The Bellman-Ford Algorithm

Single-Source Shortest Paths, Revisted

Algorithms: Design and Analysis, Part II

The Single-Source Shortest Path Problem

Input: directed graph $G=(V,E)$, edge lengths c_e for each $e \in E$, source vertex $s \in V$.

[can assume no parallel edges]

Goal: For every destination $v \in V$, compute the length of a shortest s-v path.

sum of edge costs

On Dijkstra's Algorithm

Good news: $O(m \log n)$ running time using heaps.

m = number of edges

n = number of vertices

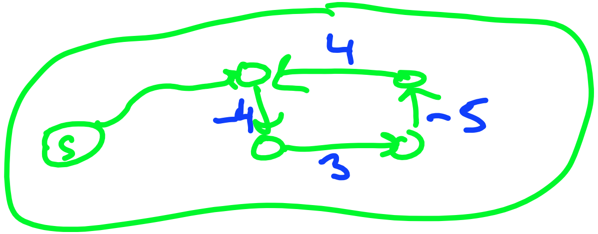
Bad news:

① not always correct with negative edge lengths [e.g., if edges ⟷ financial transactions]

② not very distributed (relevant for Internet routing)

Solution: the Bellman-Ford algorithm

On Negative Cycles



Question: how to define shortest paths when G has a negative cycle?

Solution #1: compute the shortest s-v path, with cycles allowed.

Problem: undefined (or $-\infty$). [will keep traversing negative cycle]

Solution #2: compute shortest cycle-free s-v path.

Problem: NP-hard (no polynomial algorithm, unless P=NP)

Solution #3: (for now) assume input graph has no negative cycles.

Later: will show how to quickly check this condition.

Quiz

Quiz: Suppose the input graph G has no negative cycles. Which of the following is true? [Pick the strongest true statement.] [n = # of vertices, m = # of edges]

(A) for every v, there is a shortest s-v path with $\leq n-1$ edges

(B) for every v, there is a shortest s-v path with $\leq n$ edges.

(C) for every v, there is a shortest s-v path with $\leq m$ edges.

(D) a shortest path can have an arbitrarily large number of edges in it.