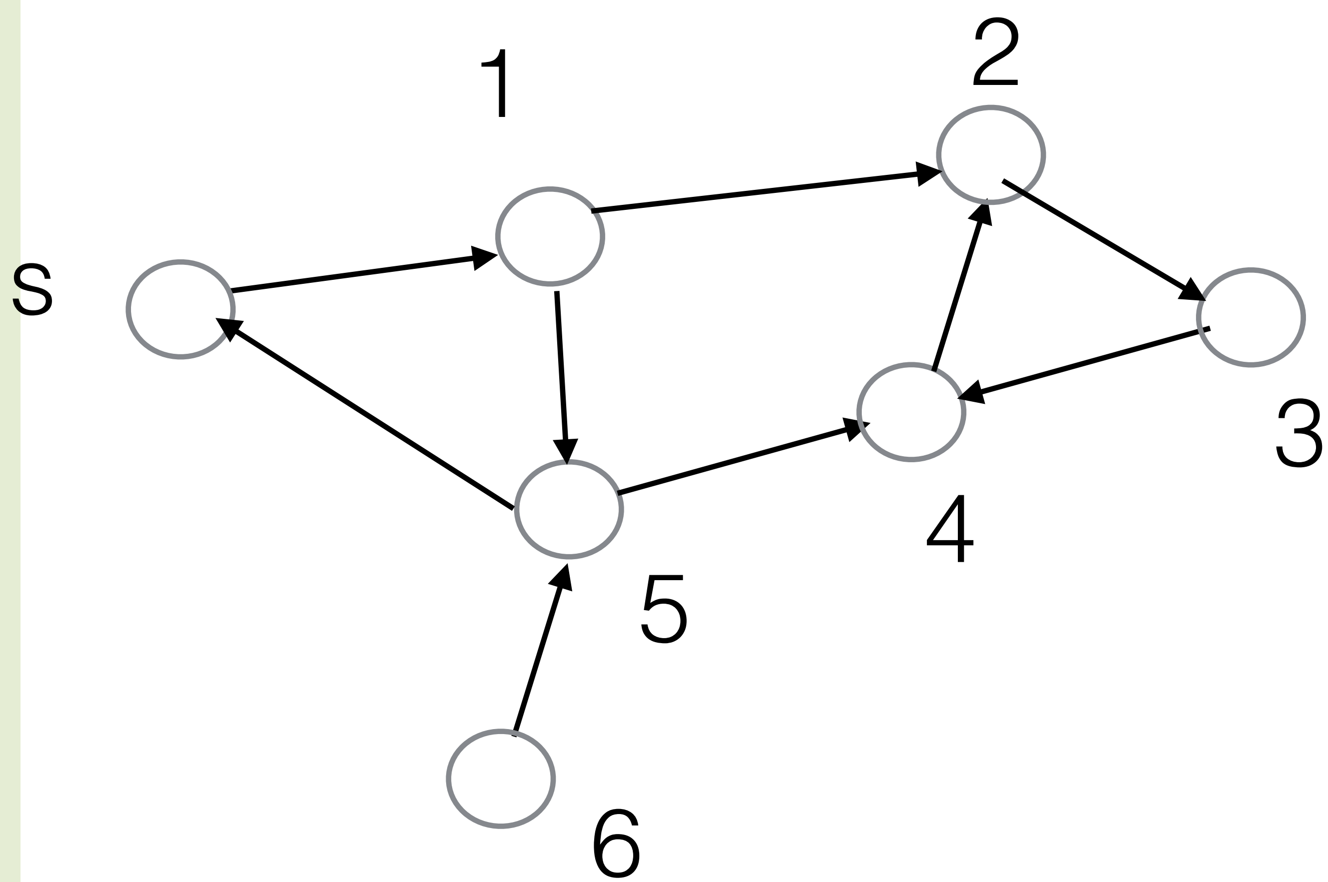
More Shortest Paths

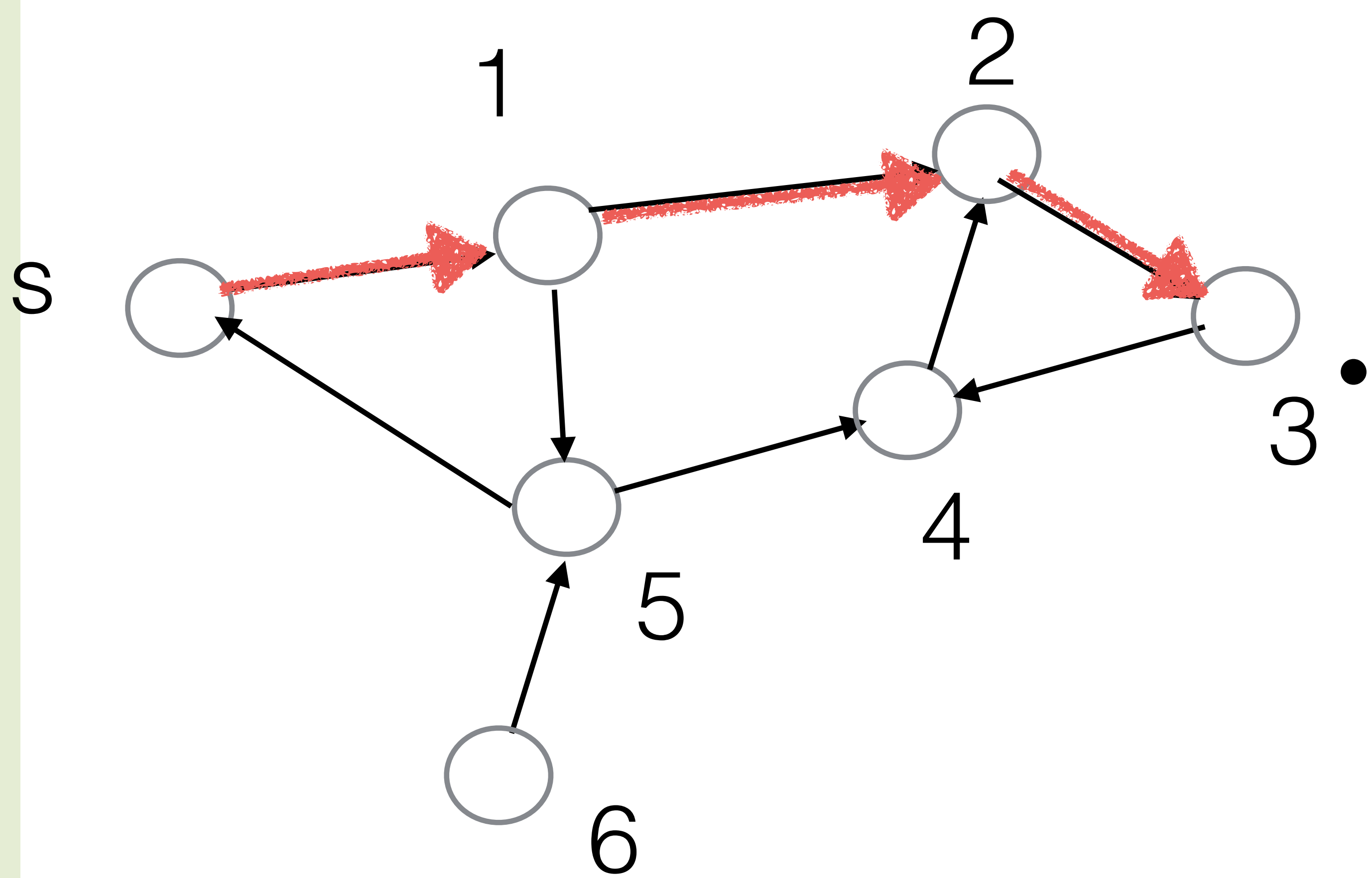
Lecture 20

Shortest Paths

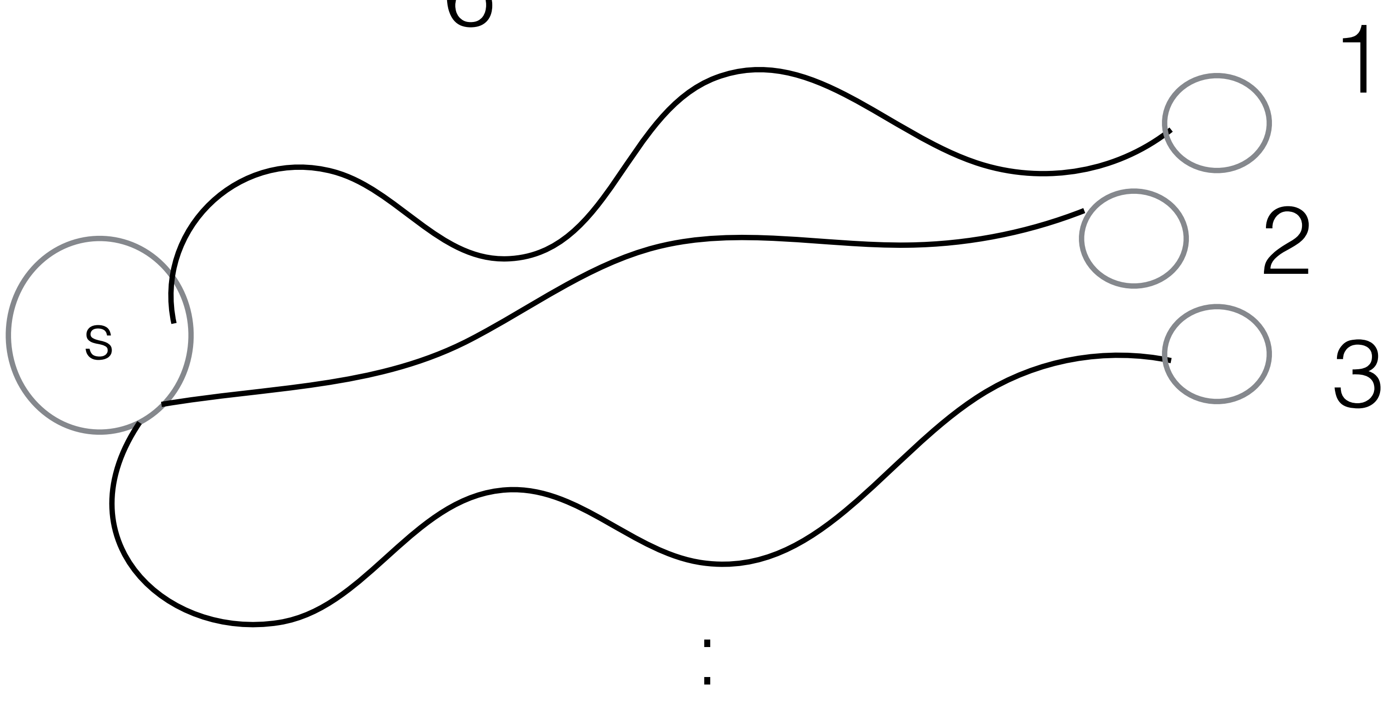


- Single source shortest path (one s, all t)
- All pairs shortest paths (all s, all t)

Shortest Paths



- Single source shortest path (one s, all t) : Dijkstra



Dijkstra

a.k.a "Closest first search"

Algorithm:

if all $w(e) \geq 0$ then

each node leaves priority queue once

≤ 1 priority queue operation per edge

$O(|E| \log V)$

if there is $w(e) < 0$ then

$O(2^{|V|})$ time

Every SSSP algorithm

```
INITSSSP(s):
    dist(s) ← 0
    pred(s) ← NULL
    for all vertices v ≠ s
        dist(v) ← ∞
        pred(v) ← NULL
```

```
GENERICSSSP(s):
    INITSSSP(s)
    put s in the bag
    while the bag is not empty
        take u from the bag
        for all edges u→v
            if u→v is tense
                RELAX(u→v)
