How simple robots benefit from looking back

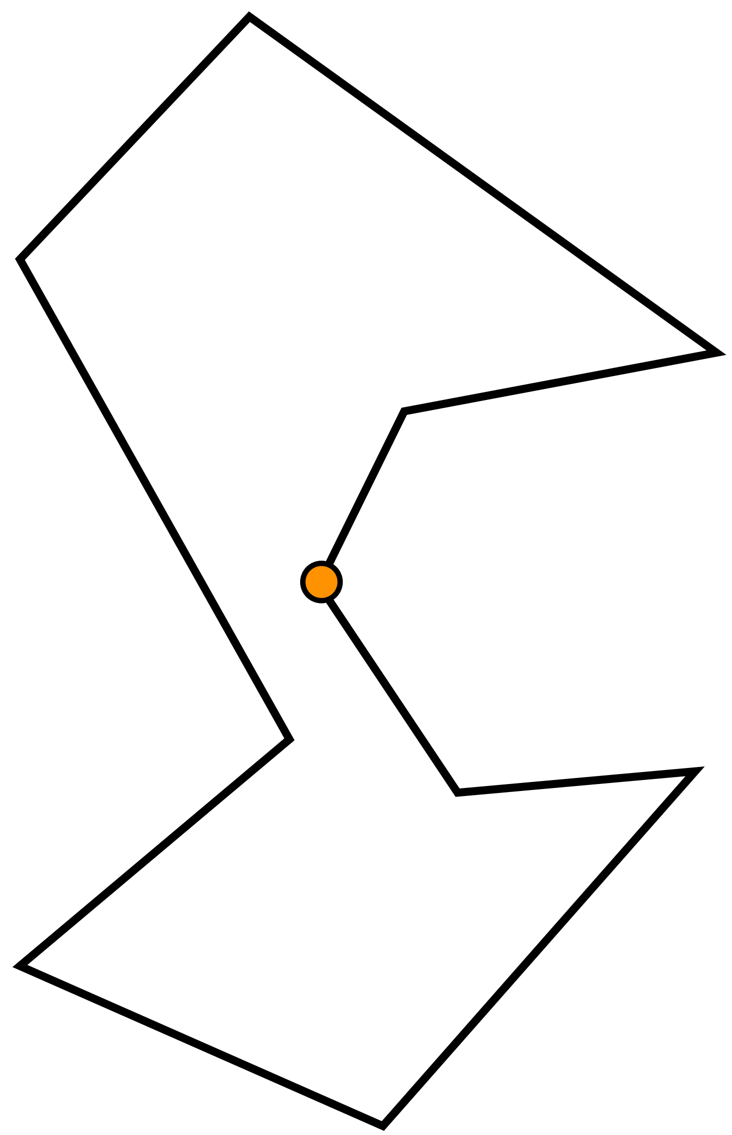
J. Chalopin, S. Das, Y. Disser, M. Mihalák, P. Widmayer

