CS380
Artificial Intelligence for Games

Floyd-Warshall's Search

Motivation

- $O(n^3)$ algorithm for **all pairs** shortest path problem.
- Works with graphs with negative edge lengths but without negative cycles.
- At least as good as n Bellman-Fords, better in dense graphs.
- In graphs with nonnegative edge costs, better than n Dijkstra's in dense graphs.

Floyd-Warshall's Search

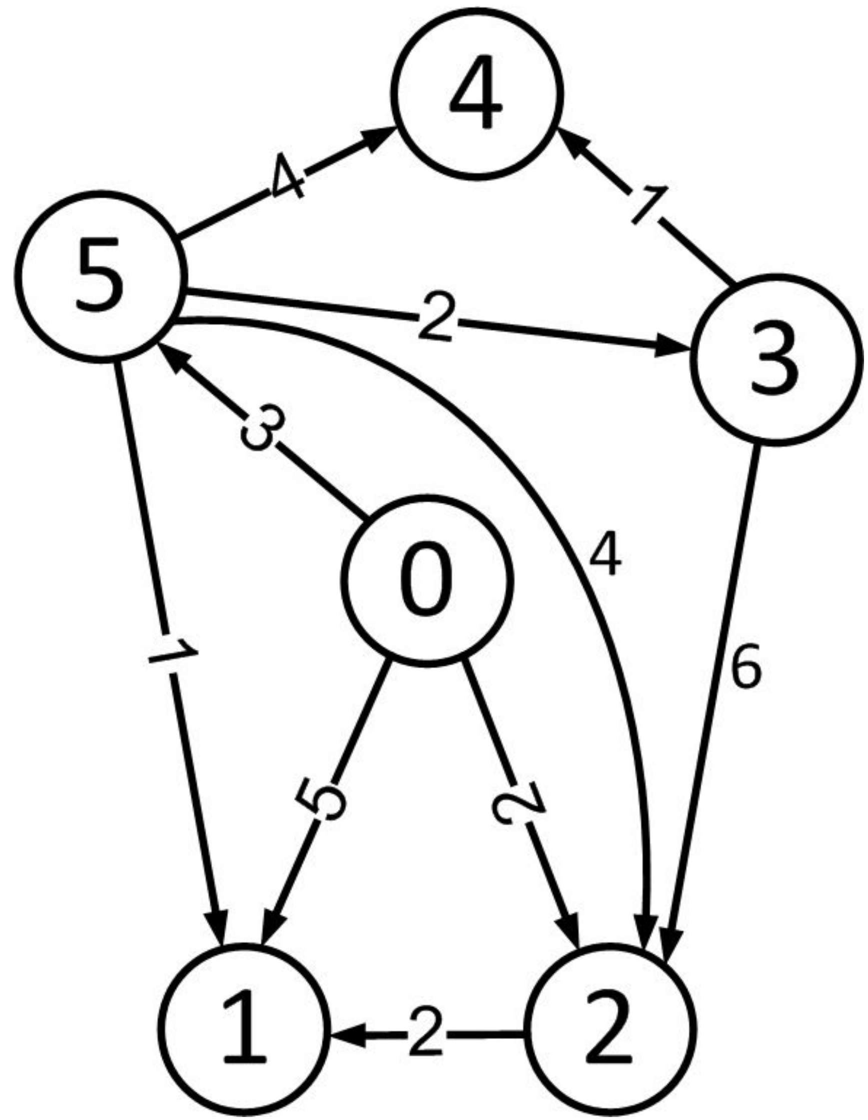
```
let dist be a |V| × |V| array of minimum distances initialized to ∞ (infinity)
let next be a |V| × |V| array of vertex indices initialized to null

procedure FloydWarshallWithPathReconstruction() is
    for each edge (u, v) do
        dist[u][v] ← w(u, v)  // The weight of the edge (u, v)
        next[u][v] ← v
    for each vertex v do
        dist[v][v] ← 0
        next[v][v] ← v
    for k from 1 to |V| do // standard Floyd-Warshall implementation
        for i from 1 to |V|
            for j from 1 to |V|
                if dist[i][j] > dist[i][k] + dist[k][j] then
                    dist[i][j] ← dist[i][k] + dist[k][j]
                    next[i][j] ← next[i][k]
```