                put v in the bag
```

Exponential time if weights negative.

Shimbel-Bellman-Ford

SHIMBEL-MOORE-WOODBURY-DANTZIG-BELLMAN-FORD-BROSH:
Relax ***ALL*** the tense edges and recurse.

```
SHIMBELSSSP(s)
    INITSSSP(s)
    repeat V times:
        for every edge u→v
            if u→v is tense
                RELAX(u→v)
    for every edge u→v
        if u→v is tense
            return "Negative cycle!"
```

Strong claim:
Worst case O(VE) time

How do I know V times is enough?

Dynamic Programming!

Shimbel-Bellman-Ford

```
SHIMBELSSSP(s)
  INITSSSP(s)
  repeat V times:
      for every edge u→v
          if u→v is tense
              RELAX(u→v)
  for every edge u→v
      if u→v is tense
          return "Negative cycle!"
```

Dynamic Programming!
Recurrence for shortest path dist?

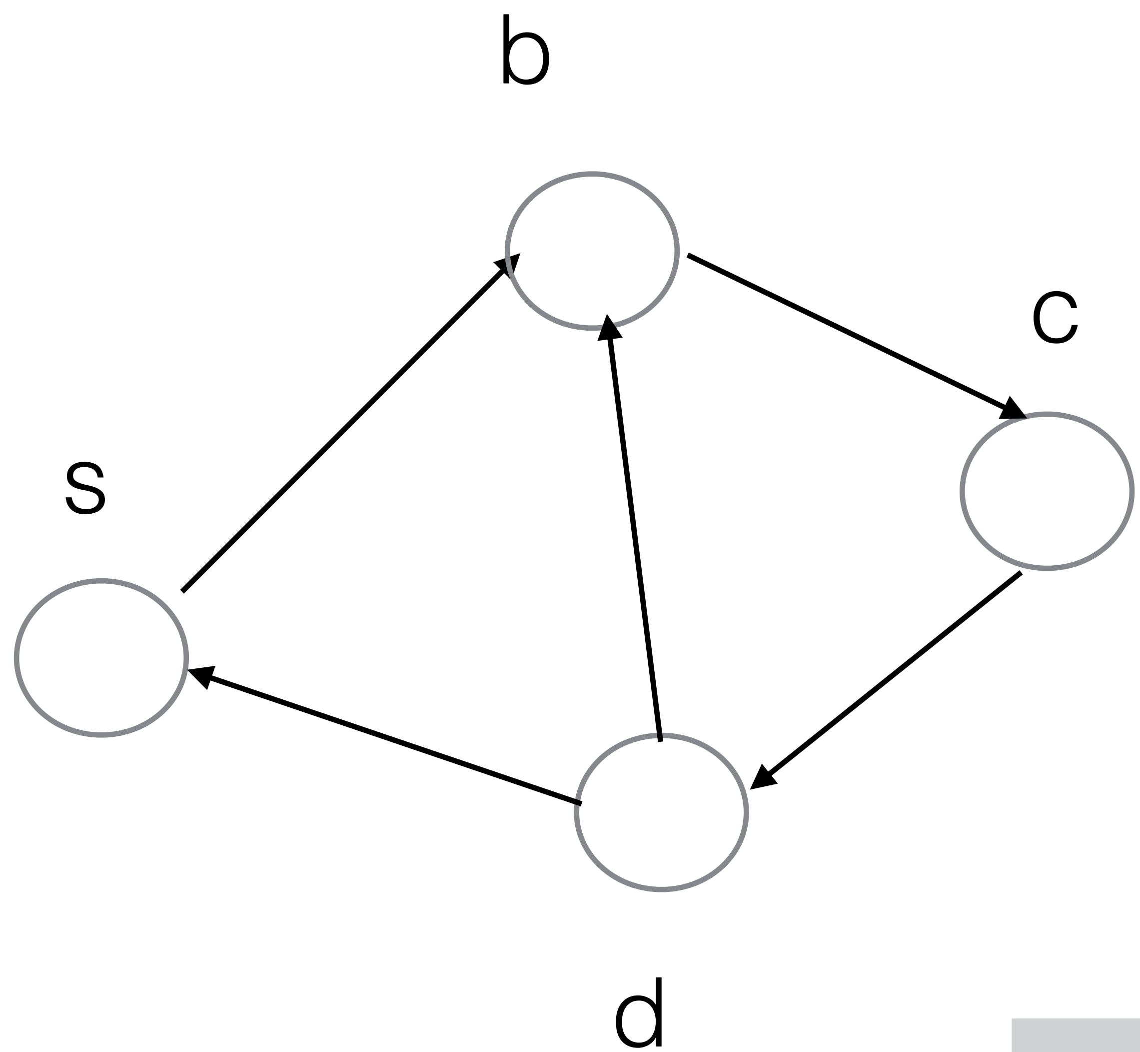
dist(v) =shortest path distance form s to v

Problem with recurrence?

