NPTEL MOOC,JAN-FEB 2015
Week 4, Module 4

DESIGN AND ANALYSIS OF ALGORITHMS

All-pairs shortest paths

MADHAVAN MUKUND, CHENNAI MATHEMATICAL INSTITUTE
http://www.cmi.ac.in/~madhavan

Weighted graphs

- Negative weights are allowed, but not negative cycles
- Shortest paths are still well defined
- Bellman-Ford algorithm computes single-source shortest paths
- Can we compute shortest paths between all pairs of vertices?

About shortest paths

- Shortest paths will never loop
  - Never visit the same vertex twice
  - At most length n-1
- Use this to inductively explore all possible shortest paths efficiently

Inductively exploring shortest paths

- Simplest shortest path from i to j is a direct edge (i,j)
- General case:

$$i \to v_1 \to v_2 \to v_3 \ldots \to v_m \to j$$

  - All of $\{v_1, v_2, v_3 \ldots, v_m\}$ are distinct, and different from i and j
  - Restrict what vertices can appear in this set

Inductively exploring shortest paths …

- Recall that V = {1,2,…,n}
- $W^k(i,j)$: weight of shortest path from i to j among paths that only go via {1,2,…,k}
  - {k+1,…,n} cannot appear on the path
  - i, j themselves need not be in {1,2,…,k}
- $W^0(i,j)$: direct edges
  - {1,2,…,n} cannot appear between i and j

Inductively exploring shortest paths …

- From $W^{k-1}(i,j)$ to $W^{k}(i,j)$
  - Case 1: Shortest path via {1,2,…,k} does not use vertex k
    - $W^{k}(i,j) = W^{k-1}(i,j)$
  - Case 2: Shortest path via {1,2,…,k} does go via k
    - k can appear only once along this path
    - Break up as paths i to k and k to j, each via {1,2,…,k-1}
    - $W^{k}(i,j) = W^{k-1}(i,k) + W^{k-1}(k,j)$
- Conclusion: $W^{k}(i,j) = \min(W^{k-1}(i,j)\ ,\ W^{k-1}(i,k) + W^{k-1}(k,j))$

Floyd-Warshall algorithm

- W^0 is adjacency matrix with edge weights
  - $W^0[i][j]$ = weight(i,j) if there is an edge (i,j),
    ∞, otherwise
- For k in 1,2,…,n
  - Compute $W^k(i,j)$ from $W^{k-1}(i,j)$ using
    $W^k(i,j) = \min(W^{k-1}(i,j)\ ,\ W^{k-1}(i,k) + W^{k-1}(k,j))$
- W^n contains weights of shortest paths for all pairs

