Activity 1

Multiply the following matrices:

- $$\begin{bmatrix} 1 & 0 \\ 2 & -1 \end{bmatrix} \begin{bmatrix} 3 & 1 & 4 \\ -2 & 0 & 1 \end{bmatrix}$$
- $$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 5 \\ 6 \\ 7 \end{bmatrix}$$
- $$\begin{bmatrix} 5 \\ 6 \\ 7 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

Activity 2

Define the function $f : \mathbb{R}^3 \longrightarrow \mathbb{R}^3$ by $f(\mathbf{x}) = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 0 & 1 \end{bmatrix} \begin{bmatrix} \\ \mathbf{x} \\ \\ \end{bmatrix}$

- Write a rule for f in the form $f(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}) = \begin{bmatrix} ? \\ ? \\ ? \end{bmatrix}$.
- Let g be the inverse of f. Write a rule for g.
- What is the domain of g?
- What is the matrix B such that $g(\mathbf{y}) = B\mathbf{y}$?

Quiz

Let $f : \mathcal{U} \longrightarrow \mathcal{V}$ be a linear transformation. Recall that the *kernel* of f is $\{\mathbf{u} \in \mathcal{U} \ : \ f(u) = \mathbf{0}\}$. This is a subspace of $\mathcal{U}$. We say a vector space is *trivial* if it has only one element, which is necessarily a zero vector since every vector space contains a zero vector.

- Prove that if the kernel of f is not trivial then f is not one-to-one:
  Proof: *Suppose the kernel of f is not trivial. We show that f is not one-to-one....*
- Prove that if the kernel of f is trivial then f is one-to-one.
  Proof: *Suppose the kernel of f is trivial. We show that f is one-to-one....*

[5] The Basis

René Descartes

Born 1596.

After studying law in college,....

> *I entirely abandoned the study of letters. Resolving to seek no knowledge other than that of which could be found in myself or else in the great book of the world, I spent the rest of my youth traveling, visiting courts and armies, mixing with people of diverse temperaments and ranks, gathering various experiences, testing myself in the situations which fortune offered me, and at all times reflecting upon whatever came my way so as to derive some profit from it.*

He had a practice of lying in bed in the morning, thinking about mathematics....

Coordinate systems

In 1618, he had an idea...
while lying in bed and watching a fly on the ceiling.

He could describe the location of the fly in terms of two numbers: its distance from the two walls.

He realized that this works even if the two walls were not perpendicular.

He realized that you could express geometry in algebra.

- The walls play role of what we now call *axes*.
- The two numbers are what we now call *coordinates*

Coordinate systems

In terms of vectors (and generalized beyond two dimensions),

- *coordinate system* for a vector space $\mathcal{V}$ is specified by generators $\mathbf{a}_1, \ldots, \mathbf{a}_n$ of $\mathcal{V}$
- Every vector $\mathbf{v}$ in $\mathcal{V}$ can be written as a linear combination

$$\mathbf{v} = \alpha_1 \mathbf{a}_1 + \cdots + \alpha_n \mathbf{a}_n$$

- We represent vector $\mathbf{v}$ by the vector $[\alpha_1, \ldots, \alpha_n]$ of coefficients.
  called the *coordinate representation* of $\mathbf{v}$ in terms of $\mathbf{a}_1, \ldots, \mathbf{a}_n$.

But assigning coordinates to points is not enough. In order to avoid confusion, we must ensure that each point is assigned coordinates in exactly one way. How?

We will discuss unique representation later.

Coordinate representation

Definition: The *coordinate representation* of $\mathbf{v}$ in terms of $\mathbf{a}_1, \ldots, \mathbf{a}_n$ is the vector $[\alpha_1, \ldots, \alpha_n]$ such that

$$\mathbf{v} = \alpha_1 \mathbf{a}_1 + \cdots + \alpha_n \mathbf{a}_n$$

In this context, the coefficients are called the *coordinates*.

Example: The vector $\mathbf{v} = [1, 3, 5, 3]$ is equal to

$$1\,[1, 1, 0, 0] + 2\,[0, 1, 1, 0] + 3\,[0, 0, 1, 1]$$

so the coordinate representation of $\mathbf{v}$ in terms of the vectors $[1, 1, 0, 0], [0, 1, 1, 0], [0, 0, 1, 1]$ is $[1, 2, 3]$.

