

Floyd's Algorithm (15 pages; 19/10/26)

Dijkstra's algorithm finds the shortest distance (or time etc) between a given node and each of the other nodes in a network.

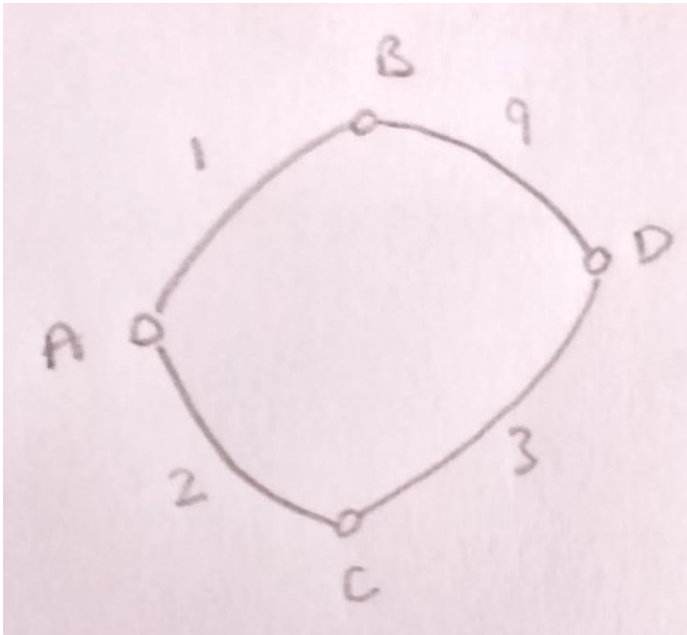
Floyd's algorithm finds the shortest distance between every pair of nodes.

Warning: There are two slightly different variations of Floyd's algorithm:

The one that is generally recommended is described here.

A variation of this method (referred to in these notes as the 'alternative method') is used in some Edexcel textbooks.

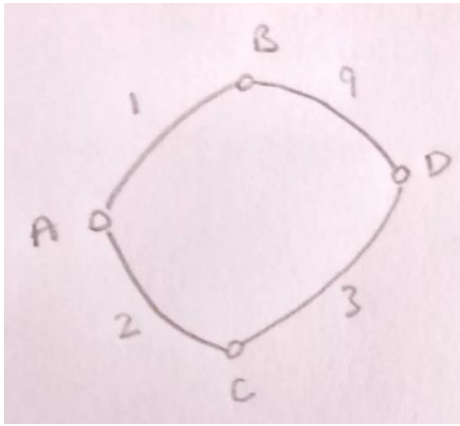
Edexcel have said that they will accept either method for the exam.



The shortest route from D to B is DCAB (a distance of 6).

Step 1 of Floyd's algorithm:

Create the initial distance table, based on direct routes only.



Distance table:

		To			
		A	B	C	D
From	A	-	1	2	∞
	B	1	-	∞	9
	C	2	∞	-	3
	D	∞	9	3	-

Step 2: Create the initial Route table

Initial Route table:

		To			
		A	B	C	D
From	A	A	B	C	D
	B	A	B	C	D
	C	A	B	C	D
	D	A	B	C	D

The Route table shows the first node to travel to.

So initially we are saying that, to get from D to B, the first node we travel to is B (a distance of 9, from the Distance table).

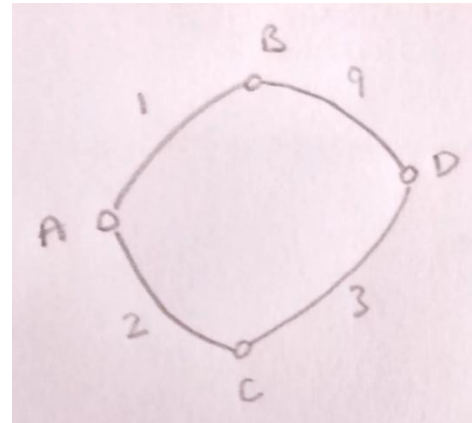
Whilst to get from D to A, the first node we travel to is A (a distance of ∞ !)

		To			
		A	B	C	D
From	A	-	1	2	∞
	B	1	-	∞	9
	C	2	∞	-	3
	D	∞	9	3	-

This is our initial attempt at finding the shortest routes – and will be improved on.

Step 3: Investigate whether shorter distances can be found in each case, if we go via A.

