

LECTURE 16

Friday February 10, 2017

*based on notes by Denis Pankratov*Graph Algorithms (*continued*)All-pairs Shortest Path (*continued*)**Input:** $G = (V, E)$ (directed or undirected) $C : E \rightarrow \mathbb{R}$ **Output:** $dist[u][v], \forall u, v \in V, dist[u][v] = d(u, v)$ *Floyd-Warshall* solves this in $\mathcal{O}(|V|^3)$.Can do better for sparse graphs (*i.e.* $|E| \ll |V|^2$)!We would like to run *Dijkstra*, but we have *negative edges*.*Johnson's algorithm* is based on this idea:

1. Reweighting procedure that makes all edge weights *positive* & preserves shortest paths.
Can be done in $\mathcal{O}(|V||E|)$
2. Run *Dijkstra* $|V|$ times. Takes $\mathcal{O}((|V|^2 + |V||E|) \log |V|)$ using binary heaps

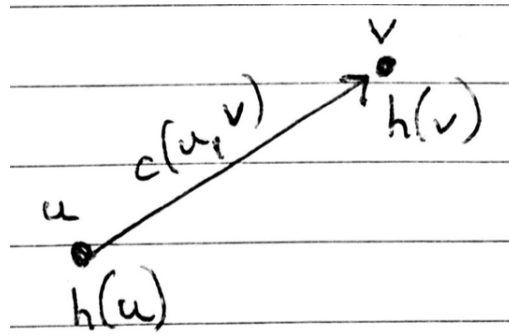
In particular, if $|E| = \mathcal{O}(|V|)$, runtime $\mathcal{O}(|V|^2 \log |V|)$.Exercise: Subtracting most negative weight from every weight of edges does *not* preserve shortest paths.

Find an example.

General reweighting procedure

$$G = (V, E), c : E \rightarrow \mathbb{R}$$

$$h : V \rightarrow \mathbb{R}$$

Figure 1: *Example*

$$\tilde{c}(u, v) = c(u, v) + h(v) - h(u)$$

Claim: (v_0, v_1, \dots, v_k) is a shortest path with respect to (w.r.t.) $c \Leftrightarrow$ it is a short path w.r.t \tilde{c}

Proof:

$$\begin{aligned} \tilde{c}(v_0, \dots, v_k) &= \sum_{i=1}^k \tilde{c}(v_{i-1}, v_i) \\ &= \sum_{i=1}^k c(v_{i-1}, v_i) + h(v_i) - h(v_{i-1}) \\ &= c(v_0, \dots, v_k) + h(v_k) - h(v_0) \end{aligned}$$

Since $h(v_k)$ & $h(v_0)$ are independent of the actual path, this completes the proof.

Claim: $v_0, v_1, \dots, v_k = v_0$, *negative cycle* w.r.t $c \Leftrightarrow v_0, \dots, v_k = v_0$, *negative cycle* w.r.t \tilde{c}

Proof: $\tilde{c}(v_0, \dots, v_k) = c(v_0, \dots, v_k) + h(v_k) - h(v_0) = c(v_0, \dots, v_k)$

Need to construct h so that $\tilde{c} \geq 0$

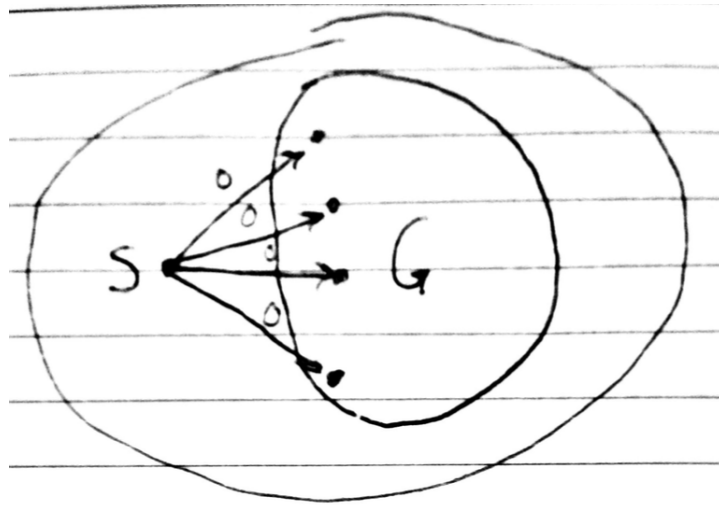
Augment $G = (V, E)$

Pick a new node $s \notin V$

Add s to V

Add edges (s, u) to E for every $u \in V$

Set weight of (s, u) to 0

Figure 2: *Visualization*

$$G' = (V', E'), c'$$

$$V' = V \cup \{s\}$$

$$(u, v) \in E' \Leftrightarrow (u, v) \in E \text{ or } u = s$$

$$C'(u, v) = \begin{cases} c(u, v), & (u, v) \in E \\ 0, & \text{if } u = s \end{cases}$$

Run *Bellman-Ford* on G' from s .