dist(v) =

0	if v=s
$\min_{u \to v} \{w(u \to v) + dist(u)\}$	o.w

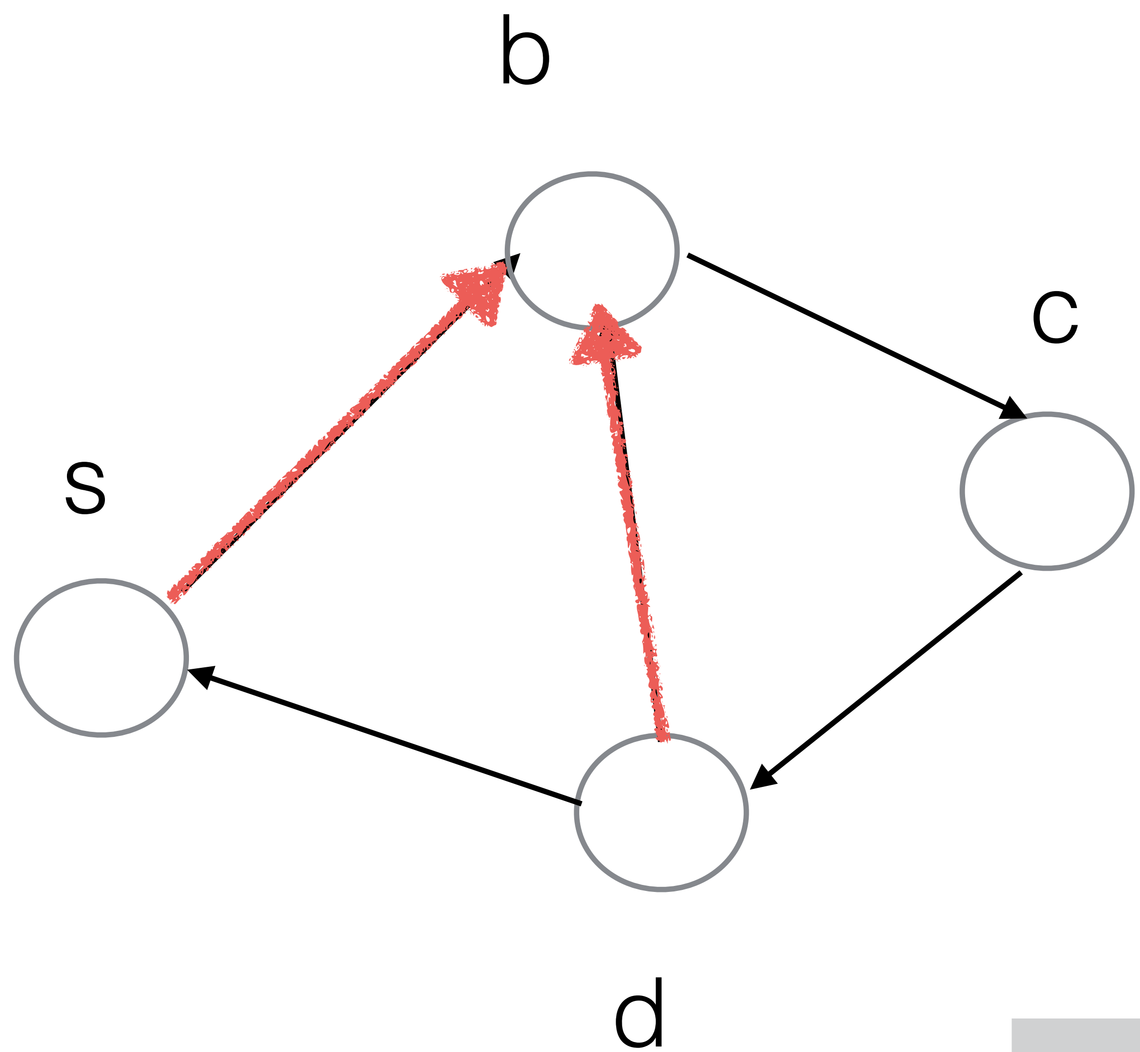
Shimbel-Bellman-Ford



dist(v) =

0	if v=s
$\min_{u \to v} \{w(u \to v)+dist(u)\}$	o.w

Shimbel-Bellman-Ford



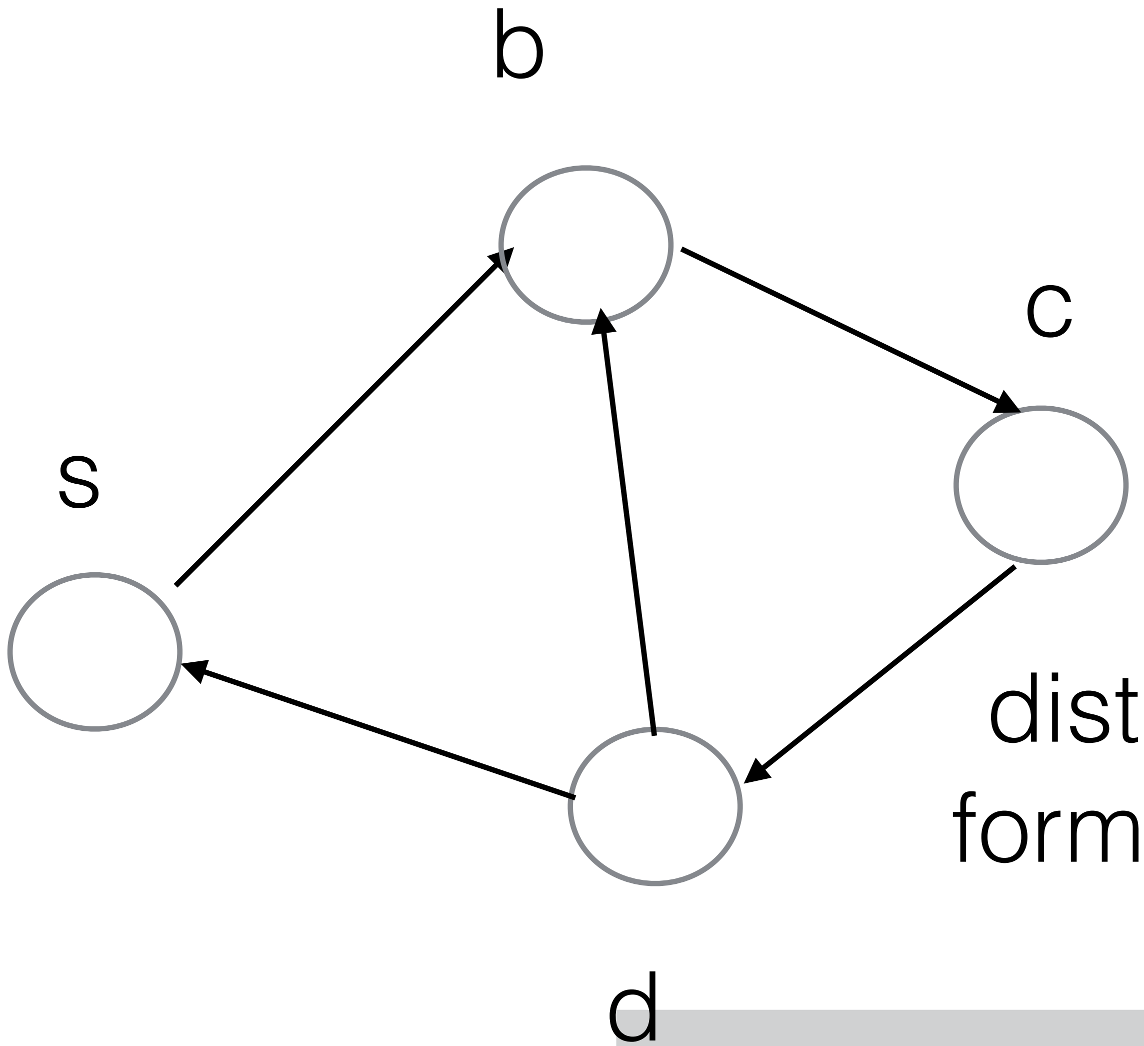
dist(b)?

needs dist(d)
needs dist(c)
needs dist(b)

dist(v) =

0	if v=s
$\min_{u \to v} \{w(u \to v)+dist(u)\}$	o.w

Shimbel-Bellman-Ford



dist(b)?

needs dist(d)
needs dist(c)
needs dist(b)

$dist_{i}(v)$=shortest path distance form s to v using at most **i** edges

$dist_{i}(v) =$

0	if v=s
∞	i=0,v≠s
$\min\{dist_{i-1}(v), \min_{u\to v}\{w(u\to v)+dist_{i-1}(u)\}\}$	ow

Shimbel-Bellman-Ford

$$dist_i(v) = \begin{cases} 0 & \text{if } i = 0 \text{ and } v = s \\ \infty & \text{if } i = 0 \text{ and } v \neq s \\ \min \left\{ \begin{array}{c} dist_{i-1}(v), \\ \min\limits_{u \to v \in E} (dist_{i-1}(u) + w(u \to v)) \end{array} \right\} & \text{otherwise} \end{cases}$$