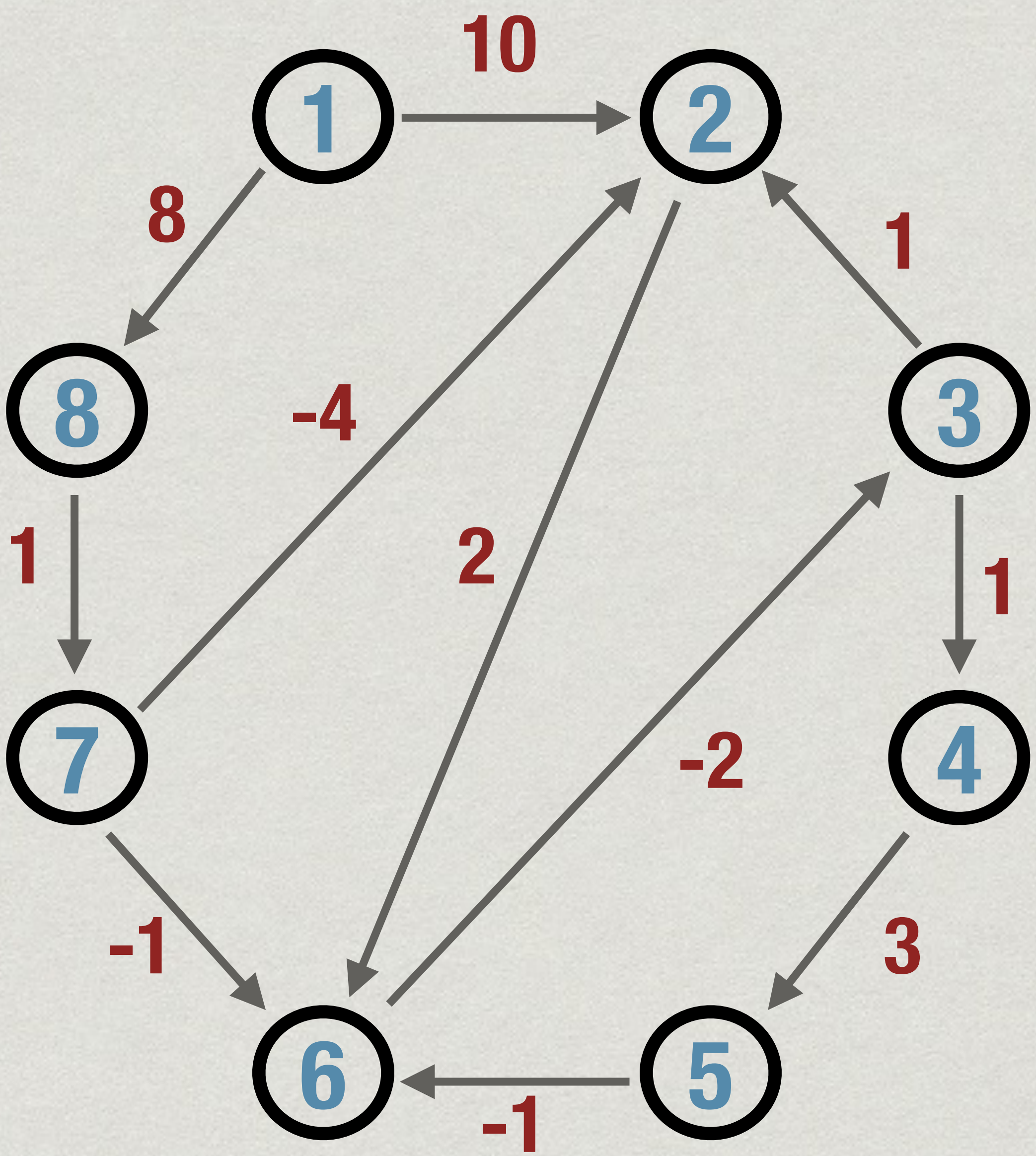
Floyd-Warshall algorithm

- W^0 is adjacency matrix with edge weights
  - $W^0[i][j]$ = weight(i,j) if there is an edge (i,j),
    ∞, otherwise
- For k in 1,2,…,n
  - Compute $W^k(i,j)$ from $W^{k-1}(i,j)$ using
    $W^k(i,j) = \min(W^{k-1}(i,j) , W^{k-1}(i,k) + W^{k-1}(k,j))$
- W^n contains weights of shortest paths for all pairs

