

I wrote following MATLAB codes for this question:

-----------------------------------------------------------------------------

% @Author: Baihan Lin, @Date: Oct 2016

% initiate adjacency matrix A.

A = [0 0 1 0 0 0 0 0 1;

0 0 0 1 1 0 0 0 0;

1 0 0 0 0 0 1 0 0;

0 1 0 0 1 0 1 0 0;

0 1 0 1 0 1 0 0 0;

0 0 0 0 1 0 0 1 1;

0 0 1 1 0 0 0 0 0;

0 0 0 0 0 1 0 0 0;

1 0 0 0 0 1 0 0 0;];

% find minimum steps for every vertex can connect to each other (complete).

n = whenComplete(A, 1);

n % minimum steps to reach a complete graph (if n = -1, not connected).

An = A^n % the final complete adjacency graph in n steps.

-----------------------------------------------------------------------------

% to find out in how many steps can each vertex be connected to another one.

function n = whenComplete(A, n)

% A is the input adjacency matrix

% n is the starting steps (set to -1 if it is not connected at all)

if n ~= -1

% only in connected graphs, we find minimum steps towards complete.

if prod(sum(A)) == 0 % if one column of 0, not connected, set n to -1.

n = -1;

else

if prod(prod(A^n)) == 0 % not complete graph yet within n step.

n = whenComplete(A, n+1);

end

end

end

end

Here is the output on MATLAB console:

-----------------------------------------------------------------------------

n =

6

An =

21 15 1 11 10 24 17 1 1

15 41 10 43 43 35 27 9 11

1 10 21 25 17 9 3 7 17

11 43 25 65 46 42 19 10 17

10 43 17 46 73 21 34 25 33

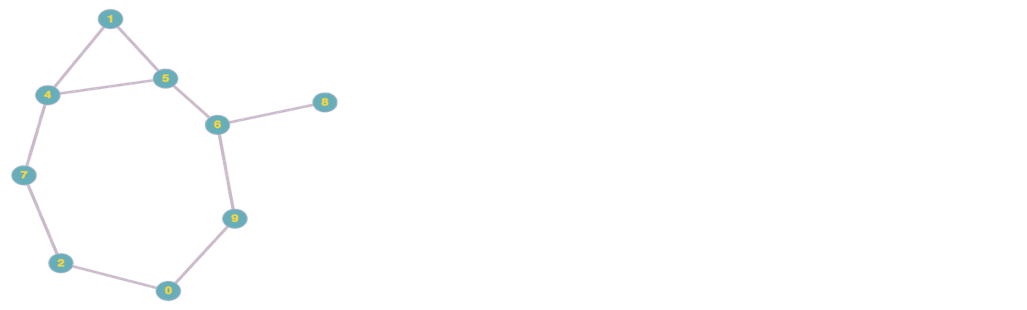
24 35 9 42 21 54 17 2 3

17 27 3 19 34 17 29 8 9

1 9 7 10 25 2 8 12 17

1 11 17 17 33 3 9 17 28

-----------------------------------------------------------------------------



As shown in Figure 1, based on the adjacency matrix, this graph is connected.

But we can further prove this by definition of this adjacency matrix: *A­­k* gives the number of walks of length k from vertex *i* to vertex *j* in graph *G*.

In another word, if *G* is connected, there must be a number *k* of walks to satisfy the path from any vertex *i* to any vertex *j* in *G*.