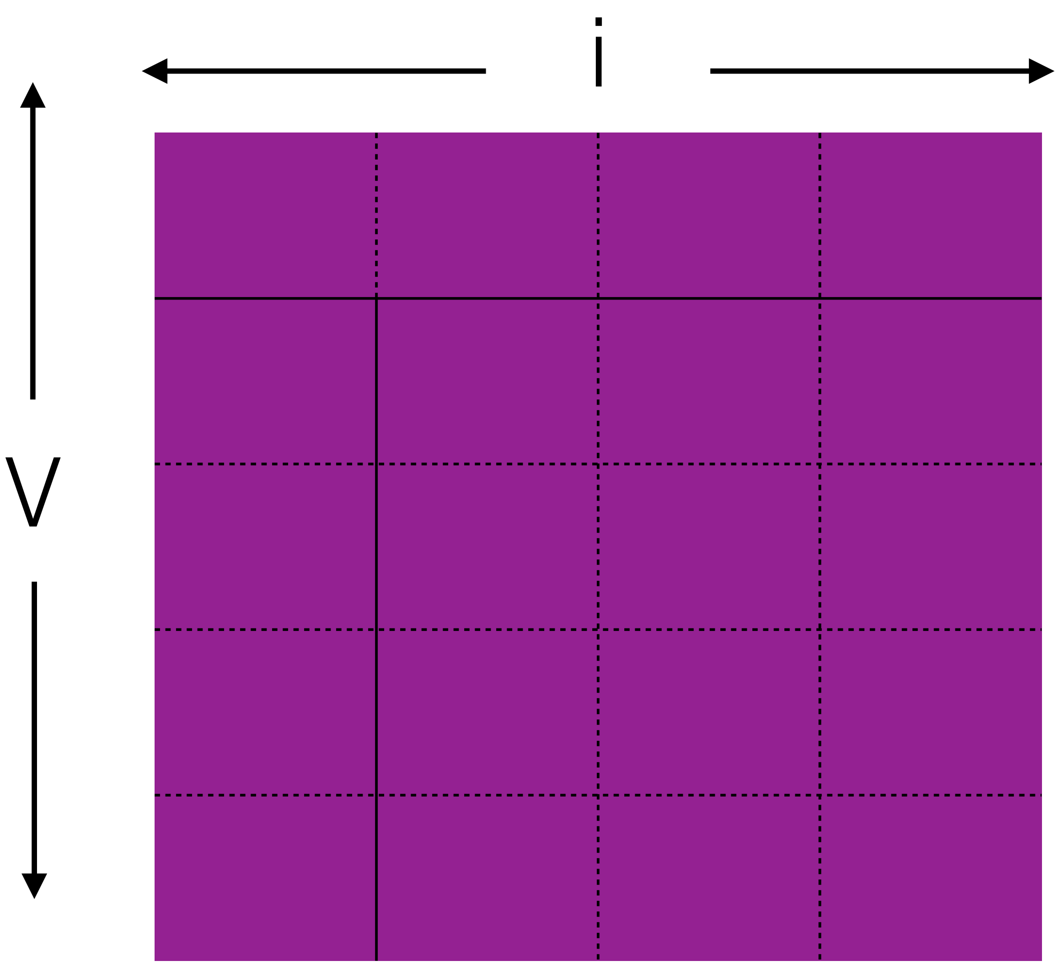
how large can i get?

no larger than |V| if no negative cycles

True shortest path distance is $dist_{|V|}(v)$

Shimbel-Bellman-Ford

$$dist_i(v) = \begin{cases} 0 & \text{if } i = 0 \text{ and } v = s \\ \infty & \text{if } i = 0 \text{ and } v \neq s \\ \min \left\{ \begin{array}{c} dist_{i-1}(v), \\ \min\limits_{u \to v \in E} (dist_{i-1}(u) + w(u \to v)) \end{array} \right\} & \text{otherwise} \end{cases}$$

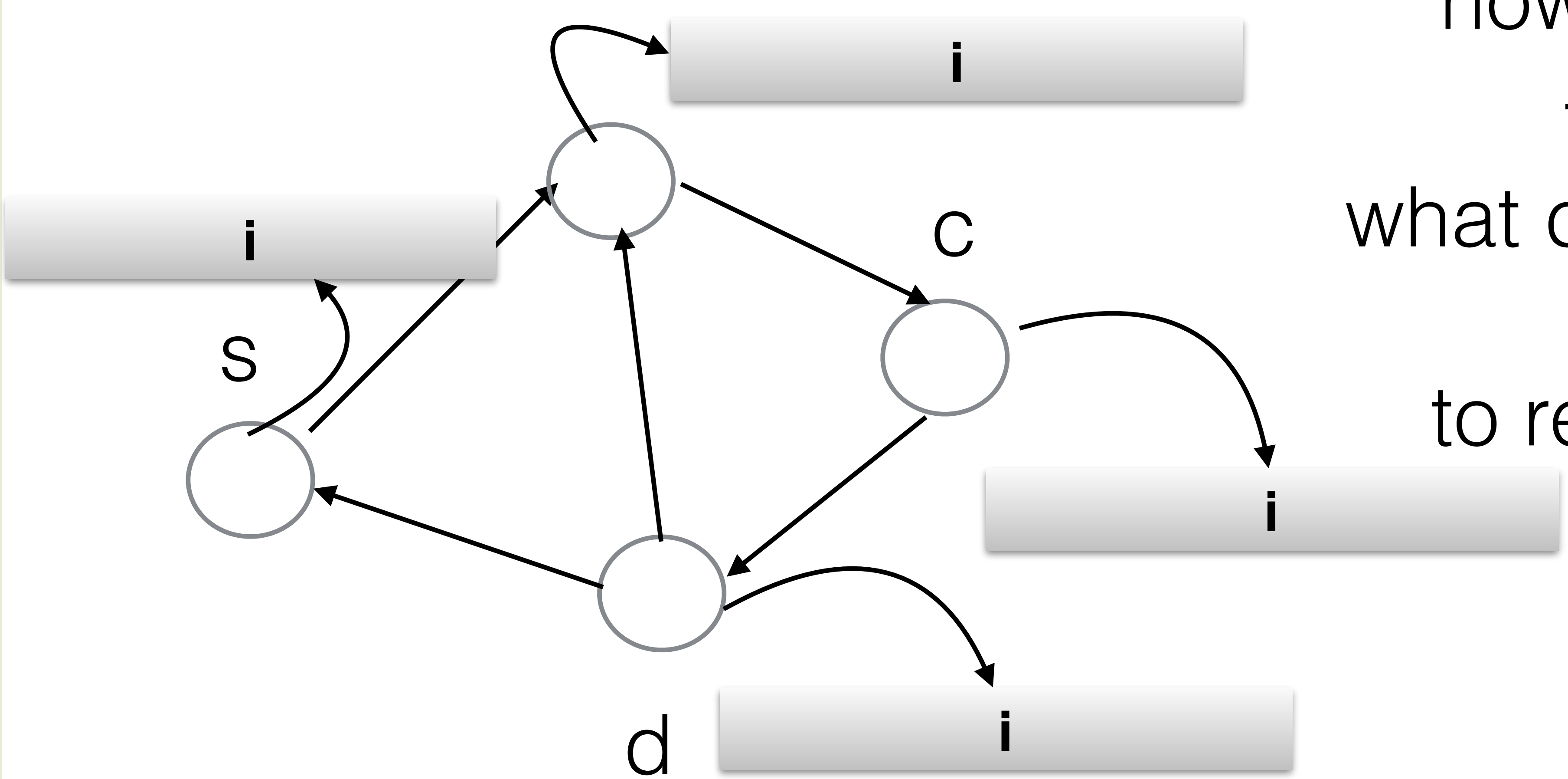


how to memoize it?