Introduction

Introduction

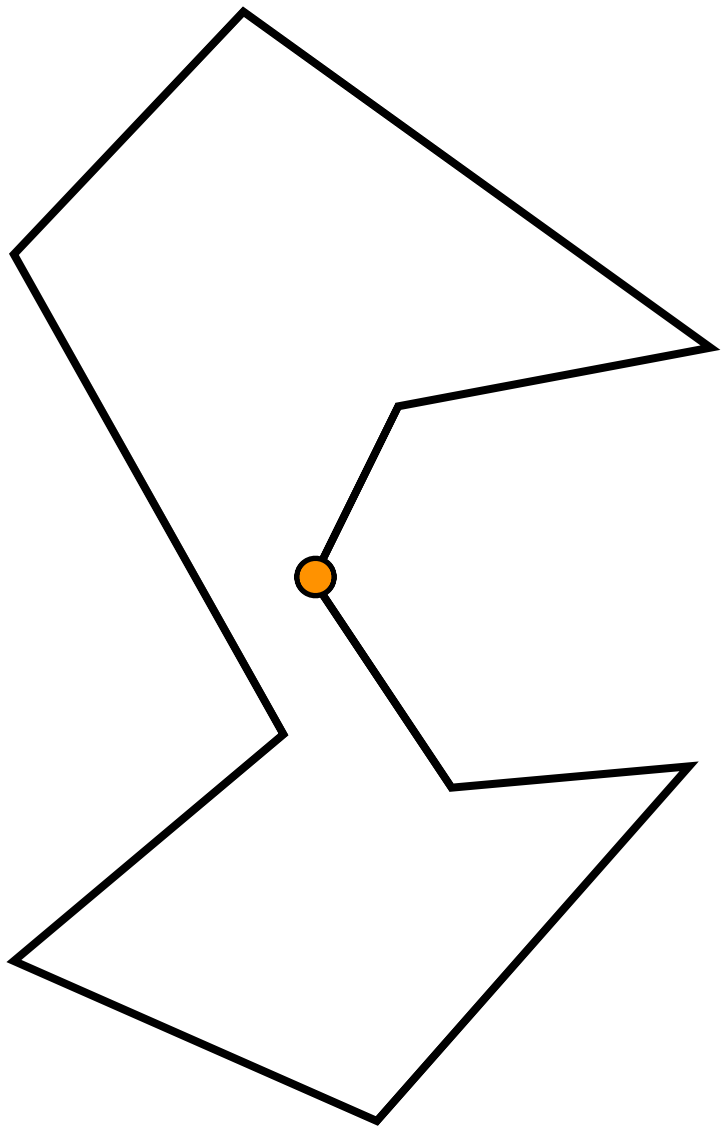
motivation

Introduction
motivation



- Robot inside an unknown polygon

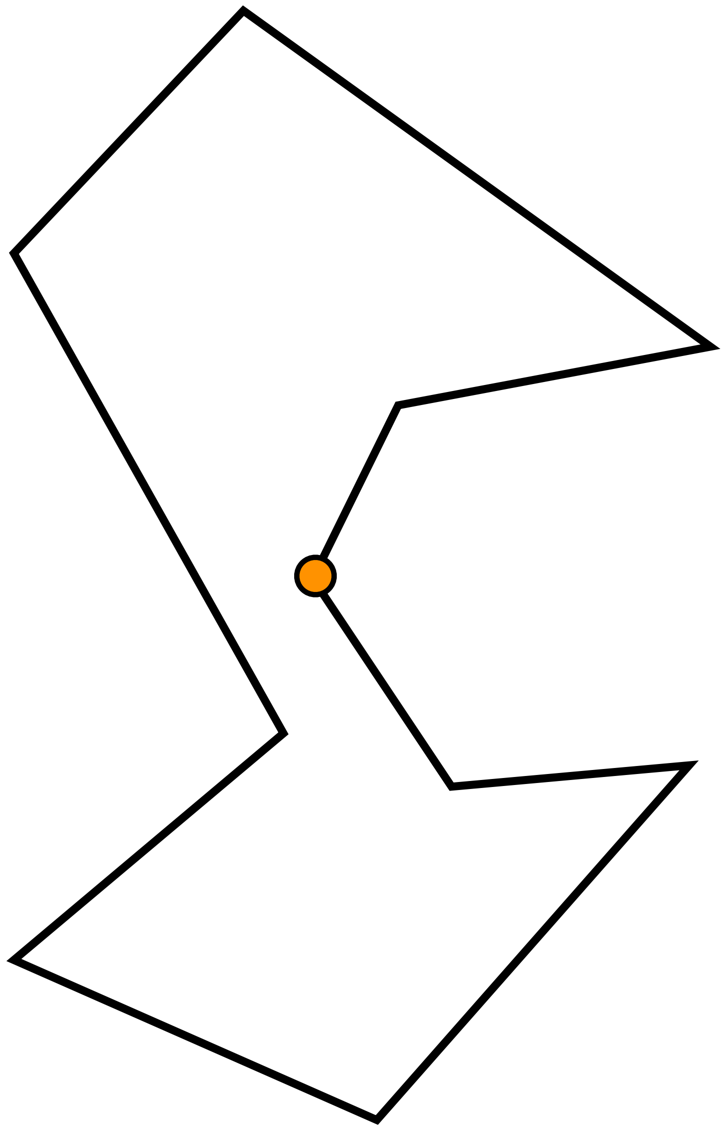
Introduction
motivation



- Robot inside an unknown polygon
- Tasks:

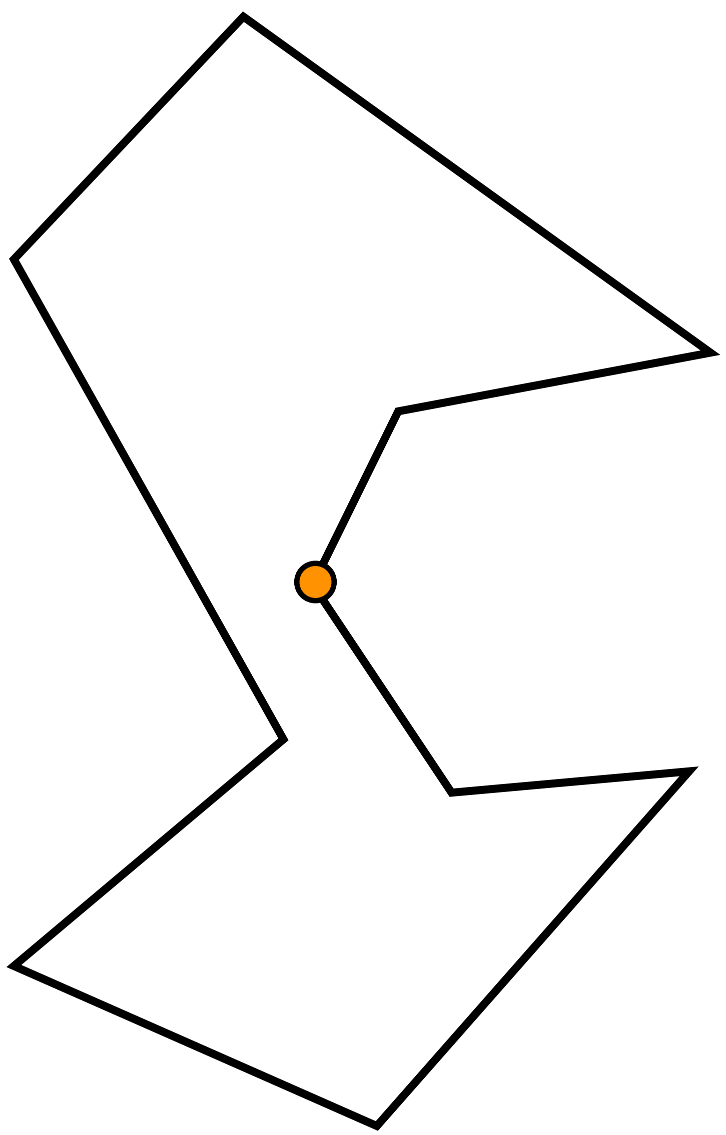
Introduction

motivation



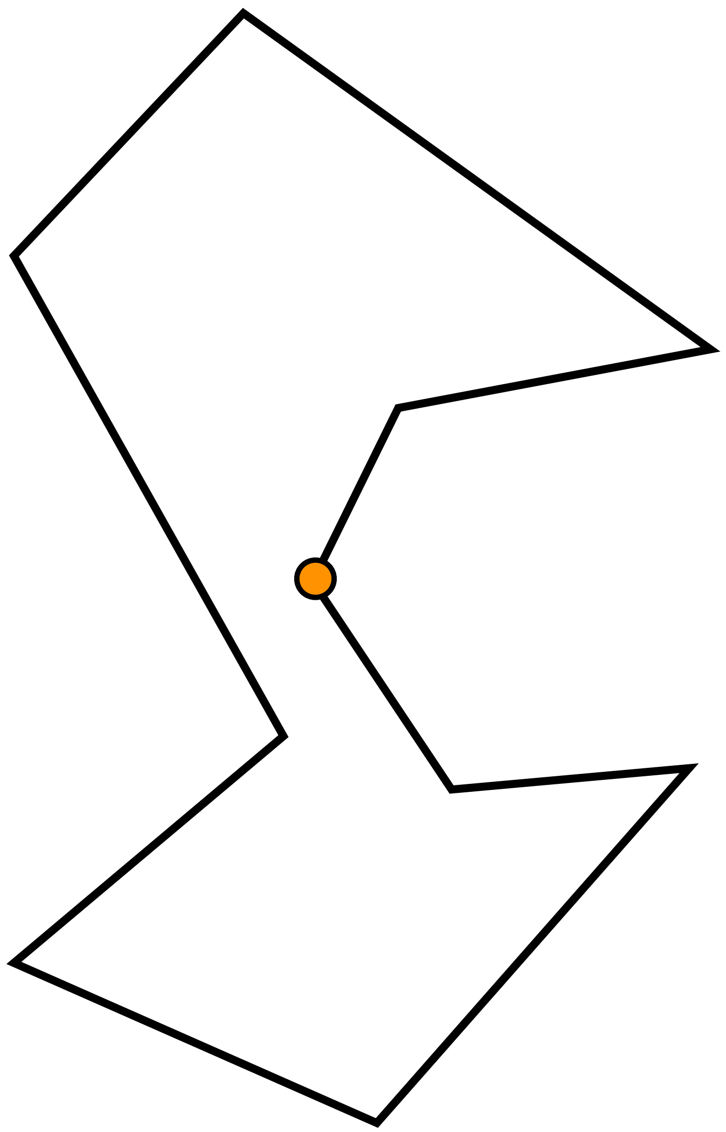
- Robot inside an unknown polygon
- Tasks:
  - meet identical robots

