

CS101 Algorithms and Data Structures

Shortest Path: Floyd-Warshall
Textbook Ch 24, 25



Outline

- Dijkstra's algorithm
- Floyd-Warshall algorithm

Dijkstra's algorithm

We will iterate $|V|$ times:

- Find the unvisited vertex v that has a minimum distance to it
- Mark it as visited
- Consider its every adjacent vertex w that is unvisited:
 - Is the distance to v plus the weight of the edge (v, w) less than our currently known shortest distance to w ?
 - If so, update the shortest distance to w and record v as the previous pointer

Continue iterating until all vertices are visited or **all remaining vertices have a distance of infinity**

Outline

- Dijkstra's algorithm
- Floyd-Warshall algorithm

Background

Dijkstra's algorithm finds the shortest path between two nodes

- Run time: $O(|E| \ln(|V|))$

If we wanted to find the shortest path between all pairs of nodes, we could apply Dijkstra's algorithm to each vertex:

- Run time: $O(|V| |E| \ln(|V|))$

Background

Now, Dijkstra's algorithm has the following run times:

- Best case:
If $|E| = \Theta(|V|)$, running Dijkstra for each vertex is $O(|V|^2 \ln(|V|))$
- Worst case:
If $|E| = \Theta(|V|^2)$, running Dijkstra for each vertex is $O(|V|^3 \ln(|V|))$

Problem

Question: for the worst case, can we find a $O(|V|^3 \ln |V|)$ algorithm?

We will look at the Floyd-Warshall algorithm

- It works with positive or negative weights with no negative cycle

$$F-W : O(|V|^3)$$

✓ ~~✗~~ no ~~is~~ cycle

Strategy

First, let's consider only edges that connect vertices directly:

$$d_{i,j}^{(0)} = \begin{cases} 0 & \text{If } i = j \\ w_{i,j} & \text{If there is an edge from } i \text{ to } j \\ \infty & \text{Otherwise} \end{cases}$$

(directly)

Here, $w_{i,j}$ is the weight of the edge connecting vertices i and j

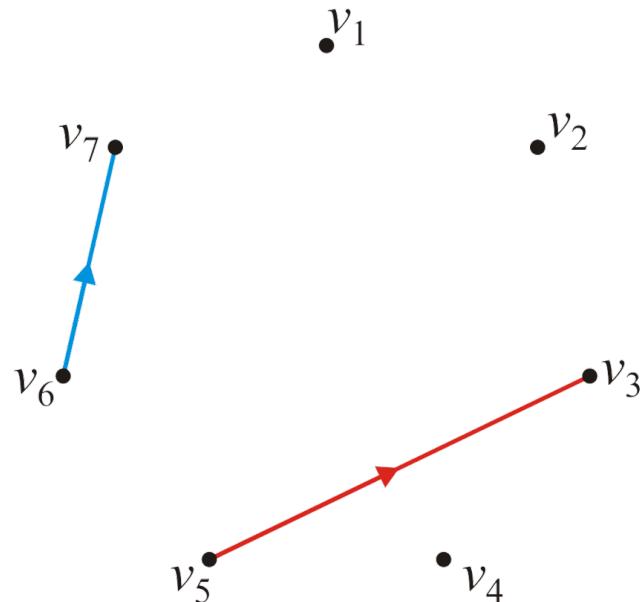
- Note, this can be a directed graph; i.e., it may be that $d_{i,j}^{(0)} \neq d_{j,i}^{(0)}$



Strategy

Consider this graph of seven vertices

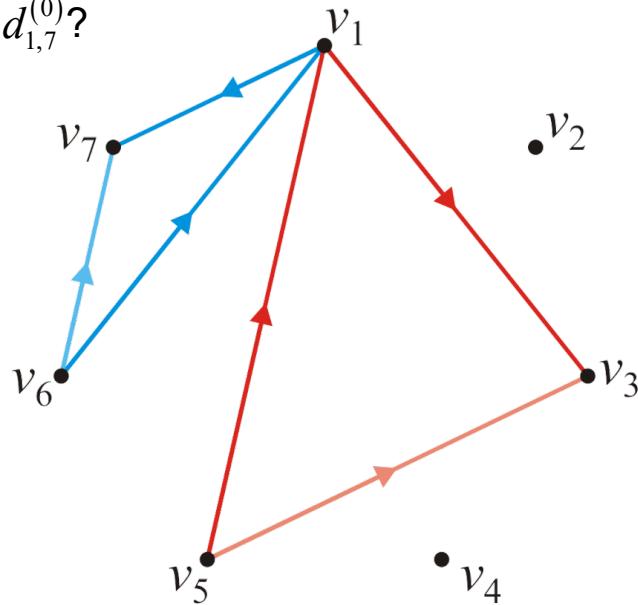
- The edges defining the values $d_{5,3}^{(0)}$ and $d_{6,7}^{(0)}$ are highlighted



Strategy

Suppose now, we want to see whether or not the path going through vertex v_1 is shorter than a direct edge?

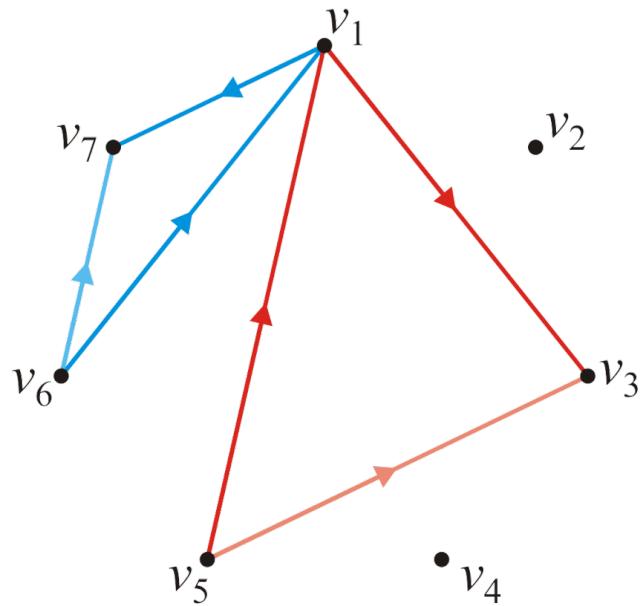
- Is $d_{5,3}^{(0)} > d_{5,1}^{(0)} + d_{1,3}^{(0)}$?
- Is $d_{6,7}^{(0)} > d_{6,1}^{(0)} + d_{1,7}^{(0)}$?



Strategy

Thus, for each pair of edges, we will define $d_{i,j}^{(1)}$ by calculating:

$$d_{i,j}^{(1)} = \min \left\{ d_{i,j}^{(0)}, d_{i,1}^{(0)} + d_{1,j}^{(0)} \right\}$$



Strategy

We need just run the algorithm for each pair of vertices:

```
for ( int i = 0; i < num_vertices; ++i ) {  
    for ( int j = 0; j < num_vertices; ++j ) {  
        d[i][j] = std::min( d[i][j], d[i][0] + d[0][j] );  
    }  
}
```

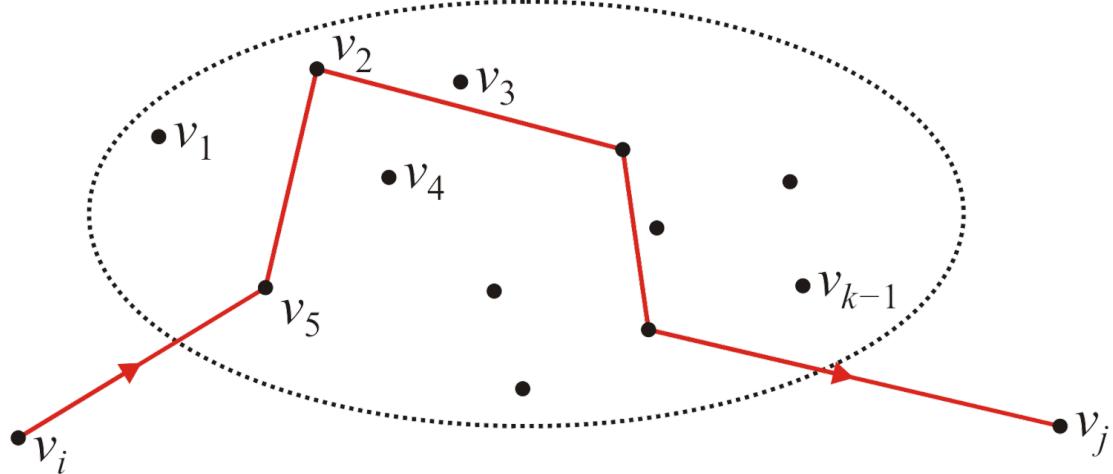


Q

The General Step

Define $d_{i,j}^{(k-1)}$ as the shortest distance, but only allowing intermediate visits to vertices v_1, v_2, \dots, v_{k-1}

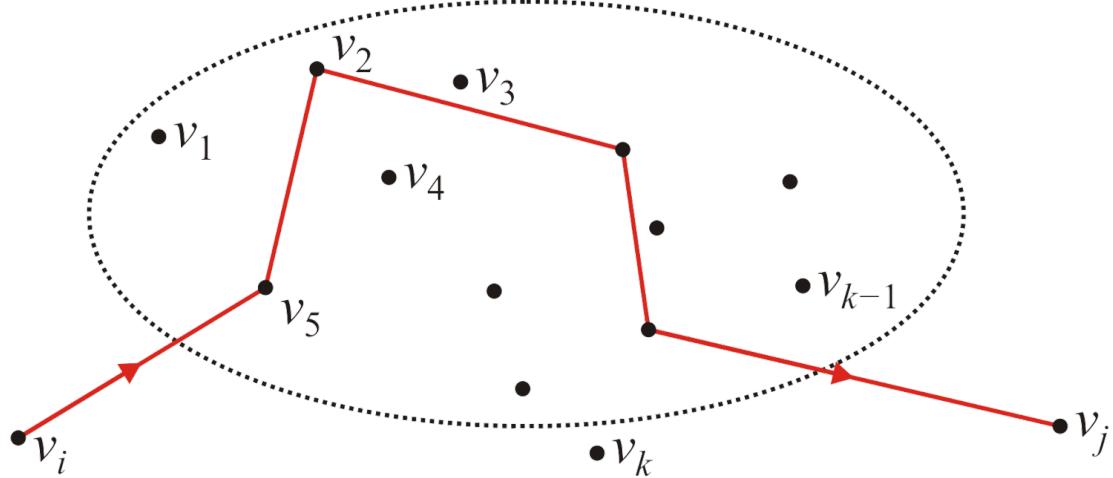
- Suppose we have an algorithm that has found these values for all pairs



The General Step

How could we find $d_{i,j}^{(k)}$; that is, the shortest path allowing intermediate visits to vertices $v_1, v_2, \dots, v_{k-1}, v_k$?

- Two possibilities: the shortest path includes or does not include v_k

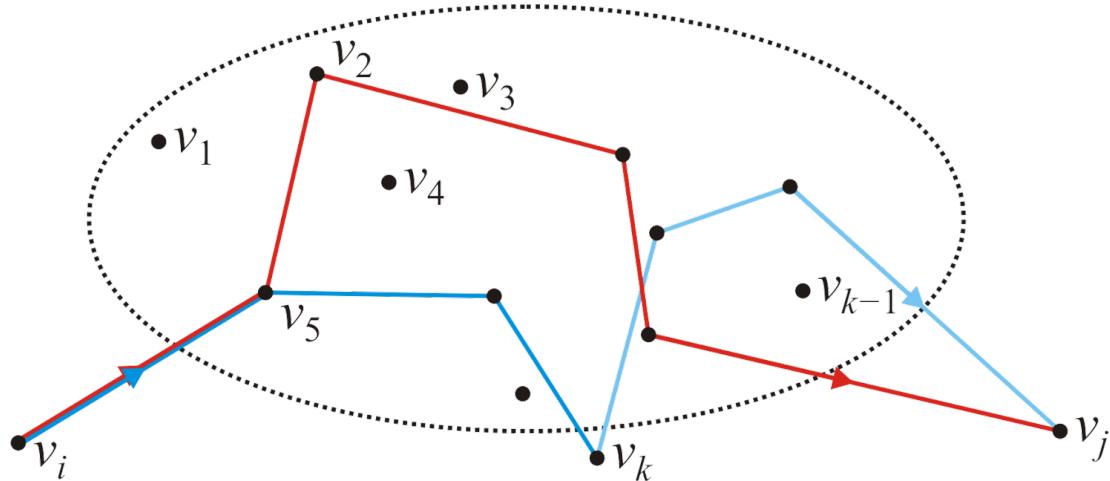


The General Step

(1)

If the shortest path includes v_k , then it must consist of:

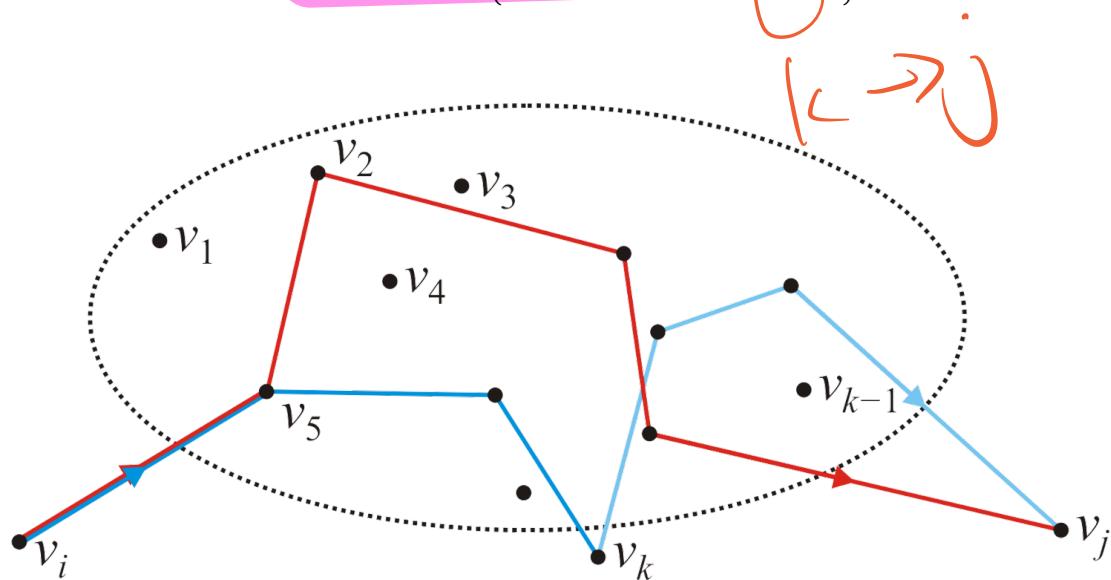
- the shortest path from v_i to v_k
- and then the shortest path from v_k to v_j
- both only allowing intermediate visits to vertices v_1, v_2, \dots, v_{k-1}



The General Step

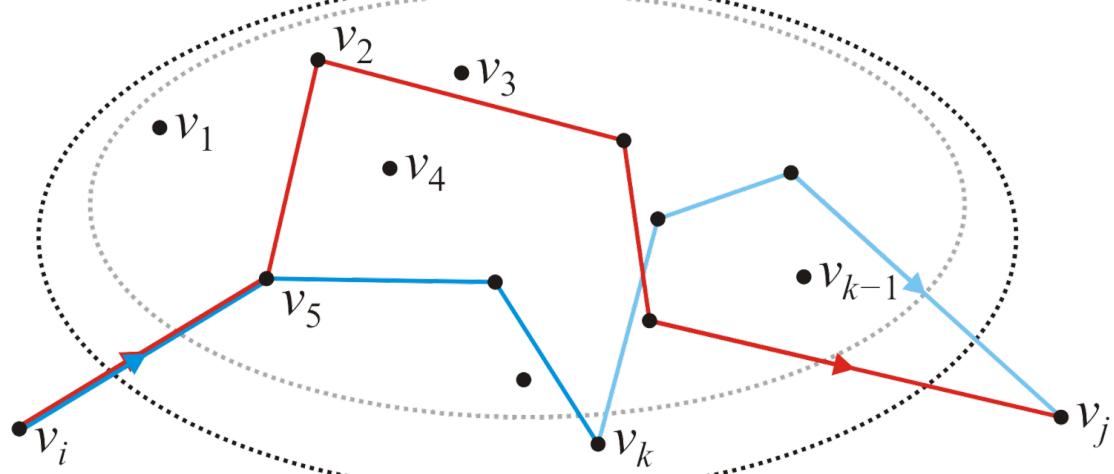
With v_1, v_2, \dots, v_{k-1} as intermediates, we already know the shortest paths from v_i to v_j , v_i to v_k and v_k to v_j

Thus, we calculate $d_{i,j}^{(k)} = \min \{ d_{i,j}^{(k-1)}, d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)} \}$



The General Step

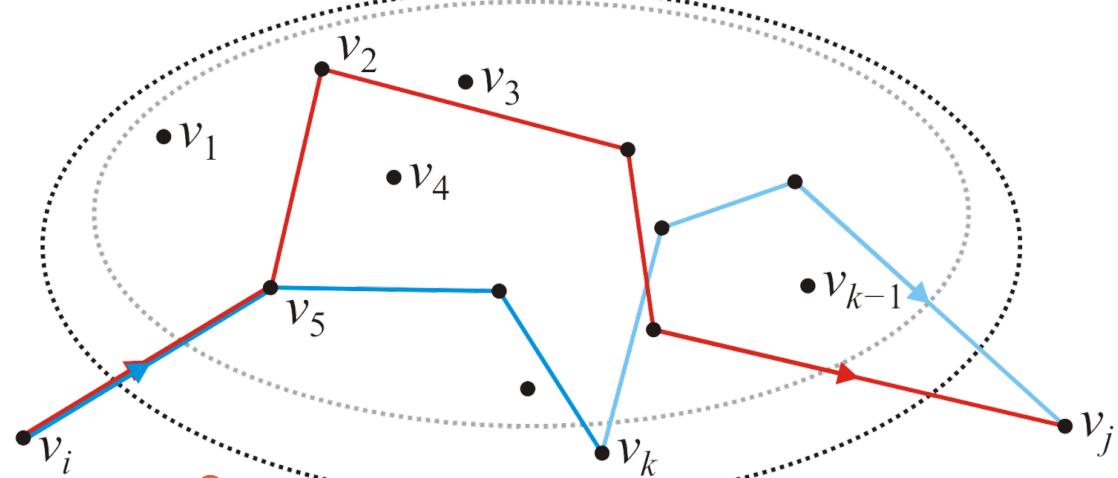
Finding $d_{i,j}^{(k)}$ for all pairs of vertices gives us all shortest paths from v_i to v_j possibly going through vertices v_1, v_2, \dots, v_k



The General Step

The calculation is straight forward:

```
for ( int i = 0; i < num_vertices; ++i ) {  
    for ( int j = 0; j < num_vertices; ++j ) {  
        d[i][j] = std::min( d[i][j], d[i][k-1] + d[k-1][j] );  
    }  
}
```



imple

Imp

The Floyd-Warshall Algorithm



```
// Initialize the matrix d  
// ...  
  
for ( int k = 0; k < num_vertices; ++k ) {  
    for ( int i = 0; i < num_vertices; ++i ) {  
        for ( int j = 0; j < num_vertices; ++j ) {  
            d[i][j] = std::min( d[i][j], d[i][k] + d[k][j] );  
        }  
    }  
}
```

