Floyd's Algorithm (21 pages; 17/6/20)

Contents

(I) Introduction

(1) Whereas Dijkstra's algorithm finds the shortest distance (or time, cost 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.

(2) Suppose that we wish to find the cheapest way of travelling from London to Penzance by train. Initially we might have established the following ticket prices (represented by direct arcs in a network):

London to Penzance: £100
London to Bristol: £50
Bristol to Penzance: £40
London to Reading: £15
Reading to Bristol: £30

A typical application of Floyd's algorithm would be equivalent to reasoning as follows: The £100 price can be improved on by considering Bristol as the initial destination (that would give 50 + 40 = £90, if 2 tickets were bought), but noting that the best price for Bristol involves making Reading the initial destination. Then the best price (buying 3 tickets) would be 15 + 30 + 40 = £85).

(3) The standard method for Floyd's algorithm is described here. The Edexcel Pearson textbook employs a variant of this method (referred to here as the 'Edexcel' method) - but this isn't recommended. Edexcel have said that they will now accept both methods (it is possible that other exam boards may only accept the standard method).

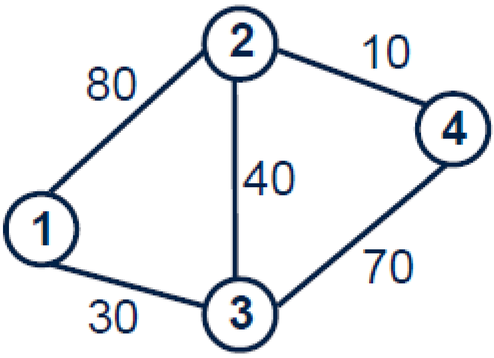
(4) Other issues
(i) Some textbooks insist on using numbers for the nodes, whilst others manage quite well with letters.

(ii) Referring to the distance matrix described below, some textbooks update the cells in the leading diagonal (ie from top left to bottom right) in the same way as for other cells. However, the values are not actually needed for anything, and it is quite acceptable (and quicker) to enter dashes instead. Exam questions may indicate the approach to be adopted.

(5) Directed arcs
It is possible for some, or all, of the arcs to be directed (see Example 3 below). If none of the arcs are directed then the distance matrix will be symmetric at each stage of the process.

(II) Example 1 - Approach A (standard method; usual order)

Note: Nodes are numbered here (but letters could have been used instead). For this example, the cells on the leading diagonal of the distance matrix are treated in the same way as other cells (in order to illustrate this possibility). See Example 2 for the alternative (and recommended) possibility of just entering dashes instead.



Step 1

Initially, we create a table of distances (or times, costs etc), or **distance matrix**, as shown in D_0 below - where only direct connections are shown (otherwise an ∞ symbol is recorded, to show that a route has not yet been found). [In the train example, this corresponds to not being allowed to purchase a ticket from London to Penzance, for some reason.]

The distance from node 2 to node 3, for example, is recorded in row 2, column 3.

Note: Were there to be any loops in the network, the infinity symbols on the leading diagonal would be replaced by the weights of these loops.

D_0	**1**	**2**	**3**	**4**
1	∞	80	30	∞
2	80	∞	40	10
3	30	40	∞	70
4	∞	10	70	∞

The route matrix is intended to reveal the best route (established so far) between two nodes. Initially we assume that the best route is the direct one (if there is no direct route, we are effectively pretending that there is one, with an infinite distance).

R_0	**1**	**2**	**3**	**4**
1	1	2	3	4
2	1	2	3	4
3	1	2	3	4
4	1	2	3	4

Step 2

We now examine each pair of nodes, in turn, to see whether the distance obtained so far can be improved on by instead having node 1 as the initial destination. [In the train example, this corresponds to investigating the possibility of first of all travelling to some intermediate destination.]

Clearly, any routes that start at node 1 cannot be improved on in this way, and for this reason the 1st row is highlighted.

For routes that finish at node 1, no improvement will be found, as it would only mean using the current best route to node 1. For this reason the 1st column is highlighted.

