CS4102 Algorithms
Spring 2019

<u>Warm up:</u>

Modify Dijkstra's Algorithm to find the shortest paths by *product* of edge weights (assume all weights are at least 1)

Dijkstra's Algorithm

Initialize $d_v = \infty$ for each node v

Keep a priority queue PQ of nodes, using d_v as key

Pick a start node s, set $d_s = 0$

While PQ is not empty:

    $v = PQ.extractmin()$

    for each $u \in V$ s.t. $(v, u) \in E$:

        $PQ.decreaseKey(u, \min(d_u, d_v + w(v, u)))$

Modify Dijkstra's Algorithm to find the shortest paths by *product* of edge weights (assume all weights are at least 1)

Dijkstra's Algorithm (for min product)

Initialize $d_v = \infty$ for each node v

Keep a priority queue PQ of nodes, using d_v as key

Pick a start node s, set $d_s = 1$

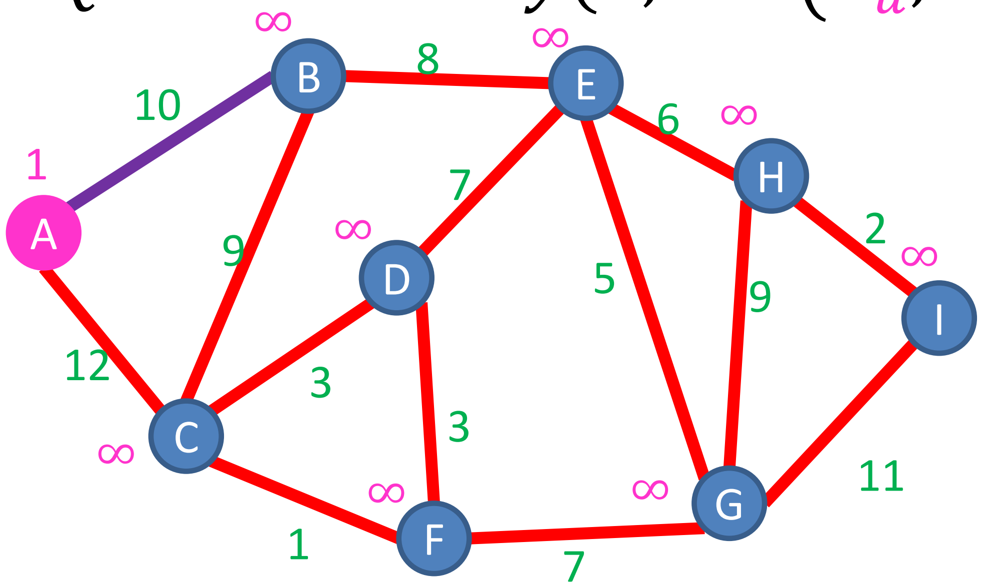
While PQ is not empty:

$\quad v = PQ.extractmin()$

$\quad$for each $u \in V$ s.t. $(v, u) \in E$:

$\quad\quad PQ.decreaseKey(u, \min(d_u, d_v \cdot w(v, u)))$

How do we know this works?



Shortest path by product

Goal: find the path $(s = v_1, v_2, \dots, v_{k-1}, v_k)$ which minimizes:

$$w(v_1, v_2) \cdot w(v_2, v_3) \cdot \ldots \cdot w(v_{k-1}, v_k)$$

Observation: $\log(x \cdot y) = \log x + \log y$

$$\log(\, w(v_1, v_2) \cdot w(v_2, v_3) \cdot \ldots \cdot w(v_{k-1}, v_k)\,)$$
$$= \log w(v_1, v_2) + \log w(v_2, v_3) + \cdots + \log w(v_{k-1}, v_k)$$

New Goal: find the path $(s = v_1, v_2, \dots, v_{k-1}, v_k)$ which minimizes:

$$\log\big(w(v_1, v_2)\big) + \log\big(w(v_2, v_3)\big) + \cdots + \log(w(v_{k-1}, v_k))$$

Dijkstra's Algorithm (for min product)

Initialize $d_v = \infty$ for each node v

Keep a priority queue PQ of nodes, using d_v as key

Pick a start node s, set $d_s = 0$

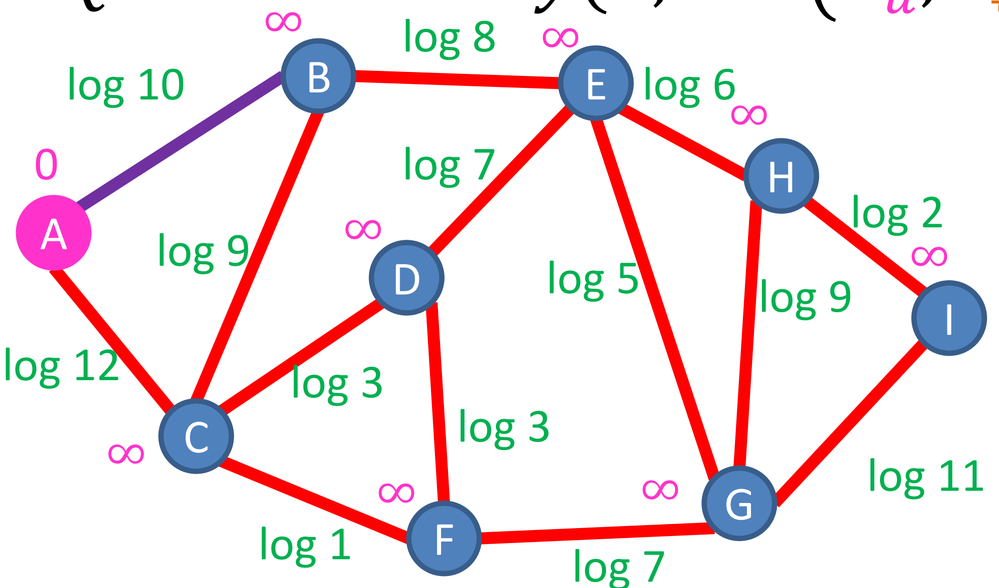
While PQ is not empty:

$\quad v = PQ.extractmin()$

$\quad$ for each $u \in V$ s.t. $(v, u) \in E$:

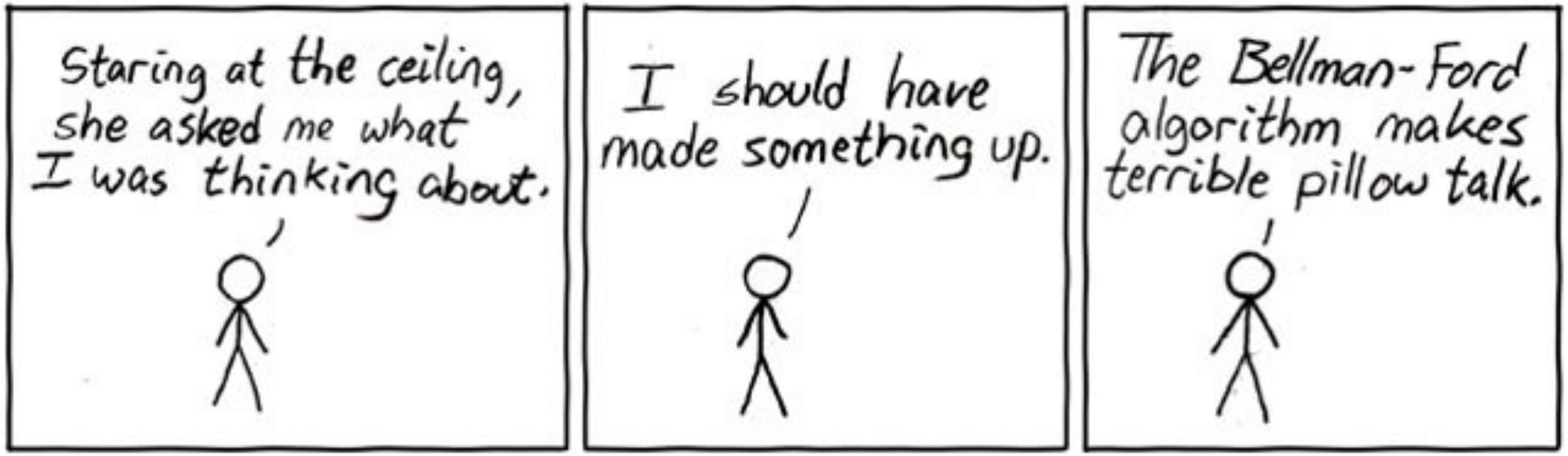
$\quad\quad PQ.decreaseKey(u, \min(d_u, \cancel{d_v \cdot w(v,u)}))$

$d_v + \log(w(v,u))$



Today's Keywords

- Graphs
- Shortest path
- Bellman-Ford
  - OG DP
- Floyd-Warshall



CLRS Readings

- Chapter 22
- Chapter 23
- Chapter 24

Homeworks

- HW7 Due **Tomorrow** April 16 @11pm
  - Written (use latex)
  - Graphs
- HW8 Released Tomorrow April 16 @11:30pm
  - Due Tuesday April 23 @11pm
  - Programming (Python or Java)
  - Graphs/Flow (hint, see Wednesday’s class)

Currency Exchange

1 Dollar = 0.8783121137 Euro

1 Dollar = 3.87 Ringgit

Currency code ▲▼	Currency name ▲▼	Units per USD	USD per Unit
USD	US Dollar	1.0000000000	1.0000000000
EUR	Euro	0.8783121137	1.1385474303
GBP	British Pound	0.6956087704	1.4375896950
INR	Indian Rupee	66.1909310706	0.0151078098
AUD	Australian Dollar	1.3050318080	0.7662648480
CAD	Canadian Dollar	1.2997506294	0.7693783541
SGD	Singapore Dollar	1.3478961522	0.7418969172
CHF	Swiss Franc	0.9590451582	1.0427037678
MYR	Malaysian Ringgit	3.8700000000	0.2583979328
JPY	Japanese Yen	112.5375383115	0.0088859239
CNY	Chinese Yuan Renminbi	6.4492409303	0.1550570076
NZD	New Zealand Dollar	1.4480018872	0.6906068347
THB	Thai Baht	35.1005319022	0.0284895968
HUF	Hungarian Forint	275.7012427385	0.0036271146
AED	Emirati Dirham	3.6730000000	0.2722570106
HKD	Hong Kong Dollar	7.7563973683	0.1289258341
MXN	Mexican Peso	17.3168505322	0.0577472213
ZAR	South African Rand	14.7201431400	0.0679341220

Currency Exchange

1 Dollar = 0.8783121137 Euro

Currency code ▲▼	Currency name ▲▼	Units per EUR	EUR per Unit
USD	US Dollar	1.1386632306	0.8782227907
EUR	Euro	1.0000000000	1.0000000000
GBP	British Pound	0.7921136388	1.2624451227
INR	Indian Rupee	75.3658843112	0.0132686030
AUD	Australian Dollar	1.4859561878	0.6729673514
CAD	Canadian Dollar	1.4796754127	0.6758238945
SGD	Singapore Dollar	1.5347639238	0.6515660060
CHF	Swiss Franc	1.0917416715	0.9159676012
MYR	Malaysian Ringgit	4.4140052400	0.2265516114
JPY	Japanese Yen	128.1388820287	0.0078040325
CNY	Chinese Yuan Renminbi	7.3411003512	0.1362193612
NZD	New Zealand Dollar	1.6484648003	0.6066250246
THB	Thai Baht	39.9627318192	0.0250233143
HUF	Hungarian Forint	313.9042436792	0.0031856849
AED	Emirati Dirham	4.1823100458	0.2391023117

1 Euro= 4.1823100458 Dirham

Currency code ▲▼	Currency name ▲▼	Units per AED	AED per Unit
USD	US Dollar	0.2722570106	3.6730000000
EUR	Euro	0.2391289974	4.1818433177
GBP	British Pound	0.1893997890	5.2798369266
INR	Indian Rupee	18.0207422309	0.0554916100
AUD	Australian Dollar	0.3552996418	2.8145257760
CAD	Canadian Dollar	0.3538334124	2.8261887234
SGD	Singapore Dollar	0.3669652245	2.7250538559
CHF	Swiss Franc	0.2610686193	3.8304105746
MYR	Malaysian Ringgit	1.0548325619	0.9480177576
JPY	Japanese Yen	30.6399242607	0.0326371564
CNY	Chinese Yuan Renminbi	1.7555154332	0.5696332719
NZD	New Zealand Dollar	0.3941937299	2.5368237088
THB	Thai Baht	9.5553789460	0.1046530970
HUF	Hungarian Forint	75.0637936939	0.0133220019
AED	Emirati Dirham	1.0000000000	1.0000000000

