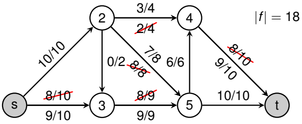
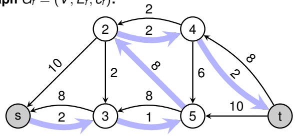
Graph $G = (V, E, c)$**:**



Residual Graph $G_f = (V, E_f, c_f)$**:**



6.3: Maximum Flows

Frank Stajano Thomas Sauerwald

Lent 2015

UNIVERSITY OF CAMBRIDGE

Announcements

- Deadline for Microchallenge 7 today!
- There is a list of errata for the slides on the webpage
- There might be a little bit of time in the last lecture to revisit one of the previous topics and briefly discuss some data structure/algorithm/proof etc. which may or may not have been covered in previous lectures.
  If you have any suggestion, please send an email today.

Outline

Analysis of Ford-Fulkerson

Matchings in Bipartite Graphs

Introduction and Line Intersection

Max-Flow Min-Cut Theorem

Theorem

The value of the max-flow is equal to the capacity of the min-cut, that is

$$\max_{f} |f| = \min_{S,T \subseteq V} \text{cap}(S, T).$$

Analysis of Ford-Fulkerson

```
def FordFulkerson(G)
    initialize flow to 0 on all edges
    while an augmenting path in G_f can be found:
        push as much extra flow as possible through it
```

Analysis of Ford-Fulkerson

```
def FordFulkerson(G)
    initialize flow to 0 on all edges
    while an augmenting path in G_f can be found:
        push as much extra flow as possible through it
```

Lemma

If all capacities $c(u, v)$ are integral, then the flow at every iteration of Ford-Fulkerson is integral.

Analysis of Ford-Fulkerson

```
def FordFulkerson(G)
    initialize flow to 0 on all edges
    while an augmenting path in G_f can be found:
        push as much extra flow as possible through it
```

Lemma

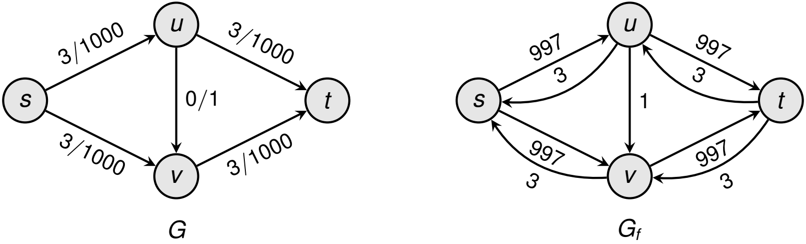
If all capacities $c(u, v)$ are integral, then the flow at every iteration of Ford-Fulkerson is integral.

Theorem

For integral capacities $c(u, v)$, Ford-Fulkerson terminates after $V \cdot C$ *iterations*, where $C := \max_{u,v} c(u, v)$ and returns the maximum flow.

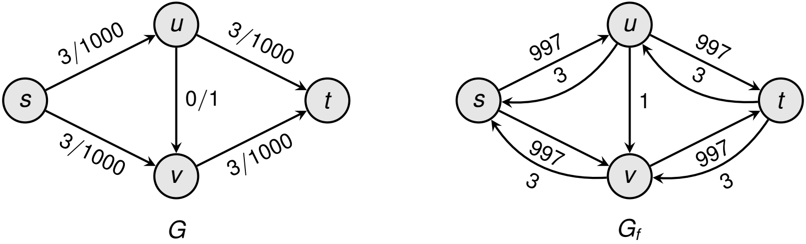
at the time of termination, no augmenting path
$\Rightarrow$ Ford-Fulkerson returns maxflow (Key Lemma)

Slow Convergence of Ford-Fulkerson (Figure 26.7)



Number of iterations is at least $C := \max_{u,v} c(u, v)$!

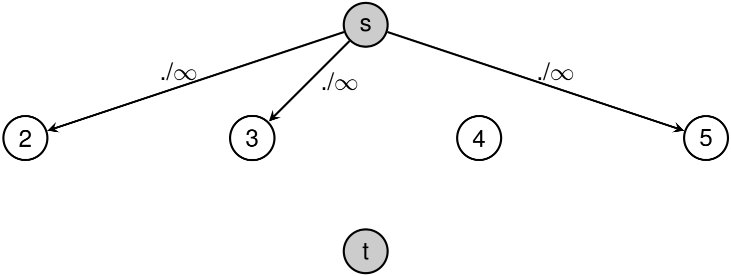
Slow Convergence of Ford-Fulkerson (Figure 26.7)



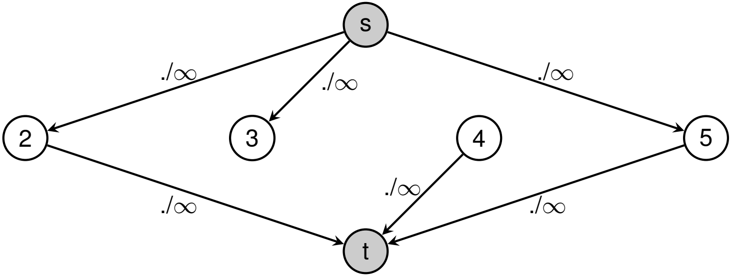
Number of iterations is at least $C := \max_{u,v} c(u,v)$!

For irrational capacities, Ford-Fulkerson may even fail to terminate!

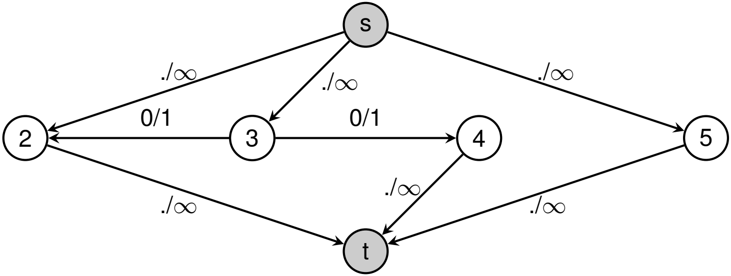
Non-Termination of Ford-Fulkerson for Irrational Capacities



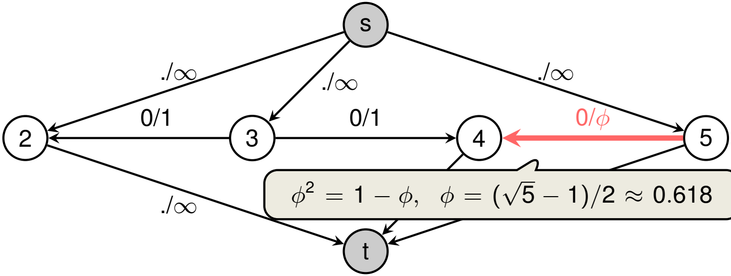
Non-Termination of Ford-Fulkerson for Irrational Capacities



Non-Termination of Ford-Fulkerson for Irrational Capacities

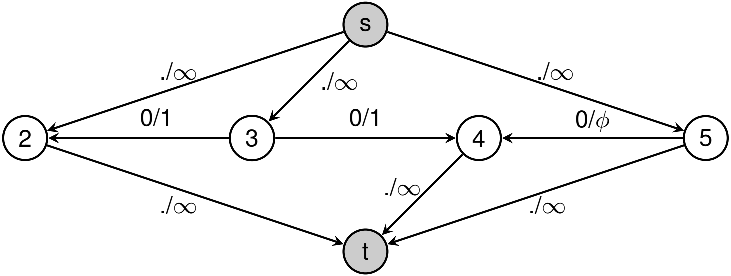


Non-Termination of Ford-Fulkerson for Irrational Capacities

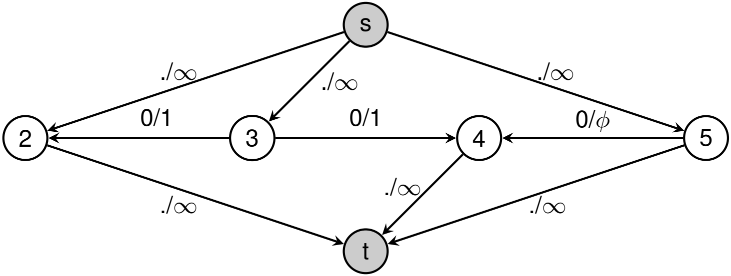


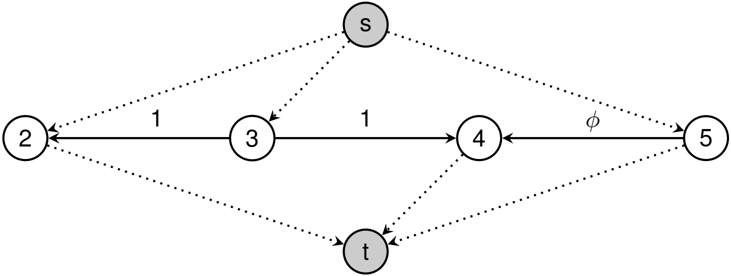
$$\phi^2 = 1 - \phi, \quad \phi = (\sqrt{5}-1)/2 \approx 0.618$$

Non-Termination of Ford-Fulkerson for Irrational Capacities

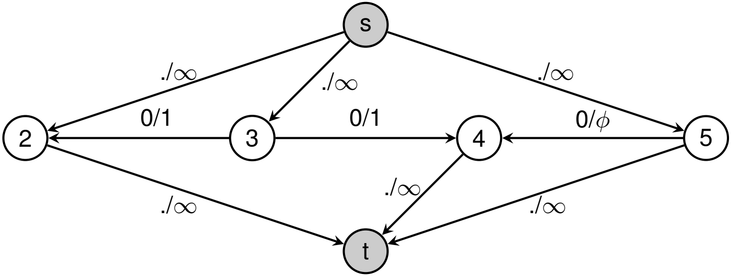


Non-Termination of Ford-Fulkerson for Irrational Capacities

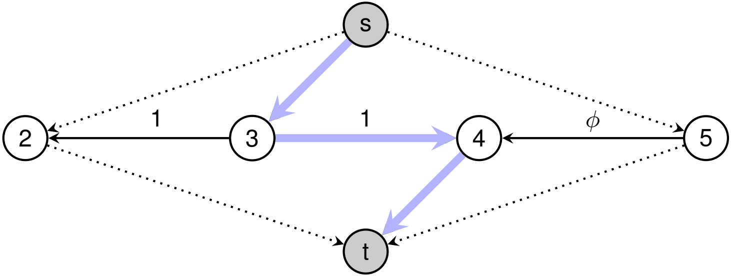




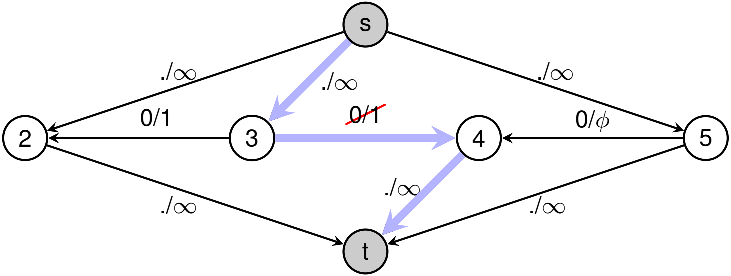
Non-Termination of Ford-Fulkerson for Irrational Capacities



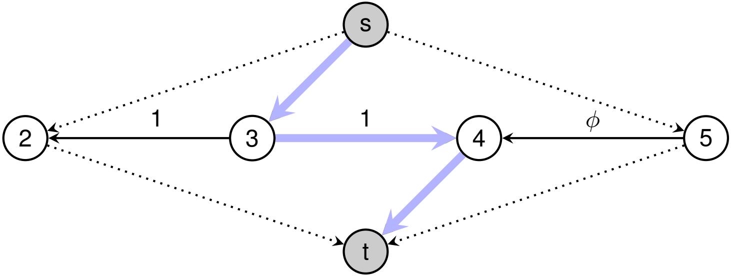
Iteration: 1, $|f| = 0$



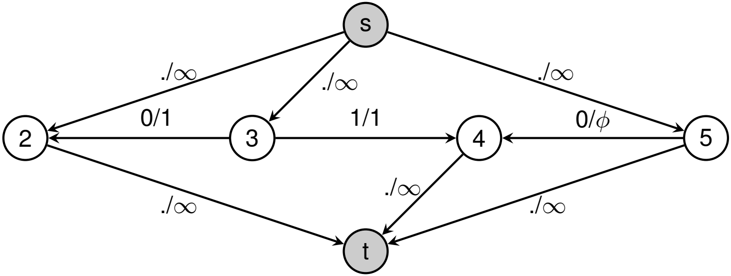
Non-Termination of Ford-Fulkerson for Irrational Capacities



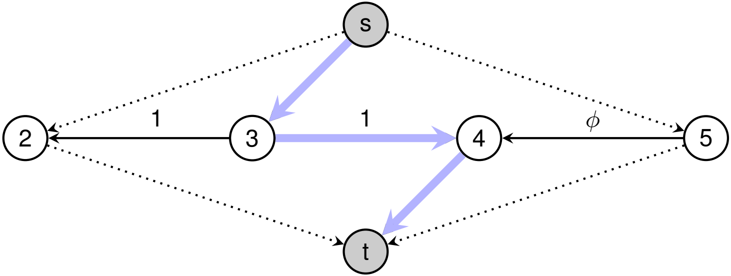
Iteration: 1, $|f| = 0$



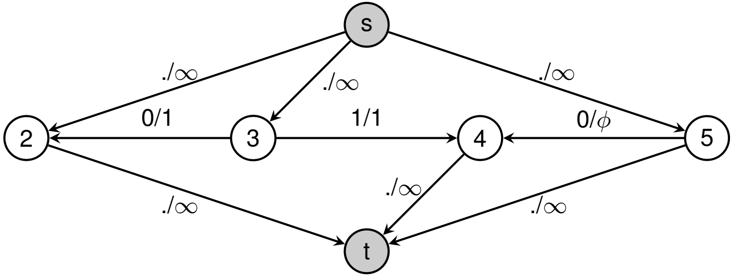
Non-Termination of Ford-Fulkerson for Irrational Capacities



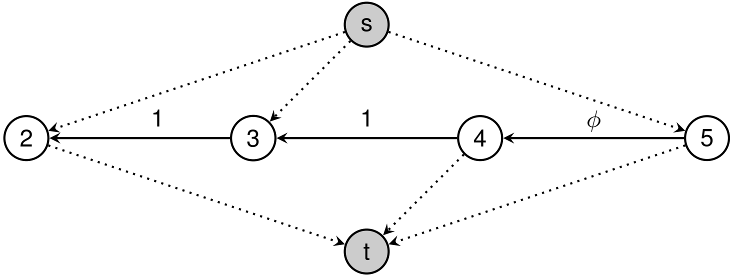
Iteration: 1, $|f| = 1$



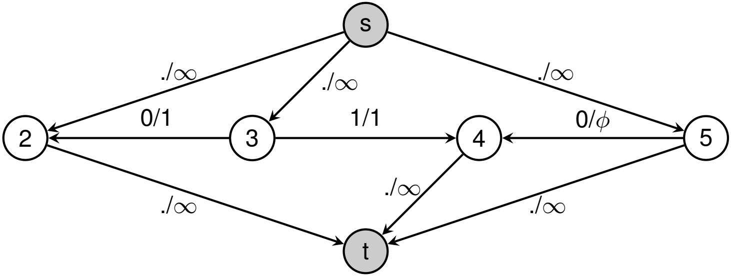
Non-Termination of Ford-Fulkerson for Irrational Capacities



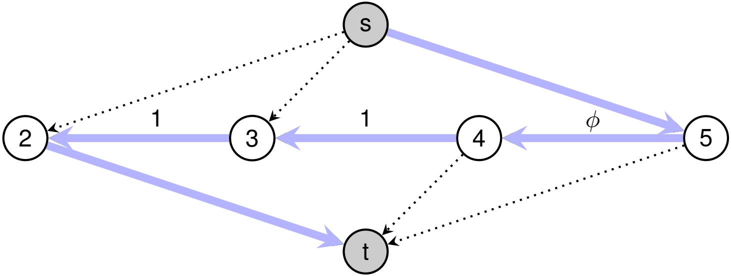
Iteration: 1, $|f| = 1$



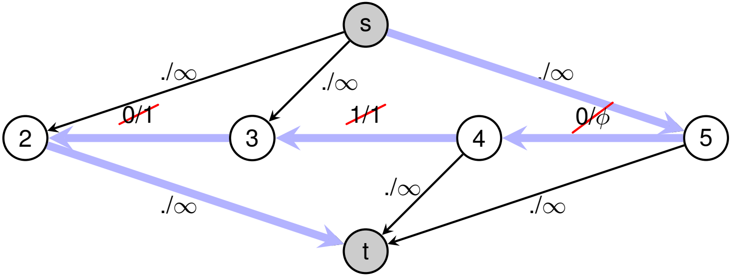
Non-Termination of Ford-Fulkerson for Irrational Capacities



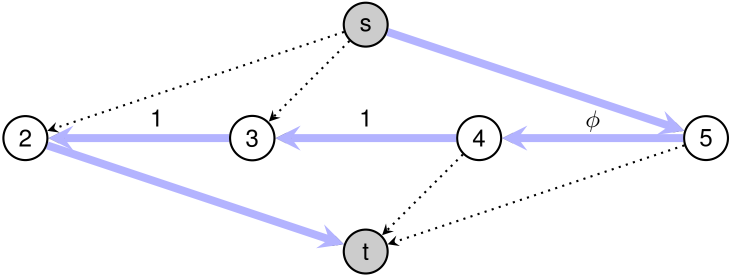
Iteration: 2, $|f| = 1$



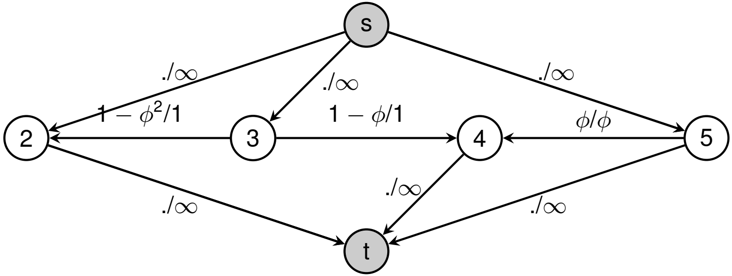
Non-Termination of Ford-Fulkerson for Irrational Capacities



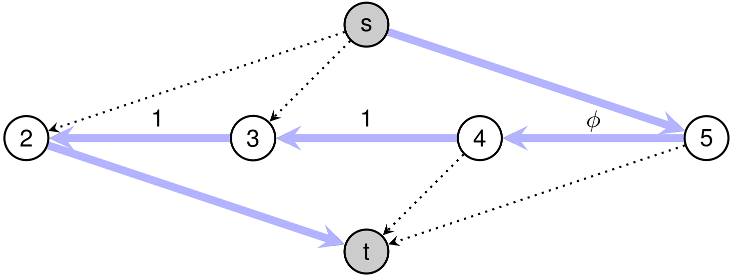
Iteration: 2, $|f| = 1$



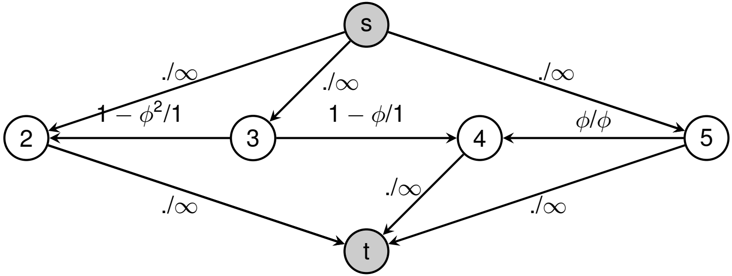
Non-Termination of Ford-Fulkerson for Irrational Capacities



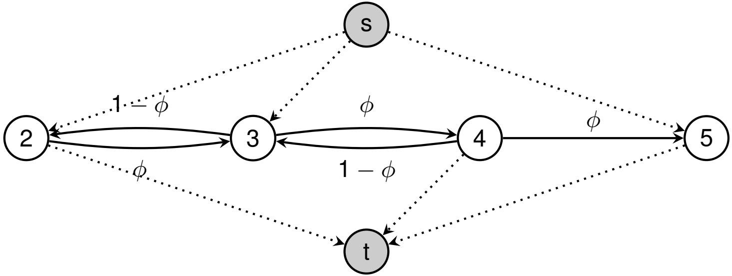
Iteration: 2, $|f| = 1 + \phi$



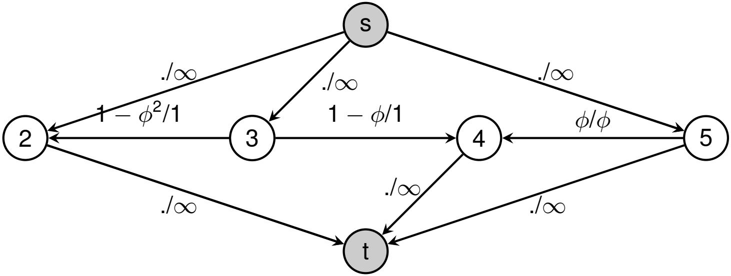
Non-Termination of Ford-Fulkerson for Irrational Capacities