3 | 2
| 1

Run time? $\Theta(|V|^3)$

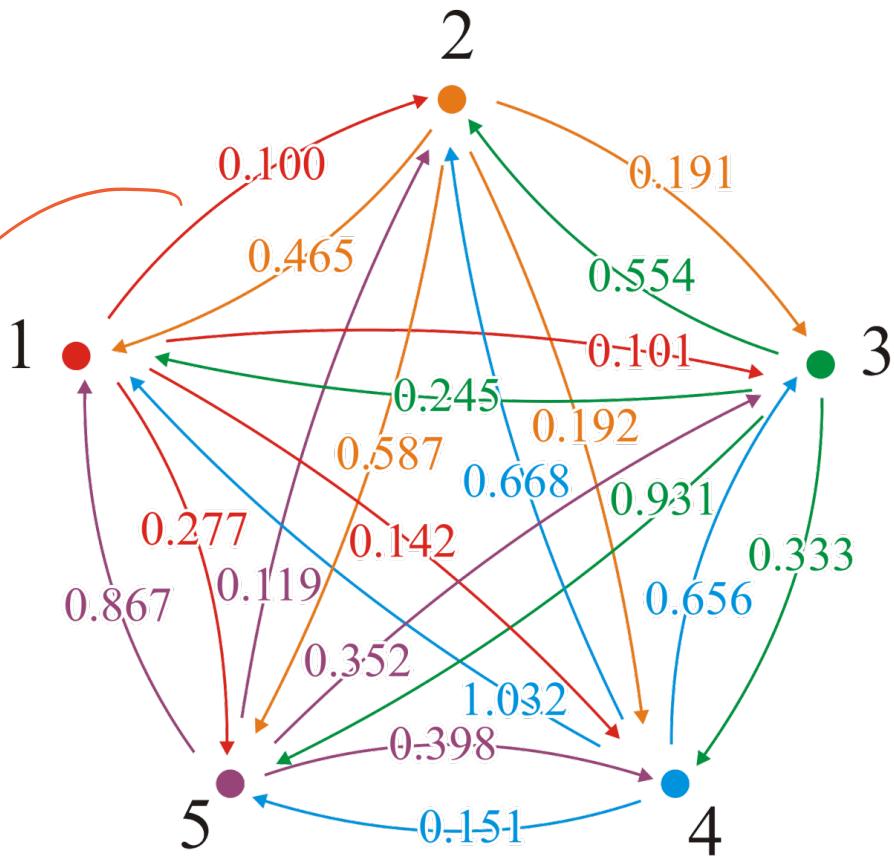
Example

Consider this graph



directed.

diff

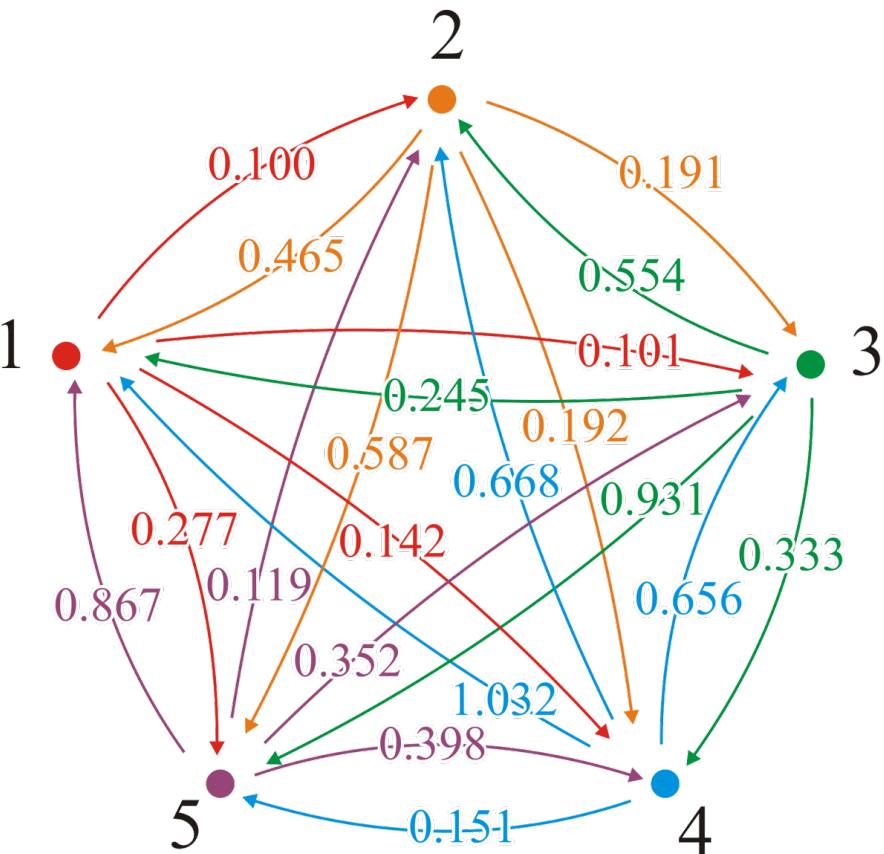


Example

The adjacency matrix is

$$\begin{pmatrix} 0 & 0.100 & 0.101 & 0.142 & 0.277 \\ 0.465 & 0 & 0.191 & 0.192 & 0.587 \\ 0.245 & 0.554 & 0 & 0.333 & 0.931 \\ 1.032 & 0.668 & 0.656 & 0 & 0.151 \\ 0.867 & 0.119 & 0.352 & 0.398 & 0 \end{pmatrix}$$

This would define our matrix $\mathbf{D} = (d_{ij})$



Example

With the first pass, $k = 1$, we attempt passing through vertex v_1

0	0.100	0.101	0.142	0.277
0.465	0	0.191	0.192	0.587
0.245	0.554	0	0.333	0.931
1.032	0.668	0.656	0	0.151
0.867	0.119	0.352	0.398	0

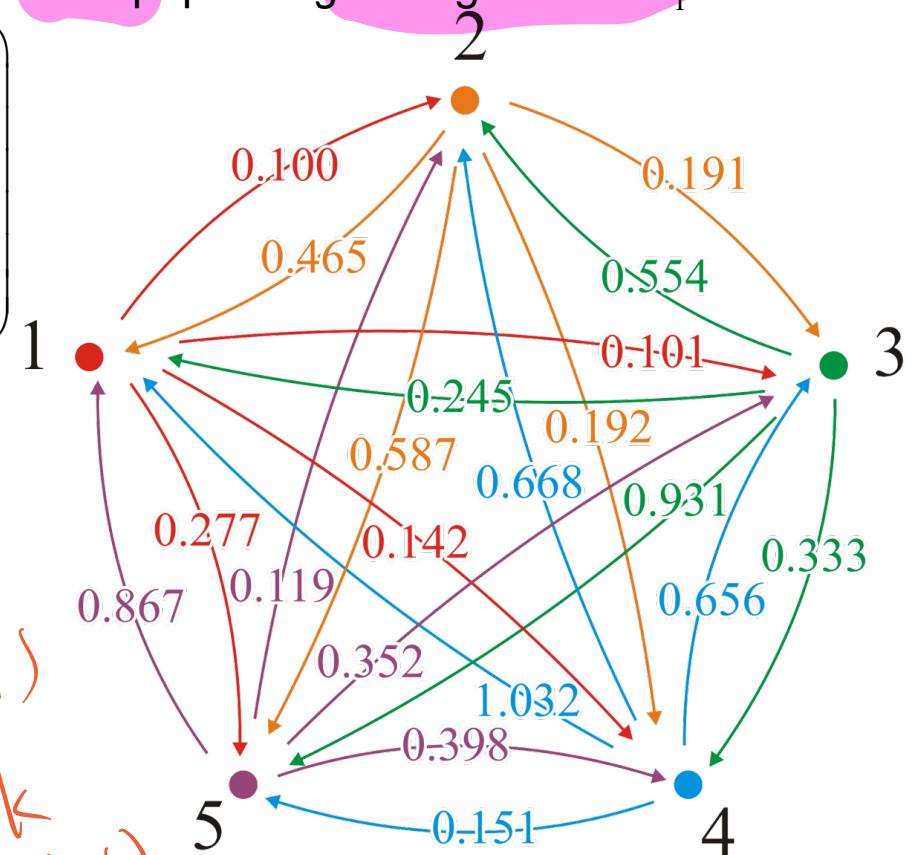
(k)

$d_{ij} = \min$

$\{d_{ij}, d_{ik}\}$

$(k-1)$

d_{ik}, d_{jk}



Example

With the first pass, $t = 1$, we attempt passing through vertex v_1

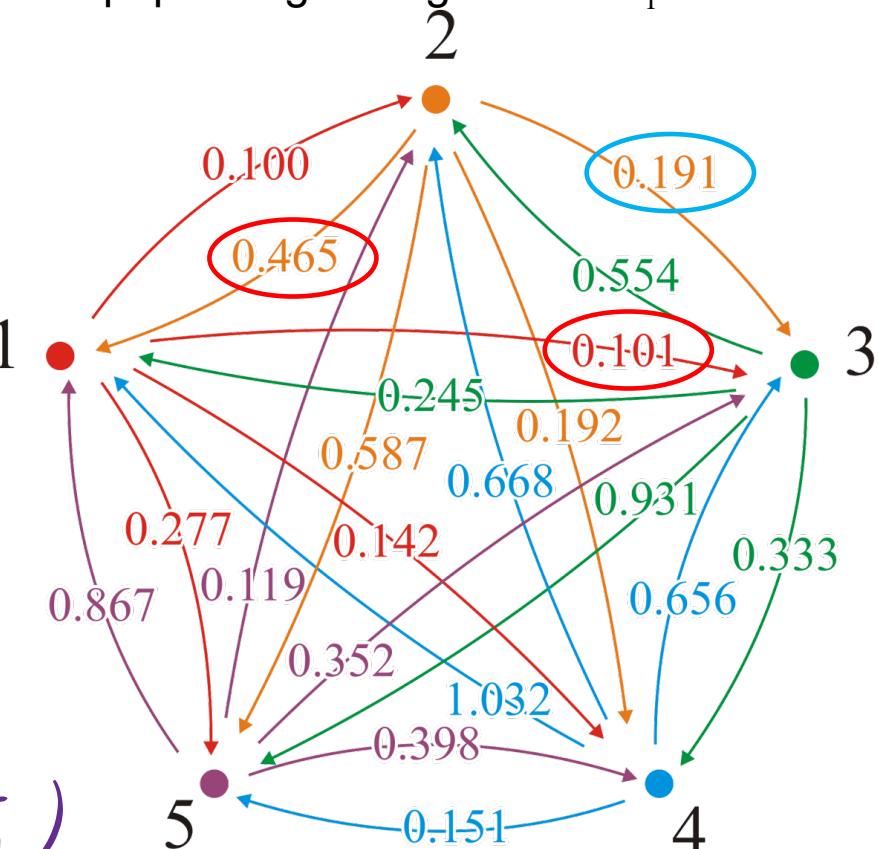
0	0.100	0.101	0.142	0.277
0.465	0	0.191	0.192	0.587
0.245	0.554	0	0.333	0.931
1.032	0.668	0.656	0	0.151
0.867	0.119	0.352	0.398	0

We would start:

$$(2, 3) \rightarrow (2, 1, 3)$$

$$0.191 \nless 0.465 + 0.101$$

$(2, 3) \leftarrow (2, \psi, (2, \bar{N})$



Example

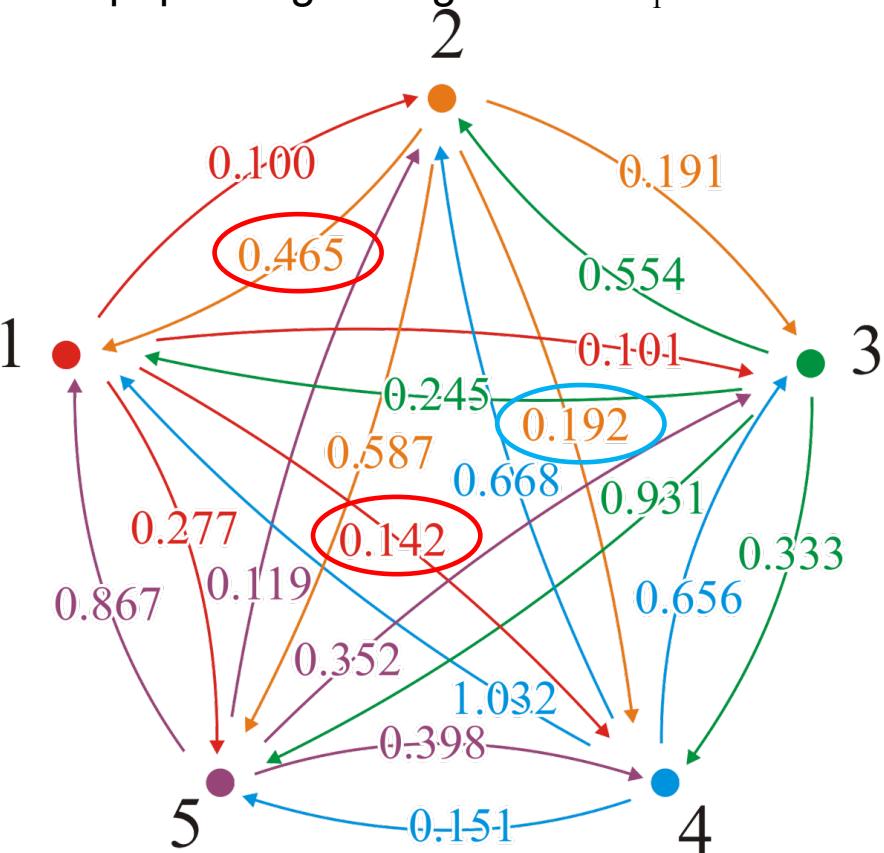
With the first pass, $k = 1$, we attempt passing through vertex v_1

0	0.100	0.101	0.142	0.277
0.465	0	0.191	0.192	0.587
0.245	0.554	0	0.333	0.931
1.032	0.668	0.656	0	0.151
0.867	0.119	0.352	0.398	0

We would start:

$$(2, 4) \rightarrow (2, 1, 4)$$

$$0.192 \nleq 0.465 + 0.142$$



Example

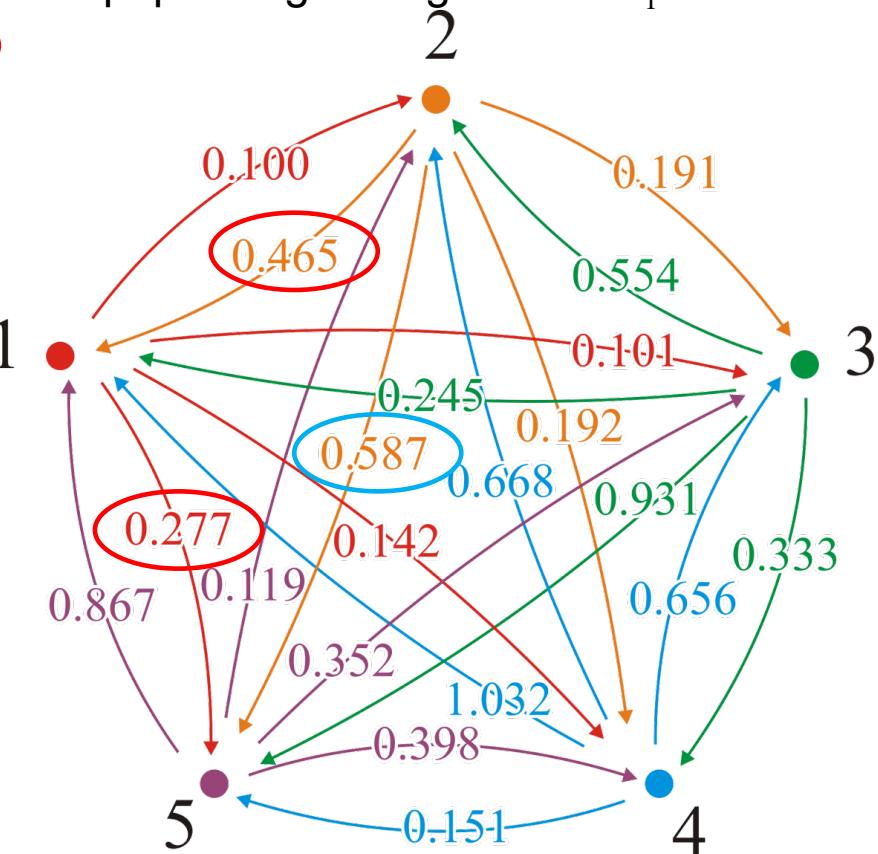
With the first pass, $k = 1$, we attempt passing through vertex v_1

0	0.100	0.101	0.142	0.277
0.465	0	0.191	0.192	0.587
0.245	0.554	0	0.333	0.931
1.032	0.668	0.656	0	0.151
0.867	0.119	0.352	0.398	0

We would start:

$$(2, 5) \rightarrow (2, 1, 5)$$

$$0.587 \nless 0.465 + 0.277$$



Example

to

With the first pass, $k = 1$, we attempt passing through vertex v_1

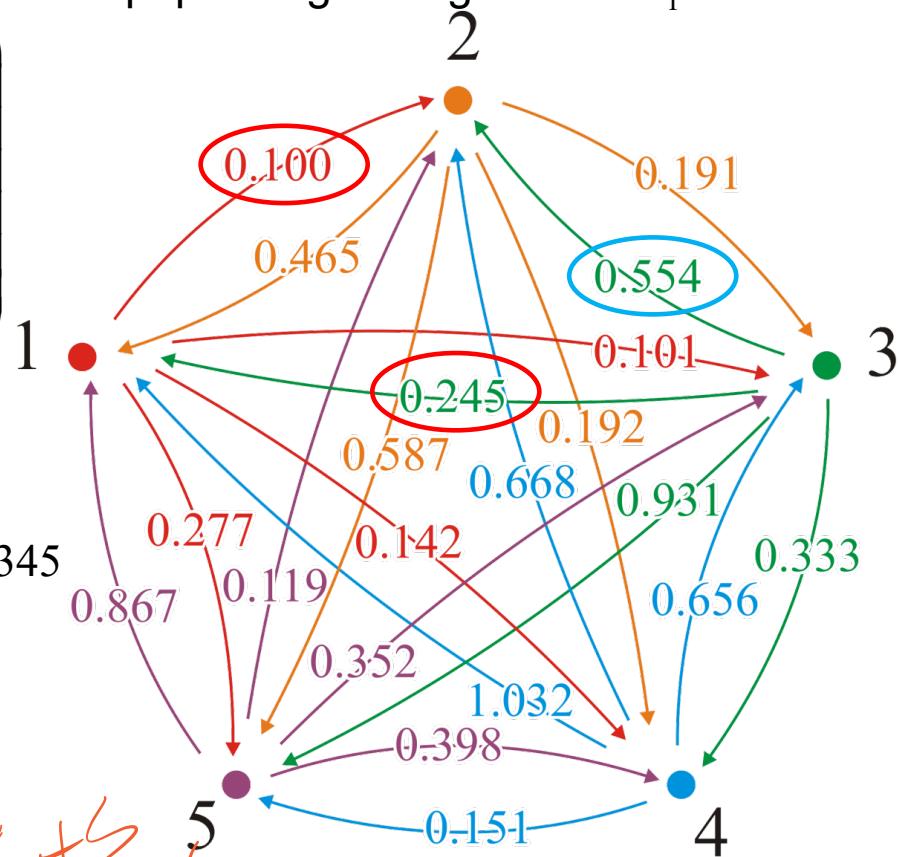
0	0.100	0.101	0.142	0.277
0.465	0	0.191	0.192	0.587
0.245	0.554	0	0.333	0.931
1.032	0.668	0.656	0	0.151
0.867	0.119	0.352	0.398	0

from

Here is a shorter path:

$$(3, 2) \rightarrow (3, 1, 2)$$

$$0.554 > 0.245 + 0.100 = 0.345$$



未 P + 3

Example

With the first pass, $k = 1$, we attempt passing through vertex v_1

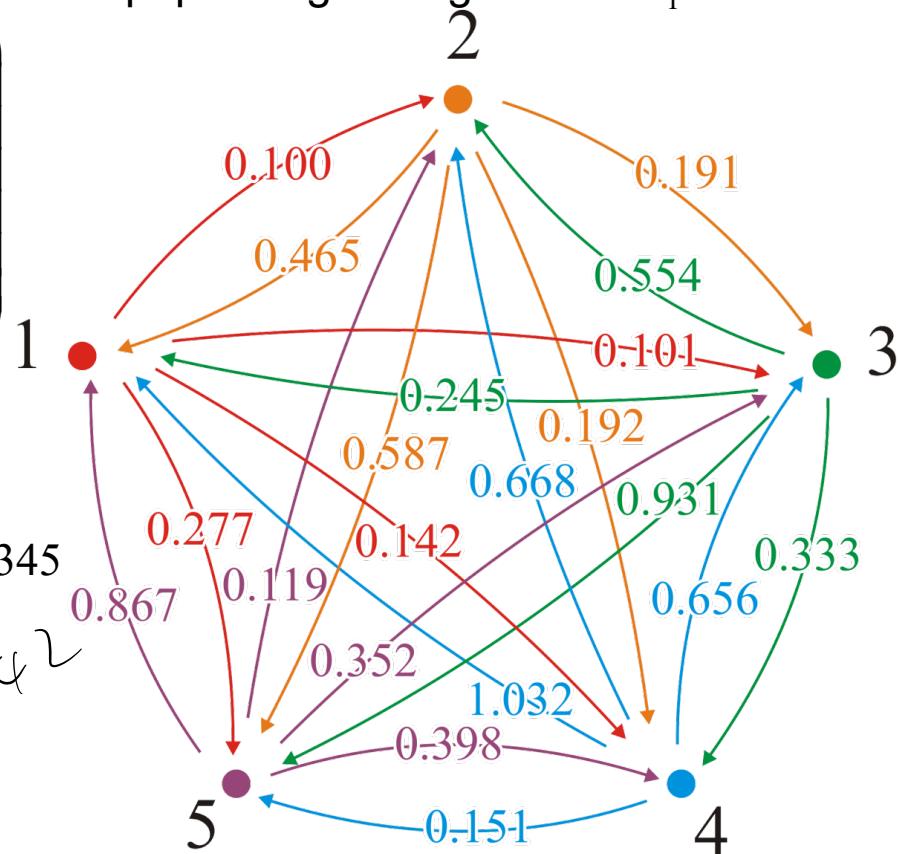
0	0.100	0.101	0.142	0.277
0.465	0	0.191	0.192	0.587
0.245	0.345	0	0.333	0.931
1.032	0.668	0.656	0	0.151
0.867	0.119	0.352	0.398	0