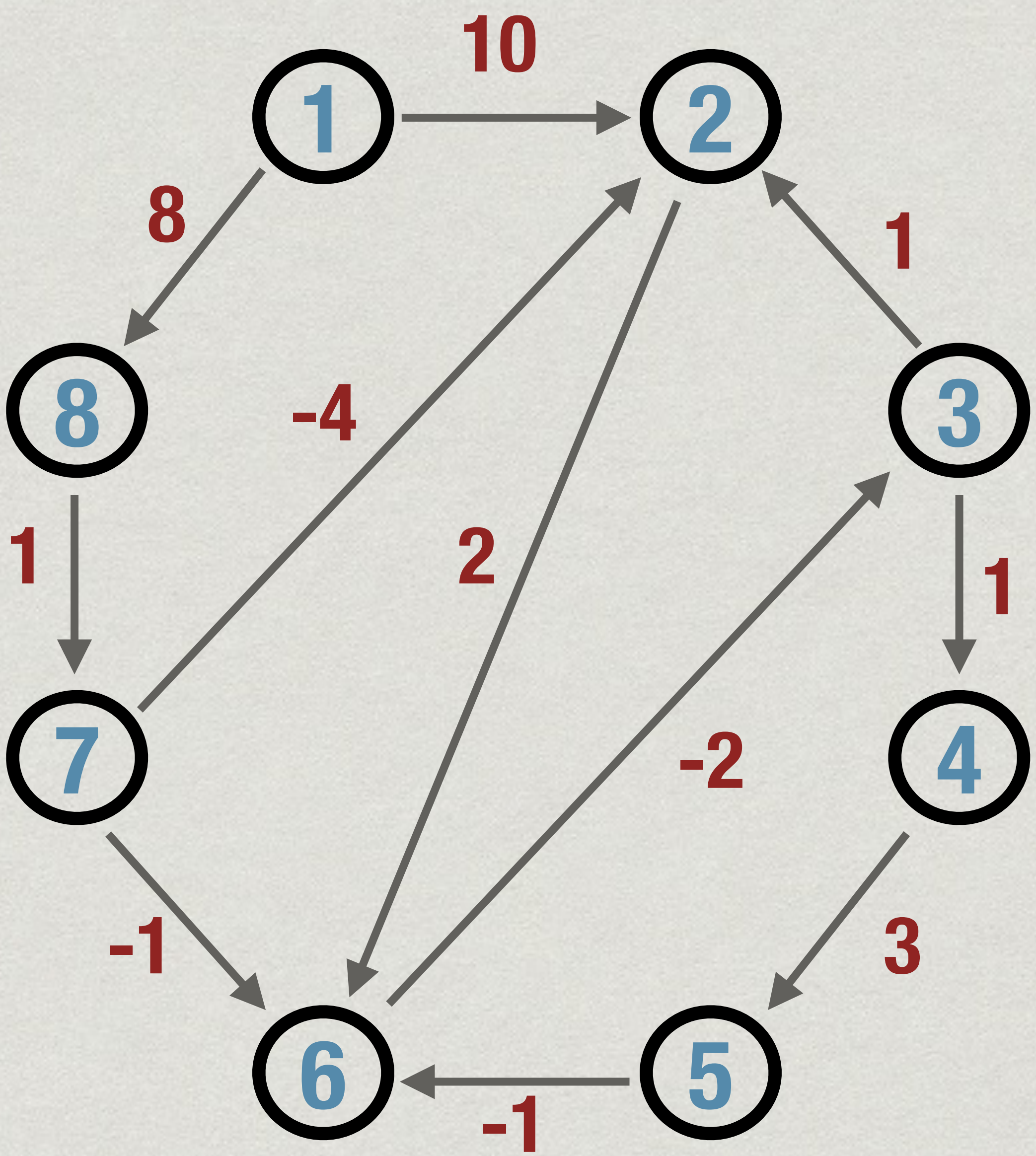
Floyd-Warshall algorithm

```
function FloydWarshall

for i = 1 to n
 for j = 1 to n
   W[i][j][0] = infinity

for each edge (i,j) in E
  W[i][j][0] = weight(i,j)

for k = 1 to n
  for i = 1 to n
    for j = 1 to n
      W[i][j][k] = min(W[i][j][k-1],
                        W[i][k][k-1] + W[k][j][k-1])
```