D_0	**1**	**2**	**3**	**4**
1	∞	80	30	∞
2	80	∞	40	10
3	30	40	∞	70
4	∞	10	70	∞

For example, the distance of 40 from node 3 to node 2 cannot be improved on by travelling to node 1 first of all (as this would give a total distance of 30 (from node 3 to node 1) plus 80 (from node 1 to node 2). However, although a route from node 2 to itself isn't required, the theoretical distance of ∞ can be improved on by travelling first to node 1, to give a total of $80 + 80 = 160$.

D_1 below shows the improved distance table after this step.

Note: The use of square brackets to indicate a changed item is not a standard convention, and would need to be explained in an exam answer.

D_1	**1**	**2**	**3**	**4**
1	∞	80	30	∞
2	80	[160]	40	10
3	30	40	[60]	70
4	∞	10	70	∞

Step 3

The route matrix is then updated for each cell in D_1 that has been changed. This can be done by looking across to the column that was highlighted when D_1 was constructed (ie the 1st column in this case) and bringing the contents of that cell over to the cell that has been changed. So, cell (2,2) in R_1 has been changed from 2 to 1, and the route from node 2 to node 2 (for what it's worth) is from node 2 to node 1, and then from node 1 to node 2 (the contents of cell (1,2) indicate that the next node in the route from node 1 to node 2 is node 2 itself). A better example of using the route matrix will appear in R_2 .

R_1	1	2	3	4
1	1	2	3	4
2	1	[1]	3	4
3	1	2	[1]	4
4	1	2	3	4

Step 4

The process is then repeated, considering the possibility of making node 2 the initial destination in each case.

D_1	1	2	3	4
1	∞	80	30	∞
2	80	160	40	10
3	30	40	60	70
4	∞	10	70	∞

D_2	1	2	3	4
1	[160]	80	30	[90]
2	80	160	40	10
3	30	40	60	[50]
4	[90]	10	[50]	[20]

For example, in D_1 the current distance from node 1 to node 4 (in general, this needn't be a direct route) is ∞. In D_2 we discover that this can be improved on by making node 2 our 1st destination, giving a total distance of 90 (80 from node 1 to node 2, plus 10 from node 2 to node 4).

Notes
(i) In practice, the improvement for a particular cell is found by looking across to the highlighted column (to find 80 in this case), and down to the highlighted row (to find 10), and comparing the total of these with the contents of the cell.

(ii) In general (later on in the process - especially for larger networks), the above route from node 1 to node 2 need not be a direct one, and similarly for the route from node 2 to node 4.

R_2	1	2	3	4
1	[2]	2	3	[2]
2	1	1	3	4
3	1	2	1	[2]
4	[2]	2	[2]	[2]

R_2 shows that the current best route from node 1 to node 4 is 124: the contents of cell (1,4) tell us to go to node 2, and the contents of cell (2,4) tell us to go from node 2 to node 4.

Step 5

The process is repeated again, considering the possibility of making node 3 the initial destination in each case (with the qualification that the route to node 3 may go via other nodes).

D_2	1	2	3	4
1	160	80	30	90
2	80	160	40	10
3	30	40	60	50
4	90	10	50	20

D_3	1	2	3	4
1	[60]	[70]	30	[80]
2	[70]	[80]	40	10
3	30	40	60	50
4	[80]	10	50	20

R_3	1	2	3	4
1	[3]	[3]	3	[3]
2	[3]	[3]	3	4
3	1	2	1	2
4	[2]	2	2	2

Note: The [2] in cell (4,1) of the route matrix is the 1st change where the cell in the highlighted column was different from its original value. The 'Edexcel' variation would record a [3] in cell (4,1). (Some information is lost in doing this, and has to be worked out later.)

Step 6

The process is repeated one last time, considering the possibility of making node 4 the initial destination in each case.

D_3	1	2	3	4
1	60	70	30	80
2	70	80	40	10
3	30	40	60	50
4	80	10	50	20

D_4	1	2	3	4
1	60	70	30	80
2	70	[20]	40	10
3	30	40	60	50
4	80	10	50	20

R_4	1	2	3	4
1	3	3	3	3
2	3	[4]	3	4
3	1	2	1	2
4	2	2	2	2

The final routes can now be read from R_4. For example, the route from node 1 to node 4 is 1324: the contents of cell (1,4) tell us to go to node 3, the contents of cell (3,4) tell us to go next to node 2, and the contents of cell (2,4) tell us to go from node 2 to node 4.