Example: What is the coordinate representation of the vector $[6, 3, 2, 5]$ in terms of the vectors $[2, 2, 2, 3], [1, 0, -1, 0], [0, 1, 0, 1]$?

Since

$$[6, 3, 2, 5] = 2\,[2, 2, 2, 3] + 2\,[1, 0, -1, 0] - 1\,[0, 1, 0, 1],$$

the coordinate representation is $[2, 2, -1]$.

Coordinate representation

Definition: The *coordinate representation* of $\mathbf{v}$ in terms of $\mathbf{a}_1, \ldots, \mathbf{a}_n$ is the vector $[\alpha_1, \ldots, \alpha_n]$ such that

$$\mathbf{v} = \alpha_1 \mathbf{a}_1 + \cdots + \alpha_n \mathbf{a}_n$$

In this context, the coefficients are called the *coordinates*.

Now we do an example with vectors over $GF(2)$.

Example: What is the coordinate representation of the vector [0,0,0,1] in terms of the vectors [1,1,0,1], [0,1,0,1], and [1,1,0,0]?

Since

$$[0,0,0,1] = 1\,[1,1,0,1] + 0\,[0,1,0,1] + 1\,[1,1,0,0]$$

the coordinate representation of $[0,0,0,1]$ is $[1,0,1]$.

Coordinate representation

Definition: The *coordinate representation* of $\mathbf{v}$ in terms of $\mathbf{a}_1, \ldots, \mathbf{a}_n$ is the vector $[\alpha_1, \ldots, \alpha_n]$ such that

$$\mathbf{v} = \alpha_1 \mathbf{a}_1 + \cdots + \alpha_n \mathbf{a}_n$$

In this context, the coefficients are called the *coordinates*.

Why put the coordinates in a vector?

Makes sense in view of linear-combinations definitions of matrix-vector multiplication.

Let $A = \left[\begin{array}{c|c|c} & & \\ \mathbf{a}_1 & \cdots & \mathbf{a}_n \\ & & \end{array}\right]$.

- "$\mathbf{u}$ is the coordinate representation of $\mathbf{v}$ in terms of $\mathbf{a}_1, \ldots, \mathbf{a}_n$" can be written as matrix-vector equation $A\mathbf{u} = \mathbf{v}$
- To go from a coordinate representation $\mathbf{u}$ to the vector being represented, we multiply A times $\mathbf{u}$.
- To go from a vector $\mathbf{v}$ to its coordinate representation, we can solve the matrix-vector equation $A\mathbf{x} = \mathbf{v}$.

Linear Combinations: Lossy compression

Say you need to store or transmit many 2-megapixel images:

How do we represent the image compactly?

- *Obvious method:* 2 million pixels $\Longrightarrow$ 2 million numbers
- *Strategy 1:* Use sparsity! Find the "nearest" k-sparse vector. Later we'll see this consists of suppressing all but the largest k entries.
- *More sophisticated strategy?*

Linear Combinations: Lossy compression

Strategy 2: Represent image vector by its coordinate representation:

- Before compressing any images, select vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$.
- Replace each image vector with its coordinate representation in terms of $\mathbf{v}_1, \ldots, \mathbf{v}_n$.

For this strategy to work, we need to ensure that *every* image vector can be represented as a linear combination of $\mathbf{v}_1, \ldots, \mathbf{v}_n$.

Given some D-vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ over $\mathbb{F}$, how can we tell whether *every* vector in $\mathbb{F}^D$ can be written as a linear combination of $\mathbf{v}_1, \ldots, \mathbf{v}_n$?

We also need the number of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ to be much smaller than the number of pixels.

Given D, what is minimum number of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ such that every vector in $\mathbb{F}^D$ can be written as a linear combination?

Linear Combinations: Lossy compression

Strategy 3: A hybrid approach

Step 1: Select vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$.

Step 2: For each image to compress, find its coordinate representation $\mathbf{u}$ in terms of $\mathbf{v}_1, \ldots, \mathbf{v}_n$

Step 3: Replace $\mathbf{u}$ with the closest k-sparse vector $\tilde{\mathbf{u}}$, and store $\tilde{\mathbf{u}}$.

