ALL PAIRS SHORTEST PATHS

David Kauchak
CS 140 – Fall 2022

1

Admin

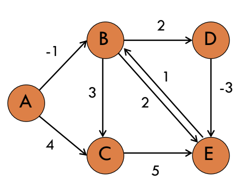
Assignment 8 (how did it go?)

Assignment 9 (out later today)

Checkpoint 3: next Thursday!

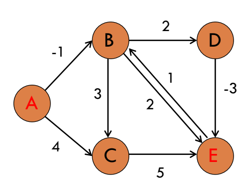
2

Shortest Paths



3

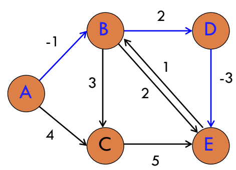
Shortest Paths



What is the shortest path from A to E?

4

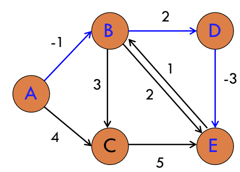
Shortest Paths



-2

5

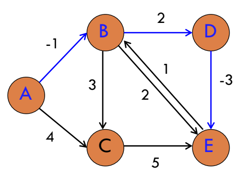
Shortest Paths



What algorithm would we use to calculate this?

6

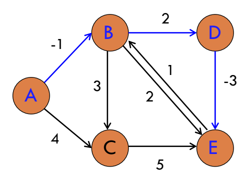
Shortest Paths



- Bellman-Ford (since the graph has negative edges)
- O(VE)

7

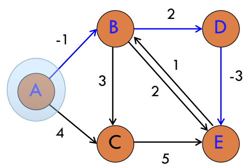
Shortest Paths



- Bellman-Ford (since the graph has negative edges)
- O(VE)
- Called a single-source shortest path algorithm. Why?

8

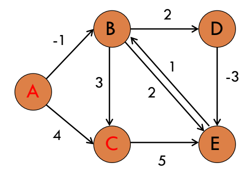
Shortest Paths



- Bellman-Ford (since the graph has negative edges)
- O(VE)
- Calculate all paths from a **single vertex.**

9

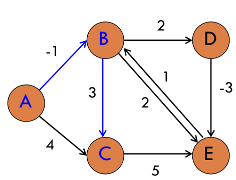
Shortest Paths



What is the shortest path from A to C?
If we already calculated A to E using Bellman-Ford do we need to do any work?

10

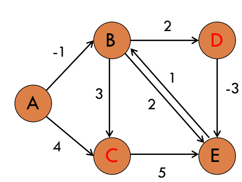
Shortest Paths



No new calculations!
Bellman-Ford calculates all shortest paths starting at A.

11

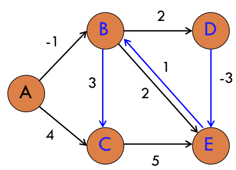
Shortest Paths



What is the shortest path from D to C?
If we already calculated A to E using Bellman-Ford do we need to do any work?

12

Shortest Paths

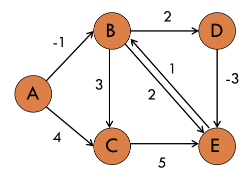


Different source.
Have to run Bellman-Ford again!

13

All pairs shortest paths

All pairs shortest paths: calculate the shortest paths between *all* vertices



14

All pairs shortest paths

All pairs shortest paths: calculate the shortest paths between *all* vertices

Easy solution?

15

All pairs shortest paths

All pairs shortest paths: calculate the shortest paths between *all* vertices

Run Bellman-Ford from each vertex!

Running time (in terms of E and V)?

16

All pairs shortest paths

All pairs shortest paths: calculate the shortest paths between *all* vertices

Run Bellman-Ford from each vertex!

O(V^2E)
- Bellman-Ford: O(VE)
- V calls, one for each vertex

17

Floyd-Warshall: key idea

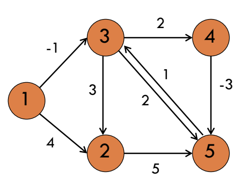
Label all vertices with a number from 1 to V

$d_{ij}{}^k$ = shortest path from vertex i to vertex j using only vertices $\{1, 2, \ldots, k\}$

18

Floyd-Warshall: key idea

$d_{ij}{}^k$ = shortest path from vertex i to vertex j using only vertices $\{1, 2, \ldots, k\}$



What is $d_{15}{}^2$?

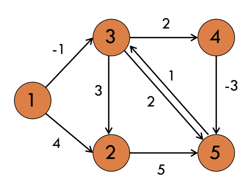
What is $d_{15}{}^3$?

What is $d_{41}{}^4$?

19

Floyd-Warshall: key idea

$d_{ij}{}^k$ = shortest path from vertex i to vertex j using only vertices $\{1, 2, \ldots, k\}$

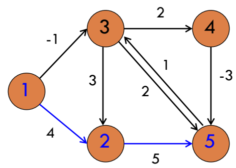


What is $d_{15}{}^2$?

20

Floyd-Warshall: key idea

d_{ij}^{k} = shortest path from vertex i to vertex j
using only vertices $\{1, 2, \dots, k\}$

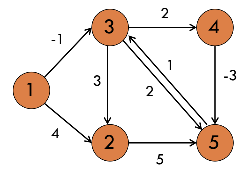


$d_{15}^{2} = 9$. Can only use 2.

21

Floyd-Warshall: key idea

d_{ij}^{k} = shortest path from vertex i to vertex j
using only vertices $\{1, 2, \dots, k\}$

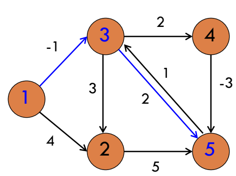


What is d_{15}^{3}?

22

Floyd-Warshall: key idea

d_{ij}^{k} = shortest path from vertex i to vertex j
using only vertices $\{1, 2, \dots, k\}$

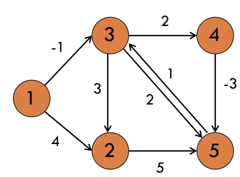


$d_{15}^{3} = 1$. Can't use vertex 4.