Notes
(i) Were the cycle to be repeated again (ie considering each node in turn as a possible initial destination), it will be found that no improvement results.

(ii) Floyd's algorithm always produces the optimal solution.

(iii) The same result is obtained if the order in which initial destination nodes are considered is changed, as can be seen in Approach B below.

(III) Example 1 - Approach B (standard method; different order)

New order of initial destination nodes: 3421

D_0	**1**	**2**	**3**	**4**
1	∞	80	30	∞
2	80	∞	40	10
3	30	40	∞	70
4	∞	10	70	∞

D_1	1	2	3	4
1	[60]	[70]	30	[100]
2	[70]	[80]	40	10
3	30	40	∞	70
4	[100]	10	70	[140]

R_1	1	2	3	4
1	[3]	[3]	3	[3]
2	[3]	[3]	3	4
3	1	2	3	4
4	[3]	2	3	4

D_1	1	2	3	4
1	60	70	30	100
2	70	80	40	10
3	30	40	∞	70
4	100	10	70	140

D_2	1	2	3	4
1	60	70	30	100
2	70	[20]	40	10
3	30	40	[140]	70
4	100	10	70	140

R_2	1	2	3	4
1	3	3	3	3
2	3	[4]	3	4
3	1	2	[4]	4
4	3	2	3	4

D_2	1	2	3	4
1	60	70	30	100
2	70	20	40	10
3	30	40	140	70
4	100	10	70	140

D_3	1	2	3	4
1	60	70	30	[80]
2	70	20	40	10
3	30	40	[80]	[50]
4	[80]	10	[50]	[20]

R_3	1	2	3	4
1	3	3	3	[3]
2	3	4	3	4
3	1	2	[2]	[2]
4	[2]	2	[2]	[2]

D_3	1	2	3	4
1	60	70	30	80
2	70	20	40	10
3	30	40	80	50
4	80	10	50	20

D_4	1	2	3	4
1	60	70	30	80
2	70	20	40	10
3	30	40	[60]	50
4	80	10	50	20

R_4	1	2	3	4
1	3	3	3	3
2	3	4	3	4
3	1	2	[1]	2
4	2	2	2	2

(IV) Comparison of Approaches A & B

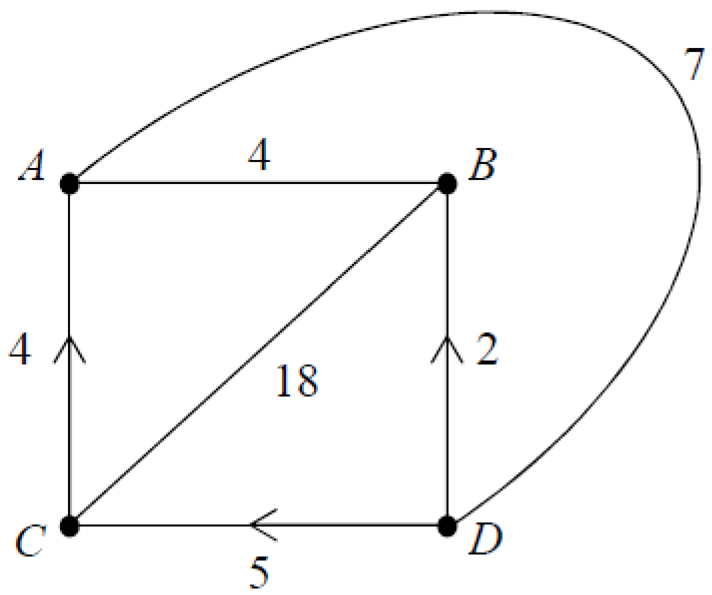
(A: 1234 ; B: 3421)

