15-210

Parallel and Sequential Algorithms and Data Structures

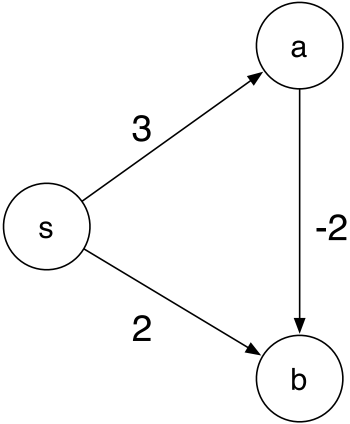
Lecture 14

Shortest Weighted Paths–II

Synopsis

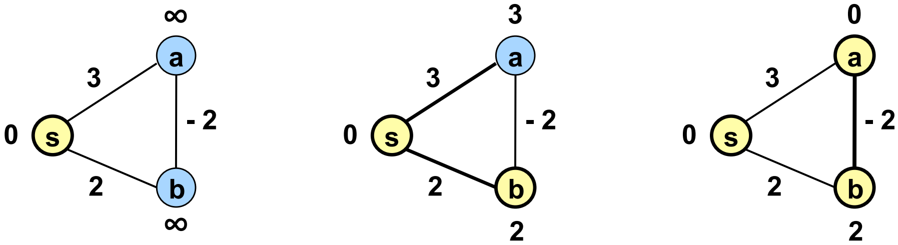
- Graphs with negative edge weights.
- Bellman Ford Algorithm
- Cost Analysis

Graphs with Negative Weights



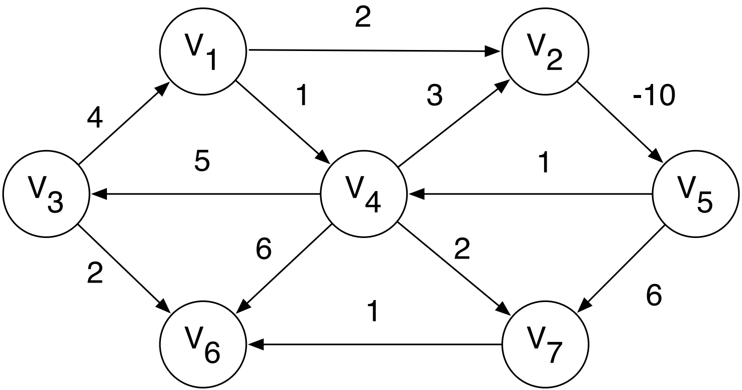
- What is a problem with this graph?

Graphs with Negative Weights



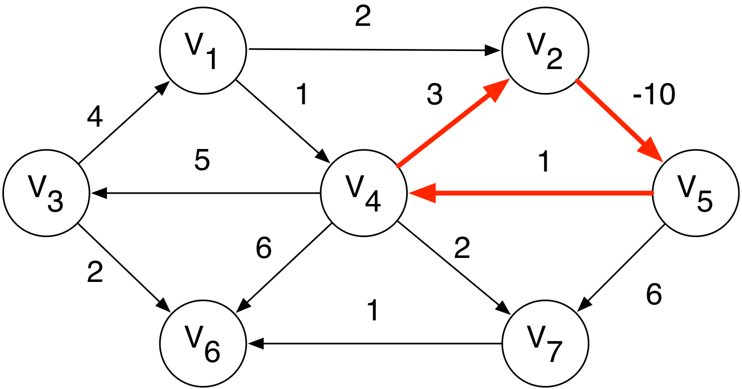
- Dijkstra fails! (Why?)

Graphs with Negative Weights



- What is a problem with this graph?

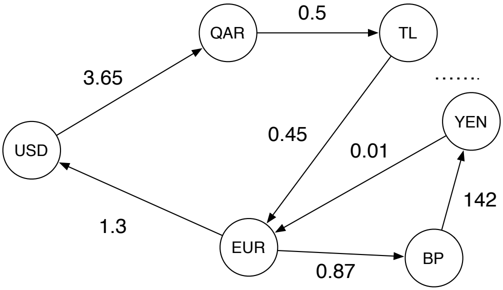
Graphs with Negative Weights



- Negative cost cycle!
- There is no shortest path from v_3 to v_5

Graphs with Negative Weights

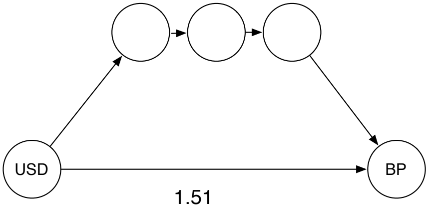
- Currency Exchange Arbitrage



- 100 USD $\rightarrow$ 365 QAR $\rightarrow$ 177.5 TL $\rightarrow$ 80.68 EUR $\rightarrow$ 104.9 USD
  - You just made 5 USD out of thin air!