Step 4: To recover an image from $\tilde{\mathbf{u}}$, calculate the corresponding linear combination of $\mathbf{v}_1, \ldots \mathbf{v}_n$.

Greedy algorithms for finding a set of generators

Question: *For a given vector space $\mathcal{V}$, what is the minimum number of vectors whose span equals $\mathcal{V}$?*

How can we obtain a minimum number of vectors?

Two natural approaches come to mind, the *Grow* algorithm and the *Shrink* algorithm.

Grow algorithm

def GROW($\mathcal{V}$)
  $S = \emptyset$
  repeat while possible:
    find a vector **v** in $\mathcal{V}$ that is not in Span S, and put it in S.

The algorithm stops when there is no vector to add, at which time S spans all of $\mathcal{V}$. Thus, if the algorithm stops, it will have found a generating set.

But is it bigger than necessary?

Shrink Algorithm

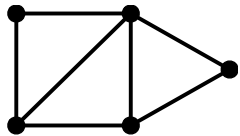
def SHRINK($\mathcal{V}$)
  S = some finite set of vectors that spans $\mathcal{V}$
  repeat while possible:
    find a vector **v** in S such that Span $(S - \{v\}) = \mathcal{V}$, and remove **v** from S.

The algorithm stops when there is no vector whose removal would leave a spanning set.
At every point during the algorithm, S spans $\mathcal{V}$, so it spans $\mathcal{V}$ at the end.
Thus, if the algorithm stops, the algorithm will have found a generating set.

The question is, again: is it bigger than necessary?

When greed fails

Is it obvious that Grow algorithm and Shrink algorithm find smallest sets of generators? Look at example for a problem in *graphs*...

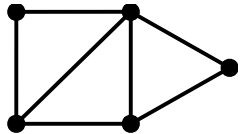


Points are called *nodes*, links are called *edges*.
Each edge has two *endpoints*, the nodes it connects. The endpoints of an edge are *neighbors*.

Definition: A *dominating set* in a graph is a set S of nodes such that every node is in S or a neighbor of a node in S.

When greed fails: dominating set

Definition: A *dominating set* in a graph is a set S of nodes such that every node is in S or a neighbor of a node in S.

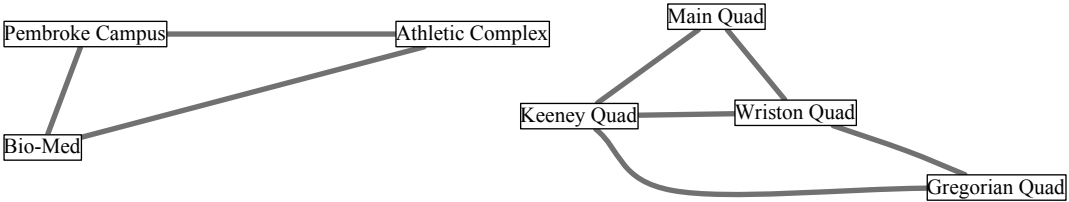


Grow Algorithm:
initialize $S = \emptyset$
while S is not a dominating set,
    add a node to S that is not currently adjacent to S

Shrink Algorithm:
initialize $S =$ all nodes
while there is a node x such that $S - \{x\}$ is a dominating set,
    remove x from S

Neither algorithm is guaranteed to find the smallest solution.

Minimum spanning forest



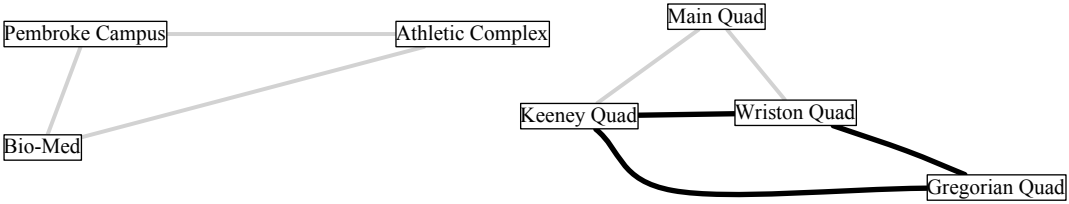
Definition: A sequence of edges $[\{x_1, x_2\}, \{x_2, x_3\}, \{x_3, x_4\}, \ldots, \{x_{k-1}, x_k\}]$ with no repeats is called an x_1-to-x_k *path*.

