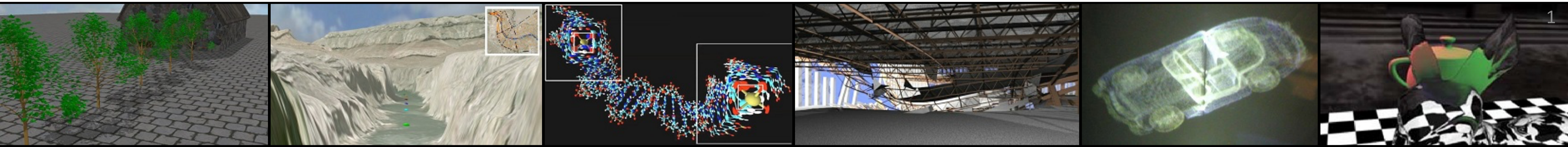
COT 4521: Introduction to Computational Geometry

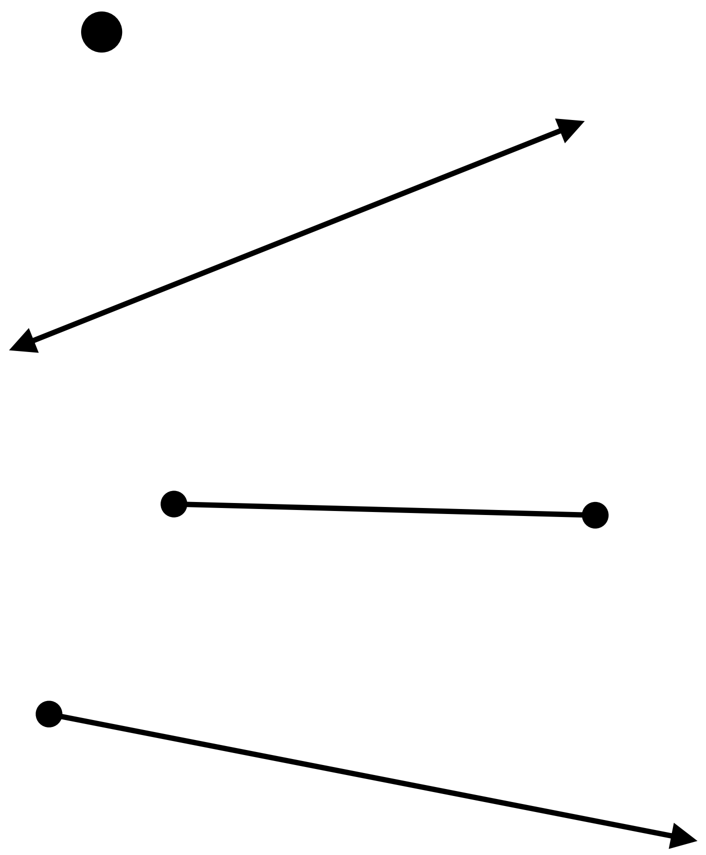
Preliminaries

Paul Rosen
Assistant Professor
University of South Florida



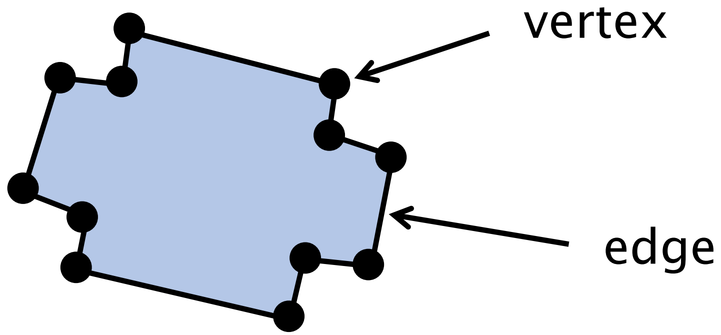
Basic Objects of CG

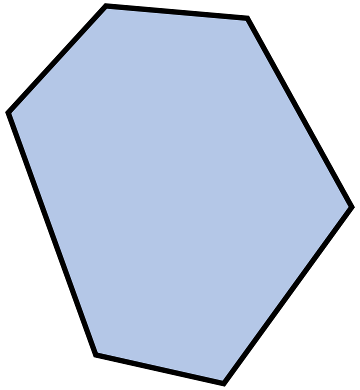
- Point—Specified by two coordinates (x, y)
- Line—Extends to infinity in both directions
- Line segment—Specified by 2 endpoints
- Ray—Extends to infinity in 1 direction



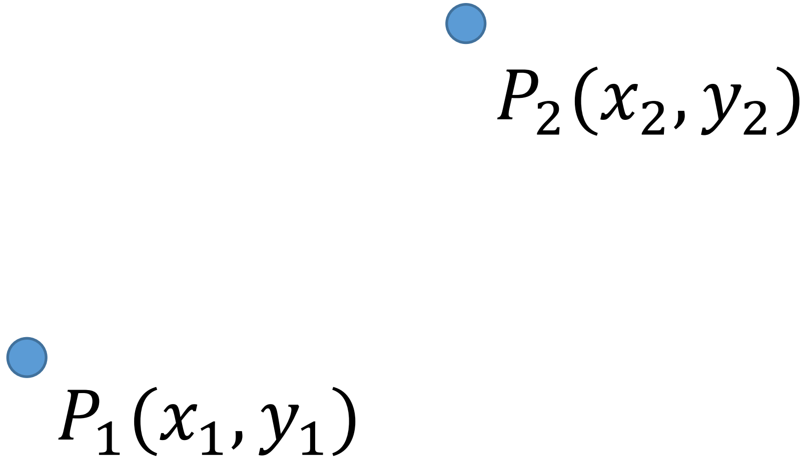
BASIC OBJECTS OF CG

- POLYGON
  - We assume edges do not cross
- CONVEX POLYGON
  - Every interior angle is at most 180 degrees
  - Precise definition of *convex*: For any 2 points inside the polygon, the line segment joining them lies entirely inside the polygon (we'll cover this later)





2D Points



What's the distance between these points?

Choice 1: difference between the point locations

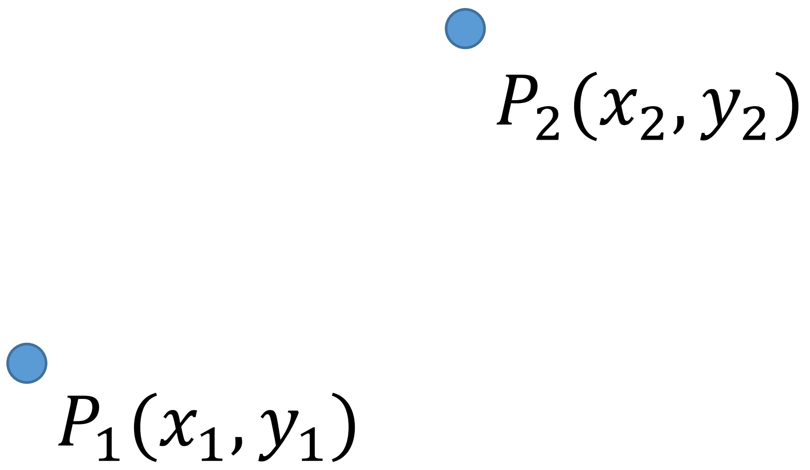
Choice 2: Euclidean distance

Choice 3: Commute time distance

Choice 4: Ill-defined question

EUCLIDEAN DISTANCE

$$d(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$



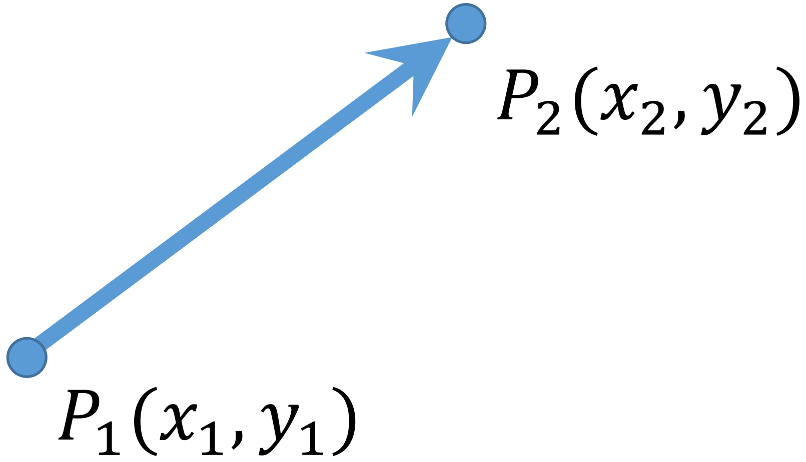
EUCLIDEAN DISTANCE

- PERFORMANCE TRICK
  - Square root is kind of slow and imprecise
  - If we only need to check whether the distance is less than some certain length, say R

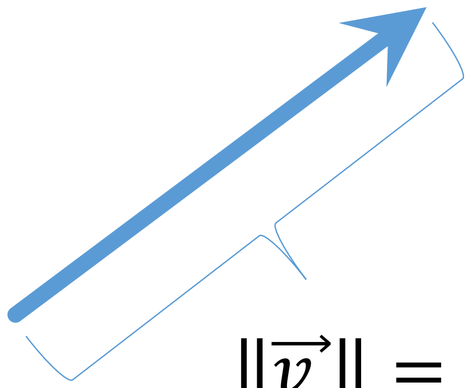
$$\text{if } \left((x_2 - x_1)^2 + (y_2 - y_1)^2 < R^2\right) \ldots$$