1 Dirham= 1.0548325619 Ringgit

1 Dollar= 0.8783121137 * 4.1823100458 * 1.0548325619 Ringgit
= **3.87479406049 Ringgit**

Directly: 1 Dollar = 3.87 Ringgit

Currency Exchange

1 Dollar = 3.87479406049 Ringgit

Currency code ▲▼	Currency name ▲▼	Units per USD	USD per Unit
USD	US Dollar	1.0000000000	1.0000000000
EUR	Euro	0.8783121137	1.1385474303
GBP	British Pound	0.6956087704	1.4375896950
INR	Indian Rupee	66.1909310706	0.0151078098
AUD	Australian Dollar	1.3050318080	0.7662648480
CAD	Canadian Dollar	1.2997506294	0.7693783541
SGD	Singapore Dollar	1.3478961522	0.7418969172
CHF	Swiss Fra[illegible]	[illegible]451582	1.0427037678
MYR	Malaysian Ringgit	3.8700000000	0.2583979328
JPY	Japanese Yen	112.5375383115	0.0088859239
CNY	Chinese Yuan Renminbi	6.4492409303	0.1550570076
NZD	[illegible]	[illegible]	0.6906068347
THB	[illegible]	[illegible]	0.0284895968
HUF	Hungarian F[illegible]	275.7012427385	0.0036271146
AED	Emirati Dirha[illegible]	3.6730000000	0.2722570106
HKD	Hong Kong Dollar	7.7563973683	0.1289258341
MXN	Mexican Peso	17.3168505322	0.0577472213
ZAR	South African Rand	14.7201431400	0.0679341220

1 Ringgit = 0.2583979328 Dollar

1 Dollar = 3.87479406049 * 0.2583979328 Dollar
= 1.00123877526 Dollar

Free Money!

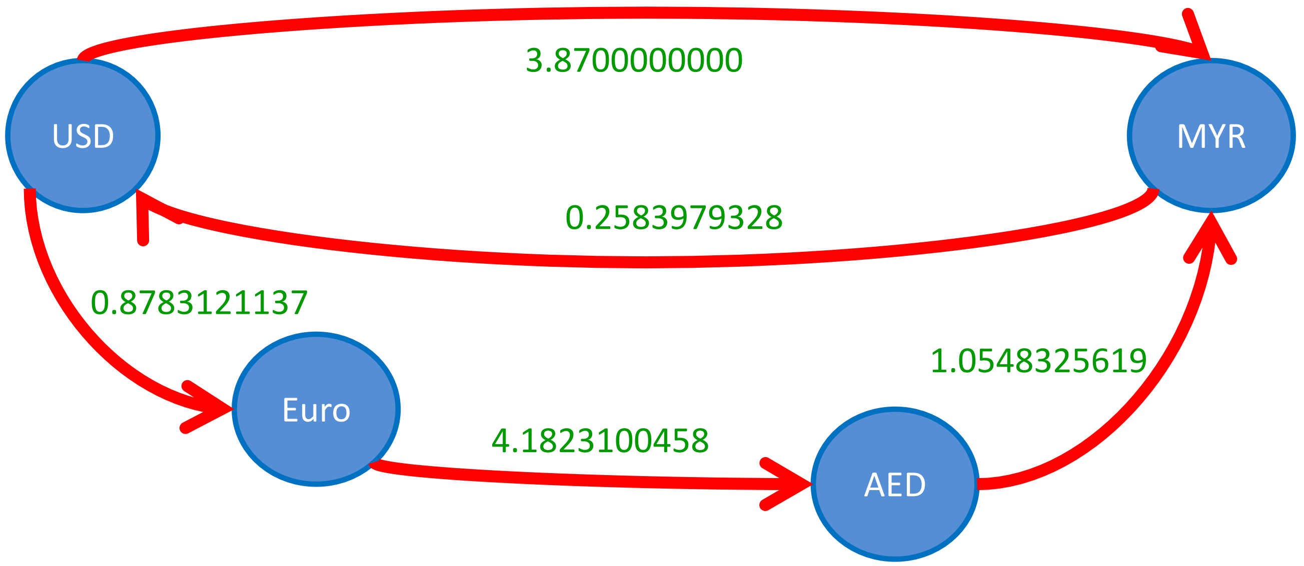
Best Currency Exchange

Best way to transfer USD to MYR:

Given a graph of currencies (edges are exchange rates)
find the shortest path by product of edge weights

$$\max_{\mathrm{p}} \prod_{e \in p} w(e)$$

Invert edge weights to make it a minimization problem



Best Currency Exchange

Best way to transfer USD to MYR:

Given a graph of currencies (edges are exchange rates)
find the shortest path by product of edge weights

$$\min_{p} \prod_{e \in p} \frac{1}{w(e)}$$

Take log of edge weights to make summation

Best Currency Exchange

Best way to transfer USD to MYR:

Given a graph of currencies (edges are exchange rates)

