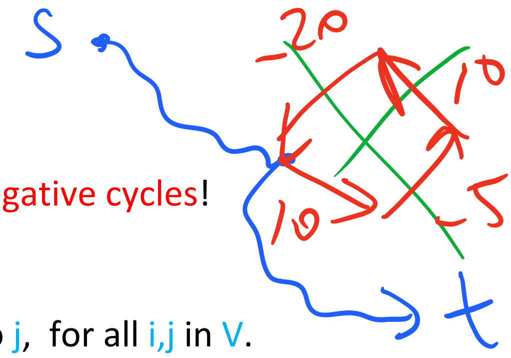
Lecture 11

Monday, June 12, 2017 11:10 PM

# All-Pairs Shortest Paths




**Given:** Digraph $G=(V,E)$, where $V=\{1,2,\dots,n\}$,
possibly negative costs $c(i,j)$, BUT no negative cycles!
( $c(i,j) = \infty$ means no edge $(i,j)$ in $G$ )

**Compute:** $D(i,j)$ = cost of cheapest path from $i$ to $j$, for all $i,j$ in $V$.

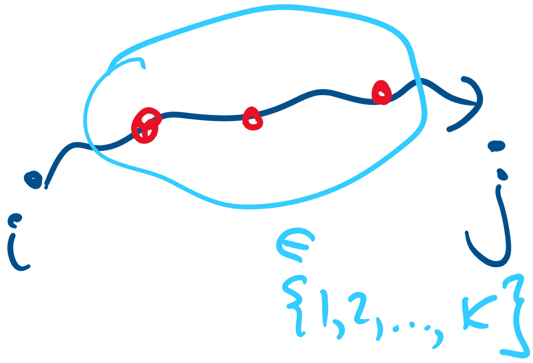
Later, will also want an algorithm that, given $(i,j)$, finds a cheapest path from $i$ to $j$.

**Observation:** Every cheapest path from $i$ to $j$ must be simple, i.e., with no cycles!

## Floyd-Warshall DP algorithm

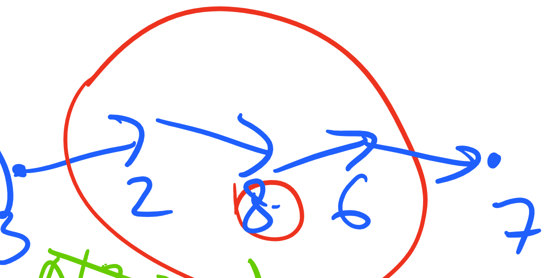
**Step 1:** Array

$$A(k,i,j)$$




**Step 2:** Recurrence

$$\begin{cases} A(0,i,i) = 0, \ \forall i \\ A(0,i,j) = c(i,j) \end{cases}$$



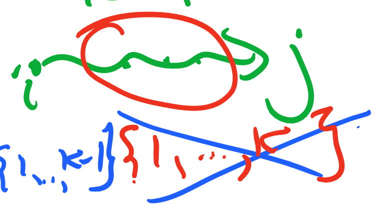

~~A(7,3,7)~~

$$A(8,3,7)$$

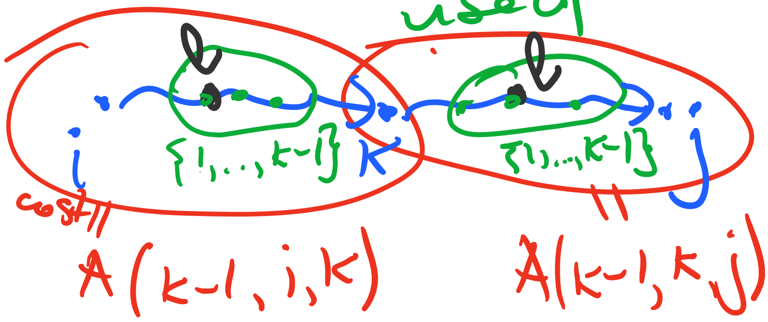
$$A(n,i,j) = D(i,j)$$

$$A(k,i,j) = \min\{A(k-1,i,j),\ A(k-1,i,k) + A(k-1,k,j)\}$$

Case 1: node k is not used



Case 2: node K IS used




Step 3: Algorithm to fill in the array.