Floyd-Warshall's Search

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

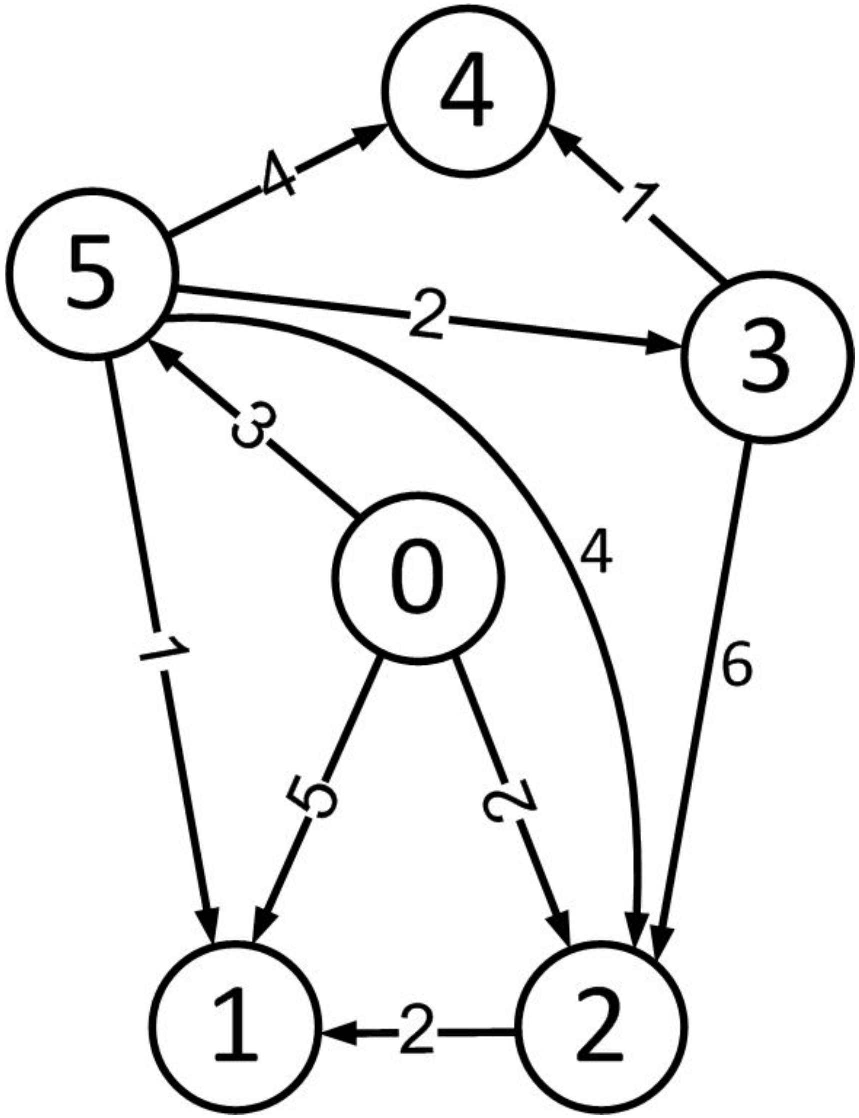
next	0	1	2	3	4	5
0						
1						
2						
3						
4						
5						



Floyd-Warshall's Search

dist	0	1	2	3	4	5
0	0	5	2	+∞	+∞	3
1	+∞	0	+∞	+∞	+∞	+∞
2	+∞	2	0	+∞	+∞	+∞
3	+∞	+∞	6	0	1	+∞
4	+∞	+∞	+∞	+∞	0	+∞
5	+∞	1	4	2	4	0

next	0	1	2	3	4	5
0	0	1	2	nul	nul	5
1	nul	1	nul	nul	nul	nul
2	nul	1	2	nul	nul	nul
3	nul	nul	2	3	4	nul
4	nul	nul	nul	nul	4	nul
5	nul	1	2	3	4	5