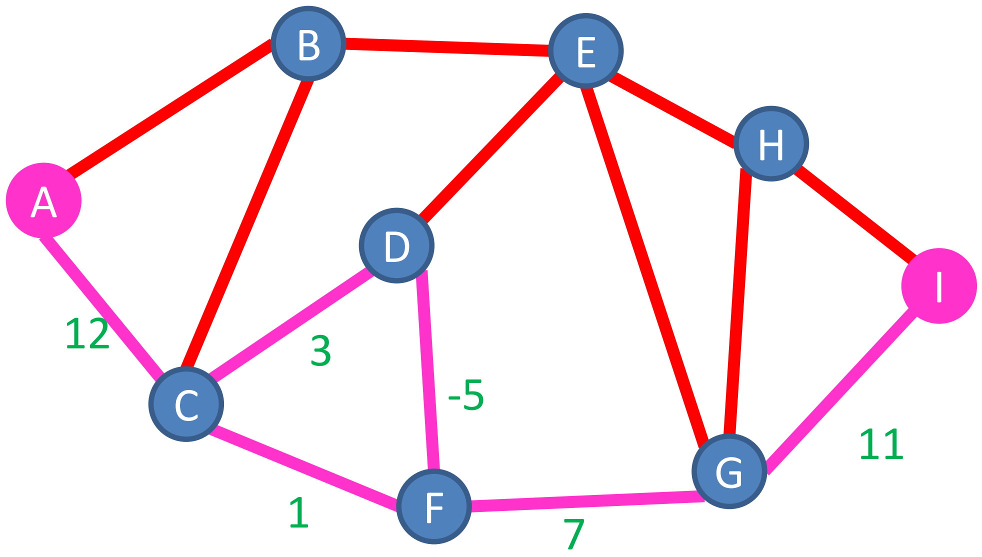
find the shortest path by product of edge weights

$$\min_{\mathrm{p}} \sum_{e \in p} \log \frac{1}{w(e)}$$

Now a shortest path problem!

Negative Edge Weights!

Problem with negative edges



$w(C, F, D, C) = -1$

Weight if we take the cycle 0 times: 31
Weight if we take the cycle 1 time: 30
Weight if we take the cycle 2 times: 29
…

There is no shortest path from A to I!

What we need: an algorithm that finds the shortest path in graphs with negative edge weights (if one exists)

Note

Any simple path has at most $V-1$ edges

Pigeonhole Principle!

More than $V-1$ edges means some node appears twice (i.e., there is a cycle)

If there is a shortest path of more than $V-1$ edges, there is a negative weight cycle

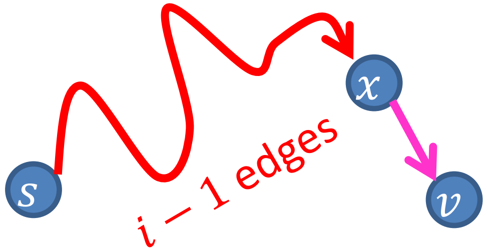
Bellman-Ford

Idea: Use Dynamic Programming!

$Short(i, v) =$ weight of the shortest path from s to v using at most i edges

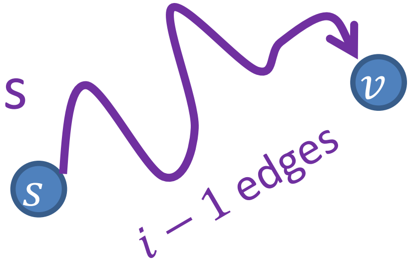
Two options:

A path of $i - 1$ edges from s to some node x, then edge (x, v)



OR

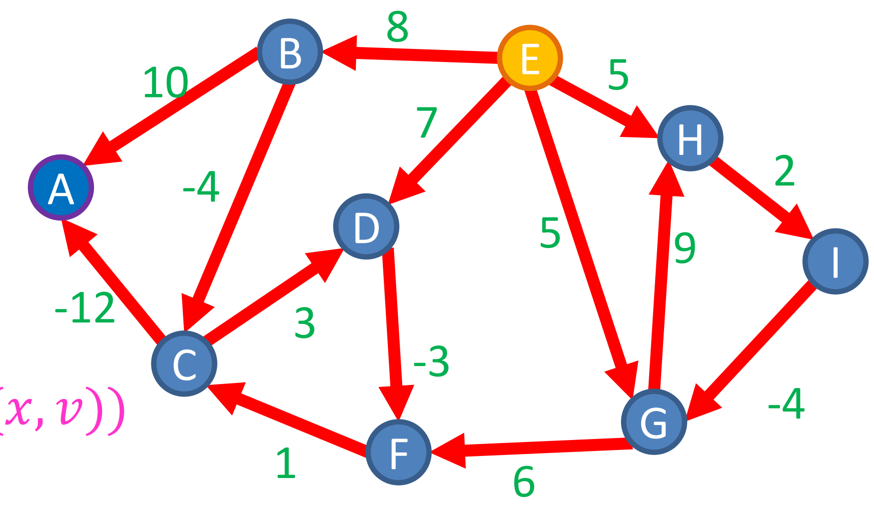
A path from s to v of at most $i - 1$ edges



$$Short(i, v) = \min \begin{cases} \min_x (Short(i-1, x) + w(x, v)) \\ Short(i-1, v) \end{cases}$$

Bellman Ford

Start node is E
Initialize all others to ∞



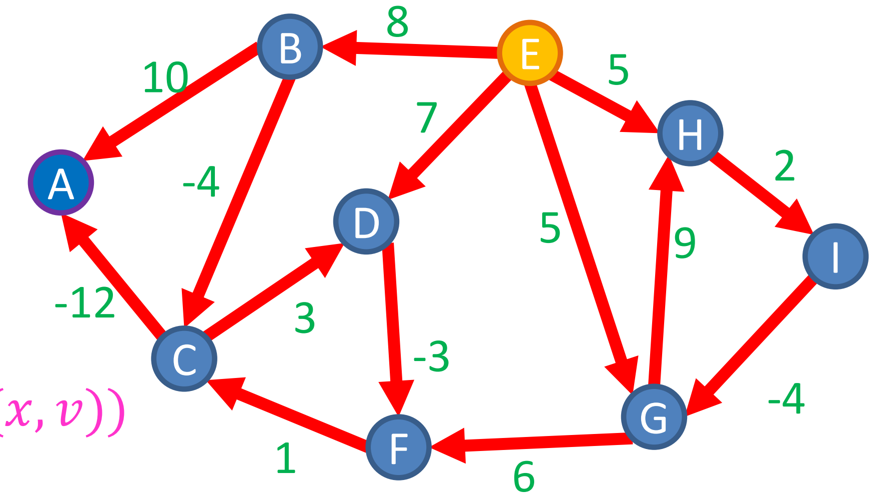
$$Short(i, v) = \text{weight of the shortest path from } s \text{ to } v \text{ using at most } i \text{ edges}$$

$$Short(i, v) = \min \begin{cases} \min_x (Short(i-1, x) + w(x, v)) \\ Short(i-1, v) \end{cases}$$