Introduction
motivation



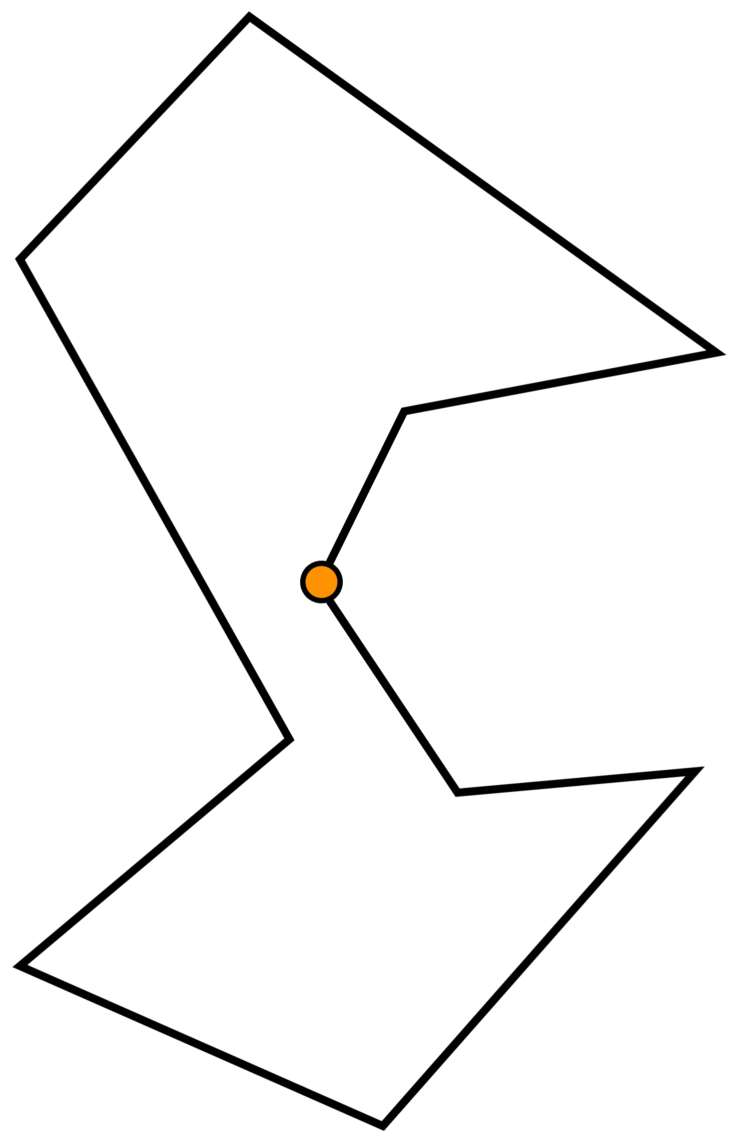
- Robot inside an unknown polygon
- Tasks:
  - meet identical robots
  - draw a map

Introduction
motivation



- Robot inside an unknown polygon
- Tasks:
  - meet identical robots
  - draw a map
- Q: How simple can we make the robot?