VECTORS

$$\overrightarrow{v_{12}} = P_2 - P_1$$
$$= < x_2 - x_1, y_2 - y_1 >$$

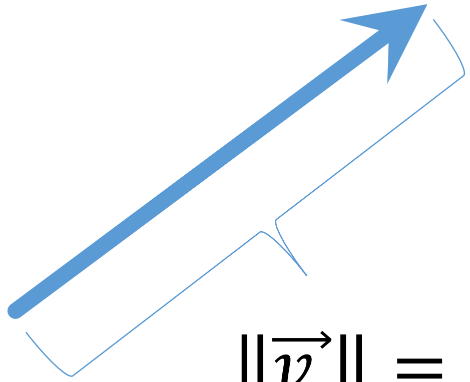


VECTOR LENGTH/MAGNITUDE



$$\|\overrightarrow{v}\| = \sqrt{v_x^2 + v_y^2}$$

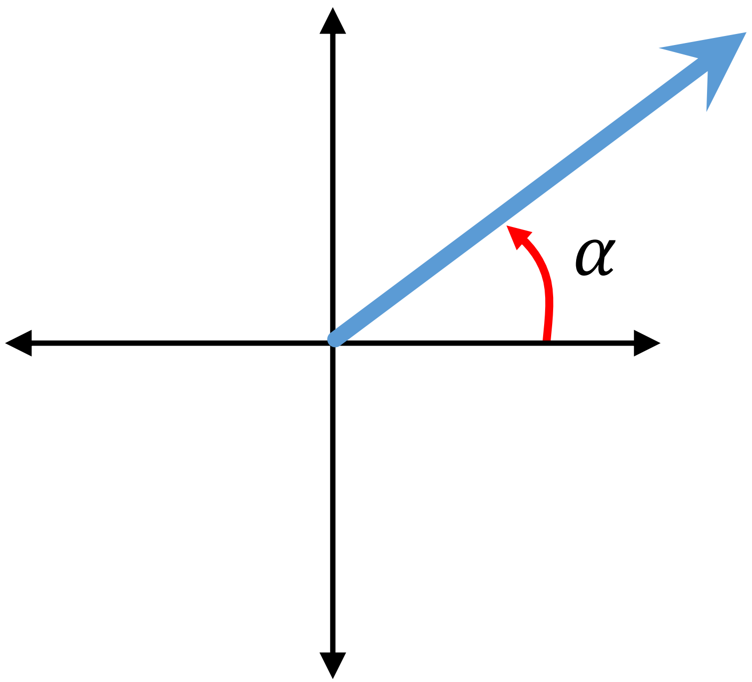
Vector Length/Magnitude



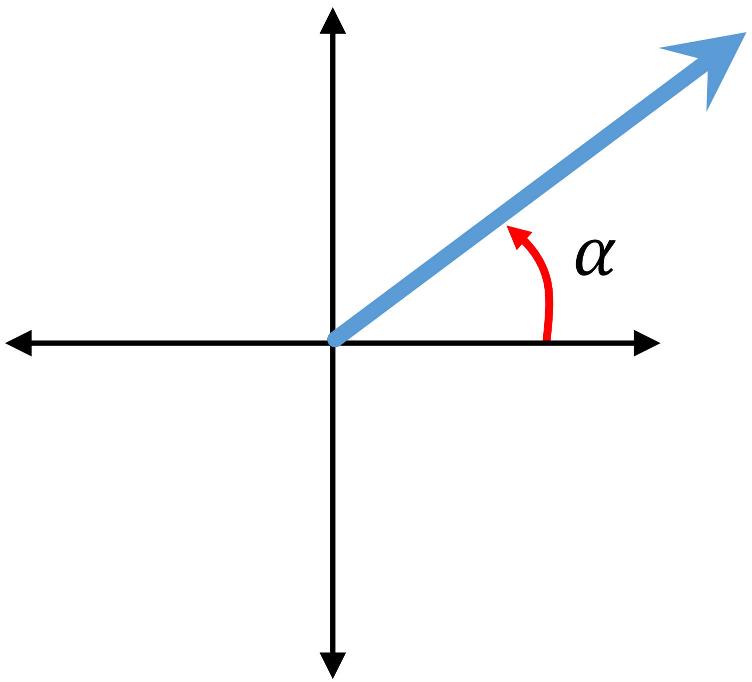
$$\|\vec{v}\| = \sqrt{v_x^2 + v_y^2}$$

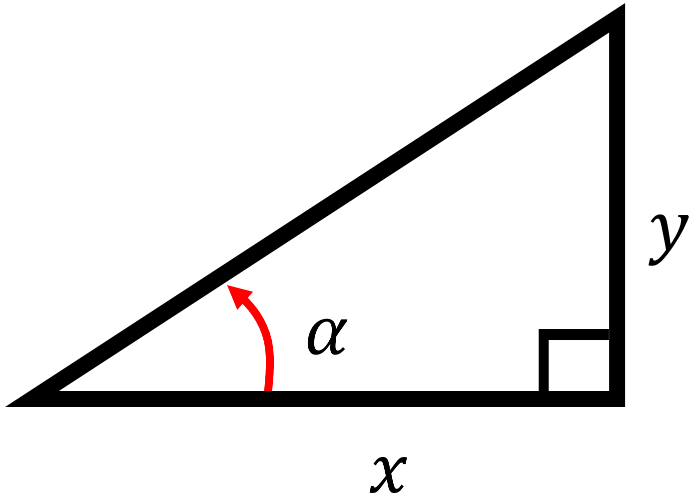
Equivalent to Euclidean distance…

Vector Angle



Vector Angle





$$\tan(\alpha) = \frac{y}{x}$$

Vector Angle

$$\alpha = \tan^{-1}\frac{y}{x}$$

or

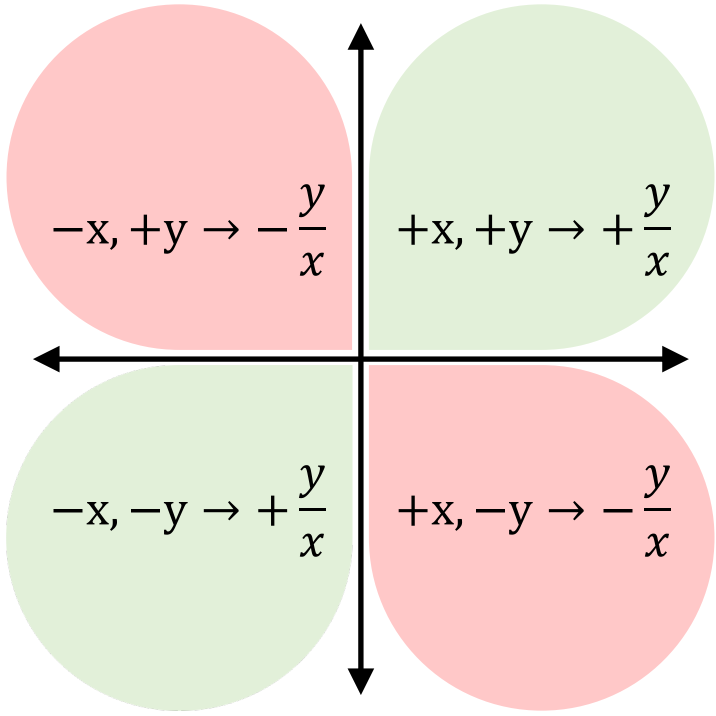
$$\alpha = \mathrm{atan}\left(\frac{y}{x}\right)$$

(results in radians, not degrees)

Problems?

