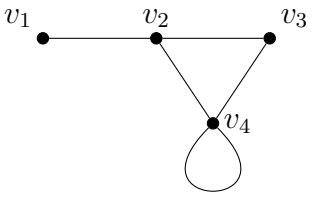
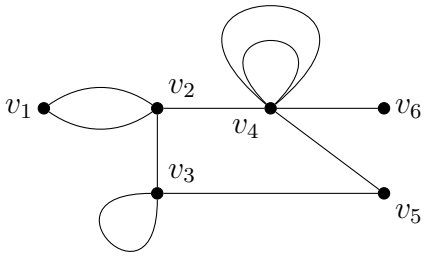
Intermediate Math Circles
Wednesday, February 15, 2017
Problem Set 2

1. Write down the adjacency matrix for each of the following graphs.

(a)



(b)



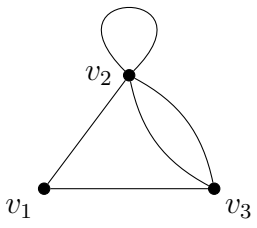
Solution: If A_1 denotes the adjacency matrix of the graph in (a) and A_2 denotes the adjacency matrix of the graph in (b), then

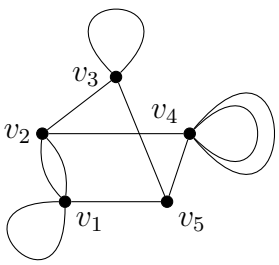
$$A_1 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \quad \text{and} \quad A_2 = \begin{bmatrix} 0 & 2 & 0 & 0 & 0 & 0 \\ 2 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 2 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

2. Draw a graph corresponding to each of the following adjacency matrices.

$$A_1 = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 0 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 1 & 2 & 0 & 0 & 1 \\ 2 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 2 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Solution: Your graphs may look very different from what is given below, and that's okay. The orientation of the graph is not important. What *is* important is that the number of edges between each pair of vertices is correct. The graph on the left has adjacency matrix A_1 and the graph on the right has adjacency matrix A_2.





3. Explain how to decide whether or not a graph is simple by looking at its adjacency matrix alone. Decide which of the following adjacency matrices corresponds to a simple graph *without drawing the graph.*

$$A_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 2 \\ 0 & 2 & 0 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}, \quad A_3 = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Solution: A graph G is simple if it has no loops and no multiple edges. How are these properties reflected in the adjacency matrix A?

(i) A loop is an edge from a vertex to itself, and hence the number of loops on vertex v_i is the (i, i)-entry of A. Thus, vertex v_i has a loop precisely when $A_{i,i} \neq 0$.

(ii) The number of edges between vertices v_i and v_j is the (i, j)-entry of A. This means that v_i and v_j have multiple edges between them exactly when $A_{i,j} \geq 2$.

Putting these conditions together, we conclude that G is simple if, and only if $A_{i,i} = 0$ for all i (that is, the entries of A on the diagonal from the top left corner to the bottom right corner are all 0) and $A_{i,j} \leq 1$ for all i and j.

With this in mind, let's look at the matrices A_1, A_2, and A_3 above.

- A_1 does not correspond to a simple graph as $(A_1)_{2,3} = 2$. There are two edges between vertices v_2 and v_3.
- A_2 does not correspond to a simple graph as $(A_1)_{3,3} = 1$. There is a loop on vertex v_3.
- Every entry of A_3 is no more than 1 and all entries on the diagonal (from top left to bottom right) are 0. This means there are no loops or multiple edges and hence the graph determined by A_3 is simple.

4. (a) Let G be a graph with vertices $v_1, v_2, \ldots, v_n$ and adjacency matrix A.

(i) Define B to be the $n \times n$ matrix whose (i, j)-entry is given by

$$B_{i,j} = A_{i,1}A_{1,j} + A_{i,2}A_{2,j} + \cdots + A_{i,n}A_{n,j}.$$

Show that $B_{i,j}$ is the number of walks of length 2 from v_i to v_j.

Hint: Remember that $A_{i,k}$ is the number of edges from v_i to v_k. With this in mind, how can we interpret each product $A_{i,k}A_{k,j}$?

Solution: As stated in the hint, $A_{i,k}$ is the number of edges between v_i and v_k, and likewise $A_{k,j}$ is the number of edges between v_k and v_j. Thus, we can think of $A_{i,k}$ as the number of ways to move from v_i to v_k in one step, and $A_{k,j}$ as the number of ways to move from v_k to v_j in one step. Their product $A_{i,k}A_{k,j}$ is then the number of ways to move from v_i to v_k in one step, and then move from v_k to v_j in a second step. This is exactly the number of walks of length 2 from v_i to v_j

that pass through v_k.

From this we see that

$$B_{i,j} = A_{i,1}A_{1,j} + A_{i,2}A_{2,j} + \cdots + A_{i,n}A_{n,j}$$

is the sum of these products over all possible choices of k. That is, $B_{i,j}$ counts the number of ways to move from v_i to v_j in 2 steps with arbitrary intermediate vertex. This is exactly the number of walks of length 2 from v_i to v_j.

(ii) With B as above, define C to be the $n \times n$ matrix whose (i, j)-entry is given by

$$C_{i,j} = B_{i,1}A_{1,j} + B_{i,2}A_{2,j} + \cdots + B_{i,n}A_{n,j}.$$

Show that $C_{i,j}$ is the number of walks of length 3 from v_i to v_j.

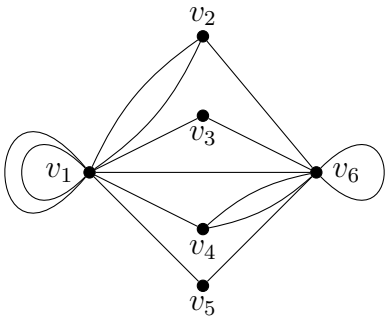
Solution: The expression defining $C_{i,j}$ looks suspiciously similar to that defining $B_{i,j}$. With this in mind, we will try to argue as we did in part (i). Note that $B_{i,k}$ is the number of ways to get from v_i to v_k in two steps, while $A_{k,j}$ is the number of ways to get from v_k to v_j in 1 step. Thus, their product $B_{i,k}A_{k,j}$ is the number of ways to move in 2 steps from v_i to v_k and then in a third step from v_k to v_j.

Again we observe that

$$C_{i,j} = B_{i,1}A_{1,j} + B_{i,2}A_{2,j} + \cdots + B_{i,n}A_{n,j}$$

is the sum of such products over all possible choices of k, and hence it counts the number of ways to move from v_i to v_j in 3 steps with arbitrary intermediate vertices. This is exactly the number of walks of length 3 from v_i to v_j.

(b) A graph G and its corresponding adjacency matrix A are given below. Use the results of part (a) to compute the number of walks of length 2 from v_1 to v_6.



$$A = \begin{bmatrix} 2 & 2 & 1 & 1 & 1 & 1 \\ 2 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 2 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 2 & 1 & 1 \end{bmatrix}$$

Solution: From part (i) of problem (a) we know that the number of paths of length 2 from vertex v_1 to vertex v_6 is given by

$$B_{1,6} = A_{1,1}A_{1,6} + A_{1,2}A_{2,6} + A_{1,3}A_{3,6} + A_{1,4}A_{4,6} + A_{1,5}A_{5,6} + A_{1,6}A_{6,6}.$$

The entries $A_{1,k}$ are found in the first row of A, and the entries $A_{k,6}$ are found in the sixth column of A. Using these values in the above expression expression, we calculate

$$\begin{aligned} B_{1,6} &= (2 \times 1) + (2 \times 1) + (1 \times 1) + (1 \times 2) + (1 \times 1) + (1 \times 1) \\ &= 2 + 2 + 1 + 2 + 1 + 1 \\ &= 9. \end{aligned}$$

Thus, there are 9 walks of length 2 from v_1 to v_6.

5. Let G be a graph. If G is not connected, show that there is some way to order the vertices of G so that its adjacency matrix has the form

$$A = \left[\begin{array}{cccc|cccc} A_{1,1} & A_{1,2} & \cdots & A_{1,m} & 0 & 0 & \cdots & 0 \\ A_{2,1} & A_{2,2} & \cdots & A_{2,m} & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{m,1} & A_{m,2} & \cdots & A_{m,m} & 0 & 0 & \cdots & 0 \\ \hline 0 & 0 & \cdots & 0 & A_{m+1,m+1} & A_{m+1,m+2} & \cdots & A_{m+1,n} \\ 0 & 0 & \cdots & 0 & A_{m+2,m+1} & A_{m+2,m+2} & \cdots & A_{m+2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & A_{n,m+1} & A_{n,m+2} & \cdots & A_{n,n} \end{array}\right]_{n \times n}$$

for some integer m with $1 \leq m < n$. Explain why this is never possible when G is connected.

Solution: If G is not connected, then there are two vertices a and b in G that are not connected by a path. Let $a = v_1$, $b = v_n$, and let $v_2, v_3, \ldots, v_m$ be the vertices in G that are connected to v_1 by a path. Order the remaining vertices as $v_{m+1}, v_{m+2}, \ldots, v_{n-1}$ arbitrarily.

We claim that under this ordering, the adjacency matrix A has the above form. To show that this is the case, we need only show that the entries $A_{i,m+j}$ are 0 when $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n-m$ (these are the entries in the upper right block of the big matrix above). [1]

Suppose that $A_{i,m+j}$ is non-zero for some i and j. This means that there is an edge between v_i and v_{m+j}. Since there is a path from v_1 to v_i and an edge from v_i to v_{m+j}, we can add this edge to the path to obtain a new path from v_1 to v_{m+j}. But, by construction, the vertices that can be connected to v_1 by a path are exactly $v_1, v_2, \ldots, v_m$. Since v_{m+j} is *not* one of these vertices, we've reached a contradiction. It must then be the case that $A_{i,m+j} = 0$ for all $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n-m$.

[1] Why don't we need to worry about the lower left block? Entries in this block correspond to edges from v_{m+j} to v_i where i and j are as above. But of course, these are the same as edges between v_i and v_{m+j} (i.e., entries in the upper right block). That said, if we can show the entries in the upper block are zero, so too must be the entries in the lower one.

Conversely, if G is connected then no ordering of the vertices will produce an adjacency matrix of this form. Indeed, a matrix of this form suggests that none of the first m vertices in this ordering (call them $v_1, v_2, \ldots, v_m$) can be connected by an edge to any of the remaining $n - m$ vertices (say $v_{m+1}, v_{m+2}, \ldots, v_n$). This means that, in particular, no path can be made between v_1 and v_n, thereby violating the assumption of connectedness

6. For each sequence d below, use the Havel-Hakimi algorithm to draw a simple graph with degree sequence d or show that such a graph does not exist.

   (a) $d = (3, 3, 3, 2, 1)$

Solution: The sequences we obtain from the Havel-Hakimi algorithm are

$$(3,3,3,2,1) \xrightarrow{reduce} (2,2,1,1) \xrightarrow{reduce} (1,0,1) \xrightarrow{reorder} (1,1,0) \xrightarrow{reduce} (0,0).$$

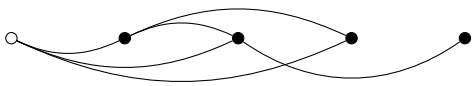
This shows that such a simple graph exists. To begin the backtracking process, start with the null graph on 2 vertices.

Since the previous step was "reduce", we add a vertex on the left, and connect it to the next vertex on its right to get a simple graph with degree sequence $(1, 1, 0)$. As in the notes, new vertices will temporarily be coloured white.

The previous step was "reorder", so instead of adding an edge we will rearrange the graph to get a new graph with degree sequence $(1, 0, 1)$.

The previous step was "reduce", so a new vertex will be added on the left and edges will connect it to the 2 vertices on its immediate right.

Finally, to obtain a simple graph with degree sequence $(3, 3, 3, 2, 1)$ we perform the action for a "reduce" step and add a vertex on the left. Connect this vertex to the 3 vertices on its immediate right.



(b) $d = (4, 4, 4, 2, 2)$

Solution: The sequences we obtain from the Havel-Hakimi algorithm are

$$(4, 4, 4, 2, 2) \xrightarrow{reduce} (3, 3, 1, 1) \xrightarrow{reduce} (2, 0, 0) \xrightarrow{reduce} (-1, -1).$$

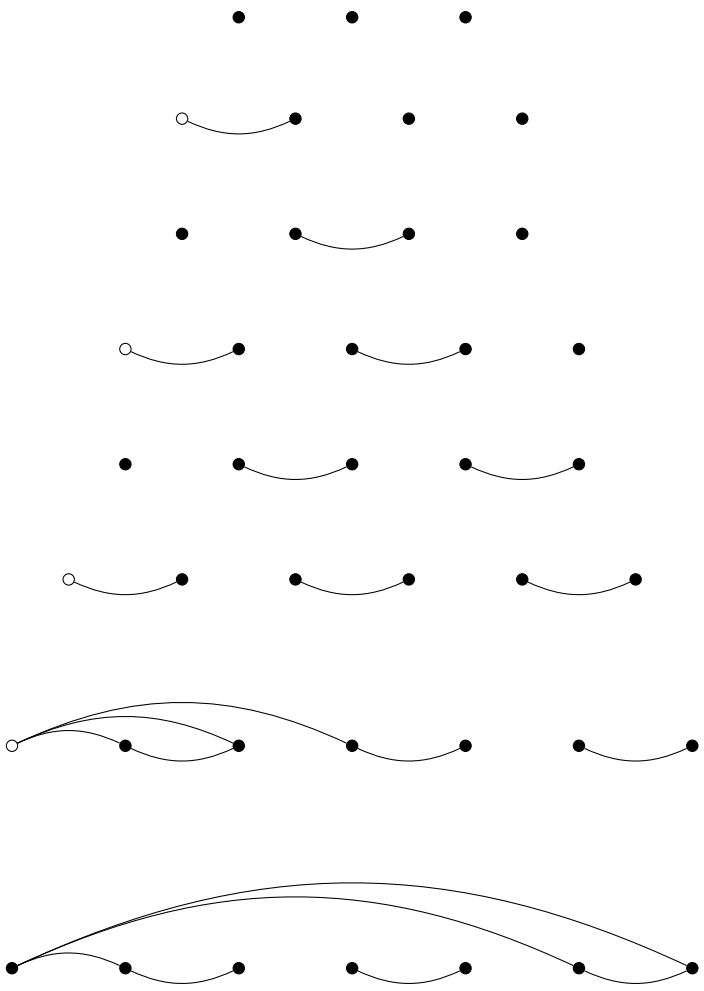
Since the final sequence contains negative entries, the algorithm stops and we conclude that no such simple graph exists.

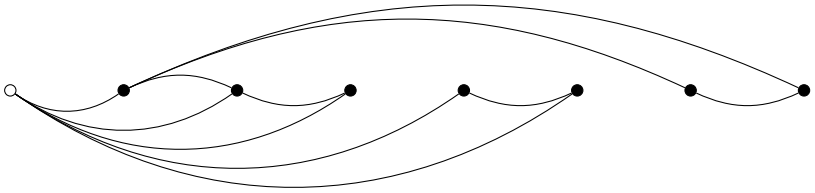
(c) $d = (5, 4, 3, 2, 2, 2, 2, 2)$

Solution: The sequences we obtain from the Havel-Hakimi algorithm are

$$\begin{aligned}(5, 4, 3, 2, 2, 2, 2, 2) &\xrightarrow{reduce} (3, 2, 1, 1, 1, 2, 2) \xrightarrow{reorder} (3, 2, 2, 2, 1, 1, 1) \\ &\xrightarrow{reduce} (1, 1, 1, 1, 1, 1) \xrightarrow{reduce} (0, 1, 1, 1, 1) \xrightarrow{reorder} (1, 1, 1, 1, 0) \\ &\xrightarrow{reduce} (0, 1, 1, 0) \xrightarrow{reorder} (1, 1, 0, 0) \xrightarrow{reduce} (0, 0, 0).\end{aligned}$$

Below are the graphs for each step of the backtracking process.





7. Explain why the backtracking process following the Havel-Hakimi algorithm always produces a simple graph.

Solution: We only execute the backtracking process when the Havel-Hakimi algorithm gives us a positive answer. A positive answer occurs if and only if the input sequence is reduced to a sequence with every entry equal to 0. Here is where the backtracking process begins.

We start with the null graph on some number of vertices, which is clearly a simple graph. The previous step must have been "reduce" and so a vertex is added on the left and an edge is added to each vertex on its right until it has the correct degree. Notice that we did not add any edges from the new vertex to itself (i.e., no loops) and vertices on the right received at most one new edge from the new vertex. This shows that there are no loops or multiple edges in the new graph, and hence it is simple.

In general, if we are are some stage of the backtracking process and currently have a simple graph, then neither action we take (determined by whether the previous step was "reduce" or "reorder") will affect the simplicity of the graph. Indeed, if the previous step was "reduce" and a new vertex was added, then the argument above ensures that the resulting graph is simple. If instead the previous step was "reorder", then all that occurs is a shuffle of the vertices. Since no edges were changed, the graph must still be simple.

This proves that at every stage of the backtracking process the graph in question is simple. In particular, the final graph produced will be a simple graph.