1 \end{pmatrix}.$$

In general, it's not hard to see that adding any multiple of $(1,1,1,1)^T$ to a solution produces another solution.

Let's now construct a graph G, called the *constraint graph* for this system (see Figure 7.4). The vertex set of G is $V = \{v_0, \ldots, v_n\}$. Draw a directed edge with weight 0 from v_0 to each other vertex. Then, for each difference constraint $x_j - x_i \le b$, draw an edge from v_i to v_j with weight b.

Theorem 7.1 (Theorem 24.9 of CLRS)**.** *Given a system $A\mathbf{x} \le \mathbf{b}$ of difference constraints, let $G = (V, E)$ be the corresponding constraint graph. If G contians no negative-weight cycles, then*

$$\mathbf{x} = \begin{pmatrix} \delta(v_0, v_1) \\ \delta(v_0, v_2) \\ \vdots \\ \delta(v_0, v_n) \end{pmatrix}$$

is a feasible solution. If G does contain negative-weight cycles, then there is no feasible solution.

Proof. Assume there are no negative-weight cycles. Since shortest paths satisfy the triangle inequality, we have

$$\underbrace{\delta(v_0, v_j)}_{x_j} \le \underbrace{\delta(v_0, v_i)}_{x_i} + \underbrace{w(v_i, v_j)}_{b}.$$

Rearranging, we obtain

$$x_j - x_i \le b.$$

Next, consider the case of a negative-weight cycle. Without loss of generality, assume the cycle is of the form $c = \langle v_1, v_2, \ldots, v_k, v_1 \rangle$. Then any solution $\mathbf{x}$ to the system of difference constraints must satisfy

$$
\begin{array}{rrcll}
 & x_2 - x_1 & \leq & w(v_1, v_2) & \\
 & x_3 - x_2 & \leq & w(v_2, v_3) & \\
 & & \vdots & & \\
 & x_k - x_{k-1} & \leq & w(v_{k-1}, v_k) & \\
+ & x_1 - x_k & \leq & w(v_k, v_1) & \\
\hline
 & 0 & \leq & w(c) & < \quad 0,
\end{array}
$$

a contradiction. Therefore there cannot be any solutions to the system of difference constraints. □

6.046J / 18.410J Design and Analysis of Algorithms
Spring 2012