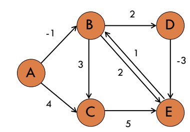
SHORTEST PATHS

Devansh Jalota and David Kauchak
CS 140 – Fall 2024

1

All pairs shortest paths

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



2

Why do we care?

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

Application 1: Calculate diameter in networks, i.e., the longest of all shortest paths (e.g., longest possible transit time between messages in a communication network)

Application 2: Navigational applications (e.g., Google Maps) often need to store APSPs to query in real-time routing

3

All pairs shortest paths

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

Easy solution?

4

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

5

This Class

- Floyd-Warshall Algorithm for APSP
  - For general graphs
- Johnson's Algorithm
  - Improvement in runtime for sparse graphs

6

This Class

- **Floyd-Warshall Algorithm for APSP**
  - **For general graphs**
- Johnson's Algorithm
  - Improvement in runtime for sparse graphs

7

Floyd-Warshall: key idea

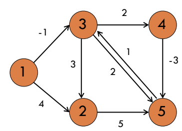
Label all vertices with a number from 1 to V

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

8

Floyd-Warshall: key idea

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

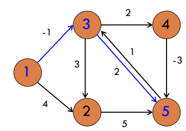


What is d_{15}^{3}?

9

Floyd-Warshall: key idea

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



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

10

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})?

11

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

12

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!

13

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

14

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

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

15

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

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

16

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

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

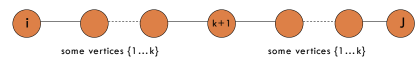
17

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?

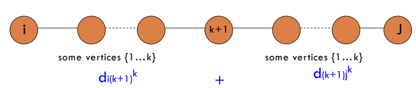
18

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

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



19

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?

20

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

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

Pick whichever is shorter

21

Floyd-Warshall

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

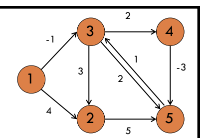
Floyd-Warshall(G = (V,E,W)):
d^0 = W // initialize with edge weights
for k = 1 to V
  for i = 1 to V
    for j = 1 to V

$$d_{ij}{}^{k} = \min(d_{ij}{}^{k-1}, d_{ik}{}^{k-1} + d_{kj}{}^{k-1})$$

return d^V

22

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
      $d_{ij}{}^{k} = \min(d_{ij}{}^{k-1}, d_{ik}{}^{k-1} + d_{kj}{}^{k-1})$
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

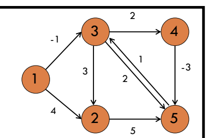
k = 1

	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

23

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
      $d_{ij}{}^{k} = \min(d_{ij}{}^{k-1}, d_{ik}{}^{k-1} + d_{kj}{}^{k-1})$
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					

24

Floyd-Warshall(G = (V,E,W)):

$d^0 = W$ // initialize with edge weights

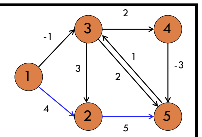
for k = 1 to V

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

$$\begin{bmatrix} 0 & 4 & -1 & \infty & \infty \\ \infty & 0 & \infty & \infty & 5 \\ \infty & 3 & 0 & 2 & 2 \\ \infty & \infty & \infty & 0 & -3 \\ \infty & \infty & 1 & \infty & 0 \end{bmatrix}$$

k = 2

$$\begin{bmatrix} 0 & 4 & -1 & \infty & 9 \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix}$$

minimum

25

Floyd-Warshall(G = (V,E,W)):

$d^0 = W$ // initialize with edge weights

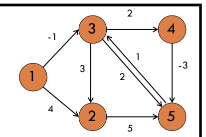
for k = 1 to V

for i = 1 to V

for j = 1 to V

$d_{ij}^{k} = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$

return d^V



k = 2

$$\begin{bmatrix} 0 & 4 & -1 & \infty & 9 \\ \infty & 0 & \infty & \infty & 5 \\ \infty & 3 & 0 & 2 & 2 \\ \infty & \infty & \infty & 0 & -3 \\ \infty & \infty & 1 & \infty & 0 \end{bmatrix}$$

k = 3

$$\begin{bmatrix} 0 & ? & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix}$$

26

Floyd-Warshall(G = (V,E,W)):