Introduction
motivation

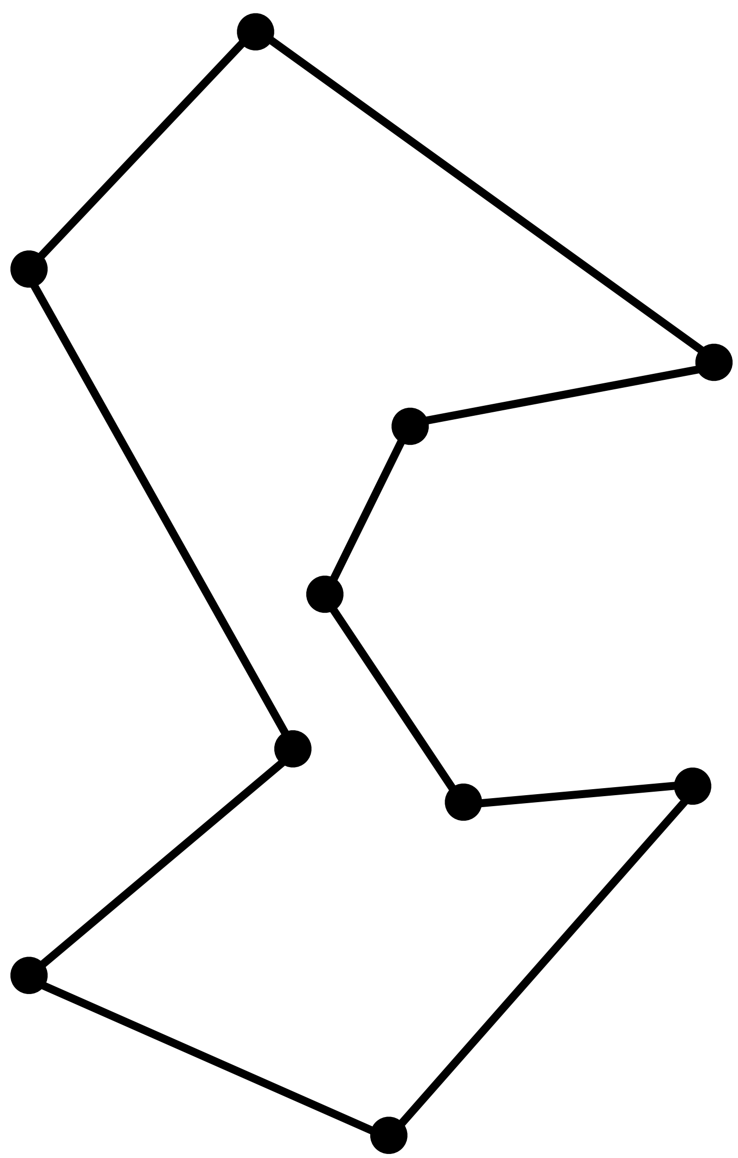


- Robot inside an unknown polygon
- Tasks:
  - meet identical robots
  - draw a map
- Q: How simple can we make the robot?