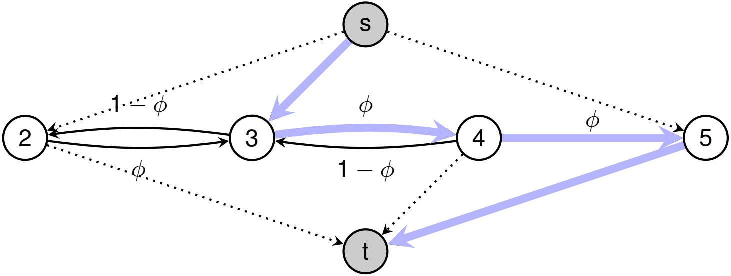
Iteration: 2, $|f| = 1 + \phi$



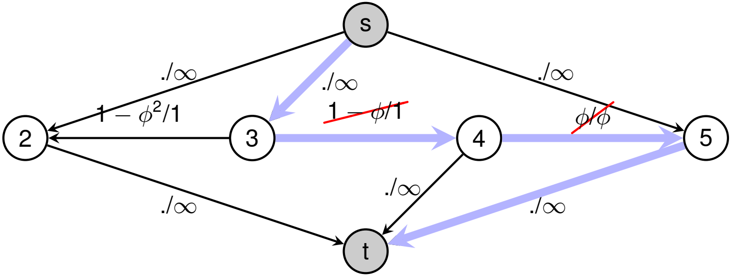
Non-Termination of Ford-Fulkerson for Irrational Capacities



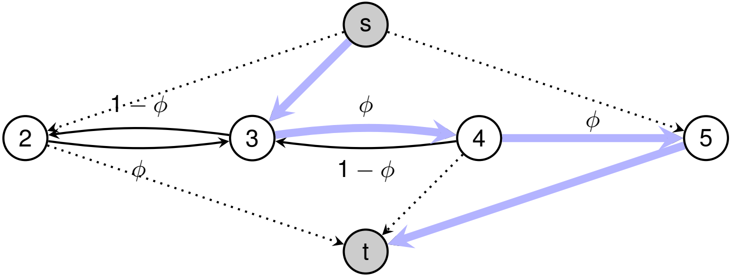
Iteration: 3, $|f| = 1 + \phi$



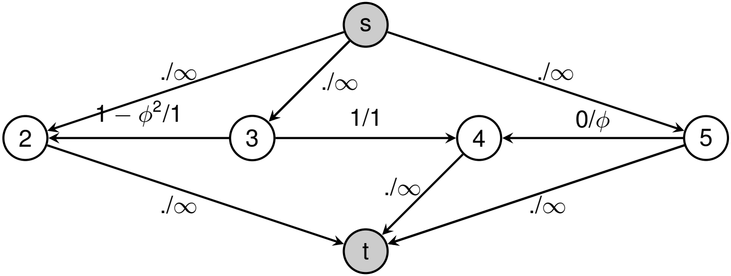
Non-Termination of Ford-Fulkerson for Irrational Capacities



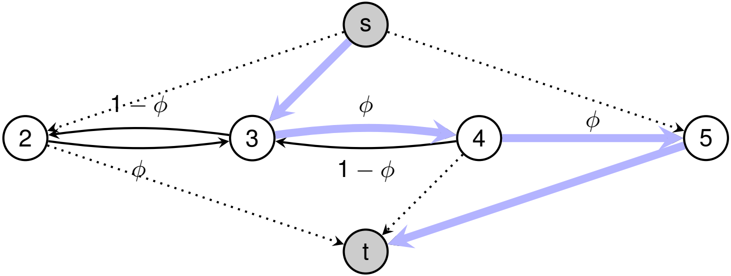
Iteration: 3, $|f| = 1 + \phi$



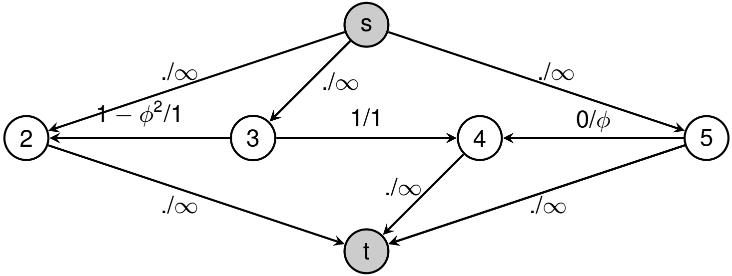
Non-Termination of Ford-Fulkerson for Irrational Capacities



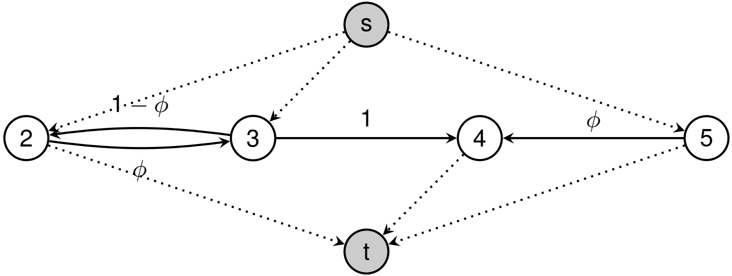
Iteration: 3, $|f| = 1 + 2 \cdot \phi$



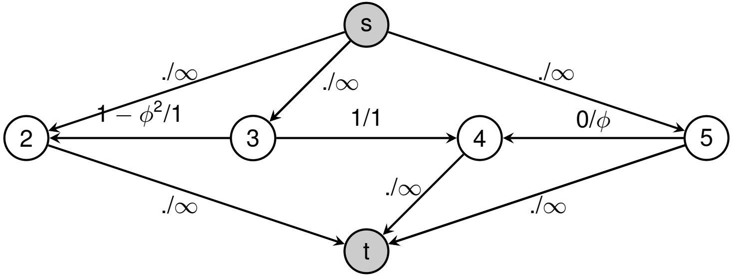
Non-Termination of Ford-Fulkerson for Irrational Capacities



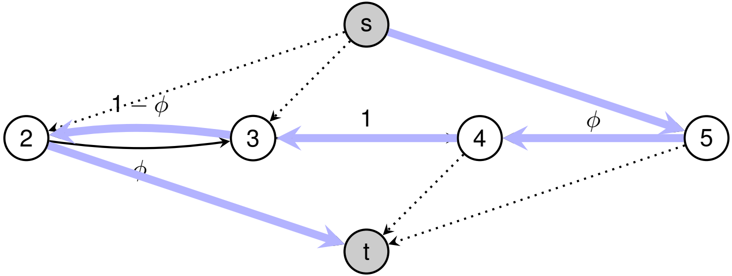
Iteration: 3, $|f| = 1 + 2 \cdot \phi$



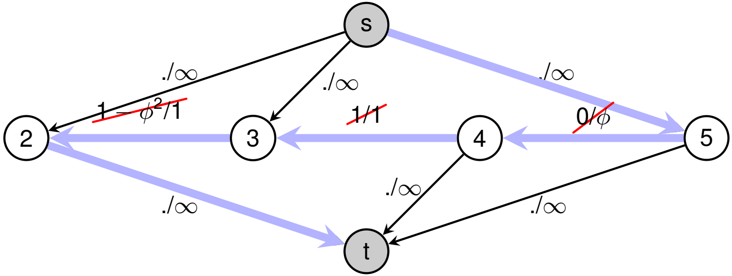
Non-Termination of Ford-Fulkerson for Irrational Capacities



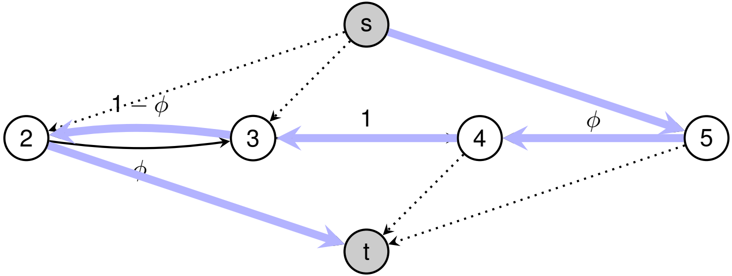
Iteration: 4, $|f| = 1 + 2 \cdot \phi$



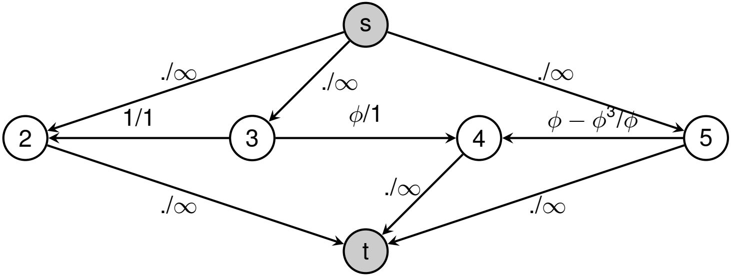
Non-Termination of Ford-Fulkerson for Irrational Capacities



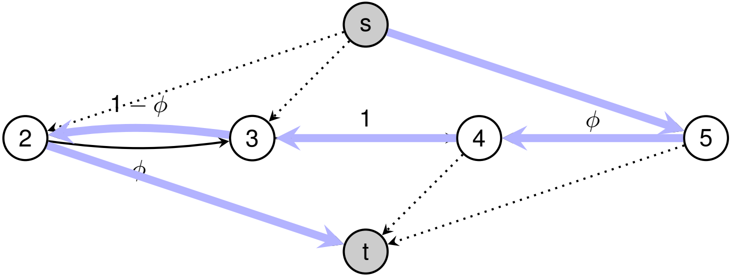
Iteration: 4, $|f| = 1 + 2 \cdot \phi$



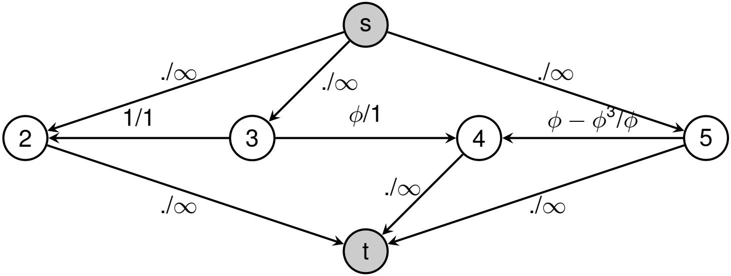
Non-Termination of Ford-Fulkerson for Irrational Capacities



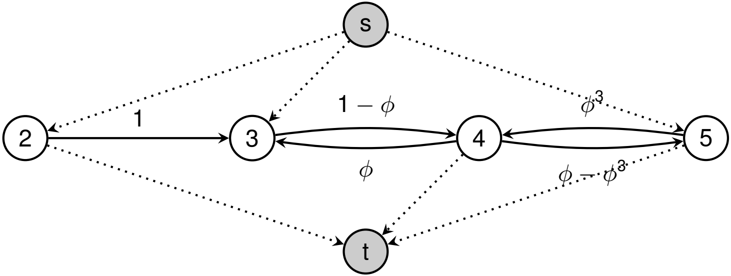
Iteration: 4, $|f| = 1 + 2 \cdot \phi + \phi^2$



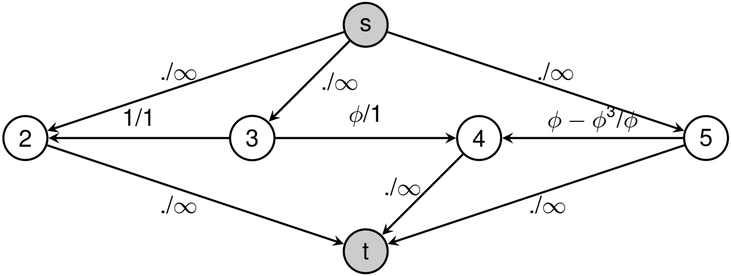
Non-Termination of Ford-Fulkerson for Irrational Capacities



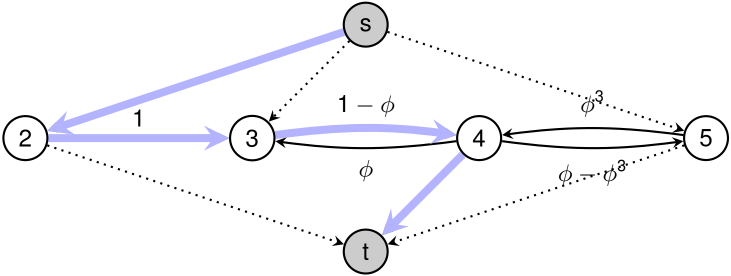
Iteration: 4, $|f| = 1 + 2 \cdot \phi + \phi^2$



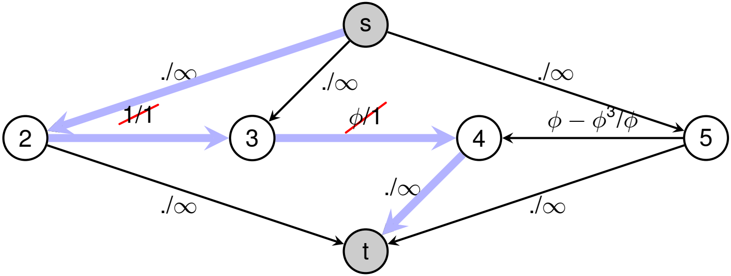
Non-Termination of Ford-Fulkerson for Irrational Capacities



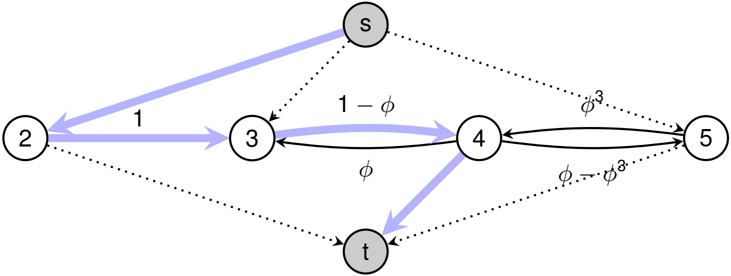
Iteration: 5, $|f| = 1 + 2 \cdot \phi + \phi^2$



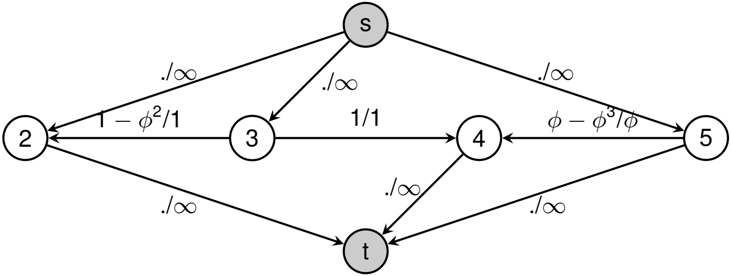
Non-Termination of Ford-Fulkerson for Irrational Capacities



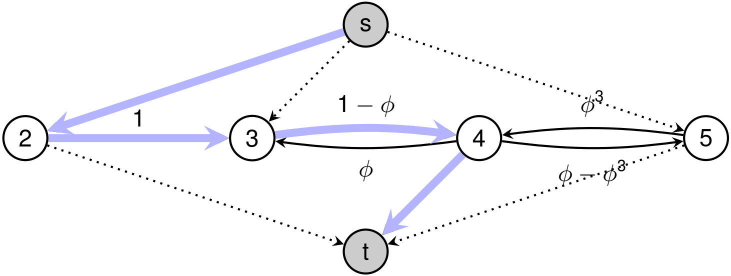
Iteration: 5, $|f| = 1 + 2 \cdot \phi + \phi^2$



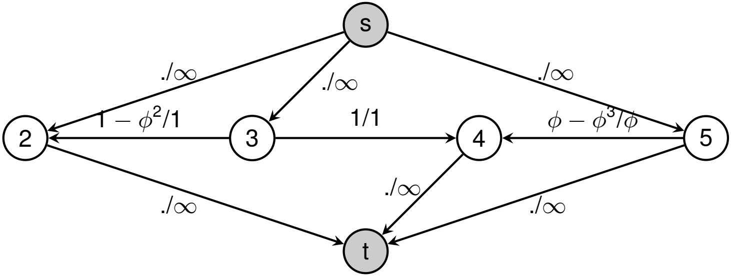
Non-Termination of Ford-Fulkerson for Irrational Capacities



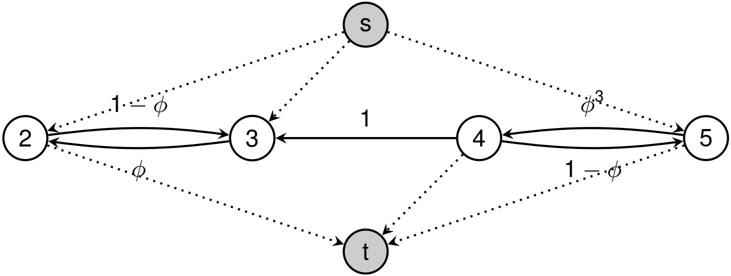
Iteration: 5, $|f| = 1 + 2 \cdot \phi + 2 \cdot \phi^2$



Non-Termination of Ford-Fulkerson for Irrational Capacities

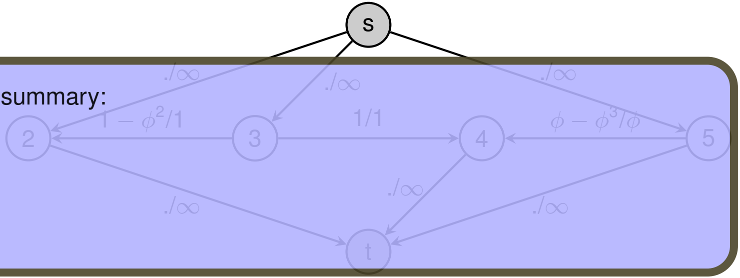


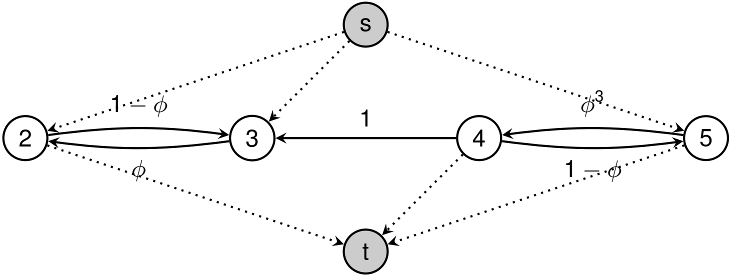
Iteration: 5, $|f| = 1 + 2 \cdot \phi + 2 \cdot \phi^2$



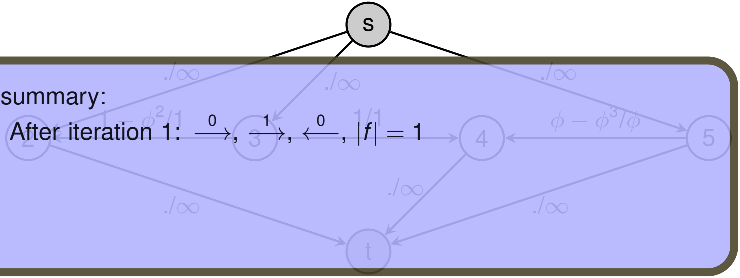
Non-Termination of Ford-Fulkerson for Irrational Capacities

In summary:



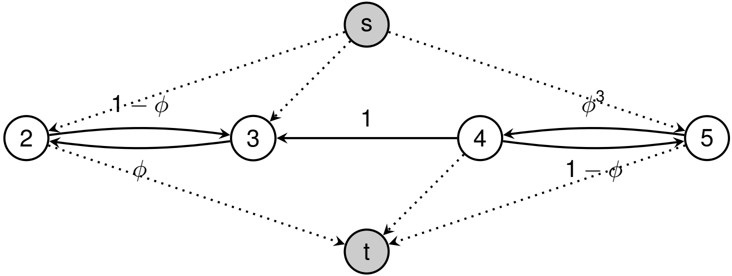


Non-Termination of Ford-Fulkerson for Irrational Capacities

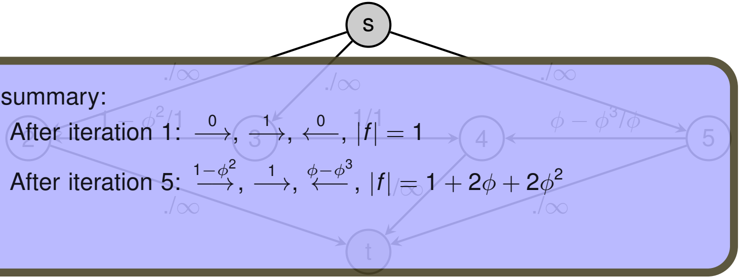


In summary:

- After iteration 1: $\xrightarrow{0}$, $\xrightarrow{1}$, $\xleftarrow{0}$, $|f| = 1$

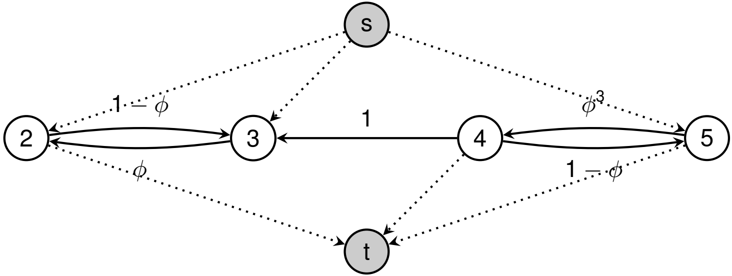


Non-Termination of Ford-Fulkerson for Irrational Capacities



In summary:

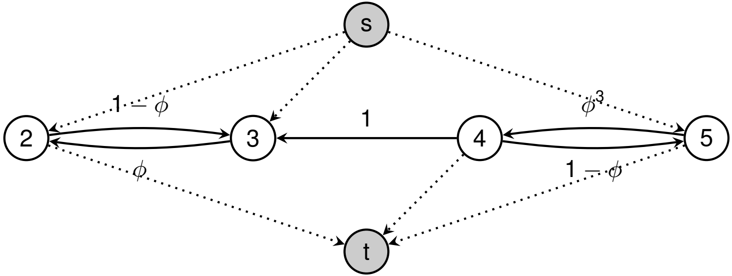
- After iteration 1: $\xrightarrow{0}$, $\xrightarrow{1}$, $\xleftarrow{0}$, $|f| = 1$
- After iteration 5: $\xrightarrow{1-\phi^2}$, $\xrightarrow{1}$, $\xleftarrow{\phi-\phi^3}$, $|f| = 1 + 2\phi + 2\phi^2$