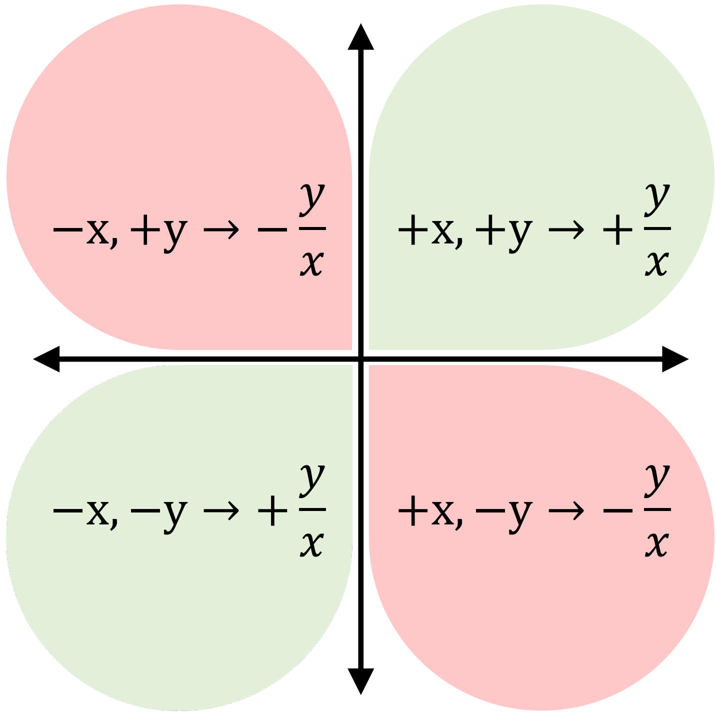
VECTOR ANGLE

- PROBLEM 1: DIVISION BY ZERO
  - When α is $\frac{\pi}{2}$ or $-\frac{\pi}{2}$
- PROBLEM 2: $\frac{y}{x}$ DOESN'T GIVE A 1-TO-1 MAPPING

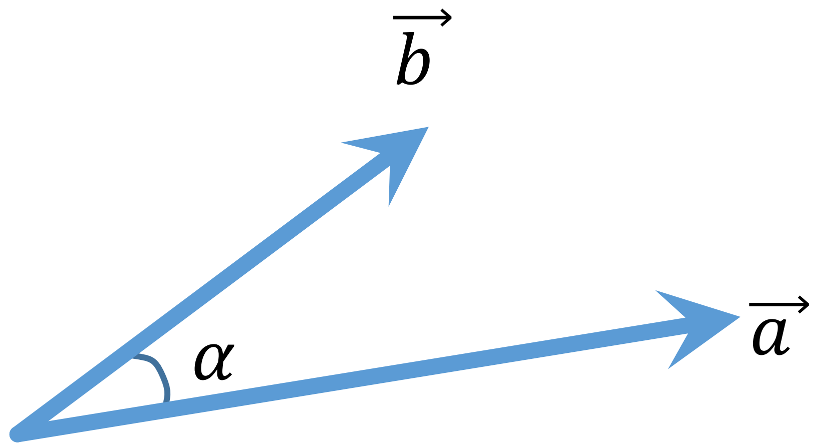


Vector Angle

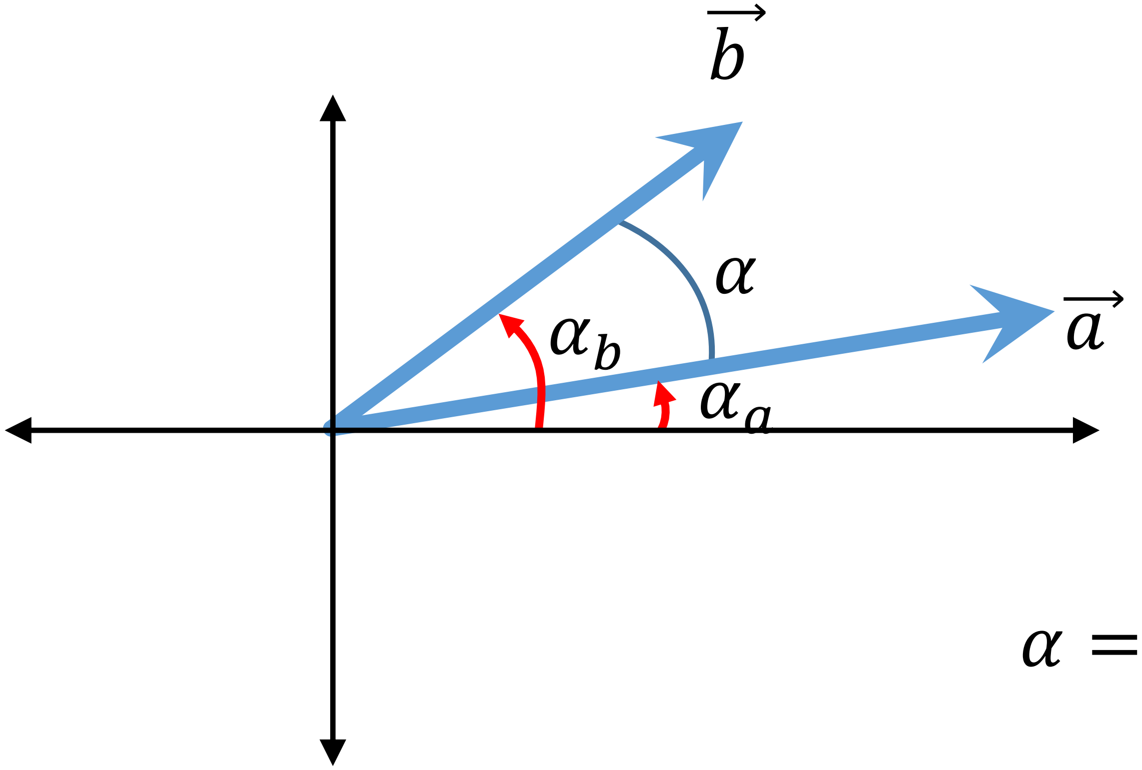
- Problem 1: Division by zero
  - When α is $\frac{\pi}{2}$ or $-\frac{\pi}{2}$
- Problem 2: $\frac{y}{x}$ doesn't give a 1-to-1 mapping
- Solution: $\boldsymbol{\alpha = \mathrm{atan}\,2(y, x)}$
  - Note: the arguments are $(\boldsymbol{y}, \boldsymbol{x})$, not (x, y)!!!



