

MAT1830

Lecture 30: Walks, paths and trails

A *walk* of length ℓ in a graph G is a sequence of vertices

$$V_1, V_2, V_3, \dots, V_\ell, V_{\ell+1}$$

where there is an edge of G joining vertex V_i to vertex V_{i+1} for all $i \in \{1, 2, \dots, \ell\}$.

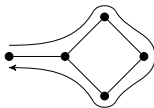
We say that this walk *traverses* the edges $V_1V_2, V_2V_3, \dots, V_\ell V_{\ell+1}$. If $V_{\ell+1} = V_1$ the walk is said to be *closed*. Note that the length of the walk counts the number of “steps” and so is one less than the number of vertices in the sequence.

A *trail* is a walk that traverses each edge at most once.

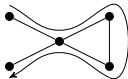
Remembering the definition from the last lecture, a walk that visits each vertex at most once corresponds to a path.

30.1 Examples

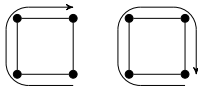
In these pictures, a walk is indicated by a directed curve running alongside the actual edges in the walk.



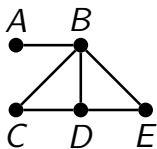
A walk which is not a trail or a path. (Repeated edge, repeated vertex.)



A trail which is not a path. (Repeated vertex.)



A nonclosed walk and a closed walk.



Question In the pictured graph, are the following sequences of vertices walks, trails, paths, or none of these?

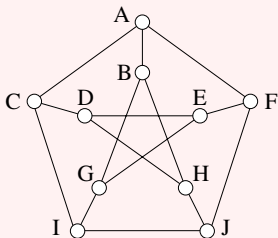
ABDEBD walk (uses *BD* twice)

CDEBA path (and trail and walk, no repeated vertices)

ADEBD none (*AD* isn't an edge in the graph)

ABCDBE trail (and walk, repeats a vertex but not an edge)

Flux Exercise



In the pictured graph, the sequence of vertices DEFJIGE is

- A. A path.
- B. A trail, but not a path.
- C. A walk, but not a trail.
- D. Not a walk.

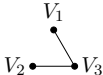
30.2 Adjacency matrix

If two vertices are joined by an edge we say that they are *adjacent*.

A simple graph G with vertices V_1, V_2, \dots, V_n is described by an *adjacency matrix* which has (i, j) entry (i^{th} row and j^{th} column) a_{ij} given by

$$a_{ij} = \begin{cases} 1 & \text{if } V_i \text{ is adjacent to } V_j \text{ in } G, \\ 0 & \text{otherwise.} \end{cases}$$

For example, the graph



has adjacency matrix $\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$

Question

30.1 A graph G has adjacency matrix

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

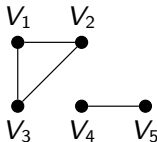
Decide, without drawing the graph, whether G is connected or not. Then draw G .

Answer

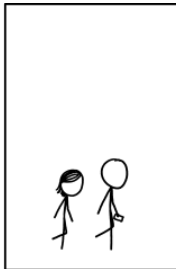
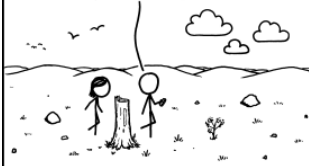
Let the vertex of the graph corresponding to row/column i be V_i .

Looking at the matrix we can see that V_4 and V_5 have an edge between them, but they are not joined to any other vertices.

So the graph must be disconnected.



MY LIFE IS BASICALLY A BIG CONTROLLED TRIAL OF WHETHER I'M MORE LIKELY TO WALK INTO SOMETHING WHILE LOOKING AT A BOOK, MY PHONE, OR THE SKY.



THE WEIRD THING IS THAT THE RATE FOR THE CONTROL GROUP IS SO HIGH.

*WALKING IS
HARD, OKAY?*



30.3 Adjacency matrix powers

The *product* of matrices

$$\begin{pmatrix} a_{11} & a_{12} & \cdots \\ a_{21} & a_{22} & \cdots \\ \cdots & \cdots & \cdots \end{pmatrix} \times \begin{pmatrix} b_{11} & b_{12} & \cdots \\ b_{21} & b_{22} & \cdots \\ \cdots & \cdots & \cdots \end{pmatrix}$$

is the matrix whose (i, j) entry is

$$a_{i1}b_{1j} + a_{i2}b_{2j} + a_{i3}b_{3j} + \cdots ,$$

—the “dot product” of the i th row

$$a_{i1} \quad a_{i2} \quad a_{i3} \quad \cdots$$

of the matrix on the left with the j th column

$$b_{1j}$$

$$b_{2j}$$

$$b_{3j}$$

$$\vdots$$

of the matrix on the right.

Example of matrix multiplication

$$\begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \\ 0 & 3 & 2 \end{pmatrix} \times \begin{pmatrix} 0 & 3 & 1 \\ 2 & 0 & 1 \\ 1 & 2 & 0 \end{pmatrix} = \begin{pmatrix} 7 & 9 & 3 \\ 3 & 5 & 2 \\ 8 & 4 & 3 \end{pmatrix}$$

The red entry on the right comes from the red row and column on the left.

The calculation is $1 \times 3 + 2 \times 0 + 3 \times 2 = 9$.

Note We won't ask you to multiply matrices in assessments for this unit.

