Intermediate Math Circles
Wednesday, February 15, 2017
Graph Theory II

1 Adjacency Matrices

Up to now we have explored two different ways to present a graph:

(i) in terms of vertex and edge sets, and

(ii) pictorially.

Explicitly writing down every vertex and edge of a graph is a bit exhausting, and so option (ii) has almost always been preferable. Unfortunately, computers process graphs more easily when they are presented in the form of option (i). Today we will discuss a third option that is easy for computers to handle, yet presents far more compactly than the data in (i). This approach makes use of a mathematical object known as a *matrix*.

1.1 What is a Matrix?

A **matrix** A is a rectangular grid of numbers or symbols. It is said to be of size $m \times n$ (read "m by n") when it consists of m rows and n columns, where m and n are positive integers. For example, we would say that the matrix

$$A = \begin{bmatrix} 4 & 0 & -1 \\ 2 & \pi & 6 \end{bmatrix}$$

is of size 2×3. The entry in row i and column j of the matrix is often referred to as the (i, j)-entry and is denoted $A_{i,j}$. In the above example, the $(1, 1)$-entry is 4, the $(1, 3)$-entry is -1, and the $(2, 1)$-entry is 2. What is the $(2, 3)$-entry?

1.2 Representing Graphs by Matrices

Let's think about what information was required to describe a graph. We needed to know the number of vertices, but the labels assigned to the vertices were arbitrary. Similarly, we needed to know the number of edges between any two vertices, but the names of those edges were unimportant. As far as we're concerned, the graphs in Figure 1 below are the same.

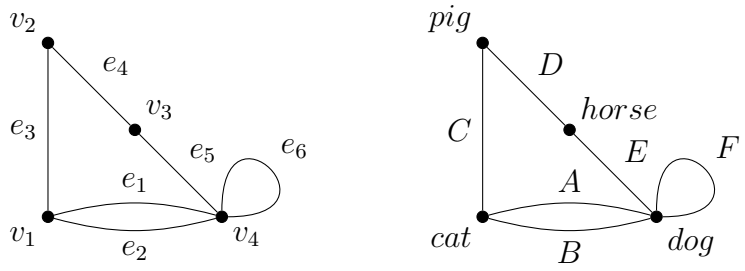


Figure 1: Different labels on the same graph.

What we're saying is the following:

To define a graph, we need only specify the number of vertices and the number of edges between any two vertices.

This information can be stored in a matrix!
Let's suppose we have a graph G with vertices $v_1, v_2, \ldots, v_n$. In order to represent G by a matrix, we create a row and a column for each vertex. Thus, a graph with n vertices will be represented by an $n \times n$ matrix. What will the entries of this matrix be? In row i and column j, we record the number of edges between vertex v_i and vertex v_j. This matrix is called the **adjacency matrix** for the graph G.

1.2.1 From Graph to Matrix

Let's construct the adjacency matrix A for the graph G in Figure 1. We will stick to the names given to the graph on the left.
The size of the matrix is determined by the number of vertices; since G has 4 vertices, A will be a 4×4 matrix. Now for the entries.

- The $(1,1)$-entry $A_{1,1}$ is the number of edges from v_1 to itself, and hence $A_{1,1} = 0$.
- The $(1,2)$-entry $A_{1,2}$ is the number of edges from v_1 to v_2, and hence $A_{1,2} = 1$.
- The $(1,3)$-entry $A_{1,3}$ is the number of edges from v_1 to v_3, and hence $A_{1,3} = 0$.
- The $(1,4)$-entry $A_{1,4}$ is the number of edges from v_1 to v_4, and hence $A_{1,4} = 2$.

This tells us that the entries in the first row of A are $0, 1, 0, 2$. We can repeat this same process to obtain the entries of rows $2, 3$ and 4. For instance, the first entry $A_{2,1}$ of row 2 is 1 since there is exactly one edge from v_2 to v_1. The resulting matrix is

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

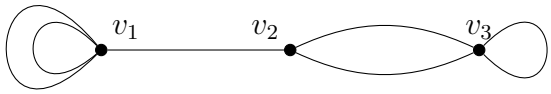
1.2.2 From Matrix to Graph

Executing the above process in reverse allows us to construct a graph from a given adjacency matrix. Consider the matrix

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

If this is the adjacency matrix of a graph G, what does G look like? Well, we know G has 3 vertices since A is a 3×3 matrix. Let's call the vertices v_1, v_2, and v_3. According to the $(1,1)$-entry of A, vertex v_1 has 2 loops, and the $(1,2)$-entry indicates an edge between v_1 and v_2. Since the $(1,3)$-entry is 0, it must be that no edges connect v_1 and v_3.
What about v_2? The $(2,1)$-entry tells us that one edge connects v_1 to v_2, though this was already accounted for. We also know that v_2 has no loops as the $(2,2)$-entry is 0, and there are two

edges from v_2 to v_3. We can carry out this analysis on v_3 to find that the only new edge is a single loop. The graph is drawn below.

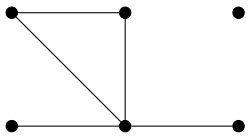


1.3 Summary

- A matrix A is a rectangular grid of numbers. It is size $m \times n$ when there are m rows and n columns. Then entry in row i and column j is denoted $A_{i,j}$ and is called the (i, j)-entry of A.
- To every graph G we can associate a matrix A, called the **adjacency matrix** for G. This matrix contains information that allows us to recover the graph.
- If G is a graph with vertices $v_1, v_2, \ldots, v_n$, it's adjacency matrix A is of size $n \times n$. The (i, j)-entry of A is the number of edges between v_i and v_j.