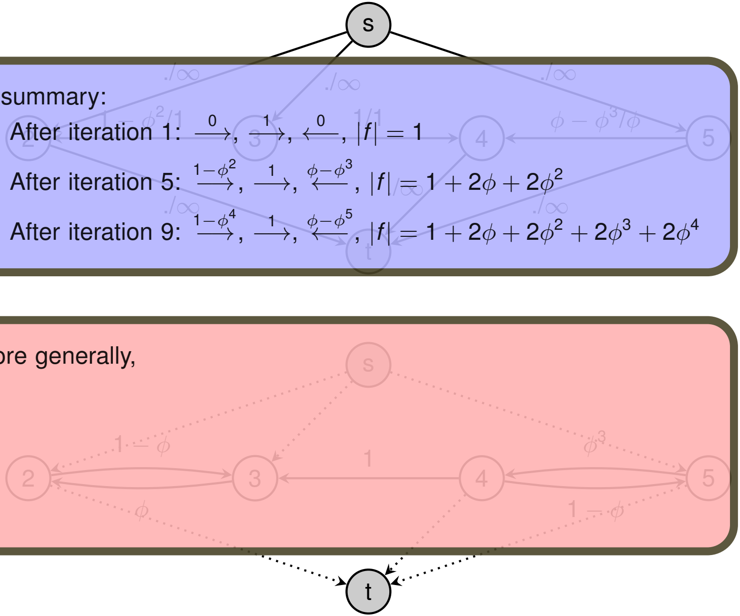
Non-Termination of Ford-Fulkerson for Irrational Capacities

In summary:

- After iteration 1: $\xrightarrow{0}$, $\xrightarrow{1}$, $\xleftarrow{0}$, $|f| = 1$
- After iteration 5: $\xrightarrow{1-\phi^2}$, $\xrightarrow{1}$, $\xleftarrow{\phi-\phi^3}$, $|f| = 1 + 2\phi + 2\phi^2$
- After iteration 9: $\xrightarrow{1-\phi^4}$, $\xrightarrow{1}$, $\xleftarrow{\phi-\phi^5}$, $|f| = 1 + 2\phi + 2\phi^2 + 2\phi^3 + 2\phi^4$



Non-Termination of Ford-Fulkerson for Irrational Capacities

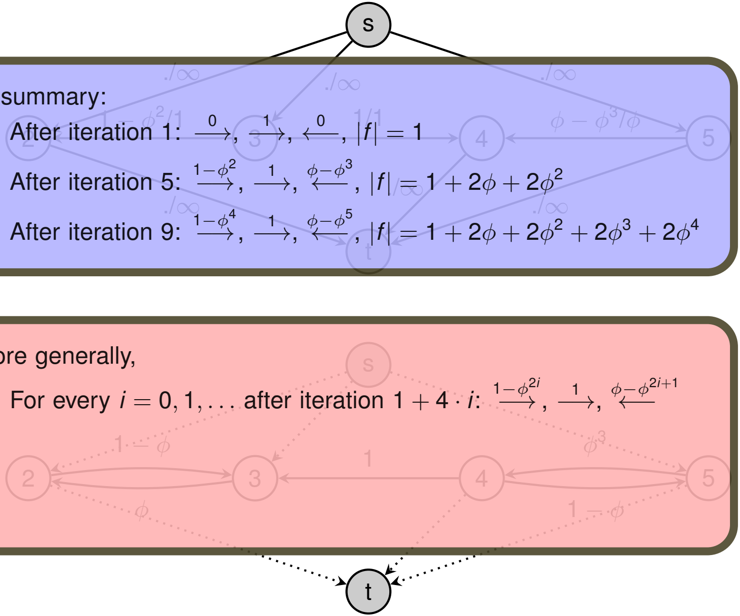


In summary:

- After iteration 1: $\xrightarrow{0}$, $\xrightarrow{1}$, $\xleftarrow{0}$, $|f| = 1$
- After iteration 5: $\xrightarrow{1-\phi^2}$, $\xrightarrow{1}$, $\xleftarrow{\phi-\phi^3}$, $|f| = 1 + 2\phi + 2\phi^2$
- After iteration 9: $\xrightarrow{1-\phi^4}$, $\xrightarrow{1}$, $\xleftarrow{\phi-\phi^5}$, $|f| = 1 + 2\phi + 2\phi^2 + 2\phi^3 + 2\phi^4$

More generally,

Non-Termination of Ford-Fulkerson for Irrational Capacities



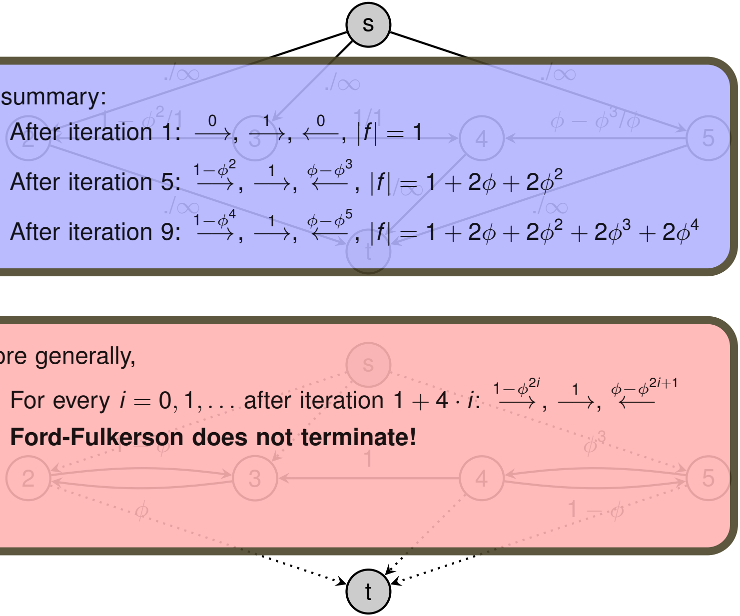
In summary:

- After iteration 1: $\xrightarrow{0}$, $\xrightarrow{1}$, $\xleftarrow{0}$, $|f| = 1$
- After iteration 5: $\xrightarrow{1-\phi^2}$, $\xrightarrow{1}$, $\xleftarrow{\phi-\phi^3}$, $|f| = 1 + 2\phi + 2\phi^2$
- After iteration 9: $\xrightarrow{1-\phi^4}$, $\xrightarrow{1}$, $\xleftarrow{\phi-\phi^5}$, $|f| = 1 + 2\phi + 2\phi^2 + 2\phi^3 + 2\phi^4$

More generally,

- For every $i = 0, 1, \ldots$ after iteration $1 + 4 \cdot i$: $\xrightarrow{1-\phi^{2i}}$, $\xrightarrow{1}$, $\xleftarrow{\phi-\phi^{2i+1}}$

Non-Termination of Ford-Fulkerson for Irrational Capacities



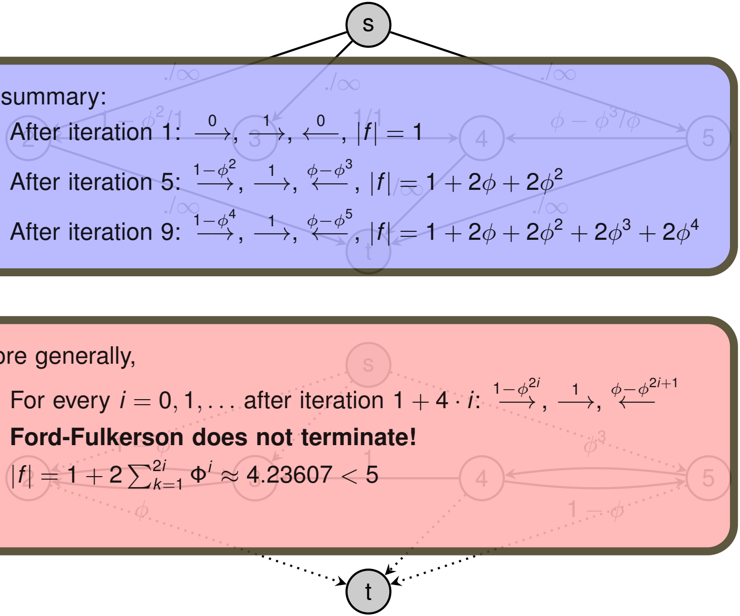
In summary:

- After iteration 1: $\xrightarrow{0}$, $\xrightarrow{1}$, $\xleftarrow{0}$, $|f| = 1$
- After iteration 5: $\xrightarrow{1-\phi^2}$, $\xrightarrow{1}$, $\xleftarrow{\phi-\phi^3}$, $|f| = 1 + 2\phi + 2\phi^2$
- After iteration 9: $\xrightarrow{1-\phi^4}$, $\xrightarrow{1}$, $\xleftarrow{\phi-\phi^5}$, $|f| = 1 + 2\phi + 2\phi^2 + 2\phi^3 + 2\phi^4$

More generally,

- For every $i = 0, 1, \ldots$ after iteration $1 + 4 \cdot i$: $\xrightarrow{1-\phi^{2i}}$, $\xrightarrow{1}$, $\xleftarrow{\phi-\phi^{2i+1}}$
- **Ford-Fulkerson does not terminate!**

Non-Termination of Ford-Fulkerson for Irrational Capacities

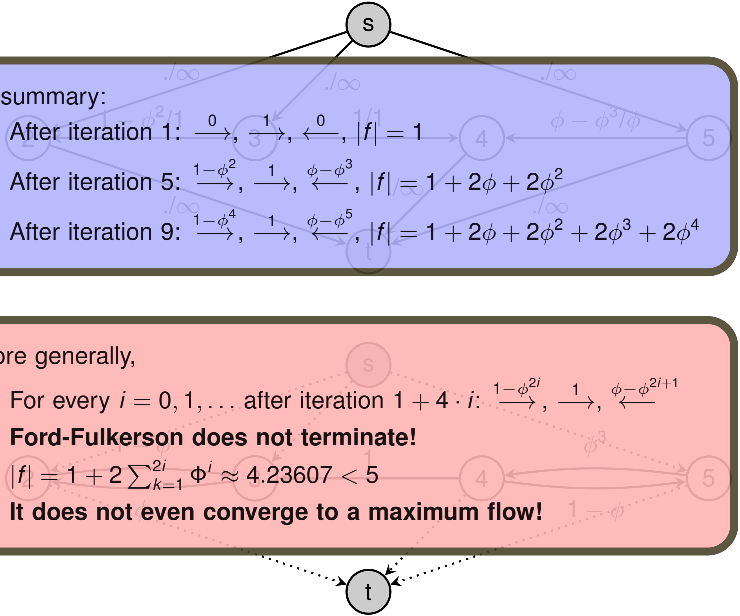


In summary:

- After iteration 1: $\xrightarrow{0}$, $\xrightarrow{1}$, $\xleftarrow{0}$, $|f| = 1$
- After iteration 5: $\xrightarrow{1-\phi^2}$, $\xrightarrow{1}$, $\xleftarrow{\phi-\phi^3}$, $|f| = 1 + 2\phi + 2\phi^2$
- After iteration 9: $\xrightarrow{1-\phi^4}$, $\xrightarrow{1}$, $\xleftarrow{\phi-\phi^5}$, $|f| = 1 + 2\phi + 2\phi^2 + 2\phi^3 + 2\phi^4$

More generally,

- For every $i = 0, 1, \ldots$ after iteration $1 + 4 \cdot i$: $\xrightarrow{1-\phi^{2i}}$, $\xrightarrow{1}$, $\xleftarrow{\phi-\phi^{2i+1}}$
- **Ford-Fulkerson does not terminate!**
- $|f| = 1 + 2\sum_{k=1}^{2i} \Phi^i \approx 4.23607 < 5$



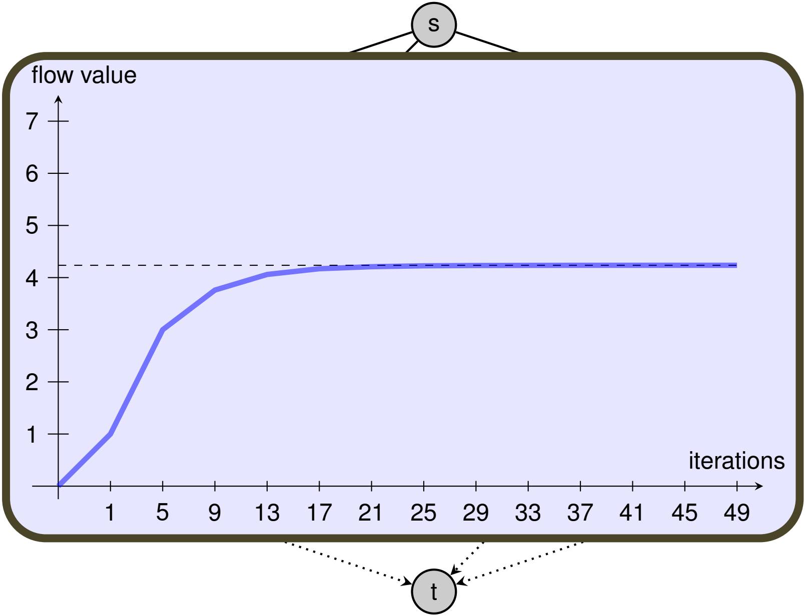
In summary:

- After iteration 1: $\xrightarrow{0}$, $\xrightarrow{1}$, $\xleftarrow{0}$, $|f| = 1$
- After iteration 5: $\xrightarrow{1-\phi^2}$, $\xrightarrow{1}$, $\xleftarrow{\phi-\phi^3}$, $|f| = 1 + 2\phi + 2\phi^2$
- After iteration 9: $\xrightarrow{1-\phi^4}$, $\xrightarrow{1}$, $\xleftarrow{\phi-\phi^5}$, $|f| = 1 + 2\phi + 2\phi^2 + 2\phi^3 + 2\phi^4$

More generally,

- For every $i = 0, 1, \ldots$ after iteration $1 + 4 \cdot i$: $\xrightarrow{1-\phi^{2i}}$, $\xrightarrow{1}$, $\xleftarrow{\phi-\phi^{2i+1}}$
- **Ford-Fulkerson does not terminate!**
- $|f| = 1 + 2\sum_{k=1}^{2i} \Phi^i \approx 4.23607 < 5$
- **It does not even converge to a maximum flow!**

Non-Termination of Ford-Fulkerson for Irrational Capacities



Summary and Outlook

Ford-Fulkerson Method

- works only for integral (rational) capacities

Summary and Outlook

Ford-Fulkerson Method

- works only for integral (rational) capacities
- Runtime: $O(E \cdot |f^{\max}|) = O(E \cdot V \cdot C)$

Summary and Outlook

Ford-Fulkerson Method

- works only for integral (rational) capacities
- Runtime: $O(E \cdot |f^{\max}|) = O(E \cdot V \cdot C)$

Capacity-Scaling Algorithm

Summary and Outlook

Ford-Fulkerson Method

- works only for integral (rational) capacities
- Runtime: $O(E \cdot |f^{\max}|) = O(E \cdot V \cdot C)$

Capacity-Scaling Algorithm

- Idea: Find an augmenting path with high capacity
- Consider subgraph of G_f consisting of edges (u, v) with $c_f(u, v) > \Delta$
- scaling parameter Δ, which is initially $2^{\lceil \log_2 C \rceil}$ and 1 after termination
- Runtime: $O(E^2 \cdot \log C)$

Summary and Outlook

Ford-Fulkerson Method

- works only for integral (rational) capacities
- Runtime: $O(E \cdot |f^{\max}|) = O(E \cdot V \cdot C)$

Capacity-Scaling Algorithm

- Idea: Find an augmenting path with high capacity
- Consider subgraph of G_f consisting of edges (u, v) with $c_f(u, v) > \Delta$
- scaling parameter Δ, which is initially $2^{\lceil \log_2 C \rceil}$ and 1 after termination
- Runtime: $O(E^2 \cdot \log C)$

Edmonds-Karp Algorithm

Summary and Outlook

Ford-Fulkerson Method

- works only for integral (rational) capacities
- Runtime: $O(E \cdot |f^{\max}|) = O(E \cdot V \cdot C)$

Capacity-Scaling Algorithm

- Idea: Find an augmenting path with high capacity
- Consider subgraph of G_f consisting of edges (u, v) with $c_f(u, v) > \Delta$
- scaling parameter Δ, which is initially $2^{\lceil \log_2 C \rceil}$ and 1 after termination
- Runtime: $O(E^2 \cdot \log C)$

Edmonds-Karp Algorithm

- Idea: Find the shortest augmenting path in G_f
- Runtime: $O(E^2 \cdot V)$

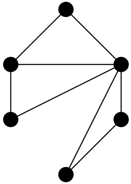
Outline

Application: Maximum-Bipartite-Matching Problem

Matching

A matching is a subset $M \subseteq E$ such that for all $v \in V$, at most one edge of M is incident to v.

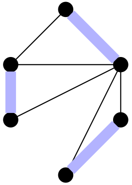
Application: Maximum-Bipartite-Matching Problem



Matching

A matching is a subset $M \subseteq E$ such that for all $v \in V$, at most one edge of M is incident to v.

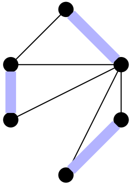
Application: Maximum-Bipartite-Matching Problem



Matching

A matching is a subset $M \subseteq E$ such that for all $v \in V$, at most one edge of M is incident to v.

Application: Maximum-Bipartite-Matching Problem

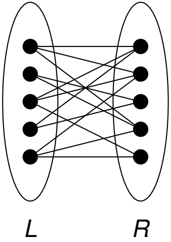


Matching

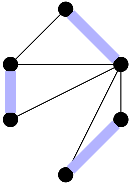
A matching is a subset $M \subseteq E$ such that for all $v \in V$, at most one edge of M is incident to v.

Bipartite Graph

A graph G is bipartite if V can be partitioned into L and R so that all edges go between L and R.



Application: Maximum-Bipartite-Matching Problem

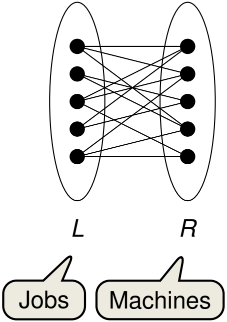


Matching

A matching is a subset $M \subseteq E$ such that for all $v \in V$, at most one edge of M is incident to v.

Bipartite Graph

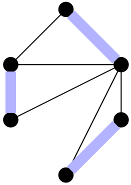
A graph G is bipartite if V can be partitioned into L and R so that all edges go between L and R.



Application: Maximum-Bipartite-Matching Problem

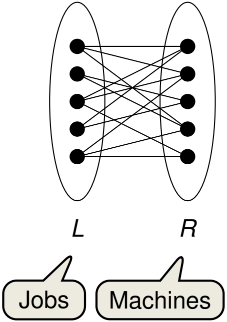
Matching

A matching is a subset $M \subseteq E$ such that for all $v \in V$, at most one edge of M is incident to v.



Bipartite Graph

A graph G is bipartite if V can be partitioned into L and R so that all edges go between L and R.

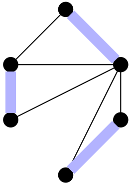


Given a bipartite graph $G = (L \cup R, E)$, find a matching of maximum cardinality.

Application: Maximum-Bipartite-Matching Problem

Matching

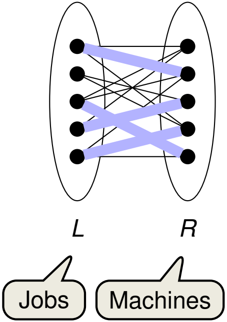
A matching is a subset $M \subseteq E$ such that for all $v \in V$, at most one edge of M is incident to v.



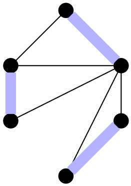
Bipartite Graph

A graph G is bipartite if V can be partitioned into L and R so that all edges go between L and R.

Given a bipartite graph $G = (L \cup R, E)$, find a matching of maximum cardinality.



Application: Maximum-Bipartite-Matching Problem

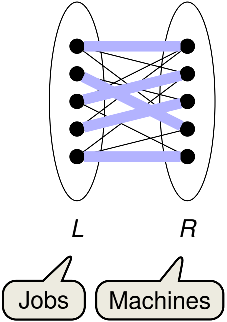


Matching

A matching is a subset $M \subseteq E$ such that for all $v \in V$, at most one edge of M is incident to v.

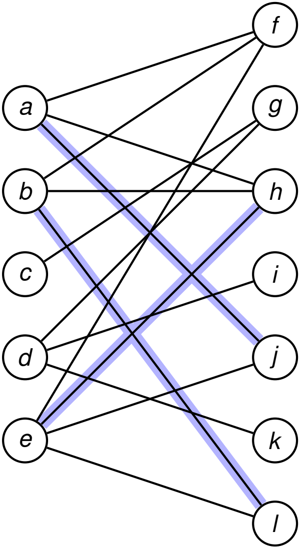
Bipartite Graph

A graph G is bipartite if V can be partitioned into L and R so that all edges go between L and R.

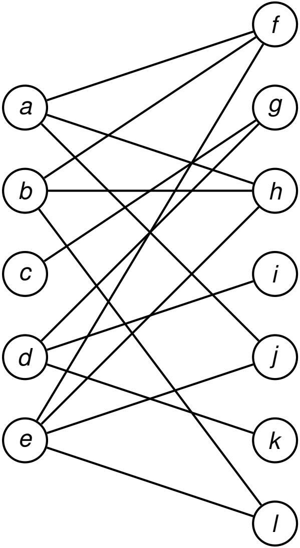


Given a bipartite graph $G = (L \cup R, E)$, find a matching of maximum cardinality.