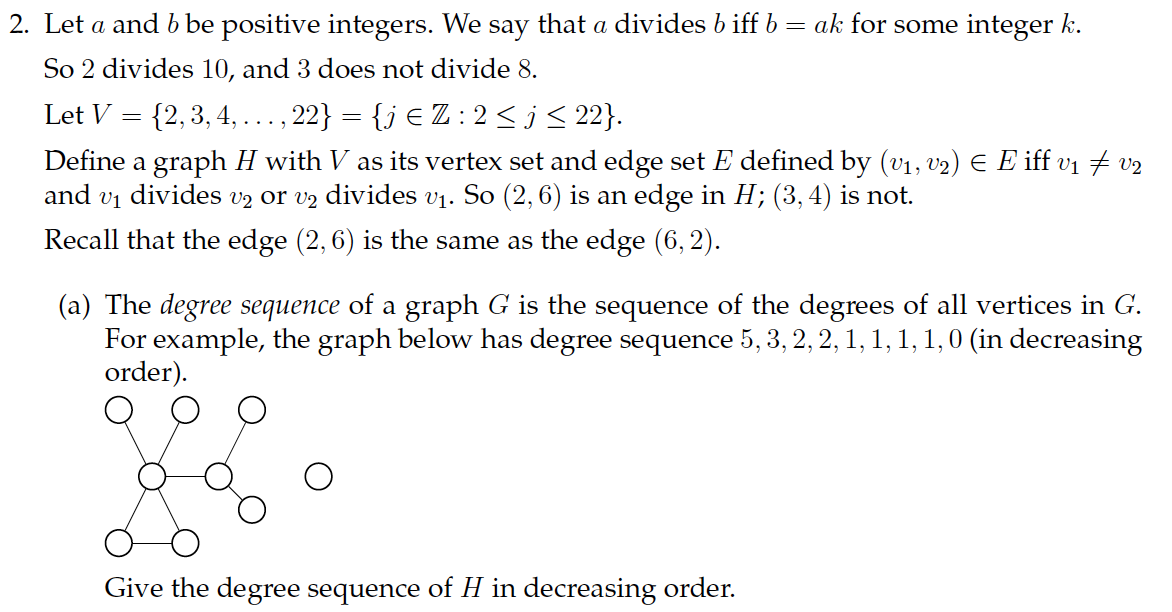
If so, it means from every vertex, it is possible to travel to another vertex in certain steps, which suggests the connectivity of the graph.

Figure 1. Graph based on adjacency matrix

Therefore, I wrote a program to find the minimum steps it takes, to let every vertex in this graph to travel towards any other vertices (more details in comments).

I found that when n = 6, every vertex has at least one possible ways to travel to any other vertex in 6 steps. Since A6 has no zero entries, we know there is a path (of length 6) between every pair of vertices.

Thus, this proves that the graph is connected.



Based on the definition of *E*, we can enumerate out all combinations of (*v1, v2*):

For vertex 2, we have 10 edges:

(2, 4), (2, 6), (2, 8), (2, 10), (2, 12), (2, 14), (2, 16), (2, 18), (2, 20), (2, 22)

in another word, vertex 2 has degree 10

For vertex 3, we have 6 edges:

(3, 6), (3, 9), (3, 12), (3, 15), (3, 18), (3, 21)

in another word, vertex 3 has degree 6

For vertex 4, we have 5 edges:

(2, 4), (4, 8), (4, 12), (4, 16), (4, 20)

in another word, vertex 4 has degree 5

For vertex 5, we have 3 edges:

(5, 10), (5, 15), (5, 20)

in another word, vertex 5 has degree 3

For vertex 6, we have 4 edges:

(2, 6), (3, 6), (6, 12), (6, 18)

in another word, vertex 6 has degree 4

For vertex 7, we have 2 edges:

(7, 14), (7, 21)

in another word, vertex 7 has degree 2

For vertex 8, we have 3 edges:

(2, 8), (4, 8), (8, 16)

in another word, vertex 8 has degree 3

For vertex 9, we have 2 edges:

(3, 9), (9, 18)

in another word, vertex 9 has degree 2

For vertex 10, we have 2 edges:

(5, 10), (10, 20)

in another word, vertex 10 has degree 2

For vertex 11, we have 1 edges:

(11, 22)

in another word, vertex 11 has degree 1

For vertex 12, we have 4 edges:

(2, 12), (3, 12), (4, 12), (6, 12)

in another word, vertex 12 has degree 4

For vertex 13, we have 0 edges:

in another word, vertex 13 has degree 0

For vertex 14, we have 2 edges:

(2, 14), (7, 14)

in another word, vertex 14 has degree 2

For vertex 15, we have 2 edges:

(3, 15), (5, 15)

in another word, vertex 15 has degree 2

For vertex 16, we have 3 edges:

(2, 16), (4, 16), (8, 16)

in another word, vertex 16 has degree 3

For vertex 17, we have 0 edges:

in another word, vertex 17 has degree 0

For vertex 18, we have 4 edges:

(2, 18), (3, 18), (6, 18), (9, 18)

in another word, vertex 18 has degree 4

For vertex 19, we have 0 edges:

in another word, vertex 19 has degree 0

For vertex 20, we have 4 edges:

(2, 20), (4, 20), (5, 20), (10, 20)

in another word, vertex 2o has degree 4

For vertex 21, we have 3 edges:

(3, 21), (7, 21)

in another word, vertex 21 has degree 3

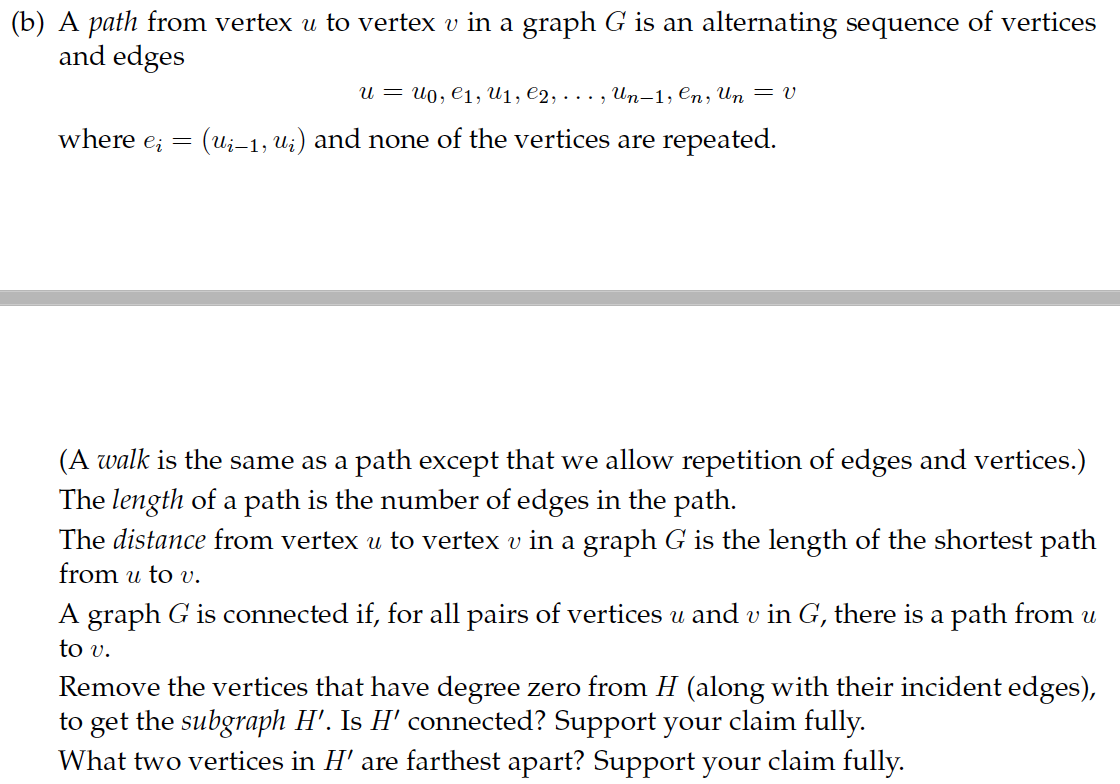
For vertex 22, we have 2 edges:

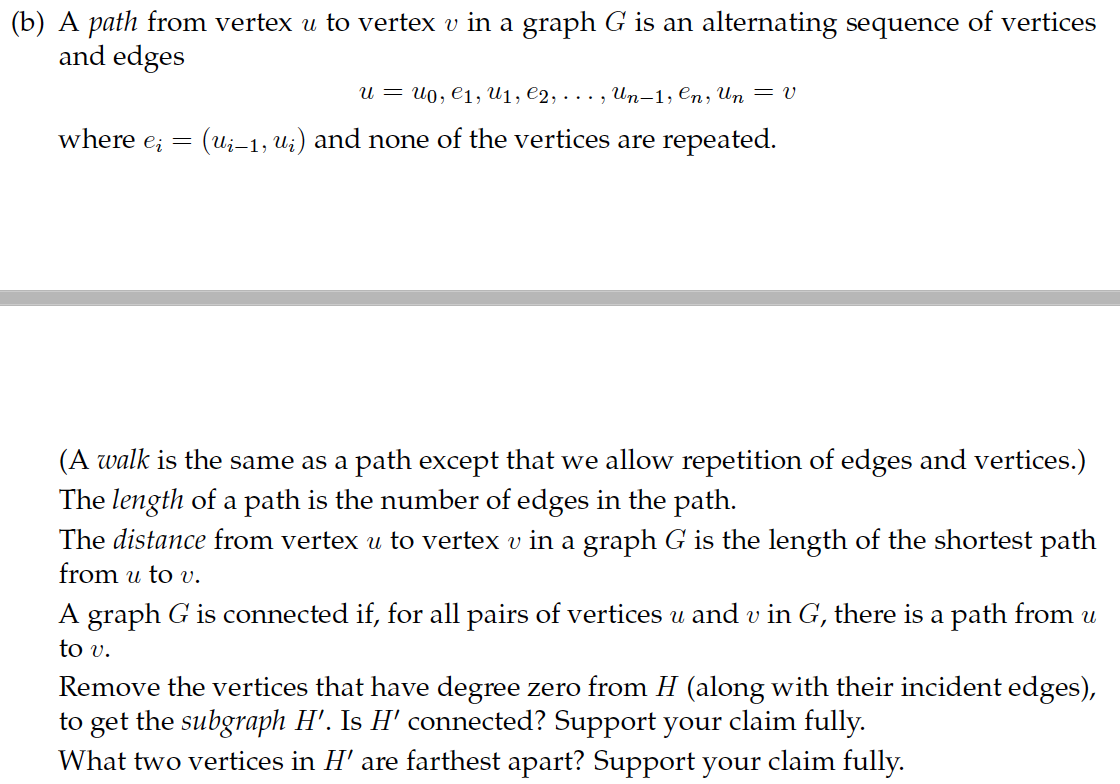
(2, 22), (11, 22)

in another word, vertex 22 has degree 2

As these vertices are vertices of graph *H*, the degree sequence of *H* is:

10, 6, 5, 4, 4, 4, 4, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 1, 0, 0, 0 (in decreasing order)





From (a), we find that vertices 13, 17 and 19 have degree 0. Thus we remove them from *H*, *H’* consists of vertices 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15, 16, 18, 20, 21, and 22.

The adjacency matrix for *H’* is

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **14** | **15** | **16** | **18** | **20** | **21** | **22** |
| **2** | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| **3** | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| **4** | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| **6** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| **7** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| **8** | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **9** | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| **10** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| **11** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| **12** | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **14** | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **15** | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **16** | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **18** | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **20** | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **21** | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **22** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

So as question 2, I wrote following MATLAB codes to approach this question:

-----------------------------------------------------------------------------

% @Author: Baihan Lin, @Date: Oct 2016

Hp = [0 0 1 0 1 0 1 0 1 0 1 1 0 1 1 1 0 1;

0 0 0 0 1 0 0 1 0 0 1 0 1 0 1 0 1 0;

1 0 0 0 0 0 1 0 0 0 1 0 0 1 0 1 0 0;

0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0;

1 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0;

0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0;

1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0;

0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0;

0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0;

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1;

1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0;

1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0;

0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0;

1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0;

1 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0;

1 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0;

0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0;

1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0;];

% find minimum steps for every vertex can connect to each other (complete).

nH = whenComplete(Hp, 1);

nH % minimum steps to reach a complete graph (if n = -1, not connected).

Hn = Hp^nH % the final complete adjacency graph in n steps.

-----------------------------------------------------------------------------

% to find out in how many steps can each vertex be connected to another one.

function n = whenComplete(A, n)

% A is the input adjacency matrix

% n is the starting steps (set to -1 if it is not connected at all)

if n ~= -1

% only in connected graphs, we find minimum steps towards complete.

if prod(sum(A)) == 0 % if one column of 0, not connected, set n to -1.

n = -1;

else

if prod(prod(A^n)) == 0 % not complete graph yet within n step.

n = whenComplete(A, n+1);

end

end

end

end

-----------------------------------------------------------------------------

Here is the output on MATLAB console:

--------------------------------------------------------------------------------

nH =

6

Hn =

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2281 | 1148 | 1515 | 586 | 1173 | 220 | 981 | 450 | 767 | 141 | 1192 | 419 | 274 | 981 | 967 | 1028 | 204 | 413 |
| 1127 | 768 | 694 | 282 | 664 | 140 | 416 | 298 | 330 | 63 | 615 | 193 | 157 | 416 | 575 | 432 | 138 | 187 |
| 1478 | 691 | 1065 | 375 | 807 | 119 | 711 | 287 | 574 | 82 | 837 | 332 | 201 | 711 | 673 | 740 | 150 | 327 |
| 384 | 205 | 261 | 136 | 181 | 38 | 152 | 65 | 144 | 22 | 177 | 62 | 45 | 152 | 145 | 180 | 24 | 62 |
| 1168 | 668 | 829 | 265 | 798 | 98 | 582 | 338 | 476 | 56 | 811 | 314 | 245 | 582 | 708 | 619 | 210 | 299 |
| 219 | 141 | 120 | 56 | 98 | 37 | 65 | 43 | 47 | 15 | 87 | 21 | 15 | 65 | 78 | 66 | 12 | 21 |
| 963 | 417 | 729 | 227 | 580 | 64 | 551 | 200 | 434 | 48 | 660 | 276 | 169 | 550 | 520 | 585 | 127 | 271 |
| 447 | 299 | 291 | 94 | 337 | 43 | 200 | 165 | 164 | 22 | 326 | 114 | 113 | 200 | 301 | 216 | 104 | 104 |
| 307 | 132 | 219 | 101 | 151 | 24 | 138 | 54 | 133 | 18 | 158 | 59 | 51 | 138 | 121 | 166 | 27 | 58 |
| 141 | 64 | 83 | 36 | 56 | 15 | 49 | 22 | 35 | 12 | 55 | 15 | 9 | 49 | 41 | 50 | 6 | 15 |
| 1172 | 616 | 858 | 261 | 809 | 86 | 660 | 326 | 528 | 54 | 897 | 357 | 266 | 660 | 757 | 714 | 220 | 342 |
| 419 | 195 | 349 | 94 | 316 | 21 | 278 | 115 | 232 | 15 | 359 | 165 | 109 | 278 | 298 | 301 | 92 | 156 |
| 241 | 148 | 180 | 54 | 223 | 14 | 147 | 108 | 137 | 8 | 241 | 94 | 102 | 147 | 215 | 179 | 84 | 85 |
| 963 | 417 | 729 | 227 | 580 | 64 | 550 | 200 | 434 | 48 | 660 | 276 | 169 | 551 | 520 | 585 | 127 | 271 |
| 962 | 578 | 695 | 211 | 709 | 78 | 522 | 302 | 427 | 41 | 759 | 297 | 236 | 522 | 673 | 564 | 204 | 282 |
| 824 | 357 | 646 | 214 | 537 | 48 | 512 | 188 | 430 | 36 | 632 | 269 | 188 | 512 | 499 | 580 | 132 | 263 |
| 202 | 138 | 153 | 34 | 209 | 12 | 126 | 104 | 109 | 6 | 219 | 91 | 88 | 126 | 203 | 141 | 86 | 78 |
| 413 | 189 | 344 | 94 | 301 | 21 | 273 | 105 | 227 | 15 | 344 | 156 | 100 | 273 | 283 | 295 | 79 | 153 |

----------------------------------------------------------------------------------

If *H’* is connected, there must be a number *k* of walks to satisfy the path from any vertex *i* to any vertex *j* in *H’*. If so, it means from every vertex, it is possible to travel to another vertex in certain steps, which suggests the connectivity of the graph.

Therefore, I wrote a program to find the minimum steps it takes, to let every vertex in this graph to travel towards any other vertices (more details in comments).

I found that when n = 6, every vertex has at least one possible ways to travel to any other vertex in 6 steps. Since *H’*6 has no zero entries, we know there is a path (of length 6) between every pair of vertices in this graph *H’*.

Thus, this proves that the graph *H’* is connected.

For the next question, I wrote another MATLAB function to find the farthest vertices.

% find minimum distance between every two vertices.