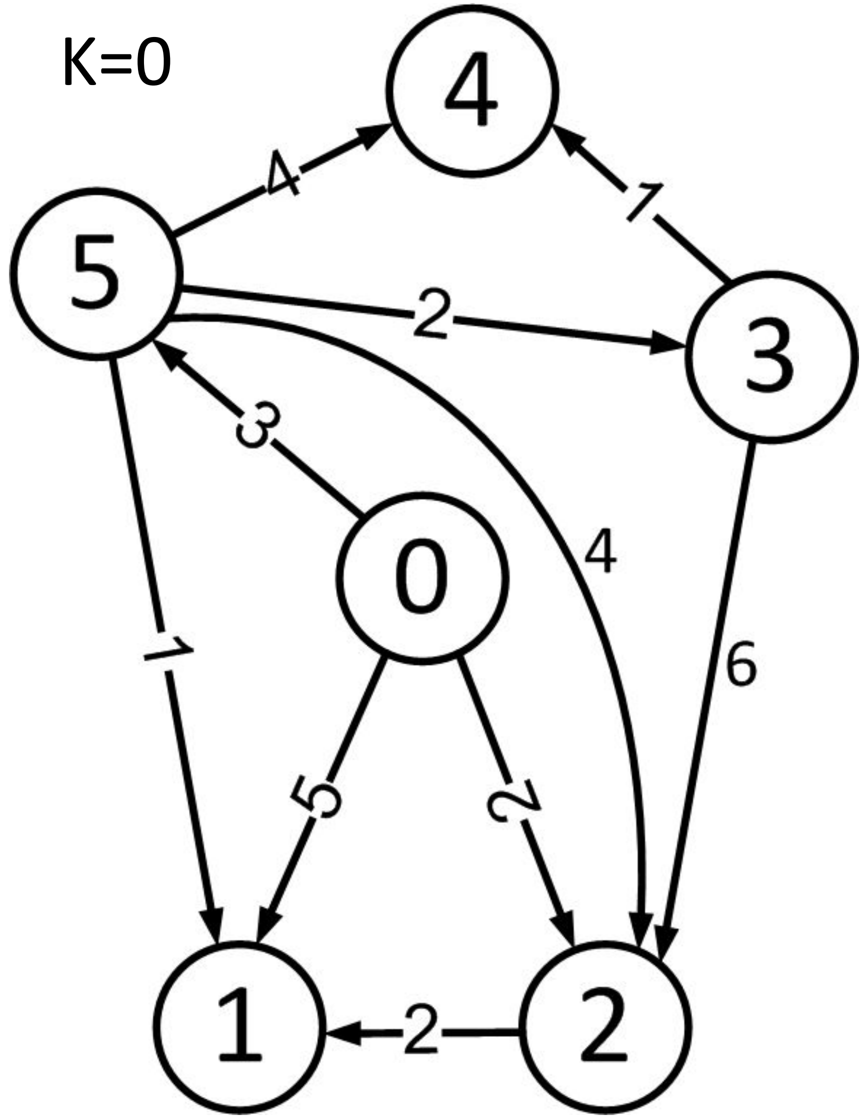


Floyd-Warshall's Search

dist	0	1	2	3	4	5
0	0	5	2	+∞	+∞	3
1	+∞	0	+∞	+∞	+∞	+∞
2	+∞	2	0	+∞	+∞	+∞
3	+∞	+∞	6	0	1	+∞
4	+∞	+∞	+∞	+∞	0	+∞
5	+∞	1	4	2	4	0

next	0	1	2	3	4	5
0	0	1	2	nul	nul	5
1	nul	1	nul	nul	nul	nul
2	nul	1	2	nul	nul	nul
3	nul	nul	2	3	4	nul
4	nul	nul	nul	nul	4	nul
5	nul	1	2	3	4	5