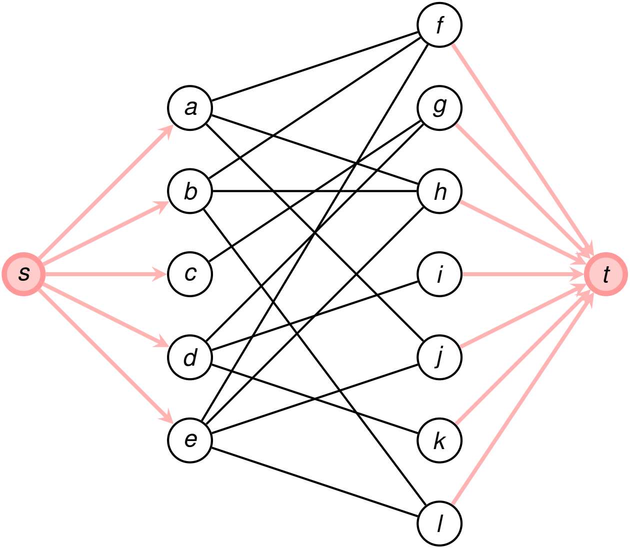
Matchings in Bipartite Graphs via Maximum Flows



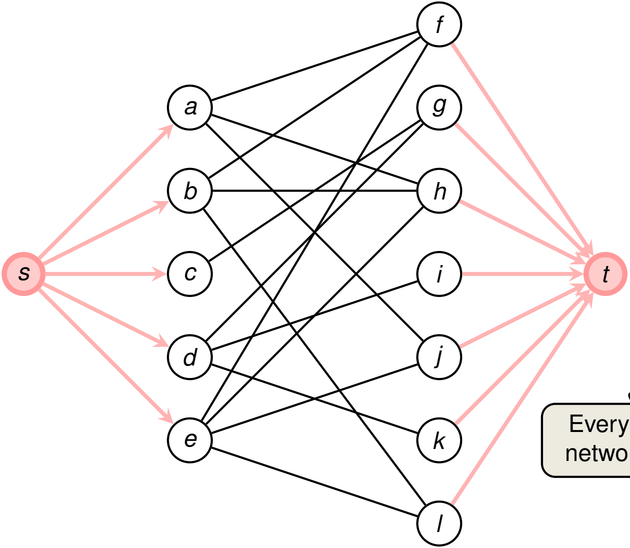
Matchings in Bipartite Graphs via Maximum Flows



Matchings in Bipartite Graphs via Maximum Flows

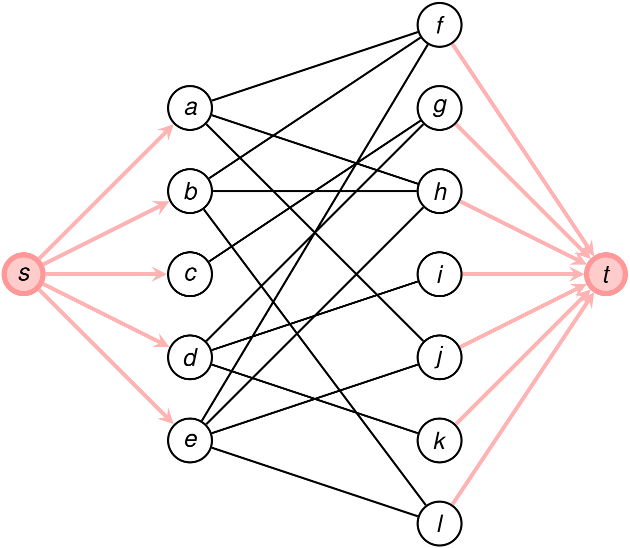


Matchings in Bipartite Graphs via Maximum Flows

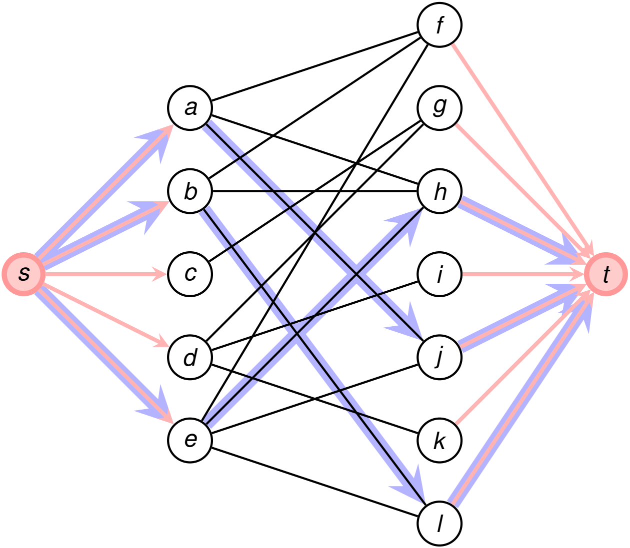


Every edge in the flow network has capacity 1

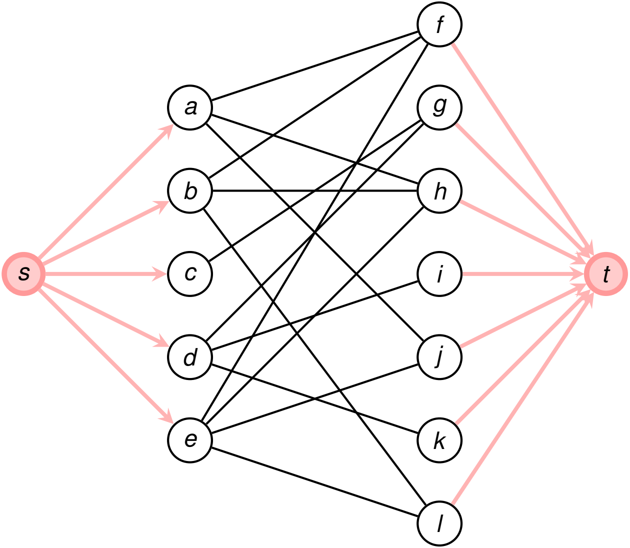
Matchings in Bipartite Graphs via Maximum Flows



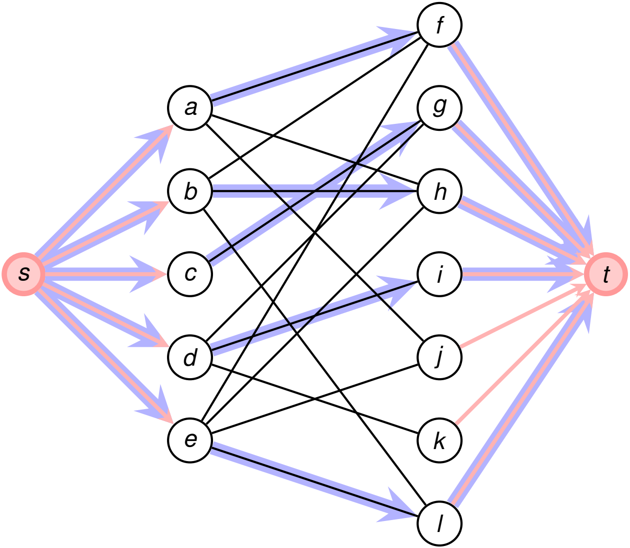
Matchings in Bipartite Graphs via Maximum Flows



Matchings in Bipartite Graphs via Maximum Flows



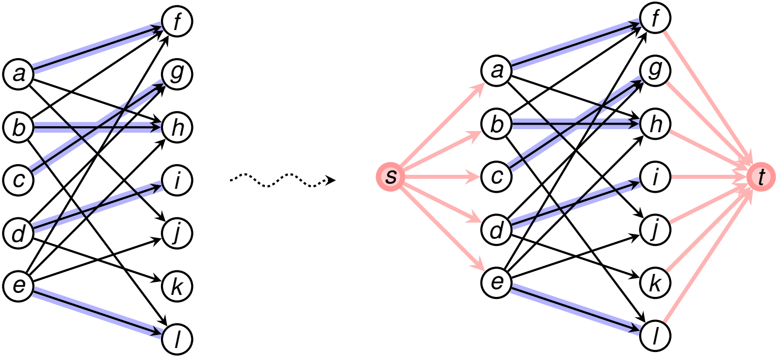
Matchings in Bipartite Graphs via Maximum Flows



Correspondence between Maximum Matchings and Max Flow

Theorem (Corollary 26.11)

The cardinality of a maximum matching M in a bipartite graph G equals the value of a maximum flow f in the corresponding flow network $\widetilde{G}$.

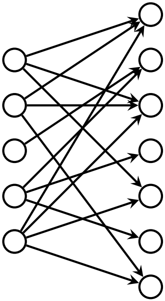


Graph G

Graph $\widetilde{G}$

From Matching to Flow

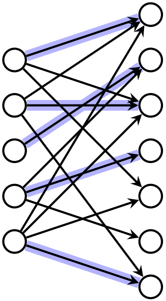
- Given a maximum matching of cardinality k



Graph G

From Matching to Flow

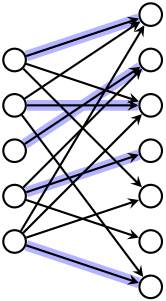
- Given a maximum matching of cardinality k



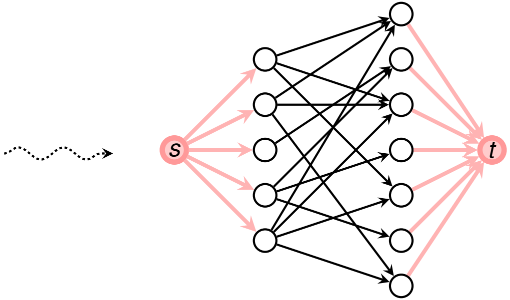
Graph G

From Matching to Flow

- Given a maximum matching of cardinality k
- Consider flow f that sends one unit along each each of k paths



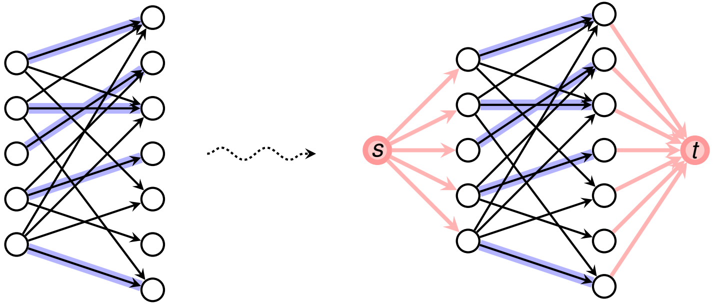
Graph G



Graph $\widetilde{G}$

From Matching to Flow

- Given a maximum matching of cardinality k
- Consider flow f that sends one unit along each each of k paths

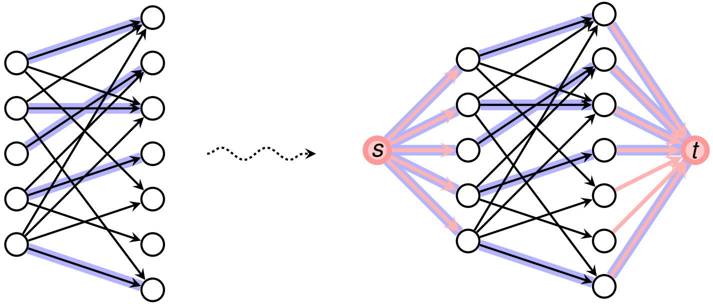


Graph G

Graph $\widetilde{G}$

From Matching to Flow

- Given a maximum matching of cardinality k
- Consider flow f that sends one unit along each each of k paths

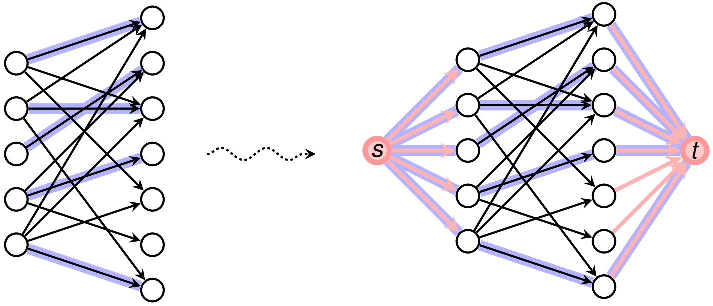


Graph G Graph $\widetilde{G}$

From Matching to Flow

- Given a maximum matching of cardinality k
- Consider flow f that sends one unit along each each of k paths

$\Rightarrow$ f is a flow and has value k



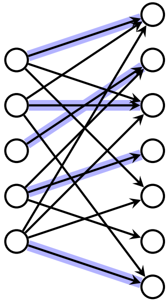
Graph G Graph $\widetilde{G}$

From Matching to Flow

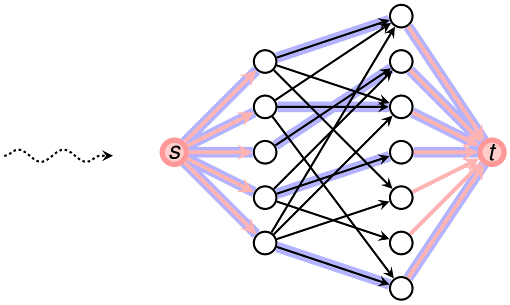
- Given a maximum matching of cardinality k
- Consider flow f that sends one unit along each each of k paths

$\Rightarrow$ f is a flow and has value k

max cardinality matching $\leq$ value of maxflow



Graph G



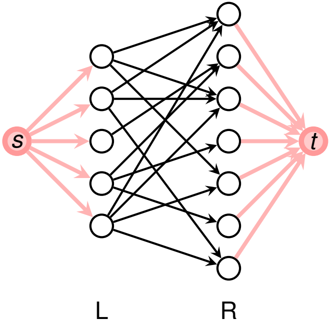
Graph $\widetilde{G}$

From Flow to Matching

- Let f be a maximum flow in $\widetilde{G}$ of value k

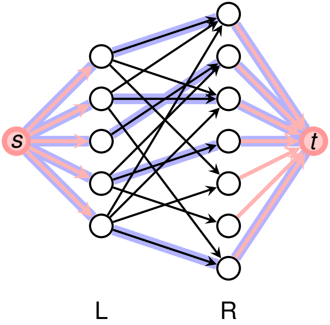
From Flow to Matching

- Let f be a maximum flow in $\widetilde{G}$ of value k



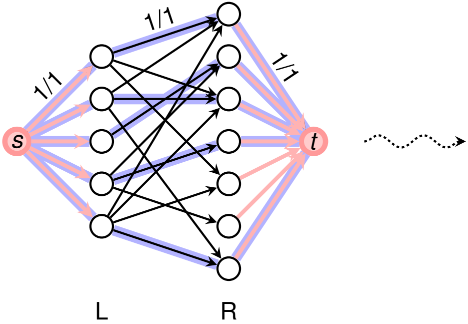
From Flow to Matching

- Let f be a maximum flow in $\widetilde{G}$ of value k



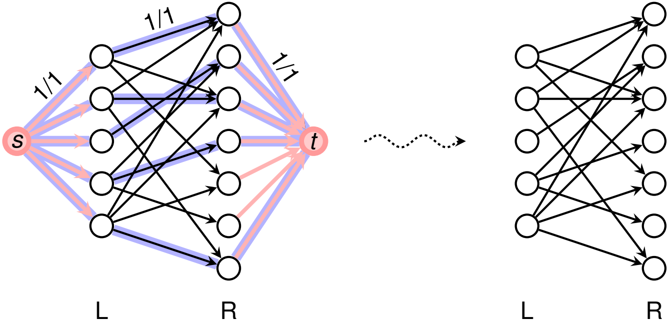
From Flow to Matching

- Let f be a maximum flow in $\widetilde{G}$ of value k
- Integrality Theorem $\Rightarrow f(u,v) \in \{0,1\}$ and k integral



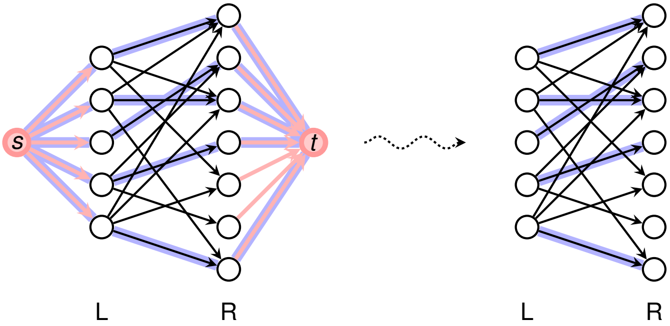
From Flow to Matching

- Let f be a maximum flow in $\widetilde{G}$ of value k
- Integrality Theorem $\Rightarrow f(u, v) \in \{0, 1\}$ and k integral
- Let M' be all edges from L to R which carry a flow of one



From Flow to Matching

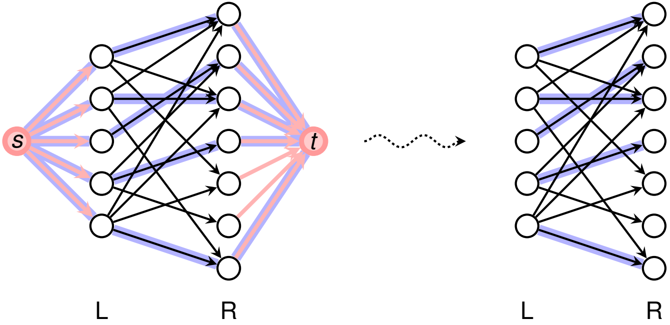
- Let f be a maximum flow in $\widetilde{G}$ of value k
- Integrality Theorem $\Rightarrow f(u, v) \in \{0, 1\}$ and k integral
- Let M' be all edges from L to R which carry a flow of one



From Flow to Matching

- Let f be a maximum flow in $\widetilde{G}$ of value k
- Integrality Theorem $\Rightarrow f(u,v) \in \{0,1\}$ and k integral
- Let M' be all edges from L to R which carry a flow of one

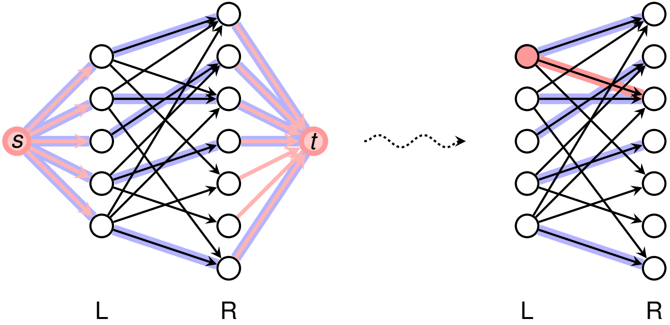
a) Flow Conservation



From Flow to Matching

- Let f be a maximum flow in $\widetilde{G}$ of value k
- Integrality Theorem $\Rightarrow f(u,v) \in \{0,1\}$ and k integral
- Let M' be all edges from L to R which carry a flow of one

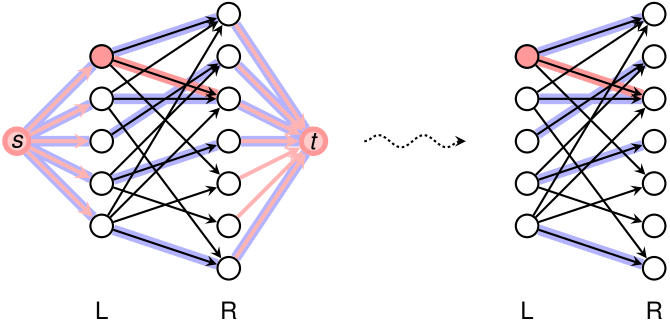
a) Flow Conservation $\Rightarrow$ every node in L sends at most one unit



From Flow to Matching

- Let f be a maximum flow in $\widetilde{G}$ of value k
- Integrality Theorem $\Rightarrow f(u,v) \in \{0,1\}$ and k integral
- Let M' be all edges from L to R which carry a flow of one

a) Flow Conservation $\Rightarrow$ every node in L sends at most one unit

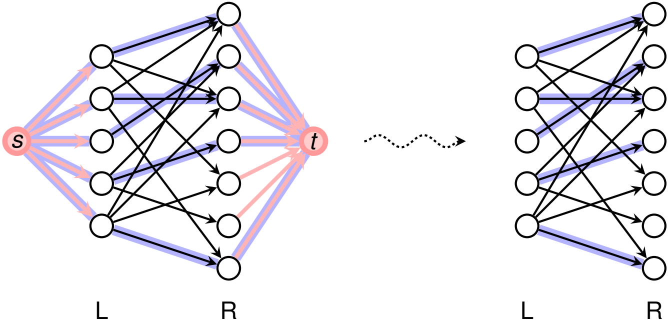


From Flow to Matching

- Let f be a maximum flow in $\widetilde{G}$ of value k
- Integrality Theorem $\Rightarrow f(u,v) \in \{0,1\}$ and k integral
- Let M' be all edges from L to R which carry a flow of one

a) Flow Conservation $\Rightarrow$ every node in L sends at most one unit

b) Flow Conservation $\Rightarrow$ every node in R receives at most one unit



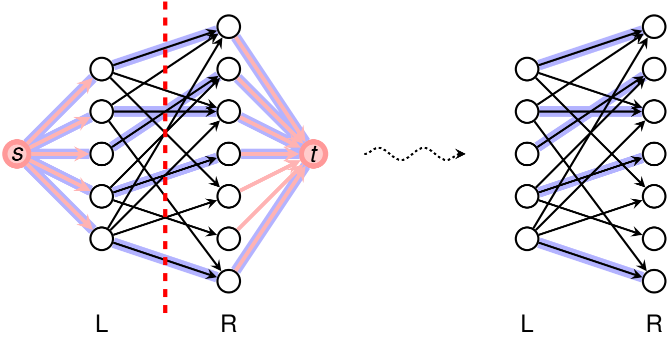
From Flow to Matching

- Let f be a maximum flow in $\widetilde{G}$ of value k
- Integrality Theorem $\Rightarrow f(u,v) \in \{0,1\}$ and k integral
- Let M' be all edges from L to R which carry a flow of one

a) Flow Conservation $\Rightarrow$ every node in L sends at most one unit

b) Flow Conservation $\Rightarrow$ every node in R receives at most one unit

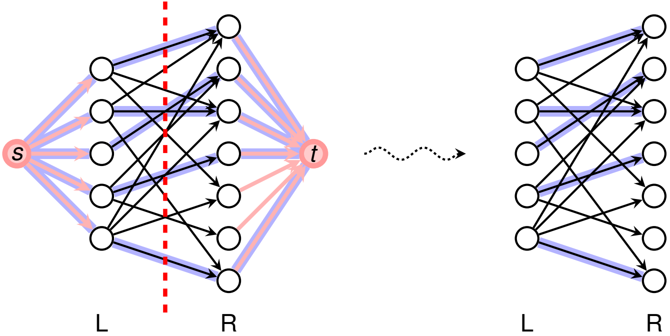
c) Cut $(L \cup \{s\}, R \cup \{t\})$



From Flow to Matching

- Let f be a maximum flow in $\widetilde{G}$ of value k
- Integrality Theorem $\Rightarrow f(u,v) \in \{0,1\}$ and k integral
- Let M' be all edges from L to R which carry a flow of one

a) Flow Conservation $\Rightarrow$ every node in L sends at most one unit
b) Flow Conservation $\Rightarrow$ every node in R receives at most one unit
c) Cut $(L \cup \{s\}, R \cup \{t\})$ $\Rightarrow$ net flow is k