23

Floyd-Warshall: key idea

d_{ij}^{k} = shortest path from vertex i to vertex j
using only vertices $\{1, 2, \dots, k\}$

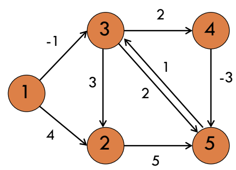


What is d_{41}^{4}?

24

Floyd-Warshall: key idea

$d_{ij}{}^{k}$ = shortest path from vertex i to vertex j
using only vertices $\{1, 2, \ldots, k\}$

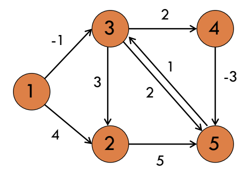


$d_{41}{}^{4} = \infty$. No possible path.

25

Floyd-Warshall: key idea

$d_{ij}{}^{k}$ = shortest path from vertex i to vertex j
using only vertices $\{1, 2, \ldots, k\}$

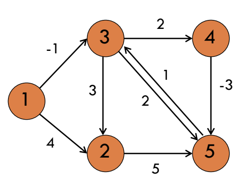


What is $d_{33}{}^{5}$?

26

Floyd-Warshall: key idea

$d_{ij}{}^{k}$ = shortest path from vertex i to vertex j
using only vertices $\{1, 2, \ldots, k\}$



$d_{33}{}^{5} = 0$. $d_{ii}{}^{k} = 0$ for all i.

27

Floyd-Warshall: key idea

Label all vertices with a number from 1 to V

$d_{ij}{}^{k}$ = shortest path from vertex i to vertex j
using only vertices $\{1, 2, \ldots, k\}$

If we want all possibilities, how many values are there (i.e. what is the size of $d_{ij}{}^{k}$)?

28

Floyd-Warshall: key idea

Label all vertices with a number from 1 to V

$d_{ij}{}^{k}$ = shortest path from vertex i to vertex j
using only vertices $\{1, 2, \ldots, k\}$

V^3

- i: all vertices
- j: all vertices
- k: all vertices

29

Floyd-Warshall: key idea

Label all vertices with a number from 1 to V

$d_{ij}{}^{k}$ = shortest path from vertex i to vertex j
using only vertices $\{1, 2, \ldots, k\}$

What is $d_{ij}{}^{V}$?

- Distance of the shortest path from i to j
- If we can calculate this, for all (i, j), we're done!

30

Recursive relationship

$d_{ij}{}^{k}$ = shortest path from vertex i to vertex j
using only vertices $\{1, 2, \ldots, k\}$

Assume we know $d_{ij}{}^{k}$

How can we calculate $d_{ij}{}^{k+1}$, i.e. shortest path now including vertex k+1? (Hint: in terms of $d_{ij}{}^{k}$)

Two options:
1) Vertex k+1 doesn't give us a shorter path
2) Vertex k+1 does give us a shorter path

31

Recursive relationship

$d_{ij}{}^{k}$ = shortest path from vertex i to vertex j
using only vertices $\{1, 2, \ldots, k\}$

Two options:
1) Vertex k+1 doesn't give us a shorter path
2) Vertex k+1 does give us a shorter path

$d_{ij}{}^{k+1}$ = ?

32

Recursive relationship

d_{ij}^{k} = shortest path from vertex i to vertex j using only vertices $\{1, 2, \ldots, k\}$

Two options:
1) Vertex k+1 doesn't give us a shorter path
2) Vertex k+1 does give us a shorter path

$d_{ij}^{k+1} = d_{ij}^{k}$

33

Recursive relationship

d_{ij}^{k} = shortest path from vertex i to vertex j using only vertices $\{1, 2, \ldots, k\}$

Two options:
1) Vertex k+1 doesn't give us a shorter path
2) Vertex k+1 does give us a shorter path

$d_{ij}^{k+1} = ?$

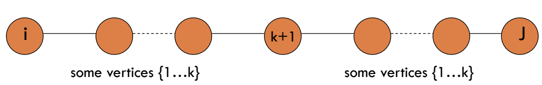
34

Recursive relationship

d_{ij}^{k} = shortest path from vertex i to vertex j using only vertices $\{1, 2, \ldots, k\}$

Two options:
1) Vertex k+1 doesn't give us a shorter path
2) Vertex k+1 does give us a shorter path

$d_{ij}^{k+1} = ?$



What is the cost of this path?

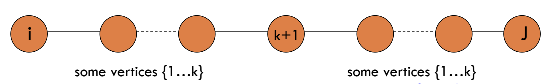
35

Recursive relationship

d_{ij}^{k} = shortest path from vertex i to vertex j using only vertices $\{1, 2, \ldots, k\}$

Two options:
1) Vertex k+1 doesn't give us a shorter path
2) Vertex k+1 does give us a shorter path

$d_{ij}^{k+1} = d_{i(k+1)}^{k} + d_{(k+1)j}^{k}$



$d_{i(k+1)}^{k}$ + $d_{(k+1)j}^{k}$

36

Recursive relationship

d_{ij}^{k} = shortest path from vertex i to vertex j using only vertices $\{1, 2, \ldots, k\}$

Two options:
1) Vertex k+1 doesn't give us a shorter path
2) Vertex k+1 does give us a shorter path

d_{ij}^{k+1} = ?

How do we combine these two options?

37

Recursive relationship

d_{ij}^{k} = shortest path from vertex i to vertex j using only vertices $\{1, 2, \ldots, k\}$