- 2d array!
- number vertices arbitrarily

Shimbel-Bellman-Ford

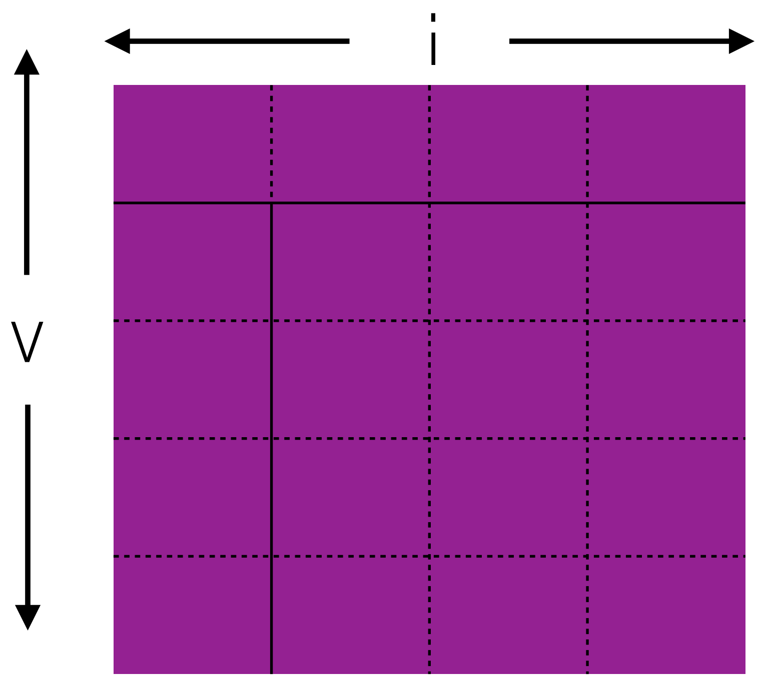
$$dist_i(v) = \begin{cases} 0 & \text{if } i = 0 \text{ and } v = s \\ \infty & \text{if } i = 0 \text{ and } v \neq s \\ \min \left\{ \begin{array}{c} dist_{i-1}(v), \\ \min\limits_{u \to v \in E} (dist_{i-1}(u) + w(u \to v)) \end{array} \right\} & \text{otherwise} \end{cases}$$

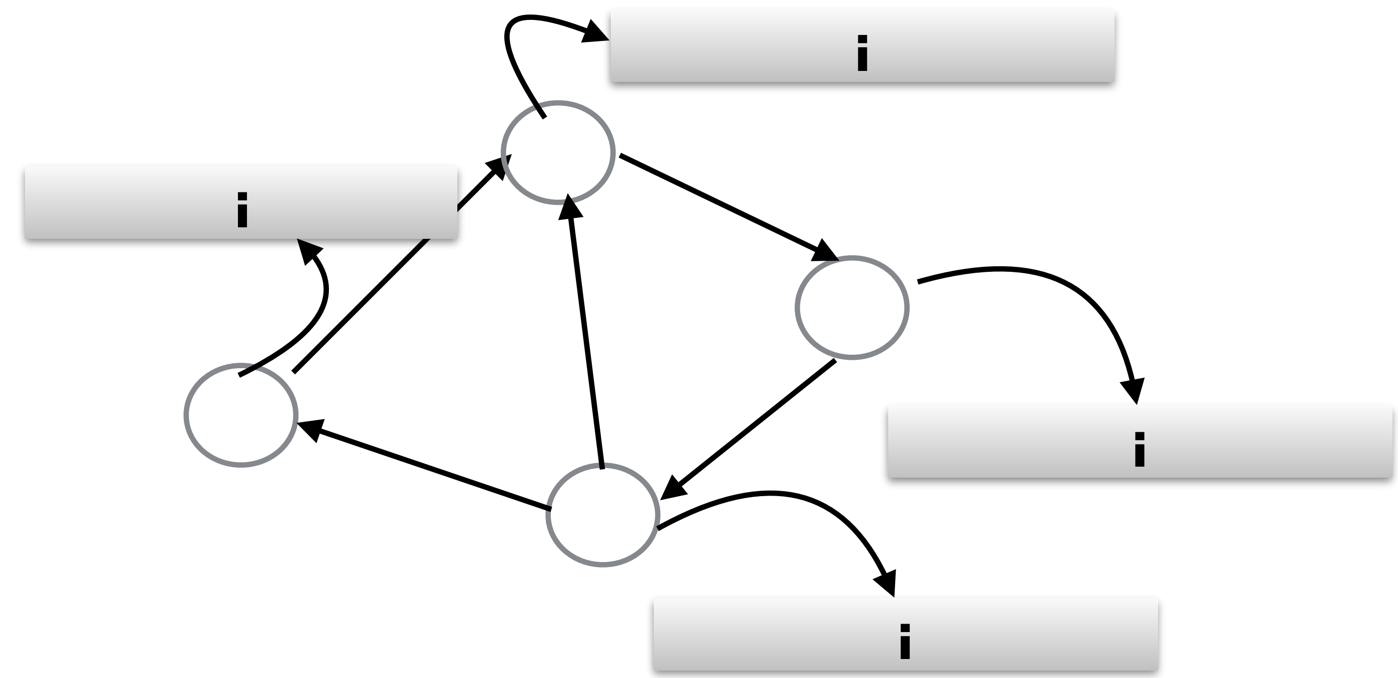


how to memoize it?
- graph itself

what data structure am I given
to represent graph?