$d^0 = W$ // initialize with edge weights

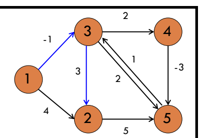
for k = 1 to V

for i = 1 to V

for j = 1 to V

$d_{ij}^{k} = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$

return d^V



k = 2

$$\begin{bmatrix} 0 & 4 & -1 & \infty & 9 \\ \infty & 0 & \infty & \infty & 5 \\ \infty & 3 & 0 & 2 & 2 \\ \infty & \infty & \infty & 0 & -3 \\ \infty & \infty & 1 & \infty & 0 \end{bmatrix}$$

k = 3

$$\begin{bmatrix} 0 & 2 & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix}$$

minimum

Found a shorter path!

27

Floyd-Warshall(G = (V,E,W)):

$d^0 = W$ // initialize with edge weights

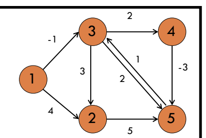
for k = 1 to V

for i = 1 to V

for j = 1 to V

$d_{ij}^{k} = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$

return d^V



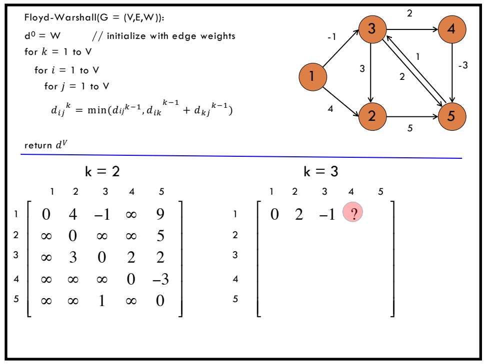
k = 2

$$\begin{bmatrix} 0 & 4 & -1 & \infty & 9 \\ \infty & 0 & \infty & \infty & 5 \\ \infty & 3 & 0 & 2 & 2 \\ \infty & \infty & \infty & 0 & -3 \\ \infty & \infty & 1 & \infty & 0 \end{bmatrix}$$

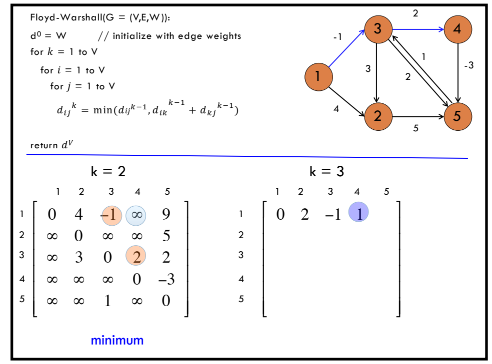
k = 3

$$\begin{bmatrix} 0 & 2 & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix}$$

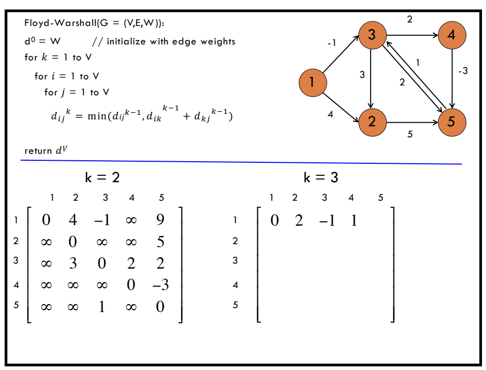
28


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
d_ij^k = 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

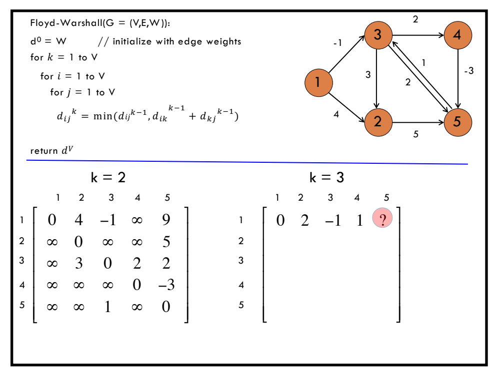
29


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
d_ij^k = 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
minimum