K=0

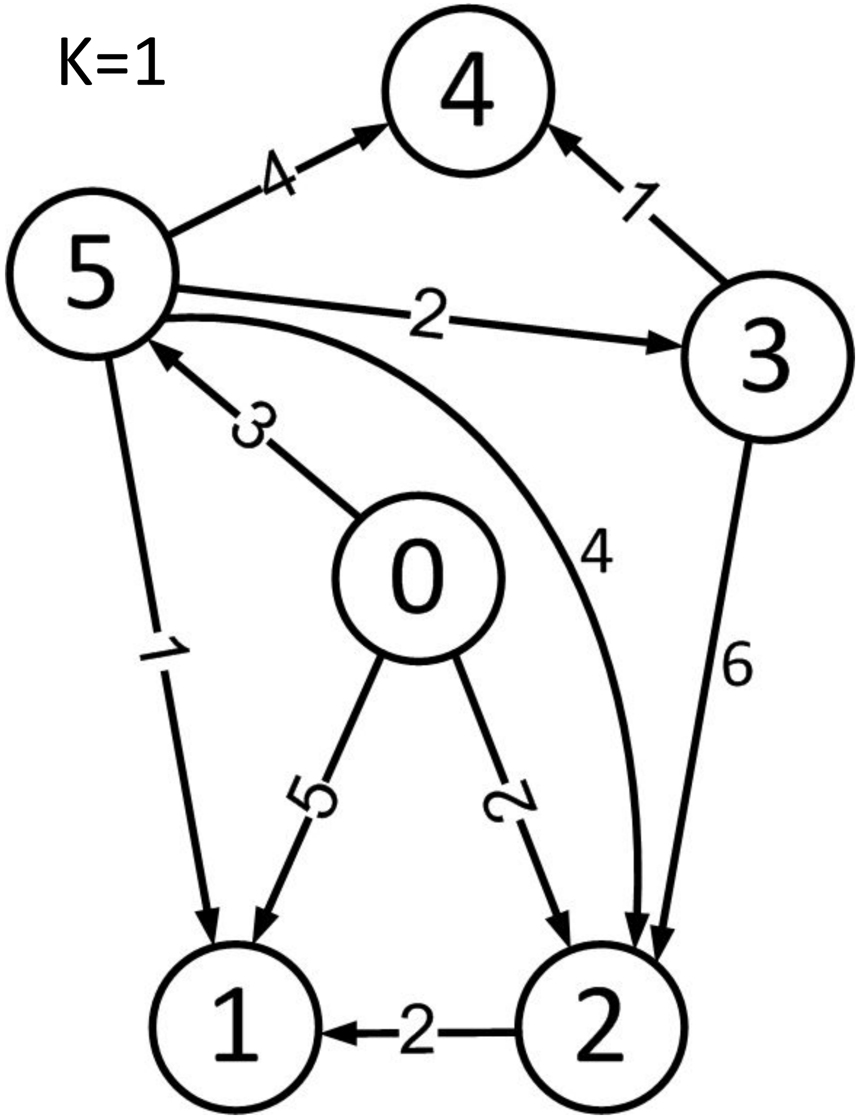


Floyd-Warshall’s Search

dist	0	1	2	3	4	5
0	0	5	2	+∞	+∞	3
1	+∞	0	+∞	+∞	+∞	+∞
2	+∞	2	0	+∞	+∞	+∞
3	+∞	+∞	6	0	1	+∞
4	+∞	+∞	+∞	+∞	0	+∞
5	+∞	1	4	2	4	0

next	0	1	2	3	4	5
0	0	1	2	nul	nul	5
1	nul	1	nul	nul	nul	nul
2	nul	1	2	nul	nul	nul
3	nul	nul	2	3	4	nul
4	nul	nul	nul	nul	4	nul
5	nul	1	2	3	4	5