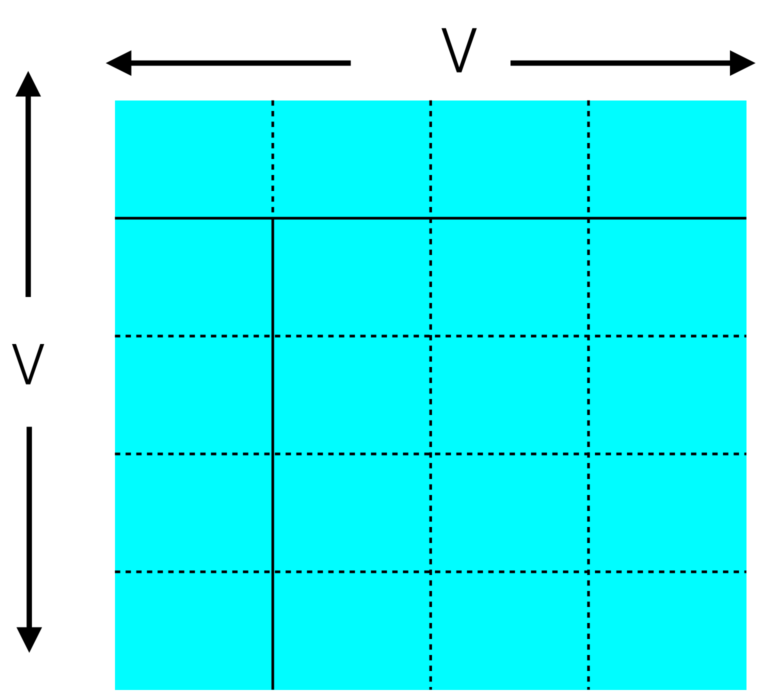
Shimbel-Bellman-Ford

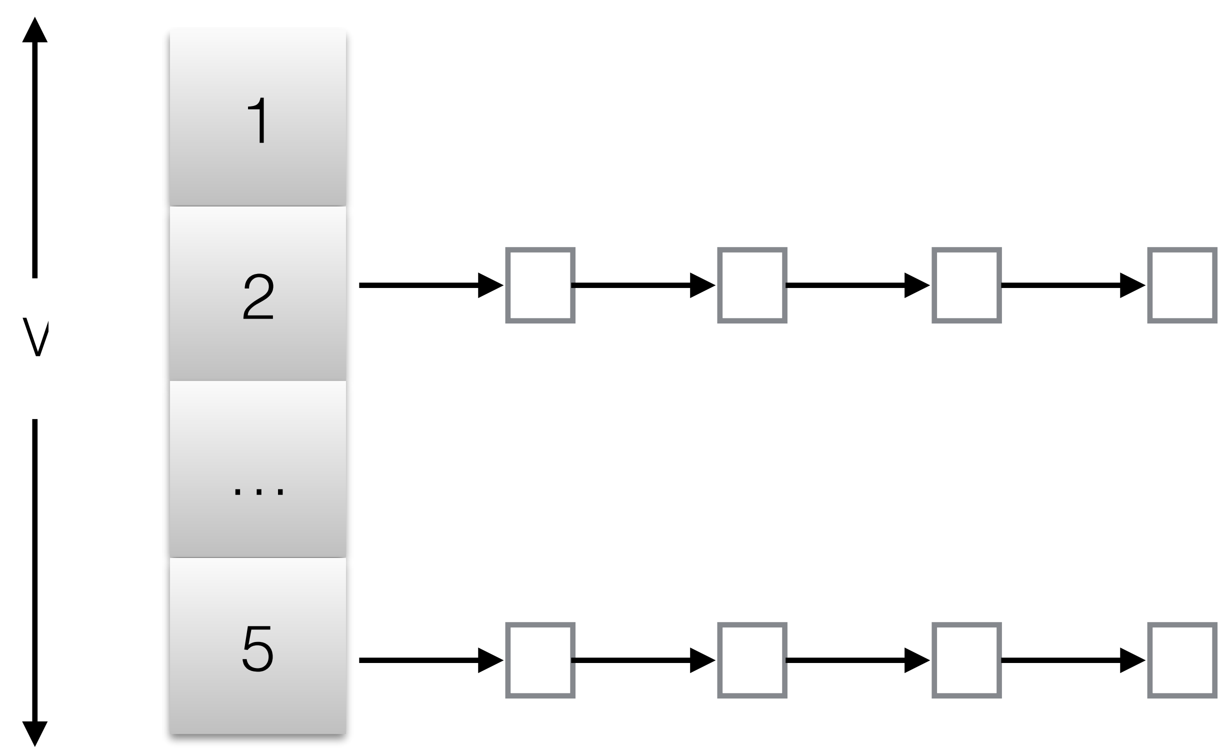




Adjacency matrix

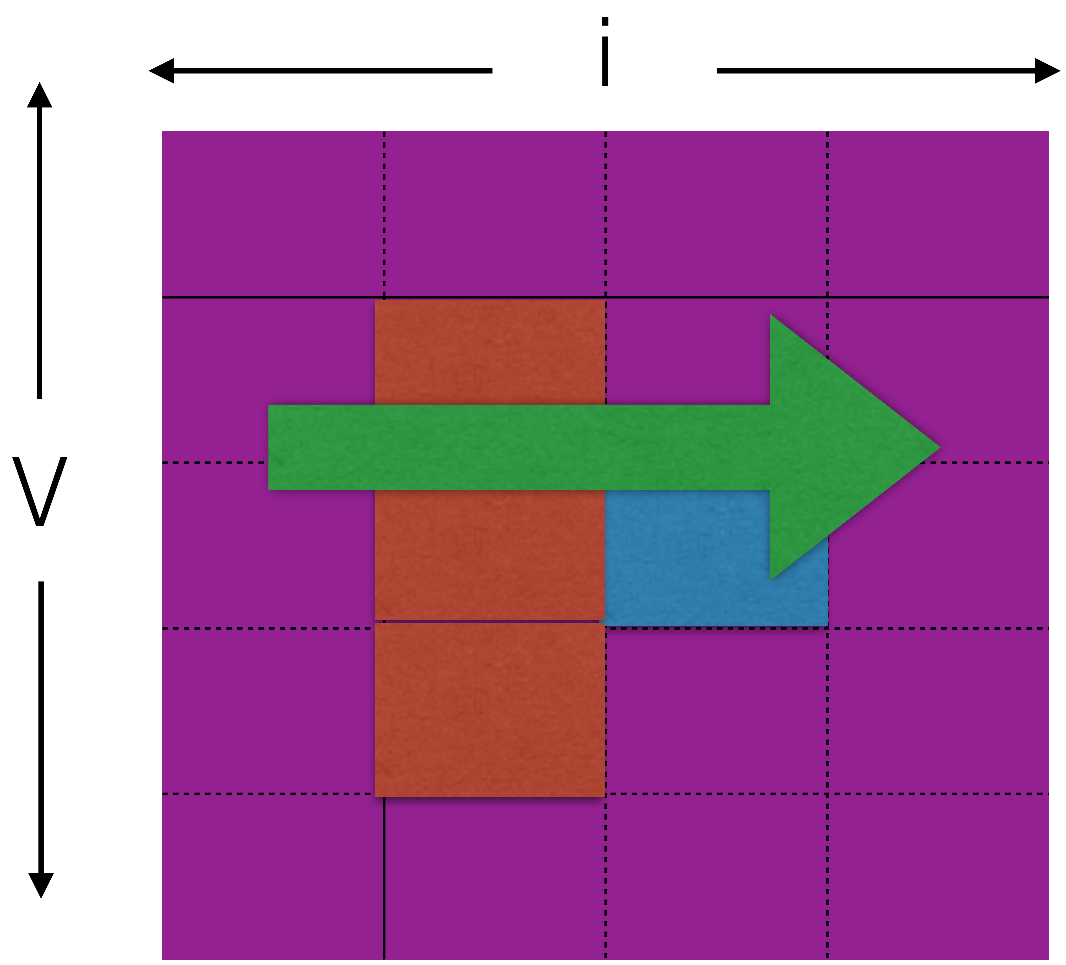
Adjacency list





Shimbel-Bellman-Ford

$$dist_i(v) = \begin{cases} 0 & \text{if } i = 0 \text{ and } v = s \\ \infty & \text{if } i = 0 \text{ and } v \neq s \\ \min \left\{ \begin{array}{c} dist_{i-1}(v), \\ \min\limits_{u \to v \in E} (dist_{i-1}(u) + w(u \to v)) \end{array} \right\} & \text{otherwise} \end{cases}$$



suppose i use 2d array

memorization order?

outer loop from left to right

any way for inner loop

running time? $O(V^2)$?

O(VE)

Shimbel-Bellman-Ford

```
SHIMBELDP(s)
  dist[0,s] ← 0
  for every vertex v ≠ s
      dist[0,v] ← ∞
  for i ← 1 to V − 1
      for every vertex v
          dist[i,v] ← dist[i − 1,v]
          for every edge u→v
              if dist[i,v] > dist[i − 1,u] + w(u→v)
                  dist[i,v] ← dist[i − 1,u] + w(u→v)
```

2 nested loops.
equivalent to doing something for every edge

Shimbel-Bellman-Ford

```
SHIMBELDP2(s)
  dist[0, s] ← 0
  for every vertex v ≠ s
      dist[0, v] ← ∞
  for i ← 1 to V − 1
      for every vertex v
          dist[i, v] ← dist[i − 1, v]
      for every edge u→v
          if dist[i, v] > dist[i, u] + w(u→v)
              dist[i, v] ← dist[i, u] + w(u→v)
```

If edge is tense, relax it!