	1→ 4		**2→ 1**		**4→ 3**	
	A	**B**	**A**	**B**	**A**	**B**
$\boldsymbol{R_1}$	14	134	21	231	43	43
$\boldsymbol{R_2}$	124	134	21	231	423	43
$\boldsymbol{R_3}$	1324*	1324#	231	231	423	423
$\boldsymbol{R_4}$	1324	1324	231	231	423	423

* 124 → 1324: The route from 1 to 4 with 2 as the initial destination is being abandoned in favour of the route with 3 as the initial destination (and 324 is currently the best route from 3 to 4)

134 → 1324: The best route from 3 to 4 is now via 2, and this has an effect on the route from 1 to 4

(V) Example 2 - directed arcs (standard method; usual order)



[Dashes are entered on the leading diagonal, to save time.]

D_0	A	B	C	D
A	-	4	∞	7
B	4	-	18	∞
C	4	18	-	∞
D	7	2	5	-

R_0	**A**	**B**	**C**	**D**
A	A	B	C	D
B	A	B	C	D
C	A	B	C	D
D	A	B	C	D

D_0	**A**	**B**	**C**	**D**
A	-	4	∞	7
B	4	-	18	∞
C	4	18	-	∞
D	7	2	5	-

D_1	**A**	**B**	**C**	**D**
A	-	4	∞	7
B	4	-	18	[11]
C	4	[8]	-	[11]
D	7	2	5	-

R_1	**A**	**B**	**C**	**D**
A	A	B	C	D
B	A	B	C	[A]
C	A	[A]	C	[A]
D	A	B	C	D

D_1	**A**	**B**	**C**	**D**
A	-	4	∞	7
B	4	-	18	11
C	4	8	-	11
D	7	2	5	-

D_2	**A**	**B**	**C**	**D**
A	-	4	[22]	7
B	4	-	18	11
C	4	8	-	11
D	[6]	2	5	-

R_2	A	B	C	D
A	A	B	[B]	D
B	A	B	C	A
C	A	A	C	A
D	[B]	B	C	D

D_3 & R_3: same as D_2 & R_2

D_3	A	B	C	D
A	-	4	22	7
B	4	-	18	11
C	4	8	-	11
D	6	2	5	-

D_4	A	B	C	D
A	-	4	[12]	7
B	4	-	[16]	11
C	4	8	-	11
D	6	2	5	-

R_4	**A**	**B**	**C**	**D**
A	A	B	[D]	D
B	A	B	[A]	A
C	A	A	C	A
D	B	B	C	D

Examples of routes

	A → C	B → C	B → D
R_1	AC	BC	BAD
R_2	ABC	BC	BAD
R_3	ABC	BC	BAD
R_4	ADC	BADC*	BAD

* D is considered as the initial destination, and the current best route from B to D is BAD (established in R_1)

(V) Example 3 (standard method; usual order)

Suppose that there are 7 nodes (1-7), and the usual order of considering initial destination nodes is followed.

A possible sequence of improvements for the route from node 3 to node 5 is:

		Notes
R_0	35	All the routes in R_0 are direct (ie involving 2 nodes only).
R_1	315	Some of the routes involve 3 nodes, with 1 as the 2nd node.
R_2	3215	An improvement has been found, by making 2 the initial destination. Currently the best route from 2 to 5 is 215.
R_3	3215	No change, as we are already starting at node 3, so there is no point in making it the initial destination (which is why the 3rd row is highlighted)
R_4	32145	An improved route has been found for 1 to 5, going via 4.
R_5	32145	No change, as making 5 the initial destination would just give 32145 (as this is currently the best route from 3 to 5). This is why the 5th column is highlighted.
R_6	34625	An improvement has been found, by making 6 the initial destination. Currently the best route from 3 to 6 is 346 (established in R_4), and the best route from 6 to 5 is 625 (established in R_2). Note that we didn't find a route starting with 34 in R_4, because the route 4625 hadn't yet been discovered.
R_7	314765	An improvement has been found, by making 7 the initial destination. The best route from 3 to 7 is 3147, and the best route from 7 to 5 is 765.

Note: This is a fairly extreme example. Usually the route doesn't keep changing drastically, and often quickly settles down to the optimal one.