We update the table

$$(3, 2) \rightarrow (3, 1, 2)$$

$$0.554 > 0.245 + 0.100 = 0.345$$

$$\begin{matrix} (3, 4) \\ (3, 1, 4) \\ 0.245 + 0.142 \\ 0.345 \end{matrix}$$



Example

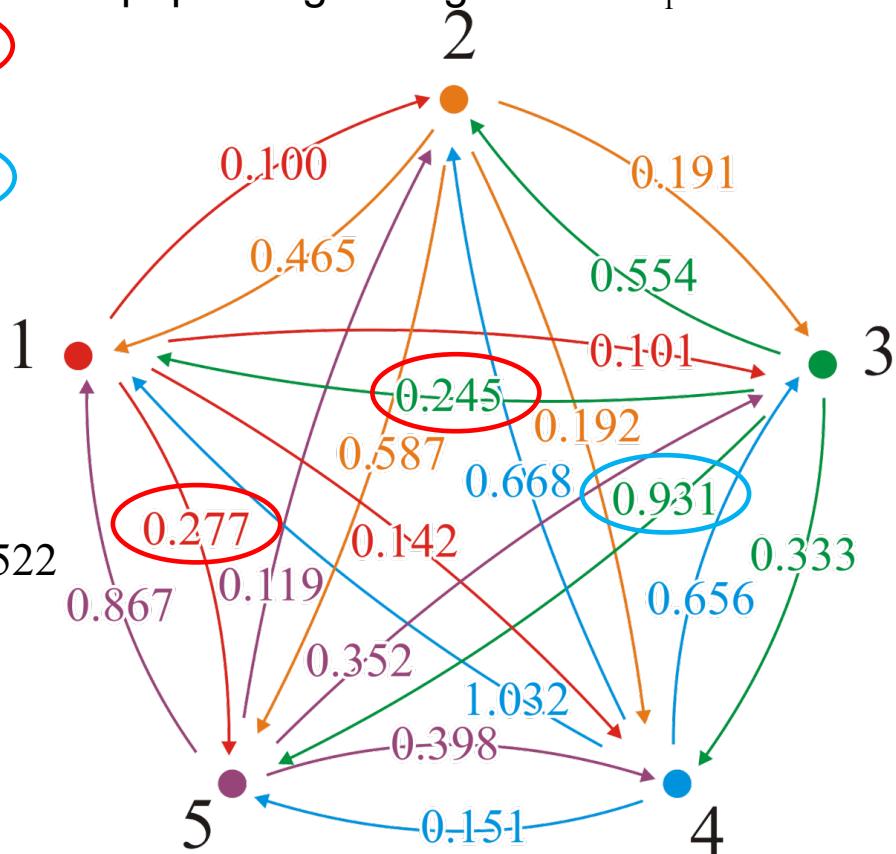
With the first pass, $k = 1$, we attempt passing through vertex v_1

0	0.100	0.101	0.142	0.277
0.465	0	0.191	0.192	0.587
0.245	0.345	0	0.333	0.931
1.032	0.668	0.656	0	0.151
0.867	0.119	0.352	0.398	0

And a second shorter path:

$$(3, 5) \rightarrow (3, 1, 5)$$

$$0.931 > 0.245 + 0.277 = 0.522$$

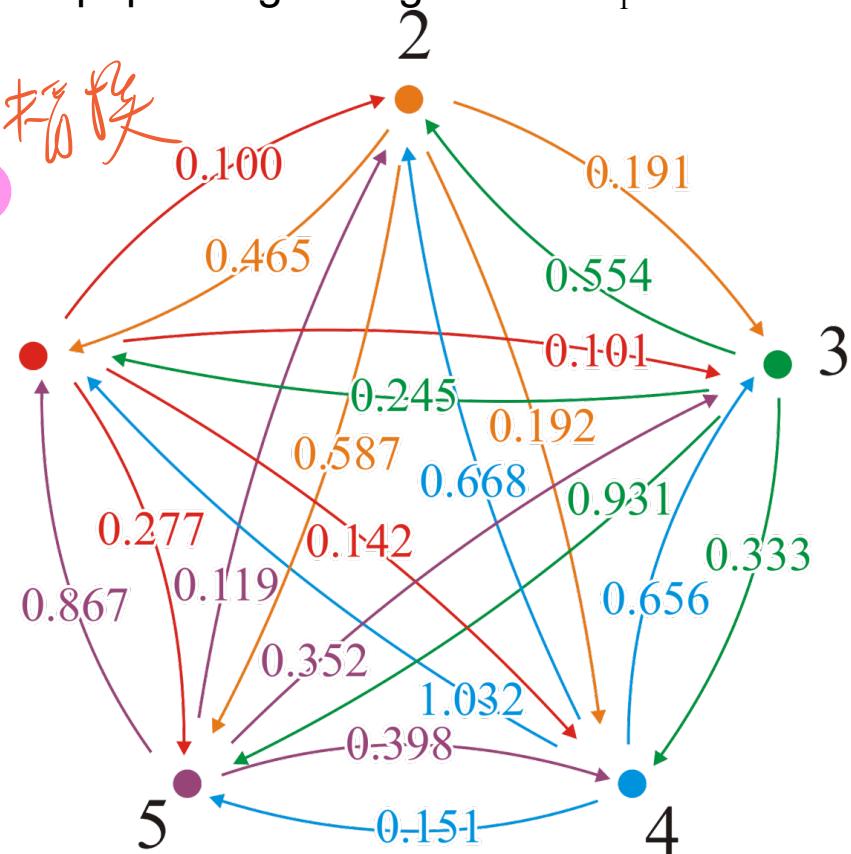


Example

With the first pass, $k = 1$, we attempt passing through vertex v_1

0	0.100	0.101	0.142	0.277
0.465	0	0.191	0.192	0.587
0.245	0.345	0	0.333	0.522
1.032	0.668	0.656	0	0.151
0.867	0.119	0.352	0.398	0

We update the table



Example

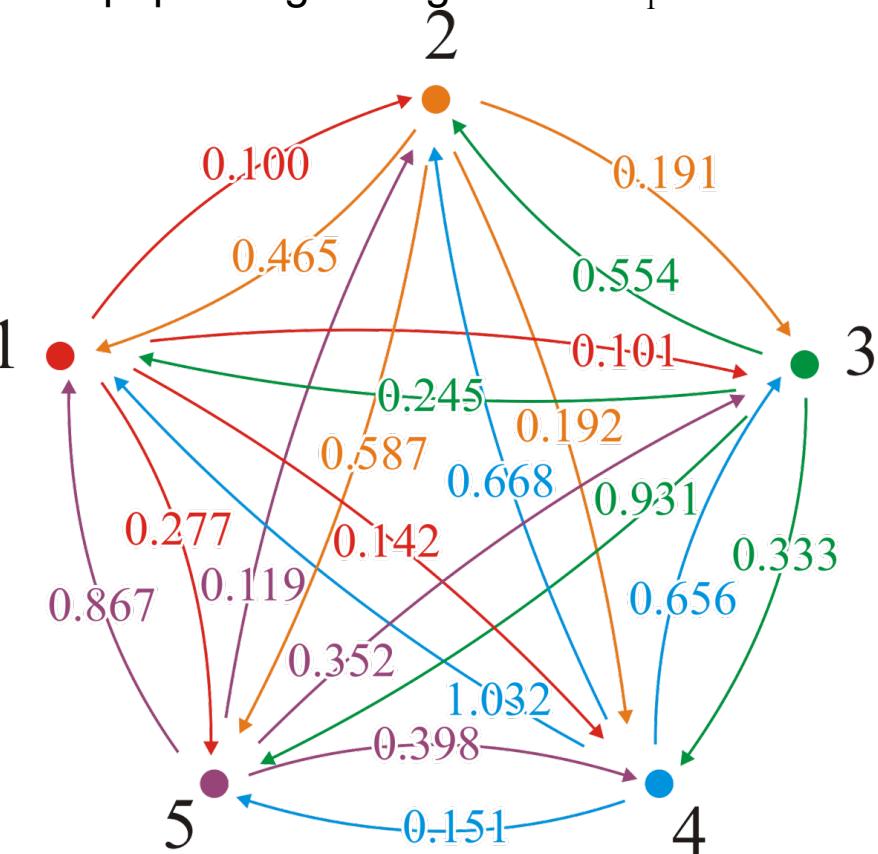
With the first pass, $k = 1$, we attempt passing through vertex v_1

0	0.100	0.101	0.142	0.277
0.465	0	0.191	0.192	0.587
0.245	0.345	0	0.333	0.522
1.032	0.668	0.656	0	0.151
0.867	0.119	0.352	0.398	0

Continuing...

We find that no other shorter paths through vertex v_1 exist

$(4, 2)$ $(5, 2)$
 $(4, 3)$ $(5, 3)$
 $(4, 5)$ $(5, 4)$



Example

With the next pass, $k = 2$, we attempt passing through vertex v_2

0	0.100	0.101	0.142	0.277
0.465	0	0.191	0.192	0.587
0.245	0.345	0	0.333	0.522
1.032	0.668	0.656	0	0.151
0.867	0.119	0.352	0.398	0

There are three shorter paths:

$$(5, 1) \rightarrow (5, 2, 1)$$

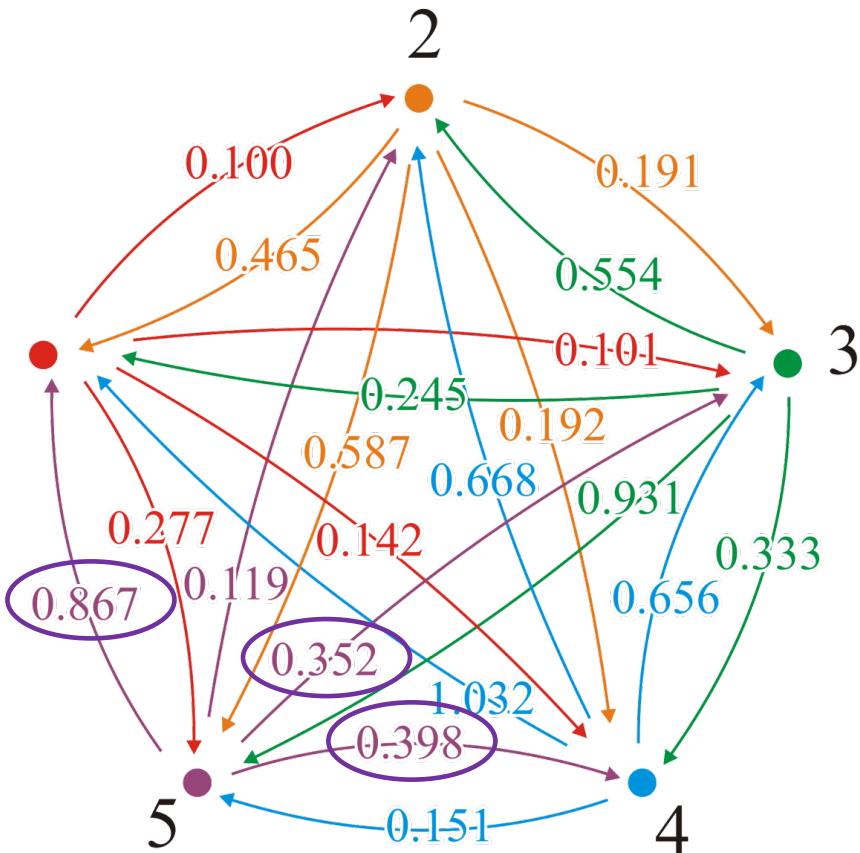
$$0.867 > 0.119 + 0.465 = 0.584$$

$$(5, 3) \rightarrow (5, 2, 3)$$

$$0.352 > 0.119 + 0.191 = 0.310$$

$$(5, 4) \rightarrow (5, 2, 4)$$

$$0.398 > 0.119 + 0.192 = 0.311$$

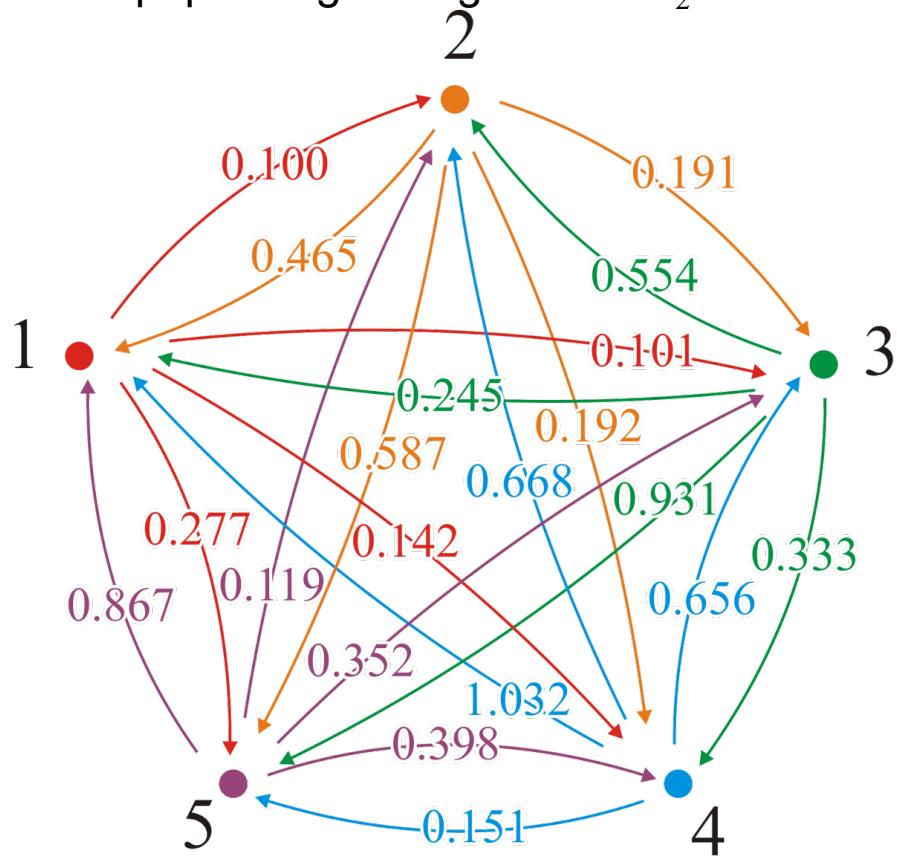


Example

With the next pass, $k = 2$, we attempt passing through vertex v_2

0	0.100	0.101	0.142	0.277
0.465	0	0.191	0.192	0.587
0.245	0.345	0	0.333	0.522
1.032	0.668	0.656	0	0.151
0.584	0.119	0.310	0.311	0

We update the table



Example

With the next pass, $k = 3$, we attempt passing through vertex v_3

0	0.100	0.101	0.142	0.277
0.465	0	0.191	0.192	0.587
0.245	0.345	0	0.333	0.522
1.032	0.668	0.656	0	0.151
0.584	0.119	0.310	0.311	0