Example



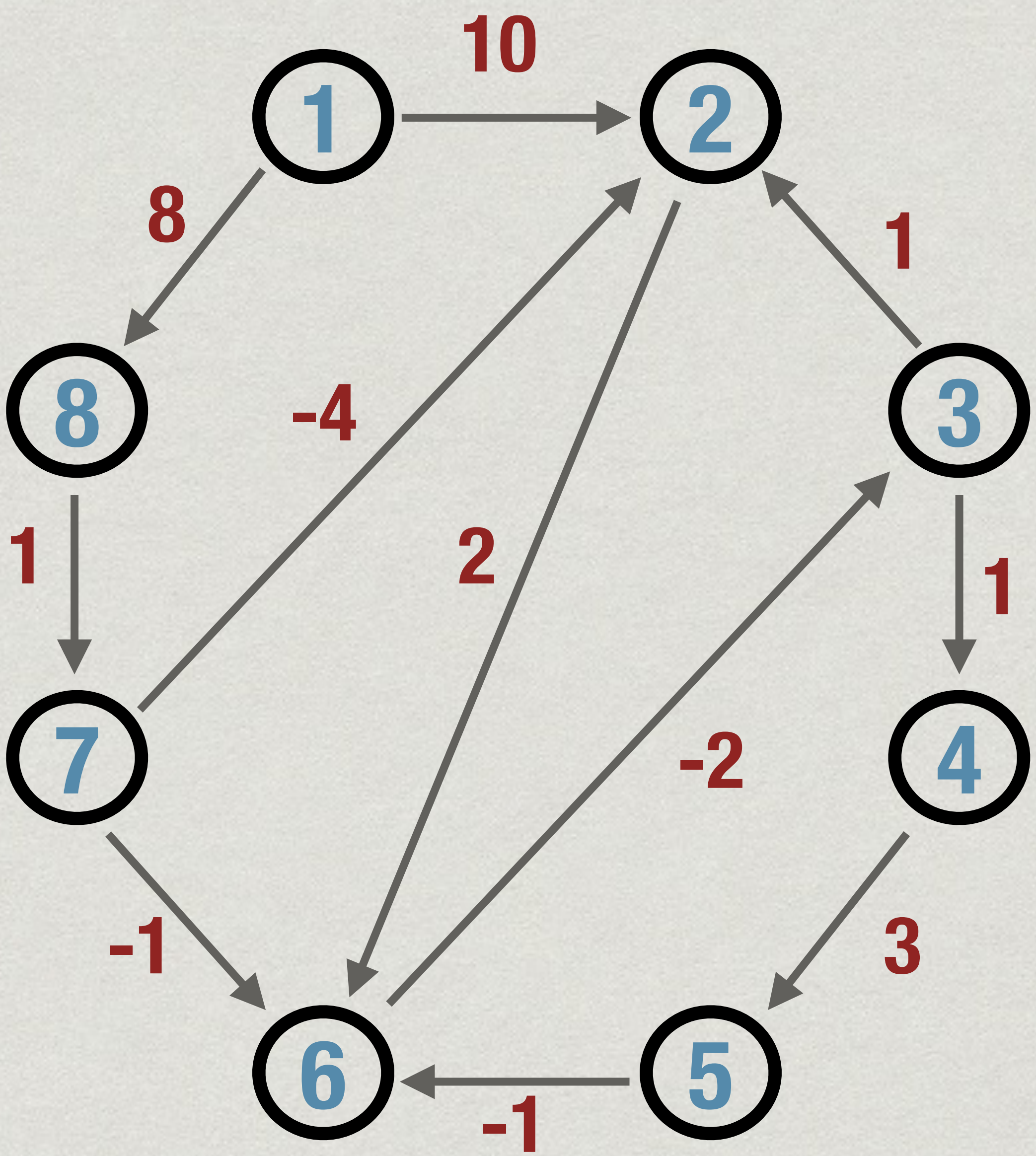
Example



W^0

	1	2	3	4	5	6	7	8
1	∞	10	∞	∞	∞	∞	∞	8
2	∞	∞	∞	∞	∞	2	∞	∞
3	∞	1	∞	1	∞	∞	∞	∞
4	∞	∞	∞	∞	3	∞	∞	∞
5	∞	∞	∞	∞	∞	-1	∞	∞
6	∞	∞	-2	∞	∞	∞	∞	∞
7	∞	-4	∞	∞	∞	-1	∞	∞
8	∞	∞	∞	∞	∞	∞	1	∞