Angle Between 2 Vectors

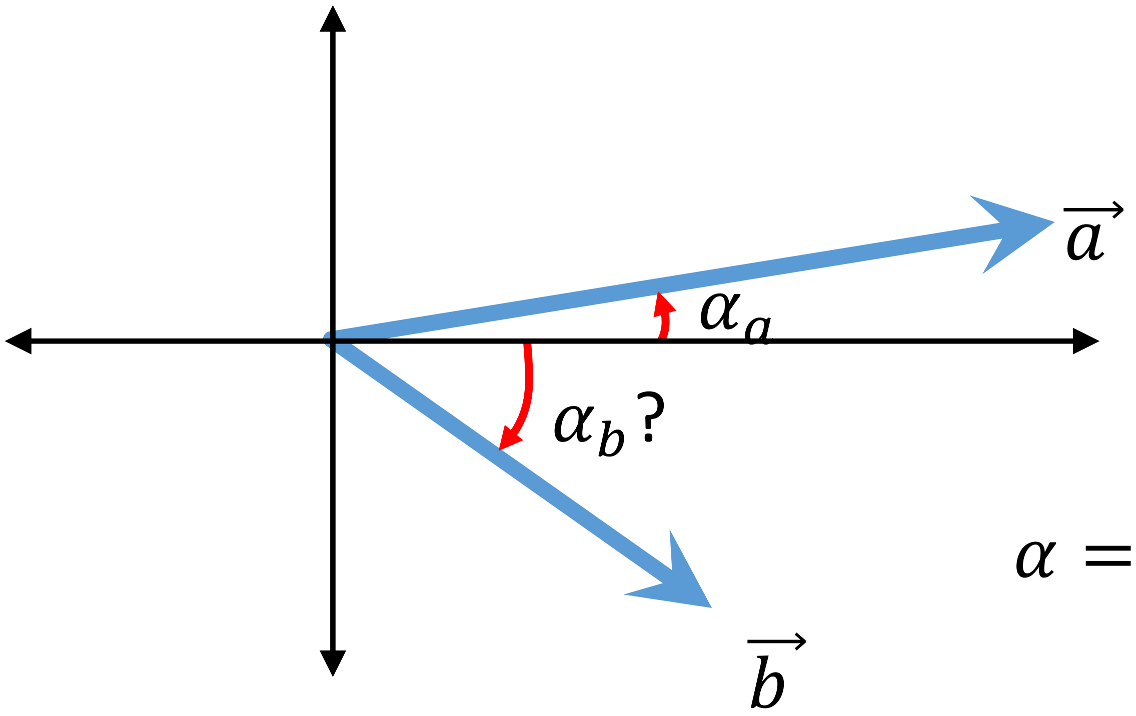


Angle Between 2 Vectors



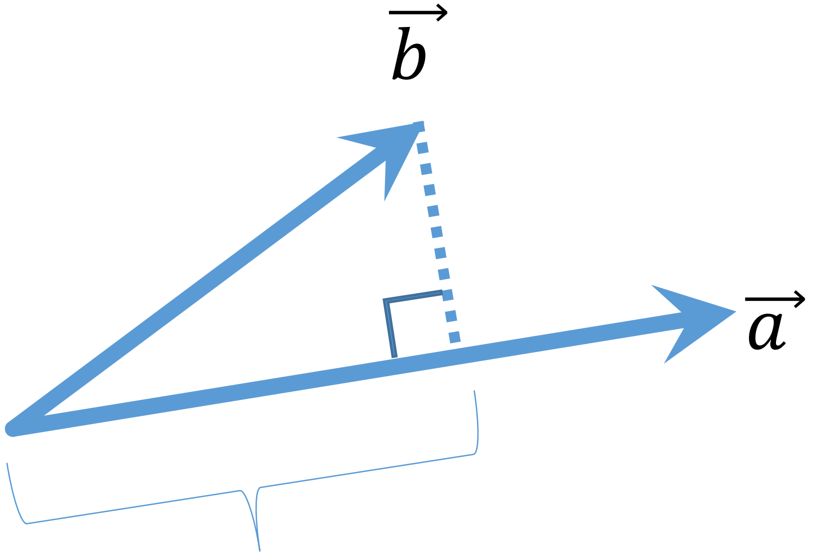
$$\alpha = \alpha_b - \alpha_a$$

Angle Between 2 Vectors



$$\alpha = \alpha_b - \alpha_a ?$$

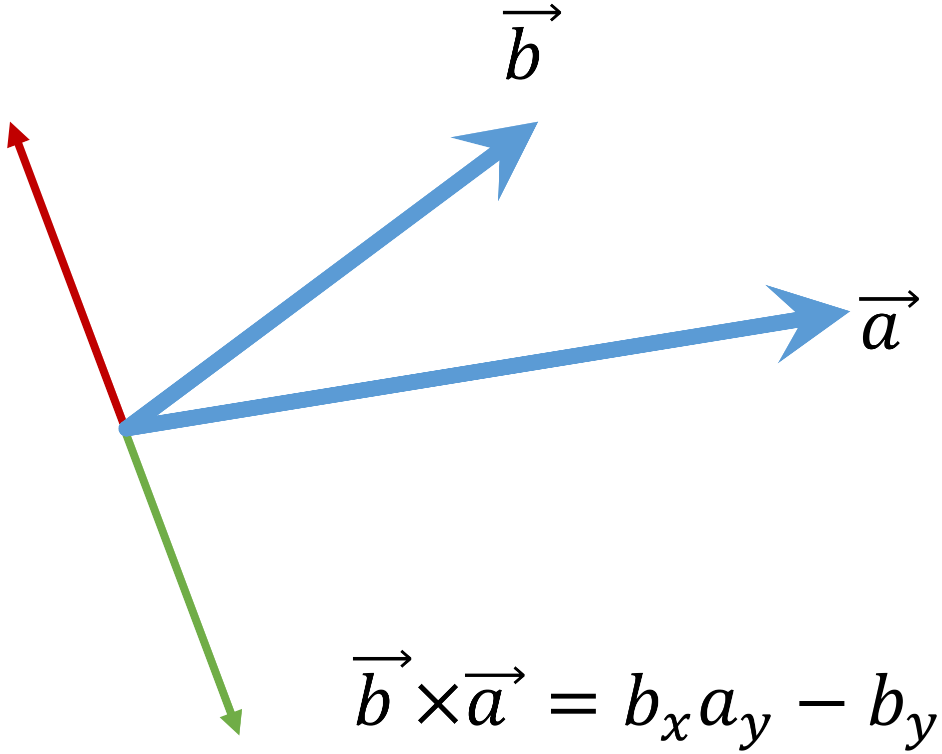
Dot Product



$$\vec{a} \cdot \vec{b} = a_x b_x + a_y b_y$$

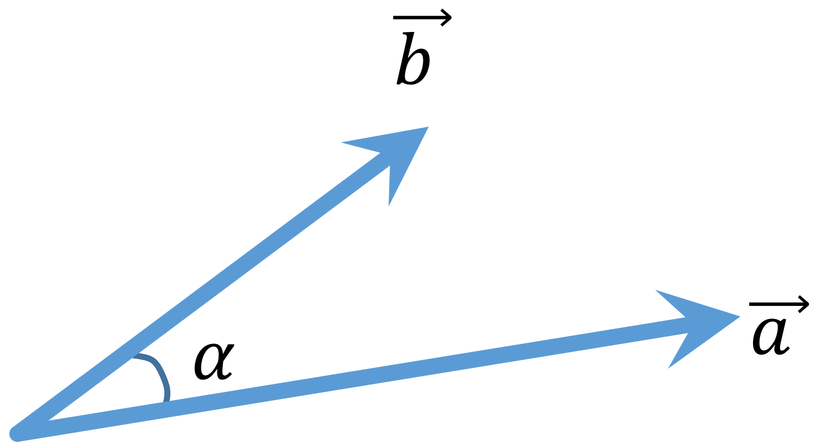
Cross Product

$$\vec{a} \times \vec{b} = a_x b_y - a_y b_x$$



$$\vec{b} \times \vec{a} = b_x a_y - b_y a_x$$

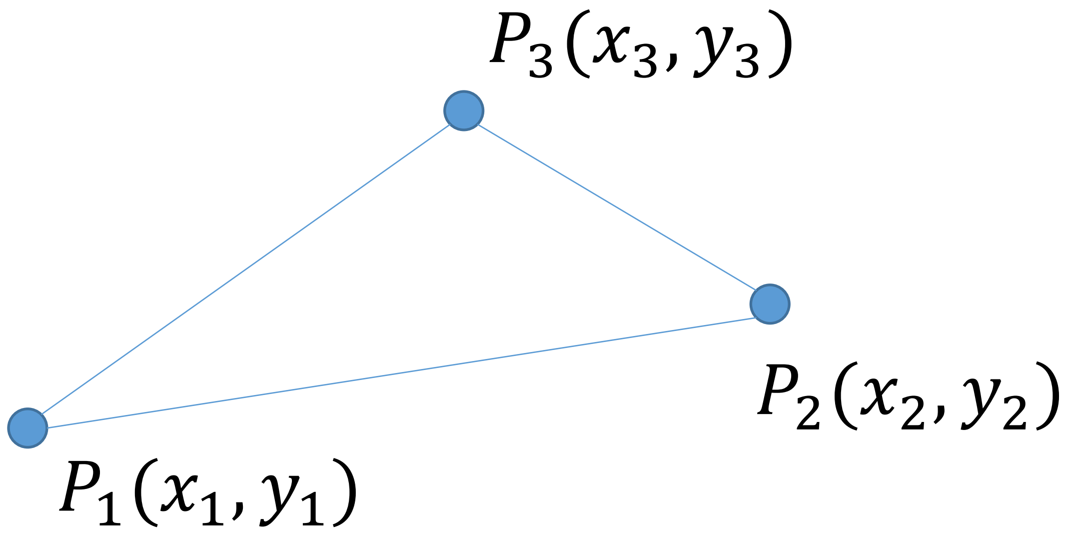
Angle Between 2 Vectors



$$\cos\alpha = \frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\|\vec{b}\|} \qquad \sin\alpha = \frac{\|\vec{a}\times\vec{b}\|}{\|\vec{a}\|\|\vec{b}\|}$$

Orientation of Triangles

How do you use the tools just discussed to determine orientation?



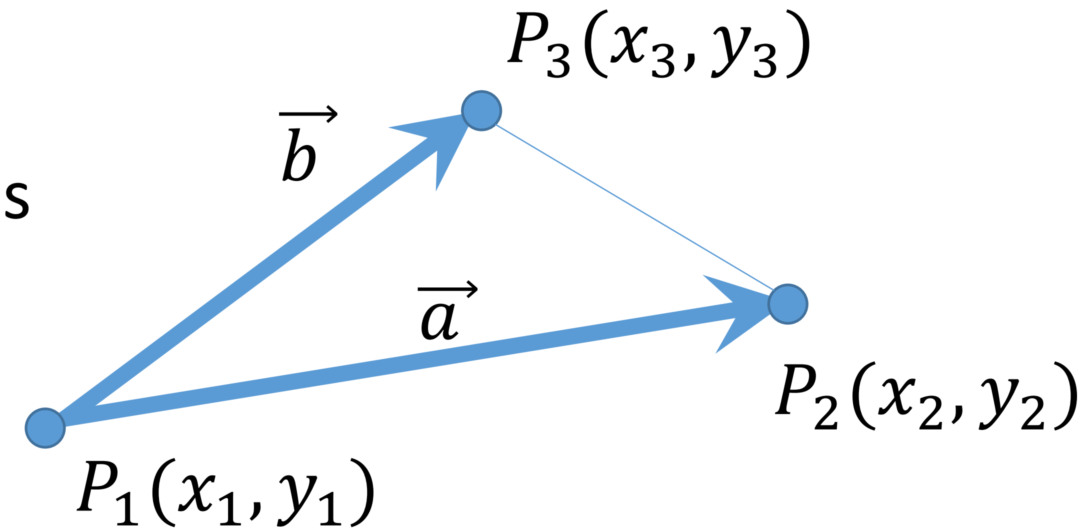
Orientation of Triangles

How do you use the tools just discussed to determine orientation?