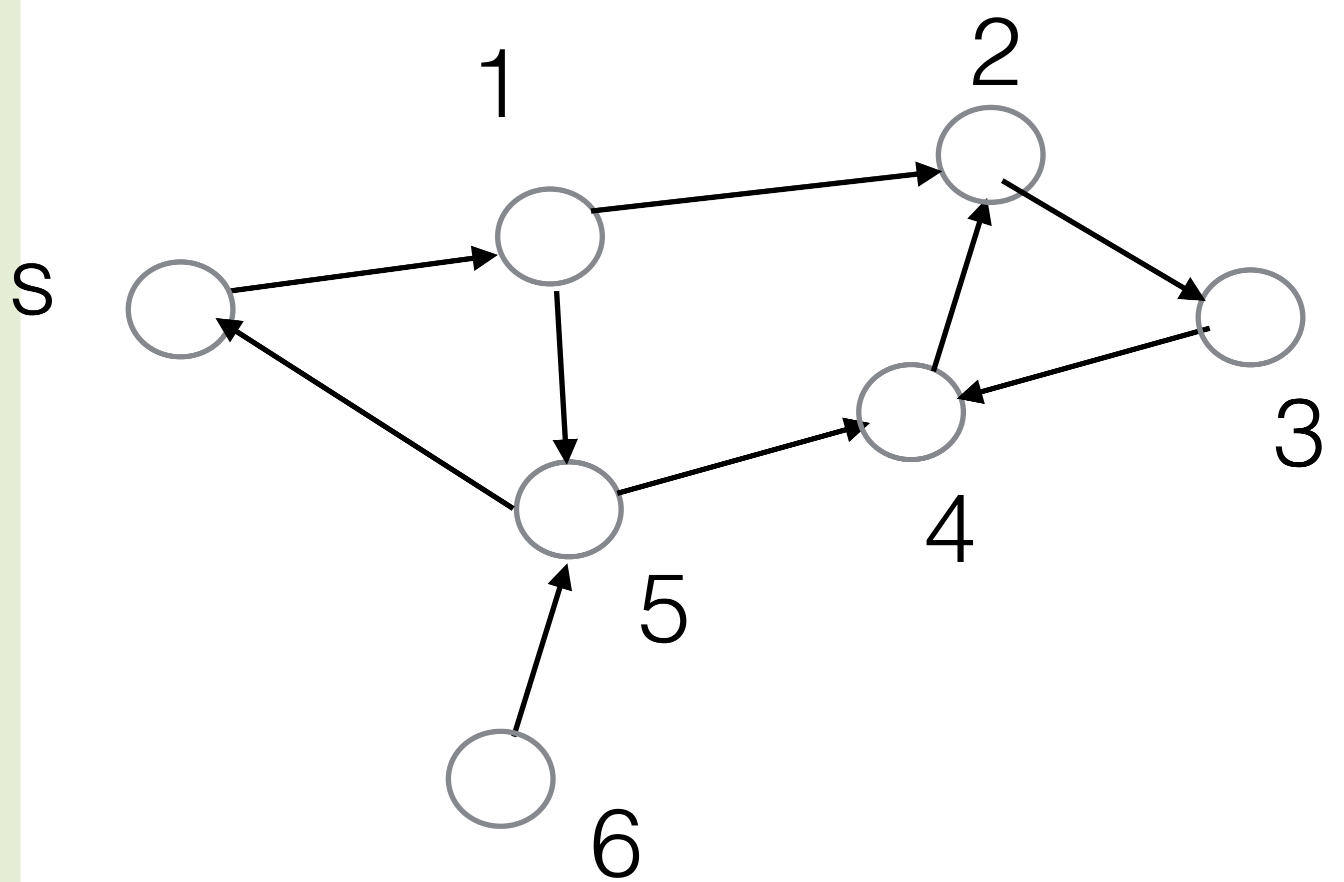
Shimbel-Bellman-Ford

no need for 2d Array,
can store dist(v) on node v

```
ShimbelDP3(s)
    dist[s] ← 0
    for every vertex v ≠ s
        dist[v] ← ∞
    for i ← 1 to V − 1
        for every edge u→v
            if dist[v] > dist[u] + w(u→v)
                dist[v] ← dist[u] + w(u→v)
```

every time we pass through the loop, we consider paths with more edges. Only need to go up to |V| edge paths.

Shortest Paths



- Single source shortest path (one s, all t)
- All pairs shortest paths dist(u,v) for all u,v

Output 2d array: dist[u][v]
can't hope for faster than $O(V^2)$ algorithm

All pairs? BF n times? $O(V^2E)$
There is faster algorithm $O(V^3)$ Floyd-Warshall

Floyd-Warshall

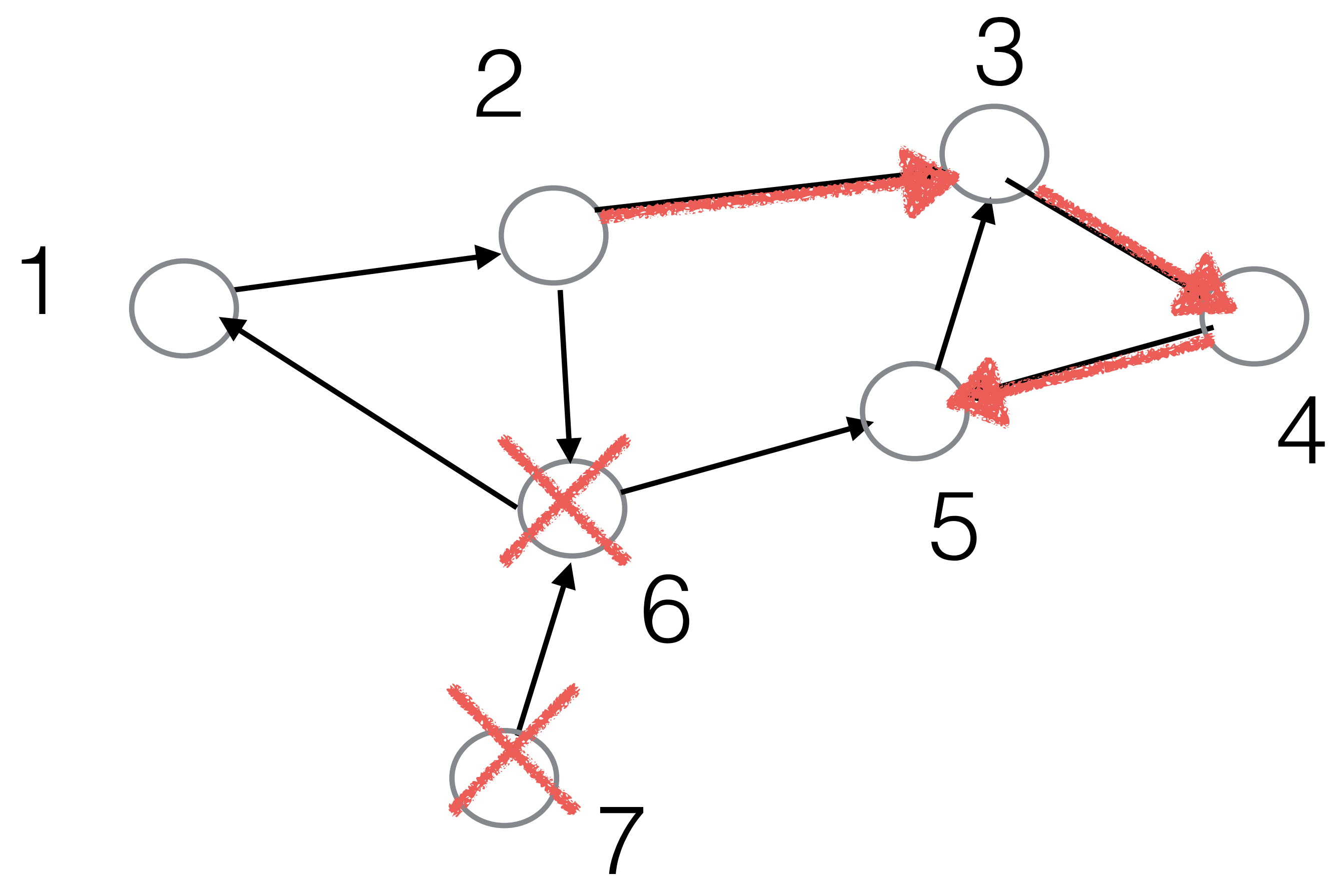
dist(u,v,?) =shortest path distance form u to v

dist(u,v,?) =

	if v=u
	o.w

Floyd-Warshall

dist(u,v,r) =shortest path distance
form u to v using vertices only indexed from 1 to r



length of shortest path from u=2 to v=5 can only use nodes 1,2,3,4 (at most r=4 intermediate nodes)

Floyd-Warshall

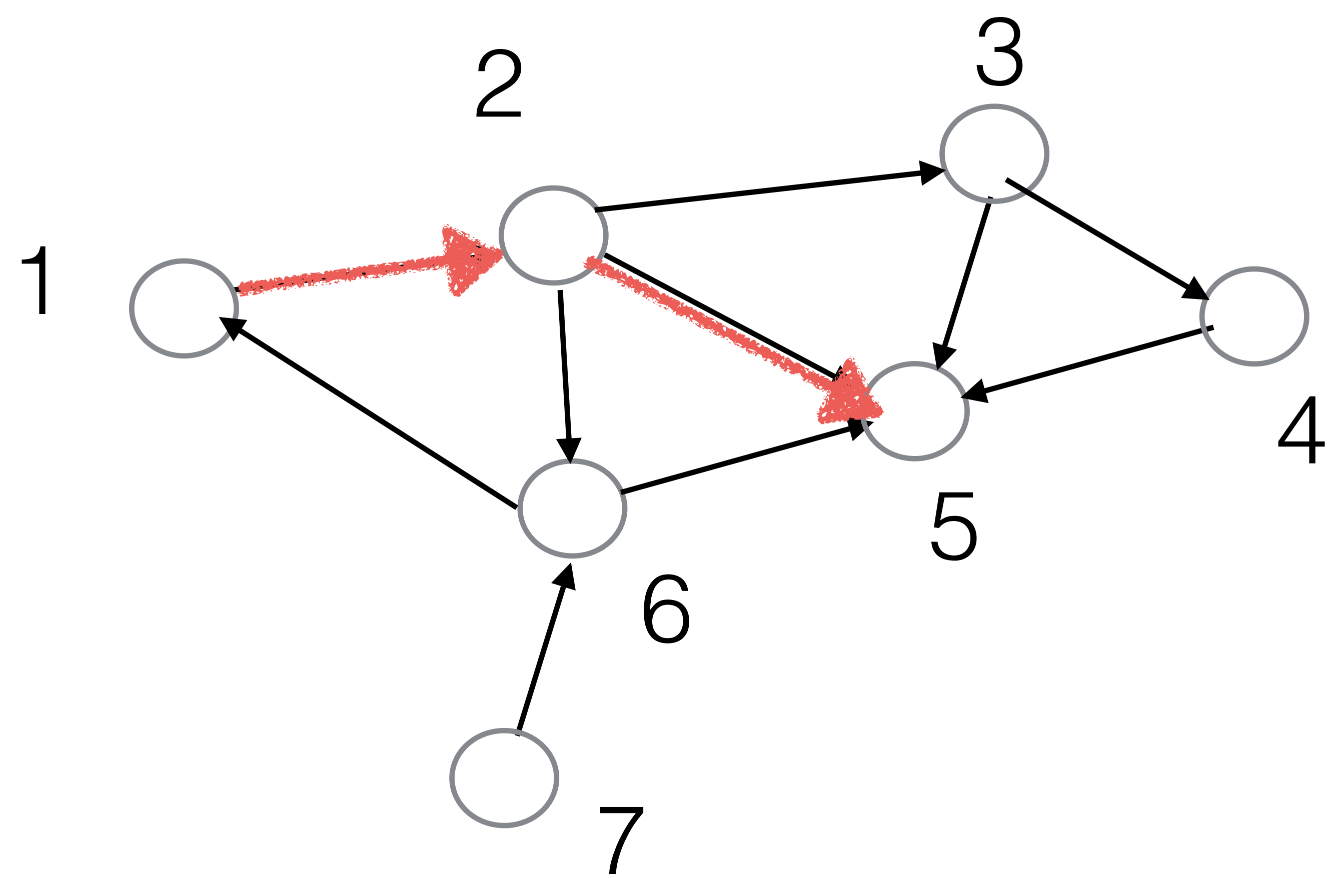
dist(u,v,r) =shortest path distance
form u to v using vertices only indexed from 1 to r