Two options:
1) Vertex k+1 doesn't give us a shorter path
2) Vertex k+1 does give us a shorter path

$$d_{ij}^{k+1} = \min(dijk, d_{i(k+1)}^{k} + d_{(k+1)j}^{k})$$

Pick whichever is shorter

38

Floyd-Warshall

Calculate d_{ij}^{k} for increasing k, i.e. k = 1 to V

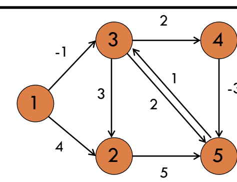
Floyd-Warshall(G = (V,E,W)):
d^0 = W // initialize with edge weights
for k = 1 to V
for i = 1 to V
for j = 1 to V
$\text{dijk} = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$

return d^V

39

Floyd-Warshall(G = (V,E,W)):
d^0 = W // initialize with edge weights
for k = 1 to V
for i = 1 to V
for j = 1 to V
$\text{dijk} = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$

return d^V



k = 0

	1	2	3	4	5
1	0	4	−1	∞	∞
2	∞	0	∞	∞	5
3	∞	3	0	2	2
4	∞	∞	∞	0	−3
5	∞	∞	1	∞	0

adjacency matrix

k = 1

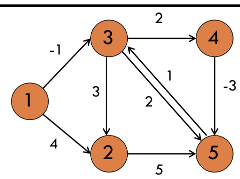
	1	2	3	4	5
1	0	4	−1	∞	∞
2	∞	0	∞	∞	5
3	∞	3	0	2	2
4	∞	∞	∞	0	−3
5	∞	∞	1	∞	0

no change

40

Floyd-Warshall(G = (V,E,W)):
d^0 = W // initialize with edge weights
for k = 1 to V
for i = 1 to V
for j = 1 to V
dijk = $\min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$

return d^V



k = 1

	1	2	3	4	5
1	0	4	−1	∞	∞
2	∞	0	∞	∞	5
3	∞	3	0	2	2
4	∞	∞	∞	0	−3
5	∞	∞	1	∞	0

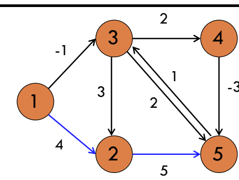
k = 2

	1	2	3	4	5
1	0	4	−1	∞	?
2					
3					
4					
5					

41

Floyd-Warshall(G = (V,E,W)):
d^0 = W // initialize with edge weights
for k = 1 to V
for i = 1 to V
for j = 1 to V
dijk = $\min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$

return d^V



k = 1

	1	2	3	4	5
1	0	4	−1	∞	∞
2	∞	0	∞	∞	5
3	∞	3	0	2	2
4	∞	∞	∞	0	−3
5	∞	∞	1	∞	0

minimum

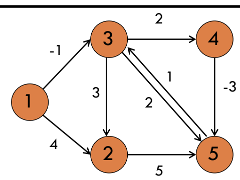
k = 2

	1	2	3	4	5
1	0	4	−1	∞	9
2					
3					
4					
5					

42

Floyd-Warshall(G = (V,E,W)):
d^0 = W // initialize with edge weights
for k = 1 to V
for i = 1 to V
for j = 1 to V
dijk = $\min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$

return d^V



k = 2

	1	2	3	4	5
1	0	4	−1	∞	9
2	∞	0	∞	∞	5
3	∞	3	0	2	2
4	∞	∞	∞	0	−3
5	∞	∞	1	∞	0

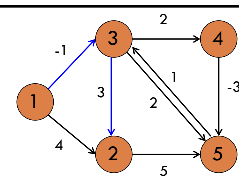
k = 3

	1	2	3	4	5
1	0	?			
2					
3					
4					
5					

43

Floyd-Warshall(G = (V,E,W)):
d^0 = W // initialize with edge weights
for k = 1 to V
for i = 1 to V
for j = 1 to V
dijk = $\min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$

return d^V



k = 2

	1	2	3	4	5
1	0	4	−1	∞	9
2	∞	0	∞	∞	5
3	∞	3	0	2	2
4	∞	∞	∞	0	−3
5	∞	∞	1	∞	0

minimum

k = 3

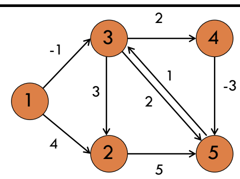
	1	2	3	4	5
1	0	2			
2					
3					
4					
5					

Found a shorter path!

44

Floyd-Warshall(G = (V,E,W)):
$d^0 = W$ // initialize with edge weights
for k = 1 to V
for i = 1 to V
for j = 1 to V
$d_{ijk} = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$

return d^V



k = 2

	1	2	3	4	5
1	0	4	−1	∞	9
2	∞	0	∞	∞	5
3	∞	3	0	2	2
4	∞	∞	∞	0	−3
5	∞	∞	1	∞	0

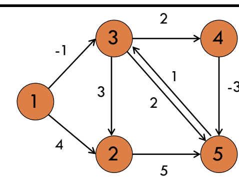
k = 3

	1	2	3	4	5
1	0	2			
2					
3					
4					
5					

45

Floyd-Warshall(G = (V,E,W)):
$d^0 = W$ // initialize with edge weights
for k = 1 to V
for i = 1 to V
for j = 1 to V
$d_{ijk} = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$

return d^V



k = 2

	1	2	3	4	5
1	0	4	−1	∞	9
2	∞	0	∞	∞	5
3	∞	3	0	2	2
4	∞	∞	∞	0	−3
5	∞	∞	1	∞	0

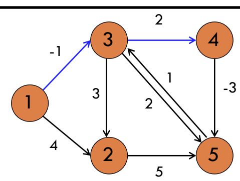
k = 3

	1	2	3	4	5
1	0	2	−1	?	
2					
3					
4					
5					

46

Floyd-Warshall(G = (V,E,W)):
$d^0 = W$ // initialize with edge weights
for k = 1 to V
for i = 1 to V
for j = 1 to V
$d_{ijk} = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$

return d^V



k = 2

	1	2	3	4	5
1	0	4	−1	∞	9
2	∞	0	∞	∞	5
3	∞	3	0	2	2
4	∞	∞	∞	0	−3
5	∞	∞	1	∞	0

minimum

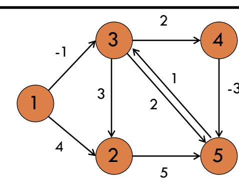
k = 3

	1	2	3	4	5
1	0	2	−1	1	
2					
3					
4					
5					

47

Floyd-Warshall(G = (V,E,W)):
$d^0 = W$ // initialize with edge weights
for k = 1 to V
for i = 1 to V
for j = 1 to V
$d_{ijk} = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$