$i =$ \ $v =$	A	B	C	D	E	F	G	H	I
0	∞	∞	∞	∞	**0**	∞	∞	∞	∞
1									
2									
3									
4									
5									
6									
7									

Bellman Ford

Start node is E
Initialize all others to ∞



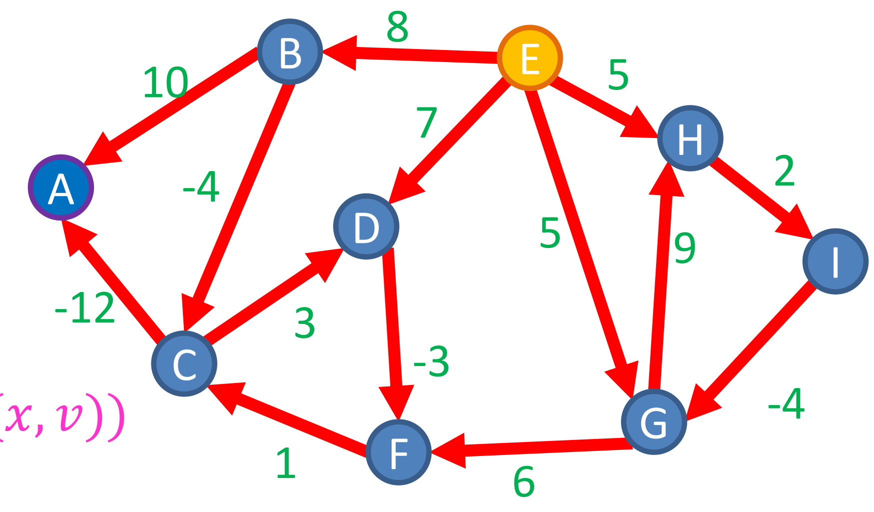
$Short(i, v) =$ weight of the shortest path from s to v using at most i edges

$$Short(i,v) = \min \begin{cases} \min_x (Short(i-1, x) + w(x, v)) \\ Short(i-1, v) \end{cases}$$

$i =$ \ $v =$	A	B	C	D	E	F	G	H	I
0	∞	∞	∞	∞	**0**	∞	∞	∞	∞
1	∞	**8**	∞	**7**	**0**	∞	**5**	**5**	∞
2									
3									
4									
5									
6									
7									

Bellman Ford

Start node is E
Initialize all others to ∞

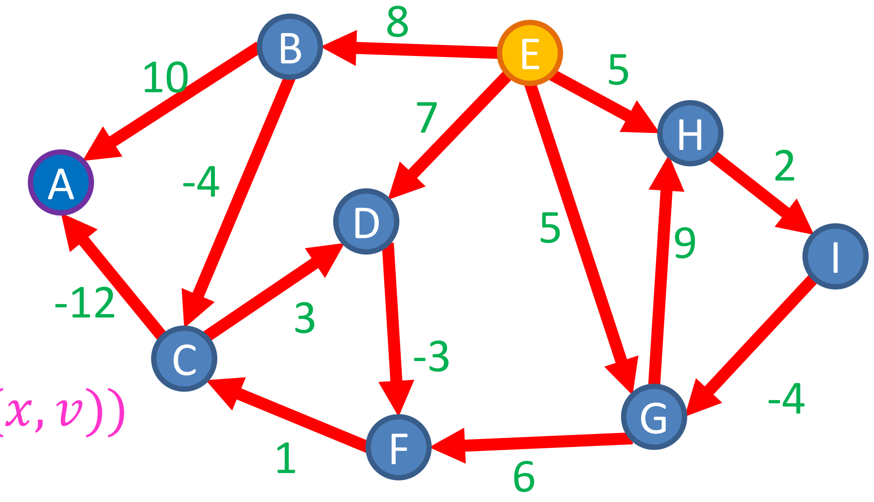


$$Short(i, v) = \text{weight of the shortest path from } s \text{ to } v \text{ using at most } i \text{ edges}$$

$$Short(i, v) = \min \begin{cases} \min_x \left(Short(i-1, x) + w(x, v) \right) \\ Short(i-1, v) \end{cases}$$

$i =$ \ $v =$	A	B	C	D	E	F	G	H	I
0	∞	∞	∞	∞	**0**	∞	∞	∞	∞
1	∞	**8**	∞	**7**	**0**	∞	**5**	**5**	∞
2	**18**	**8**	**4**	**7**	**0**	**4**	**5**	**5**	**7**
3									
4									
5									
6									
7									

Bellman Ford



Start node is E
Initialize all others to ∞

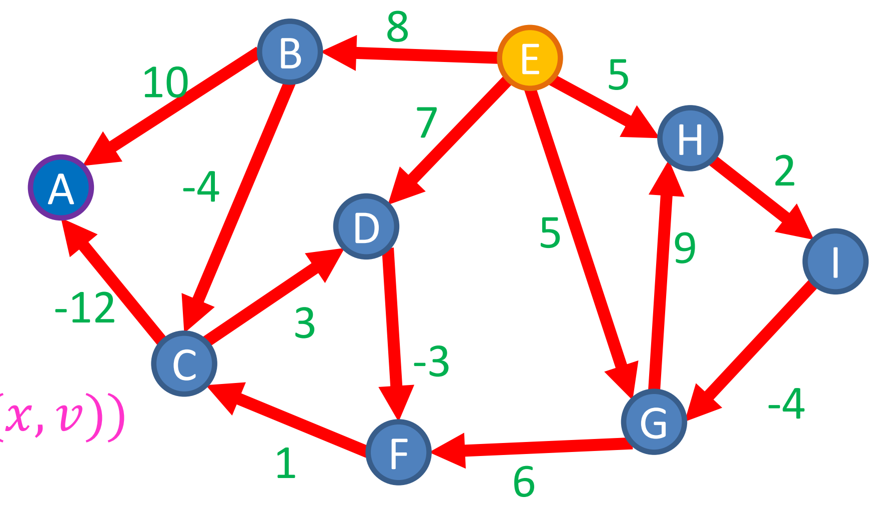
$$Short(i, v) = \text{weight of the shortest path from } s \text{ to } v \text{ using at most } i \text{ edges}$$

$$Short(i, v) = \min \begin{cases} \min_x (Short(i-1, x) + w(x, v)) \\ Short(i-1, v) \end{cases}$$

