The Floyd–Warshall Algorithm

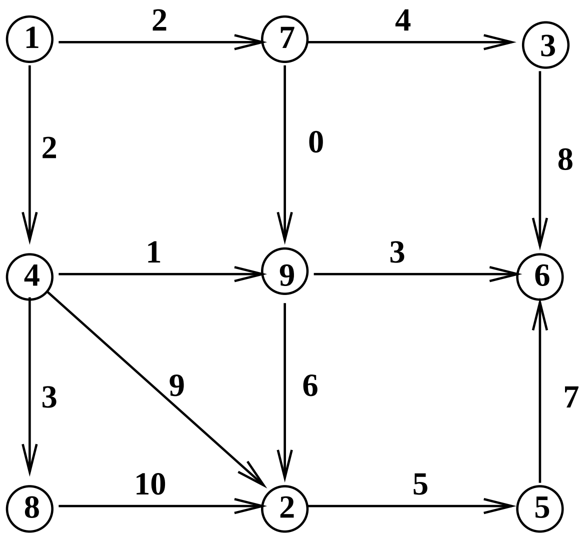
UNLV: Analysis of Algorithms   Lawrence L. Larmore

The All Pairs Minimum Path Problem

We are given a weighted directed graph $G = (V, E, W)$. That is, if $e = (u, v) \in E$, then $W(e) = W(u, v)$ is the *weight* of the edge e. A solution to the problem consists of arrays $\{V[u, v]\}_{u,v \in V}$ and $\{back[u, v]\}_{u,v \in V}$, such that $V[u, v]$ is the minimum weight of any path from u to v, while $back[u, v]$ is the next-to-the-last vertex of one of those minimum paths. There is no solution if G has a negative cycle. For convenience, we assume that the vertices of G are the integers $1, \ldots n$.

```
for all 1  i  n
   V[i,i] := 0
   back[i,i] := * \\ undefined
endfor
for all 1 <= i <= n
   for all 1  j  n
      if there is an edge from i to j
         back[i,j] := i
         V[i,j] := W(i,j)
      else
         back[i,j] := * \\ undefined
         V[i,j] := infinity
      endif
   endfor
endfor
for all 1 <= j <= n
   for all 1 <= i <= n
      for all 1 <= k <= n
         temp := V[i,j] + V[j,k]
         if temp < V[i,k]
            V[i,k] := temp
            back[i,k] := back[j,k]
         endif
      endfor
   endfor
endfor
```

Output of the Floyd–Warshall Algorithm



We consider the Floyd-Warshall algorithm on the graph illustrated above. The arrays V and *back* are initialized below.

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

back	1	2	3	4	5	6	7	8	9
1	⊥	⊥	⊥	1	⊥	⊥	1	⊥	⊥
2	⊥	⊥	⊥	⊥	2	⊥	⊥	⊥	⊥
3	⊥	⊥	⊥	⊥	⊥	3	⊥	⊥	⊥
4	⊥	4	⊥	⊥	⊥	⊥	⊥	4	4
5	⊥	⊥	⊥	⊥	⊥	5	⊥	⊥	⊥
6	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥
7	⊥	⊥	7	⊥	⊥	⊥	⊥	⊥	7
8	⊥	8	⊥	⊥	⊥	⊥	⊥	⊥	⊥
9	⊥	9	⊥	⊥	⊥	9	⊥	⊥	⊥

What will V and *back* be after one iteration of the outer loop?

We now show V and *back* after two iterations of the outer loop.

V	1	2	3	4	5	6	7	8	9
1	0	∞	∞	2	∞	∞	2	∞	∞
2	∞	0	∞	∞	5	∞	∞	∞	∞
3	∞	∞	0	∞	∞	8	∞	∞	∞
4	∞	9	∞	0	14	∞	∞	3	1
5	∞	∞	∞	∞	0	7	∞	∞	∞
6	∞	∞	∞	∞	∞	0	∞	∞	∞
7	∞	∞	4	∞	∞	∞	0	∞	0
8	∞	10	∞	∞	15	∞	∞	0	∞
9	∞	6	∞	∞	11	3	∞	∞	0

back	1	2	3	4	5	6	7	8	9
1	⊥	⊥	⊥	1	⊥	⊥	1	⊥	⊥
2	⊥	⊥	⊥	⊥	2	⊥	⊥	⊥	⊥
3	⊥	⊥	⊥	⊥	⊥	3	⊥	⊥	⊥
4	⊥	4	⊥	⊥	2	⊥	⊥	4	4
5	⊥	⊥	⊥	⊥	⊥	5	⊥	⊥	⊥
6	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥
7	⊥	⊥	7	⊥	⊥	⊥	⊥	⊥	7
8	⊥	8	⊥	⊥	2	⊥	⊥	⊥	⊥
9	⊥	9	⊥	⊥	2	9	⊥	⊥	⊥

Fill in the values of the arrays after 6 iterations of the outer loop. (For convenience, do not write ∞ or $\perp$; simply leave the entry blank.)

V	1	2	3	4	5	6	7	8	9
1	0								
2		0							
3			0						
4				0					
5					0				
6						0			
7							0		
8								0	
9									0

back	1	2	3	4	5	6	7	8	9
1									
2									
3									
4									
5									
6									
7									
8									
9									

Fill in the arrays after 8 iterations of the outer loop. (For convenience, do not write ∞ or $\perp$; simply leave the entry blank.)

V	1	2	3	4	5	6	7	8	9
1	0								
2		0							
3			0						
4				0					
5					0				
6						0			
7							0		
8								0	
9									0

back	1	2	3	4	5	6	7	8	9
1									
2									
3									
4									
5									
6									
7									
8									
9									