2 The Havel-Hakimi Algorithm

Let's start with a new definition. If G is a graph with vertices $v_1, v_2, \ldots, v_n$, we define the **degree sequence** of G to be the list $\deg(v_1), \deg(v_2), \ldots, \deg(v_n)$. This list is usually ordered from largest to smallest.
For example, the graph



has degree sequence $(4, 2, 2, 1, 1, 0)$ when ordered from largest to smallest. This graph is **simple** (no loops or multiple edges) and so this example shows that there exists a simple graph with degree sequence $(4, 2, 2, 1, 1, 0)$.
Switching gears, we will attempt answer the following question: If $d = (d_1, d_2, \ldots, d_n)$ is a sequence of non-negative integers ordered from largest to smallest, does there exist a simple graph G with degree sequence d? It's not too hard to see that the answer can sometimes be "no". For example, there is no simple graph with degree sequence $(10, 10, 10)$ since any vertex in G can have degree at most 2. To see a slightly less obvious example, note that there is no simple graph with degree sequence $(2, 2, 1)$ (why?).
It turns out that there is an easy test we can apply to answer the question posed above. An algorithm by Havel and Hakimi from 1955 does exactly this: given $d = (d_1, d_2, \ldots, d_n)$ the algorithm either produces a simple graph with degree sequence d, or tells us that such a graph does not exist.

The Havel-Hakimi Algorithm.

Input: *A sequence of integers $d = (d_1, d_2, \ldots, d_n)$ ordered from largest to smallest.*

Output: *A simple graph G with degree sequence d, or a proof that such a graph does not exist.*

(1) If $d_i \geq n$ or $d_i < 0$ for some i, then stop. ***No such graph exists.***

(2) If $d_i = 0$ for all i, then stop. ***The graph exists.***

(3) If we don't stop at step (1) or step (2), replace d by a new degree sequence by

- *(i) removing the first element d_1,*
- *(ii) subtracting 1 from each of the next d_1 elements, and*
- *(iii) reordering from largest to smallest.*

(4) Run the algorithm again on the new sequence.

2.1 Examples and the Backtracking Process

Let's test our algorithm with the sequence $d = (3, 3, 2, 2, 1, 1)$.

- In the first iteration we remove the first "3" and decrease each of the next 3 terms by 1. The new sequence is $(2, 1, 1, 1, 1)$.
- In the second iteration we remove the first "2" and decrease each of the next 2 terms by 1. The new sequence is $(0, 0, 1, 1)$. We reorder from largest to smallest and continue with the sequence $(1, 1, 0, 0)$
- In the third iteration we remove the first "1" and decrease the next 1 term by 1. The new sequence is $(0, 0, 0)$.
- In the fourth iteration we are stopped at step (2). Since all entries of our sequence is 0, such a simple graph exists!

Now that we know the graph exists, how to we obtain it? Let's look at the steps we took along the way:

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

The idea is to move in reverse: find a graph with degree sequence $(0, 0, 0)$, then backtrack to find a graph with degree sequence $(1, 1, 0, 0)$, then backtrack to find a graph with degree sequence $(0, 0, 1, 1)$, etc. Eventually we arrive at a graph with degree sequence $(3, 3, 2, 2, 1, 1)$. How does the backtracking work?

- **If the previous step was "reduce":** add a new vertex on the left and connect it to each vertex on its right until its degree is correct.
- **If the previous step was "reorder":** reorder your current graph to match the new sequence.

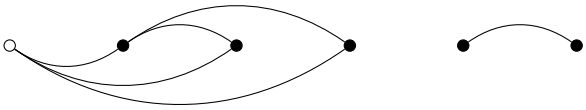
Let's try this procedure on our example. Start with a graph that has degree sequence $(0, 0, 0)$:

We would now like a graph with degree sequence $(1, 1, 0, 0)$. Since the previous step was "reduce", add a vertex on the left and connect it to the first vertex on its right. To see the difference, the new vertex has been temporarily coloured white.

Now to obtain a graph with degree sequence $(0, 0, 1, 1)$. The previous step was "reorder", so instead of adding a vertex we will simply reorder our current graph.

To obtain a graph with degree sequence $(2, 1, 1, 1, 1)$, we note that the previous step was "reduce" and so we add a vertex on the left (the white one) and connect it to the two vertices on its immediate right.

On to the final step! We want a simple graph with degree sequence $(3, 3, 2, 2, 1, 1)$. Since the previous step was "reduce", let's add a vertex on the left and connect it to the next 3 vertices on its right. This concludes the first example.



For a second exmaple, let's try the algorithm on the sequence $(4, 4, 3, 1, 1, 1, 0)$. After 1 iteration we arrive at the sequence $(3, 2, 0, 0, 1, 0)$ and reorder to get $(3, 2, 1, 0, 0, 0)$. Repeat the algorithm to obtain sequence $(1, 0, -1, 0, 0)$. We reorder the sequence as $(1, 0, 0, 0, -1)$ and run the algorithm a final time. Since there is a negative entry, we stop at step (1) and conclude that *no such graph exists*.

As an exercise, try running the algorithm on the sequence $(4, 3, 3, 3, 1)$. You should conclude that there *is* a simple graph with this degree sequence. By running the algorithm in reverse, see if you can construct a graph that looks something like this: