SSSP and Bellman-Ford

Comp 160: algorithms

Tufts University

Single-Source Shortest Paths

Input: directed graph $G(E, V)$, $s \in V$, weight $w : E \to \mathbb{R}$

Output: Shortest path from s to everywhere

Notation

- $\to$ = edge
- $\rightsquigarrow$ = path
- $w(u, v)$ = weight of edge (u, v)
- $w(\overset{p}{\rightsquigarrow})$ = weight of path p = sum of weights of edges of p
- $v.d$ = "score" = Currently known weight of minimum-weight path from s to v.
- $\delta(s, v)$ = "min score" = shortest weight over all possible paths from s to v.

SSSP variations

Why are we asking for SSSP and not:

SSSP Single Source Shortest Paths

Main topic of interest

SDSP Single Destination Shortest Paths

How to reach sink from anywhere

Equivalent question (reverse direction of each edge and run SSSP)

SPSP Single Pair Shortest Paths

Simpler question (fix source s and destination d)

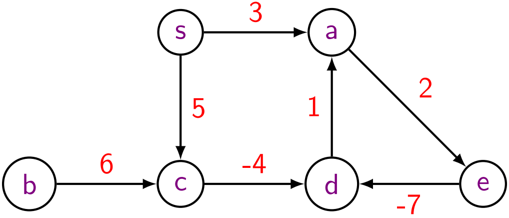
Best solution = run SSSP and drop extra info

APSP All Pairs Shortest Paths

Paths from anywhere to anywhere

Beyond scope of 160 (if interested see CLRS chapter 25)

What is a solution?



Shortest path can have infinite loops

Some nodes can be unreachable (i.e., *b*)

Shortest path trees

Lemma

If G has no negative cycles, the union of all shortest paths $s \rightsquigarrow t$ is a forest

Lemma

If G has no negative cycles and every vertex is reachable *from s, the union of all shortest paths $s \rightsquigarrow t$ is a* tree

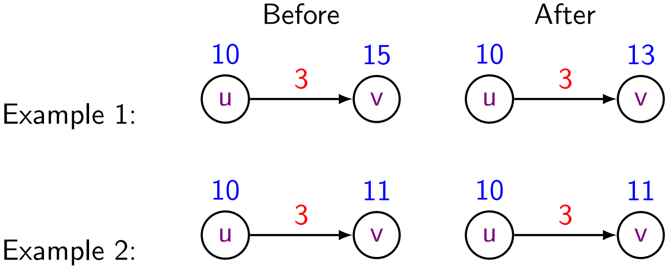
We represent a solution by:

A parent $v.\pi$ for every vertex v (other than root)

A score $v.d$ representing distance to s (∞ if unreachable)

Relaxing - Example

Relax(u, v) = If possible, improve the score at v using (u, v).



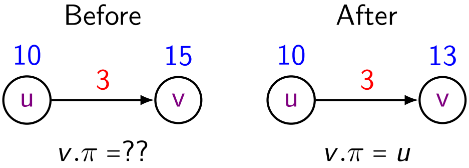
Fun fact: *"Relax" is a historical name.*

Relaxing Code

```
$\textsc{Relax}(u, v, w)$
    if $(v.d > u.d + w(u, v))$
        $v.d \leftarrow u.d + w(u, v)$
        $v.\pi = u$
```

Since we found a new path to v we must udpate its parent as well.

Example:



Useful Lemma 1: Optimal Substructure

Lemma 24.1 (CLRS)

If the shortest path $s \rightsquigarrow t$ is through v, this also gives shortest paths $s \rightsquigarrow v$ and $v \rightsquigarrow t$.

Proof: By contradiction.

Useful Lemma 2: Convergence property

Lemma 24.14 (CLRS)

If $s \rightsquigarrow u \rightarrow v$ is a shortest path and u has min score, then after $\textsc{Relax}(u, v)$, we conclude v will have min score.

