[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 $A\mathbf{x} = \mathbf{v}$.
  (Because the columns of A are generators for $\mathcal{V}$ and $\mathbf{v}$ belongs to $\mathcal{V}$, the equation must have at least one solution.)

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 $\textsc{Grow}(\mathcal{V})$
  $S = \emptyset$
  repeat while possible:
    find a vector $\mathbf{v}$ 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

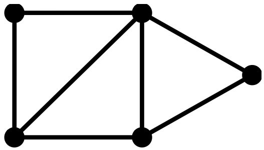
def $\text{SHRINK}(\mathcal{V})$
  $S =$ some finite set of vectors that spans $\mathcal{V}$
  repeat while possible:
    find a vector $\mathbf{v}$ in S such that Span $(S - \{v\}) = \mathcal{V}$, and remove $\mathbf{v}$ from S.

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*...

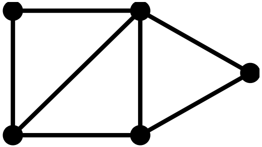


Points are called *nodes*, links are called *edges*.
Each edge has two *endpoints*, the nodes it connects. The endpoints of an edge are *neighbors*.

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.

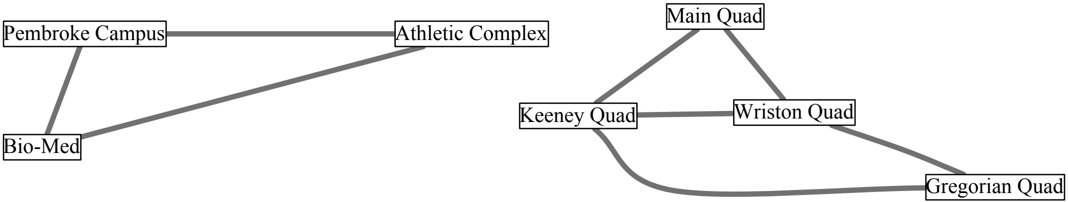


Grow Algorithm:
initialize $S = \emptyset$
while S is not a dominating set,
    add a node to S that is not currently adjacent to S

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



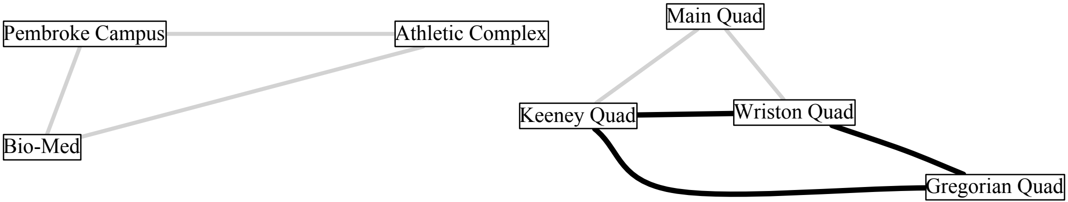
Definition: A sequence of edges $[\{x_1, x_2\}, \{x_2, x_3\}, \{x_3, x_4\}, \ldots, \{x_{k-1}, x_k\}]$ with no repeats is called an x_1-to-x_k *path*.

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



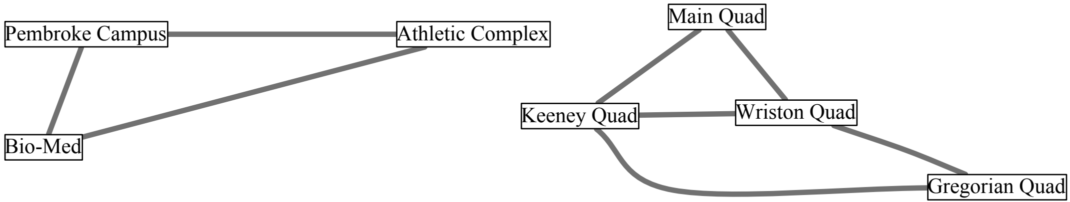
Definition: A sequence of edges $[\{x_1, x_2\}, \{x_2, x_3\}, \{x_3, x_4\}, \ldots, \{x_{k-1}, x_k\}]$ with no repeats is called an x_1-to-x_k *path*.

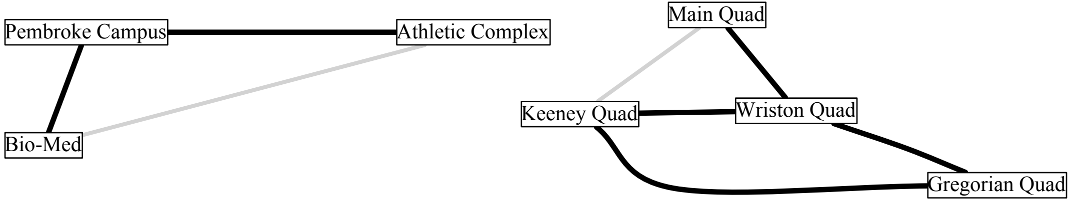
Example "Main Quad"-to-"Gregorian Quad" paths in above graph:

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

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

Minimum spanning forest: spanning

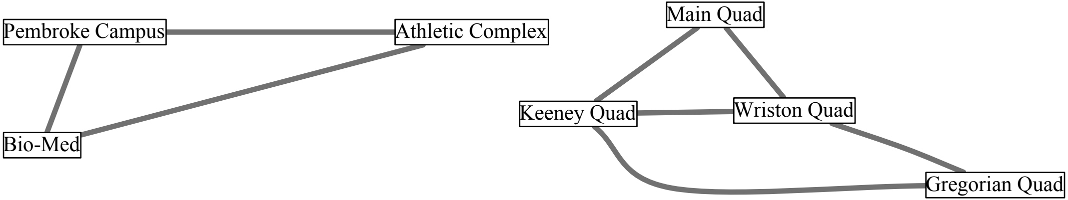




Definition: A set S of edges is *spanning* for a graph G if, for every edge $\{x, y\}$ of G, there is an x-to-y path consisting of edges of S.

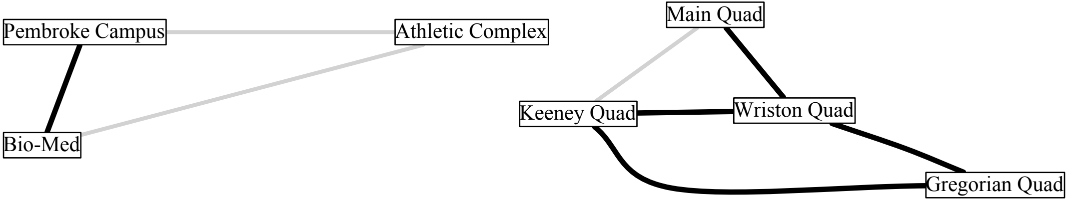
Soon we see connection between this use of "spanning" and its use with vectors.

Minimum spanning forest: forest



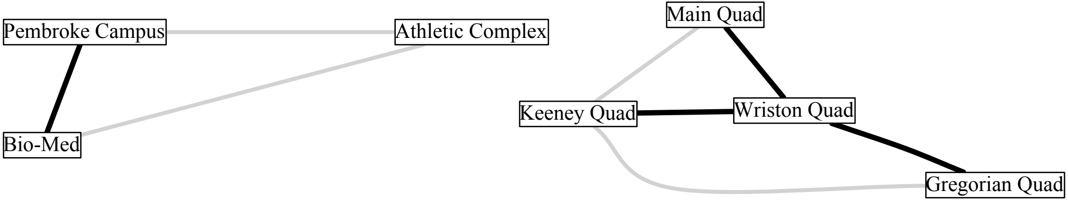
Definition: A set of edges of G is a *forest* if the set includes no cycles.

Minimum spanning forest: forest



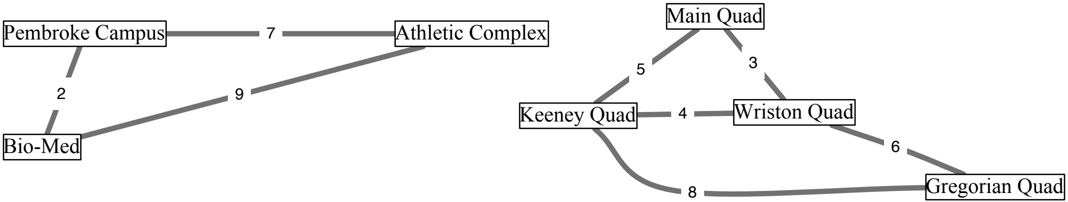
Definition: A set of edges of G is a *forest* if the set includes no cycles.

Minimum spanning forest: forest



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

Minimum spanning forest



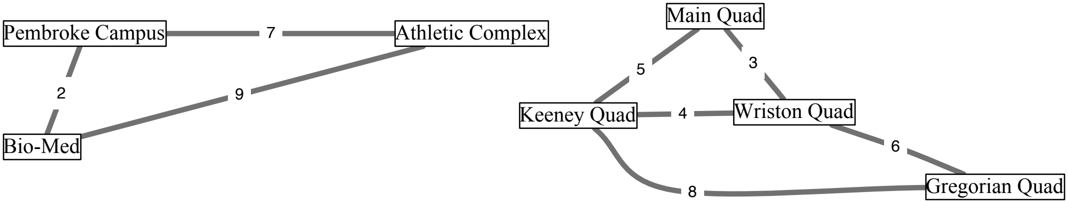
Minimum spanning forest problem:

- *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.

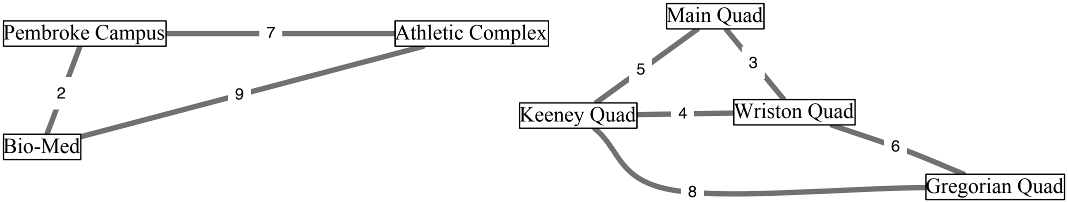
Minimum spanning forest: Grow algorithm



def $\textsc{Grow}(G)$
$S := \emptyset$
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.

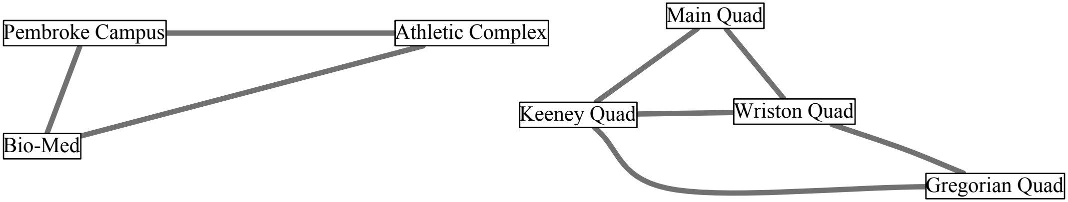
Minimum spanning forest: Shrink algorithm



def $\textsc{Shrink}(G)$
  $S = \{\text{all edges}\}$
  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

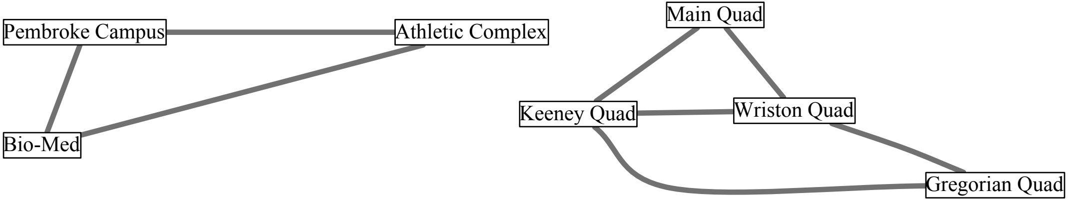


Let D = set of nodes {Pembroke, Athletic, Main, Keeney, Wriston}

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

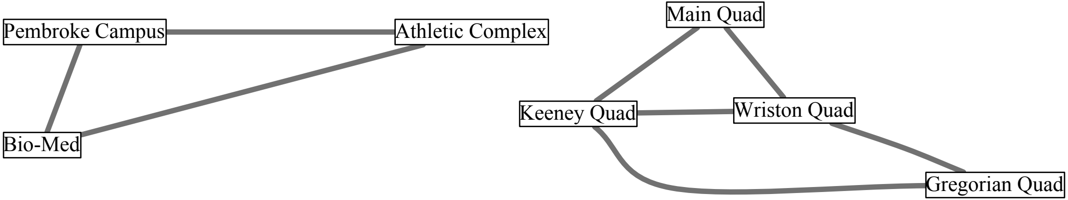
`Pembroke`	`Athletic`	`Bio-Med`	`Main`	`Keeney`	`Wriston`	`Gregorian`
1			1			1

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`},

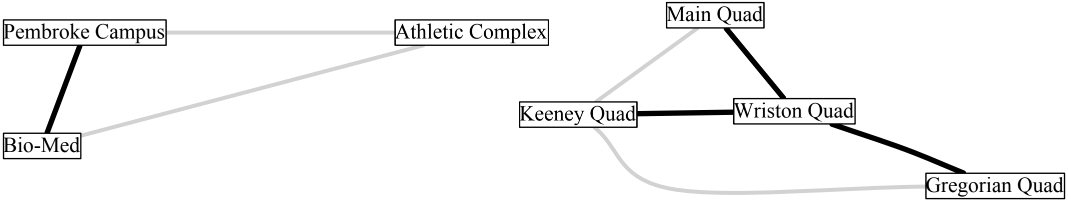
`Pembroke`	`Athletic`	`Bio-Med`	`Main`	`Keeney`	`Wriston`	`Gregorian`
				1		1

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.

Formulating *Minimum Spanning Forest* in linear algebra



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.

Example: The span of the vectors representing

{`Pembroke, Bio-Med`}, {`Main, Wriston`}, {`Keeney, Wriston`}, {`Wriston, Gregorian`}

- contains the vectors corresponding to

  {`Main, Keeney`}, {`Keeney, Gregorian`}, and {`Main, Gregorian`}

- but not the vectors corresponding to

  {`Athletic, Bio-Med`} or {`Bio-Med, Main`}.

Grow algorithms

def $\text{Grow}(G)$
    $S := \emptyset$
    consider the edges in increasing order
    for each edge e:
        if e's endpoints are not yet connected
            add e to S.

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

- Considering edges e of G corresponds to considering vectors $\mathbf{v}$ in $\mathcal{V}$
- Testing if e's endpoints are not connected corresponds to testing if $\mathbf{v}$ is not in Span S.

The Grow algorithm for MSF is a specialization of the Grow algorithm for vectors.

Same for the Shrink algorithms.

Linear Dependence: The Superfluous-Vector Lemma

Grow and Shrink algorithms both test whether a vector is superfluous in spanning a vector space $\mathcal{V}$. Need a criterion for superfluity.

Superfluous-Vector Lemma: For any set S and any vector $\mathbf{v} \in S$, if $\mathbf{v}$ can be written as a linear combination of the other vectors in S then Span $(S - \{\mathbf{v}\}) =$ Span S

Proof: Let $S = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$. Suppose $\mathbf{v}_n = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \cdots + \alpha_{n-1} \mathbf{v}_{n-1}$

To show: every vector in Span S is also in Span $(S - \{\mathbf{v}_n\})$.

Every vector $\mathbf{v}$ in Span S can be written as $\mathbf{v} = \beta_1 \mathbf{v}_1 + \beta_2 \mathbf{v}_2 + \cdots \beta_n \mathbf{v}_n$

Substituting for $\mathbf{v}_n$, we obtain

$$\begin{aligned} \mathbf{v} &= \beta_1 \mathbf{v}_1 + \beta_2 \mathbf{v}_2 + \cdots + \beta_n \left(\alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \cdots + \alpha_{n-1} \mathbf{v}_{n-1}\right) \\ &= (\beta_1 + \beta_n \alpha_1)\mathbf{v}_1 + (\beta_2 + \beta_n \alpha_2)\mathbf{v}_2 + \cdots + (\beta_{n-1} + \beta_n \alpha_{n-1})\mathbf{v}_{n-1} \end{aligned}$$

which shows that an arbitrary vector in Span S can be written as a linear combination of vectors in $S - \{\mathbf{v}_n\}$ and is therefore in Span $(S - \{\mathbf{v}_n\})$.

QED

Defining linear dependence

Definition: Vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ are *linearly dependent* if the zero vector can be written as a **nontrivial** linear combination of the vectors:

$$\mathbf{0} = \alpha_1 \mathbf{v}_1 + \cdots + \alpha_n \mathbf{v}_n$$

In this case, we refer to the linear combination as a *linear dependency* in $\mathbf{v}_1, \ldots, \mathbf{v}_n$.

On the other hand, if the *only* linear combination that equals the zero vector is the trivial linear combination, we say $\mathbf{v}_1, \ldots, \mathbf{v}_n$ are linearly *in*dependent.

Example: The vectors $[1, 0, 0]$, $[0, 2, 0]$, and $[2, 4, 0]$ are linearly dependent, as shown by the following equation:

$$2\,[1, 0, 0] + 2\,[0, 2, 0] - 1\,[2, 4, 0] = [0, 0, 0]$$

Therefore:

$2\,[1, 0, 0] + 2\,[0, 2, 0] - 1\,[2, 4, 0]$ is a linear dependency in $[1, 0, 0]$, $[0, 2, 0]$, $[2, 4, 0]$.

Linear dependence

Example: The vectors $[1, 0, 0]$, $[0, 2, 0]$, and $[0, 0, 4]$ are linearly independent.
How do we know?
Easy since each vector has a nonzero entry where the others have zeroes.
Consider any linear combination

$$\alpha_1 \, [1, 0, 0] + \alpha_2 \, [0, 2, 0] + \alpha_3 \, [0, 0, 4]$$

This equals $[\alpha_1, 2\alpha_2, 4\alpha_3]$

If this is the zero vector, it must be that $\alpha_1 = \alpha_2 = \alpha_3 = 0$
That is, the linear combination is trivial.

We have shown the only linear combination that equals the zero vector is the trivial linear combination.

Linear dependence in relation to other questions

$$\boxed{\text{How can we tell if vectors } \mathbf{v}_1, \ldots, \mathbf{v}_n \text{ are linearly dependent?}}$$

Definition: Vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ are *linearly dependent* if the zero vector can be written as a nontrivial linear combination $\mathbf{0} = \alpha_1 \mathbf{v}_1 + \cdots + \alpha_n \mathbf{v}_n$

By linear-combinations definition, $\mathbf{v}_1, \ldots, \mathbf{v}_n$ are linearly dependent iff there is a nonzero vector $\begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix}$ such that $\left[\begin{array}{c|c|c} \mathbf{v}_1 & \cdots & \mathbf{v}_n \end{array}\right] \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \mathbf{0}$

Therefore, $\mathbf{v}_1, \ldots, \mathbf{v}_n$ are linearly dependent iff the null space of the matrix is nontrivial.

This shows that the question

$$\boxed{\text{How can we tell if vectors } \mathbf{v}_1, \ldots, \mathbf{v}_n \text{ are linearly dependent?}}$$

is the same as a question we asked earlier:

$$\boxed{\text{How can we tell if the null space of a matrix is trivial?}}$$

Linear dependence in relation to other questions

The question

$$\boxed{\text{How can we tell if vectors } \mathbf{v}_1, \ldots, \mathbf{v}_n \text{ are linearly dependent?}}$$

is the same as a question we asked earlier:

$$\boxed{\text{How can we tell if the null space of a matrix is trivial?}}$$

Recall: *solution set of a homogeneous linear system*
$$\begin{array}{rcl} \mathbf{a}_1 \cdot \mathbf{x} & = & 0 \\ & \vdots & \\ \mathbf{a}_m \cdot \mathbf{x} & = & 0 \end{array}$$

is the null space of matrix $\begin{bmatrix} \mathbf{a}_1 \\ \hline \vdots \\ \hline \mathbf{a}_m \end{bmatrix}$.

So question is same as:

$$\boxed{\text{How can we tell if the solution set of a homogeneous linear system is trivial?}}$$

Linear dependence in relation to other questions

The question

$$\boxed{\text{How can we tell if vectors } \mathbf{v}_1, \ldots, \mathbf{v}_n \text{ are linearly dependent?}}$$

is the same as a question we asked earlier:

$$\boxed{\text{How can we tell if the null space of a matrix is trivial?}}$$

is the same as :

$$\boxed{\text{How can we tell if the solution set of a homogeneous linear system is trivial?}}$$

Recall:

If $\mathbf{u}_1$ is a solution to a linear system $\mathbf{a}_1 \cdot \mathbf{x} = \beta_1, \ldots, \mathbf{a}_m \cdot \mathbf{x} = \beta_m$ then

$$\{\text{solutions to linear system}\} = \{\mathbf{u}_1 + \mathbf{v} : \mathbf{v} \in \mathcal{V}\}$$

where $\mathcal{V} = \{$solutions to corresponding homogeneous linear system
$$\mathbf{a}_1 \cdot \mathbf{x} = 0, \ldots, \mathbf{a}_m \cdot \mathbf{x} = 0\}$$

Thus the question is the same as:

$$\boxed{\text{How can we tell if a solution } \mathbf{u}_1 \text{ to a linear system is the } \textit{only} \text{ solution?}}$$

Linear dependence and null space

The question

$$\boxed{\text{How can we tell if vectors } \mathbf{v}_1, \ldots, \mathbf{v}_n \text{ are linearly dependent?}}$$

is the same as:

$$\boxed{\text{How can we tell if the null space of a matrix is trivial?}}$$

is the same as:

$$\boxed{\text{How can we tell if the solution set of a homogeneous linear system is trivial?}}$$

is the same as:

$$\boxed{\text{How can we tell if a solution } \mathbf{u}_1 \text{ to a linear system is the } \textit{only} \text{ solution?}}$$

Linear dependence

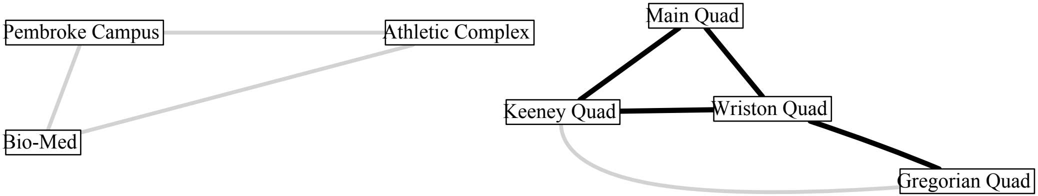
Answering these questions requires an algorithm.

Computational Problem: *Testing linear dependence*

- *input:* a list $[\mathbf{v}_1, \ldots, \mathbf{v}_n]$ of vectors
- *output:* DEPENDENDENT if the vectors are linearly dependent, and INDEPENDENT otherwise.

We'll see two algorithms later.

Linear dependence in *Minimum Spanning Forest*



We can get the zero vector by adding together vectors corresponding to edges that form a cycle: in such a sum, for each entry x, there are exactly two vectors having 1's in position x.

Example: the vectors corresponding to

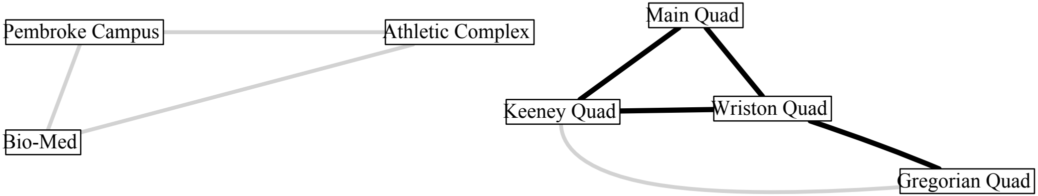
$$\{\texttt{Main, Wriston}\},\{\texttt{Main, Keeney}\}\ \{\texttt{Keeney, Wriston }\},$$

are as follows:

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

The sum of these vectors is the zero vector.

Linear dependence in *Minimum Spanning Forest*



Sum of vectors corresponding to edges forming a cycle can make a zero vector.
Therefore if a subset of S form a cycle then S is linearly dependent.

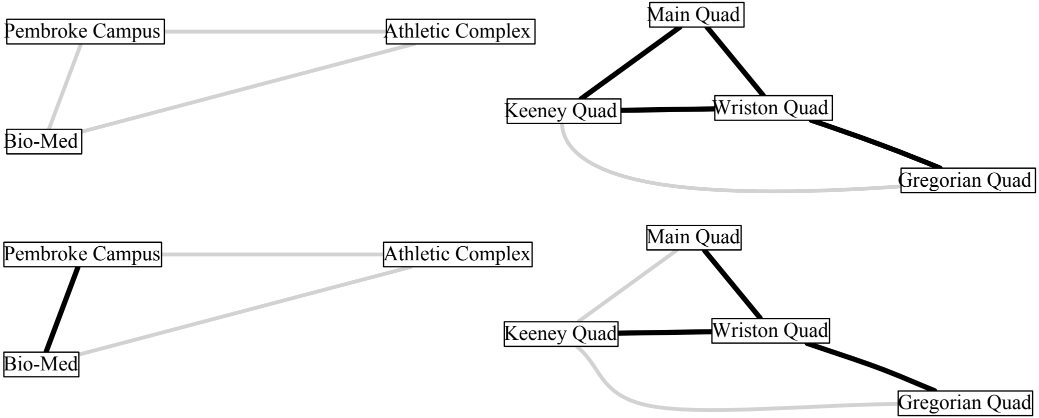
Example: The vectors corresponding to

{`Main, Keeney`}, {`Main, Wriston`}, {`Keeney, Wriston `}, {`Wriston, Gregorian`}
are linearly dependent because these edges include a cycle.

The zero vector is equal to the nontrivial linear combination

			Pembroke	Athletic	Bio-Med	Main	Keeney	Wriston	Gregorian
	1	*				1	1		
+	1	*				1		1	
+	1	*					1	1	
+	0	*						1	1

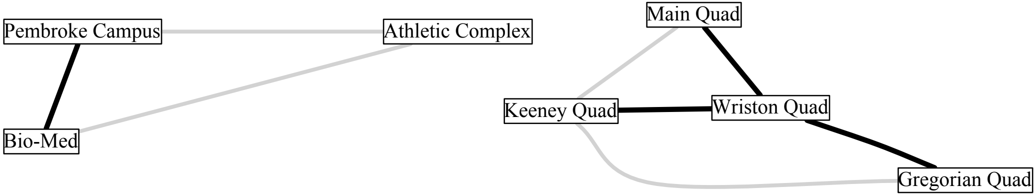
Linear dependence in *Minimum Spanning Forest*



If a subset of S form a cycle then S is linearly dependent.

On the other hand, if a set of edges contains no cycle (i.e. is a forest) then the corresponding set of vectors is linearly independent.

“Quiz”



Which edges are spanned?
Which sets are linearly dependent?

Review

Superfluous-Vector Lemma: For any set S and any vector $\mathbf{v} \in S$, if $\mathbf{v}$ can be written as a linear combination of the other vectors in S then Span $(S - \{\mathbf{v}\}) =$ Span S

Definition: Vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ are *linearly dependent* if the zero vector can be written as a **nontrivial** linear combination of the vectors:

$$\mathbf{0} = \alpha_1 \mathbf{v}_1 + \cdots + \alpha_n \mathbf{v}_n$$

In this case, we refer to the linear combination as a *linear dependency* in $\mathbf{v}_1, \ldots, \mathbf{v}_n$.

On the other hand, if the *only* linear combination that equals the zero vector is the trivial linear combination, we say $\mathbf{v}_1, \ldots, \mathbf{v}_n$ are linearly *in*dependent.

Properties of linear independence: hereditary

Lemma: If a finite set S of vectors is linearly dependent and S is a subset of T then T is linearly dependent.

In graphs, if a set S of edges includes a cycle then a superset of S also includes a cycle.

Properties of linear independence: hereditary

Lemma: If a finite set S of vectors is linearly dependent and S is a subset of T then T is linearly dependent.
Proof: If the zero vector can be written as a nontrivial linear combination of some vectors, it can be so written even if we allow some extra vectors to be in the linear combination because we can use zero coefficients on the extra vectors.

More formal proof: Write $S = \{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$ and $T = \{\mathbf{s}_1, \ldots, \mathbf{s}_n, \mathbf{t}_1, \ldots, \mathbf{t}_k\}$. Suppose S is linearly dependent. Then there are coefficients $\alpha_1, \ldots, \alpha_n$, not all zero, such that

$$\mathbf{0} = \alpha_1 \mathbf{s}_1 + \cdots + \alpha_n \mathbf{s}_n$$

Therefore

$$\mathbf{0} = \alpha_1 \mathbf{s}_1 + \cdots + \alpha_n \mathbf{s}_n + 0\,\mathbf{t}_1 + \cdots 0\,\mathbf{t}_k$$

which shows that the zero vector can be written as a nontrivial linear combination of the vectors of T,i.e. that T is linearly dependent.
QED

Properties of linear (in)dependence

Linear-Dependence Lemma Let $\mathbf{v}_1, \ldots, \mathbf{v}_n$ be vectors.

A vector $\mathbf{v}_i$ is in the span of the other vectors

if and only if

the zero vector can be written as a linear combination of $\mathbf{v}_1, \ldots, \mathbf{v}_n$

in which the coefficient of $\mathbf{v}_i$ is nonzero.

In graphs, the Linear-Dependence Lemma states that an edge e is in the span of other edges if there is a cycle consisting of e and a subset of the other edges.

Properties of linear (in)dependence

Linear-Dependence Lemma Let $\mathbf{v}_1, \ldots, \mathbf{v}_n$ be vectors.

A vector $\mathbf{v}_i$ is in the span of the other vectors
if and only if
the zero vector can be written as a linear combination of $\mathbf{v}_1, \ldots, \mathbf{v}_n$
in which the coefficient of $\mathbf{v}_i$ is nonzero.

Proof: *First direction:* Suppose $\mathbf{v}_i$ is in the span of the other vectors. That is, there exist coefficients $\alpha_1, \ldots, \alpha_{n-1}$ such that

$$\mathbf{v}_i = \alpha_1 \mathbf{v}_1 + \cdots + \alpha_{i-1}\mathbf{v}_{i-1} + \alpha_{i+1}\mathbf{v}_{i+1} + \cdots \alpha_n \mathbf{v}_n$$

Moving $\mathbf{v}_i$ to the other side, we can write

$$\mathbf{0} = \alpha_1 \mathbf{v}_1 + \cdots + (-1)\mathbf{v}_i + \cdots + \alpha_n \mathbf{v}_n$$

which shows that the all-zero vector can be written as a linear combination of $\mathbf{v}_1, \ldots, \mathbf{v}_n$ in which the coefficient of $\mathbf{v}_i$ is nonzero.

Properties of linear (in)dependence

Linear-Dependence Lemma Let $\mathbf{v}_1, \ldots, \mathbf{v}_n$ be vectors.

A vector $\mathbf{v}_i$ is in the span of the other vectors
if and only if
the zero vector can be written as a linear combination of $\mathbf{v}_1, \ldots, \mathbf{v}_n$
in which the coefficient of $\mathbf{v}_i$ is nonzero.

Proof: *Now for the other direction.* Suppose there are coefficients $\alpha_1, \ldots, \alpha_n$ such that

$$\mathbf{0} = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \cdots + \alpha_i \mathbf{v}_i + \cdots + \alpha_n \mathbf{v}_n$$

and such that $\alpha_i \neq 0$.

Dividing both sides by α_i yields

$$\mathbf{0} = (\alpha_1/\alpha_i)\,\mathbf{v}_1 + (\alpha_2/\alpha_i)\,\mathbf{v}_2 + \cdots + \mathbf{v}_i + \cdots + (\alpha_n/\alpha_i)\,\mathbf{v}_n$$

Moving every term from right to left except $\mathbf{v}_i$ yields

$$-(\alpha_1/\alpha_i)\,\mathbf{v}_1 - (\alpha_2/\alpha_i)\,\mathbf{v}_2 - \cdots - (\alpha_n/\alpha_i)\,\mathbf{v}_n = \mathbf{v}_i$$

QED

Properties of linear (in)dependence

Linear-Dependence Lemma Let $\mathbf{v}_1, \ldots, \mathbf{v}_n$ be vectors. A vector $\mathbf{v}_i$ is in the span of the other vectors if and only if the zero vector can be written as a linear combination of $\mathbf{v}_1, \ldots, \mathbf{v}_n$ in which the coefficient of $\mathbf{v}_i$ is nonzero.

Contrapositive:

$\mathbf{v}_i$ is *not* in the space of the other vectors

if and only if

for any linear combination equaling the zero vector

$$\mathbf{0} = \alpha_1 \mathbf{v}_1 + \cdots + \alpha_i \mathbf{v}_i + \cdots + \alpha_n \mathbf{v}_n$$

it must be that the coefficient α_i *is* zero.

Analyzing the Grow algorithm

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

Grow-Algorithm Corollary: The vectors obtained by the Grow algorithm are linearly independent.

In graphs, this means that the solution obtained by the Grow algorithm has no cycles (is a forest).

Analyzing the Grow algorithm

Grow-Algorithm Corollary: The vectors obtained by the Grow algorithm are linearly independent.

Proof: For $n = 1, 2, \ldots$, let $\mathbf{v}_n$ be the vector added to S in the n^{th} iteration of the Grow algorithm. We show by induction that $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ are linearly independent.

For $n = 0$, there are no vectors, so the claim is trivially true.
Assume the claim is true for $n = k - 1$. We prove it for $n = k$.
The vector $\mathbf{v}_k$ added to S in the k^{th} iteration is not in the span of $\mathbf{v}_1, \ldots, \mathbf{v}_{k-1}$.
Therefore, by the Linear-Dependence Lemma, for any coefficients $\alpha_1, \ldots, \alpha_k$ such that

$$\mathbf{0} = \alpha_1 \mathbf{v}_1 + \cdots + \alpha_{k-1} \mathbf{v}_{k-1} + \alpha_k \mathbf{v}_k$$

it must be that α_k equals zero. We may therefore write

$$\mathbf{0} = \alpha_1 \mathbf{v}_1 + \cdots + \alpha_{k-1} \mathbf{v}_{k-1}$$

By claim for $n = k - 1$, $\mathbf{v}_1, \ldots, \mathbf{v}_{k-1}$ are linearly independent, so $\alpha_1 = \cdots = \alpha_{k-1} = 0$

The linear combination of $\mathbf{v}_1, \ldots, \mathbf{v}_k$ is *trivial*. We have proved that $\mathbf{v}_1, \ldots, \mathbf{v}_k$ are linearly independent. This proves the claim for $n = k$. QED

Analyzing the Shrink algorithm

def $\textsc{Shrink}(\mathcal{V})$
  $S =$ 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.

Shrink-Algorithm Corollary: The vectors obtained by the Shrink algorithm are linearly independent.

In graphs, this means that the Shrink algorithm outputs a solution that is a forest.

Recall:

Superfluous-Vector Lemma For any set S and any vector $\mathbf{v} \in S$, if **v** can be written as a linear combination of the other vectors in S then
Span $(S - \{\mathbf{v}\}) =$ Span S

Analyzing the Shrink algorithm

Shrink-Algorithm Corollary: The vectors obtained by the Shrink algorithm are linearly independent.

Proof: Let $S = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ be the set of vectors obtained by the Shrink algorithm. Assume for a contradiction that the vectors are linearly dependent.

Then $\mathbf{0}$ can be written as a nontrivial linear combination

$$\mathbf{0} = \alpha_1 \mathbf{v}_1 + \cdots + \alpha_n \mathbf{v}_n$$

where at least one of the coefficients is nonzero.

Let α_i be one of the nonzero coefficients.

By the Linear-Dependence Lemma, $\mathbf{v}_i$ can be written as a linear combination of the other vectors.

Hence by the Superfluous-Vector Lemma, Span $(S - \{\mathbf{v}_i\}) =$ Span S,

so the Shrink algorithm should have removed $\mathbf{v}_i$. QED

Basis

If they successfully finish, the Grow algorithm and the Shrink algorithm each find a set of vectors spanning the vector space $\mathcal{V}$. In each case, the set of vectors found is linearly independent.

Definition: Let $\mathcal{V}$ be a vector space. A *basis* for $\mathcal{V}$ is a linearly independent set of generators for $\mathcal{V}$.

Thus a set S of vectors of $\mathcal{V}$ is a *basis* for $\mathcal{V}$ if S satisfies two properties:

Property B1 (*Spanning*) Span $S = \mathcal{V}$, and

Property B2 (*Independent*) S is linearly independent.

Most important definition in linear algebra.

Basis: Examples

A set S of vectors of $\mathcal{V}$ is a *basis* for $\mathcal{V}$ if S satisfies two properties:

Property B1 (*Spanning*) Span $S = \mathcal{V}$, and

Property B2 (*Independent*) S is linearly independent.

Example: Let $\mathcal{V} = \text{Span } \{[1, 0, 2, 0], [0, -1, 0, -2], [2, 2, 4, 4]\}$.
Is $\{[1, 0, 2, 0], [0, -1, 0, -2], [2, 2, 4, 4]\}$ a basis for $\mathcal{V}$?

The set *is* spanning but is *not* independent

$$1\,[1, 0, 2, 0] - 1\,[0, -1, 0, -2] - \frac{1}{2}\,[2, 2, 4, 4] = \mathbf{0}$$

so not a basis

However, $\{[1, 0, 2, 0], [0, -1, 0, -2]\}$ *is* a basis:

- Obvious that these vectors are independent because each has a nonzero entry where the other has a zero.
- To show
  $$\text{Span } \{[1, 0, 2, 0], [0, -1, 0, -2]\} = \text{Span } \{[1, 0, 2, 0], [0, -1, 0, -2], [2, 2, 4, 4]\},$$
  can use Superfluous-Vector Lemma:
  $$[2, 2, 4, 4] = 2\,[1, 0, 2, 0] - 2\,[0, -1, 0, -2]$$

Basis: Examples

Example: A simple basis for $\mathbb{R}^3$: the standard generators
$\mathbf{e}_1 = [1, 0, 0], \mathbf{e}_2 = [0, 1, 0], \mathbf{e}_3 = [0, 0, 1]$.

- *Spanning:* For any vector $[x, y, z] \in \mathbb{R}^3$,

$$[x, y, z] = x\,[1, 0, 0] + y\,[0, 1, 0] + z\,[0, 0, 1]$$

- *Independent:* Suppose

$$\mathbf{0} = \alpha_1\,[1, 0, 0] + \alpha_2\,[0, 1, 0] + \alpha_3\,[0, 0, 1] = [\alpha_1, \alpha_2, \alpha_3]$$

  Then $\alpha_1 = \alpha_2 = \alpha_3 = 0$.

Instead of "standard generators", we call them *standard basis vectors*.
We refer to $\{[1, 0, 0], [0, 1, 0], [0, 0, 1]\}$ as *standard basis* for $\mathbb{R}^3$.

In general the standard generators are usually called *standard basis vectors*.

Basis: Examples

Example: Another basis for $\mathbb{R}^3$: $[1,1,1],[1,1,0],[0,1,1]$

- *Spanning:* Can write standard generators in terms of these vectors:

$$\begin{aligned}[1,0,0] &= [1,1,1]-[0,1,1]\\ [0,1,0] &= [1,1,0]+[0,1,1]-[1,1,1]\\ [0,0,1] &= [1,1,1]-[1,1,0]\end{aligned}$$

  Since $\mathbf{e}_1,\mathbf{e}_2,\mathbf{e}_3$ can be written in terms of these new vectors, every vector in Span $\{\mathbf{e}_1,\mathbf{e}_2,\mathbf{e}_3\}$ is in span of new vectors. Thus $\mathbb{R}^3$ equals span of new vectors.

- *Linearly independent:* Write zero vector as linear combination:

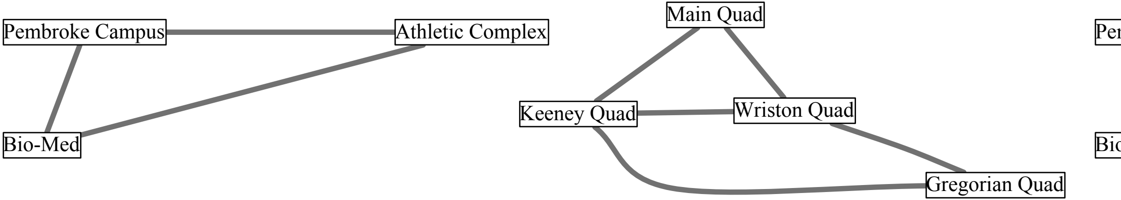
$$\mathbf{0} = x\,[1,1,1]+y\,[1,1,0]+z\,[0,1,1] = [x+y,x+y+z,x+z]$$

Looking at each entry, we get

$$\begin{aligned}0 &= x+y\\ 0 &= x+y+z\\ 0 &= x+z\end{aligned}$$

Plug $x+y=0$ into second equation to get $0=z$.
Plug $z=0$ into third equation to get $x=0$.
Plug $x=0$ into first equation to get $y=0$.
Thus the linear combination is trivial.

Basis: Examples in graphs



One kind of basis in a graph G: a set S of edges forming a spanning forest.

- *Spanning:* for each edge xy in G, there is an x-to-y path consisting of edges of S.
- *Independent:* no cycle consisting of edges of S

Towards showing that every vector space has a basis

We would like to prove that every vector space $\mathcal{V}$ has a basis.

The Grow algorithm and the Shrink algorithm each provides a way to prove this, but we are not there yet:

- The Grow-Algorithm Corollary implies that, if the Grow algorithm terminates, the set of vectors it has selected is a basis for the vector space $\mathcal{V}$.
  However, *we have not yet shown that it always terminates!*

- The Shrink-Algorithm Corollary implies that, if we can run the Shrink algorithm starting with a finite set of vectors that spans $\mathcal{V}$, upon termination it will have selected a basis for $\mathcal{V}$.
  However, *we have not yet shown that every vector space $\mathcal{V}$ is spanned by some finite set of vectors!*.

Computational problems involving finding a basis

Two natural ways to specify a vector space $\mathcal{V}$:

1. Specifying generators for $\mathcal{V}$.
2. Specifying a homogeneous linear system whose solution set is $\mathcal{V}$.

Two Fundamental Computational Problems:

Computational Problem: *Finding a basis of the vector space spanned by given vectors*

- *input:* a list $[\mathbf{v}_1, \ldots, \mathbf{v}_n]$ of vectors
- *output:* a list of vectors that form a basis for Span $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$.

Computational Problem: *Finding a basis of the solution set of a homogeneous linear system*

- *input:* a list $[\mathbf{a}_1, \ldots, \mathbf{a}_n]$ of vectors
- *output:* a list of vectors that form a basis for the set of solutions to the system $\mathbf{a}_1 \cdot \mathbf{x} = 0, \ldots, \mathbf{a}_n \cdot \mathbf{x} = 0$

Unique representation

Recall idea of *coordinate system* for a vector space $\mathcal{V}$:

- 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 its *coordinate representation* $[\alpha_1, \ldots, \alpha_n]$

Question: How can we ensure that each point has only one coordinate representation?

Answer: The generators $\mathbf{a}_1, \ldots, \mathbf{a}_n$ should form a basis.

Unique-Representation Lemma Let $\mathbf{a}_1, \ldots, \mathbf{a}_n$ be a basis for $\mathcal{V}$. For any vector $\mathbf{v} \in \mathcal{V}$, there is exactly one representation of $\mathbf{v}$ in terms of the basis vectors.

Uniqueness of representation in terms of a basis

Unique-Representation Lemma: Let $\mathbf{a}_1, \ldots, \mathbf{a}_n$ be a basis for $\mathcal{V}$. For any vector $\mathbf{v} \in \mathcal{V}$, there is exactly one representation of $\mathbf{v}$ in terms of the basis vectors.
Proof: Let $\mathbf{v}$ be any vector in $\mathcal{V}$.
The vectors $\mathbf{a}_1, \ldots, \mathbf{a}_n$ span $\mathcal{V}$, so there is at least one representation of $\mathbf{v}$ in terms of the basis vectors.

Suppose there are two such representations:

$$\mathbf{v} = \alpha_1\, \mathbf{a}_1 + \cdots + \alpha_n\, \mathbf{a}_n = \beta_1\, \mathbf{a}_1 + \cdots + \beta_n\, \mathbf{a}_n$$

We get the zero vector by subtracting one from the other:

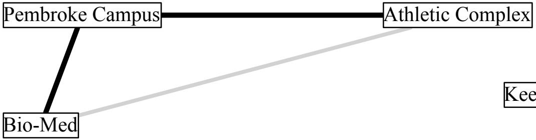
$$\begin{aligned} \mathbf{0} &= \alpha_1\, \mathbf{a}_1 + \cdots + \alpha_n\, \mathbf{a}_n - (\beta_1\, \mathbf{a}_1 + \cdots + \beta_n\, \mathbf{a}_n) \\ &= (\alpha_1 - \beta_1)\, \mathbf{a}_1 + \cdots + (\alpha_n - \beta_n)\, \mathbf{a}_n \end{aligned}$$

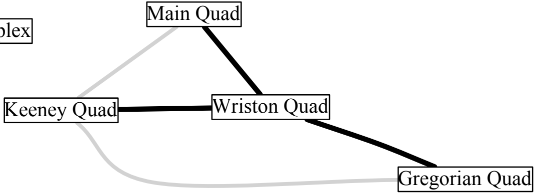
Since the vectors $\mathbf{a}_1, \ldots, \mathbf{a}_n$ are linearly independent, the coefficients $\alpha_1 - \beta_1, \ldots, \alpha_n - \beta_n$ must all be zero, so the two representations are really the same.

QED

Uniqueness of representation in terms of a basis: The case of graphs

Unique-Representation Lemma Let $\mathbf{a}_1, \ldots, \mathbf{a}_n$ be a basis for $\mathcal{V}$. For any vector $\mathbf{v} \in \mathcal{V}$, there is exactly one representation of $\mathbf{v}$ in terms of the basis vectors.





A basis for a graph is a spanning forest.
Unique Representation shows that, for each edge xy in the graph,

- there is an x-to-y path in the spanning forest, and
- there is only one such path.

Change of basis

Suppose we have a basis $\mathbf{a}_1, \ldots, \mathbf{a}_n$ for some vector space $\mathcal{V}$.
How do we go

- from a vector $\mathbf{b}$ in $\mathcal{V}$
- to the coordinate representation $\mathbf{u}$ of $\mathbf{b}$ in terms of $\mathbf{a}_1, \ldots, \mathbf{a}_n$?

By linear-combinations definition of matrix-vector multiplication,

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

By Unique-Representation Lemma,
$\mathbf{u}$ is the *only* solution to the equation

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

so we can obtain $\mathbf{u}$ by using a matrix-vector equation solver.

Function $f : \mathbb{F}^n \longrightarrow \mathcal{V}$ defined by $f(\mathbf{x}) = \left[\begin{array}{c|c|c} \mathbf{a}_1 & \cdots & \mathbf{a}_n \end{array}\right] \left[\begin{array}{c} \mathbf{x} \end{array}\right]$ is

- *onto* (because $\mathbf{a}_1, \ldots, \mathbf{a}_n$ are generators for $\mathcal{V}$)
- *one-to-one* (by Unique-Representation Lemma)

so f is an invertible function.

Change of basis

Now suppose $\mathbf{a}_1, \ldots, \mathbf{a}_n$ is one basis for $\mathcal{V}$ and $\mathbf{c}_1, \cdots, \mathbf{c}_k$ is another.

Define $f(\mathbf{x}) = \left[\begin{array}{c|c|c} \mathbf{a}_1 & \cdots & \mathbf{a}_n \end{array}\right] \left[\begin{array}{c} \mathbf{x} \end{array}\right]$ and define $g(\mathbf{y}) = \left[\begin{array}{c|c|c} \mathbf{c}_1 & \cdots & \mathbf{c}_k \end{array}\right] \left[\begin{array}{c} \mathbf{y} \end{array}\right]$.

Then both f and g are invertible functions.

The function $f^{-1} \circ g$ maps

- from coordinate representation of a vector in terms of $\mathbf{c}_1, \ldots, \mathbf{c}_k$
- to coordinate representation of a vector in terms of $\mathbf{a}_1, \ldots, \mathbf{a}_n$

In particular, if $\mathcal{V} = \mathbb{F}^m$ for some m then

f invertible implies that $\left[\begin{array}{c|c|c} \mathbf{a}_1 & \cdots & \mathbf{a}_n \end{array}\right]$ is an invertible matrix.

g invertible implies that $\left[\begin{array}{c|c|c} \mathbf{c}_1 & \cdots & \mathbf{c}_k \end{array}\right]$ is an invertible matrix.

Thus the function $f^{-1} \circ g$ has the property

$$(f^{-1} \circ g)(\mathbf{x}) = \left[\begin{array}{c|c|c} \mathbf{a}_1 & \cdots & \mathbf{a}_n \end{array}\right]^{-1} \left[\begin{array}{c|c|c} \mathbf{c}_1 & \cdots & \mathbf{c}_k \end{array}\right] \left[\begin{array}{c} \mathbf{x} \end{array}\right]$$

Change of basis

Proposition: If $\mathbf{a}_1, \ldots, \mathbf{a}_n$ and $\mathbf{c}_1, \ldots, \mathbf{c}_k$ are bases for $\mathbb{F}^m$ then multiplication by the matrix

$$B = \left[\begin{array}{c|c|c} \mathbf{a}_1 & \cdots & \mathbf{a}_n \end{array}\right]^{-1} \left[\begin{array}{c|c|c} \mathbf{c}_1 & \cdots & \mathbf{c}_k \end{array}\right]$$

maps

- from the representation of a vector with respect to $\mathbf{c}_1, \ldots, \mathbf{c}_k$
- to the representation of that vector with respect to $\mathbf{a}_1, \ldots, \mathbf{a}_n$.

Conclusion: Given two bases of $\mathbb{F}^m$, there is a matrix B such that multiplication by B converts from one coordinate representation to the other.

Remark: Converting between vector itself and its coordinate representation is a special case:

- Think of the vector itself as coordinate representation with respect to standard basis.

Change of basis: simple example

Example: To map

from coordinate representation with respect to $[1,2,3],[2,1,0],[0,1,4]$

to coordinate representation with respect to $[2,0,1],[0,1,-1],[1,2,0]$

multiply by the matrix

$$\left[\begin{array}{c|c|c} 2 & 0 & 1 \\ 0 & 1 & 2 \\ 1 & -1 & 0 \end{array}\right]^{-1} \left[\begin{array}{c|c|c} 1 & 2 & 0 \\ 2 & 1 & 1 \\ 3 & 0 & 4 \end{array}\right]$$

which is

$$\begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ \frac{2}{3} & -\frac{1}{3} & -\frac{4}{3} \\ -\frac{1}{3} & \frac{2}{3} & \frac{2}{3} \end{bmatrix} \left[\begin{array}{c|c|c} 1 & 2 & 0 \\ 2 & 1 & 1 \\ 3 & 0 & 4 \end{array}\right]$$

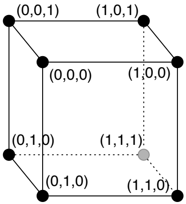
which is

$$\begin{bmatrix} -1 & 1 & -\frac{5}{3} \\ -4 & 1 & -\frac{17}{3} \\ 3 & 9 & \frac{10}{3} \end{bmatrix}$$

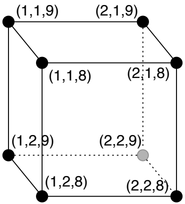
Perspective rendering

As application of change of basis, we show how to synthesize a camera view from a set of points in three dimensions, taking into account perspective.
The math will be useful in next lab, where we will go in the opposite direction, removing perspective from a real image.

We start with the points making up a wire cube:



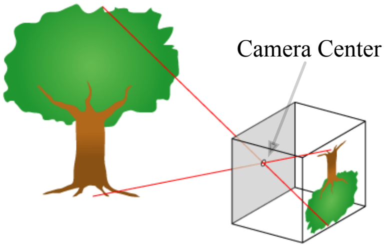
For reasons that will become apparent, we translate the cube, adding $(1, 1, 8)$ to each point.



How does a camera (or an eye) see these points?

Simplified camera model

Simplified model of a camera:

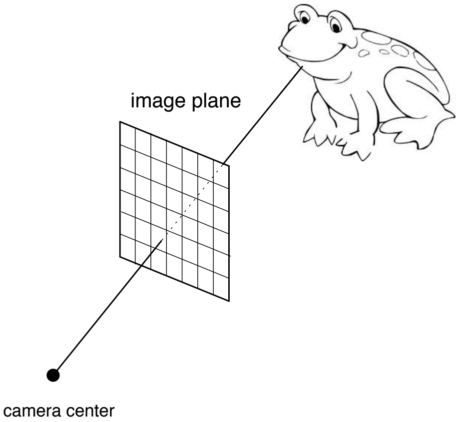


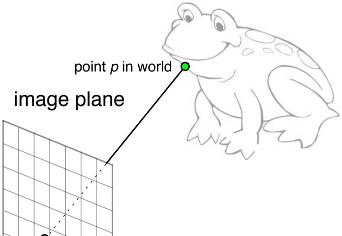
- There is a point called the *camera center*.
- There is an image sensor array in the back of the camera.
- Photons bounce off objects in the scene and travel through the camera center to the image sensor array.
- A photon from the scene only reaches the image sensor array if it travels in a straight line through the camera center.
- The image ends up being inverted.

Even more simplified camera model

Even simpler model to avoid the inversion:

- The image sensor array is between the camera center and the scene.
- The image sensor array is located in a plane, called the *image plane*.
- A photon from the scene is detected by the sensor array only if it is traveling in a straight line towards the camera center.
- The sensor element that detects the photon is the one intersected by this line.
- Need a function that maps from point **p** in world to corresponding point **q** in image plane

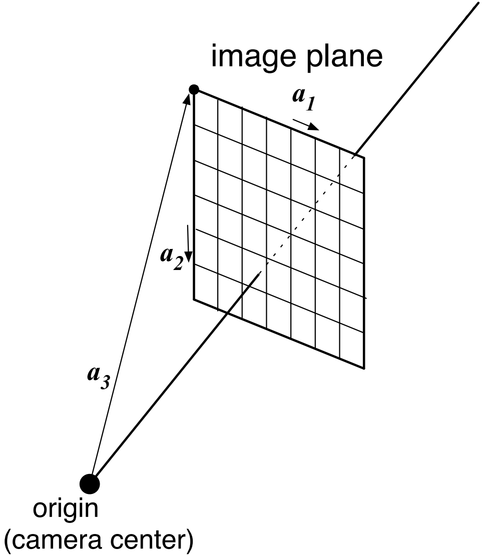




Camera coordinate system

Camera-oriented basis helps in mapping from world points to image-plane points:

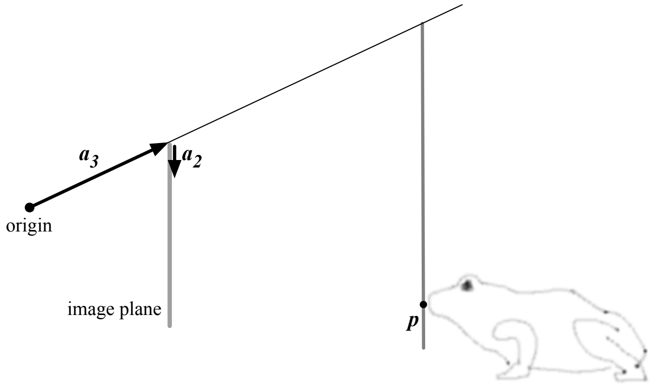
- The origin is defined to be the camera center. (That's why we translated the wire-frame cube.)
- The first vector $\mathbf{a}_1$ goes horizontally from the top-left corner of a sensor element to the top-right corner.
- The second vector $\mathbf{a}_2$ goes vertically from the top-left corner of a sensor element to the bottom-left corner.
- The third vector $\mathbf{a}_3$ goes from the origin (the camera center) to the bottom-left corner of sensor element (0,0).



From world point to camera-plane point

Side view (we see only the edge of the image plane)

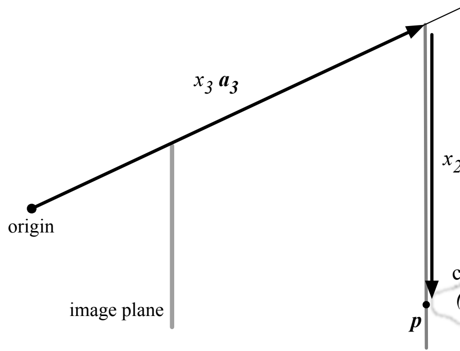
- Have a point $\mathbf{p}$ in the world
- Express it in terms of $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$
- Consider corresponding point $\mathbf{q}$ in image plane.
- Similar triangles $\Rightarrow$ coordinates of $\mathbf{q}$



Summary: Given coordinate representation (x_1, x_2, x_3) in terms of $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$,

coordinate representation of corresponding point in image plane is $(x_1/x_3, x_2/x_3, x_3/x_3)$.

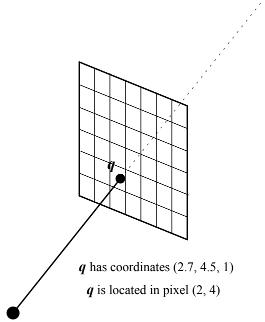
I call this *scaling down*.



Converting to pixel coordinates

Converting from a point (x_1, x_2, x_3) in the image plane to pixel coordinates

- Drop third entry x_3 (it is always equal to 1)



From world coordinates to camera coordinates to pixel coordinates

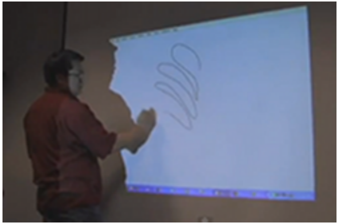
Write basis vectors of camera coordinate system using world coordinates

For each point $\mathbf{p}$ in the wire-frame cube,

- find representation in $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$
- scale down to get corresponding point in image plane
- convert to pixel coordinates by dropping third entry x_3

Wiimote whiteboard

For location of infrared point, wiimote provides coordinate representation in terms of its camera basis).



Johnny Chung Lee, wiimote whiteboard

To use as a mouse, need to find corresponding location on screen (coordinate representation in tems of screen basis)

How to transform from one coordinate representation to the other?

Can do this using a matrix H.

The challenge is to calculate the matrix H.

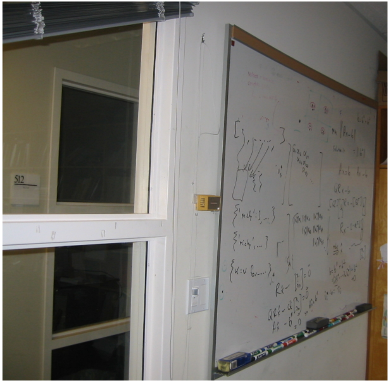
Can do this if you know the camera coordinate representation of four points whose screen coordinate representations are known.

You'll do exactly the same computation but for a slightly different problem....

Removing perspective

Given an image of a whiteboard, taken from an angle...

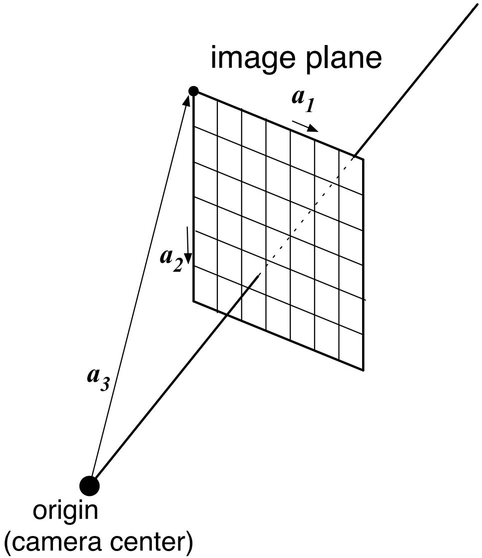
synthesize an image from straight ahead with no perspective



Camera coordinate system

We use same camera-oriented basis $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$:

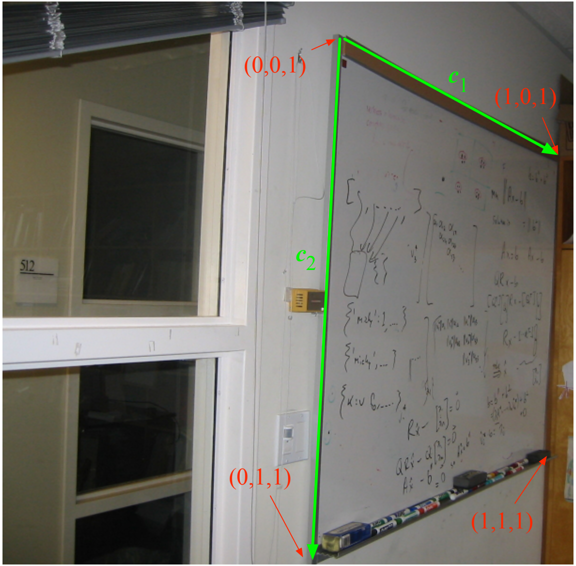
- The origin is the camera center.
- The first vector $\mathbf{a}_1$ goes horizontally from the top-left corner of the whiteboard element to the top-right corner.
- The second vector $\mathbf{a}_2$ goes vertically from the top-left corner of whiteboard to the bottom-left corner.
- The third vector $\mathbf{a}_3$ goes from the origin (the camera center) to the top-left corner of sensor element (0,0).



Converting from one basis to another

In addition, we define a *whiteboard basis* $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3$

- The origin is the camera center.
- The first vector $\mathbf{c}_1$ goes horizontally from the top-left corner of whiteboard to top-right corner.
- The second vector $\mathbf{c}_2$ goes vertically from the top-left corner of whiteboard to the bottom-left corner.
- The third vector $\mathbf{c}_3$ goes from the origin (the camera center) to the top-right corner of whiteboard.



Converting between different basis representations

Start with a point $\mathbf{p}$ written in terms of in camera coordinates

$$\mathbf{p} = \left[\begin{array}{c|c|c} & & \\ \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \\ & & \end{array}\right] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

We write the same point $\mathbf{p}$ in the whiteboard coordinate system as

$$\mathbf{p} = \left[\begin{array}{c|c|c} & & \\ \mathbf{c}_1 & \mathbf{c}_2 & \mathbf{c}_3 \\ & & \end{array}\right] \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

Combining the two equations, we obtain

$$\left[\begin{array}{c|c|c} & & \\ \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \\ & & \end{array}\right] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \left[\begin{array}{c|c|c} & & \\ \mathbf{c}_1 & \mathbf{c}_2 & \mathbf{c}_3 \\ & & \end{array}\right] \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

Converting...

$$\left[\begin{array}{c|c|c} \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \end{array}\right]\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \left[\begin{array}{c|c|c} \mathbf{c}_1 & \mathbf{c}_2 & \mathbf{c}_3 \end{array}\right]\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

Let A and C be the two matrices. As before, C has an inverse C^{-1}. Multiplying equation on the left by C^{-1}, we obtain

$$\begin{bmatrix} & & \\ & C^{-1} & \\ & & \end{bmatrix}\begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} & & \\ & C^{-1} & \\ & & \end{bmatrix}\begin{bmatrix} & & \\ & C & \\ & & \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

Since C^{-1} and C cancel out, we obtain

$$\begin{bmatrix} & & \\ & C^{-1} & \\ & & \end{bmatrix}\begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

We have shown that there is a matrix H (namely $H = C^{-1}A$) such that

$$\begin{bmatrix} & & \\ & H & \\ & & \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

From pixel coordinates to whiteboard coordinates

		(x_1, x_2)	pixel coordinates
		$\downarrow$	
		$(x_1, x_2, 1)$	represent point in image plane in terms of camera basis
		$\downarrow$	
(y_1, y_2, y_3)	$=$	$H * (x_1, x_2, 1)$	change to representation in terms of whiteboard basis
$\downarrow$			
$(y_1/y_3, y_2/y_3, y_3/y_3)$			move to corresponding point in whiteboard plane
$\downarrow$			
$(y_1/y_3, y_2/y_3)$			get coordinates within whiteboard

How to almost compute H

Write $H = \begin{bmatrix} h_{y_1,x_1} & h_{y_1,x_2} & h_{y_1,x_3} \\ h_{y_2,x_1} & h_{y_2,x_2} & h_{y_2,x_3} \\ h_{y_3,x_1} & h_{y_3,x_2} & h_{y_3,x_3} \end{bmatrix}$

The h_{ij}'s are the unknowns.

To derive equations, let **p** be some point on the whiteboard, and let **q** be the corresponding point on the image plane. Let $(x_1, x_2, 1)$ be the camera coordinates of **q**, and let (y_1, y_2, y_3) be the whiteboard coordinates of **q**. We have

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} h_{y_1,x_1} & h_{y_1,x_2} & h_{y_1,x_3} \\ h_{y_2,x_1} & h_{y_2,x_2} & h_{y_2,x_3} \\ h_{y_3,x_1} & h_{y_3,x_2} & h_{y_3,x_3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$$

Multiplying out, we obtain

$$\begin{aligned} y_1 &= h_{y_1,x_1} x_1 + h_{y_1,x_2} x_2 + h_{y_1,x_3} \\ y_2 &= h_{y_2,x_1} x_1 + h_{y_2,x_2} x_2 + h_{y_2,x_3} \\ y_3 &= h_{y_3,x_1} x_1 + h_{y_3,x_2} x_2 + h_{y_3,x_3} \end{aligned}$$

Almost computing H

$$\begin{aligned} y_1 &= h_{y_1,x_1}x_1 + h_{y_1,x_2}x_2 + h_{y_1,x_3} \\ y_2 &= h_{y_2,x_1}x_1 + h_{y_2,x_2}x_2 + h_{y_2,x_3} \\ y_3 &= h_{y_3,x_1}x_1 + h_{y_3,x_2}x_2 + h_{y_3,x_3} \end{aligned}$$

Whiteboard coordinates of the original point **p** are $(y_1/y_3, y_2/y_3, 1)$. Define

$$\begin{aligned} w_1 &= y_1/y_3 \\ w_2 &= y_2/y_3 \end{aligned}$$

so the whiteboard coordinates of **p** are $(w_1, w_2, 1)$.
Multiplying through by y_3, we obtain

$$\begin{aligned} w_1 y_3 &= y_1 \\ w_2 y_3 &= y_2 \end{aligned}$$

Substituting our expressions for y_1, y_2, y_3, we obtain

$$\begin{aligned} w_1(h_{y_3,x_1}x_1 + h_{y_3,x_2}x_2 + h_{y_3,x_3}) = h_{y_1,x_1}x_1 + h_{y_1,x_2}x_2 + h_{y_1,x_3} \\ w_2(h_{y_3,x_1}x_1 + h_{y_3,x_2}x_2 + h_{y_3,x_3}) = h_{y_2,x_1}x_1 + h_{y_2,x_2}x_2 + h_{y_2,x_3} \end{aligned}$$

$$w_1(h_{y_3,x_1}x_1 + h_{y_3,x_2}x_2 + h_{y_3,x_3}) = h_{y_1,x_1}x_1 + h_{y_1,x_2}x_2 + h_{y_1,x_3}$$
$$w_2(h_{y_3,x_1}x_1 + h_{y_3,x_2}x_2 + h_{y_3,x_3}) = h_{y_2,x_1}x_1 + h_{y_2,x_2}x_2 + h_{y_2,x_3}$$

Multiplying through and moving everything to the same side, we obtain

$$(w_1x_1)h_{y_3,x_1} + (w_1x_2)h_{y_3,x_2} + w_1h_{y_3,x_3} - x_1h_{y_1,x_1} - x_2h_{y_1,x_2} - 1h_{y_1,x_3} = 0$$
$$(w_2x_1)h_{y_3,x_1} + (w_2x_2)h_{y_3,x_2} + w_2h_{y_3,x_3} - x_1h_{y_2,x_1} - x_2h_{y_2,x_2} - 1h_{y_2,x_3} = 0$$

Thus we get two linear equations in the unknowns. The coeffients are expressed in terms of x_1, x_2, w_1, w_2.

For four points, get eight equations. Need one more...

One more equation

We can't pin down H precisely.

This corresponds to the fact that we cannot recover the scale of the picture (a tiny building that is nearby looks just like a huge building that is far away).

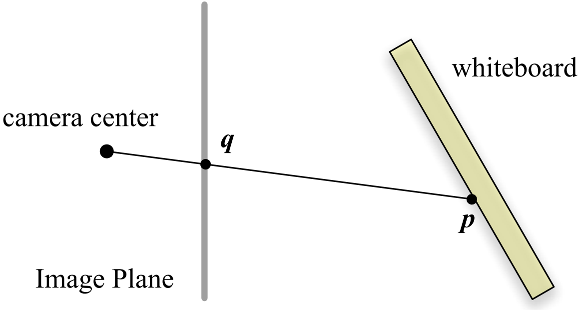
Fortunately, we don't need the true H.

As long as the H we compute is a scalar multiple of the true H, things will work out.

To arbitrarily select a scale, we add the equation $h_{y_1,x_1} = 1$.

Once you know H

1. For each point $\mathbf{q}$ in the representation of the image, we have the camera coordinates $(x_1, x_2, 1)$ of $\mathbf{q}$. We multiply by H to obtain the whiteboard coordinates (y_1, y_2, y_3) of the same point $\mathbf{q}$.
2. Recall the situation as viewed from above:



   The whiteboard coordinates of the corresponding point $\mathbf{p}$ on the whiteboard are $(y_1/y_3, y_2/y_3, 1)$. Use this formula to compute these coordinates.
3. Display the updated points with the same color matrix

Simplified Exchange Lemma

We need a tool to iteratively transform one set of generators into another.

- You have a set S of vectors.
- You have a vector z you want to inject into S.
- You want to maintain same size so must eject a vector from S.
- You want the span to not change.

Exchange Lemma tells you how to choose vector to eject.

Simplified Exchange Lemma:

- Suppose S is a set of vectors.
- Suppose $\mathbf{z}$ is a nonzero vector in Span S.
- Then there is a vector $\mathbf{w}$ in S such that

$$\text{Span } (S \cup \{\mathbf{z}\} - \{\mathbf{w}\}) = \text{Span } S$$

Simplified Exchange Lemma proof

Simplified Exchange Lemma: Suppose S is a set of vectors, and $\mathbf{z}$ is a nonzero vector in Span S. Then there is a vector $\mathbf{w}$ in S such that Span $(S \cup \{\mathbf{z}\} - \{\mathbf{w}\}) = $ Span S.

Proof: Let $S = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$. Since $\mathbf{z}$ is in Span S, can write

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

By Superfluous-Vector Lemma, Span $(S \cup \{z\}) = $ Span S.
Since $\mathbf{z}$ is nonzero, at least one of the coefficients is nonzero, say α_i.
Rewrite as

$$\mathbf{z} - \alpha_1\,\mathbf{v}_1 - \cdots - \alpha_{i-1}\,\mathbf{v}_{i-1} - \alpha_{i+1}\,\mathbf{v}_{i+1} - \cdots - \alpha_n\,\mathbf{v}_n = \alpha_i\,\mathbf{v}_i$$

Divide through by α_i:

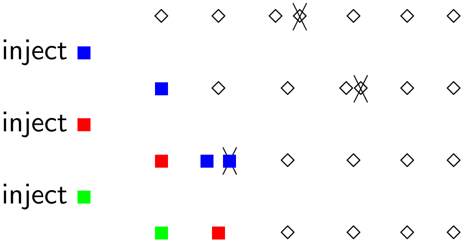
$$(1/\alpha_i)\mathbf{z} - (\alpha_1/\alpha_i)\,\mathbf{v}_1 - \cdots - (\alpha_{i-1}/\alpha_i)\,\mathbf{v}_{i-1} - (\alpha_{i+1}/\alpha_i)\,\mathbf{v}_{i+1} - \cdots - (\alpha_n/\alpha_i)\,\mathbf{v}_n = \mathbf{v}_i$$

By Superfluous-Vector Lemma, Span $(S \cup \{\mathbf{z}\}) = $ Span $(S \cup \{\mathbf{z}\} - \{\mathbf{w}\})$. QED

Exchange Lemma

Simplified Exchange Lemma: Suppose S is a set of vectors, and $\mathbf{z}$ is a nonzero vector in Span S. Then there is a vector $\mathbf{w}$ in S such that Span $(S \cup \{\mathbf{z}\} - \{\mathbf{w}\}) = \text{Span } S$.

Simplified Exchange Lemma helps in transforming one generating set into another...



Trying to put squares in—when you put in one square, you might end up taking out a previously inserted square

Need a way to protect some elements from being taken out.

Exchange Lemma

Simplified Exchange Lemma: Suppose S is a set of vectors, and $\mathbf{z}$ is a nonzero vector in Span S. Then there is a vector $\mathbf{w}$ in S such that Span $(S \cup \{\mathbf{z}\} - \{\mathbf{w}\}) = \text{Span } S$.

Need to enhance this lemma. Set of *protected* elements is A:

Exchange Lemma:

- Suppose S is a set of vectors and A is a subset of S.
- Suppose $\mathbf{z}$ is a vector in Span S such that $A \cup \{\mathbf{z}\}$ is linearly independent.
- Then there is a vector $\mathbf{w} \in S - A$ such that Span $S = \text{Span } (S \cup \{\mathbf{z}\} - \{\mathbf{w}\})$

Now, not enough that $\mathbf{z}$ be nonzero—need A to be linearly independent.

Exchange Lemma proof

Exchange Lemma: Suppose S is a set of vectors and A is a subset of S. Suppose $\mathbf{z}$ is a vector in Span S such that $A \cup \{\mathbf{z}\}$ is linearly independent.
Then there is a vector $\mathbf{w} \in S - A$ such that Span $S =$ Span $(S \cup \{\mathbf{z}\} - \{\mathbf{w}\})$

Proof: Let $S = \{\mathbf{v}_1, \ldots, \mathbf{v}_k, \mathbf{w}_1, \ldots, \mathbf{w}_\ell\}$ and $A = \{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$.
Since $\mathbf{z}$ is in Span S, can write

$$\mathbf{z} = \alpha_1\,\mathbf{v}_1 + \cdots + \alpha_k\,\mathbf{v}_k + \beta_1\;\mathbf{w}_1 + \cdots + \beta_\ell\;\mathbf{w}_\ell$$

By Superfluous-Vector Lemma, Span $(S \cup \{z\}) =$ Span S.
If coefficients $\beta_1, \ldots, \beta_\ell$ were all zero then we would have $\mathbf{z} = \alpha_1\,\mathbf{v}_1 + \cdots + \alpha_k\,\mathbf{v}_k$, contradicting the linear independence of $A \cup \{\mathbf{z}\}$.
Thus one of the coefficients $\beta_1, \ldots, \beta_\ell$ must be nonzero... say β_1. Rewrite as

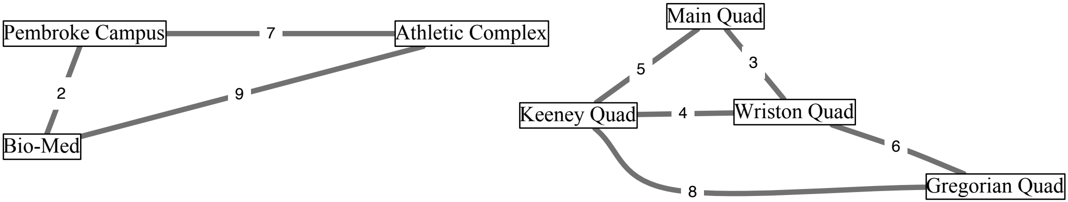
$$\mathbf{z} - \alpha_1\,\mathbf{v}_1 - \cdots - \alpha_k\,\mathbf{v}_k - \beta_2\,\mathbf{w}_2 - \cdots - \beta_\ell\,\mathbf{w}_\ell = \beta_1\,\mathbf{w}_1$$

Divide through by β_1:

$$(1/\beta_1)\mathbf{z} - (\alpha_1/\beta_1)\,\mathbf{v}_1 - \cdots - (\alpha_k/\beta_1)\,\mathbf{v}_k - (\beta_2/\beta_1)\,\mathbf{w}_2 - \cdots - (\beta_\ell/\beta_1)\,\mathbf{w}_\ell = \mathbf{w}_1$$

By Superfluous-Vector Lemma, Span $(S \cup \{\mathbf{z}\}) =$ Span $(S \cup \{\mathbf{z}\} - \{\mathbf{w}_1\})$. QED

Proof of correctness of the Grow algorithm for Minimum Spanning Forest



def $\text{Grow}(G)$
  $F := \emptyset$
  consider the edges in increasing order
  for each edge e:
    if e's endpoints are not yet connected
      add e to F.

We will show that this greedy algorithm chooses the minimum-weight spanning forest.

(Assume all weights are distinct.)

Let F = forest found by algorithm.

Let F^* = truly minimum-weight spanning forest.

Goal: show that $F = F^*$

Assume for a contradiction that they are different.

Proof of correctness of the Grow algorithm for Minimum Spanning Forest

Assume for a contradiction that F and F^* are different.
Let $e_1, e_2, \ldots, e_m$ be the edges of G in increasing order.
Let e_k be the minimum-weight edge on which F and F^* disagree.
Let A be the set of edges before e_k that are in both F and F^*.
Since at least one of the forests includes all of A and also e_k, we know $A \cup \{e_k\}$ has no cycles (is linearly independent).
Consider the moment when the Grow algorithm considers e_k. So far, the algorithm has chosen the edges in A, and e_k does not form a cycle with edges in A, so the algorithm must also choose e_k.
Since F and F^* differ on e_k, we infer that e_k is *not* in F^*.
Now we use the Exchange Lemma.

- A is a subset of F^*.
- $A \cup \{e_k\}$ is linearly independent.
- Therefore there is an edge e_n in $F^* - A$ such that
$$\text{Span } (F^* \cup \{e_k\} - \{e_n\}) = \text{Span } F^*$$

That is, $F^* \cup \{e_k\} - \{e_n\}$ is also spanning.
But e_k is cheaper than e_n so F^* is not min-weight solution. **Contradiction.** QED.