return d^V



k = 2

	1	2	3	4	5
1	0	4	−1	∞	9
2	∞	0	∞	∞	5
3	∞	3	0	2	2
4	∞	∞	∞	0	−3
5	∞	∞	1	∞	0

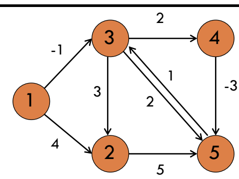
k = 3

	1	2	3	4	5
1	0	2	−1	1	
2					
3					
4					
5					

48

Floyd-Warshall(G = (V,E,W)):
d^0 = W // initialize with edge weights
for k = 1 to V
for i = 1 to V
for j = 1 to V
dijk $= \min(d_{ij}{}^{k-1}, d_{ik}{}^{k-1} + d_{kj}{}^{k-1})$

return d^V



k = 2

	1	2	3	4	5
1	0	4	−1	∞	9
2	∞	0	∞	∞	5
3	∞	3	0	2	2
4	∞	∞	∞	0	−3
5	∞	∞	1	∞	0

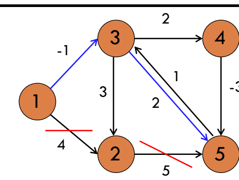
k = 3

	1	2	3	4	5
1	0	2	−1	1	?
2					
3					
4					
5					

49

Floyd-Warshall(G = (V,E,W)):
d^0 = W // initialize with edge weights
for k = 1 to V
for i = 1 to V
for j = 1 to V
dijk $= \min(d_{ij}{}^{k-1}, d_{ik}{}^{k-1} + d_{kj}{}^{k-1})$

return d^V



k = 2

	1	2	3	4	5
1	0	4	−1	∞	9
2	∞	0	∞	∞	5
3	∞	3	0	2	2
4	∞	∞	∞	0	−3
5	∞	∞	1	∞	0

minimum

k = 3

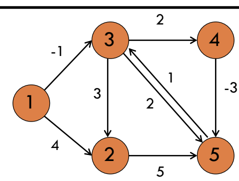
	1	2	3	4	5
1	0	2	−1	1	1
2					
3					
4					
5					

Found a shorter path!

50

Floyd-Warshall(G = (V,E,W)):
d^0 = W // initialize with edge weights
for k = 1 to V
for i = 1 to V
for j = 1 to V
dijk $= \min(d_{ij}{}^{k-1}, d_{ik}{}^{k-1} + d_{kj}{}^{k-1})$

return d^V



k = 2

	1	2	3	4	5
1	0	4	−1	∞	9
2	∞	0	∞	∞	5
3	∞	3	0	2	2
4	∞	∞	∞	0	−3
5	∞	∞	1	∞	0

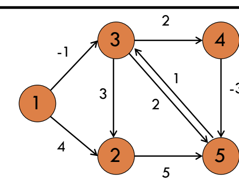
k = 3

	1	2	3	4	5
1	0	2	−1	1	1
2	∞	0	∞	∞	5
3	∞	3	0	2	2
4	∞	∞	∞	0	−3
5	∞	∞	1	∞	0

51

Floyd-Warshall(G = (V,E,W)):
d^0 = W // initialize with edge weights
for k = 1 to V
for i = 1 to V
for j = 1 to V
dijk $= \min(d_{ij}{}^{k-1}, d_{ik}{}^{k-1} + d_{kj}{}^{k-1})$

return d^V



k = 3

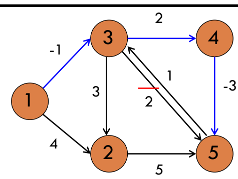
	1	2	3	4	5
1	0	2	−1	1	1
2	∞	0	∞	∞	5
3	∞	3	0	2	2
4	∞	∞	∞	0	−3
5	∞	∞	1	∞	0

k = 4

	1	2	3	4	5
1	0	2	−1	1	?
2					
3					
4					
5					

52

Floyd-Warshall(G = (V,E,W)):
$d^0 = W$ // initialize with edge weights
for k = 1 to V
for i = 1 to V
for j = 1 to V
dijk $= \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$
return d^V



k = 3

	1	2	3	4	5
1	0	2	−1	1	1
2	∞	0	∞	∞	5
3	∞	3	0	2	2
4	∞	∞	∞	0	−3
5	∞	∞	1	∞	0

minimum

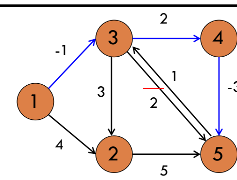
k = 4

	1	2	3	4	5
1	0	2	−1	1	−2
2					
3					
4					
5					

Found a shorter path!

53

Floyd-Warshall(G = (V,E,W)):
$d^0 = W$ // initialize with edge weights
for k = 1 to V
for i = 1 to V
for j = 1 to V
dijk $= \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$
return d^V



k = 3

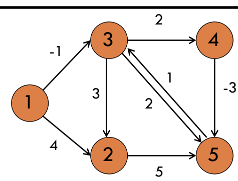
	1	2	3	4	5
1	0	2	−1	1	1
2	∞	0	∞	∞	5
3	∞	3	0	2	2
4	∞	∞	∞	0	−3
5	∞	∞	1	∞	0

k = 4

	1	2	3	4	5
1	0	2	−1	1	−2
2					
3					
4					
5					

54

Floyd-Warshall(G = (V,E,W)):
$d^0 = W$ // initialize with edge weights
for k = 1 to V
for i = 1 to V
for j = 1 to V
dijk $= \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$
return d^V



k = 3

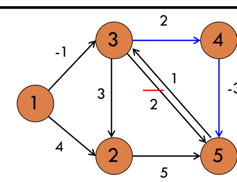
	1	2	3	4	5
1	0	2	−1	1	1
2	∞	0	∞	∞	5
3	∞	3	0	2	2
4	∞	∞	∞	0	−3
5	∞	∞	1	∞	0

k = 4

	1	2	3	4	5
1	0	2	−1	1	−2
2	∞	0	∞	∞	5
3	∞	3	0	2	?
4					
5					

55

Floyd-Warshall(G = (V,E,W)):
$d^0 = W$ // initialize with edge weights
for k = 1 to V
for i = 1 to V
for j = 1 to V
dijk $= \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$
return d^V



k = 3

	1	2	3	4	5
1	0	2	−1	1	1
2	∞	0	∞	∞	5
3	∞	3	0	2	2
4	∞	∞	∞	0	−3
5	∞	∞	1	∞	0

minimum

k = 4

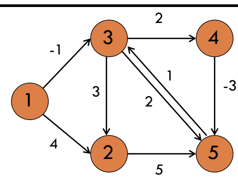
	1	2	3	4	5
1	0	2	−1	1	−2
2	∞	0	∞	∞	5
3	∞	3	0	2	−1
4					
5					

Found a shorter path!