There are three shorter paths:

$$(2, 1) \rightarrow (2, 3, 1)$$

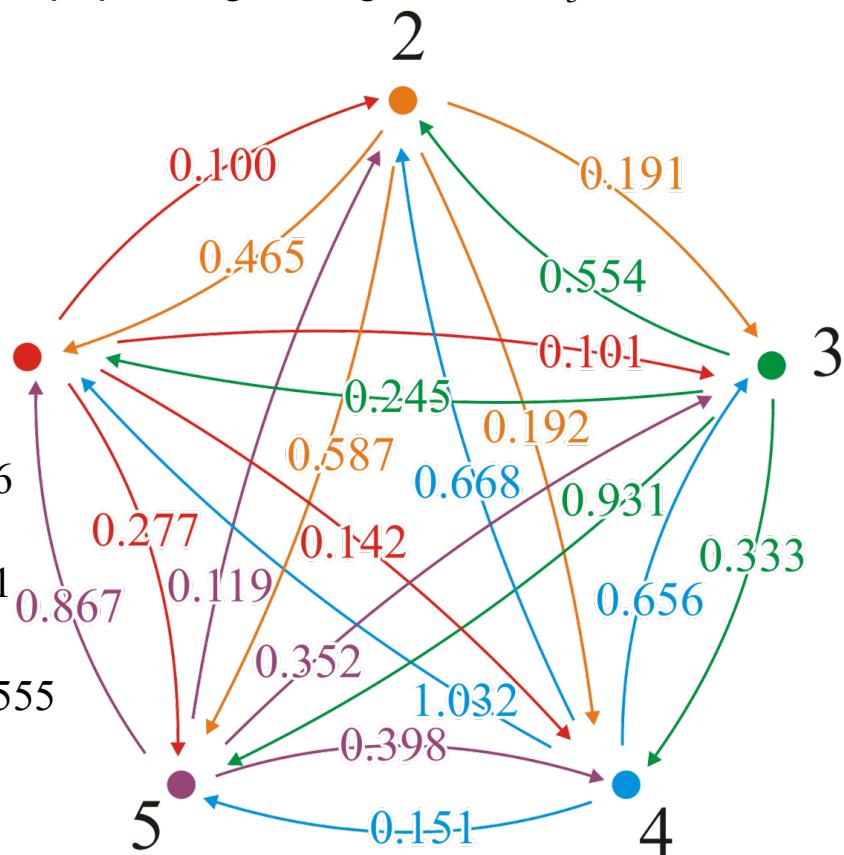
$$0.465 > 0.191 + 0.245 = 0.436$$

$$(4, 1) \rightarrow (4, 3, 1)$$

$$1.032 > 0.656 + 0.245 = 0.901$$

$$(5, 1) \rightarrow (5, 3, 1)$$

$$0.584 > 0.310 + 0.245 = 0.555$$

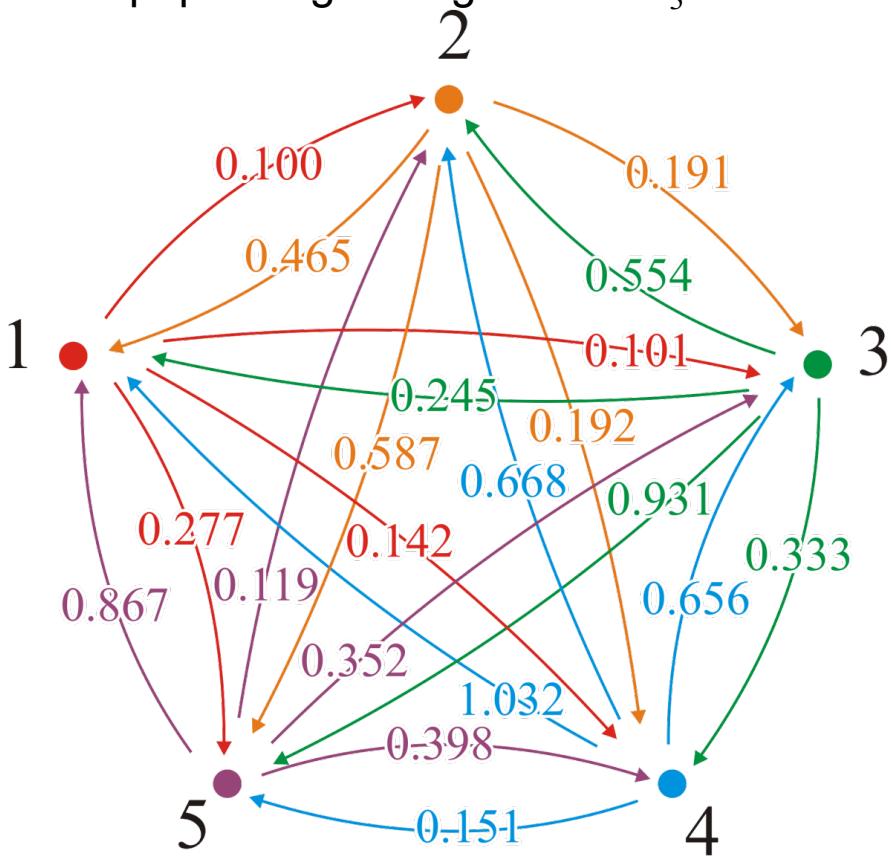


Example

With the next pass, $k = 3$, we attempt passing through vertex v_3

0	0.100	0.101	0.142	0.277
0.436	0	0.191	0.192	0.587
0.245	0.345	0	0.333	0.522
0.901	0.668	0.656	0	0.151
0.555	0.119	0.310	0.311	0

We update the table



Example

With the next pass, $k = 4$, we attempt passing through vertex v_4

0	0.100	0.101	0.142	0.277
0.436	0	0.191	0.192	0.587
0.245	0.345	0	0.333	0.522
0.901	0.668	0.656	0	0.151
0.555	0.119	0.310	0.311	0

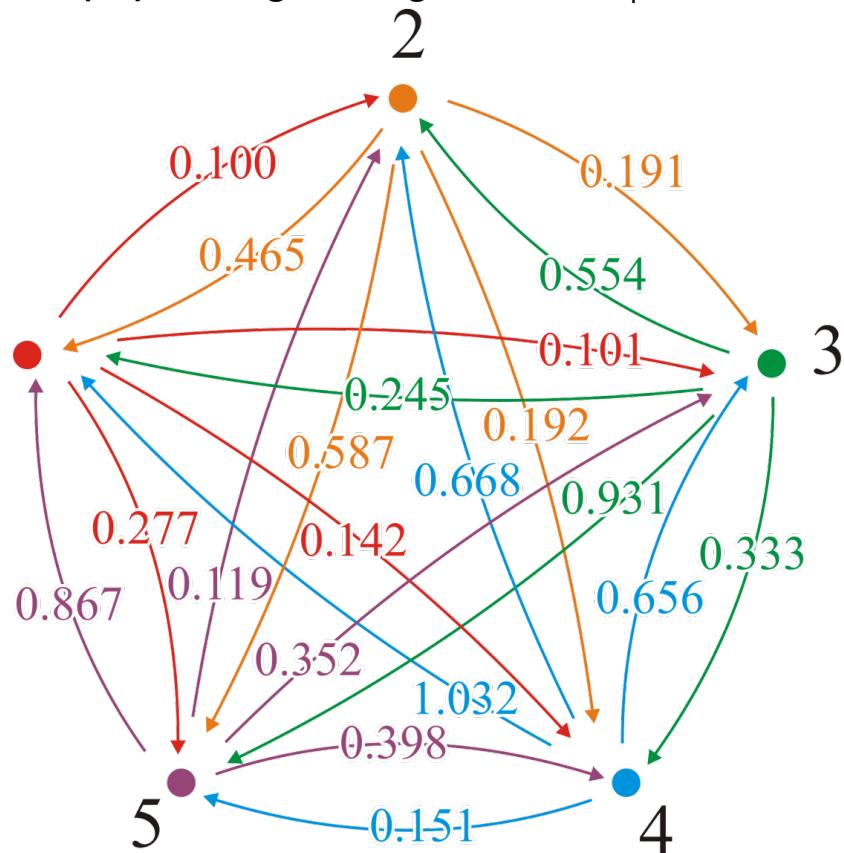
There are two shorter paths:

$$(2, 5) \rightarrow (2, 4, 5)$$

$$0.587 > 0.192 + 0.151$$

$$(3, 5) \rightarrow (3, 4, 5)$$

$$0.522 > 0.333 + 0.151$$

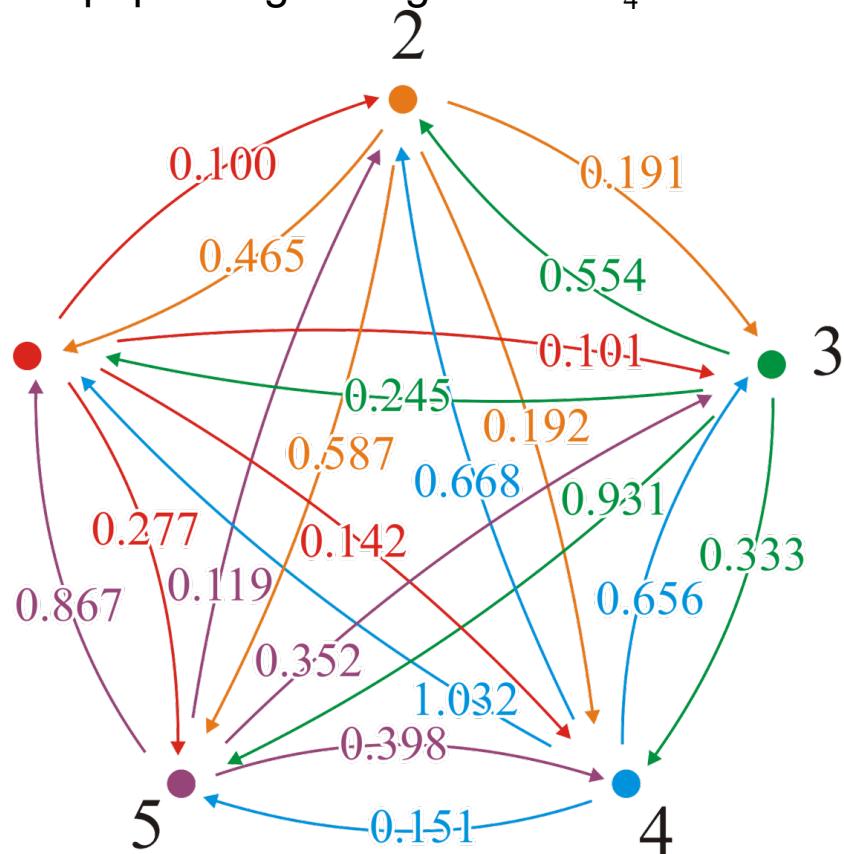


Example

With the next pass, $k = 4$, we attempt passing through vertex v_4

0	0.100	0.101	0.142	0.277
0.436	0	0.191	0.192	0.343
0.245	0.345	0	0.333	0.484
0.901	0.668	0.656	0	0.151
0.555	0.119	0.310	0.311	0

We update the table



Example

With the last pass, $k = 5$, we attempt passing through vertex v_5

0	0.100	0.101	0.142	0.277
0.436	0	0.191	0.192	0.343
0.245	0.345	0	0.333	0.484
0.901	0.668	0.656	0	0.151
0.555	0.119	0.310	0.311	0

There are three shorter paths:

$$(4, 1) \rightarrow (4, 5, 1)$$

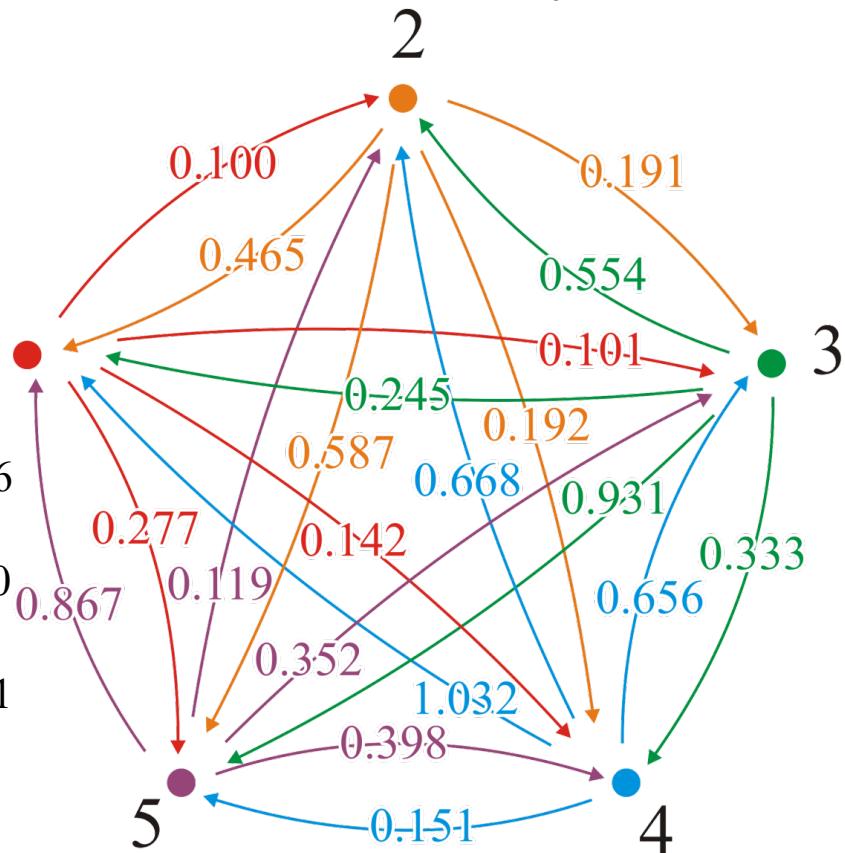
$$0.901 > 0.151 + 0.555 = 0.706$$

$$(4, 2) \rightarrow (4, 5, 2)$$

$$0.668 > 0.151 + 0.119 = 0.270$$

$$(4, 3) \rightarrow (4, 5, 3)$$

$$0.656 > 0.151 + 0.310 = 0.461$$

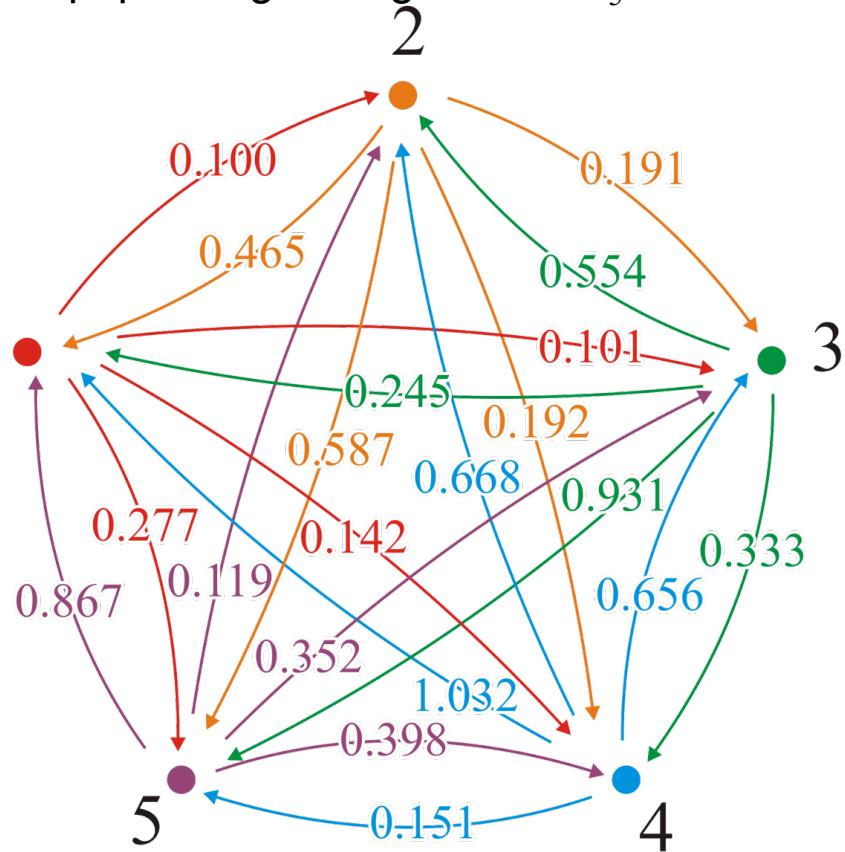


Example

With the last pass, $k = 5$, we attempt passing through vertex v_5

0	0.100	0.101	0.142	0.277
0.436	0	0.191	0.192	0.343
0.245	0.345	0	0.333	0.484
0.706	0.270	0.461	0	0.151
0.555	0.119	0.310	0.311	0

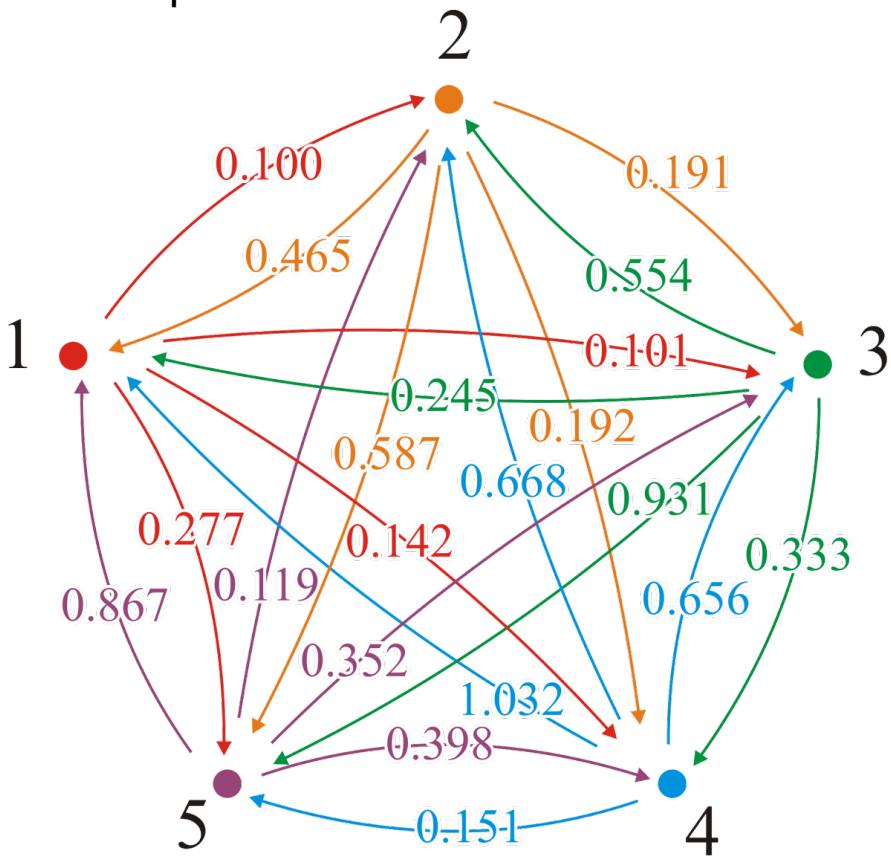
We update the table



Example

Thus, we have a table of all shortest paths:

0	0.100	0.101	0.142	0.277
0.436	0	0.191	0.192	0.343
0.245	0.345	0	0.333	0.484
0.706	0.270	0.461	0	0.151
0.555	0.119	0.310	0.311	0



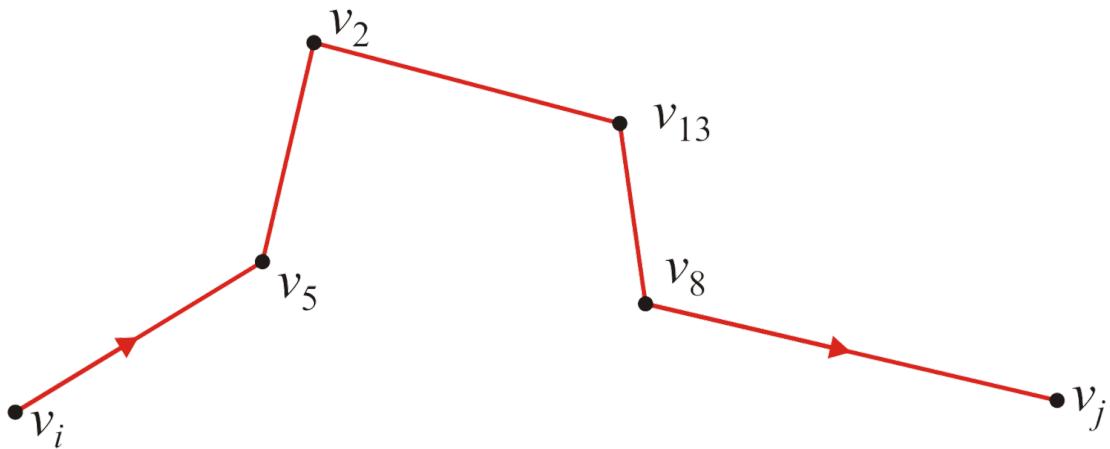
What Is the Shortest Path?

This algorithm finds the shortest distances, but what are the paths corresponding to those shortest distances?

