Lecture 21: Shortest Paths with Negative Cycles
CSCI 700 - Algorithms I

Andrew Rosenberg

Last Time

- Kruskal's Algorithm to generate MSTs
- Path counting with matrix multiplication

Today

- Negative Cycles
- Graph Recap

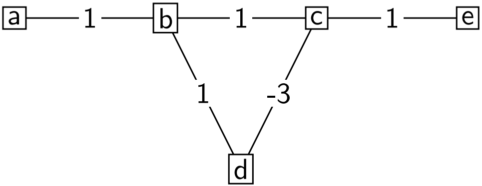
Negative Cycles

A **negative cycle** is a cycle in a weighted graph whose total weight is negative.

Why are negative cycles problematic for most shortest path algorithm (like Dijkstra's)?

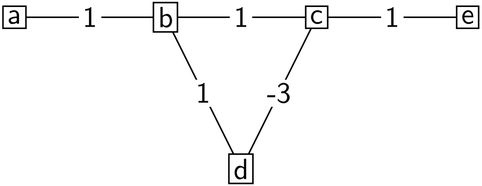
Shortest path with Negative Weight edges

What is the shortest path between a and e?



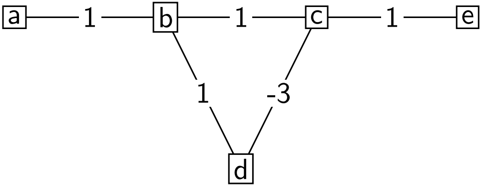
Shortest Path

Path: a,b,c,e = 3



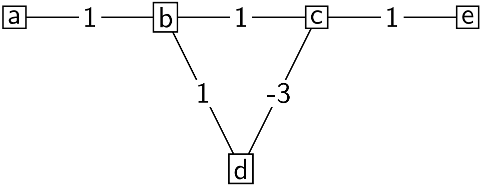
Shortest Path

Path: a,b,c,d,b,c,e $= 2$



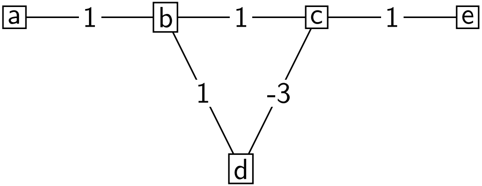
Shortest Path

Path: a,b,c,d,b,c,d,b,c,e = 1



Shortest Path

Path: a,b,c,d,b,c,d,b,c,d,b,c,e $= 0$



Detecting Negative Cycles

Bellman-Ford(G,s)

for $v \in V(G)$ **do**
  $d[v] = \infty$; $parent[v] = \emptyset$
end for
for i $= 1$ **to** $|V(G)| - 1$ **do**
  for $(u, v) \in E(G)$ **do**
    Relax(u,v)
  end for
end for
for $(u, v) \in E(G)$ **do**
  if $d[v] > d[u] + w(u, v)$ **then**
    return FALSE
  end if
  return TRUE
end for

Relax(u,v)

if $d[v] > d[u] + w(u, v)$ **then**
  $d[v] = d[u] + w(u, v)$
  $parent[v] = u$
end if

Path-relaxation Property

Path-relaxation Property: If $p = [v_0, v_1, \ldots, v_k]$ is the shortest path from $s = v_0$ to v_k and the edges of p are relaxed in the order $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$, then $d[v_k] = distance(s, v_k)$. This property holds regardless of any other relaxation steps.

Proof of Bellman-Ford

Claim: At the end of the first for loop of Bellman-Ford, if G contains no negative cycles, d[v] = distance(s,v).

Proof: Let v be a vertex reachable from s. Let $p = [v_0 = s, v_1, \ldots, v_k = v]$ be an acyclic shortest path between s and v. Path p has at most $|V| - 1$ edges. Each of the $|V| - 1$

relaxes **all** edges $E(G)$. Thus, each edge (v_{i-1}, v_i) is relaxed in the ith iteration. By the path-relaxation property,

$d[v] = d[v_k] = distance(s, v_k) = distance(s, v)$

Proof of Bellman-Ford

Claim: If G contains no negative cycles, Bellman-Ford returns $\textsc{True}$ and d[v] $=$ distance(s,v). If G contains a negative cycle reachable from s, then algorithm returns $\textsc{False}$.
Proof: By the previous proof, at the end of the first for loop d[v] $=$ distance(s,v).

At termination, we have for all edges $(u, v) \in E$

$$\begin{aligned} d[v] &= distance(s, v) \\ &\leq distance(s, u) + w(u, v) \\ &= d[u] + w(u, v) \end{aligned}$$

So none of the tests return **False**.

Proof of Bellman-Ford

Suppose that G contains a negative cycle, $c = [v_0, v_1, \ldots, v_k]$. Thus, $0 > \sum_i^k w(v_{i-1}, v_i)$.

Assume not. Assume that Bellman-Ford returns **True**. Thus, $d[v_i] \leq d[v_{i-1}] + w(v_{i-1}, v_i)$.
If we sum around the cycle, we get

$$\begin{aligned}\sum_i^k d[v_i] &\leq \sum_i^k (d[v_{i-1}] + w(v_{i-1}, v_i)) \\ &\leq \sum_i^k d[v_{i-1}] + \sum_i^k w(v_{i-1}, v_i)\end{aligned}$$

However, $\sum_i^k d[v_i] = \sum_i^k d[v_{i-1}]$. Thus

$$0 \leq \sum_i^k w(v_{i-1}, v_i)$$

Contradiction. Thus, Bellman-Ford returns $\textsc{False}$ if G contains a negative cycle.

Graph Recap

What can we do with Graphs?

- Search/Traversal (BFS, DFS)
- Shortest Paths (Dijkstra's, Bellman-Ford)
- Minimum Spanning Trees (Kruskal's, Prim's)
- Cycle Detection (DFS)
- Sorting Vertices by discovery and finishing time
- Detection of Connected Components

Bye

- Next time (12/3)
  - Hashing