- If *negative cycle* is detected, report it & terminate
- Otherwise, $h(u) = d(s, u) \leq 0$

Correction: $\tilde{c}(u, v) = c(u, v) + h(u) - h(v)$

Proof: Let $(u, v) \in E$. By optimal substructure, $d(s, v) \leq d(s, u) + c(u, v)$

$\Rightarrow c(u, v) + d(s, u) - d(s, v) \geq 0$, where the whole thing is $\tilde{c}(u, v)$ since $d(s, u) = h(u)$, $d(s, v) = h(v)$

Summary

Problem	Algorithm	Runtime
Single source $c = 1$	BFS	$\mathcal{O}(V + E)$
Single-source $c \geq 0$	<i>Dijkstra</i>	$\mathcal{O}((V + E) \log V)$
Single-source c arbitrary	<i>Bellman-Ford</i>	$\mathcal{O}(V E)$
All-pairs c arbitrary	<i>Floyd-Warshall</i>	$\mathcal{O}(V ^3)$
All-pairs c arbitrary	<i>Johnson's Algorithm</i>	$\mathcal{O}((V ^2 + V E) \log V)^*$

* using binary heaps

Network Flows

Input: $G = (V, E)$ directed graphs

$c : E \rightarrow \mathbb{R}_{>0}$ capacities

$s, t \in V$, s - source, t - sink/terminal

No backward edges:

$$(u, v) \in E \Rightarrow (v, u) \notin E$$

No edges into s

No edges out of t

Definition: A flow $f : E \rightarrow \mathbb{R}$ so that:

$$1. \forall (u, v) \in E, 0 \leq f(u, v) \leq c(u, v)$$

$$2. \forall v \in E, f^{out}(v) = f^{in}(v)$$

$$f^{out}(v) = \sum_{u:(v,u) \in E} f(v, u)$$

$$f^{in}(v) = \sum_{u:(u,v) \in E} f(u, v)$$

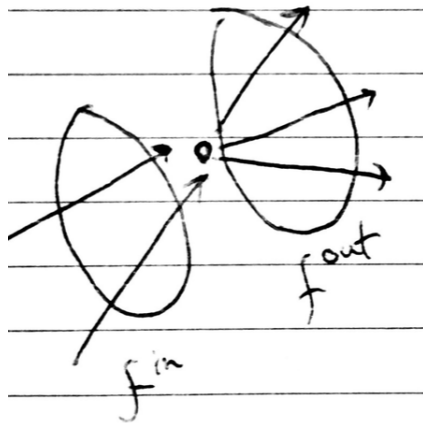


Figure 3: *Visualization*

Definition: Value of flow f :

$$|f| := f^{out}(s)$$

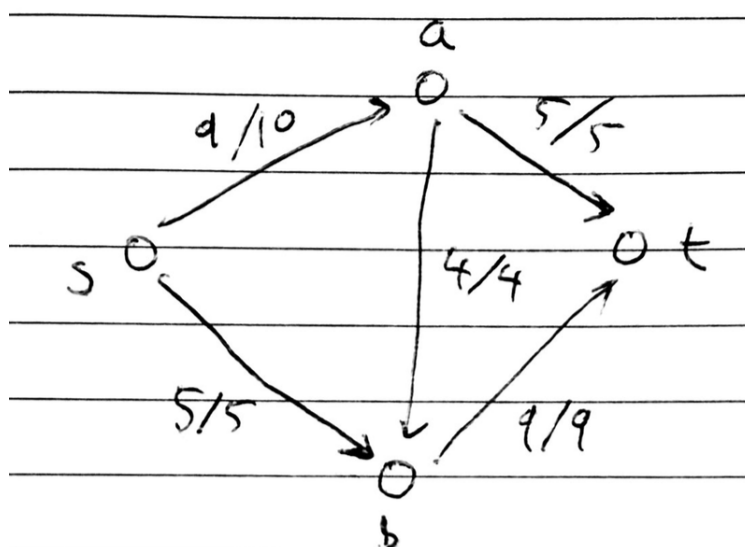


Figure 4: *Example:* $|f| = 9 + 5 = 14$

Notation (borrowed from CLRS): $9/10$, where 9 is the flow and 10 is the capacity

Goal: Design an algorithm to find maximum flow. (*Note: No known Greedy/DP algorithms*)

New algorithm paradigm: *local search*

- Start with some feasible solution
- Perform local changes to get a better solution

In the setting of network flows, *Ford-Fulkerson* is ~~algorithm~~ method, since it skips some steps.

- Start with the all 0 flow ($\forall (u, v) \in E, f(u, v) = 0$)
- Find augmenting s to t path in the residual graph, push more flow along the path