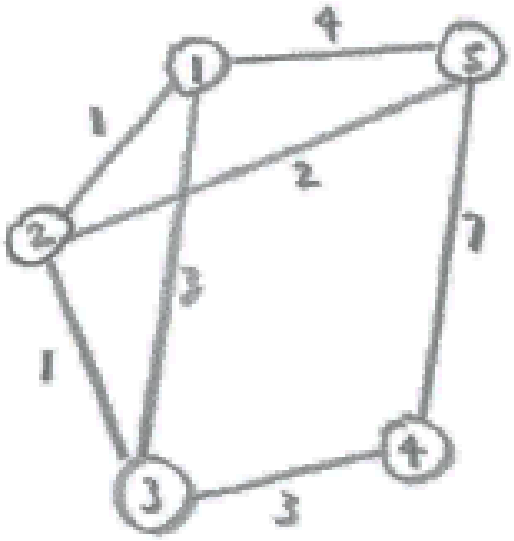
Floyd's Algorithm (10 pages; 9/3/14)

(To find the matrix of shortest distances between all pairs of nodes, together with a route matrix.)

Example

[Note: Nodes are usually labelled with numbers, as here]



We start with a table (or 'matrix') of the direct distances between nodes. Where no direct arc exists, an ∞ symbol is entered (and also between nodes of the same number). Eventually all cells will be replaced with the shortest distance - whether direct or not.

D0 [S= start node; E=end node]

S E	1	2	3	4	5
1	∞	1	3	∞	4
2	1	∞	1	∞	2
3	3	1	∞	3	∞
4	∞	∞	3	∞	7
5	4	2	∞	7	∞

The route matrix shows the first node that is visited in travelling from one node to another. Initially it is assumed that the end node is reached directly.

R0

S E	1	2	3	4	5
1	1	2	3	4	5
2	1	2	3	4	5
3	1	2	3	4	5
4	1	2	3	4	5
5	1	2	3	4	5

We then see if the distances in D0 can be improved on, by going via node 1 in each case. This is done by highlighting the 1st row and column of D0, to give D1a, as explained below.

D1a

S E	1	2	3	4	5
1	∞	1	3	∞	4
2	1	∞	1	∞	2
3	3	1	∞	3	∞
4	∞	∞	3	∞	7
5	4	2	∞	7	∞

As an example, at present there is no distance recorded for starting at node 3 and ending at node 5. If we consider this journey in two parts: node 3 to node 1, and then node 1 to node 5, we find that we now have a total distance of 7 [3 from cell (S3,E1) + 4 from cell (S1,E5)], which is an improvement on no distance at all.

Obviously, for a start node of 1, we can't make any improvement. Hence the only cells we need to look at are the ones not in the first row or column. In each case we look sideways to the shaded column, and upwards to the shaded row, and add the two figures found, comparing the total with the value in the cell we are considering.

The improved distances are shown (in red) in D1b.

D1b

S E	1	2	3	4	5
1	∞	1	3	∞	4
2	1	2	1	∞	2
3	3	1	6	3	7
4	∞	∞	3	∞	7
5	4	2	7	7	8

To create R1, we now find the new first destination for each of the cells that have just been changed.

R1

S E	1	2	3	4	5
1	1	2	3	4	5
2	1	1	3	4	5
3	1	2	1	4	1
4	1	2	3	4	5
5	1	2	1	4	1

For R1, this will just be 1 in each case, because of the 1s in the 1st column.

We now look for improved distances by going via node 2. The only cells that need to be considered are the ones not in row 2 and column 2.

D2a

S E	1	2	3	4	5
1	∞	1	3	∞	4
2	1	2	1	∞	2
3	3	1	6	3	7
4	∞	∞	3	∞	7
5	4	2	7	7	8

D2b

S E	1	2	3	4	5
1	2	1	2	∞	3
2	1	2	1	∞	2
3	2	1	2	3	3
4	∞	∞	3	∞	7
5	3	2	3	7	4

R2

S E	1	2	3	4	5
1	2	2	2	4	2
2	1	1	3	4	5
3	2	2	2	4	2
4	1	2	3	4	5
5	2	2	2	4	2

For each of the cells that were changed in D2b, the corresponding cell in R2 now becomes the value in the 2nd column of R1, which is the 1st node en route to node 2 (which needn't be 2).

D3a

S E	1	2	3	4	5
1	2	1	2	∞	3
2	1	2	1	∞	2
3	2	1	2	3	3
4	∞	∞	3	∞	7
5	3	2	3	7	4

D3b

S E	1	2	3	4	5
1	2	1	2	5	3
2	1	2	1	4	2
3	2	1	2	3	3
4	5	4	3	6	6
5	3	2	3	6	4

R3

S E	1	2	3	4	5
1	2	2	2	2	2
2	1	1	3	3	5
3	2	2	2	4	2
4	3	3	3	3	3
5	2	2	2	2	2

D4a&b (no change) (∴ no change to R3)

S E	1	2	3	4	5
1	2	1	2	5	3
2	1	2	1	4	2
3	2	1	2	3	3
4	5	4	3	6	6
5	3	2	3	6	4

D5a&b (no change) (∴ no change to R4)

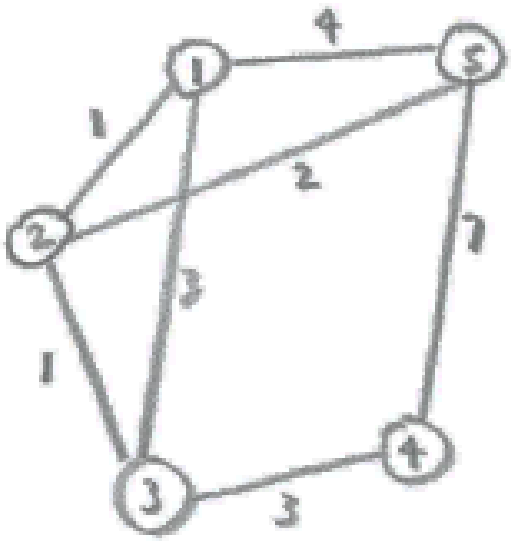
S E	1	2	3	4	5
1	2	1	2	5	3
2	1	2	1	4	2
3	2	1	2	3	3
4	5	4	3	6	6
5	3	2	3	6	4

The final distance matrix is thus:

S E	1	2	3	4	5
1	2	1	2	5	3
2	1	2	1	4	2
3	2	1	2	3	3
4	5	4	3	6	6
5	3	2	3	6	4

And the final route matrix is:

S E	1	2	3	4	5
1	2	2	2	2	2
2	1	1	3	3	5
3	2	2	2	4	2
4	3	3	3	3	3
5	2	2	2	2	2



The shortest route from node 1 to node 4 is read off the route matrix as follows:

The 1st node on this route is 2 [from cell (S1,E4)]. Then the next node is 3 [from cell (S2,E4)]. Then we go straight to 4, as cell (S3,E4) is 4. The total distance of $1 + 1 + 3 = 5$ can be read off the distance matrix, from cell (S1,E4).

Note that the shortest distance between eg node 1 and itself is "2" - which assumes (artificially) that another node has to be visited. This is only a quirk of the algorithm (resulting from recording ∞ initially for these cells).