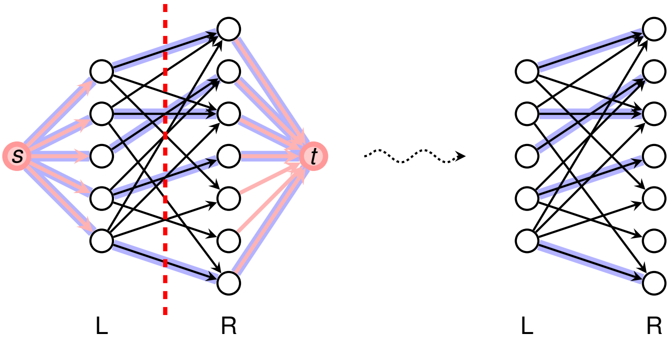
From Flow to Matching

- Let f be a maximum flow in $\widetilde{G}$ of value k
- Integrality Theorem $\Rightarrow f(u,v) \in \{0,1\}$ and k integral
- Let M' be all edges from L to R which carry a flow of one

a) Flow Conservation $\Rightarrow$ every node in L sends at most one unit

b) Flow Conservation $\Rightarrow$ every node in R receives at most one unit

c) Cut $(L \cup \{s\}, R \cup \{t\}) \Rightarrow$ net flow is $k \Rightarrow M'$ has k edges



From Flow to Matching

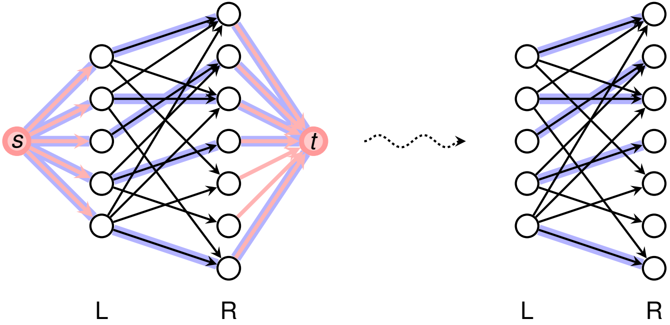
- Let f be a maximum flow in $\widetilde{G}$ of value k
- Integrality Theorem $\Rightarrow f(u,v) \in \{0,1\}$ and k integral
- Let M' be all edges from L to R which carry a flow of one

a) Flow Conservation $\Rightarrow$ every node in L sends at most one unit

b) Flow Conservation $\Rightarrow$ every node in R receives at most one unit

c) Cut $(L \cup \{s\}, R \cup \{t\})$ $\Rightarrow$ net flow is $k \Rightarrow M'$ has k edges

$\Rightarrow$ By a) & b), M' is a matching and by c), M' has cardinality k



From Flow to Matching

- Let f be a maximum flow in $\widetilde{G}$ of value k
- Integrality Theorem $\Rightarrow f(u,v) \in \{0,1\}$ and k integral
- Let M' be all edges from L to R which carry a flow of one

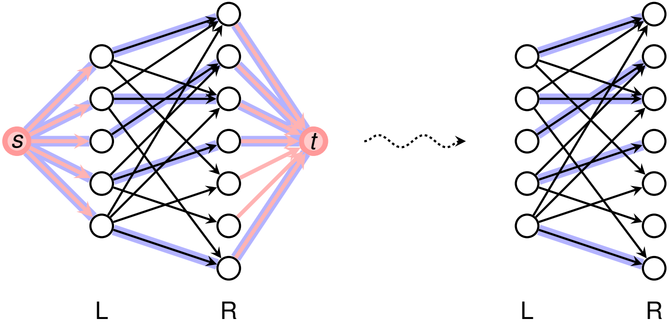
a) Flow Conservation $\Rightarrow$ every node in L sends at most one unit

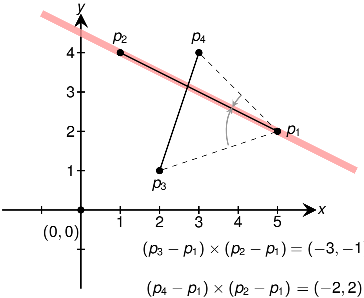
b) Flow Conservation $\Rightarrow$ every node in R receives at most one unit

c) Cut $(L \cup \{s\}, R \cup \{t\}) \Rightarrow$ net flow is $k \Rightarrow M'$ has k edges

$\Rightarrow$ By a) & b), M' is a matching and by c), M' has cardinality k

value of maxflow $\leq$ max cardinality matching





$$(p_3 - p_1) \times (p_2 - p_1) = (-3, -1) \times (-4, 2) = -10$$

$$(p_4 - p_1) \times (p_2 - p_1) = (-2, 2) \times (-4, 2) = 4$$

7: Geometric Algorithms

Frank Stajano Thomas Sauerwald

Lent 2015

UNIVERSITY OF CAMBRIDGE

Outline

Analysis of Ford-Fulkerson

Matchings in Bipartite Graphs

Introduction and Line Intersection

Introduction

Computational Geometry

- Branch that studies algorithms for geometric problems

Introduction

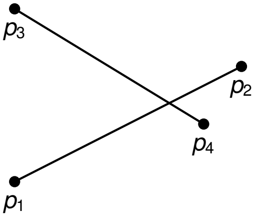
Computational Geometry

- Branch that studies algorithms for geometric problems
- typically, input is a set of points, line segments etc.

Introduction

Computational Geometry

- Branch that studies algorithms for geometric problems
- typically, input is a set of points, line segments etc.



Do these lines intersect?

Introduction

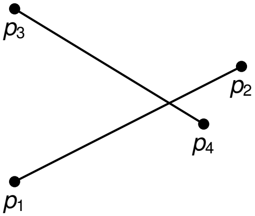
Computational Geometry

- Branch that studies algorithms for geometric problems
- typically, input is a set of points, line segments etc.

Applications

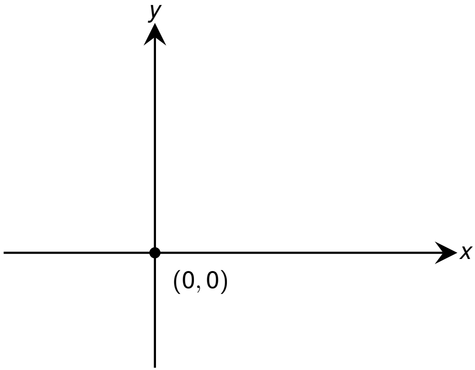
- computer graphics
- computer vision
- textile layout
- VLSI design

⋮

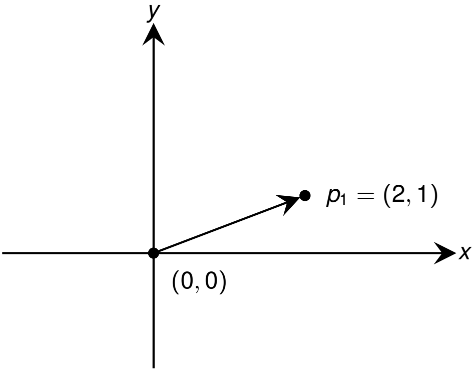


Do these lines intersect?

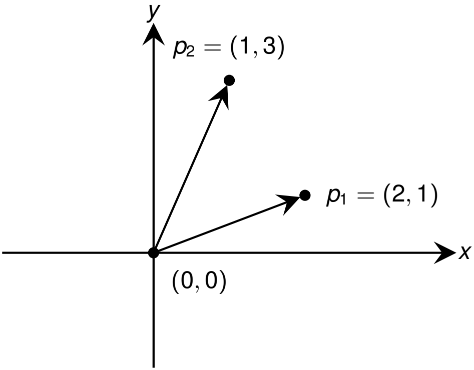
Cross Product (Area)



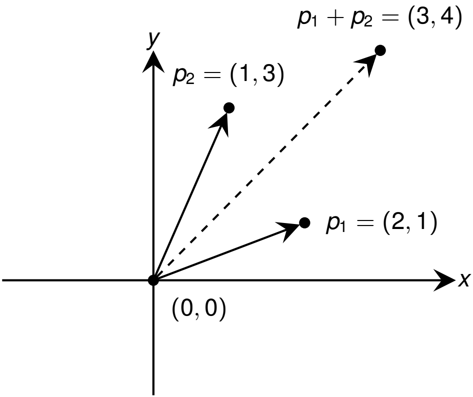
Cross Product (Area)



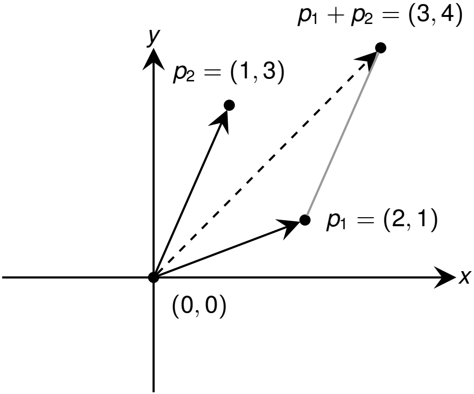
Cross Product (Area)



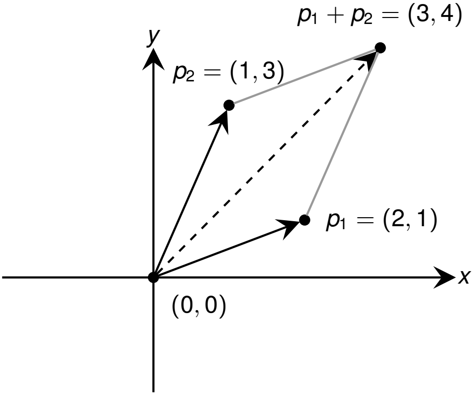
Cross Product (Area)



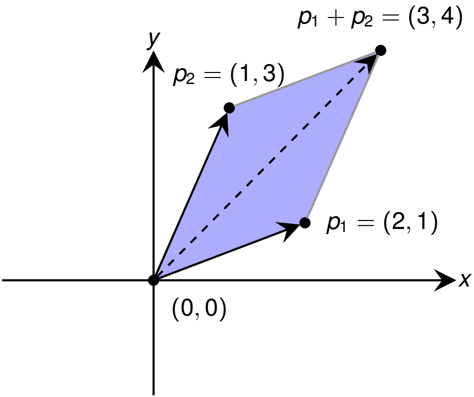
Cross Product (Area)



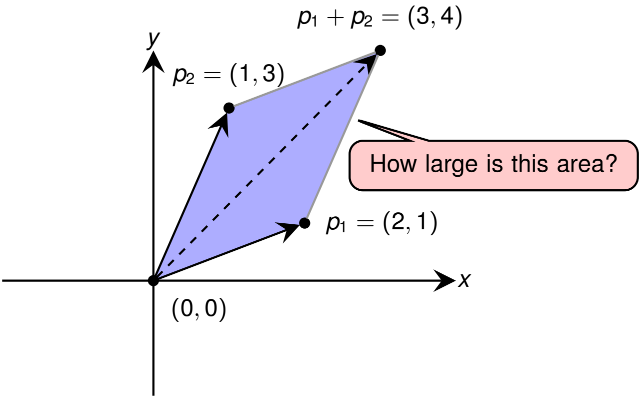
Cross Product (Area)



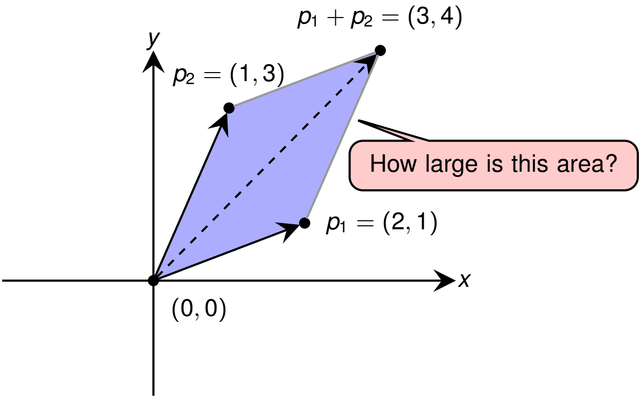
Cross Product (Area)



Cross Product (Area)

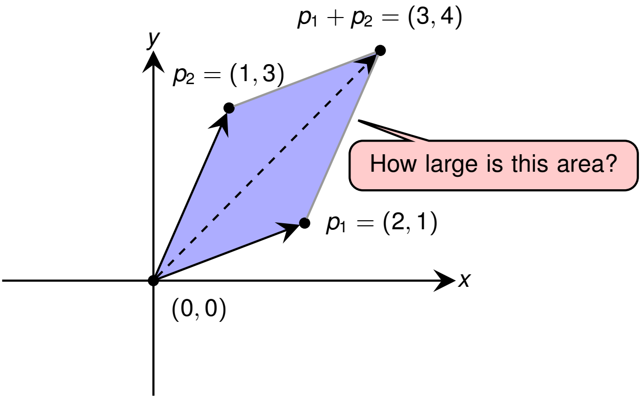


Cross Product (Area)



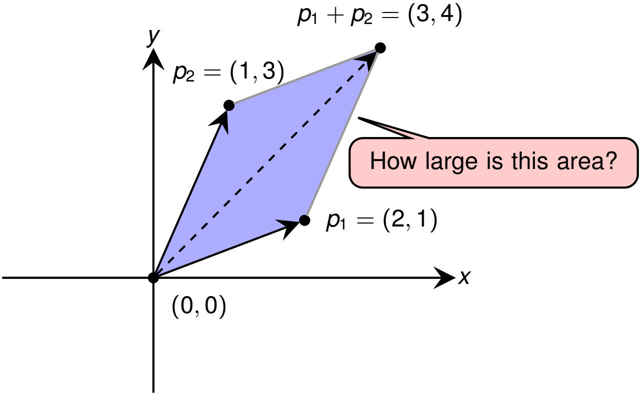
$p_1 \times p_2$

Cross Product (Area)



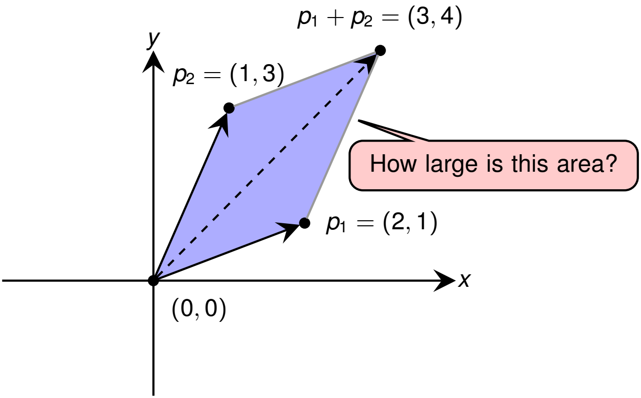
$$p_1 \times p_2 = \det \begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix}$$

Cross Product (Area)



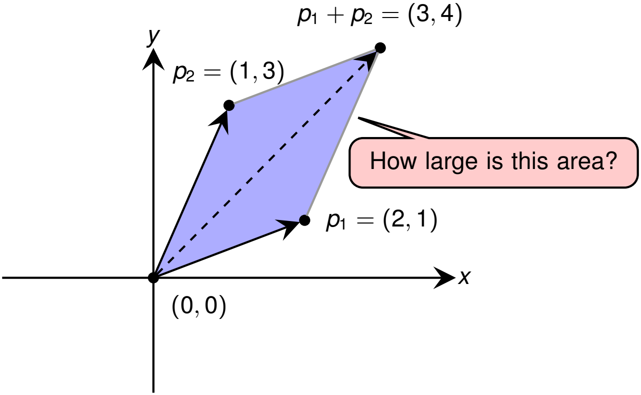
$$p_1 \times p_2 = \det \begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix} = x_1 y_2 - x_2 y_1$$

Cross Product (Area)



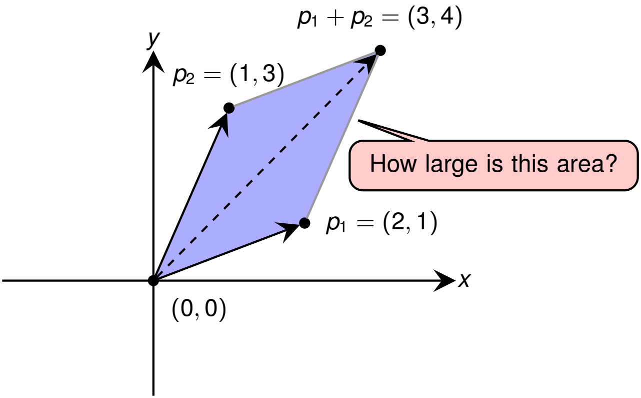
$$p_1 \times p_2 = \det \begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix} = x_1 y_2 - x_2 y_1 = 2 \cdot 3 - 1 \cdot 1$$

Cross Product (Area)



$$p_1 \times p_2 = \det \begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix} = x_1 y_2 - x_2 y_1 = 2 \cdot 3 - 1 \cdot 1 = 5$$

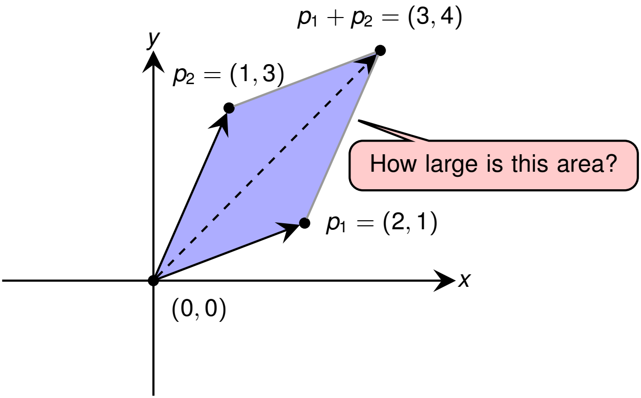
Cross Product (Area)



$$p_1 \times p_2 = \det \begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix} = x_1 y_2 - x_2 y_1 = 2 \cdot 3 - 1 \cdot 1 = 5$$

$p_2 \times p_1$

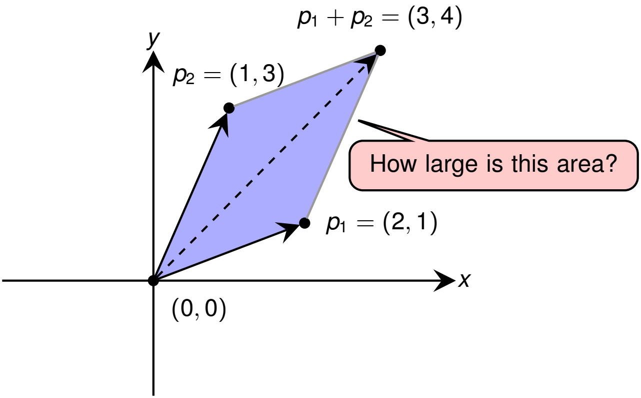
Cross Product (Area)



$$p_1 \times p_2 = \det \begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix} = x_1 y_2 - x_2 y_1 = 2 \cdot 3 - 1 \cdot 1 = 5$$

$$p_2 \times p_1 = y_1 x_2 - y_2 x_1$$

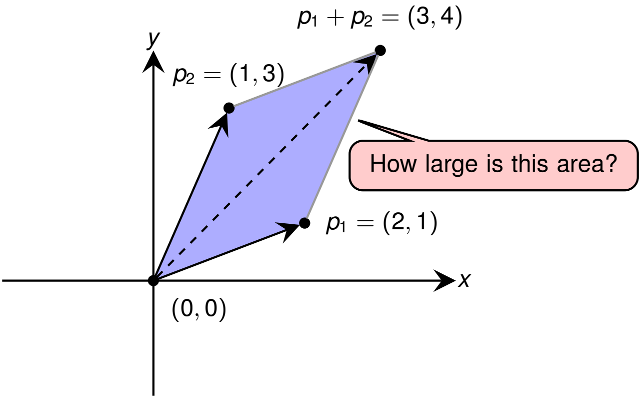
Cross Product (Area)



$$p_1 \times p_2 = \det \begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix} = x_1 y_2 - x_2 y_1 = 2 \cdot 3 - 1 \cdot 1 = 5$$

$$p_2 \times p_1 = y_1 x_2 - y_2 x_1 = -(p_1 \times p_2)$$

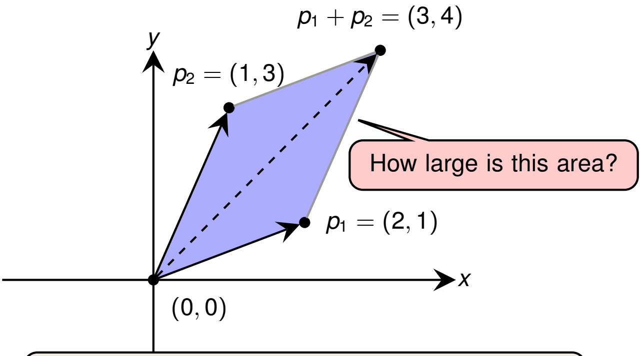
Cross Product (Area)



$$p_1 \times p_2 = \det \begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix} = x_1 y_2 - x_2 y_1 = 2 \cdot 3 - 1 \cdot 1 = 5$$

$$p_2 \times p_1 = y_1 x_2 - y_2 x_1 = -(p_1 \times p_2) = -5$$

Cross Product (Area)

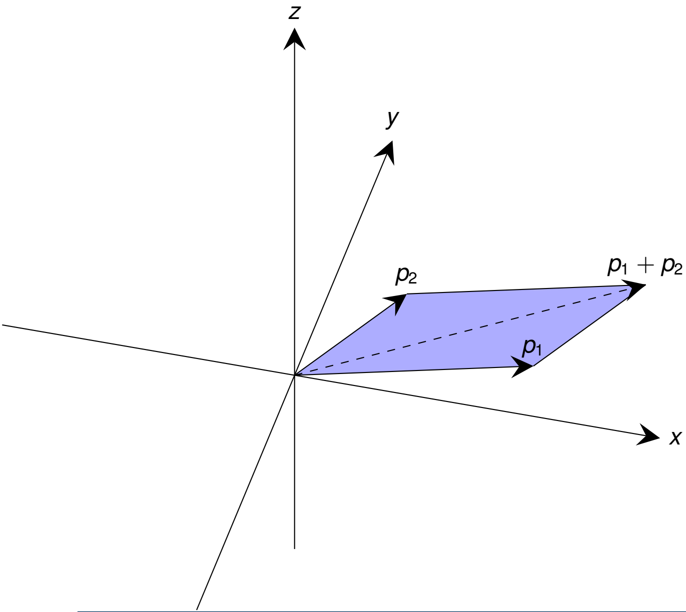


Alternatively, one could take the dot-product (but not used here): $p_1 \cdot p_2 = \|p_1\| \cdot \|p_2\| \cdot \cos(\phi)$.