56

Floyd-Warshall(G = (V,E,W)):
d^0 = W // initialize with edge weights
for k = 1 to V
  for i = 1 to V
    for j = 1 to V
      dijk $= \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$

return d^V



k = 3

	1	2	3	4	5
1	0	2	−1	1	1
2	∞	0	∞	∞	5
3	∞	3	0	2	2
4	∞	∞	∞	0	−3
5	∞	∞	1	∞	0

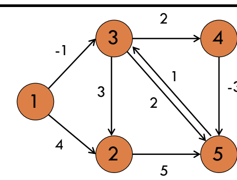
k = 4

	1	2	3	4	5
1	0	2	−1	1	−2
2	∞	0	∞	∞	5
3	∞	3	0	2	−1
4	∞	∞	∞	0	−3
5	∞	∞	1	∞	0

57

Floyd-Warshall(G = (V,E,W)):
d^0 = W // initialize with edge weights
for k = 1 to V
  for i = 1 to V
    for j = 1 to V
      dijk $= \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$

return d^V



k = 4

	1	2	3	4	5
1	0	2	−1	1	−2
2	∞	0	∞	∞	5
3	∞	3	0	2	−1
4	∞	∞	∞	0	−3
5	∞	∞	1	∞	0

k = 5

	1	2	3	4	5
1	0	2	−1	1	−2
2	∞	0	7	9	5
3	∞	3	0	2	−1
4	∞	1	−2	0	−3
5	∞	∞	1	∞	0

Done!

58

Floyd-Warshall analysis

Is it correct?

Floyd-Warshall(G = (V,E,W)):
d^0 = W // initialize with edge weights
for k = 1 to V
  for i = 1 to V
    for j = 1 to V
      dijk $= \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$

return d^V

59

Floyd-Warshall analysis

Is it correct?

Any assumptions?

Floyd-Warshall(G = (V,E,W)):
d^0 = W // initialize with edge weights
for k = 1 to V
  for i = 1 to V
    for j = 1 to V
      dijk $= \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$

return d^V

60

Floyd-Warshall analysis

Is it correct?

Assuming the graph has no negative cycles!

What happens if there is a negative cycle?

```
Floyd-Warshall(G = (V,E,W)):
d⁰ = W        // initialize with edge weights
for k = 1 to V
  for i = 1 to V
    for j = 1 to V
      dijk = min(d_ij^{k−1}, d_ik^{k−1} + d_kj^{k−1})

return d^V
```

61

Floyd-Warshall analysis

If the graph has a negative weight cycle, at the end, at least one of the diagonal entries will be a negative number, i.e., we there's a way to get back to a vertex using all of the vertices that results in a negative weight

	1	2	3	4	5
1	0	2	−1	1	−2
2	∞	0	7	9	5
3	∞	3	0	2	−1
4	∞	1	−2	0	−3
5	∞	∞	1	∞	0

62

Floyd-Warshall analysis

Run-time?

```
Floyd-Warshall(G = (V,E,W)):
d⁰ = W        // initialize with edge weights
for k = 1 to V
  for i = 1 to V
    for j = 1 to V
      dijk = min(d_ij^{k−1}, d_ik^{k−1} + d_kj^{k−1})

return d^V
```

63

Floyd-Warshall analysis

Run-time: $\Theta(V^3)$

```
Floyd-Warshall(G = (V,E,W)):
d⁰ = W        // initialize with edge weights
for k = 1 to V
  for i = 1 to V
    for j = 1 to V
      dijk = min(d_ij^{k−1}, d_ik^{k−1} + d_kj^{k−1})

return d^V
```

64

Floyd-Warshall analysis

Space usage?

```
Floyd-Warshall(G = (V,E,W)):
d^0 = W        // initialize with edge weights
for k = 1 to V
  for i = 1 to V
    for j = 1 to V
      dijk = min(d_ij^{k-1}, d_ik^{k-1} + d_kj^{k-1})

return d^V
```

65

Floyd-Warshall: key idea

Label all vertices with a number from 1 to V

$d_{ij}{}^{k}$ = shortest path from vertex i to vertex j using only vertices $\{1, 2, \ldots, k\}$

If we want all possibilities, how many values are there (i.e. what is the size of $d_{ij}{}^{k}$)?

66

Floyd-Warshall: key idea

Label all vertices with a number from 1 to V

$d_{ij}{}^{k}$ = shortest path from vertex i to vertex j using only vertices $\{1, 2, \ldots, k\}$

V^3

- i: all vertices
- j: all vertices
- k: all vertices

Can we do better?

67

Floyd-Warshall analysis

Space usage: $\Theta(V^2)$

Only need the current value and the previous

```
Floyd-Warshall(G = (V,E,W)):
d^0 = W        // initialize with edge weights
for k = 1 to V
  for i = 1 to V
    for j = 1 to V
      dijk = min(d_ij^{k-1}, d_ik^{k-1} + d_kj^{k-1})

return d^V
```

68

All pairs shortest paths

V * Bellman-Ford: $O(V^2E)$

Floyd-Warshall: $\theta(V^3)$

69

All pairs shortest paths

All pairs shortest paths for positive weight graphs: calculate the shortest paths between *all* points

Easy solution?

70

All pairs shortest paths

All pairs shortest paths for positive weight graphs: calculate the shortest paths between *all* points

Run Dijsktras from each vertex!

Running time (in terms of E and V)?

71

All pairs shortest paths

All pairs shortest paths for positive weight graphs: calculate the shortest paths between *all* points

Run Dijsktras from each vertex!

$O(V^2 \log V + V E)$

- V calls do Dijkstras
- Dijkstras: $O(V \log V + E)$

72

All pairs shortest paths

V * Bellman-Ford: $O(V^2E)$

Floyd-Warshall: $\theta(V^3)$

V * Dijkstras: $O(V^2 \log V + V E)$

Is this any better?