dist(u,v,r) =

0	if v=u
$w(u \to v)$	r=0
	r>0

Floyd-Warshall

dist(u,v,r) =shortest path distance
form u to v using vertices only indexed from 1 to r



r=3

shortest path from 1 to 5 using vertices 1,2,3
is the same as shortest path from 1 to 5 using vertices 1,2

Floyd-Warshall

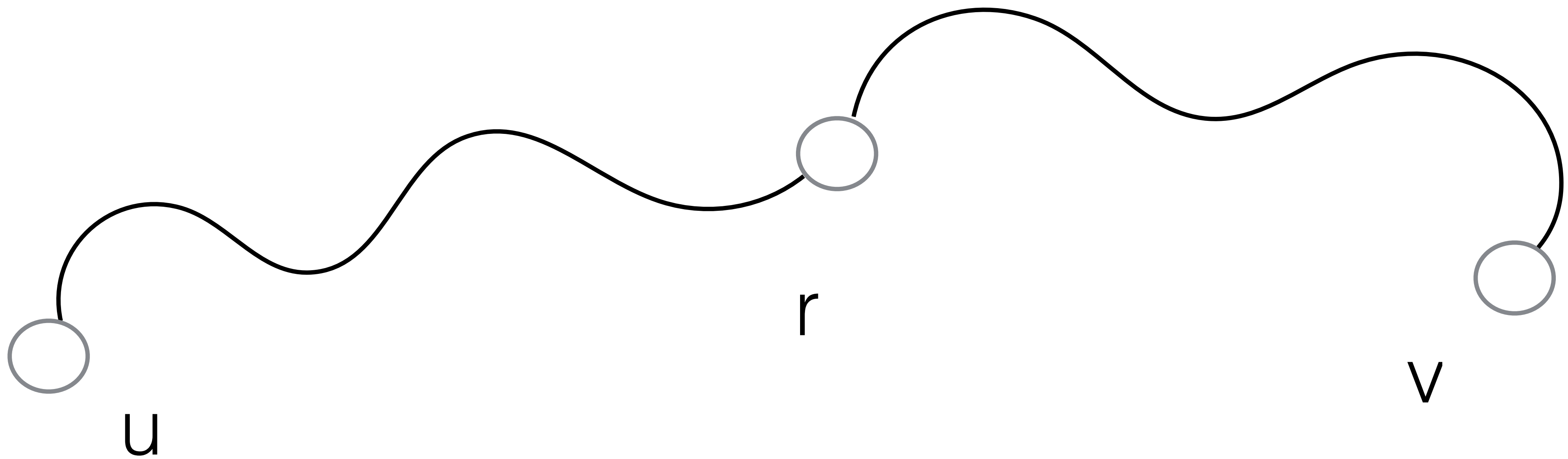
dist(u,v,r) =shortest path distance
form u to v using vertices only indexed from 1 to r

dist(u,v,r) =

0	if v=u
$w(u \rightarrow v)$	r=0
min{dist(u,v,r-1), ?}	r>0

Floyd-Warshall

dist(u,v,r) =shortest path distance
form u to v using vertices only indexed from 1 to r



Floyd-Warshall

dist(u,v,r) =shortest path distance
form u to v using vertices only indexed from 1 to r

How to memoize?

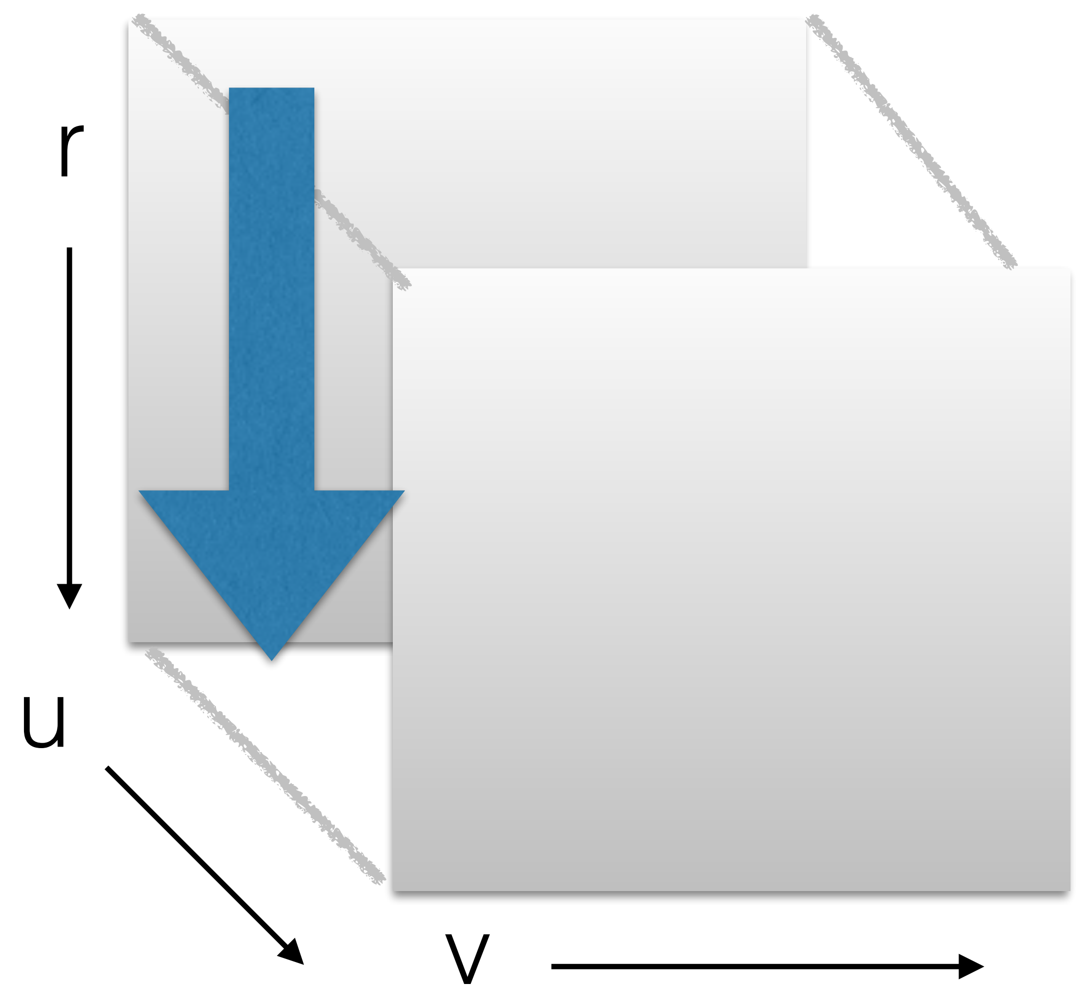
dist(u,v,r) =

0	if v=u
$w(u \to v)$	r=0
min{dist(u,v,r-1),dist(u,r,r-1)+ dist(r,v,r-1) }	r>0

Floyd-Warshall

How to memoize? 3d array!

$$dist(u,v,r) = \begin{cases} w(u \to v) & \text{if } r = 0 \\ \min\{dist(u,v,r-1),\ dist(u,r,r-1) + dist(r,v,r-1)\} & \text{otherwise} \end{cases}$$



O(n^3) time