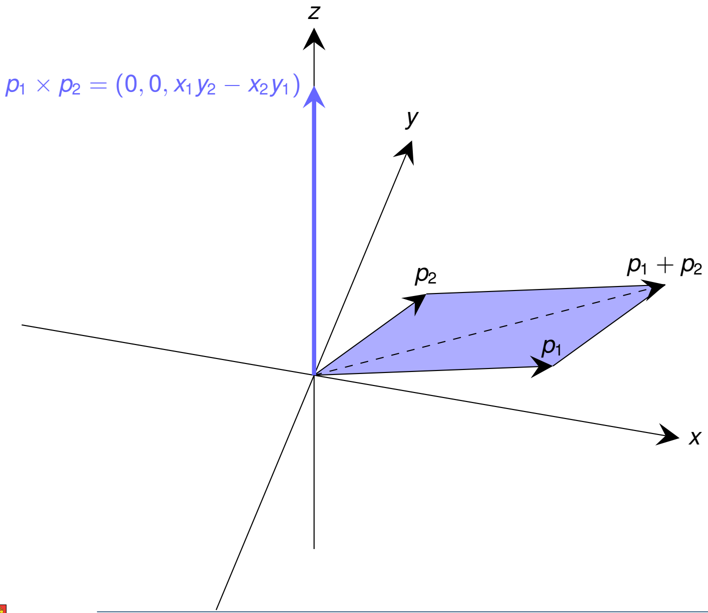
$$p_1 \times p_2 = \det \begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix} = x_1 y_2 - x_2 y_1 = 2 \cdot 3 - 1 \cdot 1 = 5$$

$$p_2 \times p_1 = y_1 x_2 - y_2 x_1 = -(p_1 \times p_2) = -5$$

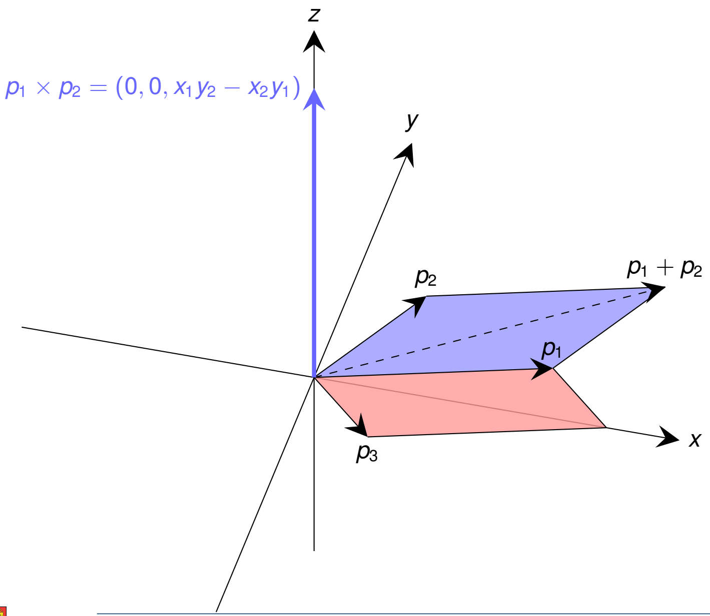
Cross Product in 3D



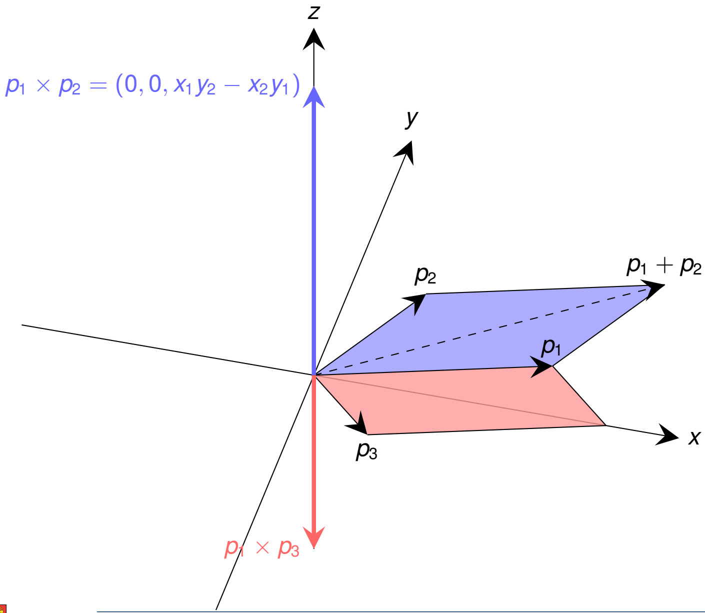
Cross Product in 3D



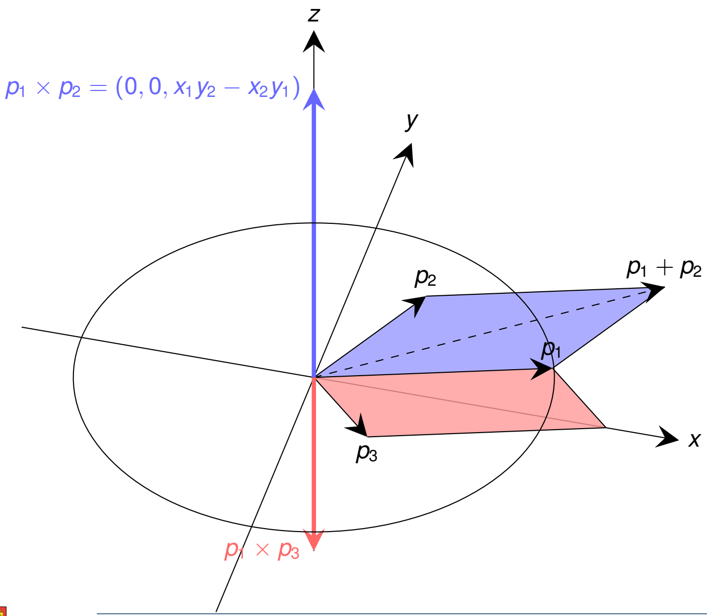
Cross Product in 3D



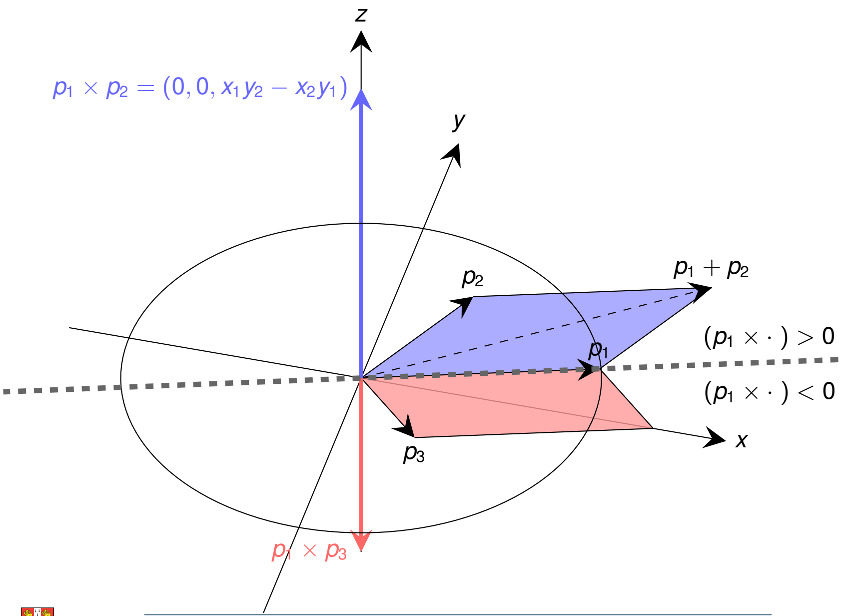
Cross Product in 3D



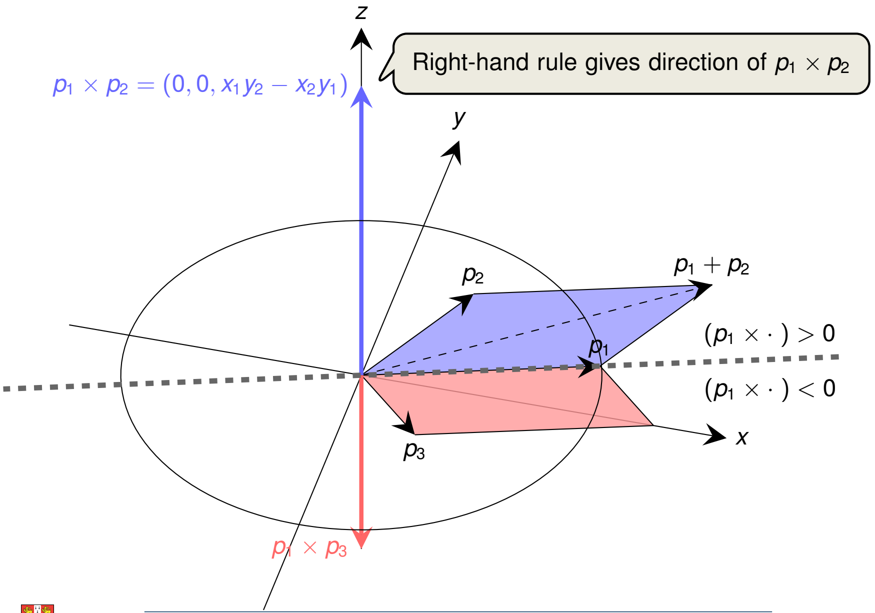
Cross Product in 3D



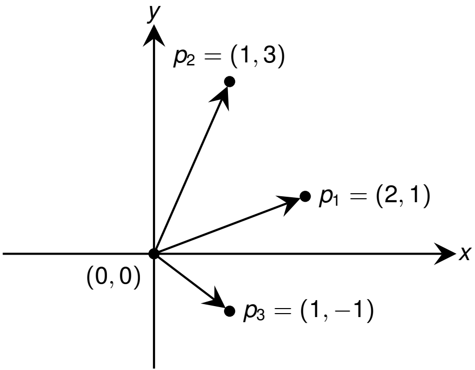
Cross Product in 3D



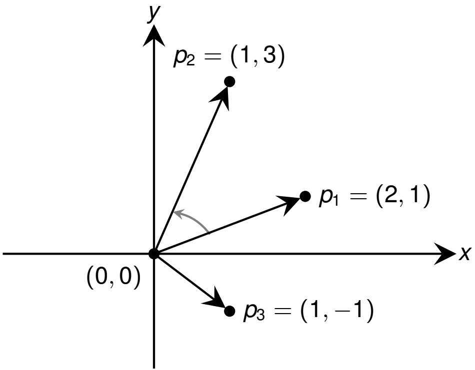
Cross Product in 3D



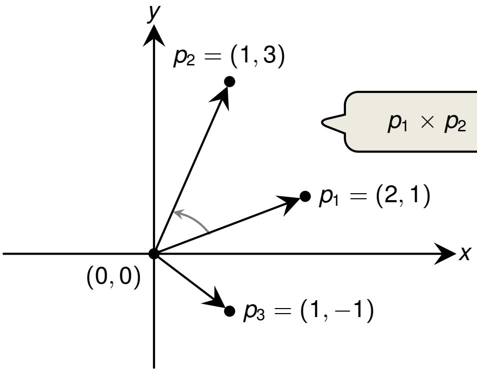
Using Cross product to determine Turns (1/2)



Using Cross product to determine Turns (1/2)

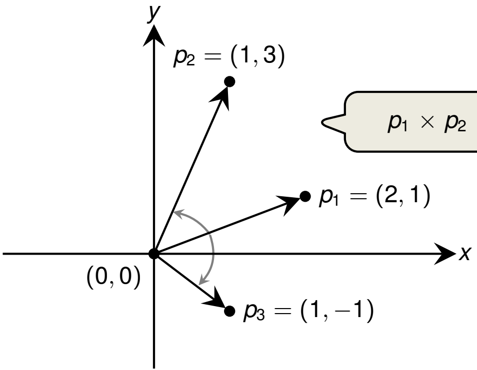


Using Cross product to determine Turns (1/2)



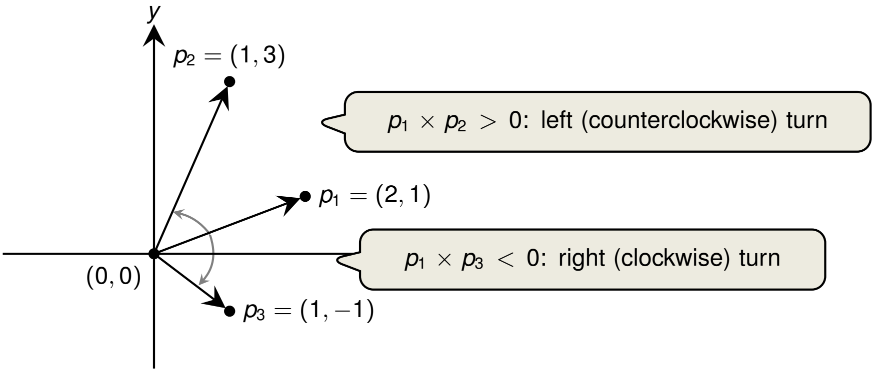
$p_1 \times p_2 > 0$: left (counterclockwise) turn

Using Cross product to determine Turns (1/2)

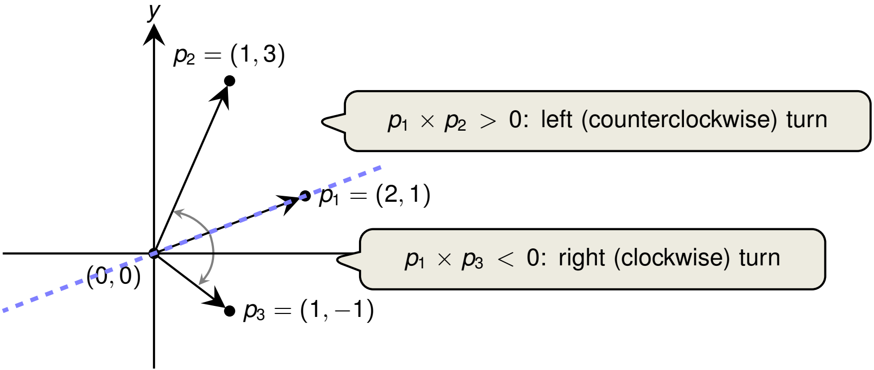


$p_1 \times p_2 > 0$: left (counterclockwise) turn

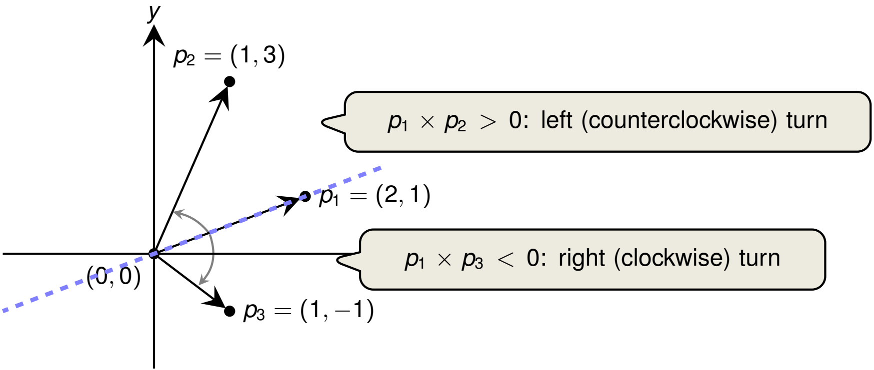
Using Cross product to determine Turns (1/2)



Using Cross product to determine Turns (1/2)

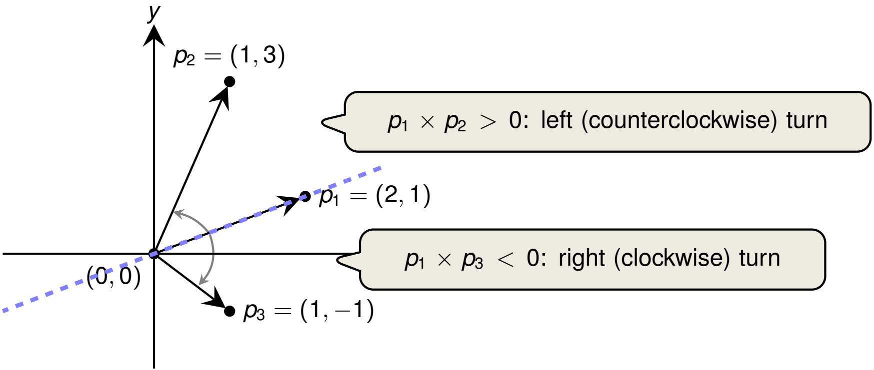


Using Cross product to determine Turns (1/2)



Sign of cross product determines turn!

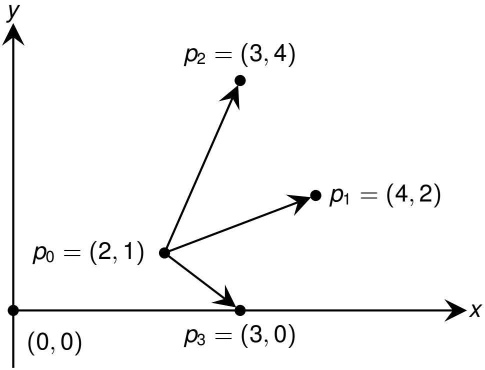
Using Cross product to determine Turns (1/2)



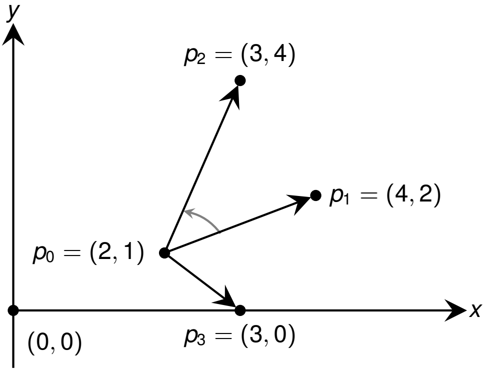
Sign of cross product determines turn!