$i =$ \ $v =$	A	B	C	D	E	F	G	H	I
0	∞	∞	∞	∞	**0**	∞	∞	∞	∞
1	∞	**8**	∞	**7**	**0**	∞	**5**	**5**	∞
2	**18**	**8**	**4**	**7**	**0**	**4**	**5**	**5**	**7**
3	**-8**	**8**	**4**	**7**	**0**	**4**	**3**	**5**	**7**
4									
5									
6									
7									

Bellman Ford

Start node is E
Initialize all others to ∞



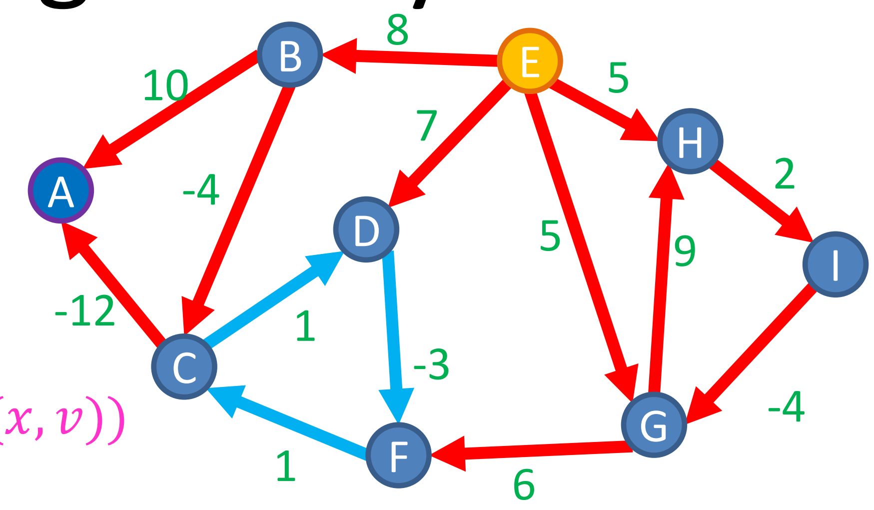
$Short(i, v) =$ weight of the shortest path from s to v using at most i edges

$$Short(i, v) = \min \begin{cases} \min_x (Short(i-1, x) + w(x, v)) \\ Short(i-1, v) \end{cases}$$

$i =$ \ $v =$	A	B	C	D	E	F	G	H	I
0	∞	∞	∞	∞	**0**	∞	∞	∞	∞
1	∞	8	∞	7	0	∞	5	5	∞
2	18	8	4	7	0	4	5	5	7
3	-8	8	4	7	0	4	3	5	7
4	-8	8	4	7	0	4	3	5	7
5	-8	8	4	7	0	4	3	5	7
6	-8	8	4	7	0	4	3	5	7
7	-8	8	4	7	0	4	3	5	7

Bellman Ford: Negative cycles

Start node is E
Initialize all others to ∞



$Short(i,v) =$ weight of the shortest path from s to v using at most i edges

$$Short(i,v) = \min\begin{cases} \min_x (Short(i-1,x) + w(x,v)) \\ Short(i-1,v) \end{cases}$$

$i =$ \ $v =$	A	B	C	D	E	F	G	H	I
0	∞	∞	∞	∞	**0**	∞	∞	∞	∞
1	∞	8	∞	7	0	∞	5		
2			4	7	0	4	5		
3			4	5	0	4	5		
4			4	5	0	2	5		
5			3	5	0	2	5		
6			3	4	0	2	5		
7			3	4	0	1	5		

If we computed row V, values change

There is a negative weight cycle!

Bellman Ford Run Time

Intialize array $Short[V][V]$ V^2

Initialize $Short[0][v] = \infty$ for each vertex V

Initialize $Short[0][s] = 0$ 1

For $i = 1, \dots, V-1$: V times

for each $e = (x, y) \in E$: E times

$Short[i][y] = \min\{$

$Short[i-1][x] + w(x, y)$, 1

$Short[i-1][y]\}$

$\Theta(V^2 + EV)$

$\Theta(EV)$

Why Use Bellman-Ford?

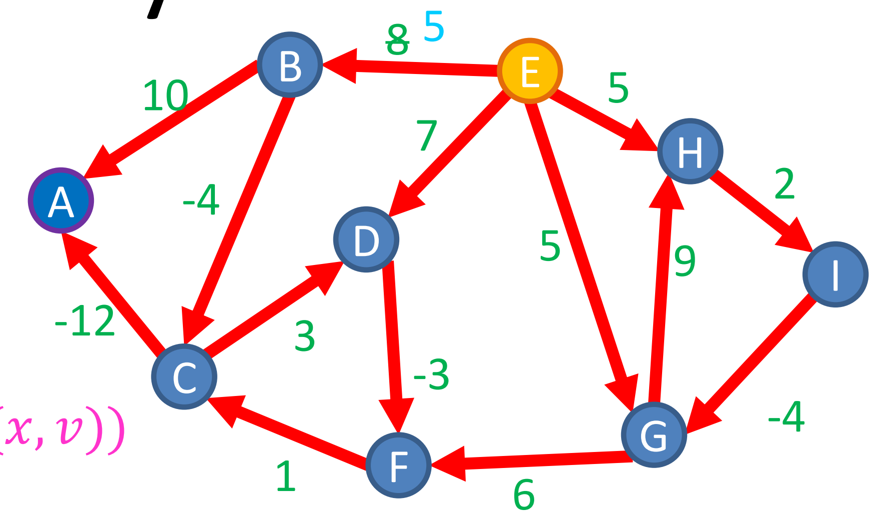
- Dijkstra's:
  - only works for positive edge weights
  - Run Time: $\Theta(E \log V)$
  - Not good for dynamic graphs (where edge weights are variable)
    - Must recalculate "from scratch"
- Bellman-Ford:
  - Works for negative edge weights
  - Run Time: $\Theta(E \cdot V)$
  - More efficient for dynamic graphs
    - $\Theta(E)$ time to recalculate