73

All pairs shortest paths

V * Bellman-Ford: $O(V^2E)$

Floyd-Warshall: $\theta(V^3)$

V * Dijkstras: $O(V^2 \log V + V E)$

If the graph is sparse!

74

All pairs shortest paths

All pairs shortest paths for positive weight graphs: calculate the shortest paths between *all* points

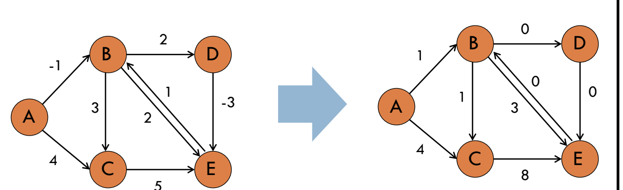
Run Dijsktras from each vertex!

Challenge: Dijkstras assumes positive weights

75

Johnson's: key idea

Reweight the graph to make all edges positive *such that shortest paths are preserved*



76

Lemma

let h be *any* function mapping a vertex to a real value

If we change the graph weights as:

$$\hat{w}(u,v) = w(u,v) + h(u) - h(v)$$

The shortest paths are preserved

77

Lemma: proof $\hat{w}(u,v) = w(u,v) + h(u) - h(v)$

Let $s, v_1, v_2, \ldots, v_k, t$ be a path from s to t

The weight in the reweighted graph is:

$$\hat{w}(s,v_1,\ldots,v_k,t) = w(s,v_1) + h(s) - h(v_1) + \hat{w}(v_1,\ldots,v_k,t)$$

$$= w(s,v_1) + h(s) - h(v_1) + w(v_1,v_2) + h(v_1) - h(v_2) + \hat{w}(v_2,\ldots,v_k,t)$$

$$= w(s,v_1) + h(s) + w(v_1,v_2) - h(v_2) + \hat{w}(v_2,\ldots,v_k,t)$$

$$= w(s,v_1) + h(s) + w(v_1,v_2) - h(v_2) + w(v_2,v_3) + h(v_2) - h(v_3) + \hat{w}(v_3,\ldots,v_k,t)$$

$$= w(s,v_1) + h(s) + w(v_1,v_2) + w(v_2,v_3) - h(v_3) + \hat{w}(v_3,\ldots,v_k,t)$$

$$\ldots$$

$$= w(s,v_1,\ldots,v_k,t) + h(s) - h(t)$$

78

Lemma: proof

$$\hat{w}(s,v_1,\ldots,v_k,t) = w(s,v_1,\ldots,v_k,t) + h(s) - h(t)$$

Claim: the weight change preserves shortest paths, i.e. if a path was the shortest from s to t in the original graph it will still be the shortest path from s to t in the new graph.

Justification?

79

Lemma: proof

$$\hat{w}(s,v_1,\ldots,v_k,t) = w(s,v_1,\ldots,v_k,t) + h(s) - h(t)$$

Claim: the weight change preserves shortest paths, i.e. if a path was the shortest from s to t in the original graph it will still be the shortest path from s to t in the new graph.

h(s) – h(t) is a constant and will be the same for all paths from s to t, so the absolute ordering of all paths from s to t will not change.

80

Lemma

let *h* be *any* function mapping a vertex to a real value

If we change the graph weights as:

$$\hat{w}(u,v) = w(u,v) + h(u) - h(v)$$

The shortest paths are preserved

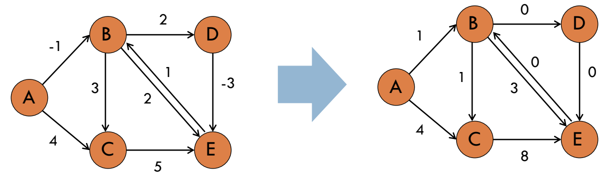
Big question: how do we pick h?

81

Selecting h

Need to pick h such that the resulting graph has all weights as positive

$$\hat{w}(u,v) = w(u,v) + h(u) - h(v)$$



82

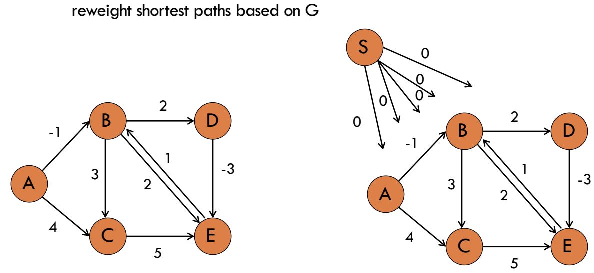
Johnson's algorithm

Create G' with one extra node s with 0 weight edges to all nodes
run Bellman-Ford(G',s)

if no negative-weight cycle
    reweight edges in G with h(v)=shortest path from s to v
    run Dijkstra's from every vertex
    reweight shortest paths based on G

83

Create G'
run Bellman-Ford(G',s)
if no negative-weight cycle
    reweight edges in G with h(v)=shortest path from s to v
    run Dijkstra's from every vertex
    reweight shortest paths based on G



84

Create G'

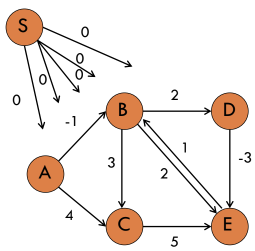
run Bellman-Ford(G',s)

if no negative-weight cycle

reweight edges in G with h(v)=shortest path from s to v

run Dijkstra's from every vertex

reweight shortest paths based on G



S→A: ?
S→B:
S→C:
S→D:
S→E:

85

Create G'

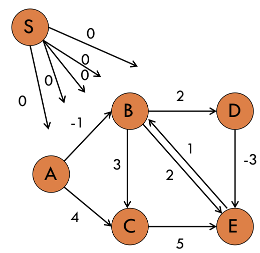
run Bellman-Ford(G',s)

if no negative-weight cycle

reweight edges in G with h(v)=shortest path from s to v

run Dijkstra's from every vertex

reweight shortest paths based on G



S→A: 0
S→B:
S→C:
S→D:
S→E:

86

Create G'