		To			
		A	B	C	D
From	A	-	1	2	∞
	B	1	-	∞	9
	C	2	∞	-	3
	D	∞	9	3	-



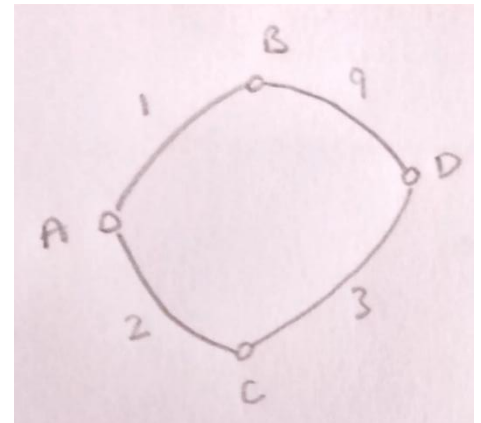
There is no need to consider going via A for any routes starting or ending at A, as these routes already include A (and hence they are 'highlighted').

For B to C, there is a distance of ∞ at present, but this can be improved on by breaking the route up into $B \rightarrow A$ and $A \rightarrow C$, giving a total distance of $1 + 2 = 3$. So the ∞ in the Distance table can be replaced with a 3.

Repeat this for the other routes.

Handwritten Route Table:

	To			
	A	B	C	D
From A	-	1	2	∞
B	1	-	(3)	9
C	2	(3)	-	3
D	∞	9	3	-



(The circled distances are the ones that have changed.)

Note: For D to B, we will **eventually** obtain a shorter route by going via A.

Step 4: Update the Route table

We now need to record the fact that, in going from B to C, we will be going via A (and similarly for C to B).

This means that we need to update the first node to be visited in going from B to C (in the Route table).

As we are now going via A, this will be the (B, A) entry in the Route table (ie the first node to be visited in going from B to A).

So we are replacing the (B, C) entry with the (B, A) entry.

Note: In the Alternative method, we replace the (B, C) entry with just A instead. (Note that (B, A) is often A anyway.)

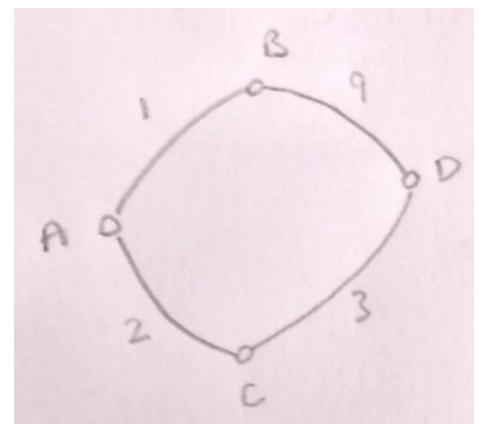
(Unfortunately – for the Alternative method - this means that the new (B, C) entry is now not necessarily the first node to be visited in going from B to C.

However, it is still possible to work out the correct route, at the end of the process.)

	To			
	A	B	C	D
From A	-	1	2	∞
B	1	-	3	9
C	2	3	-	3
D	∞	9	3	-

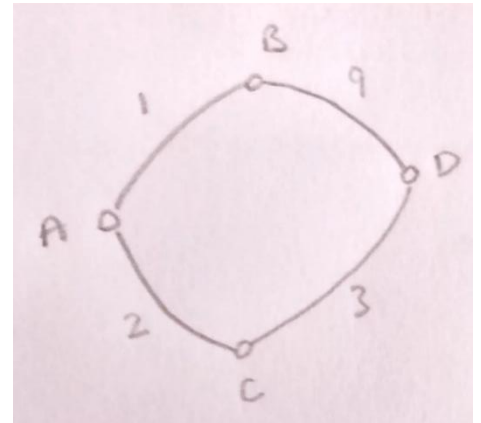
Step 5: Investigate whether the distances can be improved further by going via B.