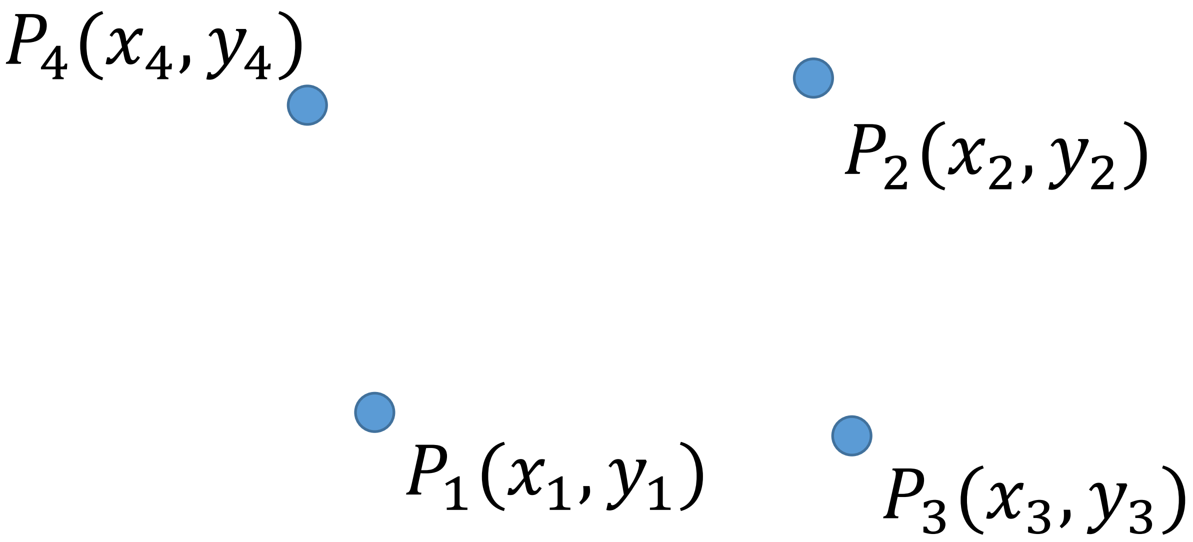
<u>Choice 1</u>: difference in angles

<u>Choice 2</u>: dot product

<u>Choice 3</u>: cross product



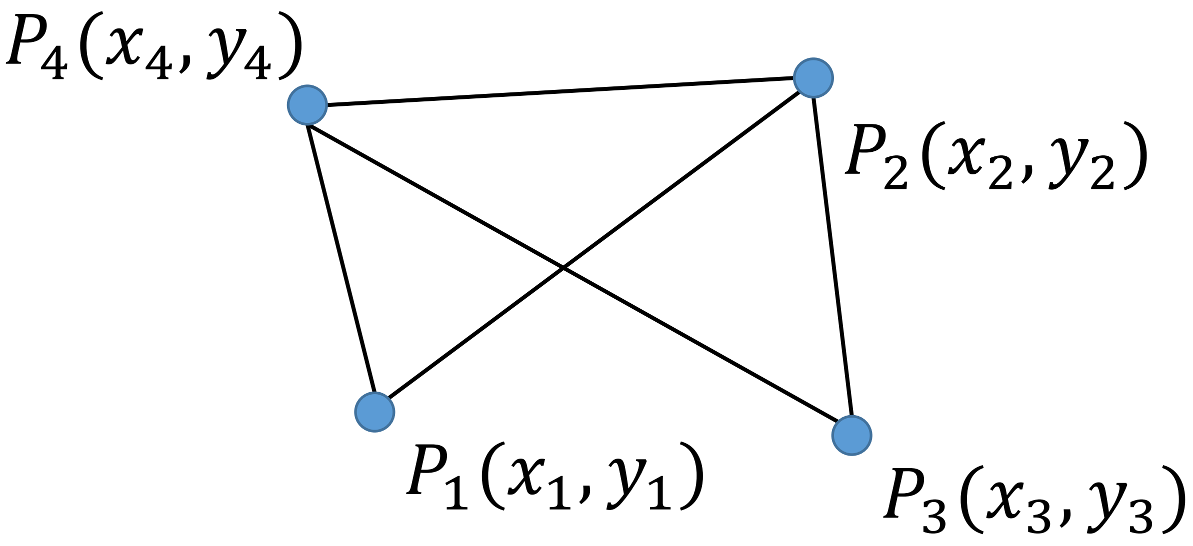
Storing Points



Point List
(x_1, y_1)
(x_2, y_2)
(x_3, y_3)
(x_4, y_4)
…
(x_n, y_n)

X List	Y List
x_1	y_1
x_2	y_2
x_3	y_3
x_4	y_4
…	…
x_n	y_n

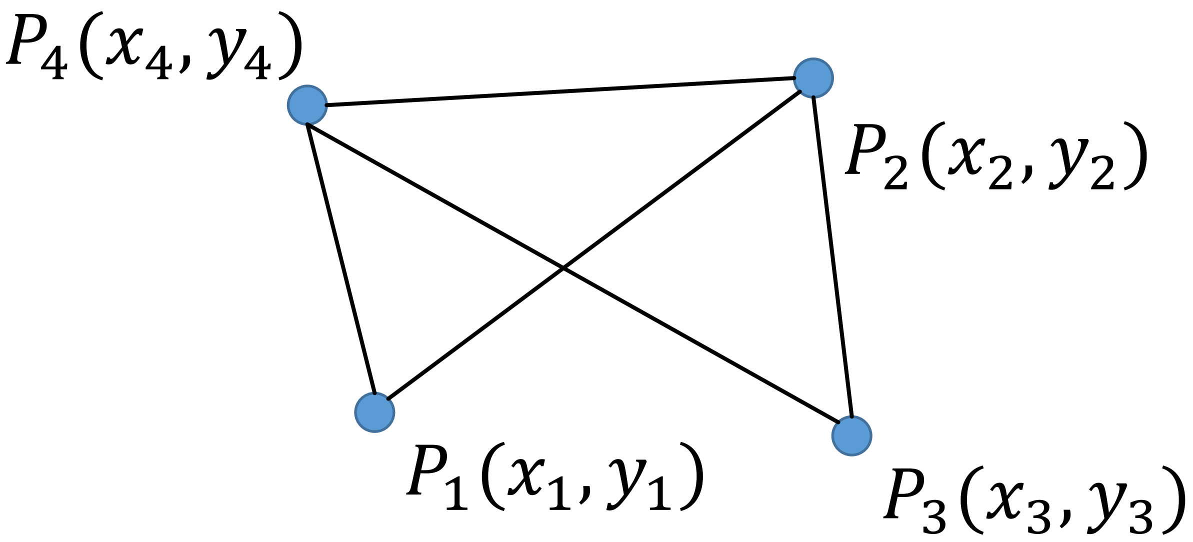
Storing Edges as Indices

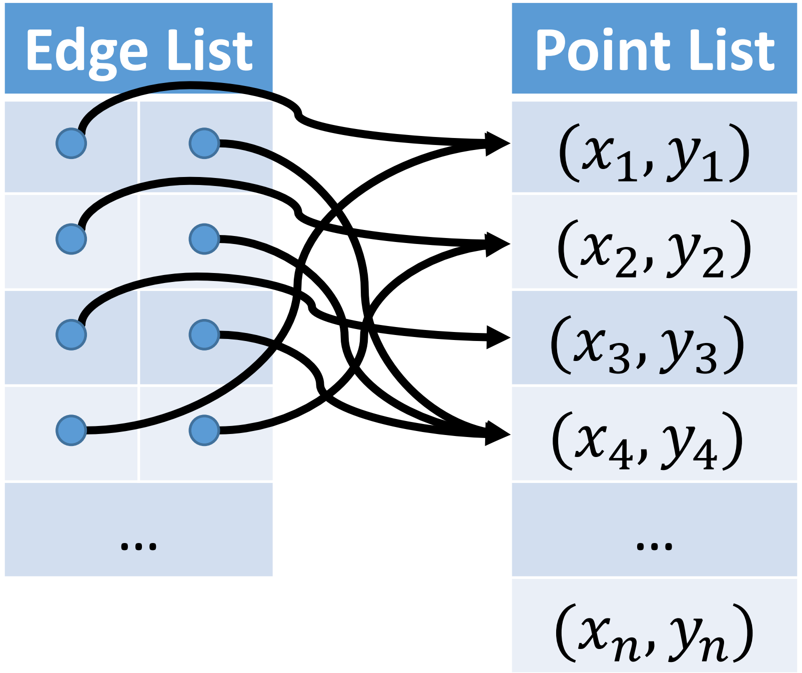


Edge List	
0	3
1	3
2	3
0	1
…	

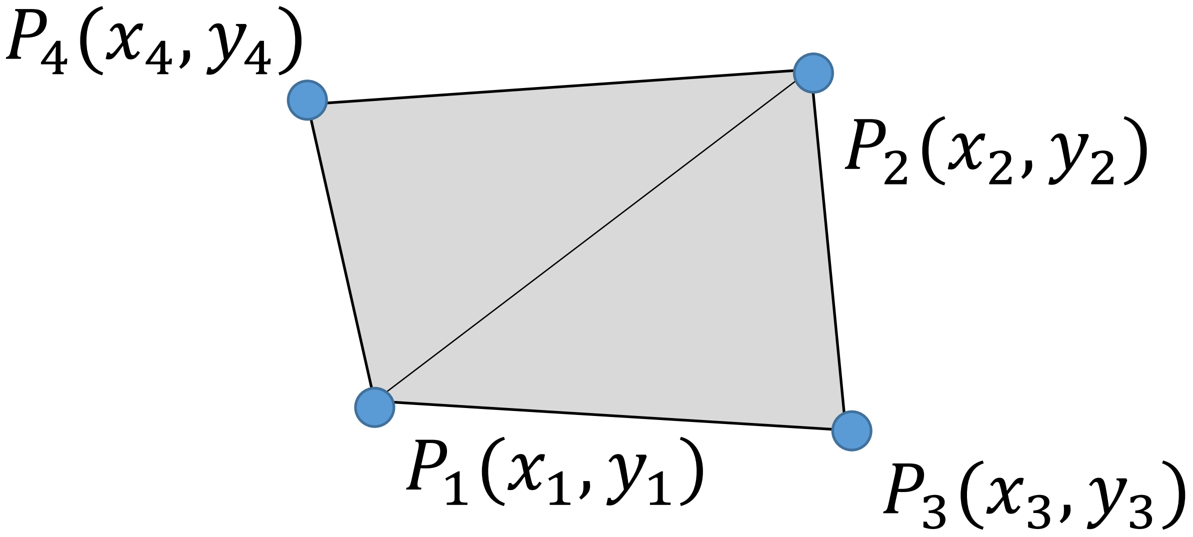
Point List	
0	(x_1, y_1)
1	(x_2, y_2)
2	(x_3, y_3)
3	(x_4, y_4)
…	…
n-1	(x_n, y_n)