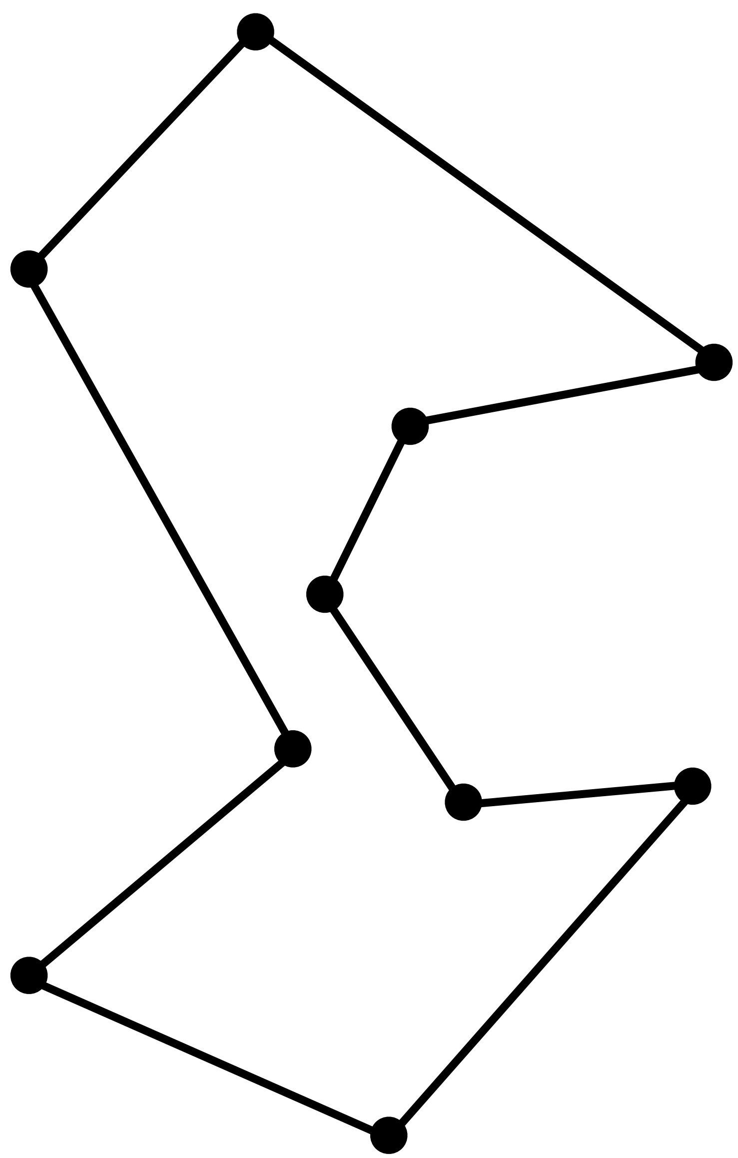
  ⇒find simplistic design

Introduction

visibilities

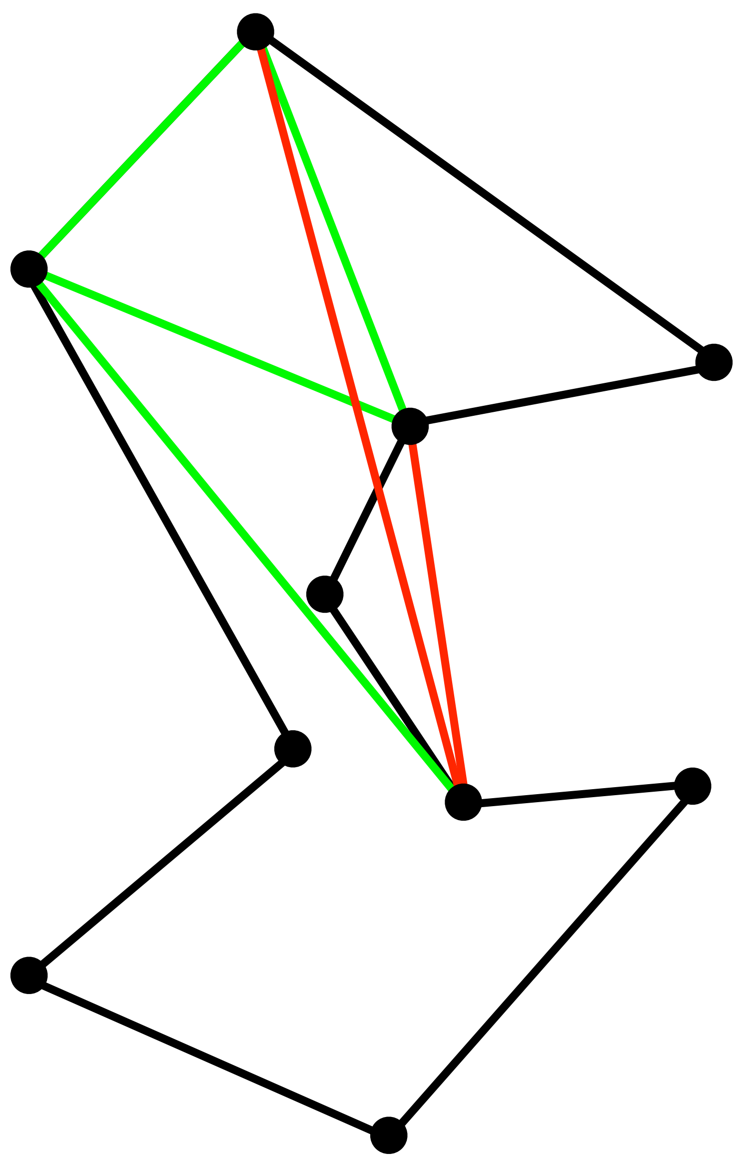


Introduction
visibilities



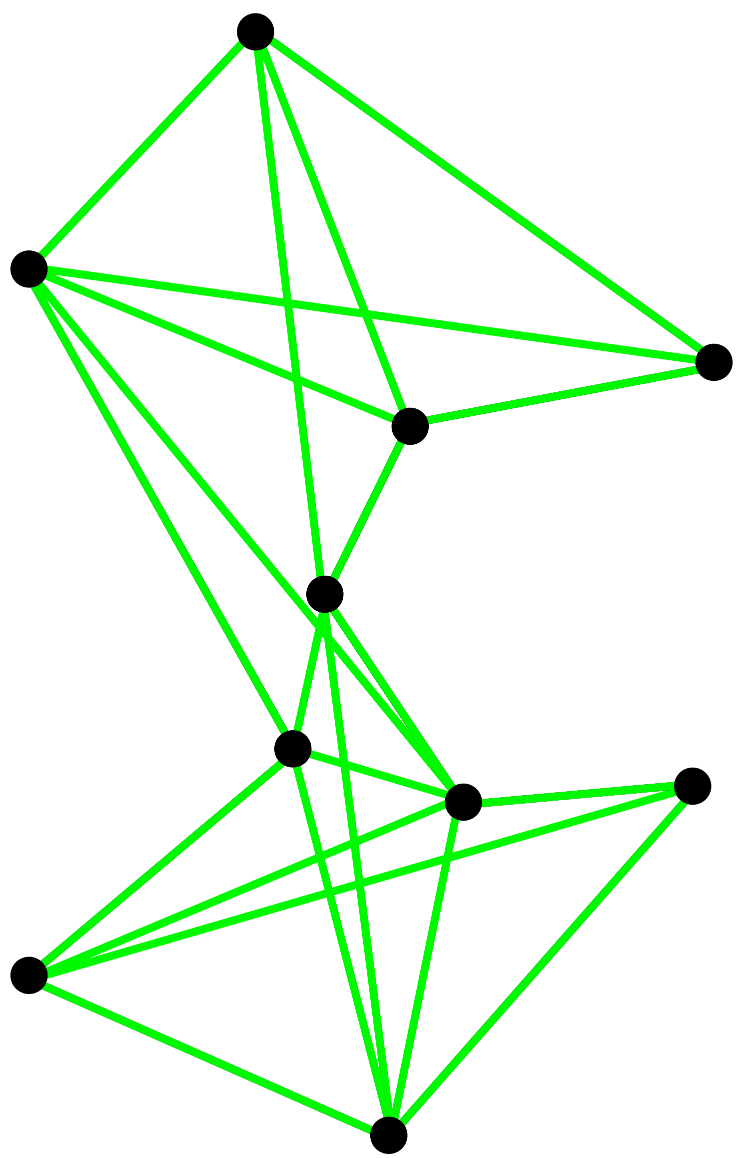
- Vertices are mut. visible: segment is inside polygon