K=1

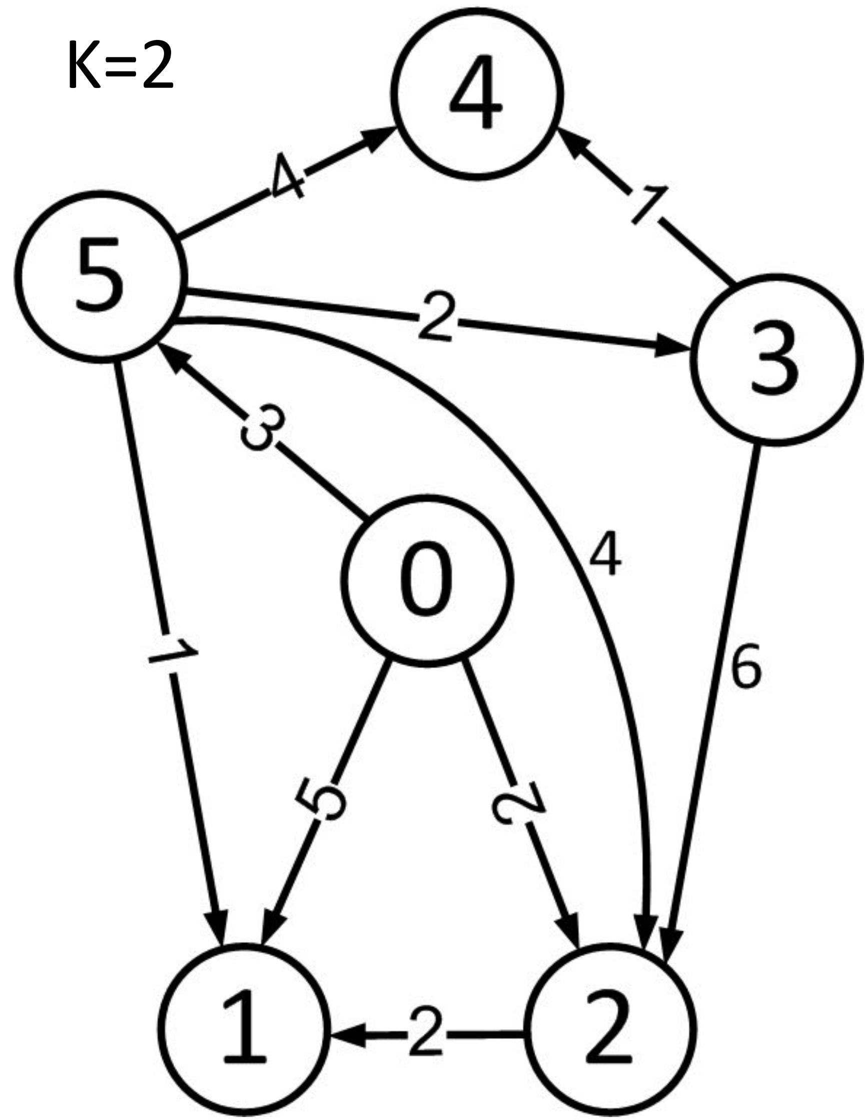


Floyd-Warshall's Search

dist	0	1	2	3	4	5
0	0	4	2	+∞	+∞	3
1	+∞	0	+∞	+∞	+∞	+∞
2	+∞	2	0	+∞	+∞	+∞
3	+∞	8	6	0	1	+∞
4	+∞	+∞	+∞	+∞	0	+∞
5	+∞	1	4	2	4	0

next	0	1	2	3	4	5
0	0	2	2	nul	nul	5
1	nul	1	nul	nul	nul	nul
2	nul	1	2	nul	nul	nul
3	nul	2	2	3	4	nul
4	nul	nul	nul	nul	4	nul
5	nul	1	2	3	4	5