Example "Main Quad"-to-"Gregorian Quad" paths in above graph:

- one goes through "Wriston Quad" ,
- one goes through "Keeney Quad"

Definition: A x-to-x path is called a *cycle*.

Minimum spanning forest



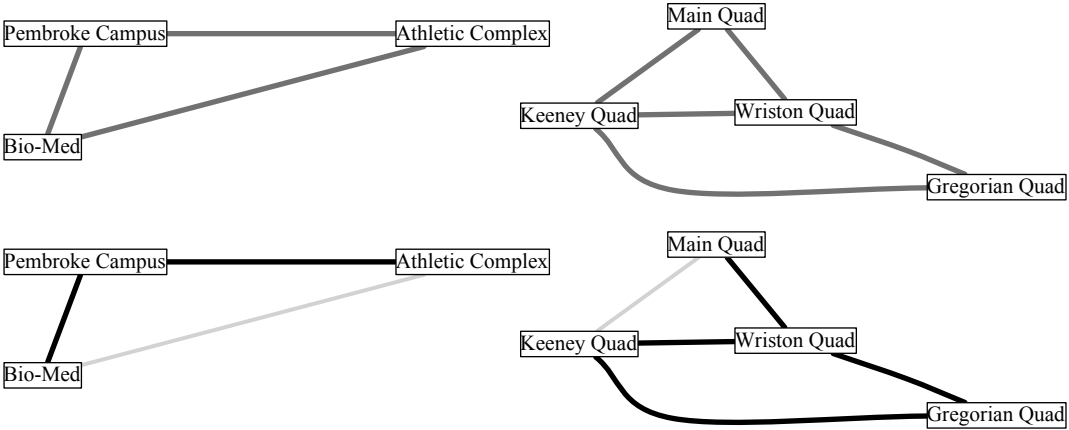
Definition: A sequence of edges $[\{x_1, x_2\}, \{x_2, x_3\}, \{x_3, x_4\}, \ldots, \{x_{k-1}, x_k\}]$ with no repeats is called an x_1-to-x_k *path*.

Example "Main Quad"-to-"Gregorian Quad" paths in above graph:

- one goes through "Wriston Quad" ,
- one goes through "Keeney Quad"

Definition: A x-to-x path is called a *cycle*.

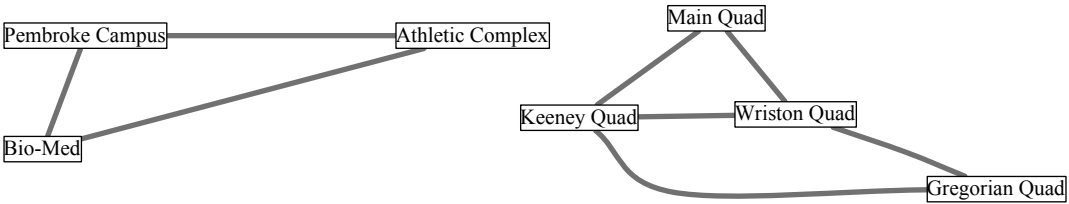
Minimum spanning forest: spanning



Definition: A set S of edges is *spanning* for a graph G if, for every edge $\{x, y\}$ of G, there is an x-to-y path consisting of edges of S.

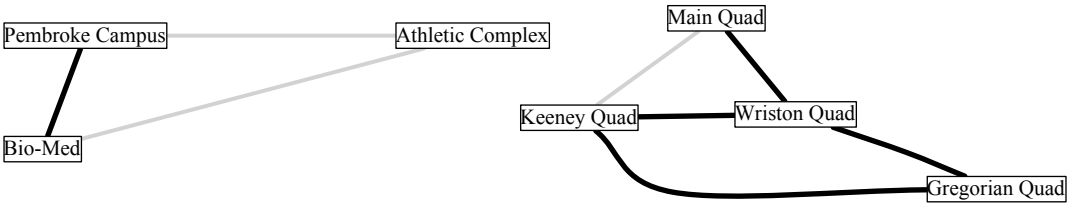
Soon we see connection between this use of "spanning" and its use with vectors.