- Recall that with Dijkstra's algorithm, we could find the shortest paths by recording the previous node
- Here we use a similar approach, but we choose to store the next node instead of the previous node

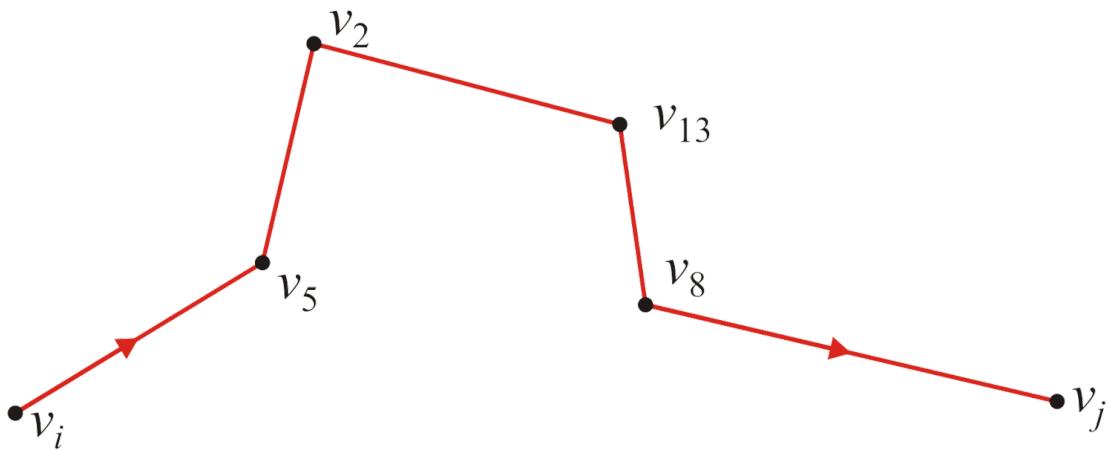
What Is the Shortest Path?

Suppose the shortest path from v_i to v_j is as follows:



What Is the Shortest Path?

Does this path consist of (v_i, v_5) and the shortest path from v_5 to v_j ?

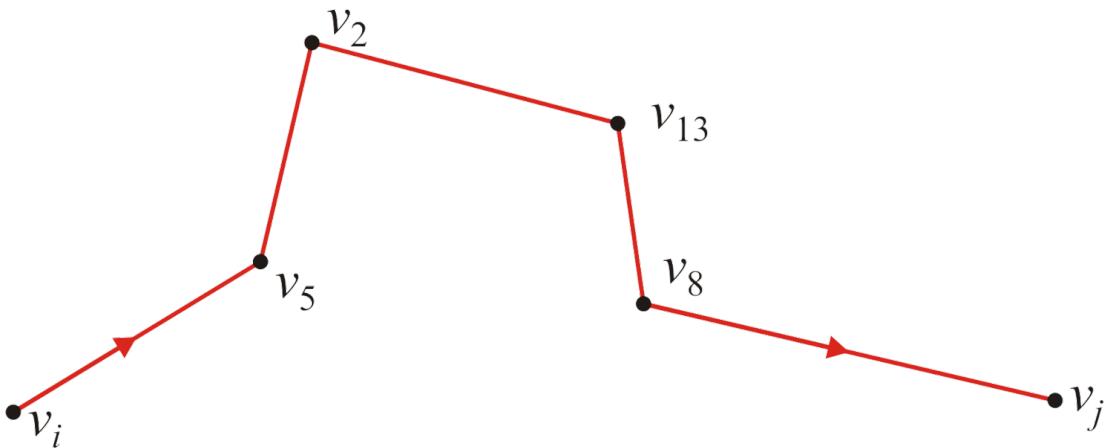


Yes

- If there was a shorter path from v_5 to v_j , then we would also find a shorter path from v_i to v_j

What Is the Shortest Path?

Does this path consist of (v_i, v_5) and the shortest path from v_5 to v_j ?

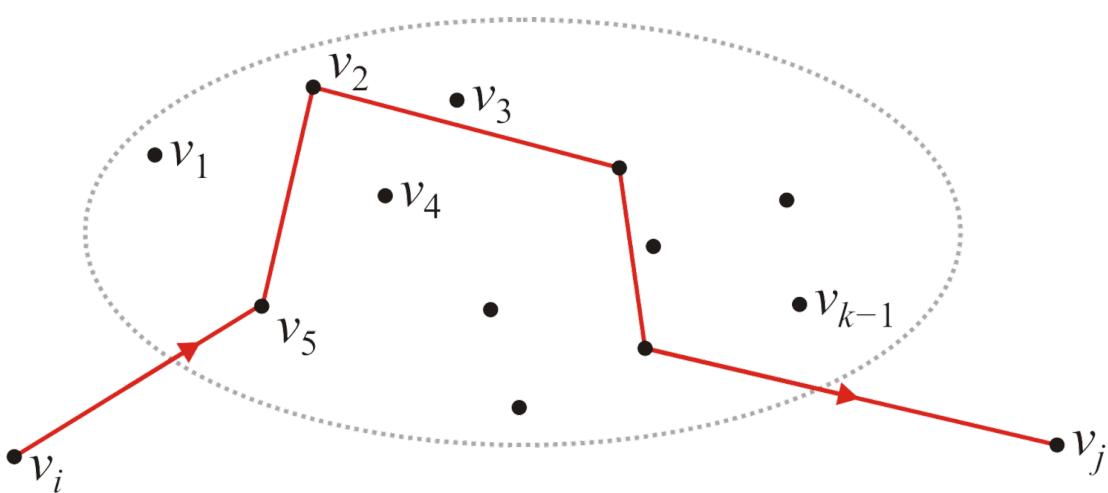


To find the shortest path from v_i to v_j , we only need to know that v_5 is the next vertex in the path — the rest of the path would be recursively recovered as the shortest path from v_5 to v_j .

What Is the Shortest Path?

Now, suppose we have the shortest path from v_i to v_j which passes through the vertices v_1, v_2, \dots, v_{k-1}

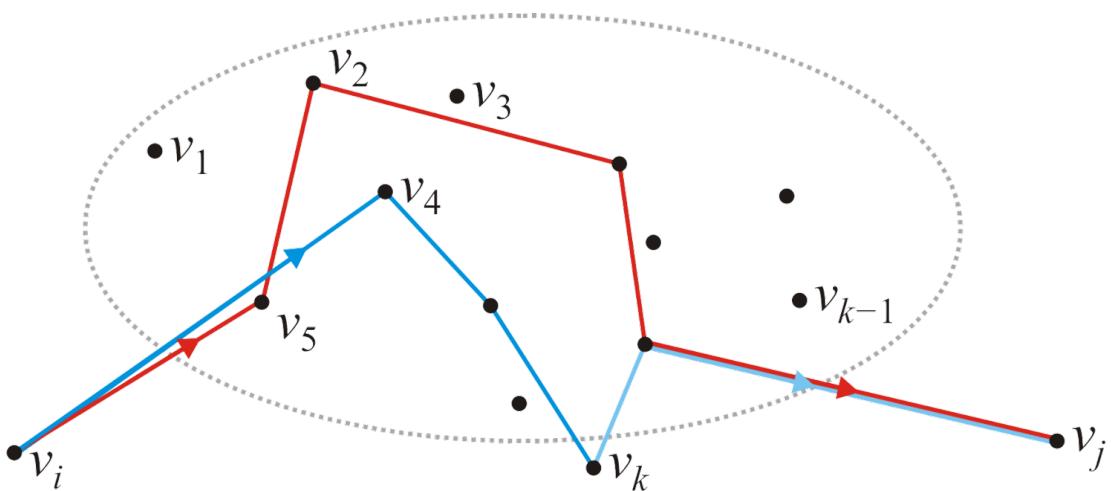
- In this example, the next vertex in the path is v_5



What Is the Shortest Path?

What if we find a shorter path passing through v_k ?

- Now the next vertex in the new path should be the next vertex in the shortest path from v_i to v_k , which is v_4 in this example



What Is the Shortest Path?

Let us store the next vertex in the shortest path. Initially:

$$p_{i,j} = \begin{cases} \emptyset & \text{If } i = j \\ j & \text{If there is an edge from } i \text{ to } j \\ \emptyset & \text{Otherwise} \end{cases}$$

What Is the Shortest Path?

When we find a shorter path, update the next node:

$$p_{i,j} = p_{i,k}$$

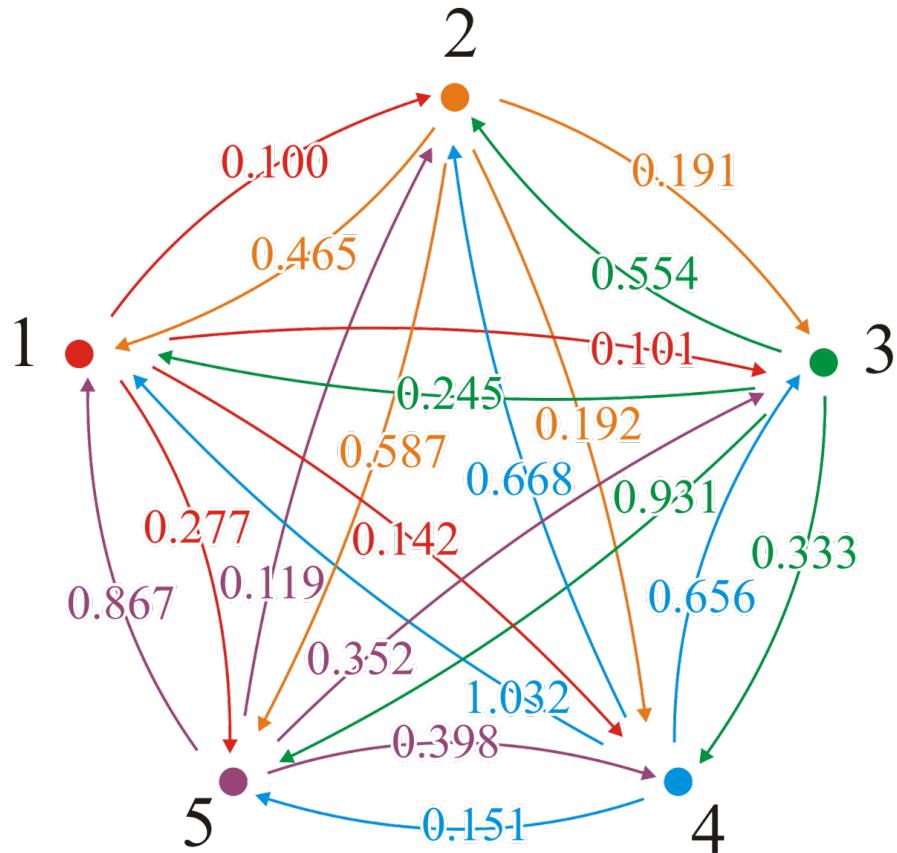
```
// Initialize the matrix p
// ...
for ( int k = 0; k < num_vertices; ++k ) {
    for ( int i = 0; i < num_vertices; ++i ) {
        for ( int j = 0; j < num_vertices; ++j ) {
            if ( d[i][j] > d[i][k] + d[k][j] ) {
                p[i][j] = p[i][k];
                d[i][j] = d[i][k] + d[k][j];
            }
        }
    }
}
```

Example

In our original example, initially, the next node is exactly that:

$$\begin{pmatrix} - & 2 & 3 & 4 & 5 \\ 1 & - & 3 & 4 & 5 \\ 1 & 2 & - & 4 & 5 \\ 1 & 2 & 3 & - & 5 \\ 1 & 2 & 3 & 4 & - \end{pmatrix}$$

This would define our matrix $\mathbf{P} = (p_{ij})$



Example

With the first pass, $k = 1$, we attempt passing through vertex v_1

-	2	3	4	5
1	-	3	4	5
1	2	-	4	5
1	2	3	-	5
1	2	3	4	-

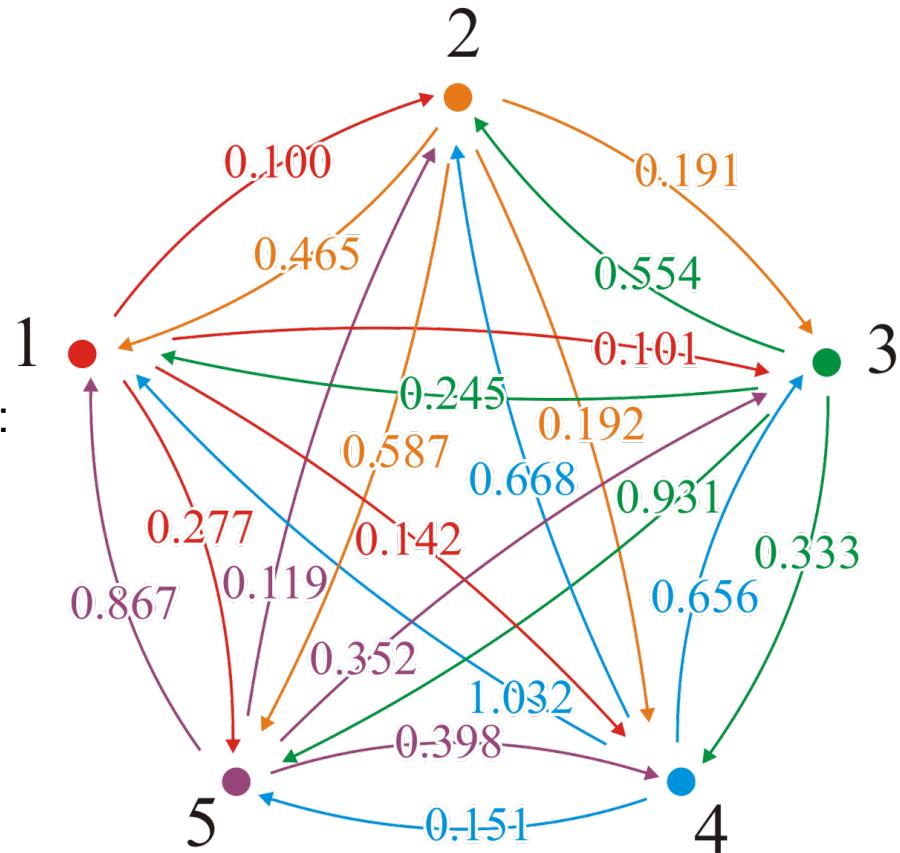
There are two shorter paths:

$$(3, 2) \rightarrow (3, 1, 2)$$

$$0.554 > 0.245 + 0.100$$

$$(3, 5) \rightarrow (3, 1, 5)$$

$$0.931 > 0.245 + 0.277$$



$$\gamma_{ij} = P_{ik}$$

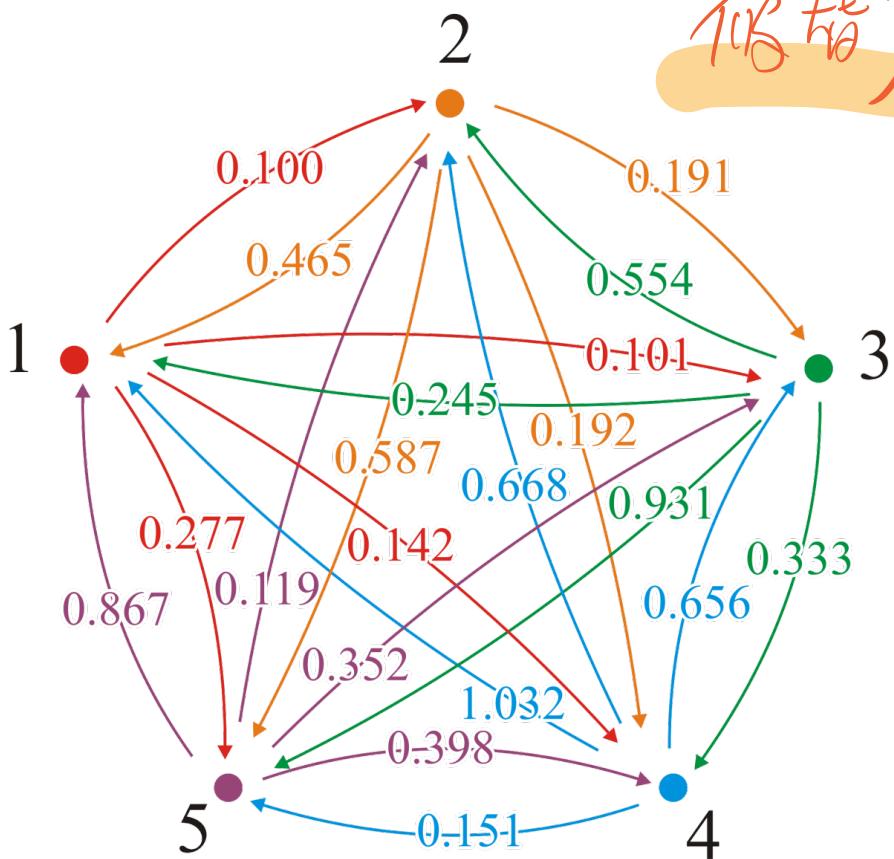
Example

(注: 下邊等於 $k \rightarrow P_{ik}$ 的總和)

With the first pass, $k = 1$, we attempt passing through vertex v_1

-	2	3	4	5
1	-	3	4	5
1	1	-	4	1
1	2	3	-	5
1	2	3	4	-

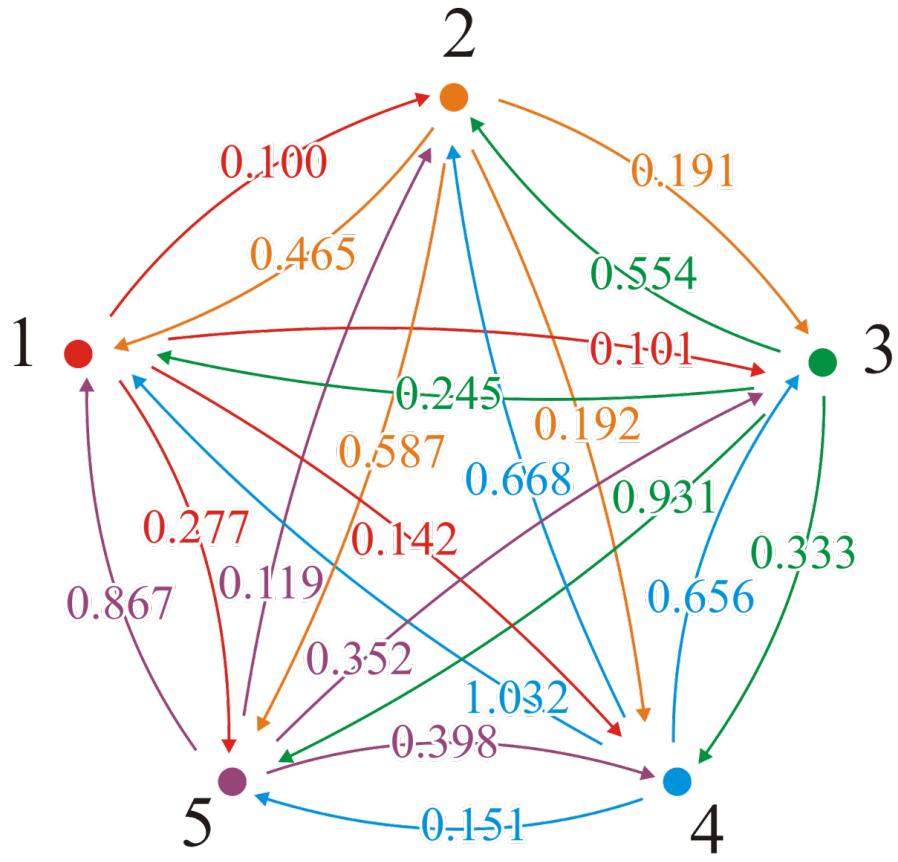
We update each of these



Example

After all the steps, we end up with the matrix $P = (p_{i,j})$:

$$\begin{pmatrix} - & 2 & 3 & 4 & 5 \\ 3 & - & 3 & 4 & 4 \\ 1 & 1 & - & 4 & 4 \\ 5 & 5 & 5 & - & 5 \\ 2 & 2 & 2 & 2 & - \end{pmatrix}$$