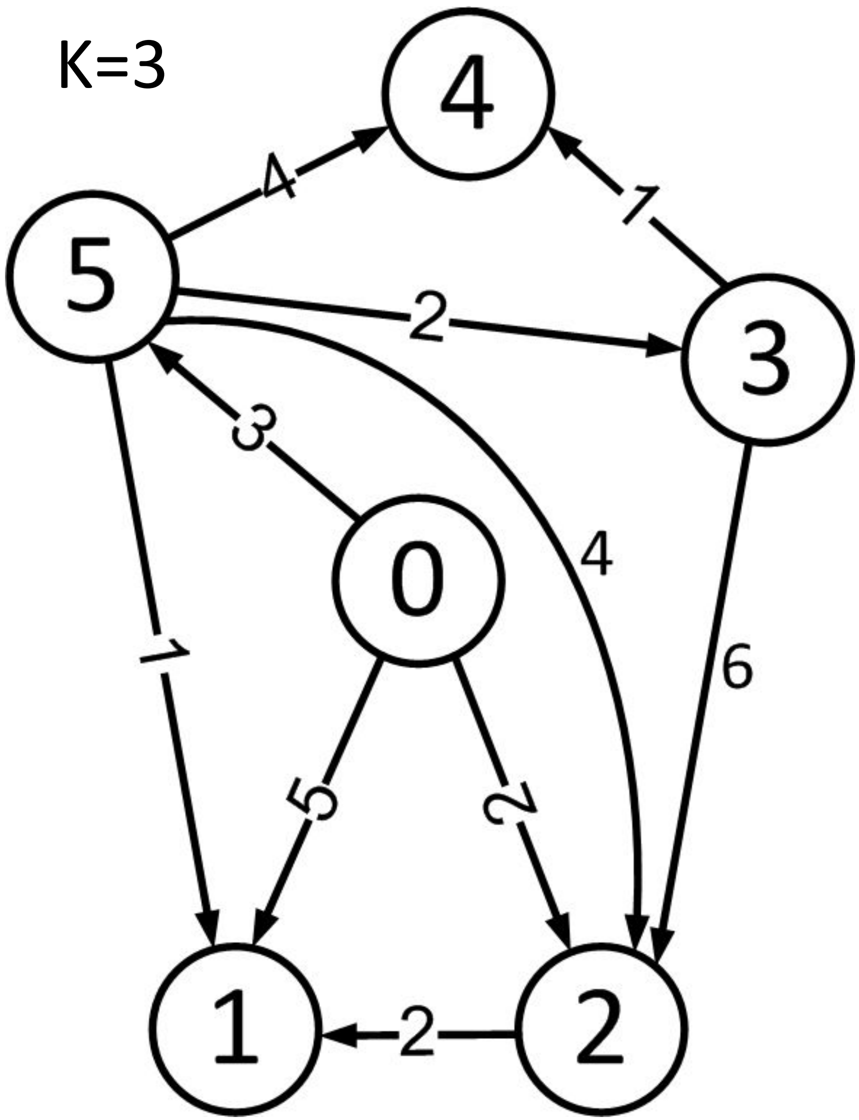
K=2



Floyd-Warshall's Search

dist	0	1	2	3	4	5
0	0	4	2	+∞	+∞	3
1	+∞	0	+∞	+∞	+∞	+∞
2	+∞	2	0	+∞	+∞	+∞
3	+∞	8	6	0	1	+∞
4	+∞	+∞	+∞	+∞	0	+∞
5	+∞	1	4	2	**3**	0

next	0	1	2	3	4	5
0	0	2	2	nul	nul	5
1	nul	1	nul	nul	nul	nul
2	nul	1	2	nul	nul	nul
3	nul	2	2	3	4	nul
4	nul	nul	nul	nul	4	nul
5	nul	1	2	3	**3**	5