Definition The k th power of a matrix A is

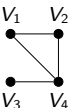
$$A^k = \underbrace{A \times A \times A \times \cdots \times A}_{k \text{ factors}}.$$

The (i, j) entry in the k^{th} power of the adjacency matrix gives the number of walks of length k between V_i and V_j . The *length* of a walk is the number of “steps” (edges) in it.

Fact If M is the adjacency matrix of a graph G with vertices V_1, \dots, V_n , then the (i, j) entry of M^k is the number of walks of length k in G from V_i to V_j .

The fact is true for M^1 . Suppose it is true for M^1, M^2, \dots, M^k .

Example

Let G be the graph . Let M be the adjacency matrix of G .

Why is the $(3, 1)$ entry of M^{k+1} the number of walks of length $k + 1$ from V_3 to V_1 ?

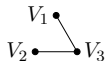
$$M^{k+1} = M^k \times M = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \times \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

where, by assumption a_{ij} is the number of walks of length k from V_i to V_j .

The $(3, 1)$ entry is $a_{31} \times 0 + a_{32} \times 1 + a_{33} \times 0 + a_{34} \times 1 = a_{32} + a_{34}$.

$$\begin{aligned} a_{32} + a_{34} &= (\# \text{ length } k \text{ walks from } V_3 \text{ to } V_2) + (\# \text{ length } k \text{ walks from } V_3 \text{ to } V_4) \\ &= \# \text{ length } k + 1 \text{ walks from } V_3 \text{ to } V_1 \end{aligned}$$

For example, suppose we want the number of walks of length 2 from V_1 to V_3 in the graph



The adjacency matrix M tells us that the following edges exist.

$$\begin{array}{ccc}
 \begin{pmatrix} \cdots & \cdots & 1 \\ \cdots & \cdots & 1 \\ 1 & 1 & 0 \end{pmatrix} & \begin{array}{l} \leftarrow V_1 \text{ to } V_3 \\ \leftarrow V_2 \text{ to } V_3 \end{array} \\
 \begin{array}{cc} \uparrow & \uparrow \\ V_3 & V_3 \\ \text{to} & \text{to} \\ V_1 & V_2 \end{array}
 \end{array}$$

So when we square this matrix, the $(3,3)$ entry in M^2

$$\begin{pmatrix} 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = 1 \times 1 + 1 \times 1 = 2$$

counts the walks from V_3 to V_3 , namely

$$V_3 \rightarrow V_1 \rightarrow V_3 \text{ and } V_3 \rightarrow V_2 \rightarrow V_3.$$

Similarly, the (i, j) entry in M^2 is the number of walks of length 2 from V_i to V_j . The (i, j) entry in M^3 is the number of walks of length 3 from V_i to V_j , and so on.

In fact,

$$M^2 \times M = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix} \times \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

has $(3, 2)$ entry

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

Hence the number of walks of length 3 from V_3 to V_2 is 2.

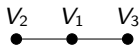
Question

30.2 Draw the graph with adjacency matrix

$$M = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Use V_1 , V_2 and V_3 as names for the vertices corresponding to columns 1, 2 and 3.

Answer



30.3 Without doing any matrix multiplication, find M^3 .

Answer

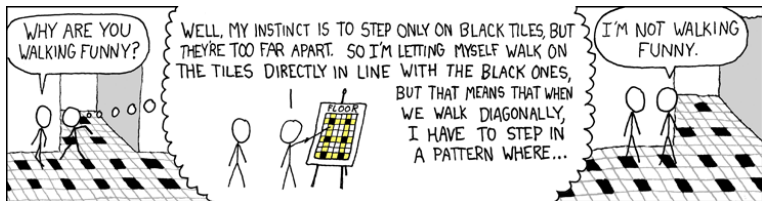
The entry in row i and column j of M^3 is the number of walks of length 3 from i to j .

$$M^3 = \begin{pmatrix} 0 & 2 & 2 \\ 2 & 0 & 0 \\ 2 & 0 & 0 \end{pmatrix}$$

Flux Exercise

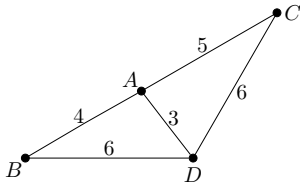
Which of the following statements are true.

- A. A path of length 10 involves exactly 10 different vertices.
- B. In any graph it is possible to have a trail of length 1 million.
- C. In any graph it is possible to have a walk of length 1 million.
- D. All of the above.
- E. None of the above.



30.4 The travelling salesman problem

Given a connected graph in which each edge has been assigned a positive weight, the travelling salesman problem asks us to find a walk of the smallest possible weight that visits every vertex. (The problem's name comes from thinking of the vertices as towns and the edge weights as distances between towns.) For example, $BADC$ is a solution to the travelling salesman problem on the edge-weighted graph given below



Solving problems like the travelling salesman problem is vital for any sort of complex logistical operation.

Remember for earlier in the semester that the travelling salesman problem is **NP-complete**, meaning there may not be an efficient algorithm that can solve every instance of it perfectly.

But this doesn't mean we have to give up on it. We can look for

- ▶ approximation algorithms
- ▶ randomised algorithms
- ▶ algorithms that work well on real-world instances

We also might have to think about more complicated problems involving more real-world constraints.