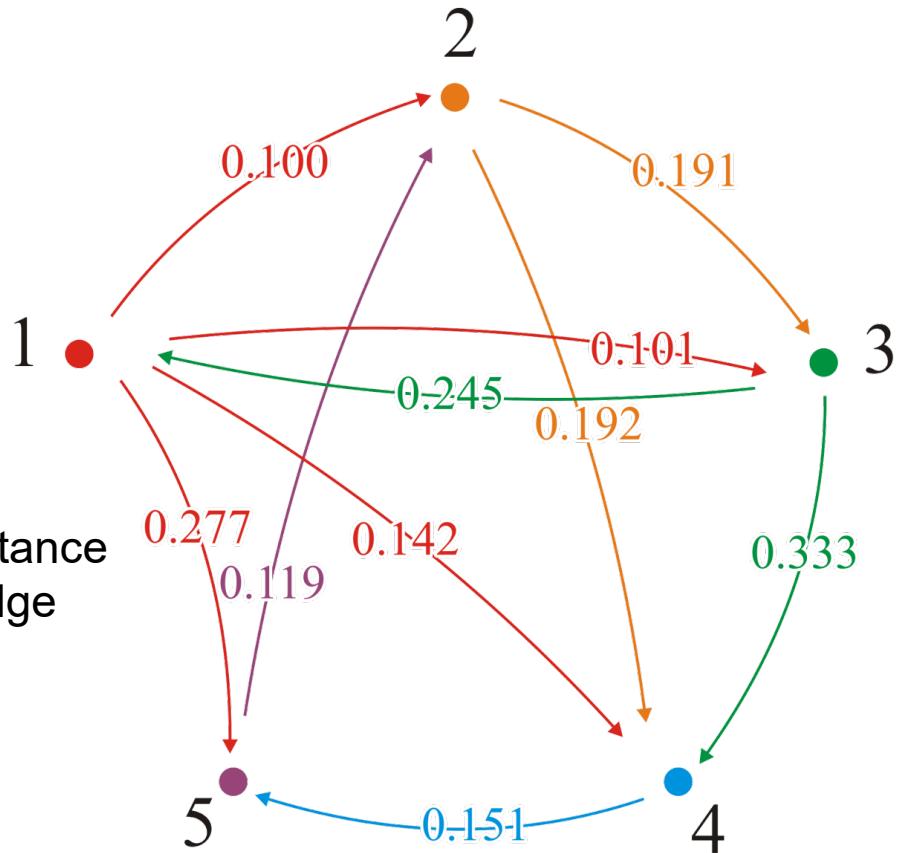
Example

These are all the adjacent edges that are still the shortest distance

-	2	3	4	5
3	-	3	4	4
1	1	-	4	4
5	5	5	-	5
2	2	2	2	-

For each of these, $p_{i,j} = j$

In all cases, the shortest distance from vertex 1 is the direct edge



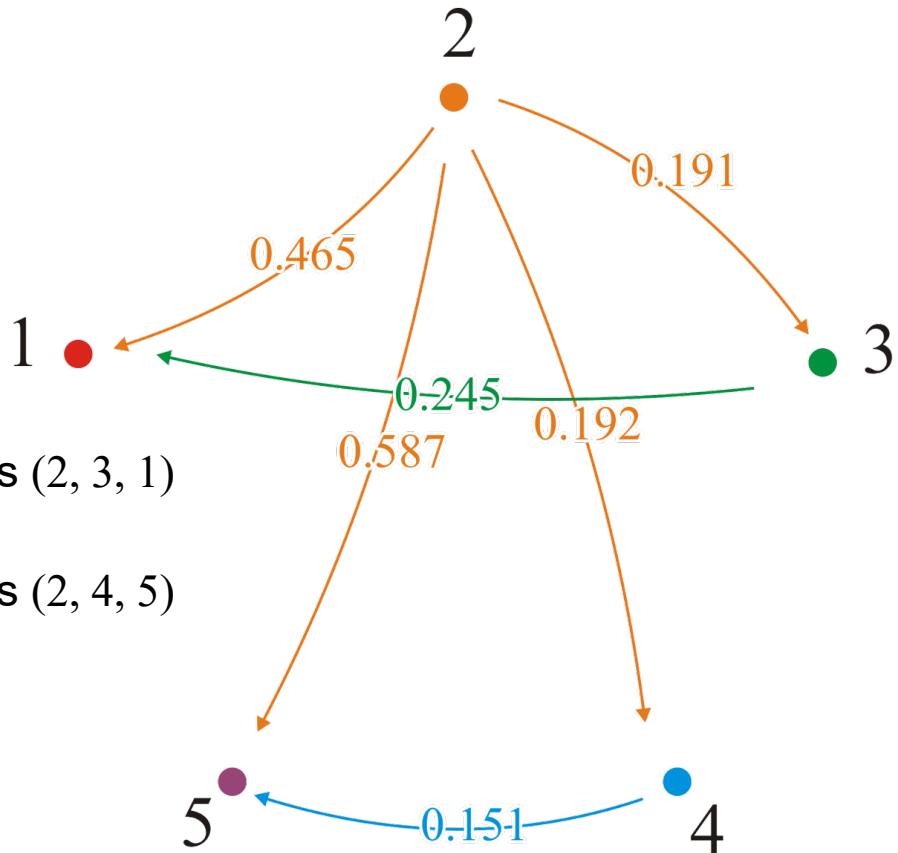
Example

From vertex v_2 , $p_{2,3} = 3$ and $p_{2,4} = 4$; we go directly to vertices v_3 and v_4

-	2	3	4	5
3	-	3	4	4
1	1	-	4	4
5	5	5	-	5
2	2	2	2	-

But $p_{2,1} = 3$ and $p_{3,1} = 1$;
the shortest path to v_1 is $(2, 3, 1)$

Also, $p_{2,5} = 4$ and $p_{4,5} = 5$;
the shortest path to v_5 is $(2, 4, 5)$



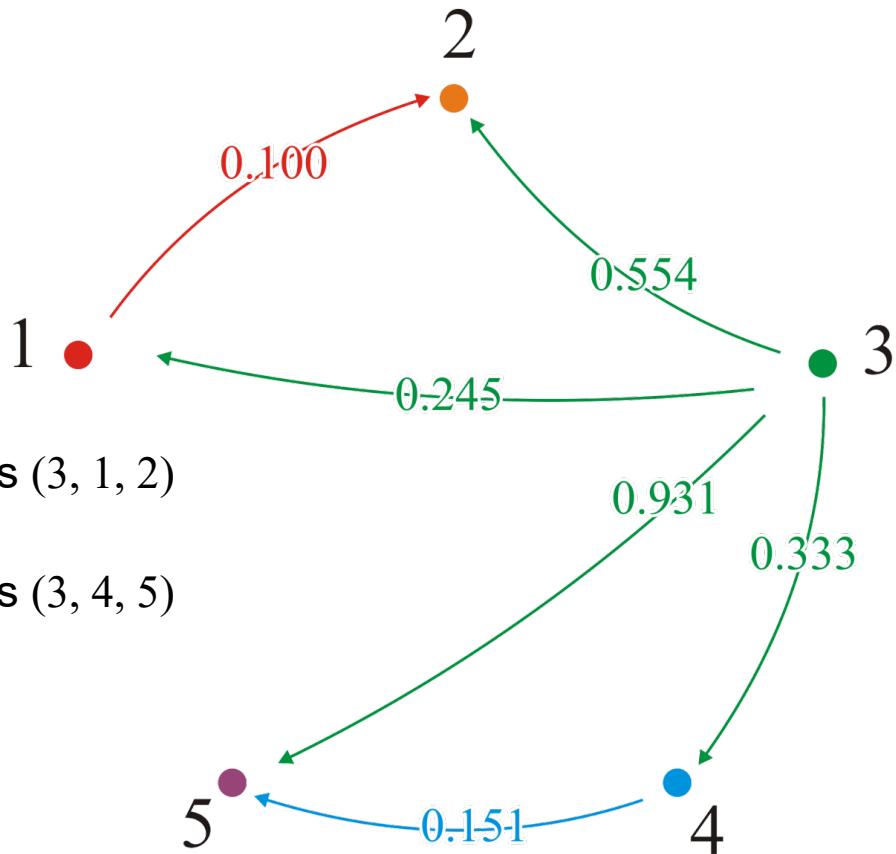
Example

From vertex v_3 , $p_{3,1} = 1$ and $p_{3,4} = 4$; we go directly to vertices v_1 and v_4

-	2	3	4	5
3	-	3	4	4
1	1	-	4	4
5	5	5	-	5
2	2	2	2	-

But $p_{3,2} = 1$ and $p_{1,2} = 2$;
the shortest path to v_2 is $(3, 1, 2)$

Also, $p_{3,5} = 4$ and $p_{4,5} = 5$;
the shortest path to v_5 is $(3, 4, 5)$

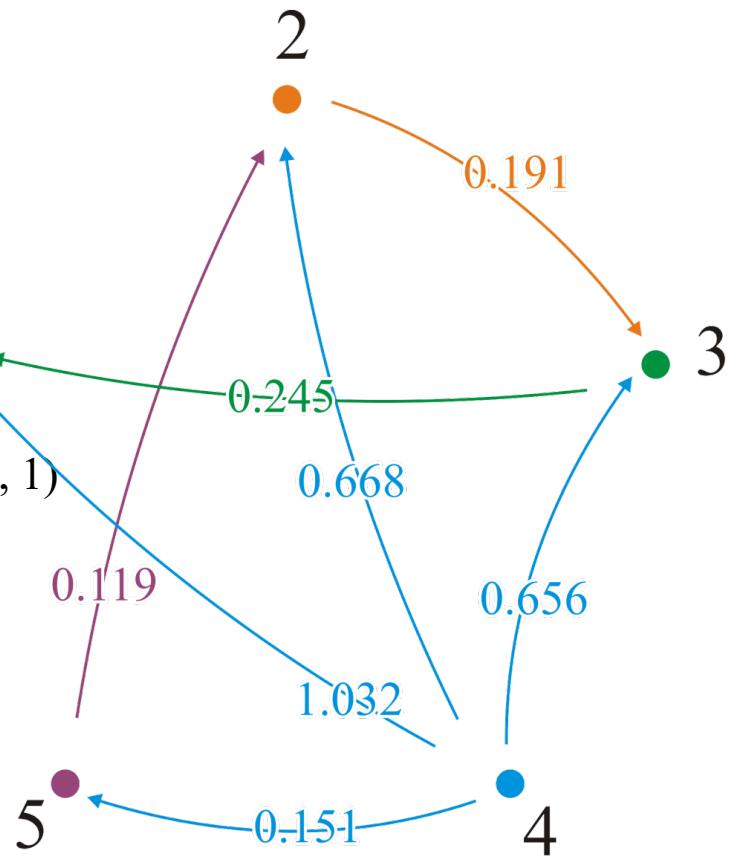


Example

From vertex v_4 , $p_{4,5} = 5$; we go directly to vertex v_5

-	2	3	4	5
3	-	3	4	4
1	1	-	4	4
5	5	5	-	5
2	2	2	2	-

But $p_{4,1} = 5$, $p_{5,1} = 2$, $p_{2,1} = 3$, $p_{3,1} = 1$;
the shortest path to v_1 is $(4, 5, 2, 3, 1)$

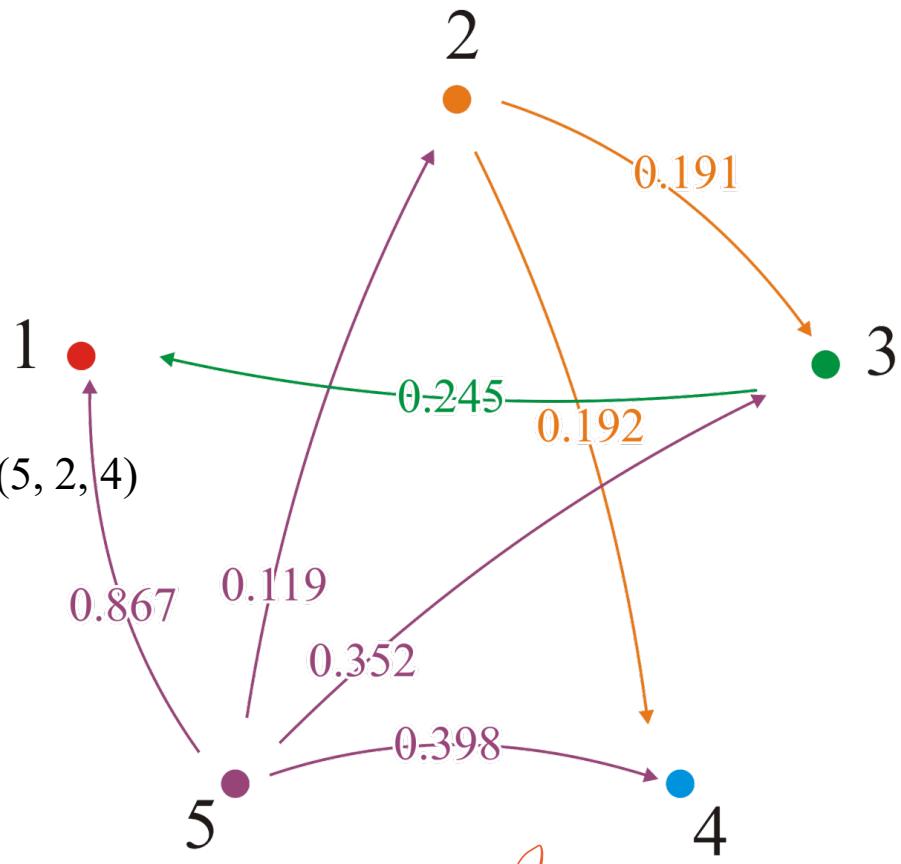


Example

From vertex v_5 , $p_{5,2} = 2$; we go directly to vertex v_2

-	2	3	4	5
3	-	3	4	4
1	1	-	4	4
5	5	5	-	5
2	2	2	2	-

But $p_{5,4} = 2$ and $p_{2,4} = 4$;
the shortest path to v_4 is $(5, 2, 4)$



Connection ~~SV~~

Which Vertices are Connected?

Finally, what if we only care if a connection exists?

- Recall that with Dijkstra's algorithm, we could find the shortest paths by recording the previous node
- In this case, can make the observation that:

A path from v_i to v_j exists if either:

A path exists through the vertices from v_1 to v_{k-1} , or

A path, through those same nodes, exists from v_i to v_k and
a path exists from v_k to v_j

Which Vertices are Connected?

The *transitive closure* is a Boolean graph:

```
bool tc[num_vertices][num_vertices];  
  
// Initialize the matrix tc: Theta(|V|^2)  
// ...  
  
// Run Floyd-Warshall  
for ( int k = 0; k < num_vertices; ++k ) {  
    for ( int i = 0; i < num_vertices; ++i ) {  
        for ( int j = 0; j < num_vertices; ++j ) {  
            tc[i][j] = tc[i][j] || (tc[i][k] && tc[k][j]);  
        }  
    }  
}
```

Bool

Q:
bit
8 00001:
取反(奇数)

Example

$i \in j$

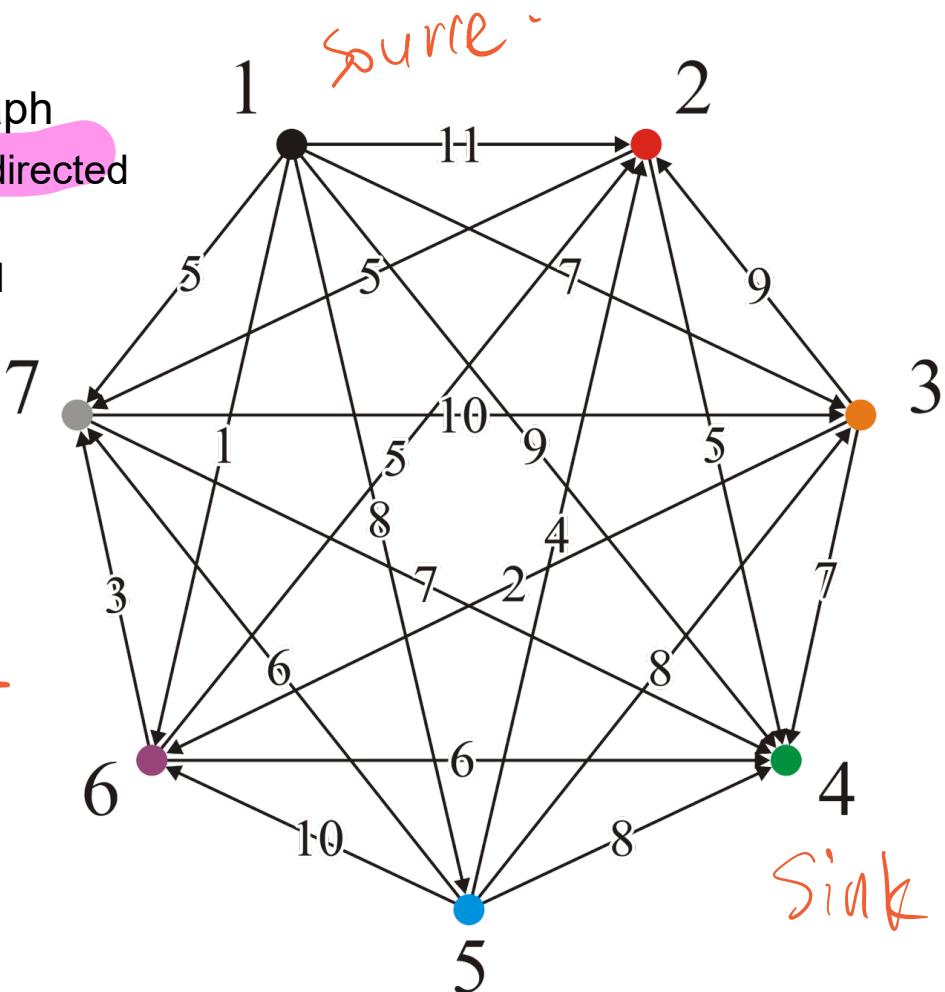
Consider this directed graph

- Each pair has only one directed edge
- Vertex v_1 is a source and v_4 is a sink

We will apply all three matrices

- Shortest distance
- Paths
- Transitive closure

TF

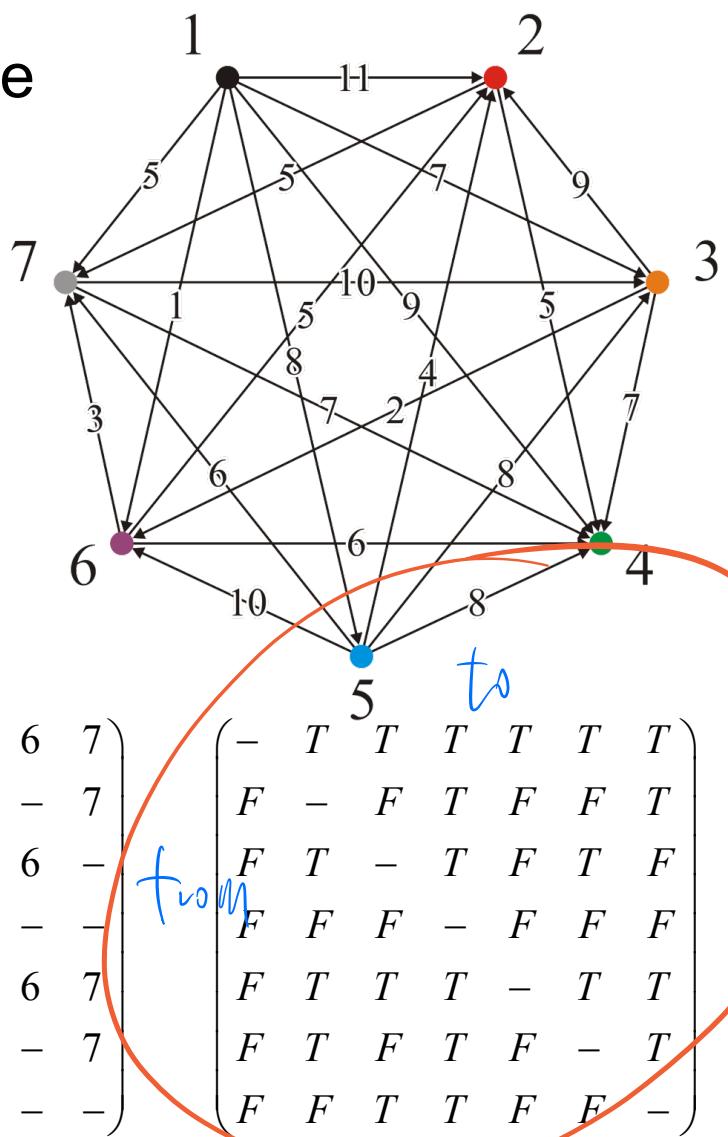


Example

We set up the three initial matrices

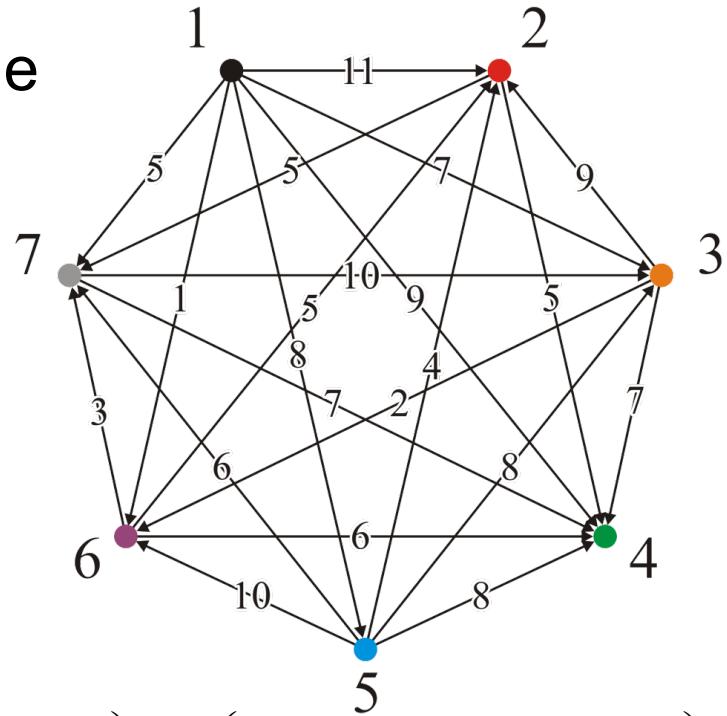
$$\begin{pmatrix} 0 & 11 & 7 & 9 & 8 & 1 & 5 \\ \infty & 0 & \infty & 5 & \infty & \infty & 5 \\ \infty & 9 & 0 & 7 & \infty & 2 & \infty \\ \infty & \infty & \infty & 0 & \infty & \infty & \infty \\ \infty & 4 & 8 & 8 & 0 & 10 & 6 \\ \infty & 5 & \infty & 6 & \infty & 0 & 3 \\ \infty & \infty & 10 & 7 & \infty & \infty & 0 \end{pmatrix}$$

$$\begin{pmatrix} - & 2 & 3 & 4 & 5 & 6 & 7 \\ - & - & - & 4 & - & - & 7 \\ - & 2 & - & 4 & - & 6 & - \\ - & - & - & - & - & - & - \\ - & 2 & 3 & 4 & - & 6 & 7 \\ - & 2 & - & 4 & - & - & 7 \\ - & - & 3 & 4 & - & - & - \end{pmatrix}$$



directed Example

At step 1, no path leads to v_1 , so no shorter paths could be found passing through v_1



$$\begin{pmatrix} 0 & 11 & 7 & 9 & 8 & 1 & 5 \\ \infty & 0 & \infty & 5 & \infty & \infty & 5 \\ \infty & 9 & 0 & 7 & \infty & 2 & \infty \\ \infty & \infty & \infty & 0 & \infty & \infty & \infty \\ \infty & 4 & 8 & 8 & 0 & 10 & 6 \\ \infty & 5 & \infty & 6 & \infty & 0 & 3 \\ \infty & \infty & 10 & 7 & \infty & \infty & 0 \end{pmatrix}$$

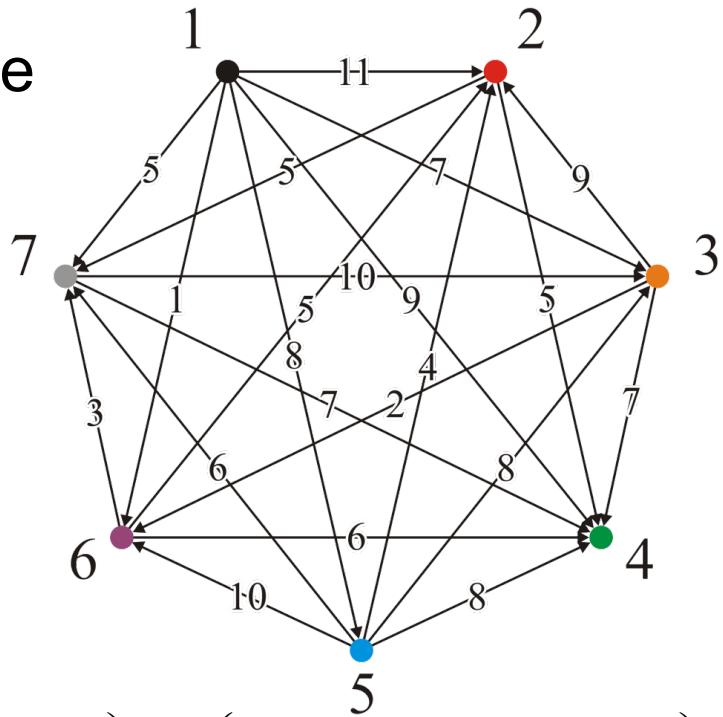
$$\begin{pmatrix} - & 2 & 3 & 4 & 5 & 6 & 7 \\ - & - & - & 4 & - & - & 7 \\ - & 2 & - & 4 & - & 6 & - \\ - & - & - & - & - & - & - \\ - & 2 & 3 & 4 & - & 6 & 7 \\ - & 2 & - & 4 & - & - & 7 \\ - & - & 3 & 4 & - & - & - \end{pmatrix}$$

$$\begin{pmatrix} - & T & T & T & T & T & T \\ F & - & F & T & F & F & T \\ F & T & - & T & F & T & F \\ F & F & F & - & F & F & F \\ F & T & T & T & - & T & T \\ F & T & F & T & F & - & T \\ F & F & T & T & F & F & - \end{pmatrix}$$

Example

At step 2, we find:

- A path $(3, 2, 7)$ of length 14



0	11	7	9	8	1	5
∞	0	∞	5	∞	∞	5
∞	9	0	7	∞	2	∞
∞	∞	∞	0	∞	∞	∞
∞	4	8	8	0	10	6
∞	5	∞	6	∞	0	3
∞	∞	10	7	∞	∞	0

-	2	3	4	5	6	7
-	-	-	4	-	-	7
-	2	-	4	-	6	-
-	-	-	-	-	-	-
-	2	3	4	-	6	7
-	2	-	4	-	-	7
-	-	3	4	-	-	-

-	T	T	T	T	T	T
F	-	F	T	F	F	T
F	T	-	T	F	T	F
F	F	F	-	F	F	F
F	T	T	T	-	T	T
F	T	F	T	F	-	T
F	F	T	T	F	F	-

i 2 j
 $b \uparrow$

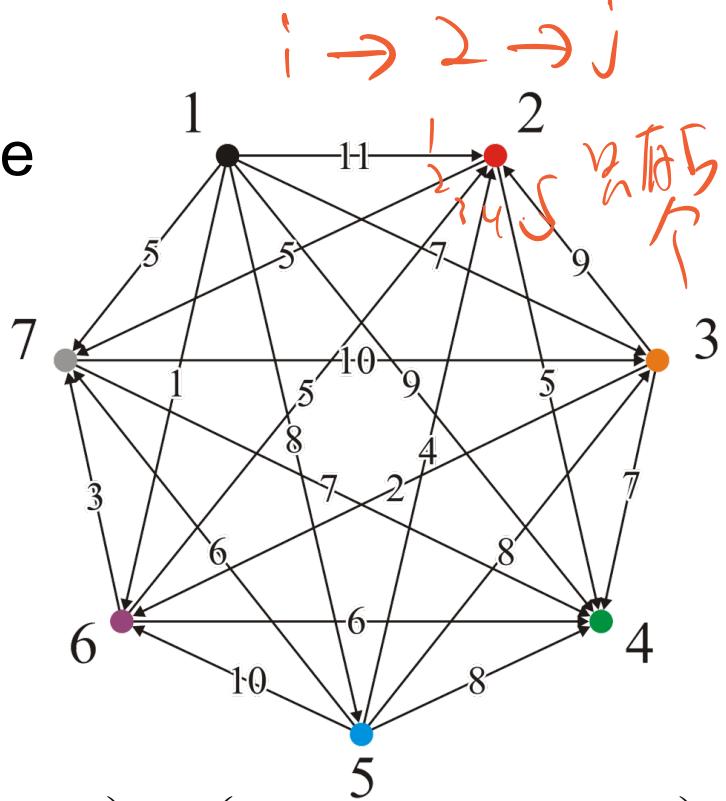
Example

At step 2, we find:

- A path $(3, 2, 7)$ of length 14

We update

~~direct~~ $d_{3,7} = 14$, $p_{3,7} = 2$ and $c_{3,7} = T$



$$\begin{pmatrix} 0 & 11 & 7 & 9 & 8 & 1 & 5 \\ \infty & 0 & \infty & 5 & \infty & \infty & 5 \\ \infty & 9 & 0 & 7 & \infty & 2 & 14 \\ \infty & \infty & \infty & 0 & \infty & \infty & \infty \\ \infty & 4 & 8 & 8 & 0 & 10 & 6 \\ \infty & 5 & \infty & 6 & \infty & 0 & 3 \\ \infty & \infty & 10 & 7 & \infty & \infty & 0 \end{pmatrix}$$

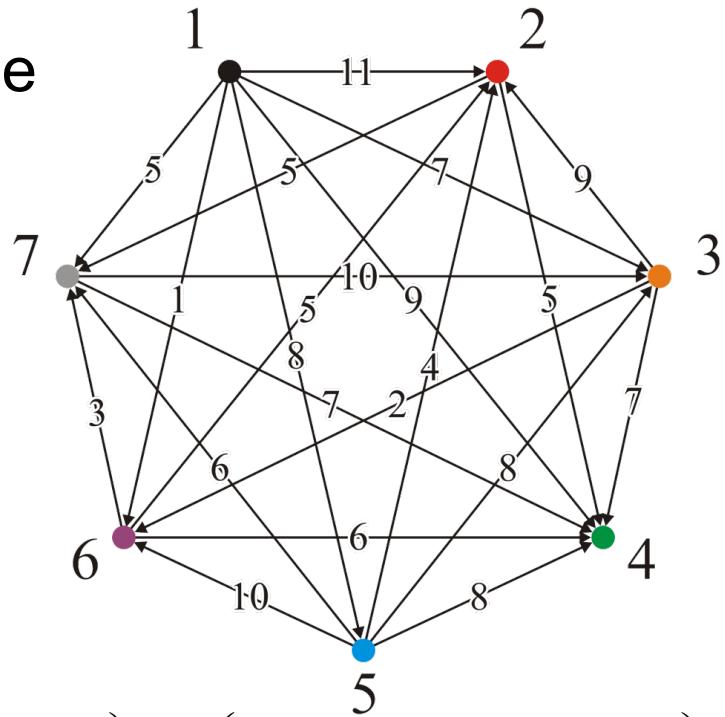
$$\begin{pmatrix} - & 2 & 3 & 4 & 5 & 6 & 7 \\ - & - & - & 4 & - & - & 7 \\ - & 2 & - & 4 & - & 6 & 2 \\ - & - & - & - & - & - & - \\ - & 2 & 3 & 4 & - & 6 & 7 \\ - & 2 & - & 4 & - & - & 7 \\ - & - & 3 & 4 & - & - & - \end{pmatrix}$$

$$\begin{pmatrix} - & T & T & T & T & T & T \\ F & - & F & T & F & F & T \\ F & T & - & T & F & T & T \\ F & F & F & - & F & F & F \\ F & T & T & T & - & T & T \\ F & T & F & T & F & - & T \\ F & F & T & T & F & F & - \end{pmatrix}$$

Example

At step 3, we find:

- A path $(7, 3, 2)$ of length 19
- A path $(7, 3, 6)$ of length 12



$$\begin{pmatrix} 0 & 11 & 7 & 9 & 8 & 1 & 5 \\ \infty & 0 & \infty & 5 & \infty & \infty & 5 \\ \infty & 9 & 0 & 7 & \infty & 2 & 14 \\ \infty & \infty & \infty & 0 & \infty & \infty & \infty \\ \infty & 4 & 8 & 8 & 0 & 10 & 6 \\ \infty & 5 & \infty & 6 & \infty & 0 & 3 \\ \infty & \infty & 10 & 7 & \infty & \infty & 0 \end{pmatrix}$$

$$\begin{pmatrix} - & 2 & 3 & 4 & 5 & 6 & 7 \\ - & - & - & 4 & - & - & 7 \\ - & 2 & - & 4 & - & 6 & 2 \\ - & - & - & - & - & - & - \\ - & 2 & 3 & 4 & - & 6 & 7 \\ - & 2 & - & 4 & - & - & 7 \\ - & - & 3 & 4 & - & - & - \end{pmatrix}$$

$$\begin{pmatrix} - & T & T & T & T & T & T \\ F & - & F & T & F & F & T \\ F & T & - & T & F & T & T \\ F & F & F & - & F & F & F \\ F & T & T & T & - & T & T \\ F & T & F & T & F & - & T \\ F & F & T & T & F & F & - \end{pmatrix}$$

Example

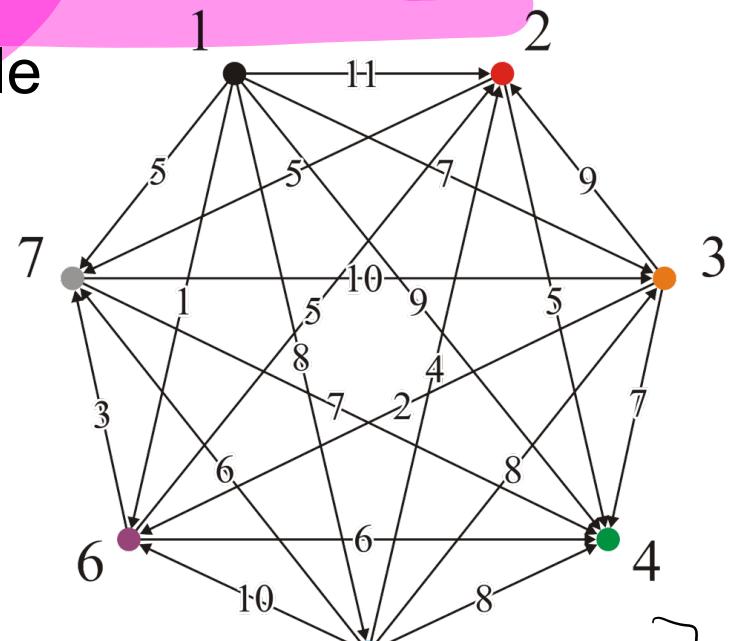
At step 3, we find:

- A path $(7, 3, 2)$ of length 19
- A path $(7, 3, 6)$ of length 12

We update

$$d_{7,2} = 19, p_{7,2} = 3 \text{ and } c_{7,2} = T$$

$$d_{7,6} = 12, p_{7,6} = 3 \text{ and } c_{7,6} = T$$



0	11	7	9	8	1	5
∞	0	∞	5	∞	∞	5
∞	9	0	7	∞	2	14
∞	∞	∞	0	∞	∞	∞
∞	4	8	8	0	10	6
∞	5	∞	6	∞	0	3
∞	19	10	7	∞	12	0

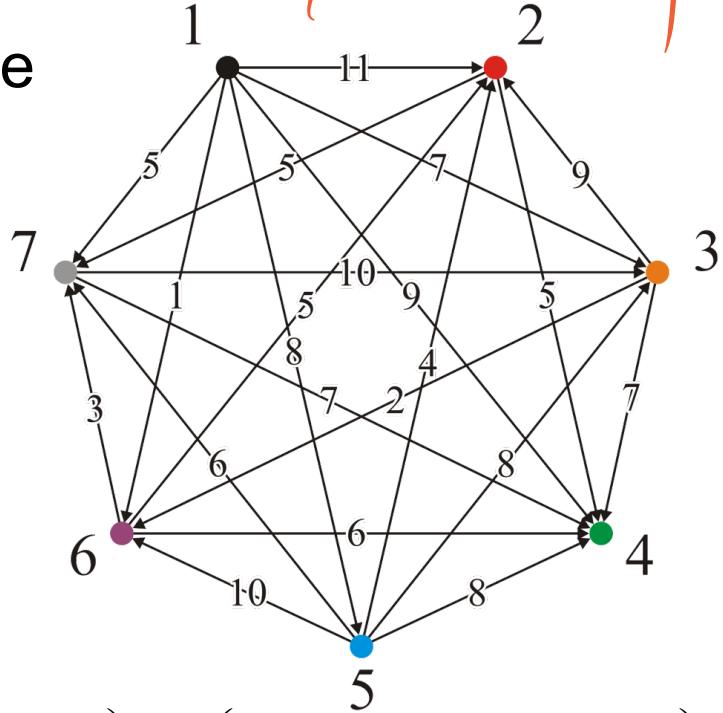
-	2	3	4	5	6	7
-	-	-	4	-	-	7
-	2	-	4	-	6	2
-	-	-	-	-	-	-
-	2	3	4	-	6	7
-	2	-	4	-	-	7
-	3	3	4	-	3	-

-	T	T	T	T	T	T
F	-	F	T	F	F	T
F	T	-	T	F	T	T
F	F	F	-	F	F	F
F	T	T	T	-	T	T
F	T	F	T	F	-	T
F	T	T	T	F	T	-

from 7 to 2 $\downarrow \rightarrow T$

Example

At step 4, there are no paths out of vertex v_4 , so we are finished



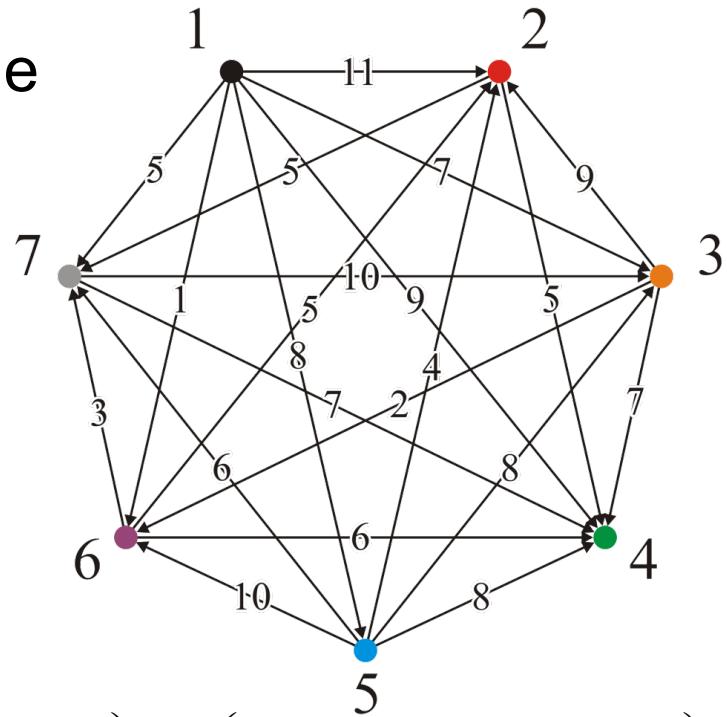
$$\begin{pmatrix} 0 & 11 & 7 & 9 & 8 & 1 & 5 \\ \infty & 0 & \infty & 5 & \infty & \infty & 5 \\ \infty & 9 & 0 & 7 & \infty & 2 & 14 \\ \infty & \infty & \infty & 0 & \infty & \infty & \infty \\ \infty & 4 & 8 & 8 & 0 & 10 & 6 \\ \infty & 5 & \infty & 6 & \infty & 0 & 3 \\ \infty & 19 & 10 & 7 & \infty & 12 & 0 \end{pmatrix}$$

$$\begin{pmatrix} - & 2 & 3 & 4 & 5 & 6 & 7 \\ - & - & - & 4 & - & - & 7 \\ - & 2 & - & 4 & - & 6 & 2 \\ - & - & - & - & - & - & - \\ - & 2 & 3 & 4 & - & 6 & 7 \\ - & 2 & - & 4 & - & - & 7 \\ - & 3 & 3 & 4 & - & 3 & - \end{pmatrix}$$

$$\begin{pmatrix} - & T & T & T & T & T & T \\ F & - & F & T & F & F & T \\ F & T & - & T & F & T & T \\ F & F & F & - & F & F & F \\ F & T & T & T & - & T & T \\ F & T & F & T & F & - & T \\ F & T & T & T & F & T & - \end{pmatrix}$$