	To			
	A	B	C	D
From A	-	1	2	∞
B	1	-	3	9
C	2	3	-	3
D	∞	9	3	-



TO

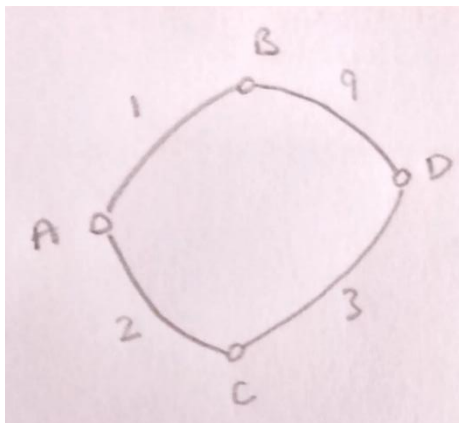
	A	B	C	D
From A	-	1	2	(10)
B	1	-	3	9
C	2	3	-	3
D	(10)	9	3	-



Step 6: Update the Route table

		To			
		A	B	C	D
From	A	A	B	C	(B)
	B	A	D	A	D
	C	A	A	C	D
	D	(B)	B	C	D

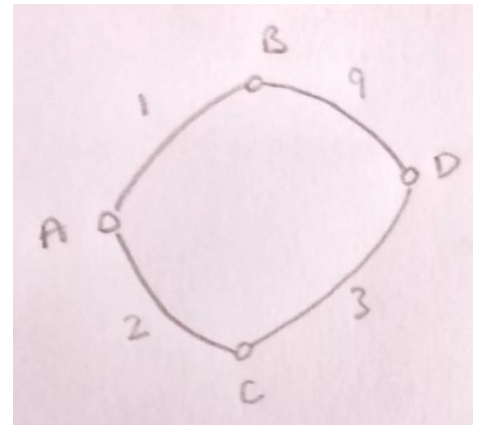
Step 7: Investigate whether the distances can be improved further by going via C, and update the Route table.



		To			
		A	B	C	D
From	A	-	1	2	10
	B	1	-	3	9
	C	2	3	-	3
	D	10	9	3	-

TO

	A	B	C	D
From A	-	1	2	(5)
B	1	-	3	(6)
C	2	3	-	3
D	(5)	(6)	3	-



TO

	A	B	C	D
From A	A	B	C	(C)
B	A	B	A	(A)
C	A	A	C	D
D	(C)	(C)	C	D

Note that (B, D) has been replaced by (B, C), which is A (whereas for the Alternative Method we would be putting C instead).

Step 8: Investigate whether the distances can be improved further by going via D, and update the Route table.

		To			
		A	B	C	D
From	A	—	1	2	5
	B	1	—	3	6
	C	2	3	—	3
	D	5	6	3	—

No further improvements can be made by considering going via D.

Final Distance table:

		To			
		A	B	C	D
From	A	—	1	2	5
	B	1	—	3	6
	C	2	3	—	3
	D	5	6	3	—

Final Route table:

		To			
		A	B	C	D
From	A	A	B	C	C
	B	A	B	A	A
	C	A	A	C	D
	D	C	C	C	D

Establish the shortest routes (and distances) for the following:

$D \rightarrow B$; $B \rightarrow C$; $A \rightarrow D$

		To			
		A	B	C	D
From	A	—	1	2	5
	B	1	—	3	6
	C	2	3	—	3
	D	5	6	3	—

		To			
		A	B	C	D
From	A	A	B	C	C
	B	A	B	A	A
	C	A	A	C	D
	D	C	C	C	D

		To			
		A	B	C	D
From	A	-	1	2	5
	B	1	-	3	6
	C	2	3	-	3
	D	5	6	3	-

		To			
		A	B	C	D
From	A	A	B	C	C
	B	A	B	A	A
	C	A	A	C	D
	D	C	C	C	D

$D \rightarrow B: D \rightarrow C \rightarrow A \rightarrow B$ (6)

$B \rightarrow C: B \rightarrow A \rightarrow C$ (3)

$A \rightarrow D: A \rightarrow C \rightarrow D$ (5)

Notes

(1) The cells in the leading diagonal (ie from top left to bottom right) of the Distance table are normally left as blanks (as the values aren't needed for anything).

In the unlikely event of an exam question not following this approach, the cells in the leading diagonal can be dealt with in exactly the same way as for the other cells.

(2) Directed arcs

It is possible for some, or all, of the arcs to be directed. The process is the same though (with some initial distances being ∞ , to indicate a 1-way arc).

(3) Floyd's algorithm always produces the optimal solution. Were the cycle to be repeated again, it would be found that no improvement results.

(4) The same result is obtained if the first iteration is carried out on (eg) the 2nd row of the table (instead of the 1st).

From the Edexcel specification:

2.2	Dijkstra's and Floyd's algorithm for finding the shortest path.	When applying Floyd's algorithm, unless directed otherwise, students will be expected to complete the first iteration on the first row of the corresponding distance and route problems, the second iteration on the second row and so on until the algorithm is complete.
-----	--	--