Graphs with Negative Weights

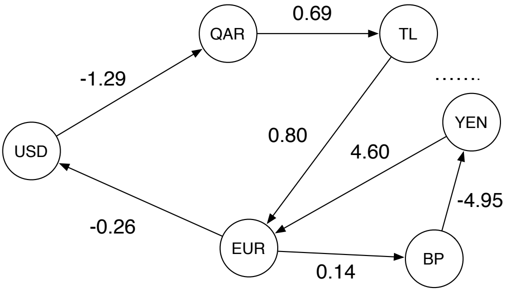
- I have USDs but I want to buy BPs.
  - I can buy directly, or
  - I can buy through some intermediate currencies!



- Which way will get me more BPs?
- I need to do this fast!

Shortest Paths

- How does this problem relate to the shortest problem?
  - Where are the negative weights?



- Weights are – log of the exchange rates!

Shortest Paths with Negative Weights

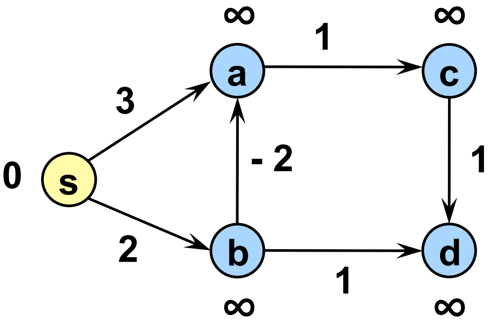
- Define $\delta^{l}_{G}(s,t)$ the shortest weighted path from s to t using at most l edges.
  - so the unweighted path length is l!
- Base cases:
  - $\delta^{0}_{G}(s,s) = 0$
  - $\delta^{0}_{G}(s,v) = \infty$ for all $v \neq s$.
- Induction

$$\delta^{k+1}(v) = \min(\delta^{k}(v), \min_{x \in N^{-}(v)} (\delta^{k}(x) + w(x,v))) .$$

- Minimum of $\delta^{k}(x) + w(x,v)$ over the in-neighbors.

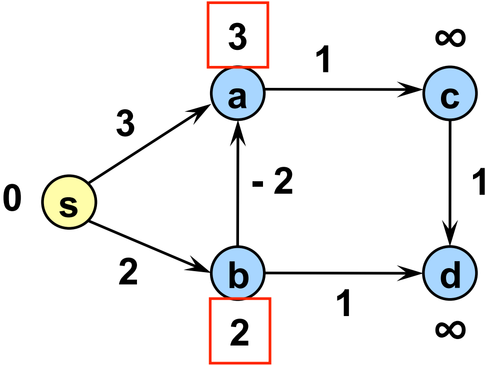
```
fun BellmanFord(G = (V, E), s) =
let
  fun BF(D, k) =
    let
      D' = {v ↦ min(D_v, min_{u ∈ N_G^-(v)}(D_u + w(u, v))) : v ∈ V}
    in
      if (k = |V|) then ⊥
      else if (all{D_v = D'_v : v ∈ V}) then D
      else BF(D', k + 1)
    end

    D = {v ↦ if v = s then 0 else ∞ : v ∈ V}
in BF(D, 0) end
```

How Bellman Ford Algorithm Works



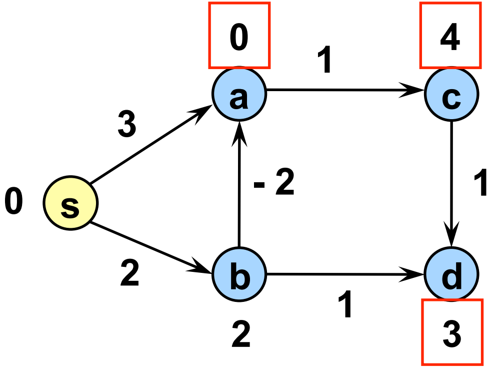
path lengths = 0

How Bellman Ford Algorithm Works



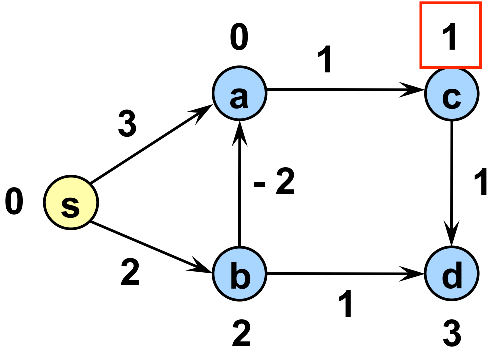
path lengths ≤ 1

How Bellman Ford Algorithm Works



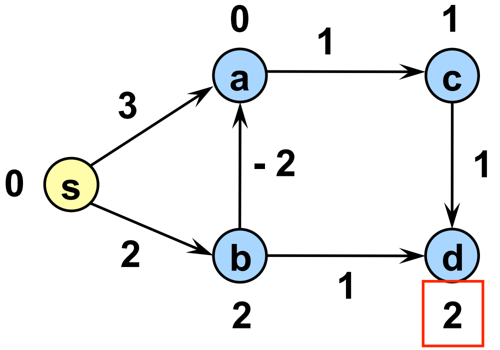
path lengths ≤ 2

How Bellman Ford Algorithm Works



path lengths ≤ 3

How Bellman Ford Algorithm Works



path lengths ≤ 4

Bellman Ford Correctness

Theorem

- Given a directed weighted graph $G = (V, E, w)$, $w : E \to R$, and a source s, the `BellmanFord` algorithm returns the shortest path length from s to every vertex **or** indicates that there is a negative weight cycle in G reachable from s.