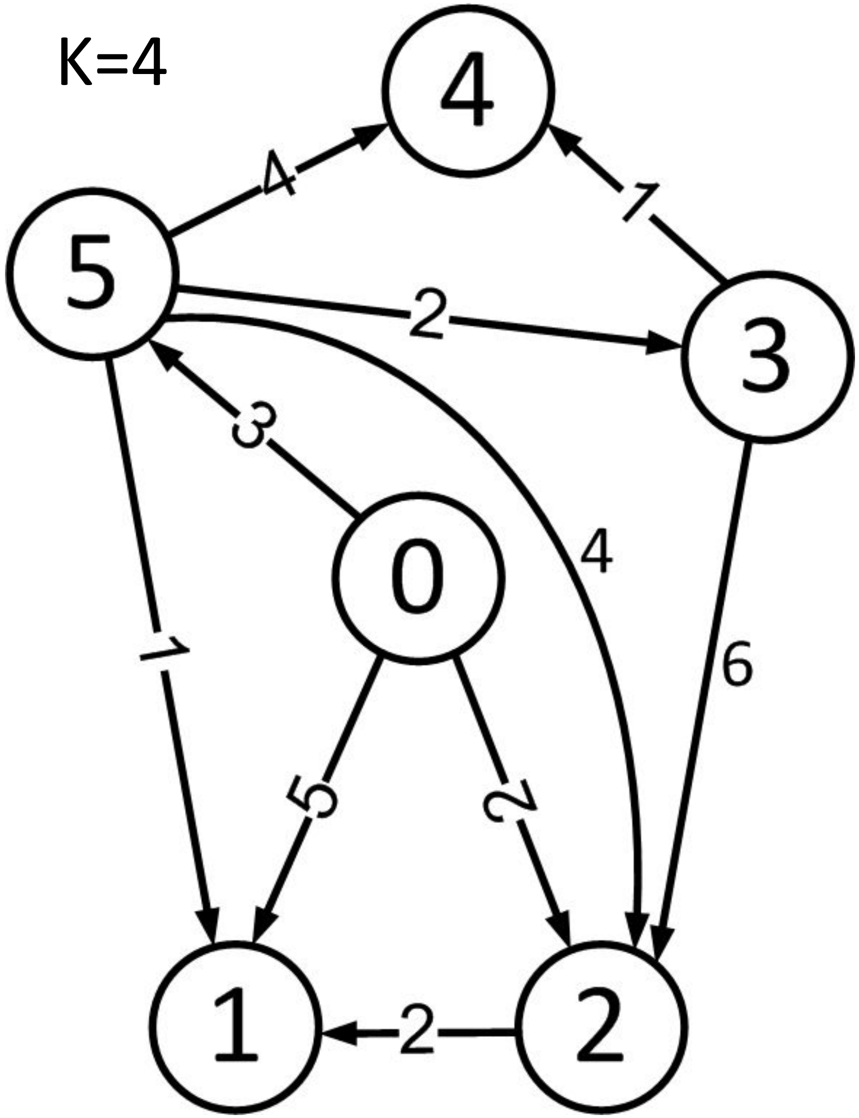


Floyd-Warshall's Search

dist	0	1	2	3	4	5
0	0	4	2	+∞	+∞	3
1	+∞	0	+∞	+∞	+∞	+∞
2	+∞	2	0	+∞	+∞	+∞
3	+∞	8	6	0	1	+∞
4	+∞	+∞	+∞	+∞	0	+∞
5	+∞	1	4	2	3	0

next	0	1	2	3	4	5
0	0	2	2	nul	nul	5
1	nul	1	nul	nul	nul	nul
2	nul	1	2	nul	nul	nul
3	nul	2	2	3	4	nul
4	nul	nul	nul	nul	4	nul
5	nul	1	2	3	3	5