Example



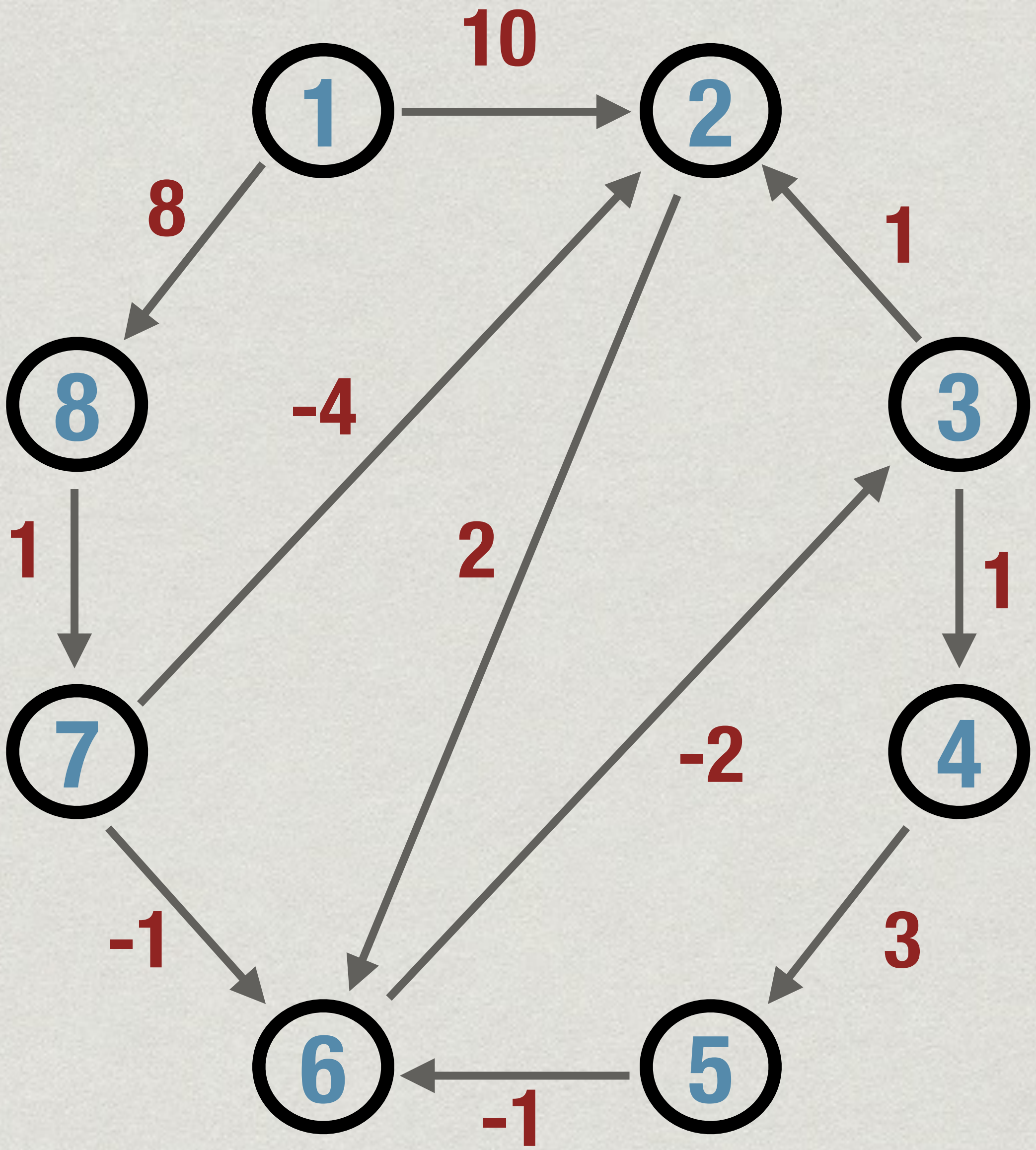
W^0

	1	2	3	4	5	6	7	8
1	∞	10	∞	∞	∞	∞	∞	8
2	∞	∞	∞	∞	∞	2	∞	∞
3	∞	1	∞	1	∞	∞	∞	∞
4	∞	∞	∞	∞	3	∞	∞	∞
5	∞	∞	∞	∞	∞	-1	∞	∞
6	∞	∞	-2	∞	∞	∞	∞	∞
7	∞	-4	∞	∞	∞	-1	∞	∞
8	∞	∞	∞	∞	∞	∞	1	∞

W^1

	1	2	3	4	5	6	7	8
1	∞	10	∞	∞	∞	∞	∞	8
2	∞	∞	∞	∞	∞	2	∞	∞
3	∞	1	∞	1	∞	∞	∞	∞
4	∞	∞	∞	∞	3	∞	∞	∞
5	∞	∞	∞	∞	∞	-1	∞	∞
6	∞	∞	-2	∞	∞	∞	∞	∞
7	∞	-4	∞	∞	∞	-1	∞	∞
8	∞	∞	∞	∞	∞	∞	1	∞