array A[K, i, j], $0 \le K \le n$, $1 \le i \le n$, $1 \le j \le n$

Runtime: $O(n^3)$

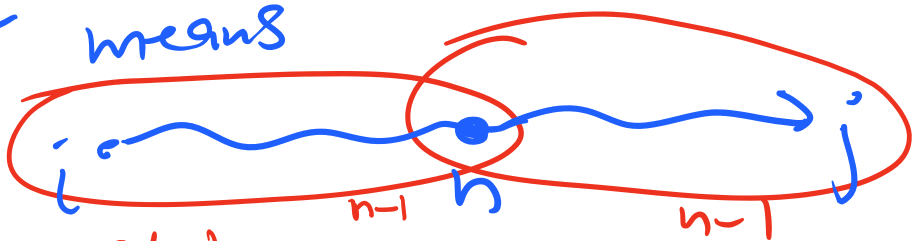
Step 4: Recover shortest paths from the array

Given (i, j), prints out cheapest i ⤳ j

$\delta(i,j) = A[n,i,j]$

|| or ≠ $A[n-1,i,j]$

≠ means




Time: $O(n)$

PrintOpt(K, i, j) % PrintOpt(n, i, j) root call

if K=0 % base case
then if i=j then return
else return edge(i,j)
endif

else return

endif

if $A(k,i,j) = A(k-1,i,j)$ then

Print Opt $(k-1, i, j)$

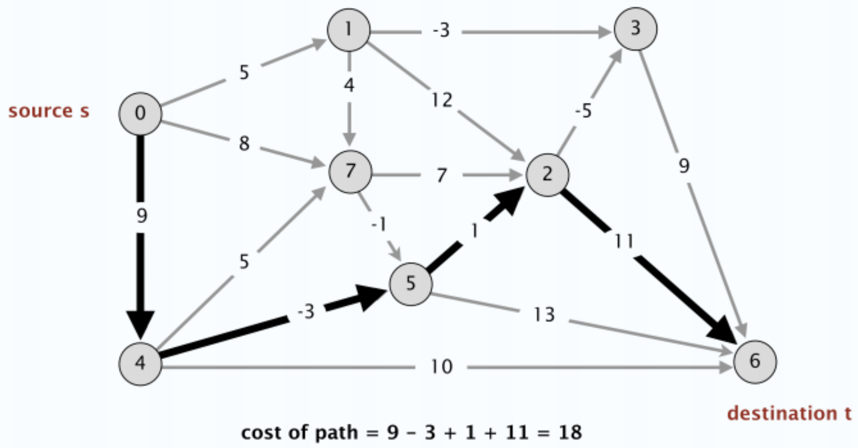
else Print Opt $(k-1, i, k)$ ;

Print Opt $(k-1, k, j)$

endif

## Shortest paths

Shortest path problem. Given a digraph $G = (V, E)$, with arbitrary edge weights or costs $c_{vw}$, find cheapest path from node $s$ to node $t$.

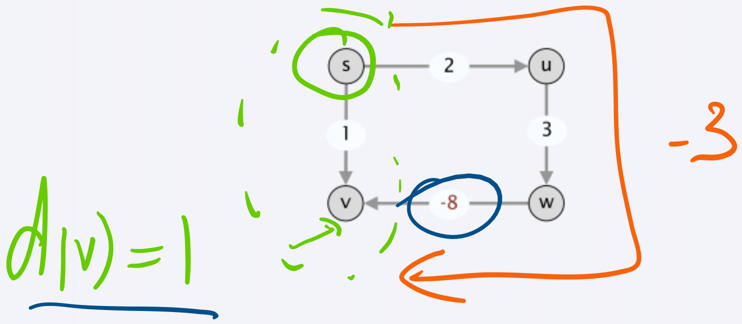



**cost of path = 9 - 3 + 1 + 11 = 18**

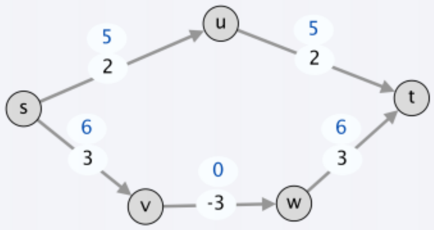
22

## Shortest paths: failed attempts

Dijkstra. Can fail if negative edge weights.