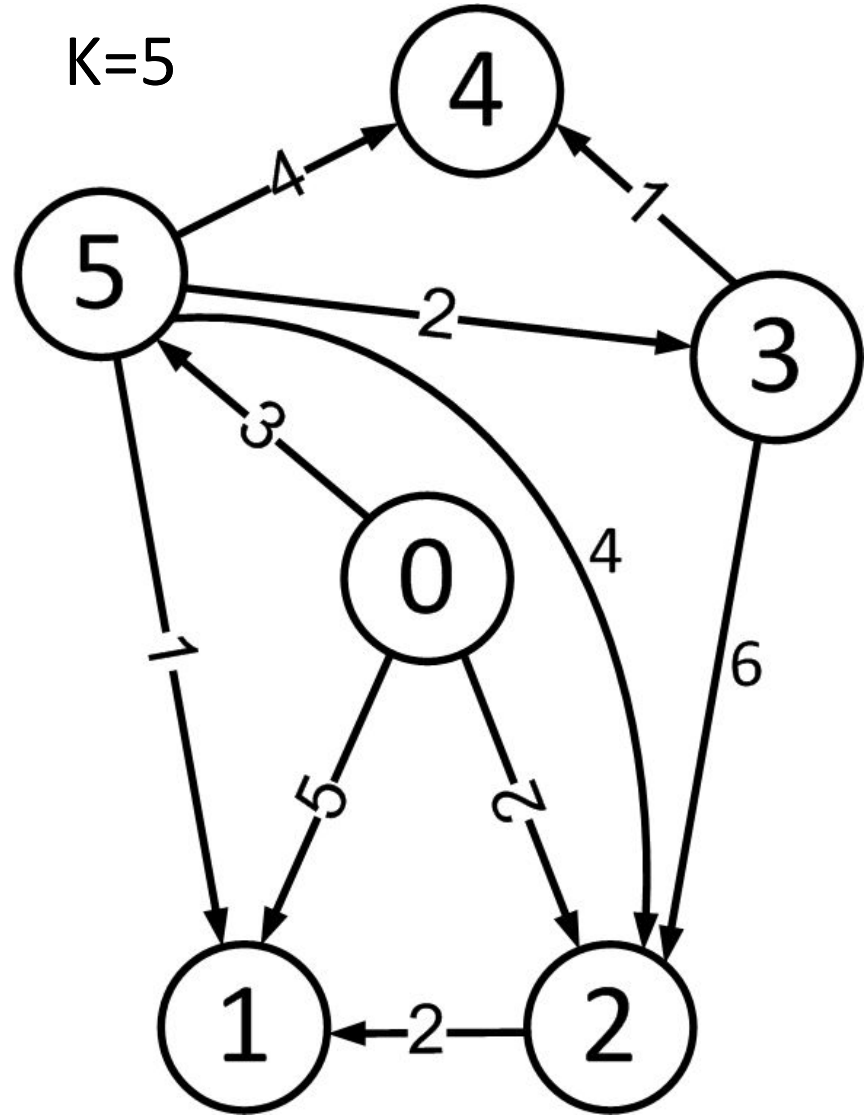
K=4



Floyd-Warshall’s Search

dist	0	1	2	3	4	5
0	0	4	2	5	6	3
1	+∞	0	+∞	+∞	+∞	+∞
2	+∞	2	0	+∞	+∞	+∞
3	+∞	8	6	0	1	+∞
4	+∞	+∞	+∞	+∞	0	+∞
5	+∞	1	4	2	3	0

next	0	1	2	3	4	5
0	0	2	2	5	5	5
1	nul	1	nul	nul	nul	nul
2	nul	1	2	nul	nul	nul
3	nul	2	2	3	4	nul
4	nul	nul	nul	nul	4	nul
5	nul	1	2	3	3	5



Floyd-Warshall's Search

Shortest path reconstruction:

```
procedure Path(u, v)
    if next[u][v] = null then
        return []
    path = [u]
    while u ≠ v
        u ← next[u][v]
        path.append(u)
    return path
```

Ex:
path from 0->4 is 0->5->3->4

dist	0	1	2	3	4	5
0	0	4	2	5	6	3
1	$+\infty$	0	$+\infty$	$+\infty$	$+\infty$	$+\infty$
2	$+\infty$	2	0	$+\infty$	$+\infty$	$+\infty$
3	$+\infty$	8	6	0	1	$+\infty$
4	$+\infty$	$+\infty$	$+\infty$	$+\infty$	0	$+\infty$
5	$+\infty$	1	4	2	3	0

next	0	1	2	3	4	5
0	0	2	2	5	5	5
1	nul	1	nul	nul	nul	nul
2	nul	1	2	nul	nul	nul
3	nul	2	2	3	4	nul
4	nul	nul	nul	nul	4	nul
5	nul	1	2	3	3	5

Floyd-Warshall's Search

How to detect negative cycle?

- The Floyd–Warshall algorithm iteratively revises path lengths between all pairs of vertices (i, j), including where $i = j$;
- Initially, the length of the path (i, i) is zero;
- A path $[i, k, \ldots, i]$ can only improve upon this if it has length less than zero, i.e. denotes a negative cycle;
- Thus, after the algorithm, (i, i) will be negative if there exists a negative-length path from i back to i.

Floyd-Warshall’s Search

- **Complete?**
  - Yes.
- **Optimal?**
  - Yes, if the state space doesn’t contain cycles with negative total values.
- **Complexity** in terms of number of vertices: n
  - Time complexity: $O(n^3)$
  - Better in dense graphs, why?
  - Space complexity: $O(n^2)$