Bellman Ford: Dynamic

Each node will update its neighbors if edge weight changes

$Short(i, v) =$ weight of the shortest path from s to v using at most i edges

$$Short(i,v) = \min\begin{cases} \min\limits_{x}\left(Short(i-1,x) + w(x,v)\right) \\ Short(i-1,v) \end{cases}$$



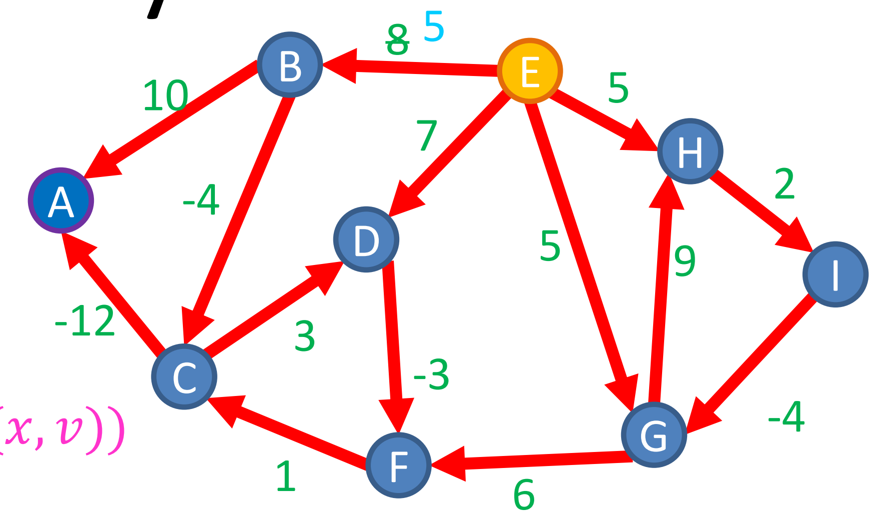
$i =$ \ $v =$	A	B	C	D	E	F	G	H	I
0	∞	∞	∞	∞	0	∞	∞	∞	∞
1	∞	5	∞	7	0	∞	5	5	∞
2	18	8	4	7	0	4	5	5	7
3	-8	8	4	7	0	4	3	5	7
4	-8	8	4	7	0	4	3	5	7
5	-8	8	4	7	0	4	3	5	7
6	-8	8	4	7	0	4	3	5	7
7	-8	8	4	7	0	4	3	5	7

Bellman Ford: Dynamic

Each node will update its neighbors if edge weight changes

$Short(i, v) =$ weight of the shortest path from s to v using at most i edges

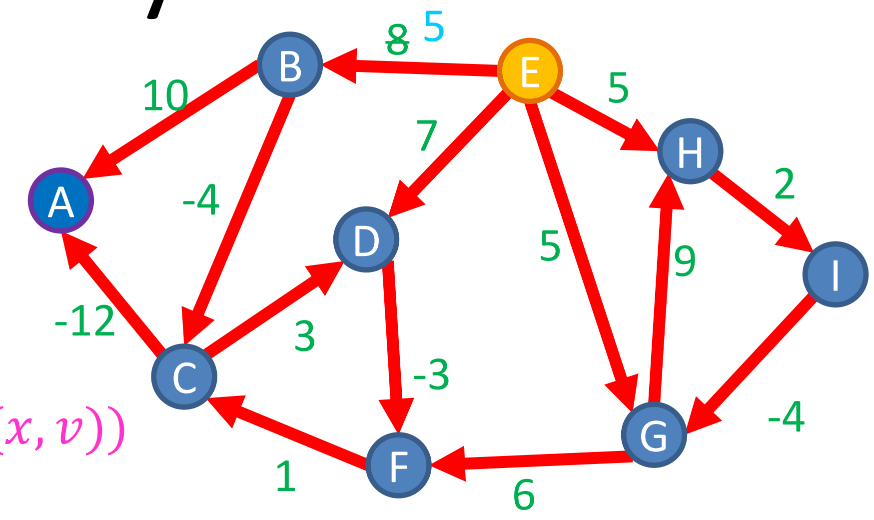
$$Short(i,v) = \min\begin{cases} \min_x \left(Short(i-1,x) + w(x,v) \right) \\ Short(i-1,v) \end{cases}$$



$i =$ \ $v =$	A	B	C	D	E	F	G	H	I
0	∞	∞	∞	∞	**0**	∞	∞	∞	∞
1	∞	5	∞	7	0	∞	5	5	∞
2	15	5	1	7	0	4	5	5	7
3	-8	5	1	7	0	4	3	5	7
4	-8	5	1	7	0	4	3	5	7
5	-8	5	1	7	0	4	3	5	7
6	-8	5	1	7	0	4	3	5	7
7	-8	5	1	7	0	4	3	5	7