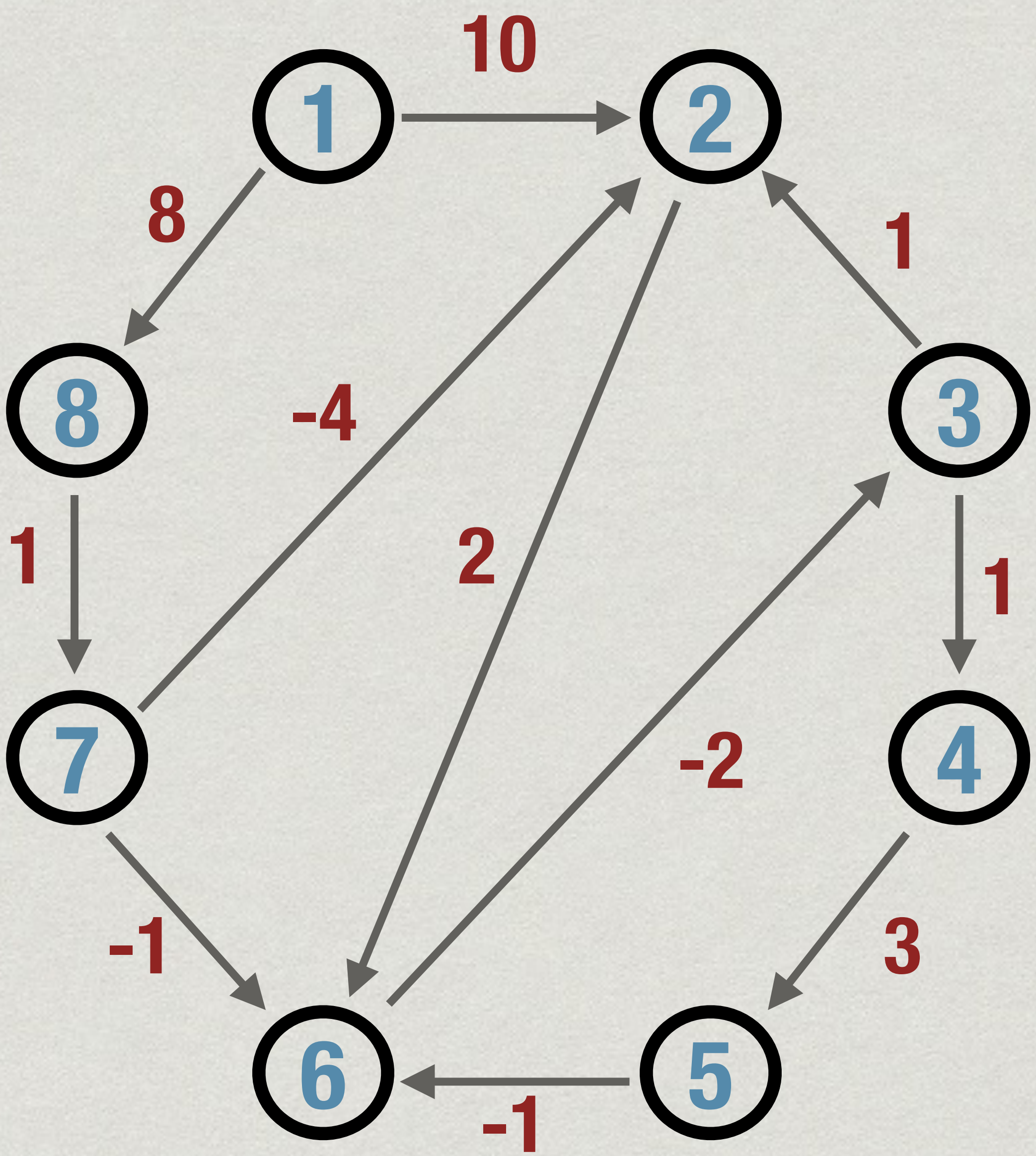
Example



W^1

	1	2	3	4	5	6	7	8
1	∞	10	∞	∞	∞	∞	∞	8
2	∞	∞	∞	∞	∞	2	∞	∞
3	∞	1	∞	1	∞	∞	∞	∞
4	∞	∞	∞	∞	3	∞	∞	∞
5	∞	∞	∞	∞	∞	-1	∞	∞
6	∞	∞	-2	∞	∞	∞	∞	∞
7	∞	-4	∞	∞	∞	-1	∞	∞
8	∞	∞	∞	∞	∞	∞	1	∞