Cross product equals zero iff vectors are colinear

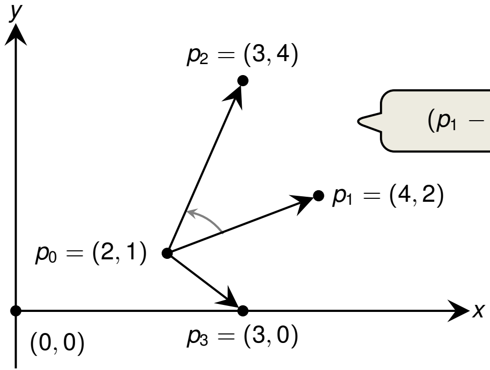
Using Cross product to determine Turns (2/2)



Using Cross product to determine Turns (2/2)

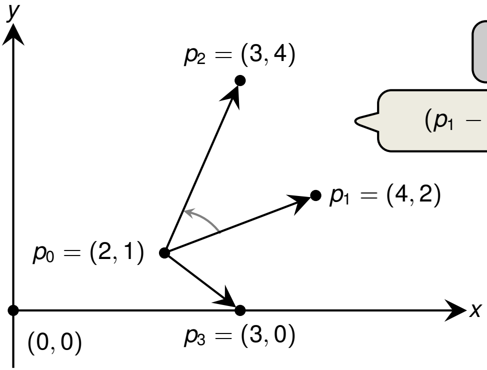


Using Cross product to determine Turns (2/2)



$(p_1 - p_0) \times (p_2 - p_0) > 0$: left turn

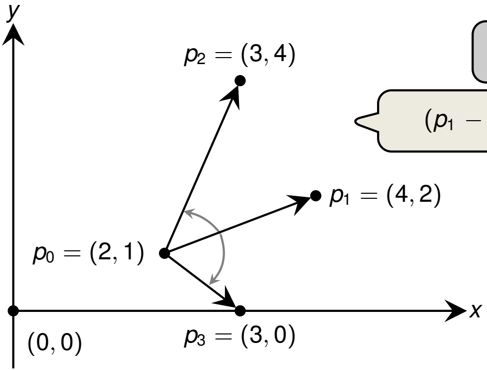
Using Cross product to determine Turns (2/2)



$(2,1) \times (1,3) = 5$

$(p_1 - p_0) \times (p_2 - p_0) > 0$: left turn

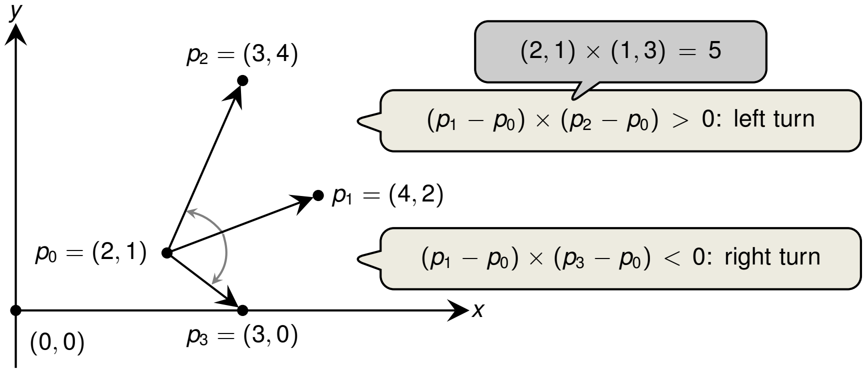
Using Cross product to determine Turns (2/2)



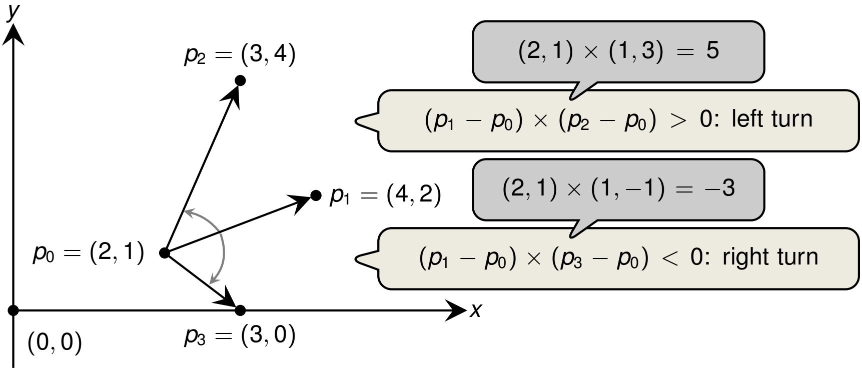
$(2,1) \times (1,3) = 5$

$(p_1 - p_0) \times (p_2 - p_0) > 0$: left turn

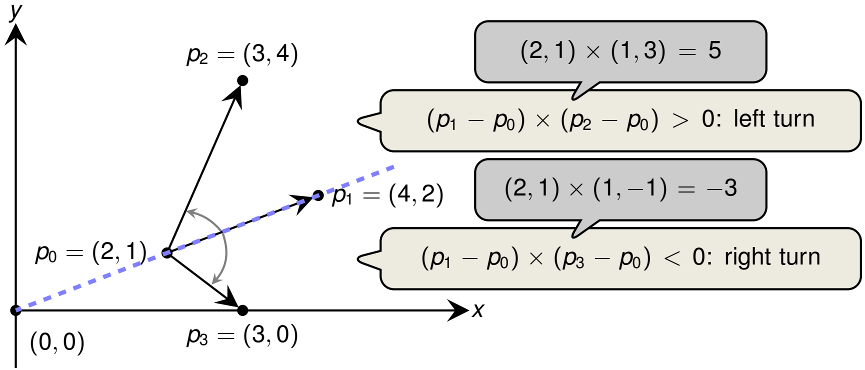
Using Cross product to determine Turns (2/2)



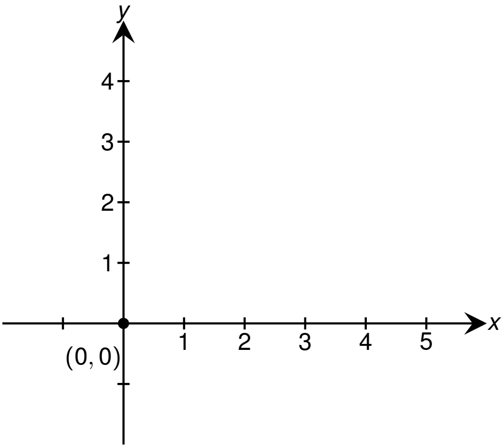
Using Cross product to determine Turns (2/2)



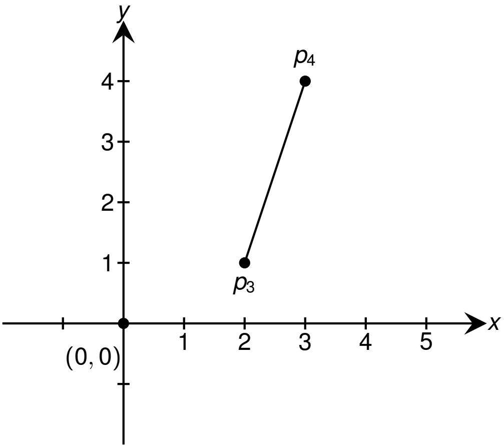
Using Cross product to determine Turns (2/2)



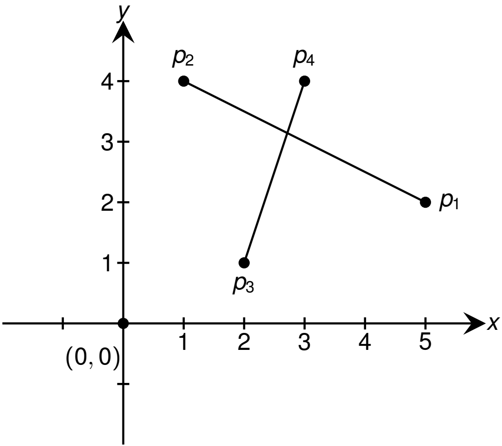
Solving Line Intersection (without Trigonometry and Division!)



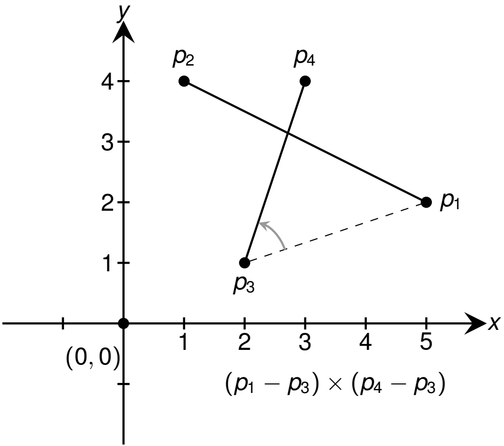
Solving Line Intersection (without Trigonometry and Division!)



Solving Line Intersection (without Trigonometry and Division!)

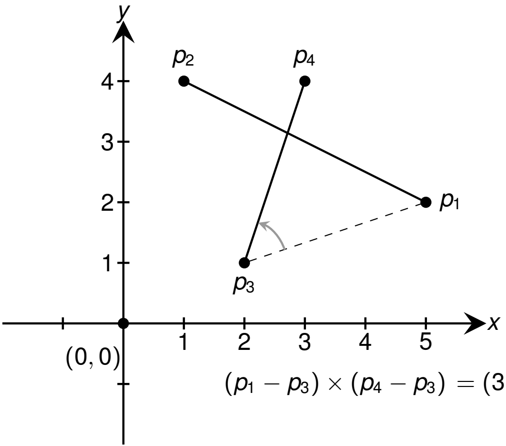


Solving Line Intersection (without Trigonometry and Division!)



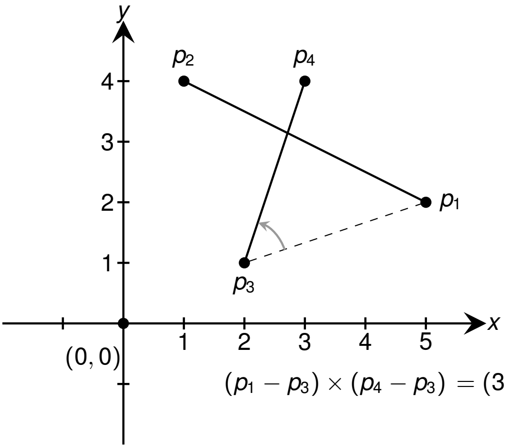
$$(p_1 - p_3) \times (p_4 - p_3)$$

Solving Line Intersection (without Trigonometry and Division!)



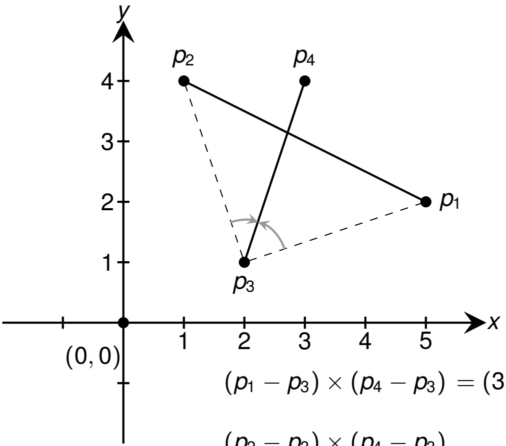
$$(p_1 - p_3) \times (p_4 - p_3) = (3, 1) \times (1, 3)$$

Solving Line Intersection (without Trigonometry and Division!)



$$(p_1 - p_3) \times (p_4 - p_3) = (3, 1) \times (1, 3) = 8$$

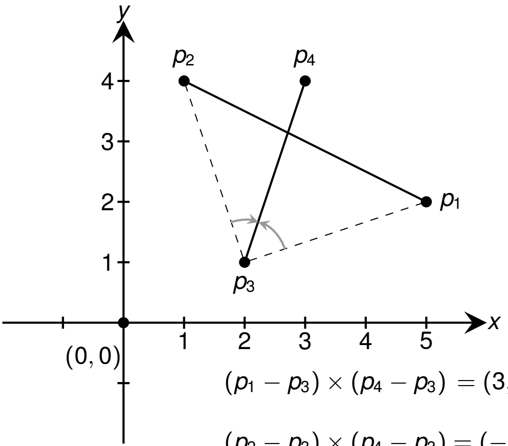
Solving Line Intersection (without Trigonometry and Division!)



$$(p_1 - p_3) \times (p_4 - p_3) = (3, 1) \times (1, 3) = 8$$

$$(p_2 - p_3) \times (p_4 - p_3)$$

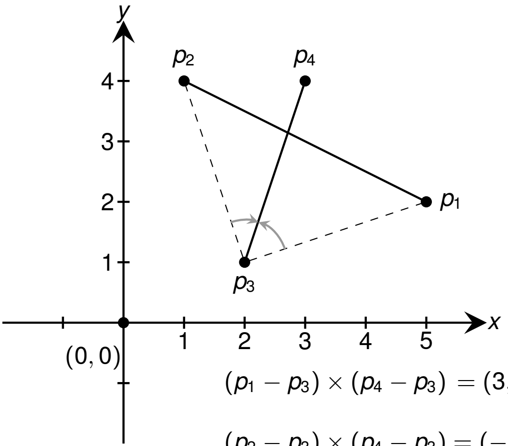
Solving Line Intersection (without Trigonometry and Division!)



$$(p_1 - p_3) \times (p_4 - p_3) = (3, 1) \times (1, 3) = 8$$

$$(p_2 - p_3) \times (p_4 - p_3) = (-1, 3) \times (1, 3)$$

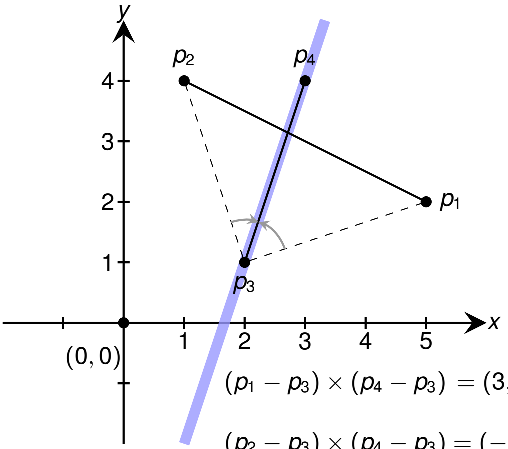
Solving Line Intersection (without Trigonometry and Division!)



$$(p_1 - p_3) \times (p_4 - p_3) = (3, 1) \times (1, 3) = 8$$

$$(p_2 - p_3) \times (p_4 - p_3) = (-1, 3) \times (1, 3) = -6$$

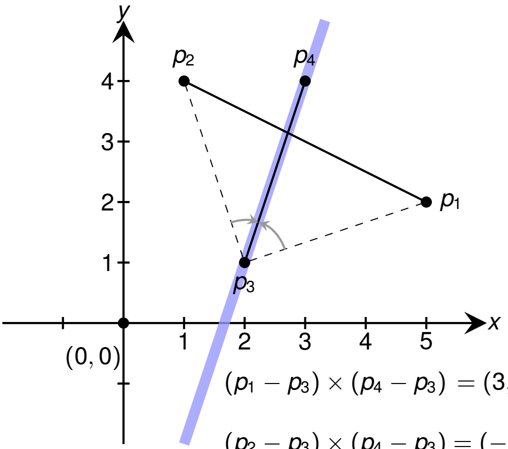
Solving Line Intersection (without Trigonometry and Division!)



$$(p_1 - p_3) \times (p_4 - p_3) = (3, 1) \times (1, 3) = 8$$

$$(p_2 - p_3) \times (p_4 - p_3) = (-1, 3) \times (1, 3) = -6$$

Solving Line Intersection (without Trigonometry and Division!)

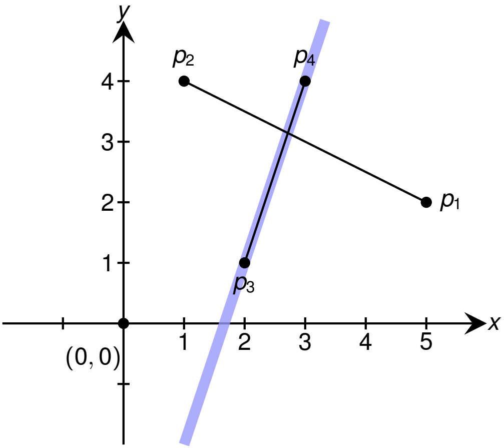


$$(p_1 - p_3) \times (p_4 - p_3) = (3, 1) \times (1, 3) = 8$$

$$(p_2 - p_3) \times (p_4 - p_3) = (-1, 3) \times (1, 3) = -6$$

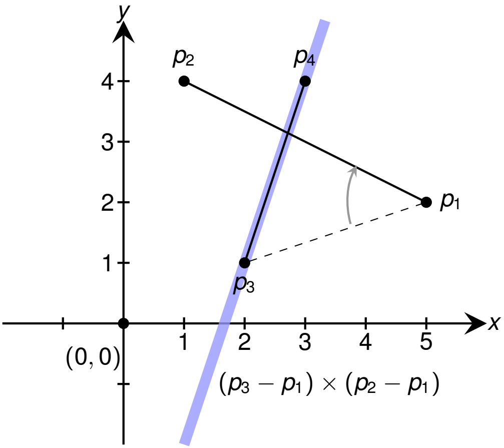
Opposite signs $\Rightarrow \overline{p_1 p_2}$ crosses (infinite) line through p_3 and p_4

Solving Line Intersection (without Trigonometry and Division!)



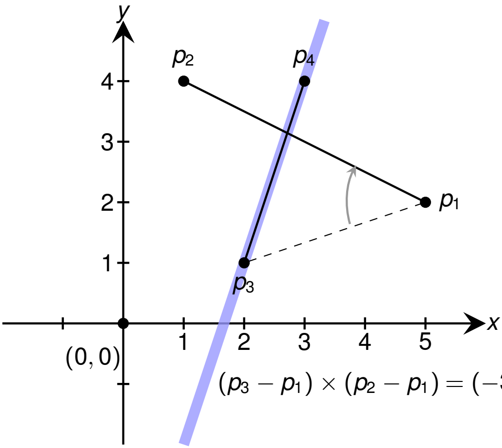
Opposite signs $\Rightarrow \overline{p_1 p_2}$ crosses (infinite) line through p_3 and p_4

Solving Line Intersection (without Trigonometry and Division!)



Opposite signs $\Rightarrow \overline{p_1 p_2}$ crosses (infinite) line through p_3 and p_4

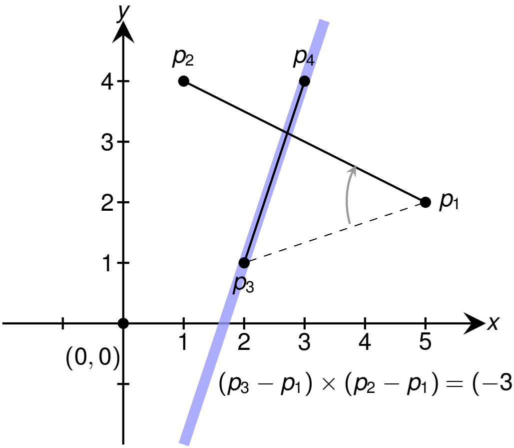
Solving Line Intersection (without Trigonometry and Division!)



$(p_3 - p_1) \times (p_2 - p_1) = (-3, -1) \times (-4, 2)$

Opposite signs $\Rightarrow \overline{p_1 p_2}$ crosses (infinite) line through p_3 and p_4

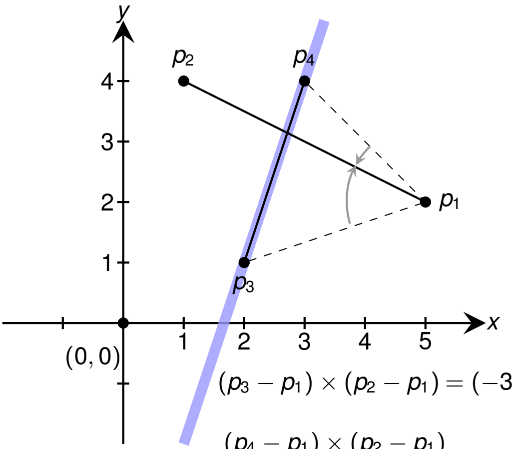
Solving Line Intersection (without Trigonometry and Division!)



$$(p_3 - p_1) \times (p_2 - p_1) = (-3, -1) \times (-4, 2) = -10$$

Opposite signs $\Rightarrow$ $\overline{p_1 p_2}$ crosses (infinite) line through p_3 and p_4

Solving Line Intersection (without Trigonometry and Division!)

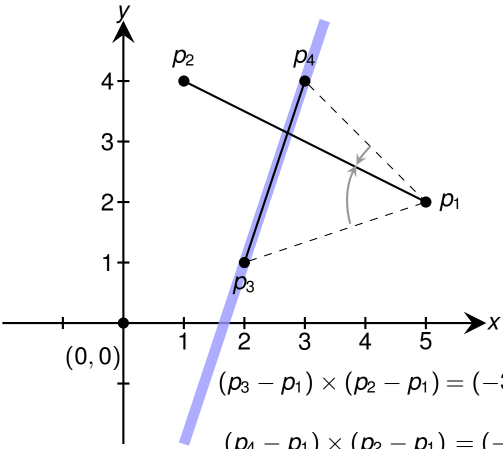


$$(p_3 - p_1) \times (p_2 - p_1) = (-3, -1) \times (-4, 2) = -10$$

$$(p_4 - p_1) \times (p_2 - p_1)$$

Opposite signs $\Rightarrow \overline{p_1 p_2}$ crosses (infinite) line through p_3 and p_4

Solving Line Intersection (without Trigonometry and Division!)

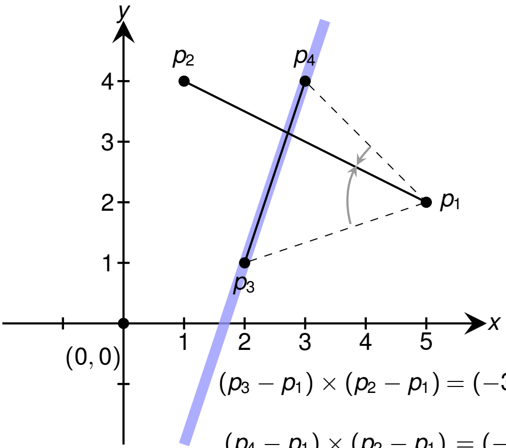


$$(p_3 - p_1) \times (p_2 - p_1) = (-3, -1) \times (-4, 2) = -10$$

$$(p_4 - p_1) \times (p_2 - p_1) = (-2, 2) \times (-4, 2)$$

Opposite signs $\Rightarrow \overline{p_1 p_2}$ crosses (infinite) line through p_3 and p_4

Solving Line Intersection (without Trigonometry and Division!)