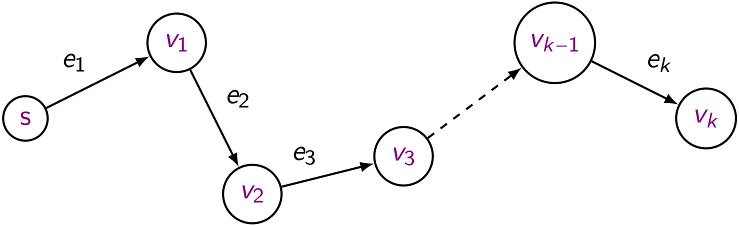
Proof:

- If $s \rightsquigarrow u \rightarrow v$ is a shortest path → min score for v is $w(s \rightsquigarrow u) + w(u, v)$
- $s \rightsquigarrow u$ is a shortest path for u by Lemma 24.1
- Since u has min score then $u.d = w(s \rightsquigarrow u)$
- So $u.d + w(u, v)$ is the min score for v
- $\textsc{Relax}(u, v)$ will assign this to $v.d$

Useful Lemma 3: Path-relaxation property

Lemma 24.15 (CLRS)

If $e_1 e_2 e_3 \ldots e_k$ is a shortest path from s to v, relaxing $e_1, e_2, e_3, \ldots e_k$ in that order (with any number of other edges relaxed in between) will correctly compute v's min score.



Proof? By induction.

Bellman-Ford Algorithm for SSSP

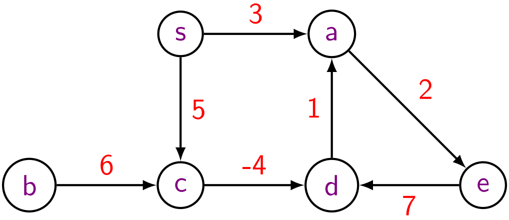
Idea:

1. Initialize $s.d = 0$, all other $v.d = \infty$. All parents NIL.
2. Relax all edges in some order
3. Repeat $|V| - 1$ times step 2

Question: Why does it work? Why $|V| - 1$??

Bellman-Ford: Example

EXAMPLE 1:



Assume edges relaxed in alphabetic order:

$a \leadsto e$

$b \leadsto c$

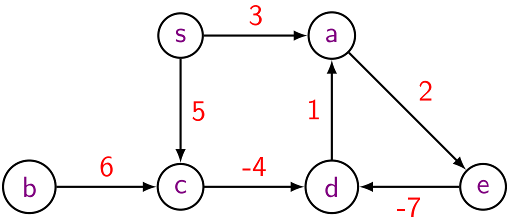
$c \leadsto d$

$d \leadsto a$

$e \leadsto d$

$s \leadsto a \leadsto c$

Negative cycles?



If there are neg cycles then relaxing the $|V|$th time will cause a change in score for some vertex. (Why?)

Bellman-Ford: Code & runtime

```
Bellman-Ford($G, w, s$)
Initialize $v.d = \infty$, $v.\pi$ =NIL for each vertex; $s.d = 0$
For $i = 1$ to $|V| - 1$
    For each edge $(u, v) \in E$
        Relax($u, v$)
(Computation finished. We check for negative cycles)
For each edge $(u, v) \in E$
    If $v.d > u.d + w(u, v)$
        return Error: negative cycle detected
return No negative cycles exist
```

Runtime?

Discussion

Simple and robust algorithm

- Detects negative cycles (can also report them)

Correctness heavily follows from 24.15

A bit slow in practice

- $\Theta(nm)$ is cubic in worst case
- Faster in particular cases (DAG)
- Useful in distributed settings

Can we find a faster algorithm?

Additional practice questions

- Give an example where BF does not produce the correct min scores until the $|V|-1^{\text{st}}$ time relaxing the edges.
- Is it possible for BF to produce the correct min scores after the first time relaxing the edges? If so give an example. If not explain why not.
- How would you handle a graph that is undirected?
- How would you handle a graph that is unweighted?
- Does the order of relaxing impact the output of BF?
- Can the score of s ever be lowered? If so give an example. If not explain why not.