30

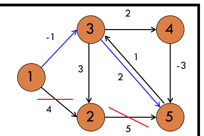

Floyd-Warshall(G = (V,E,W)):
d0 = W // initialize with edge weights
for k = 1 to V
for i = 1 to V
for j = 1 to V
d_ij^k = min(d_ij^(k−1), d_ik^(k−1) + d_kj^(k−1))
return d^V
k = 2
1 2 3 4 5
1 0 4 −1 ∞ 9
2 ∞ 0 ∞ ∞ 5
3 ∞ 3 0 2 2
4 ∞ ∞ ∞ 0 −3
5 ∞ ∞ 1 ∞ 0
k = 3
1 2 3 4 5
1 0 2 −1 1
2
3
4
5

31


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
d_ij^k = 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

32

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
$d_{ij}^{k} = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$
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

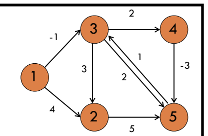
k = 3

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

Found a shorter path!

33

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
$d_{ij}^{k} = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$
return d^V



k = 2

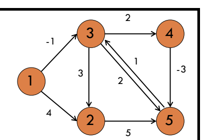
	1	2	3	4	5
1	0	4	−1	∞	9
2	∞	0	∞	∞	5
3	∞	3	0	2	2
4	∞	∞	∞	0	−3
5	∞	∞	1	∞	0

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	∞	?			

34

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
$d_{ij}^{k} = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$
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

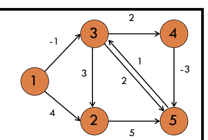
k = 3

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

Found a shorter path!

35

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
$d_{ij}^{k} = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$
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

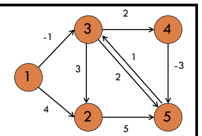
k = 3

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

36

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_ij^k = 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	∞	4	1	3	

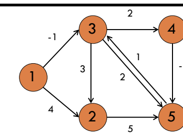
minimum

Found a shorter path!

37

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_ij^k = 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

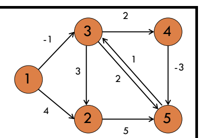
k = 3

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

38

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_ij^k = 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

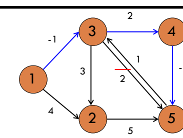
k = 4

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

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
      d_ij^k = 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

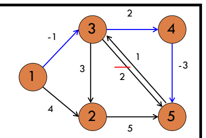
	1	2	3	4	5
1	0	2	−1	1	−2
2					
3					
4					
5					

minimum

Found a shorter path!

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
   $d_{ij}^{k} = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$
return d^V



k = 3

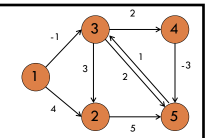
	1	2	3	4	5
1	0	2	−1	1	1
2	∞	0	∞	∞	5
3	∞	3	0	2	2
4	∞	∞	∞	0	−3
5	∞	∞	1	∞	0

k = 4

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

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
   $d_{ij}^{k} = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$
return d^V



k = 3

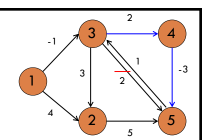
	1	2	3	4	5
1	0	2	−1	1	1
2	∞	0	∞	∞	5
3	∞	3	0	2	2
4	∞	∞	∞	0	−3
5	∞	∞	1	∞	0

k = 4

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

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
   $d_{ij}^{k} = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$
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

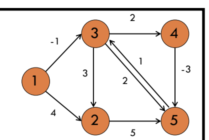
k = 4

	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!

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
   $d_{ij}^{k} = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$
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

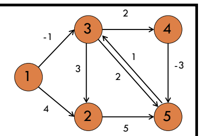
k = 4

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

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
$d_{ij}^{k} = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$

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	6	8	5
3	∞	3	0	2	−1
4	∞	1	−2	0	−3
5	∞	4	1	3	0

Done!