Storing Points as Pointers





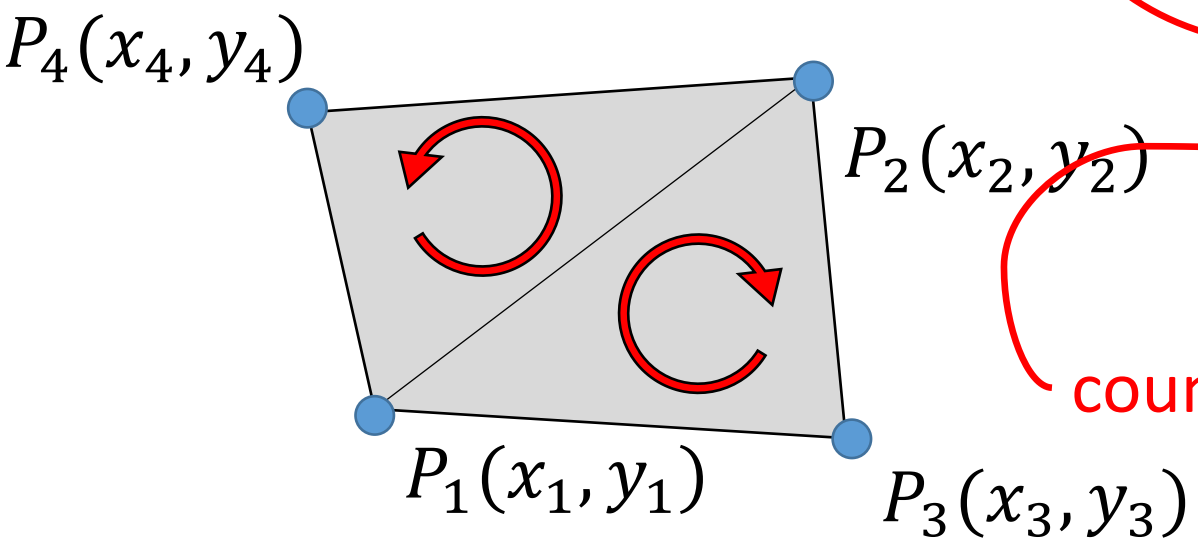
Storing Triangles with Indices



Triangle List		
0	1	2
0	1	3
…		

Point List	
0	(x_1, y_1)
1	(x_2, y_2)
2	(x_3, y_3)
3	(x_4, y_4)
…	…
n-1	(x_n, y_n)

Storing Triangles with Indices



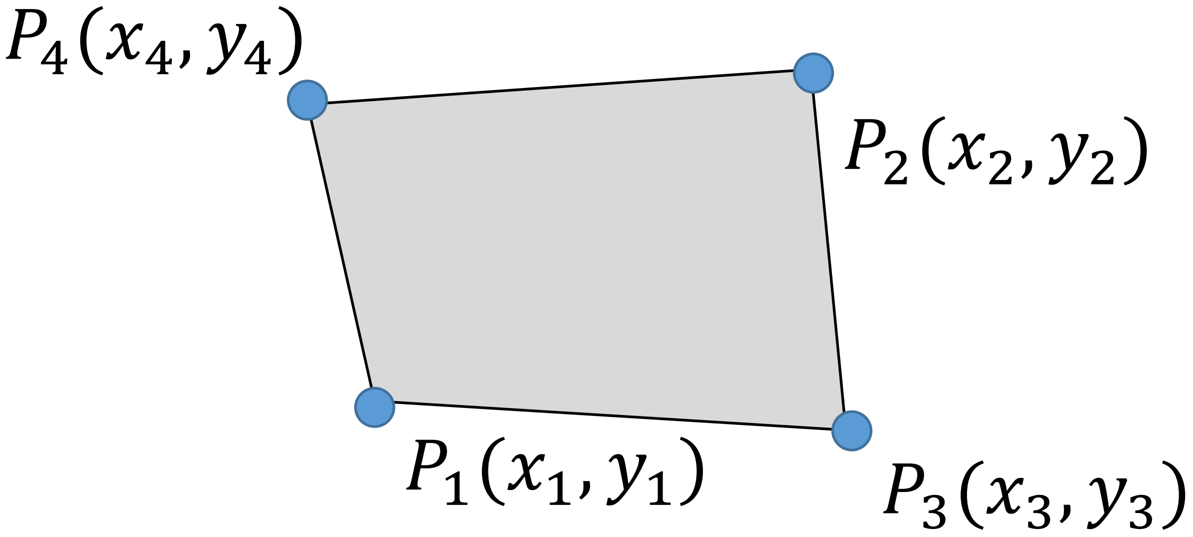
clockwise

counterclockwise

Triangle List		
0	1	2
0	1	3
…		

Point List	
0	(x_1, y_1)
1	(x_2, y_2)
2	(x_3, y_3)
3	(x_4, y_4)
…	…
n-1	(x_n, y_n)

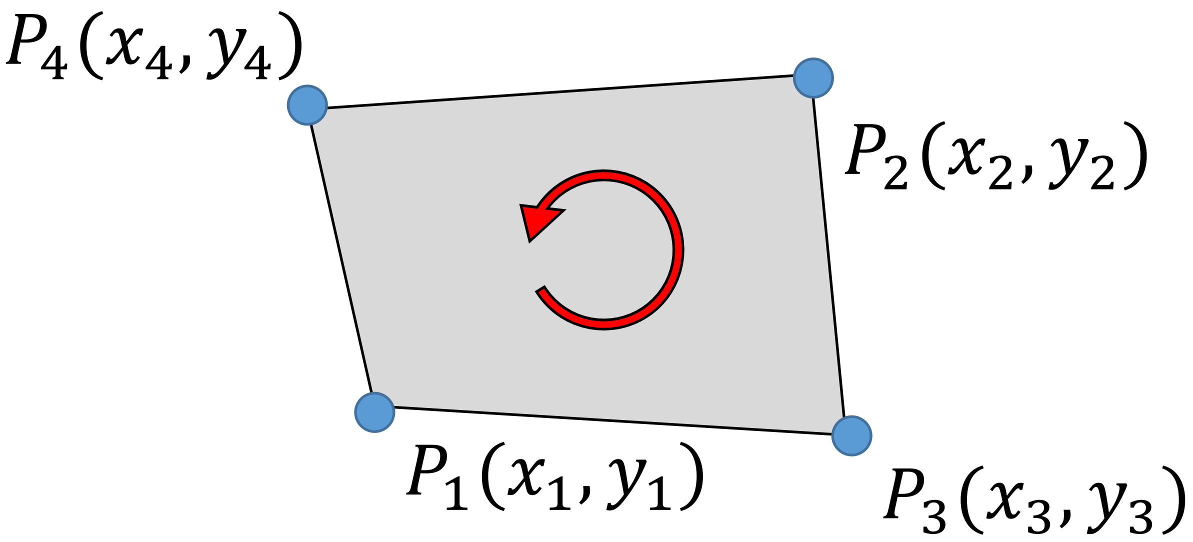
Storing Polygons with Indices



Polygon List			
0	2	1	3
…			

Point List	
0	(x_1, y_1)
1	(x_2, y_2)
2	(x_3, y_3)
3	(x_4, y_4)
…	…
n-1	(x_n, y_n)