Introduction
visibilities



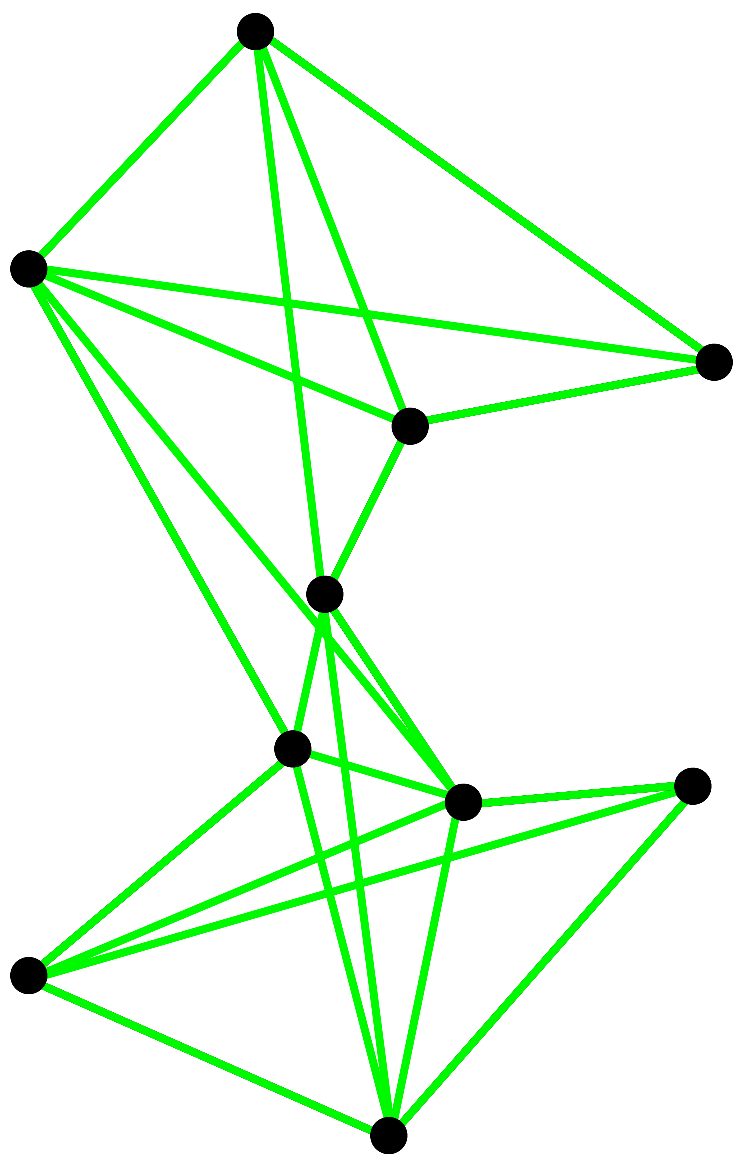
- Vertices are mut. visible: segment is inside polygon