Bellman Ford Correctness

- Use induction on the number of edges k in the path.
- Base case is correct, $D_s = 0$.
- On each step, for all $v \in V \setminus \{s\}$, a shortest path with at most $k + 1$ edges
  - must consist of a path of at most k edges for vertex u
  - followed by a single edge (u, v).
- Taking the minimum combination, gives us the shortest path with at most $k + 1$ edges.

Negative Cost Cycles

- This can go at most for $n = |V| - 1$ rounds
- If we reach round n, there must be reachable negative cost cycle.
- Otherwise, Bellman Ford will stop earlier with all simple shortest paths.

Cost Analysis

- Graph represented as a table.
  - $(\mathbb{R}\ \texttt{vtxTable})\ \texttt{vtxTable}$, where first `vtxTable` maps vertices to their in-neighbors

$$G = \{0 \mapsto \{1 \mapsto 0.7,\ 2 \mapsto 1.5\},\ 1 \mapsto \{2 \mapsto -2.0\},\ 2 \mapsto \{\}\}\ .$$

- Graph represented as a sequence of sequences.
  - $((\mathit{int} \times \texttt{eVal})\ \texttt{seq})\ \texttt{seq}$

Bellman - Ford Algorithm (again)

1 **fun** $\texttt{BellmanFord}(G = (V, E), s) =$
2 **let**
3   **fun** $BF(D, k) =$
4   **let**
5       $D' = \{v \mapsto \min(D_v, \min_{u \in N_G^-(v)}(D_u + w(u, v))) : v \in V\}$
6   **in**
7   **if** $(k = |V|)$ **then** $\perp$
8   **else if** (**all**$\{D_v = D'_v : v \in V\}$) **then** D
9   **else** $BF(D', \ k + 1)$
10   **end**
11   $D = \{v \mapsto$ **if** $v = s$ **then** 0 **else** $\infty : v \in V\}$
12 **in** $BF(D, 0)$ **end**

- Line 5 is tabulate over the vertices
- Line 8 is tabulate with a reduction over the vertices

Cost Analysis

val $D' = \{v \mapsto \min(D_v, \min_{u \in N_G^-(v)}(D_u + w(u,v))) : v \in V\}$

- Sum work and max span over vertices.
- $n = |V|$ and $m = |E|$
- For each vertex we have the following costs:
  - Find the neighbors `find G v`: $O(\log n)$ work and span.
  - Map over neighbors – find distance D_u and add: $O(\log n)$ work and span for each u in the in-neigborhood.
  - Min reduce: $O(1 + |N_G(v)|)$ work and $O(\log |N_G(v)|)$ span.

Work per Stage-1

val $D' = \{v \mapsto \min(D_v, \min_{u \in N_G^-(v)}(D_u + w(u,v))) : v \in V\}$

Operation	Over one vertex v	Over graph G
Find	$O(\log n)$	$O(n \log n)$
Map	$O(1 + \lvert N_G^-(v)\rvert \log n)$	$O(n + m \log n)$
Min Reduce	$O(1 + \lvert N_G^-(v)\rvert)$	$O(n + m)$

- Total work is $O((n+m)\log n)$ and assuming $m > n$, $O(m \log n)$

Span per Stage-1

val $D' = \{v \mapsto \min(D_v, \min_{u \in N_G^-(v)}(D_u + w(u,v))) : v \in V\}$

Operation	Over one vertex v	Over graph G
Find	$O(\log n)$	$O(\log n)$
Map	$O(1 + \log n)$	$O(1 + \log n)$
Min Reduce	$O(\log \lvert N_G^-(v) \rvert)$	$O(\log n)$

- Total span is $O(\log n)$

Work / Span per Stage - 2 – Total Cost

else if (all $\{D_v = D'_v : v \in V\})$ **then** D

- This involves a tabulate and an and-reduction.
- Work $= O(n \log n)$, Span $= O(\log n)$

- n sequential calls to BF, so total costs are:

$$\begin{aligned} W(n,m) &= O(n \cdot m \log n) \\ S(n,m) &= O(n \log n) \end{aligned}$$

Costs with ST Sequences

- We use IL (integer labeled) graphs.
- `find` $\rightarrow$ `nth`: $O(1)$ work.
- Similar improvements for looking up neighbors and distance table.

$$\begin{aligned} W(n,m) &= O(nm) \\ S(n,m) &= O(n) \end{aligned}$$

SUMMARY

- Graphs with negative edge weights.
- Bellman Ford Algorithm
- Analysis