Bellman Ford: Dynamic

Each node will update its neighbors if edge weight changes

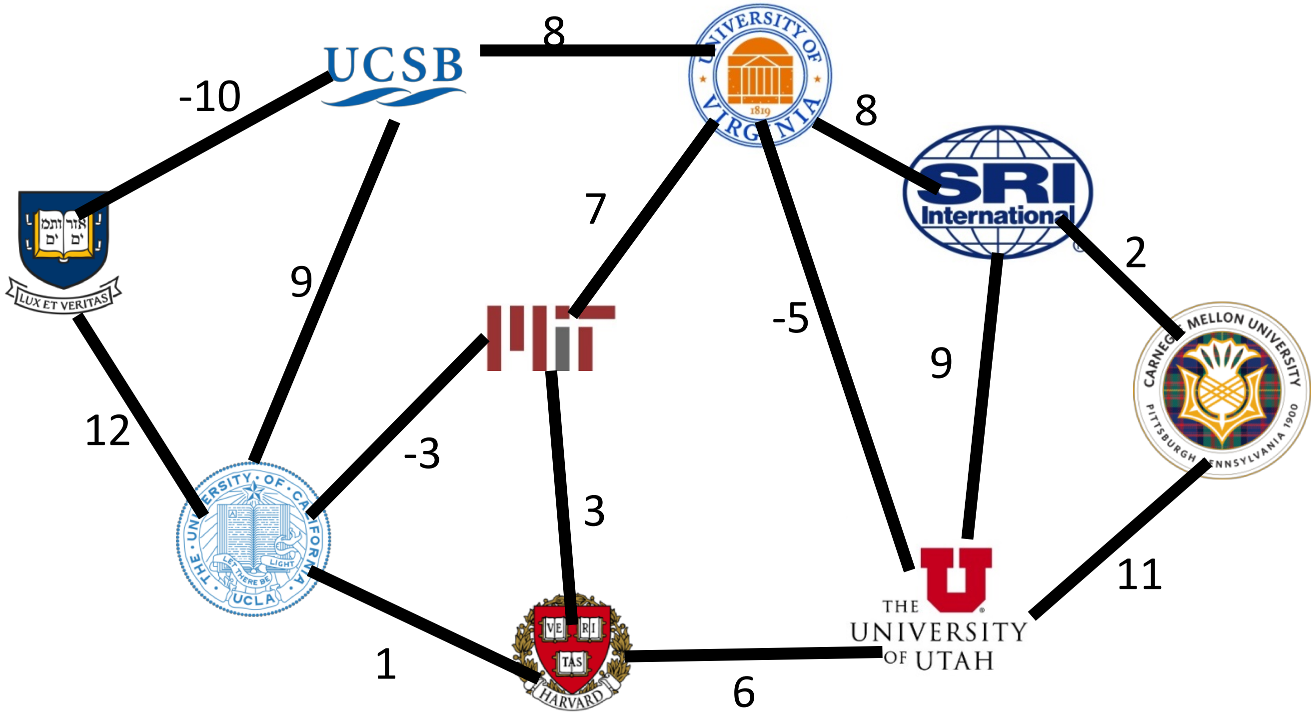


$Short(i, v) =$ weight of the shortest path from s to v using at most i edges

$$Short(i,v) = \min\begin{cases} \min_x \big(Short(i-1,x) + w(x,v) \big) \\ Short(i-1,v) \end{cases}$$

$i =$ \ $v =$	A	B	C	D	E	F	G	H	I
0	∞	∞	∞	∞	**0**	∞	∞	∞	∞
1	∞	**5**	∞	**7**	**0**	∞	**5**	**5**	∞
2	**15**	**5**	**1**	**7**	**0**	**4**	**5**	**5**	**7**
3	**-11**	**5**	**1**	**4**	**0**	**4**	**3**	**5**	**7**
4	**-11**	**5**	**1**	**4**	**0**	**4**	**3**	**5**	**7**
5	**-11**	**5**	**1**	**4**	**0**	**4**	**3**	**5**	**7**
6	**-11**	**5**	**1**	**4**	**0**	**4**	**3**	**5**	**7**
7	**-11**	**5**	**1**	**4**	**0**	**4**	**3**	**5**	**7**

All-Pairs Shortest Path



Find the quickest way to get from each place to every other place

Given a graph $G = (V, E)$ for each start node $s \in V$ and destination node $v \in V$ find the least-weight path from $s \rightarrow v$

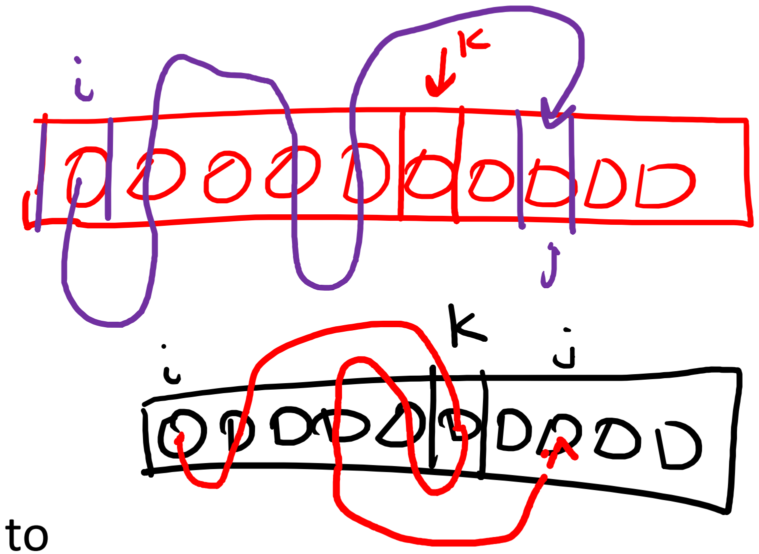
All-Pairs Shortest Path

- Can clearly be found in $O(V^2 \cdot E)$
  - Run Bellman-Ford with each node being the start

for each $s \in V$: V times

$BellmanFord(s)$ $O(V \cdot E)$

Floyd-Warshall

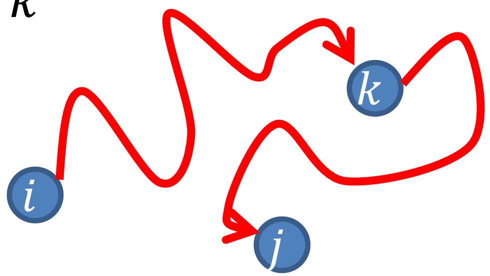


- Finds all-pairs shortest paths in $\Theta(V^3)$
- Uses Dynamic Programming

$Short(i,j,k) =$ the length of the shortest path from node i to node j using only intermediate nodes $1, \ldots, k$

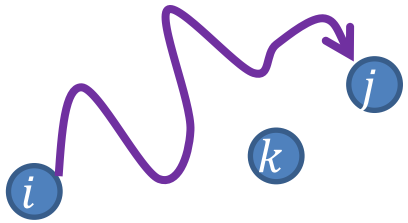
Two options:

Shortest path from i to j includes k



OR

Shortest path from i to j excludes k



$$Short(i,j,k) = \min \begin{cases} Short(i,k,k-1) + Short(k,j,k-1) \\ Short(i,j,k-1) \end{cases}$$

Shortest Paths Review

- Single Source Shortest Paths
  - Dijkstra's Algorithm $\Theta(E \log V)$
    - No negative edge weights
  - Bellman-Ford $\Theta(EV)$
    - First Dynamic Programming Algorithm
    - Allows negative edge weights (finds negative weight cycles)
    - Update memory in $\Theta(E)$ time on edge weight updates
- All Pairs Shortest Paths
  - Floyd-Warshall $\Theta(V^3)$
    - Allows negative edge weights