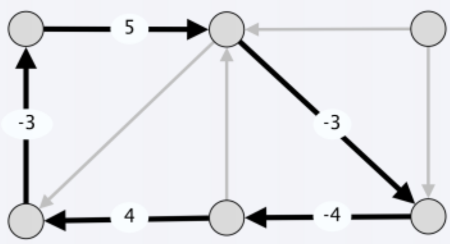

Reweighting. Adding a constant to every edge weight can fail.






## Negative cycles

Def. A negative cycle is a directed cycle such that the sum of its edge weights is negative.

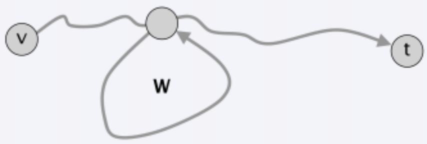



a negative cycle W : $c(W) \;=\; \sum_{e \in W} c_e \;<\; 0$

## Shortest paths and negative cycles

Lemma 1. If some path from $v$ to $t$ contains a negative cycle, then there does not exist a cheapest path from $v$ to $t$.

Pf. If there exists such a cycle $W$, then can build a $v \leadsto t$ path of arbitrarily negative weight by detouring around cycle as many times as desired. ▪



c(W) < 0

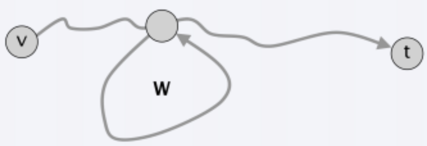


## Shortest paths and negative cycles

Lemma 2. If $G$ has no negative cycles, then there exists a cheapest path from $v$ to $t$ that is simple (and has $\leq n - 1$ edges).

Pf.

- Consider a cheapest $v \leadsto t$ path $P$ that uses the fewest number of edges.
- If $P$ contains a cycle $W$, can remove portion of $P$ corresponding to $W$ without increasing the cost. ▪

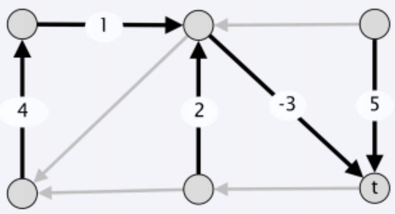


c(W) ≥ 0

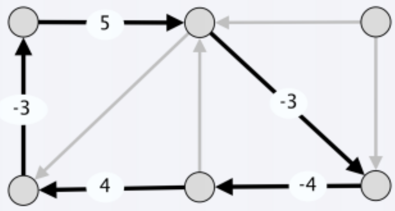
## Shortest path and negative cycle problems

Shortest path problem. Given a digraph $G = (V, E)$ with edge weights $c_{vw}$ and no negative cycles, find cheapest $v \rightsquigarrow t$ path for each node $v$.

Negative cycle problem. Given a digraph $G = (V, E)$ with edge weights $c_{vw}$, find a negative cycle (if one exists).




shortest-paths tree




negative cycle



## Shortest paths: dynamic programming

Bellman-Ford

DP algo

Def. $OPT(i, v)$ = cost of ~~shortest~~ cheapest $v \rightsquigarrow t$ path that uses $\le i$ edges.