Example

At step 5, there is one incoming edge from v_1 to v_5 , and it doesn't make any paths out of vertex v_1 any shorter...



$$\begin{pmatrix} 0 & 11 & 7 & 9 & 8 & 1 & 5 \\ \infty & 0 & \infty & 5 & \infty & \infty & 5 \\ \infty & 9 & 0 & 7 & \infty & 2 & 14 \\ \infty & \infty & \infty & 0 & \infty & \infty & \infty \\ \infty & 4 & 8 & 8 & 0 & 10 & 6 \\ \infty & 5 & \infty & 6 & \infty & 0 & 3 \\ \infty & 19 & 10 & 7 & \infty & 12 & 0 \end{pmatrix}$$

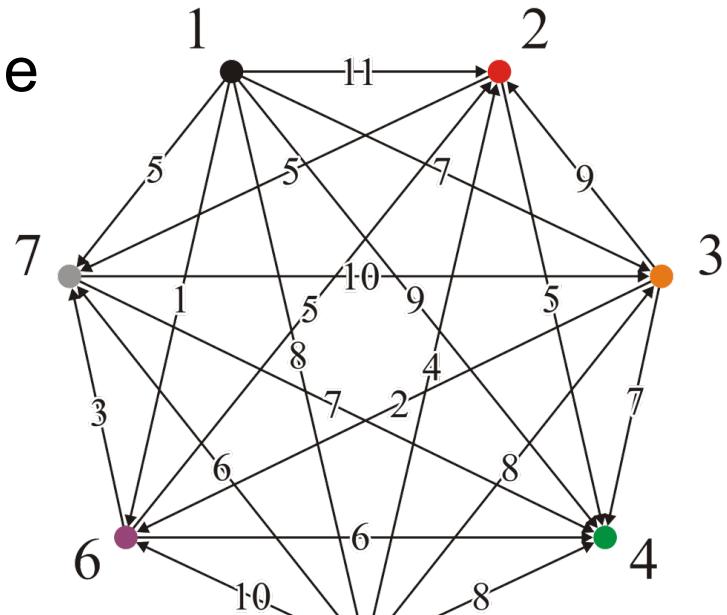
$$\begin{pmatrix} - & 2 & 3 & 4 & 5 & 6 & 7 \\ - & - & - & 4 & - & - & 7 \\ - & 2 & - & 4 & - & 6 & 2 \\ - & - & - & - & - & - & - \\ - & 2 & 3 & 4 & - & 6 & 7 \\ - & 2 & - & 4 & - & - & 7 \\ - & 3 & 3 & 4 & - & 3 & - \end{pmatrix}$$

$$\begin{pmatrix} - & T & T & T & T & T & T \\ F & - & F & T & F & F & T \\ F & T & - & T & F & T & T \\ F & F & F & - & F & F & F \\ F & T & T & T & - & T & T \\ F & T & F & T & F & - & T \\ F & T & T & T & F & T & - \end{pmatrix}$$

Example

At step 6, we find:

- A path $(1, 6, 2)$ of length 6
- A path $(1, 6, 4)$ of length 7
- A path $(1, 6, 7)$ of length 4
- A path $(3, 6, 2)$ of length 7
- A path $(3, 6, 7)$ of length 5
- A path $(7, 3, 6, 2)$ of length 17



0	11	7	9	8	1	5
∞	0	∞	5	∞	∞	5
∞	9	0	7	∞	2	14
∞	∞	∞	0	∞	∞	∞
∞	4	8	8	0	10	6
∞	5	∞	6	∞	0	3
∞	19	10	7	∞	12	0

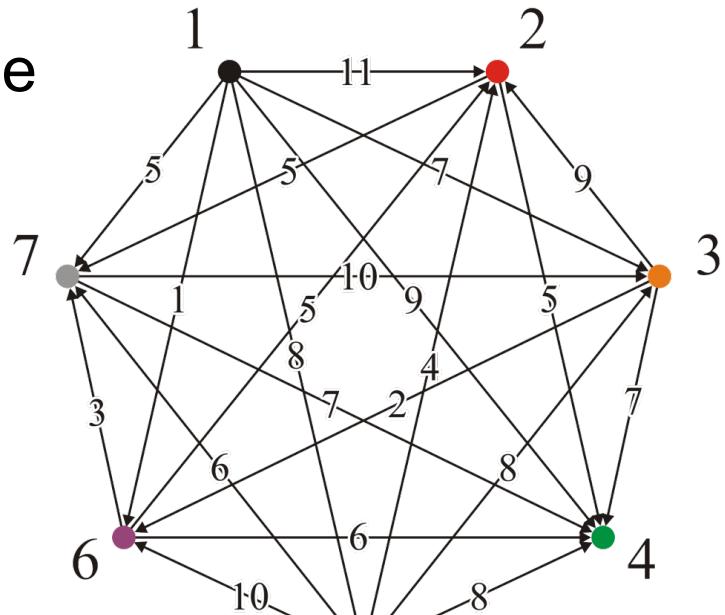
-	2	3	4	5	6	7
-	-	-	4	-	-	7
-	2	-	4	-	6	2
-	-	-	-	-	-	-
-	2	3	4	-	6	7
-	2	-	4	-	-	7
-	3	3	4	-	3	-

-	T	T	T	T	T	T
F	-	F	T	F	F	T
F	T	-	T	F	T	T
F	F	F	-	F	F	F
F	T	T	T	-	T	T
F	T	F	T	F	-	T
F	T	T	T	T	F	T

Example

At step 6, we find:

- A path $(1, 6, 2)$ of length 6
- A path $(1, 6, 4)$ of length 7
- A path $(1, 6, 7)$ of length 4
- A path $(3, 6, 2)$ of length 7
- A path $(3, 6, 7)$ of length 5
- A path $(7, 3, 6, 2)$ of length 17



0	6	7	7	8	1	4
∞	0	∞	5	∞	∞	5
∞	7	0	7	∞	2	5
∞	∞	∞	0	∞	∞	∞
∞	4	8	8	0	10	6
∞	5	∞	6	∞	0	3
∞	17	10	7	∞	12	0

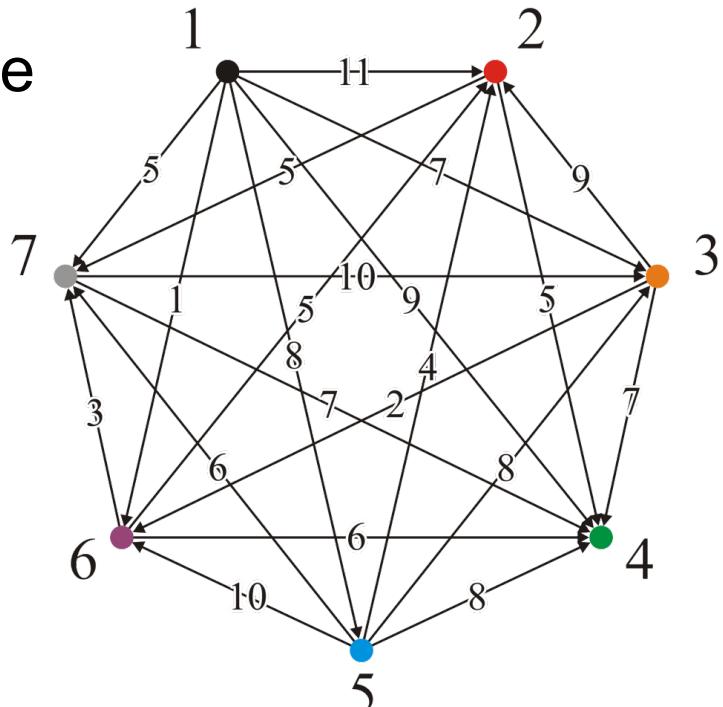
-	6	3	6	5	6	6
-	-	-	4	-	-	7
-	6	-	4	-	6	6
-	-	-	-	-	-	-
-	2	3	4	-	6	7
-	2	-	4	-	-	7
-	3	3	4	-	3	-

-	T	T	T	T	T	T
F	-	F	T	F	F	T
F	T	-	T	F	T	T
F	F	F	-	F	F	F
F	T	T	T	-	T	T
F	T	F	T	F	-	T
F	T	T	T	T	F	T

Example

At step 7, we find:

- A path $(2, 7, 3)$ of length 15
- A path $(2, 7, 6)$ of length 17
- A path $(6, 7, 3)$ of length 13



0	6	7	7	8	1	4
∞	0	∞	5	∞	∞	5
∞	7	0	7	∞	2	5
∞	∞	∞	0	∞	∞	∞
∞	4	8	8	0	10	6
∞	5	∞	6	∞	0	3
∞	17	10	7	∞	12	0

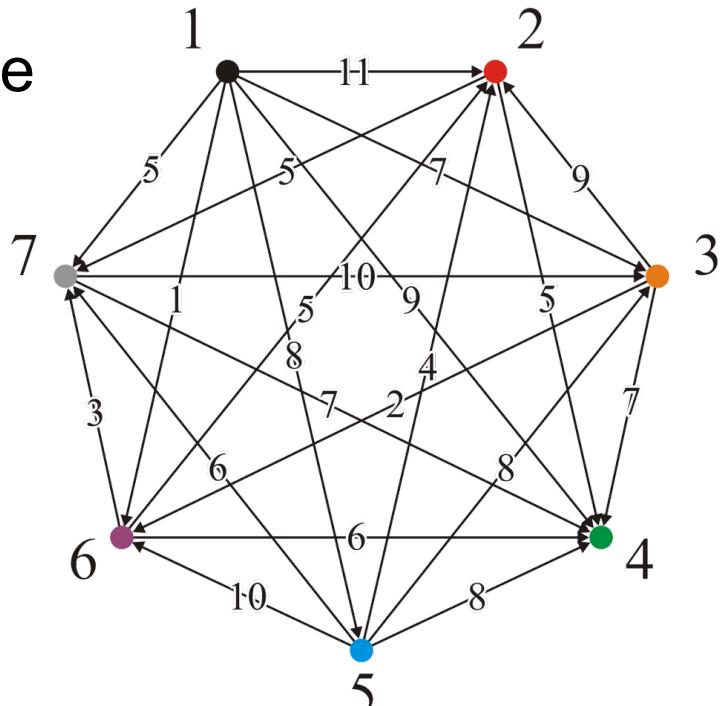
-	6	3	6	5	6	6
-	-	-	4	-	-	7
-	6	-	4	-	6	6
-	-	-	-	-	-	-
-	2	3	4	-	6	7
-	2	-	4	-	-	7
-	3	3	4	-	3	-

-	T	T	T	T	T	T
F	-	F	T	F	F	T
F	T	-	T	F	T	T
F	F	F	-	F	F	F
F	T	T	T	-	T	T
F	T	F	T	F	-	T
F	T	T	T	F	T	-

Example

Finally, at step 7, we find:

- A path $(2, 7, 3)$ of length 15
- A path $(2, 7, 6)$ of length 17
- A path $(6, 7, 3)$ of length 13



0	6	7	7	8	1	4
∞	0	15	5	∞	17	5
∞	7	0	7	∞	2	5
∞	∞	∞	0	∞	∞	∞
∞	4	8	8	0	10	6
∞	5	13	6	∞	0	3
∞	17	10	7	∞	12	0

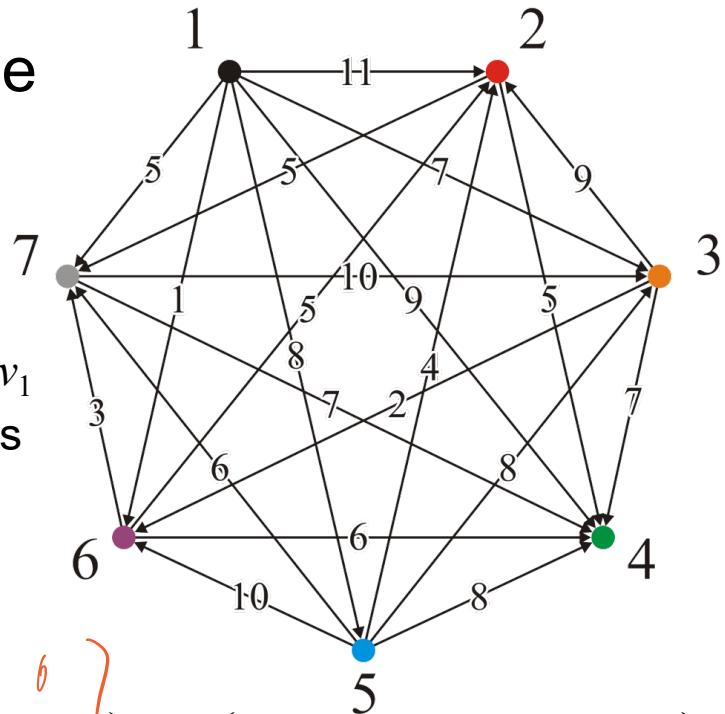
-	6	3	6	5	6	6
-	-	7	4	-	7	7
-	6	-	4	-	6	6
-	-	-	-	-	-	-
-	2	3	4	-	6	7
-	2	7	4	-	-	7
-	3	3	4	-	3	-

-	T	T	T	T	T	T
F	-	T	T	F	T	T
F	T	-	T	F	T	T
F	F	F	-	F	F	F
F	T	T	T	-	T	T
F	T	T	T	F	-	T
F	T	T	T	F	T	-

Example

Note that:

- From v_1 we can go anywhere
- From v_5 we can go anywhere but v_1
- We go between any of the vertices in the set $\{v_2, v_3, v_6, v_7\}$
- We can't go anywhere from v_4



0	6	7	7	8	1	4	1	2	3	4	5	6
∞	0	15	5	∞	17	5	2	—	7	4	—	7
∞	7	0	7	∞	2	5	3	—	6	—	4	—
∞	∞	∞	0	∞	∞	∞	4	—	—	—	—	—
∞	4	8	8	0	10	6	5	2	3	4	—	6
∞	5	13	6	∞	0	3	6	2	7	4	—	—
∞	17	10	7	∞	12	0	7	3	3	4	—	3

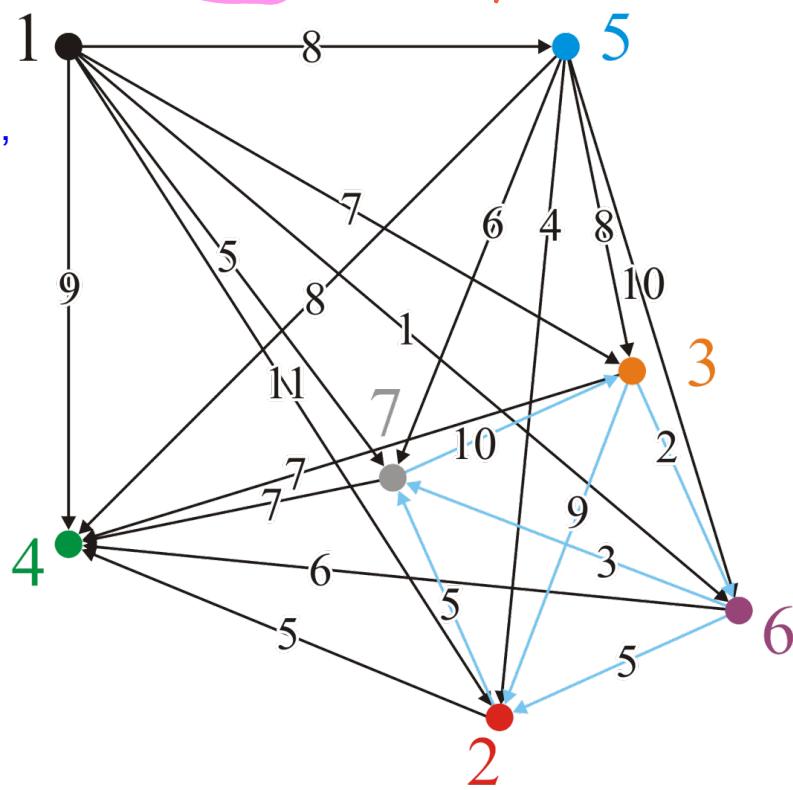
Strongly

Example

Conn OcteU -

We could reinterpret this graph as follows:

- Vertices $\{v_2, v_3, v_6, v_7\}$ form a *strongly connected* subgraph
- You can get from any one vertex to any other
- With the transitive closure graph, it is much faster finding such strongly connected components

$$\begin{pmatrix} 0 & 6 & 7 & 7 & 8 & 1 & 4 \\ \infty & 0 & 15 & 5 & \infty & 17 & 5 \\ \infty & 7 & 0 & 7 & \infty & 2 & 5 \\ \infty & \infty & \infty & 0 & \infty & \infty & \infty \\ \infty & 4 & 8 & 8 & 0 & 10 & 6 \\ \infty & 5 & 13 & 6 & \infty & 0 & 3 \\ \infty & 17 & 10 & 7 & \infty & 12 & 0 \end{pmatrix}$$


Algorithm 1 Floyd-Warshall

```

for i = 1 to n do
    for j = 1 to n do
        A[i, j, 0] = wij
        P[i, j] = -1
    end for
end for
for k = 1 to n do
    for i = 1 to n do
        for j = 1 to n do
            A[i, j, k] = A[i, j, k - 1]
            if A[i, j, k] > A[i, k, k - 1] + A[k, j, k - 1] then
                A[i, j, k] = A[i, k, k - 1] + A[k, j, k - 1]
                P[i, j] = k
            end if
        end for
    end for
end for

```

to

P	1	2	3	4	5	6	7
1	-1	5	4	-1	4	4	-1
2	5	-1	5	5	-1	5	-1
3	4	5	-1	-1	-1	-1	6
4	-1	5	-1	-1	3	3	1
5	4	-1	-1	3	-1	3	6
6	4	5	-1	3	3	-1	-1
7	-1	-1	6	1	6	-1	-1

from

from 1 to 2: 1, 4, 3, 5, 2

5 to 7: 5, 3, 6, 7

**Algorithm 2** SOLUTION(u,v,P)

```

if u == v then
    return {u}
end if
return {u} + FIND-PATH(u, v, P) + {v}

```

Algorithm 3 FIND-PATH(u,v,P)

```

if P[u, v] == -1 then
    return ∅
end if
k = P[u, v]
return FIND-PATH(u, k, P) + {k} + FIND-PATH(k, v, P)

```

(Cyclic) V!

to

Summary

This topic:

- The concept of all-pairs shortest paths
- The Floyd-Warshall algorithm
- Finding the shortest paths
- Finding the transitive closure