45

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
$d_{ij}^{k} = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$

return d^V

46

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
$d_{ij}^{k} = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$

return d^V

47

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

return d^V

48

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., 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

49

Floyd-Warshall analysis

Run-time?

```
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_ij^k = min(d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1))

return d^V
```

50

Floyd-Warshall analysis

Run-time: θ(V^3)

```
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_ij^k = min(d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1))

return d^V
```

51

Floyd-Warshall analysis

What type of algorithm is Floyd-Warshall?

```
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_ij^k = min(d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1))

return d^V
```

52

Floyd-Warshall analysis

Dynamic programming!!
Build up solutions to larger problems using solutions to smaller problems. Use a table to store the values.

```
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_ij^k = min(d_ij^{k-1}, d_ik^{k-1} + d_kj^{k-1})

return d^V
```

53

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
      d_ij^k = min(d_ij^{k-1}, d_ik^{k-1} + d_kj^{k-1})

return d^V
```

54

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)?

55

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?

56

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
      d_ij^k = min(d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1))

return d^V
```

57

All pairs shortest paths

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

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

58

All pairs shortest paths

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

Easy solution?

59

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)?

60

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

61

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?

62

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!

63

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

64

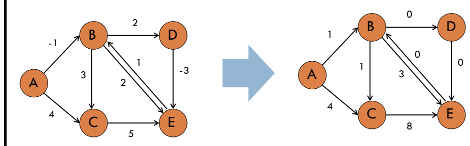
This Class

- Floyd-Warshall Algorithm for APSP
  - For general graphs
- **Johnson's Algorithm**
  - **Improvement in runtime for sparse graphs**

65

Johnson's: key idea

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



What's the shortest path from A to D?

66

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

67

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:

$$\hat{w}(s,v_1,\ldots,v_k,t) = \underbrace{w(s,v_1)+h(s)-h(v_1)}_{\text{weight for first edge}} + \underbrace{\hat{w}(v_1,\ldots,v_k,t)}_{\text{weight for remaining edges}}$$

68

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:

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

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

weight for first edge; weight for second edge; weight for remaining edges

69

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:

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

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

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

70

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:

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

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

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

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

weight for third edge; weight for remaining edges

71

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:

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

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

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

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

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

$$\ldots$$

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

72

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?

73

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.

74

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

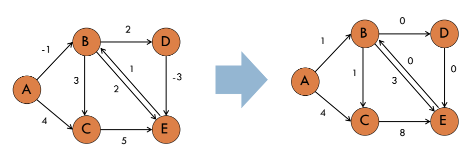
Big question: how do we pick h?

75

Selecting h

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

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



76

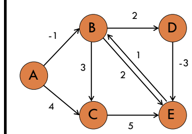
Johnson's algorithm

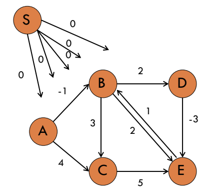
Create G' with one extra node s with 0 weight edges to all nodes
run Bellman-Ford(G',s)

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

77

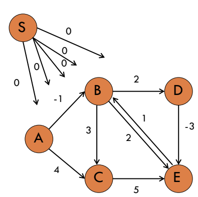
Create G'
run Bellman-Ford(G',s)
if no negative-weight cycle
 reweight edges in G with h(v)=shortest path from s to v
 run Dijkstra's from every vertex
 reweight shortest paths based on G





78

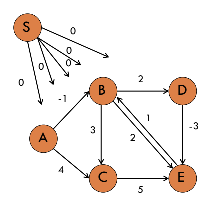
Create G'
run Bellman-Ford(G',s)
if no negative-weight cycle
 reweight edges in G with h(v)=shortest path from s to v
 run Dijkstra's from every vertex
 reweight shortest paths based on G



S➔A: ?
S➔B:
S➔C:
S➔D:
S➔E:

79

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:

80

Create G'

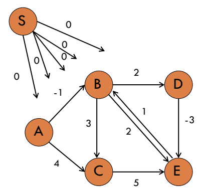
run Bellman-Ford(G',s)

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:

81

Create G'

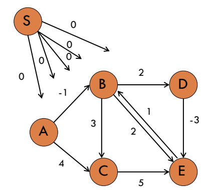
run Bellman-Ford(G',s)

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:

82

Create G'