$OPT(n-1, v) = dist(v, t)$

- Case 1: Cheapest $v \rightsquigarrow t$ path uses $\le i - 1$ edges.
  - $OPT(i, v) = OPT(i - 1, v)$

optimal substructure property
(proof via exchange argument)

- Case 2: Cheapest $v \rightsquigarrow t$ path uses exactly $i$ edges.
  - if $(v, w)$ is first edge, then $OPT$ uses $(v, w)$, and then selects best $w \rightsquigarrow t$ path using $\le i - 1$ edges

$OPT(0, t) = 0$

v w t

i edges

$$OPT(i, v) = \begin{cases} \infty & \text{if } i = 0 \\ \min\left\{ OPT(i-1, v),\ \min\limits_{(v,w) \in E} \left\{ OPT(i-1, w) + c_{vw} \right\} \right\} & \text{otherwise} \end{cases}$$

& $v \neq t$

v t

i edges

Observation. If no negative cycles, $OPT(n - 1, v)$ = cost of cheapest $v \rightsquigarrow t$ path.

Pf. By Lemma 2, cheapest $v \rightsquigarrow t$ path is simple. ▪

v t

## Shortest paths: implementation

**SHORTEST-PATHS** $(V, E, c, t)$

FOREACH node $v \in V$
　$M[0, v] \leftarrow \infty$.
$M[0, t] \leftarrow 0$.
FOR i = 1 TO $n - 1$
　FOREACH node $v \in V$
　　$M[i, v] \leftarrow M[i-1, v]$.
　　FOREACH edge $(v, w) \in E$
　　　$M[i, v] \leftarrow \min\{ M[i, v],\ M[i-1, w] + c_{vw} \}$.

time $O(n+m)$

$n$ iteration

$O(n.(n+m))$
$\approx O(n.m)$

29

## Shortest paths: implementation

Theorem 1. Given a digraph $G = (V, E)$ with no negative cycles, the dynamic programming algorithm computes the cost of the cheapest $v \rightsquigarrow t$ path for each node $v$ in $\Theta(mn)$ time and $\Theta(n^2)$ space.

Pf.

- Table requires $\Theta(n^2)$ space.
- Each iteration $i$ takes $\Theta(m)$ time since we examine each edge once. ▪

Finding the shortest paths.

- Approach 1: Maintain a *successor(i, v)* that points to next node on cheapest $v \rightsquigarrow t$ path using at most $i$ edges.
- Approach 2: Compute optimal costs $M[i, v]$ and consider only edges with $M[i, v] = M[i-1, w] + c_{vw}$.

$M[i,v] \neq M[i-1,v]$

30

$$\boxed{OPT(i, v) = OPT(i-1, v) \quad \forall v} \quad (*)$$

$$OPT(i, \cdot) = F(OPT(i-1, \cdot))$$

$$OPT(i, \cdot) = F(OPT(i-1, \cdot))$$

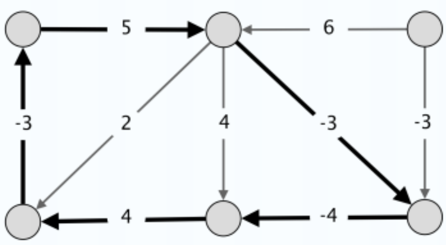
$$OPT(i+1, \cdot) = F(OPT(i-1, \cdot))$$

'' $OPT(i, \cdot)$

$$\boxed{OPT(i,v) = \min \left\{ OPT(i-1,v),\ \min_{v \to w} \{ c_{vw} + OPT(i-1,w) \} \right\}}$$

## Detecting negative cycles

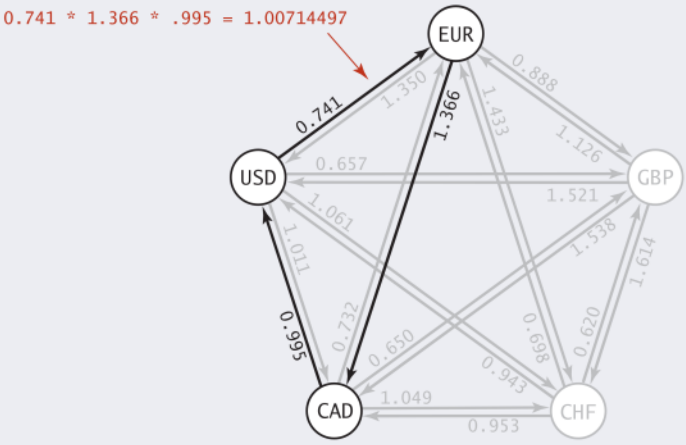
Negative cycle detection problem. Given a digraph $G = (V, E)$, with edge weights $c_{vw}$, find a negative cycle (if one exists).