Minimum spanning forest: forest



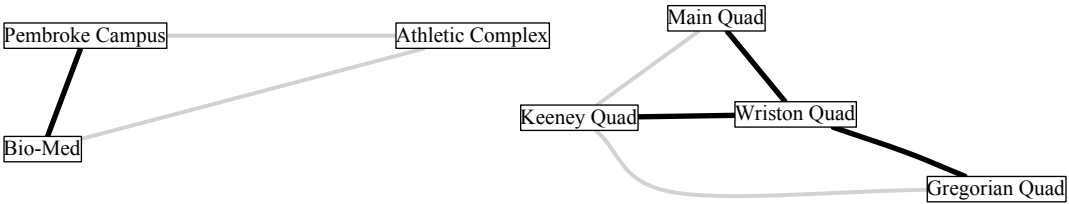
Definition: A set of edges of G is a *forest* if the set includes no cycles.

Minimum spanning forest: forest



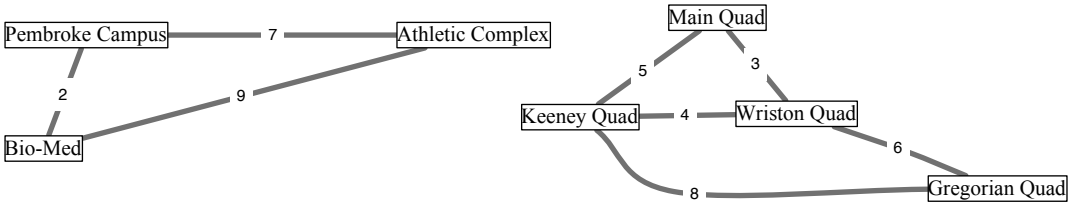
Definition: A set of edges of G is a *forest* if the set includes no cycles.

Minimum spanning forest: forest



Definition: A set of edges of G is a *forest* if the set includes no cycles.

Minimum spanning forest



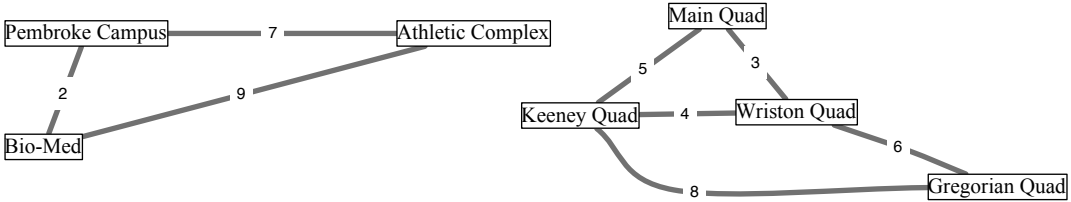
Minimum spanning forest problem:

- *input:* a graph G, and an assignment of real-number *weights* to the edges of G.
- *output:* a minimum-weight set S of edges that is spanning and a forest.

Application: Design hot-water delivery network for the university campus:

- Network must achieve same connectivity as input graph.
- An edge represents a possible pipe.
- Weight of edge is cost of installing the pipe.
- Goal: minimize total cost.

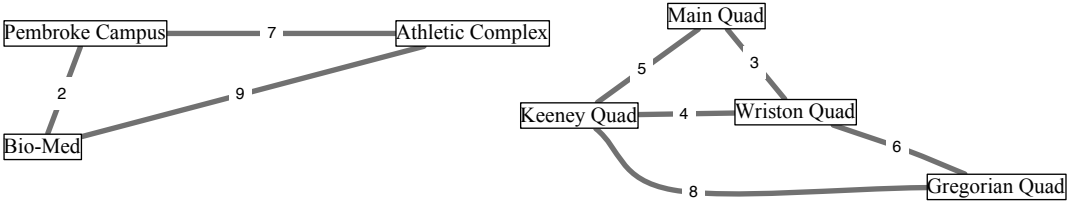
Minimum spanning forest: Grow algorithm




```
def GROW(G)
    S := ∅
    consider the edges in increasing order
    for each edge e:
        if e's endpoints are not yet connected
            add e to S.
```

Increasing order: 2, 3, 4, 5, 6, 7, 8, 9.