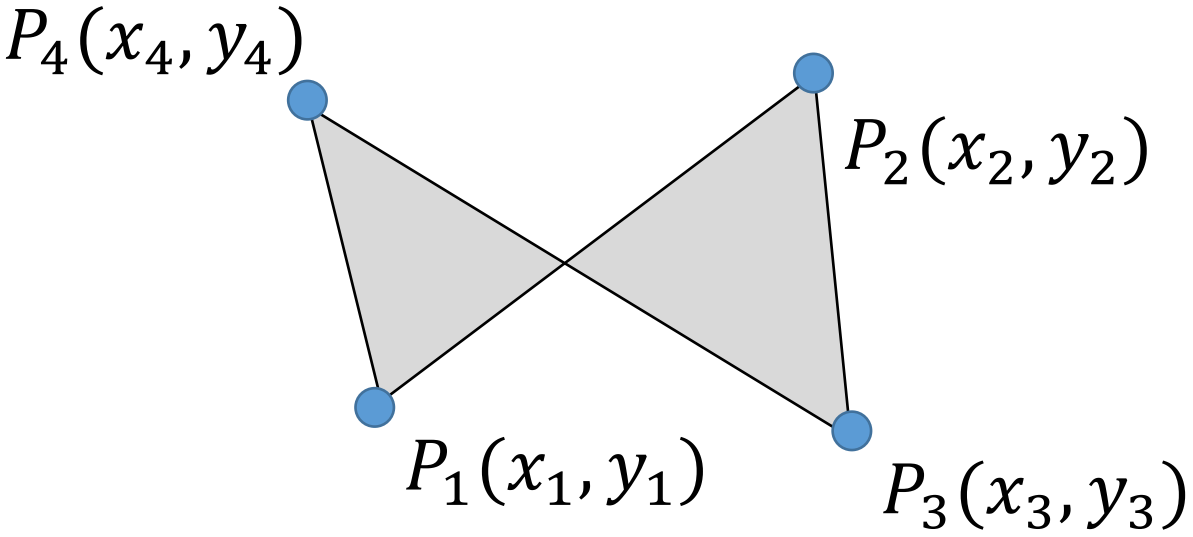
Storing Polygons with Indices



Polygon List			
0	2	1	3
…			

Point List	
0	(x_1, y_1)
1	(x_2, y_2)
2	(x_3, y_3)
3	(x_4, y_4)
…	…
n-1	(x_n, y_n)

Storing Polygons with Indices

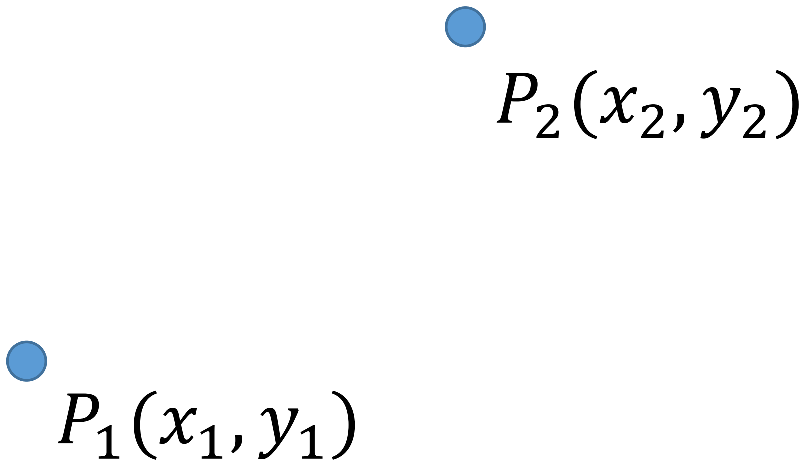


Polygon List			
0	1	2	3
…			

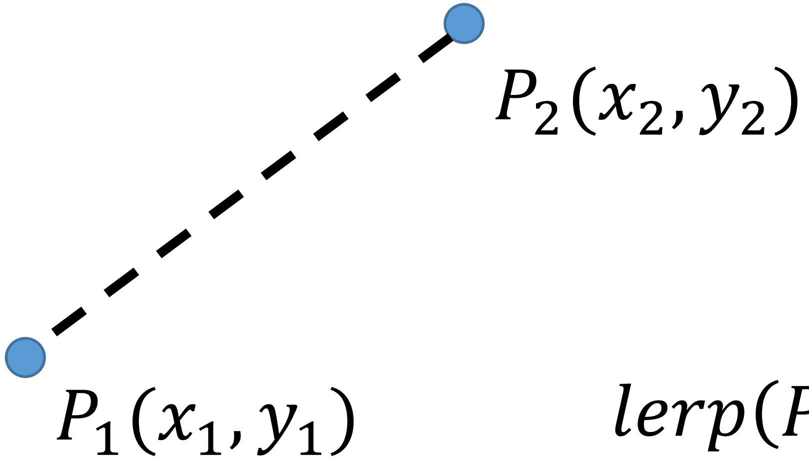
Take care with order

Point List	
0	(x_1, y_1)
1	(x_2, y_2)
2	(x_3, y_3)
3	(x_4, y_4)
…	…
n-1	(x_n, y_n)