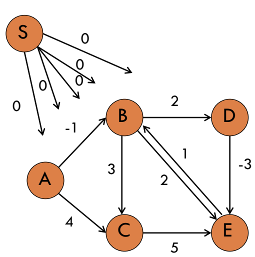
run Bellman-Ford(G',s)

if no negative-weight cycle

reweight edges in G with h(v)=shortest path from s to v

run Dijkstra's from every vertex

reweight shortest paths based on G



S→A: 0
S→B: ?
S→C:
S→D:
S→E:

87

Create G'

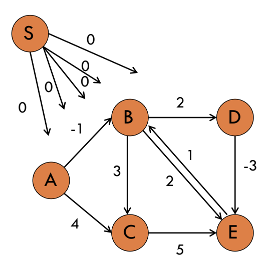
run Bellman-Ford(G',s)

if no negative-weight cycle

reweight edges in G with h(v)=shortest path from s to v

run Dijkstra's from every vertex

reweight shortest paths based on G



S→A: 0
S→B: -2
S→C:
S→D:
S→E:

88

Create G'

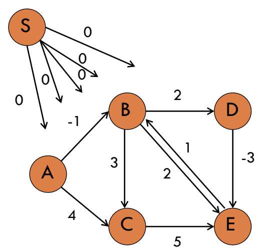
run Bellman-Ford(G',s)

if no negative-weight cycle

reweight edges in G with h(v)=shortest path from s to v

run Dijkstra's from every vertex

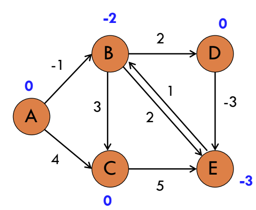
reweight shortest paths based on G



S→A: 0
S→B: -2
S→C: 0
S→D: 0
S→E: -3

89

S→A: 0
S→B: -2
S→C: 0
S→D: 0
S→E: -3



90

Create G'

run Bellman-Ford(G',s)

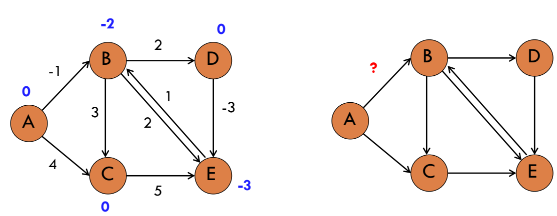
if no negative-weight cycle

reweight edges in G with h(v)=shortest path from s to v

run Dijkstra's from every vertex

reweight shortest paths based on G

$$\hat{w}(u,v) = w(u,v) + h(u) - h(v)$$



91

Create G'

run Bellman-Ford(G',s)

if no negative-weight cycle

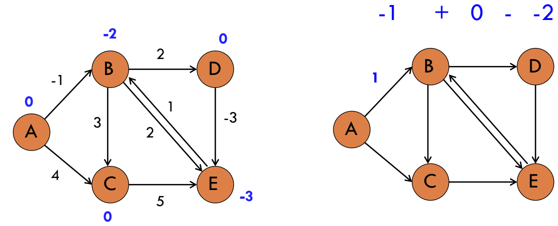
reweight edges in G with h(v)=shortest path from s to v

run Dijkstra's from every vertex

reweight shortest paths based on G

$$\hat{w}(u,v) = w(u,v) + h(u) - h(v)$$

-1 + 0 - -2



92

Create G'

run Bellman-Ford(G',s)

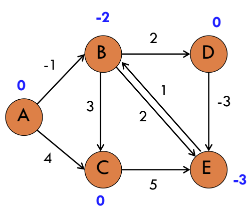
if no negative-weight cycle

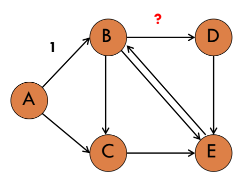
reweight edges in G with h(v)=shortest path from s to v

run Dijkstra's from every vertex

reweight shortest paths based on G

$$\hat{w}(u,v) = w(u,v) + h(u) - h(v)$$





93

Create G'

run Bellman-Ford(G',s)

if no negative-weight cycle

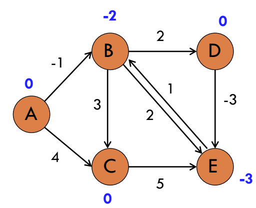
reweight edges in G with h(v)=shortest path from s to v

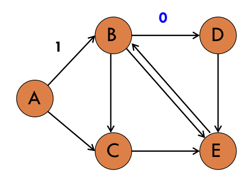
run Dijkstra's from every vertex

reweight shortest paths based on G

$$\hat{w}(u,v) = w(u,v) + h(u) - h(v)$$

2 + -2 - 0





94

Create G'

run Bellman-Ford(G',s)

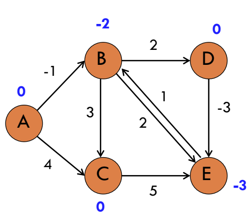
if no negative-weight cycle

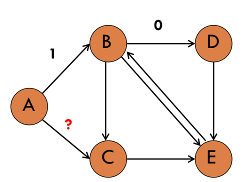
reweight edges in G with h(v)=shortest path from s to v

run Dijkstra's from every vertex

reweight shortest paths based on G

$$\hat{w}(u,v) = w(u,v) + h(u) - h(v)$$





95

Create G'

run Bellman-Ford(G',s)

if no negative-weight cycle

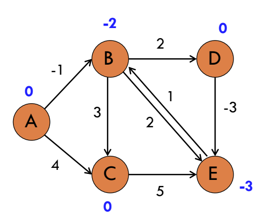
reweight edges in G with h(v)=shortest path from s to v

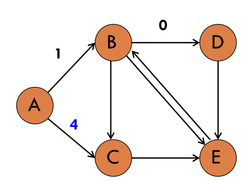
run Dijkstra's from every vertex

reweight shortest paths based on G

$$\hat{w}(u,v) = w(u,v) + h(u) - h(v)$$

4 + 0 - 0





96

Create G'

run Bellman-Ford(G',s)