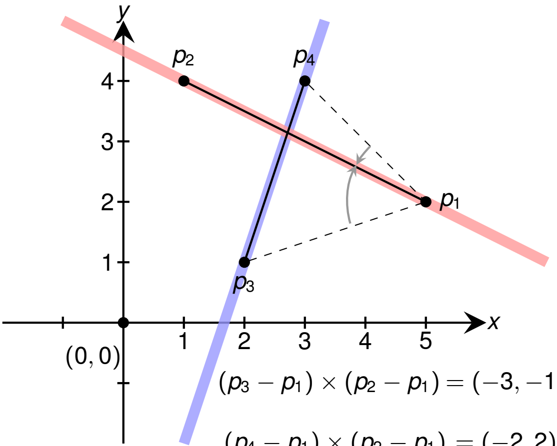


$$(p_3 - p_1) \times (p_2 - p_1) = (-3, -1) \times (-4, 2) = -10$$

$$(p_4 - p_1) \times (p_2 - p_1) = (-2, 2) \times (-4, 2) = 4$$

Opposite signs $\Rightarrow \overline{p_1 p_2}$ crosses (infinite) line through p_3 and p_4

Solving Line Intersection (without Trigonometry and Division!)



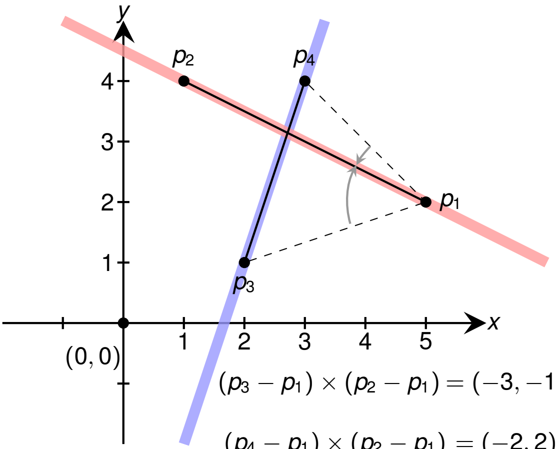
$$(p_3 - p_1) \times (p_2 - p_1) = (-3, -1) \times (-4, 2) = -10$$

$$(p_4 - p_1) \times (p_2 - p_1) = (-2, 2) \times (-4, 2) = 4$$

Opposite signs $\Rightarrow \overline{p_1 p_2}$ crosses (infinite) line through p_3 and p_4

Opposite signs $\Rightarrow \overline{p_3 p_4}$ crosses (infinite) line through p_1 and p_2

Solving Line Intersection (without Trigonometry and Division!)



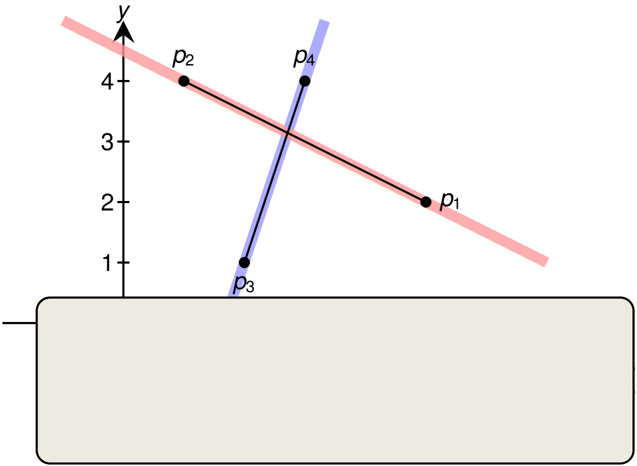
$$(p_3 - p_1) \times (p_2 - p_1) = (-3, -1) \times (-4, 2) = -10$$

$$(p_4 - p_1) \times (p_2 - p_1) = (-2, 2) \times (-4, 2) = 4$$

Opposite signs $\Rightarrow \overline{p_1 p_2}$ crosses (infinite) line through p_3 and p_4

Opposite signs $\Rightarrow \overline{p_3 p_4}$ crosses (infinite) line through p_1 and p_2

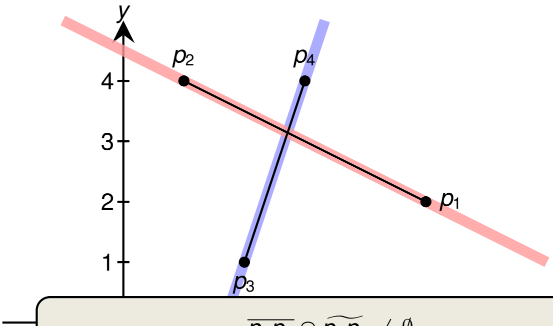
Solving Line Intersection (without Trigonometry and Division!)



Opposite signs ⇒ $\overline{p_1 p_2}$ crosses (infinite) line through p_3 and p_4

Opposite signs ⇒ $\overline{p_3 p_4}$ crosses (infinite) line through p_1 and p_2

Solving Line Intersection (without Trigonometry and Division!)

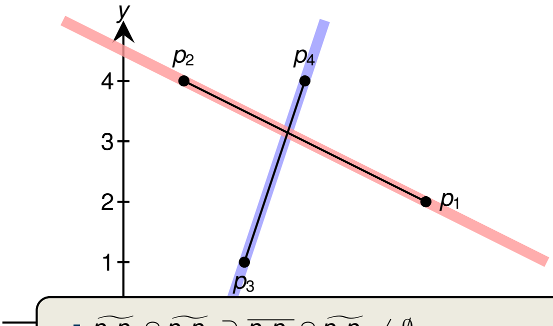


$$\overline{p_1 p_2} \cap \widetilde{p_3 p_4} \neq \emptyset$$
$$\widetilde{p_1 p_2} \cap \overline{p_3 p_4} \neq \emptyset$$

Opposite signs $\Rightarrow$ $\overline{p_1 p_2}$ crosses (infinite) line through p_3 and p_4

Opposite signs $\Rightarrow$ $\overline{p_3 p_4}$ crosses (infinite) line through p_1 and p_2

Solving Line Intersection (without Trigonometry and Division!)

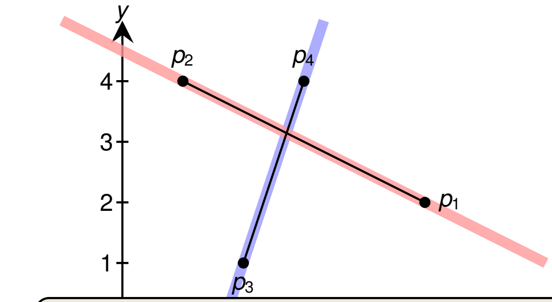


- $\widetilde{p_1 p_2} \cap \widetilde{p_3 p_4} \supseteq \overline{p_1 p_2} \cap \widetilde{p_3 p_4} \neq \emptyset$
- $\widetilde{p_1 p_2} \cap \widetilde{p_3 p_4} \supseteq \widetilde{p_1 p_2} \cap \overline{p_3 p_4} \neq \emptyset$

Opposite signs $\Rightarrow$ $\overline{p_1 p_2}$ crosses (infinite) line through p_3 and p_4

Opposite signs $\Rightarrow$ $\overline{p_3 p_4}$ crosses (infinite) line through p_1 and p_2

Solving Line Intersection (without Trigonometry and Division!)

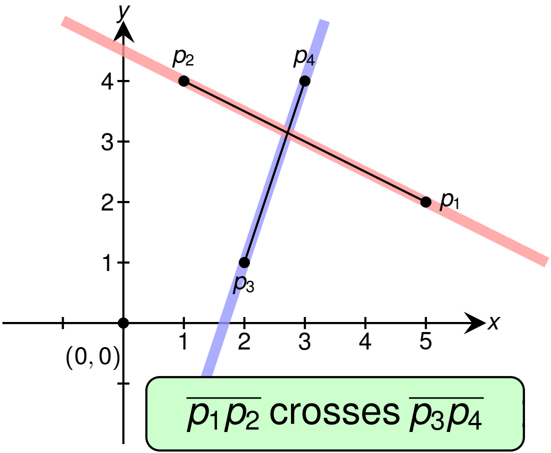


- $\widetilde{p_1p_2} \cap \widetilde{p_3p_4} \supseteq \overline{p_1p_2} \cap \widetilde{p_3p_4} \neq \emptyset$
- $\widetilde{p_1p_2} \cap \widetilde{p_3p_4} \supseteq \widetilde{p_1p_2} \cap \overline{p_3p_4} \neq \emptyset$
- Since $\widetilde{p_1p_2} \cap \widetilde{p_3p_4}$ consists of (at most) one point $\Rightarrow \overline{p_1p_2} \cap \overline{p_3p_4} \neq \emptyset$

Opposite signs $\Rightarrow \overline{p_1p_2}$ crosses (infinite) line through p_3 and p_4

Opposite signs $\Rightarrow \overline{p_3p_4}$ crosses (infinite) line through p_1 and p_2

Solving Line Intersection (without Trigonometry and Division!)

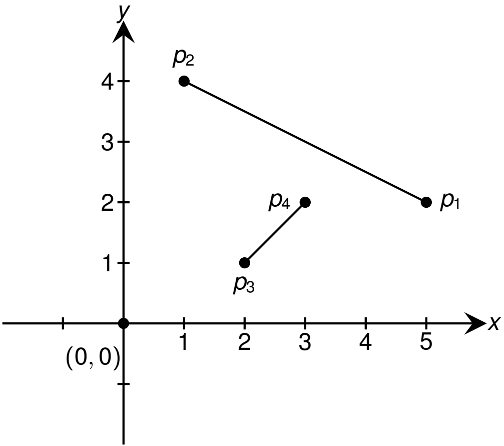


$\overline{p_1 p_2}$ crosses $\overline{p_3 p_4}$

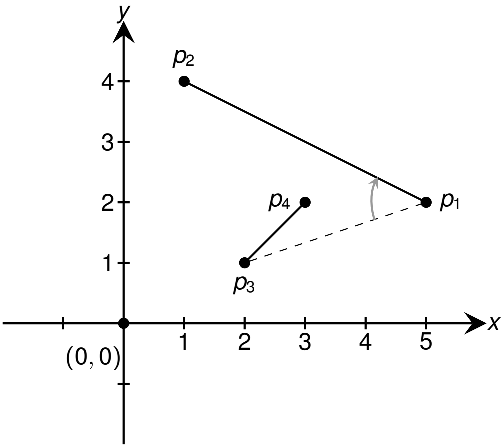
Opposite signs ⇒ $\overline{p_1 p_2}$ crosses (infinite) line through p_3 and p_4

Opposite signs ⇒ $\overline{p_3 p_4}$ crosses (infinite) line through p_1 and p_2

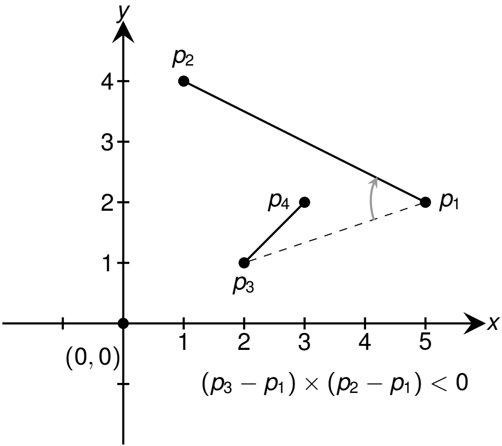
Solving Line Intersection (without Trigonometry and Division!)



Solving Line Intersection (without Trigonometry and Division!)

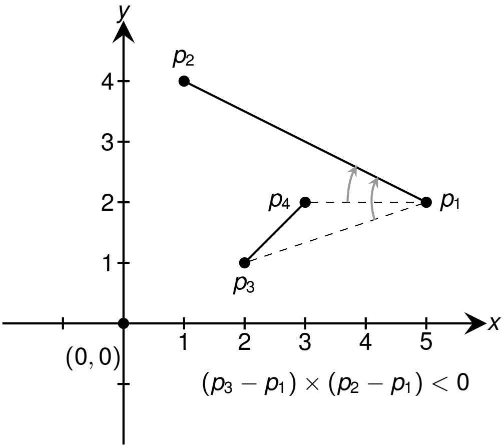


Solving Line Intersection (without Trigonometry and Division!)



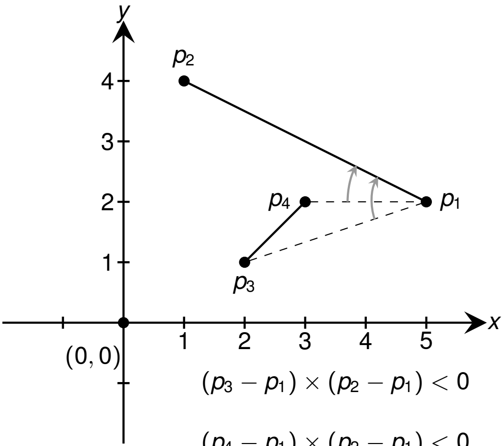
$$(p_3 - p_1) \times (p_2 - p_1) < 0$$

Solving Line Intersection (without Trigonometry and Division!)



$$(p_3 - p_1) \times (p_2 - p_1) < 0$$

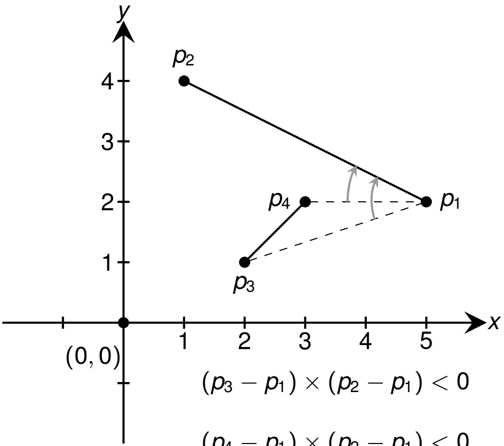
Solving Line Intersection (without Trigonometry and Division!)



$$(p_3 - p_1) \times (p_2 - p_1) < 0$$

$$(p_4 - p_1) \times (p_2 - p_1) < 0$$

Solving Line Intersection (without Trigonometry and Division!)

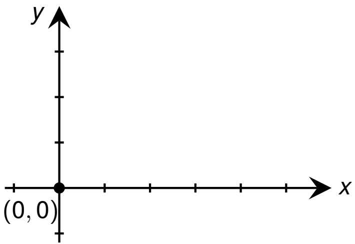


$$(p_3 - p_1) \times (p_2 - p_1) < 0$$

$$(p_4 - p_1) \times (p_2 - p_1) < 0$$

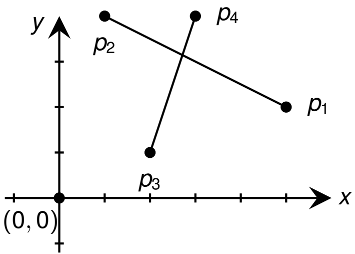
$\overline{p_1 p_2}$ does **not** cross $\overline{p_3 p_4}$

Solving Line Intersection



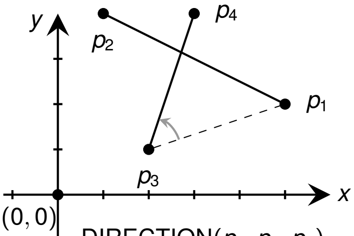

```
DIRECTION(p_i, p_j, p_k)
    return (p_k − p_i) × (p_j − p_i)
```

Solving Line Intersection



0: DIRECTION(p_i, p_j, p_k)
1: return $(p_k - p_i) \times (p_j - p_i)$

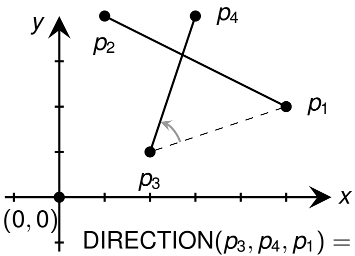
Solving Line Intersection



$$\text{DIRECTION}(p_3, p_4, p_1) = (p_1 - p_3) \times (p_4 - p_3)$$

DIRECTION(p_i, p_j, p_k)
  return $(p_k - p_i) \times (p_j - p_i)$

Solving Line Intersection

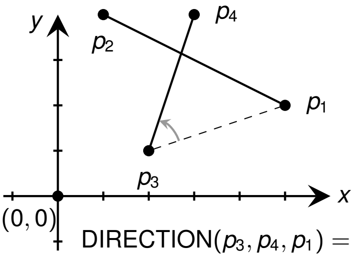


$$\text{DIRECTION}(p_3, p_4, p_1) = (p_1 - p_3) \times (p_4 - p_3)$$

```
DIRECTION(p_i, p_j, p_k)
    return (p_k − p_i) × (p_j − p_i)
```

```
SEGMENTS-INTERSECT(p_i, p_j, p_k)
    d_1 = DIRECTION(p_3, p_4, p_1)
    d_2 = DIRECTION(p_3, p_4, p_2)
    d_3 = DIRECTION(p_1, p_2, p_3)
    d_4 = DIRECTION(p_1, p_2, p_4)
    If d_1 · d_2 < 0 and d_3 · d_4 < 0 return TRUE
        · · · (handle all degenerate cases)
```

Solving Line Intersection

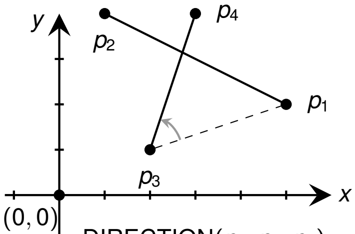


$$\text{DIRECTION}(p_3, p_4, p_1) = (p_1 - p_3) \times (p_4 - p_3)$$

DIRECTION(p_i, p_j, p_k)
    return $(p_k - p_i) \times (p_j - p_i)$

SEGMENTS-INTERSECT(p_i, p_j, p_k)
    $d_1 =$ DIRECTION(p_3, p_4, p_1)
    $d_2 =$ DIRECTION(p_3, p_4, p_2)
    $d_3 =$ DIRECTION(p_1, p_2, p_3)
    $d_4 =$ DIRECTION(p_1, p_2, p_4)
    If $d_1 \cdot d_2 < 0$ and $d_3 \cdot d_4 < 0$ return TRUE
        $\cdots$ (handle all degenerate cases)

Solving Line Intersection



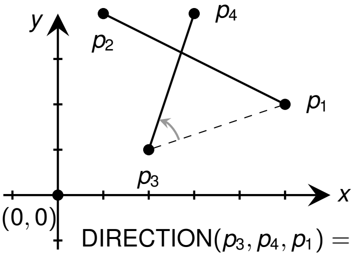
$$\text{DIRECTION}(p_3, p_4, p_1) = (p_1 - p_3) \times (p_4 - p_3)$$

```
DIRECTION(p_i, p_j, p_k)
    return (p_k − p_i) × (p_j − p_i)
```

```
SEGMENTS-INTERSECT(p_i, p_j, p_k)
    d_1 = DIRECTION(p_3, p_4, p_1)
    d_2 = DIRECTION(p_3, p_4, p_2)
    d_3 = DIRECTION(p_1, p_2, p_3)
    d_4 = DIRECTION(p_1, p_2, p_4)
    If d_1 · d_2 < 0 and d_3 · d_4 < 0 return TRUE
        ··· (handle all degenerate cases)
```

In total 4 satisfying conditions!

Solving Line Intersection



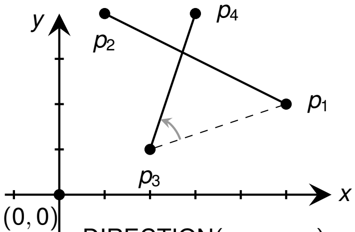
$$\text{DIRECTION}(p_3, p_4, p_1) = (p_1 - p_3) \times (p_4 - p_3)$$

```
DIRECTION(p_i, p_j, p_k)
    return (p_k − p_i) × (p_j − p_i)
```

```
SEGMENTS-INTERSECT(p_i, p_j, p_k)
    d_1 = DIRECTION(p_3, p_4, p_1)
    d_2 = DIRECTION(p_3, p_4, p_2)
    d_3 = DIRECTION(p_1, p_2, p_3)
    d_4 = DIRECTION(p_1, p_2, p_4)
    If d_1 · d_2 < 0 and d_3 · d_4 < 0 return TRUE
        · · · (handle all degenerate cases)
```

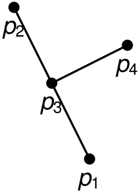
Lines could touch or be colinear

Solving Line Intersection



$$\text{DIRECTION}(p_3, p_4, p_1) = (p_1 - p_3) \times (p_4 - p_3)$$

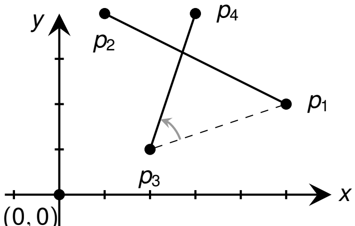
```
DIRECTION(p_i, p_j, p_k)
    return (p_k − p_i) × (p_j − p_i)
```




```
SEGMENTS-INTERSECT(p_i, p_j, p_k)
    d_1 = DIRECTION(p_3, p_4, p_1)
    d_2 = DIRECTION(p_3, p_4, p_2)
    d_3 = DIRECTION(p_1, p_2, p_3)
    d_4 = DIRECTION(p_1, p_2, p_4)
    If d_1 · d_2 < 0 and d_3 · d_4 < 0 return TRUE
        · · · (handle all degenerate cases)
```

Lines could touch or be colinear

Solving Line Intersection



$$\text{DIRECTION}(p_3, p_4, p_1) = (p_1 - p_3) \times (p_4 - p_3)$$

```
DIRECTION(p_i, p_j, p_k)
    return (p_k − p_i) × (p_j − p_i)
```

```
SEGMENTS-INTERSECT(p_i, p_j, p_k)
    d_1 = DIRECTION(p_3, p_4, p_1)
    d_2 = DIRECTION(p_3, p_4, p_2)
    d_3 = DIRECTION(p_1, p_2, p_3)
    d_4 = DIRECTION(p_1, p_2, p_4)
    If d_1 · d_2 < 0 and d_3 · d_4 < 0 return TRUE
        ··· (handle all degenerate cases)
```

Lines could touch or be colinear