Linear Interpolation

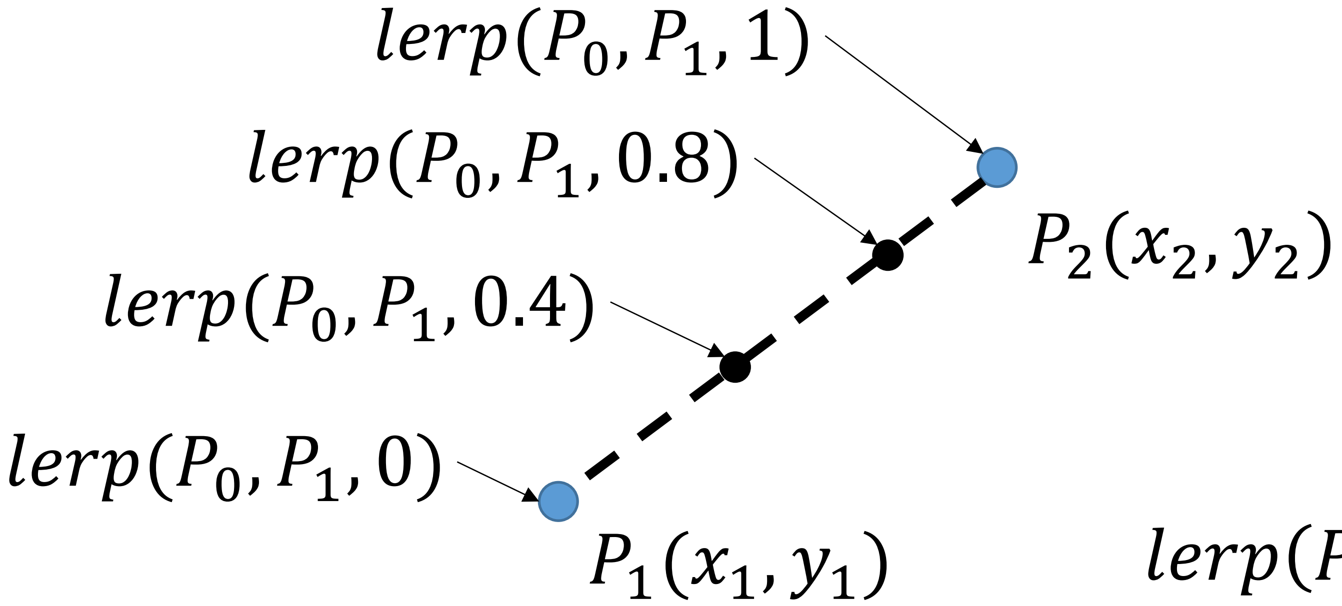


Linear Interpolation



$$lerp(P_1, P_2, a) = P_1(1 - a) + P_2 a$$

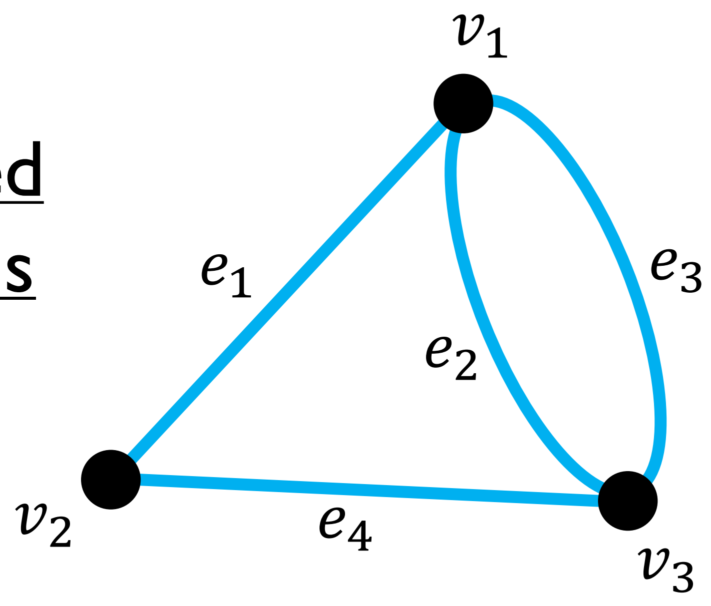
Linear Interpolation



$$lerp(P_1, P_2, a) = P_1(1 - a) + P_2 a$$

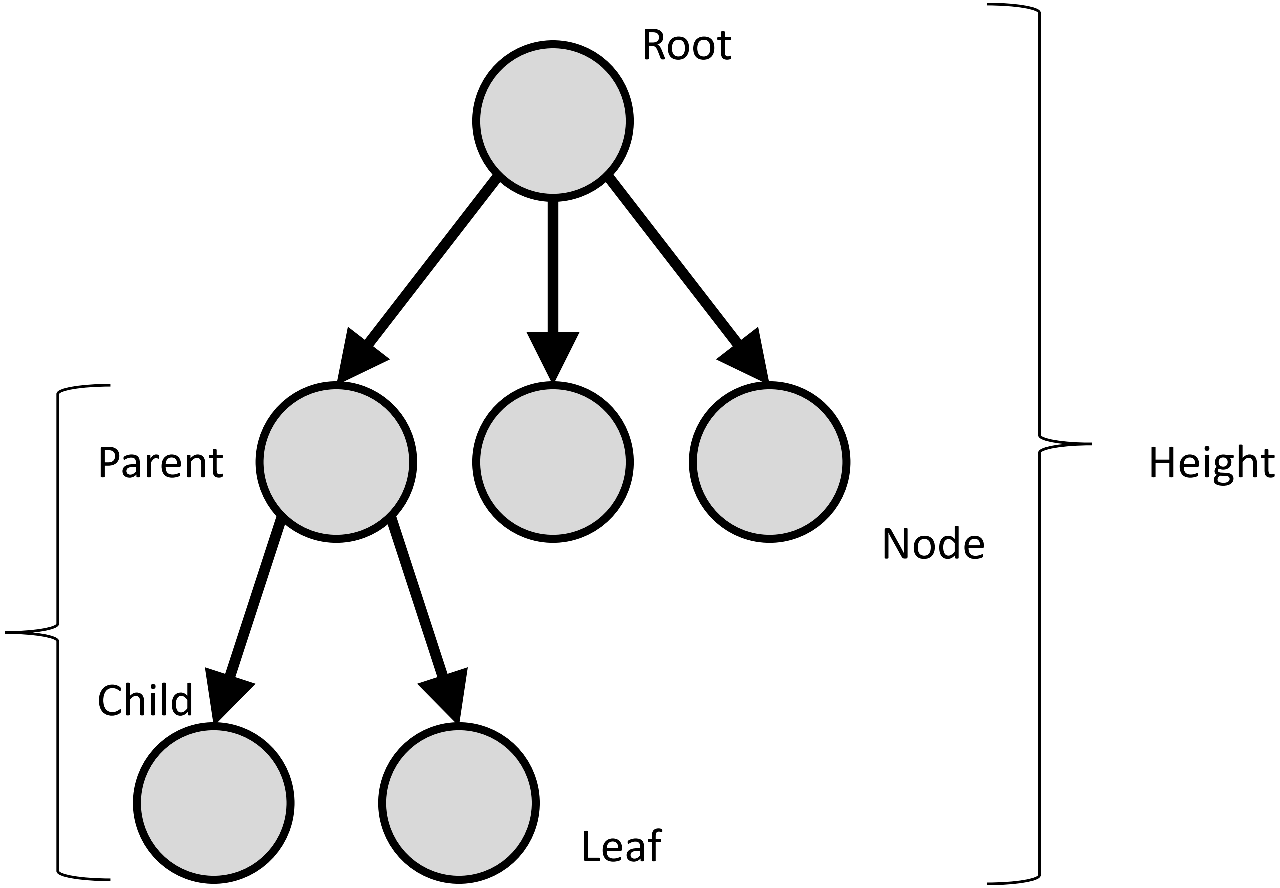
GRAPHS AND TREES

- A graph is construct of two finite sets
  - Vertices, $V = \{v_1, v_2, \ldots, v_n\}$; Edges, E= $\{e_1, e_2, \ldots, e_m\}$
- A ***walk*** from v_i to v_n is a sequence of edges $(v_i, v_j), (v_j, v_k), \ldots, (v_m, v_n)$
- A ***path*** is a walk in which <u>no edge is repeated</u>
- A ***simple path*** is a path in which <u>no vertex is repeated</u>
- A ***cycle*** with base v_i is a walk from v_i to itself with <u>no repeated edges</u>
- A ***loop*** is an edge from a vertex <u>vertex to itself</u>



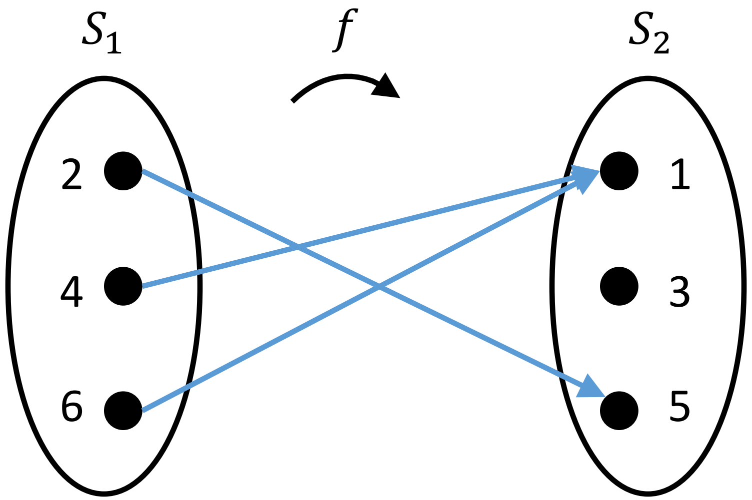
TREES

- A tree is a directed graph that has no cycles, and that has one distinct vertex (the root)



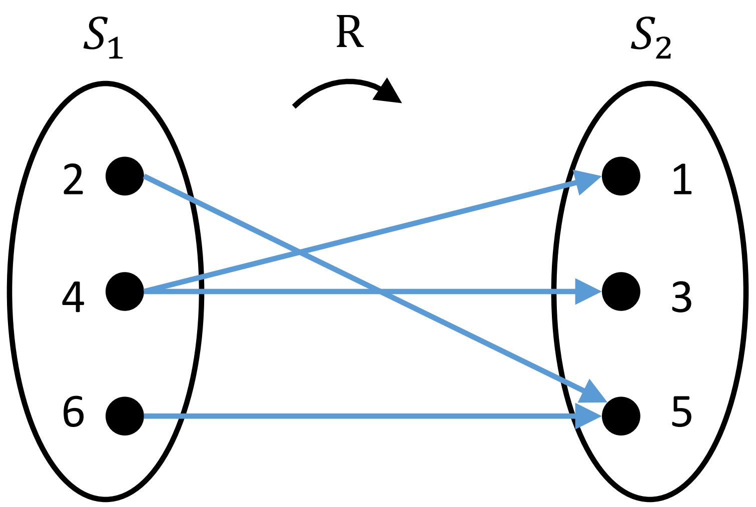
Functions

- A function is a rule that assigns to elements of one set a unique element of another set
  - $f: S_1 \rightarrow S_2$,
  - Where the domain of f is a subset of S_1 and the range of f is the subset of S_2
  - f is a total function on S_1 if the domain of f is all of S_1; otherwise f is a partial function



RELATIONS

- Some functions can be represented by a set of pairs $\{(x_1, y_1), (x_2, y_2), \dots\}$, where x_i is an element in the domain of the function, and y_i is the corresponding value in its range
- For such a set to define a function, each x_i can occur at most once as the first element of a pair.
- If this is not satisfied, the set is called a relation.



FUNCTIONS

- THE BEHAVIOR OF FUNCTIONS:
  - Big O
    - f has order at most g
    - $f(n) \leq c|g(n)| \rightarrow f(n) = O\big(g(n)\big)$
  - Big Omega
    - f has order at least g
    - $|f(n)| \geq c|g(n)| \rightarrow f(n) = \Omega(g(n))$
  - Big Theta
    - f has the same order of magnitude as g
    - $c_1|g(n)| \leq |f(n)| \leq c_2|g(n)| \rightarrow$
      $f(n) = \Theta\big(g(n)\big)$

Function Examples

- The behavior of functions:
  - Big O
    - f has order at most g
    - $f(n) \leq c|g(n)| \rightarrow f(n) = O\big(g(n)\big)$
  - Big Omega
    - f has order at least g
    - $|f(n)| \geq c|g(n)| \rightarrow f(n) = \Omega(g(n))$
  - Big Theta
    - f has the same order of magnitude as g
    - $c_1|g(n)| \leq |f(n)| \leq c_2|g(n)| \rightarrow$
      $f(n) = \Theta\big(g(n)\big)$

- $f(n) = 2n^2 + 4n,$
- $g(n) = n^3,$
- $h(n) = 9n^2 + 300$

True/False

1. $f(n) = O\big(g(n)\big)$?
2. $g(n) = \Omega(h(n))$?
3. $f(n) = \Theta\big(h(n)\big)$?
4. $O(n) + O(n) = 2(O(n))$?

PROOF TECHNIQUES

- A PROOF IS A SEQUENCE OF STEPS THAT LEAD FROM SOME KNOWN FACTS TO THE DESIRED CONCLUSION; EACH STEP MUST BE OBVIOUSLY CORRECT

PROOF TECHNIQUES

- PROOF BY CONTRADICTION:
  - To prove some fact P, we show that "not P" is false
  - That is, we suppose "not P" and demonstrate that it leads to an obviously wrong result
  - E.g.: Prove that $\sqrt{2}$ is not rational. Suppose that is rational, that is $\sqrt{2} = \frac{m}{n}$, where n and m do not have common factors

PROOF TECHNIQUES

- PROOF BY INDUCTION
  - We show that some fact is true for every natural number n, using two arguments:
    - Base: It is true for $n = 1$ (or for some small number)
    - Step: If it is true for n, then it is true for $n + 1$
  - E.g.: prove that $1 + 2 + \cdots + n = n(n+1)/2$