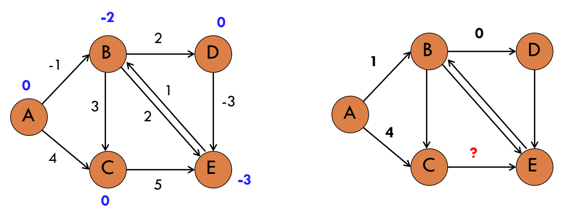
if no negative-weight cycle

reweight edges in G with h(v)=shortest path from s to v

run Dijkstra's from every vertex

reweight shortest paths based on G

$$\hat{w}(u,v) = w(u,v) + h(u) - h(v)$$



97

Create G'

run Bellman-Ford(G',s)

if no negative-weight cycle

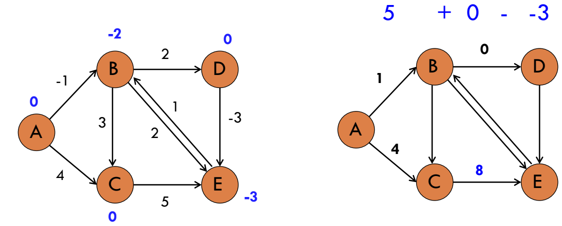
reweight edges in G with h(v)=shortest path from s to v

run Dijkstra's from every vertex

reweight shortest paths based on G

$$\hat{w}(u,v) = w(u,v) + h(u) - h(v)$$

5 + 0 - -3



98

Create G'

run Bellman-Ford(G',s)

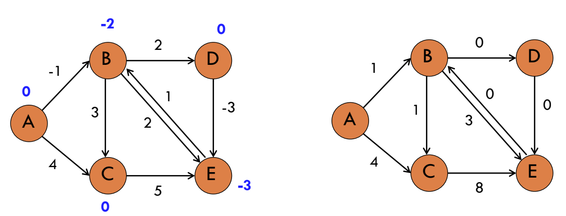
if no negative-weight cycle

reweight edges in G with h(v)=shortest path from s to v

run Dijkstra's from every vertex

reweight shortest paths based on G

$$\hat{w}(u,v) = w(u,v) + h(u) - h(v)$$



99

Create G'

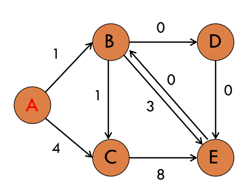
run Bellman-Ford(G',s)

if no negative-weight cycle

reweight edges in G with h(v)=shortest path from s to v

run Dijkstra's from every vertex

reweight shortest paths based on G



100

Create G'

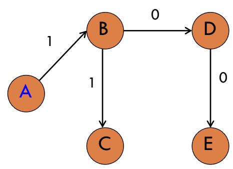
run Bellman-Ford(G',s)

if no negative-weight cycle

  reweight edges in G with h(v)=shortest path from s to v

  run Dijkstra's from every vertex

  reweight shortest paths based on G



101

Create G'

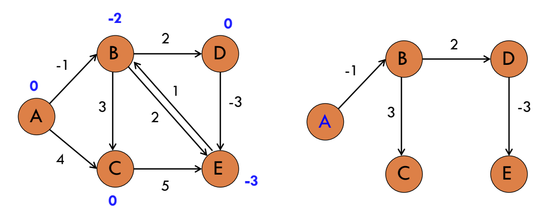
run Bellman-Ford(G',s)

if no negative-weight cycle

  reweight edges in G with h(v)=shortest path from s to v

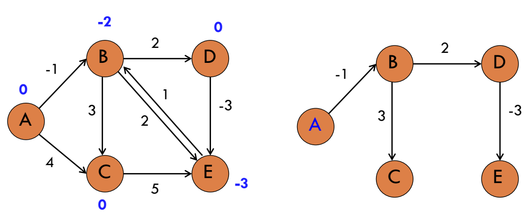
  run Dijkstra's from every vertex

  reweight shortest paths based on G



102

A→B: -1
A→C: 2
A→D: 1
A→E: -2



103

Selecting h

Need to pick h such that the resulting graph has all weights as positive

Create G' with one extra node s with 0 weight edges to all nodes

run Bellman-Ford(G',s)

if no negative-weight cycle

  reweight edges in G with h(v)=shortest path from s to v

  run Dijkstra's from every vertex

  reweight shortest paths based on G

Why does this work (i.e. how do we guarantee that reweighted graph has only positive edges)?

104

Reweighted graph is positive

Take two nodes u and v

h(u) shortest distance from s to u
h(v) shortest distance from s to v

Claim: $h(v) \le h(u) + w(u,v)$

Why?

105

Reweighted graph is positive

Take two nodes u and v

h(u) shortest distance from s to u
h(v) shortest distance from s to v

Claim: $h(v) \le h(u) + w(u,v)$

If this weren't true, we could have made a shorter path s to v using u

… but this is in contradiction with how we defined h(v)

106

Reweighted graph is positive

Take two nodes u and v

h(u) shortest distance from s to u
h(v) shortest distance from s to v

$h(v) \le h(u) + w(u,v)$

$\underbrace{w(u,v) + h(u) - h(v)} \ge 0$

What is this?

107

Reweighted graph is positive

Take two nodes u and v

h(u) shortest distance from s to u
h(v) shortest distance from s to v

$h(v) \le h(u) + w(u,v)$

$\underbrace{w(u,v) + h(u) - h(v)} \ge 0$

$\hat{w}(u,v) = w(u,v) + h(u) - h(v)$

$\hat{w}(u,v) = w(u,v) + h(u) - h(v) \ge 0$ All edge weights in reweighted graph are non-negative

108

Johnson's algorithm

Create G'
run Bellman-Ford(G',s)
if no negative-weight cycle
    reweight edges in G
    run Dijkstra's from every vertex
    reweight shortest paths based on G

Run-time?

109

Johnson's algorithm

Create G' — $\theta(V)$
run Bellman-Ford(G',s) — $O(V^2)$
if no negative-weight cycle
    reweight edges in G — $\theta(E)$
    run Dijkstra's from every vertex — $O(V^2 \log V + VE)$
    reweight shortest paths based on G — $\theta(E)$

Run-time?

110

All pairs shortest paths

V * Bellman-Ford: $O(V^2E)$

Floyd-Warshall: $\theta(V^3)$

Johnson's: $O(V^2 \log V + V E)$

111