Example



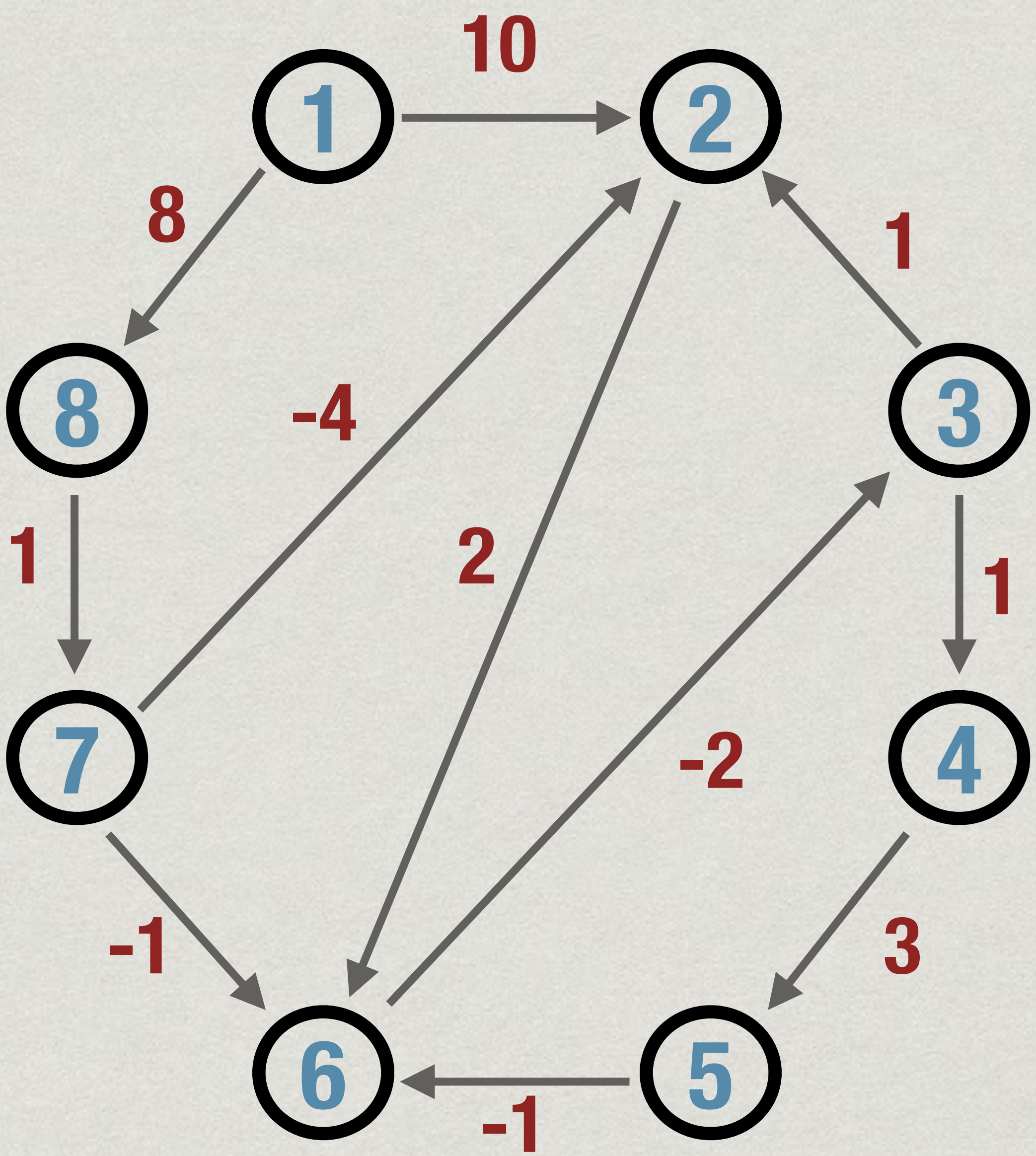
W²

	1	2	3	4	5	6	7	8
1	∞	10	∞	∞	∞	12	∞	8
2	∞	∞	∞	∞	∞	2	∞	∞
3	∞	1	∞	1	∞	3	∞	∞
4	∞	∞	∞	∞	3	∞	∞	∞
5	∞	∞	∞	∞	∞	-1	∞	∞
6	∞	∞	-2	∞	∞	∞	∞	∞
7	∞	-4	∞	∞	∞	-2	∞	∞
8	∞	∞	∞	∞	∞	∞	1	∞

W¹

	1	2	3	4	5	6	7	8
1	∞	10	∞	∞	∞	∞	∞	8
2	∞	∞	∞	∞	∞	2	∞	∞
3	∞	1	∞	1	∞	∞	∞	∞
4	∞	∞	∞	∞	3	∞	∞	∞
5	∞	∞	∞	∞	∞	-1	∞	∞
6	∞	∞	-2	∞	∞	∞	∞	∞
7	∞	-4	∞	∞	∞	-1	∞	∞
8	∞	∞	∞	∞	∞	∞	1	∞