Hd = minD(Hp, nH)

Here in part (b), the code also shares part (a)’s other MATLAB code as inputs of this function:

-----------------------------------------------------------------------------

% to minimum distance between every two vertices (min path length).

function d = minD(H, n)

% H is the input adjacency matrix

% n is the minimum steps to make the graph complete.

fin = [H ~= 0]; % finished connecting pairs

d = 1\*[H ~= 0]; % the vertex pairs with distance of 1.

for k = 2:n

d = d + k.\*((H^k ~= 0) - fin); % add

fin = [H^k ~= 0]; % update the already connected pairs

end

-----------------------------------------------------------------------------

Here is the output on MATLAB console:

--------------------------------------------------------------------------------

Hn =

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

--------------------------------------------------------------------------------

As shown in this matrix, which shows the minimum distance of path between each two vertices, we can see that the maximum distance is 5 (next page).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **14** | **15** | **16** | **18** | **20** | **21** | **22** |
| **2** | 2 | 2 | 1 | 2 | 1 | 3 | 1 | 2 | 1 | 3 | 1 | 2 | 3 | 1 | 1 | 1 | 3 | 2 |
| **3** | 2 | 2 | 2 | 3 | 1 | 3 | 3 | 1 | 3 | 4 | 1 | 3 | 2 | 3 | 1 | 3 | 2 | 3 |
| **4** | 1 | 2 | 2 | 2 | 2 | 3 | 1 | 3 | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 1 | 3 | 2 |
| **5** | 2 | 3 | 2 | 2 | 3 | 4 | 3 | 3 | 1 | 4 | 3 | 3 | 2 | 3 | 3 | 1 | 4 | 3 |
| **6** | 1 | 1 | 2 | 3 | 2 | 3 | 2 | 2 | 2 | 3 | 1 | 2 | 2 | 2 | 1 | 2 | 2 | 2 |
| **7** | 3 | 3 | 3 | 4 | 3 | 4 | 3 | 3 | 3 | **5** | 3 | 3 | 4 | 3 | 3 | 3 | 3 | 4 |
| **8** | 1 | 3 | 1 | 3 | 2 | 3 | 2 | 3 | 2 | 3 | 2 | 2 | 4 | 1 | 2 | 2 | 4 | 2 |
| **9** | 2 | 1 | 3 | 3 | 2 | 3 | 3 | 2 | 3 | 4 | 2 | 3 | 2 | 3 | 1 | 3 | 2 | 3 |
| **10** | 2 | 3 | 2 | 1 | 3 | 4 | 3 | 4 | 2 | 4 | 3 | 3 | 2 | 3 | 3 | 1 | 4 | 3 |
| **11** | 3 | 4 | 3 | 4 | 3 | **5** | 3 | 4 | 3 | 4 | 3 | 4 | **5** | 3 | 3 | 3 | **5** | 3 |
| **12** | 1 | 1 | 1 | 3 | 1 | 3 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| **14** | 2 | 3 | 2 | 3 | 2 | 3 | 2 | 3 | 2 | 4 | 2 | 3 | 4 | 2 | 2 | 2 | 3 | 3 |
| **15** | 3 | 2 | 3 | 2 | 2 | 4 | 4 | 2 | 2 | **5** | 2 | 4 | 3 | 4 | 2 | 2 | 3 | 4 |
| **16** | 1 | 3 | 1 | 3 | 2 | 3 | 1 | 3 | 2 | 3 | 2 | 2 | 4 | 2 | 2 | 2 | 4 | 2 |
| **18** | 1 | 1 | 2 | 3 | 1 | 3 | 2 | 1 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| **20** | 1 | 3 | 1 | 1 | 2 | 3 | 2 | 3 | 1 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 4 | 2 |
| **21** | 3 | 2 | 3 | 4 | 2 | 3 | 4 | 2 | 4 | **5** | 2 | 3 | 3 | 4 | 2 | 4 | 3 | 4 |
| **22** | 2 | 3 | 2 | 3 | 2 | 4 | 2 | 3 | 2 | 3 | 2 | 3 | 4 | 2 | 2 | 2 | 4 | 3 |

And the farthest paths (or vertices pairs) are:

(7 🡨🡪 11)

(11 🡨🡪15)

(11 🡨🡪21)

They all needs at least 5-degree distance apart in *H’*.