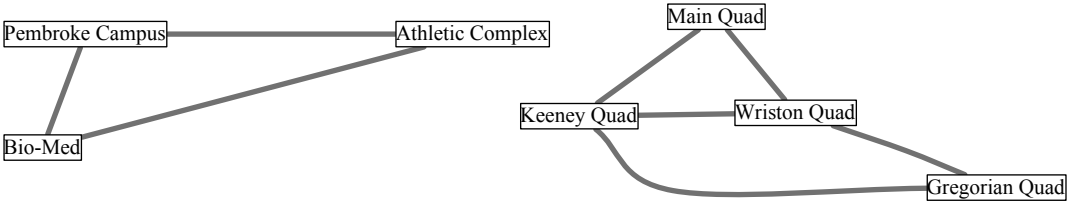
Minimum spanning forest: Shrink algorithm




```
def SHRINK(G)
    S = {all edges}
    consider the edges in order, from highest-weight to lowest-weight
    for each edge e:
        if every pair of nodes are connected via S − {e}:
            remove e from S.
```

Decreasing order: 9, 8, 7, 6, 5, 4, 3, 2.

Formulating *Minimum Spanning Forest* in linear algebra

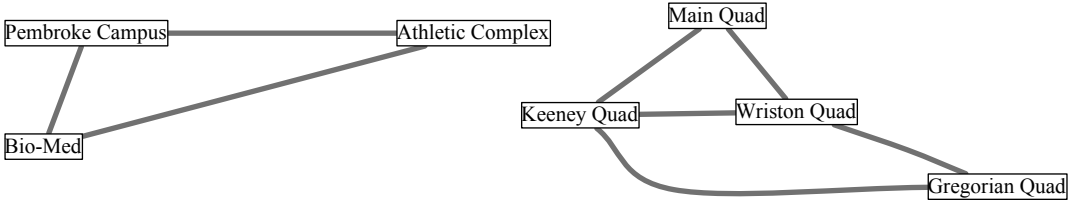


Let D = set of nodes {Pembroke, Athletic, Bio-Med, Gregorian, Main, Keeney, Wriston}

Represent a subset of D by a $GF(2)$ vector:
subset {`Pembroke, Main, Gregorian`} is represented by

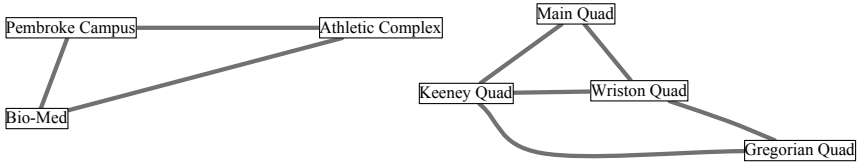
`Pembroke`	`Athletic`	`Bio-Med`	`Main`	`Keeney`	`Wriston`	`Gregorian`
1			1			1

Formulating *Minimum Spanning Forest* in linear algebra



edge	vector						
	Pembroke	Athletic	Bio-Med	Main	Keeney	Wriston	Gregorian
{Pembroke, Athletic}	1	1					
{Pembroke, Bio-Med}	1		1				
{Athletic, Bio-Med}		1	1				
{Main, Keeney}				1	1		
{Main, Wriston}				1		1	
{Keeney, Wriston}					1	1	
{Keeney, Gregorian}					1		1
{Wriston, Gregorian}						1	1

Formulating *Minimum Spanning Forest* in linear algebra



The vector representing {`Keeney, Gregorian`},

`Pembroke`	`Athletic`	`Bio-Med`	`Main`	`Keeney`	`Wriston`	`Gregorian`
				1		1

is the sum, for example, of the vectors representing {`Keeney, Main `}, {`Main, Wriston`}, and {`Wriston, Gregorian`} :

`Pembroke`	`Athletic`	`Bio-Med`	`Main`	`Keeney`	`Wriston`	`Gregorian`
			1	1		
			1		1	
					1	1

A vector with 1's in entries x and y is the sum of vectors corresponding to edges that form an x-to-y path in the graph.