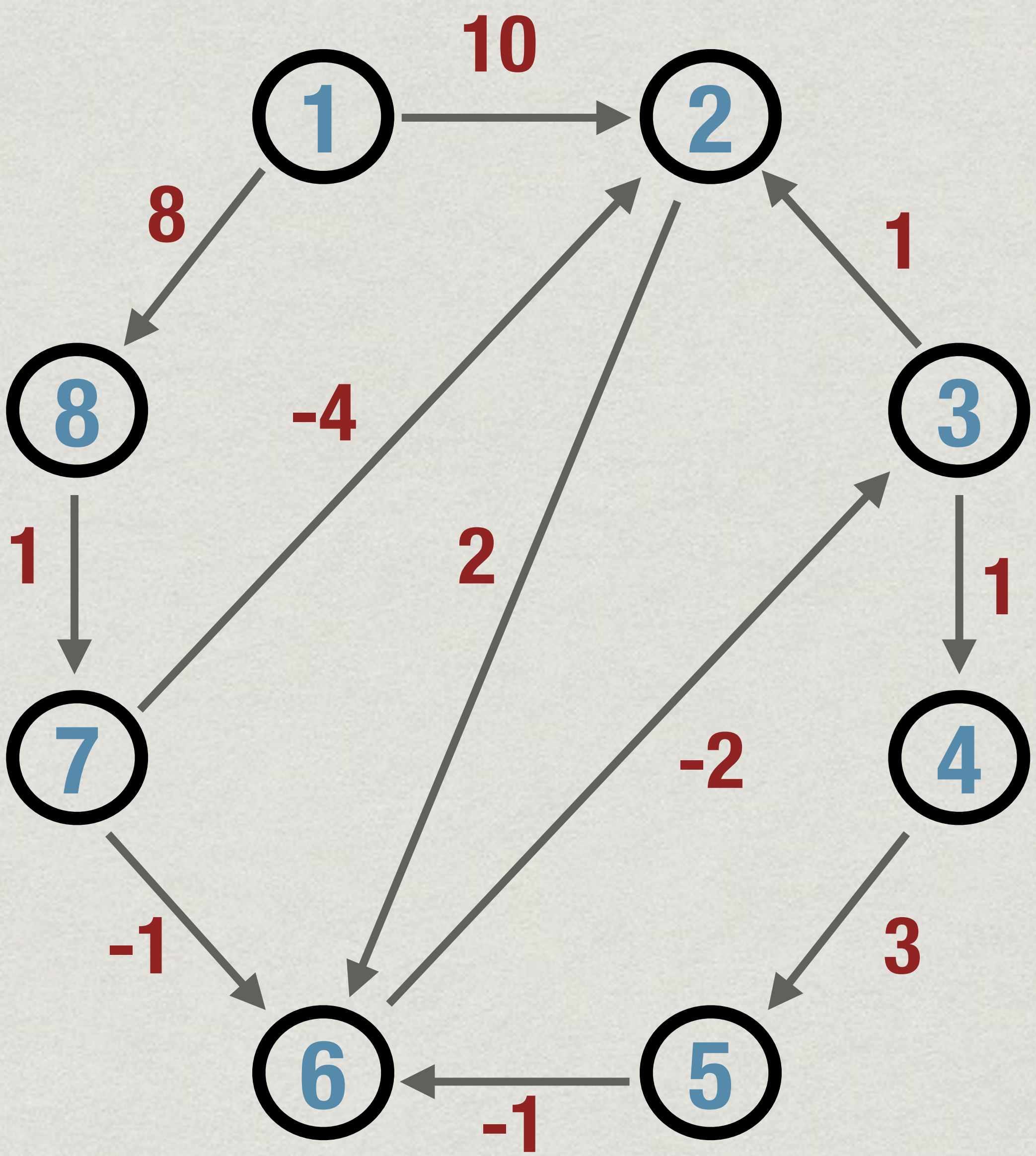
Example



W^2

	1	2	3	4	5	6	7	8
1	∞	10	∞	∞	∞	12	∞	8
2	∞	∞	∞	∞	∞	2	∞	∞
3	∞	1	∞	1	∞	3	∞	∞
4	∞	∞	∞	∞	3	∞	∞	∞
5	∞	∞	∞	∞	∞	-1	∞	∞
6	∞	∞	-2	∞	∞	∞	∞	∞
7	∞	-4	∞	∞	∞	-2	∞	∞
8	∞	∞	∞	∞	∞	∞	1	∞

Example



W^2

	1	2	3	4	5	6	7	8
1	∞	10	∞	∞	∞	12	∞	8
2	∞	∞	∞	∞	∞	2	∞	∞
3	∞	1	∞	1	∞	3	∞	∞
4	∞	∞	∞	∞	3	∞	∞	∞
5	∞	∞	∞	∞	∞	-1	∞	∞
6	∞	∞	-2	∞	∞	∞	∞	∞
7	∞	-4	∞	∞	∞	-2	∞	∞
8	∞	∞	∞	∞	∞	∞	1	∞

W^3

	1	2	3	4	5	6	7	8
1	∞	10	∞	∞	∞	12	∞	8
2	∞	∞	∞	∞	∞	2	∞	∞
3	∞	1	∞	1	∞	3	∞	∞
4	∞	∞	∞	∞	3	∞	∞	∞
5	∞	∞	∞	∞	∞	-1	∞	∞
6	∞	-1	-2	-1	∞	1	∞	∞
7	∞	-4	∞	∞	∞	-2	∞	∞
8	∞	∞	∞	∞	∞	∞	1	∞

Complexity

- Easy to see that the complexity is $O(n^3)$
  - n iterations
  - In each iteration, we update n^2 entries
- A word about space complexity
  - Naive implementation is $O(n^3)$—W[i][j][k]
  - Only need two “slices” at a time, W[i][j][k-1] and W[i][j][k]
  - Space requirement reduces to $O(n^2)$

Historical remarks

- Floyd-Warshall is a hybrid name
- Warshall originally proposed an algorithm for transitive closure
  - Generating path matrix P[i][j] from adjacency matrix A[i][j]
- Floyd adapted it to compute shortest paths

Computing paths

- A(i,j) = 1 iff there is an edge from i to j
- Want P(i,j) = 1 iff there is a path from i to j
- Iteratively compute $P^k(i,j)$ = 1 iff there is a path from i to j where all intermediate vertices are in {1,2,…,k}
  - {k+1,…,n} cannot appear on the path
  - i, j themselves need not be in {1,2,…,k}
- $P^0(i,j)$ = A(i,j): direct edges
  - {1,2,…,n} cannot appear between i and j

Inductively computing P[i][j]

- From $P^{k-1}(i,j)$ to $P^{k}(i,j)$
  - Case 1: There is a path from i to j without using vertex k
    - $P^{k}(i,j) = P^{k-1}(i,j)$
  - Case 2: Path via {1,2,…,k} does go via k
    - k can appear only once along this path
    - Break up as paths i to k and k to j, each via {1,2,…,k-1}
    - $P^{k}(i,j) = P^{k-1}(i,k)$ and $P^{k-1}(k,j)$
- Conclusion: $P^{k}(i,j) = P^{k-1}(i,j)$ or $(P^{k-1}(i,k)$ and $P^{k-1}(k,j))$

Warshall’s algorithm

```
function Warshall

for i = 1 to n
 for j = 1 to n
   P[i][j][0] = False

for each edge (i,j) in E
  P[i][j][0] = True

for k = 1 to n
  for i = 1 to n
    for j = 1 to n
      P[i][j][k] = P[i][j][k-1] or
                   (P[i][k][k-1] and P[k][j][k-1])
```