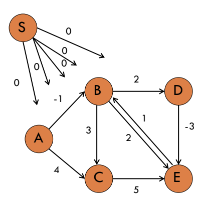
run Bellman-Ford(G',s)

if no negative-weight cycle

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

run Dijkstra's from every vertex

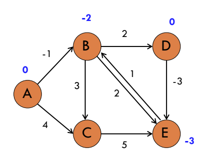
reweight shortest paths based on G



S➔A: 0
S➔B: -2
S➔C: 0
S➔D: 0
S➔E: -3

83

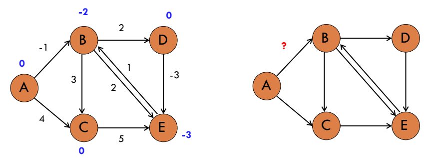
S➔A: 0
S➔B: -2
S➔C: 0
S➔D: 0
S➔E: -3



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

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



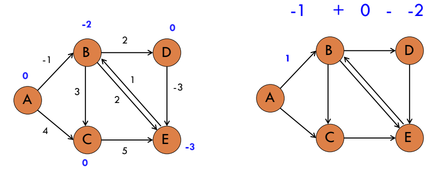
h(v) in blue

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

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

-1 + 0 - -2

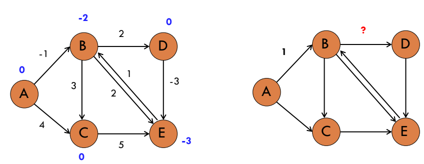


h(v) in blue

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

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



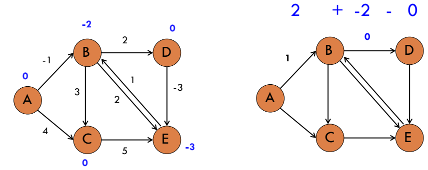
h(v) in blue

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

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

2 + -2 - 0

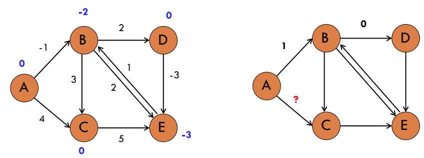


h(v) in blue

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

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



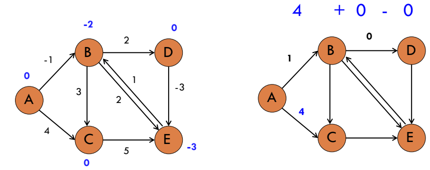
h(v) in blue

89

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

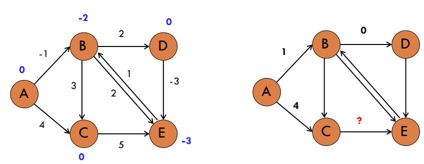


h(v) in blue

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

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



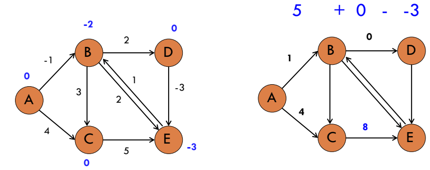
h(v) in blue

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

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

5 + 0 - -3

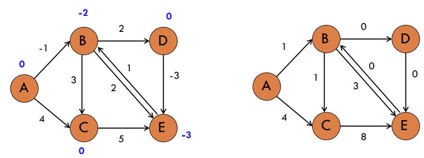


h(v) in blue

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

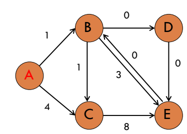
$$\hat{w}(u,v) = w(u,v) + h(u) - h(v)$$



h(v) in blue

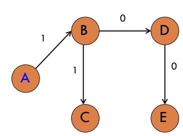
93

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



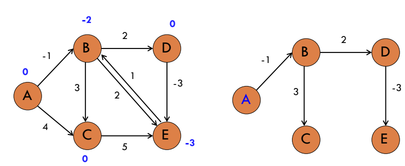
94

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



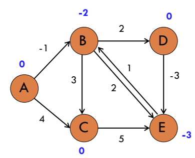
95

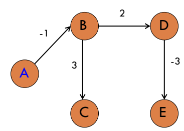
Create G'
run Bellman-Ford(G',s)
if no negative-weight cycle
reweight edges in G with h(v)=shortest path from s to v
run Dijkstra's from every vertex
reweight shortest paths based on G



96

A→B: -1
A→C: 2
A→D: 1
A→E: -2





97

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)?

98

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?

99

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)

100

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

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

$\underbrace{w(u,v) + h(u) - h(v)}_{\text{What is this?}} \ge 0$

101

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

102

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?

103

Johnson's algorithm

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

Run-time?

104

All pairs shortest paths

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

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

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

105