## Detecting negative cycles: application

Currency conversion. Given $n$ currencies and exchange rates between pairs of currencies, is there an arbitrage opportunity?

Remark. Fastest algorithm very valuable!




47

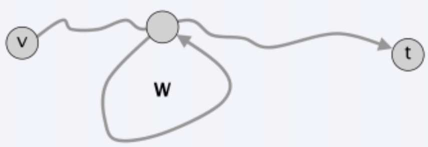
## Detecting negative cycles

Lemma 5. If $OPT(n, v) = \mathrm{OPT}(n-1, v)$ for all $v$, then no negative cycle can reach $t$.
Pf. Bellman-Ford algorithm. ▪

Lemma 6. If $OPT(n, v) < \mathrm{OPT}(n-1, v)$ for some node $v$, then (any) cheapest path from $v$ to $t$ contains a cycle $W$. Moreover $W$ is a negative cycle.

Pf. [by contradiction]

- Since $OPT(n, v) < \mathrm{OPT}(n-1, v)$, we know that shortest $v \rightsquigarrow t$ path $P$ has exactly $n$ edges.
- By pigeonhole principle, $P$ must contain a directed cycle $W$.
- Deleting $W$ yields a $v \rightsquigarrow t$ path with $< n$ edges $\Rightarrow$ $W$ has negative cost. ▪




c(W) < 0

48

Opt(i, v) < opt(n-1, v) for large enough i

## Detecting negative cycles

Theorem 4. Can find a negative cycle in $\Theta(mn)$ time and $\Theta(n^2)$ space.
Pf.

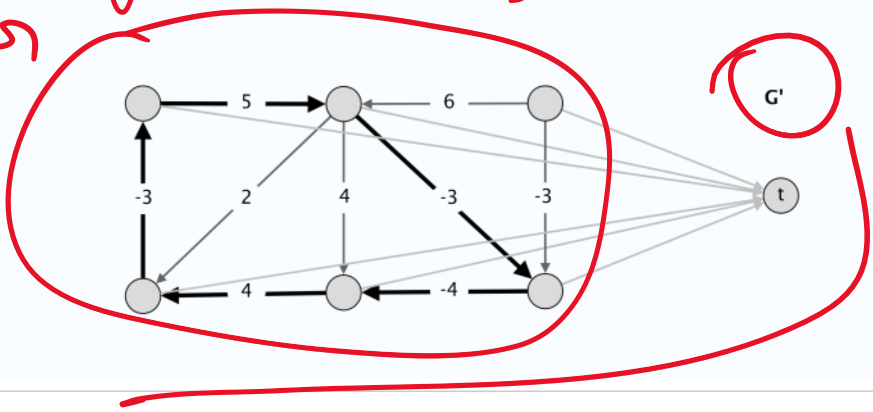
- Add new node $t$ and connect all nodes to $t$ with 0-cost edge.
- $G$ has a negative cycle iff $G'$ has a negative cycle than can reach $t$.
- If $OPT(n, v) = OPT(n-1, v)$ for all nodes $v$, then no negative cycles.

$n'$ $n'-1$

- Add new node $t$ and connect all nodes to $t$ with 0-cost edge.
- $G$ has a negative cycle iff $G'$ has a negative cycle than can reach $t$.
- If $OPT(n, v) = OPT(n - 1, v)$ for all nodes $v$, then no negative cycles.
- If not, then extract directed cycle from path from $v$ to $t$.
  (cycle cannot contain $t$ since no edges leave $t$) ▪

$n'$ $n' - 1$

G has n nodes




has $n+1$ nodes
$= n'$