Introduction
visibilities



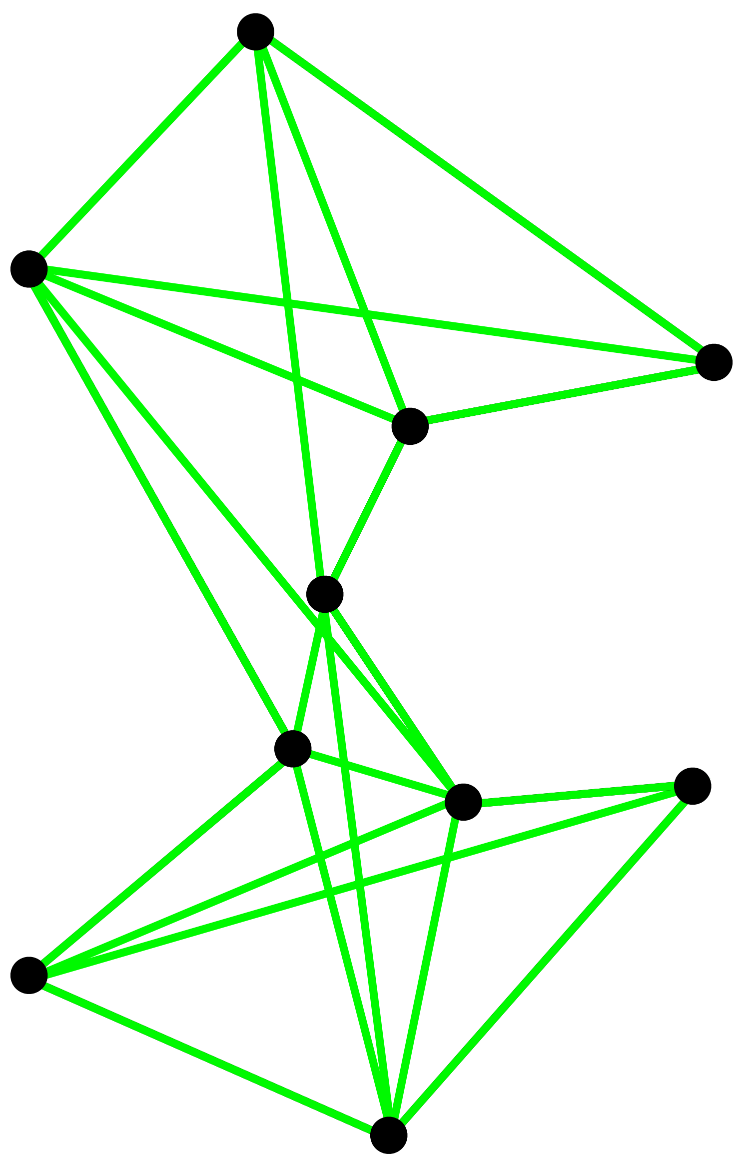
- Vertices are mut. visible: segment is inside polygon
- Visibility graph: edge for every pair of visible verts

Introduction
visibilities



- Vertices are mut. visible: segment is inside polygon
- Visibility graph: edge for every pair of visible verts
  ⇒topological map

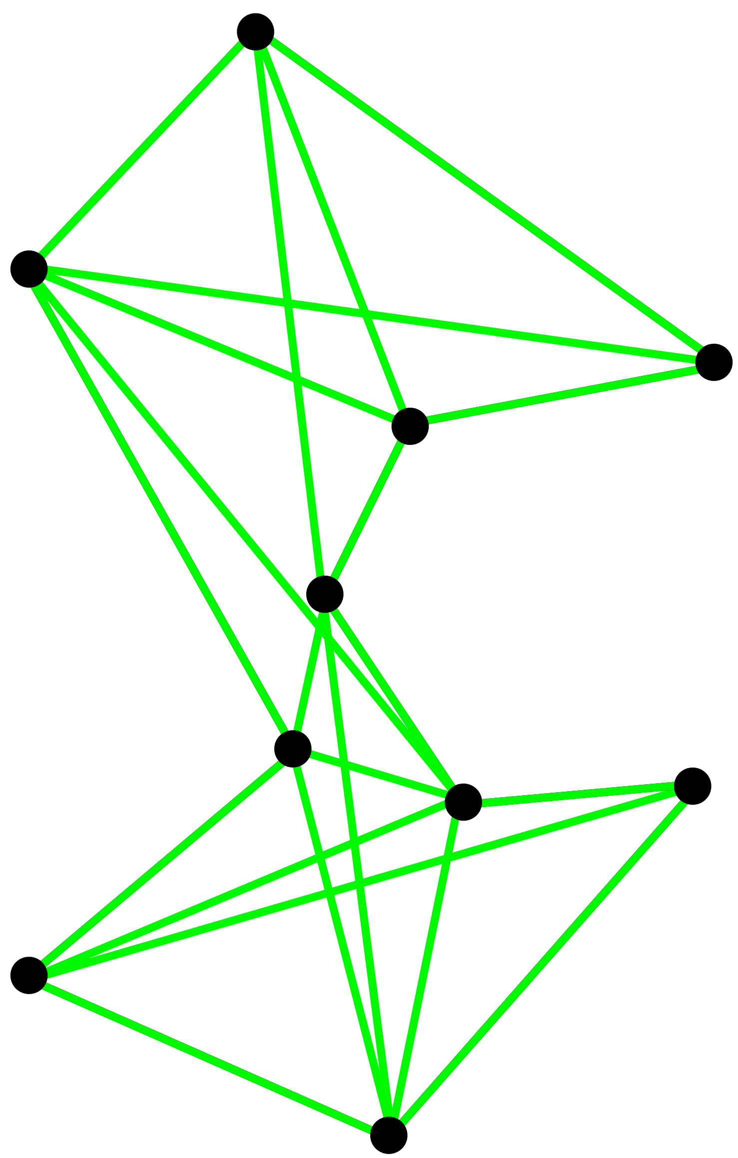
Introduction
visibilities



- Vertices are mut. visible: segment is inside polygon
- Visibility graph: edge for every pair of visible verts
  ⇒topological map
- Meeting problem:

Introduction

visibilities

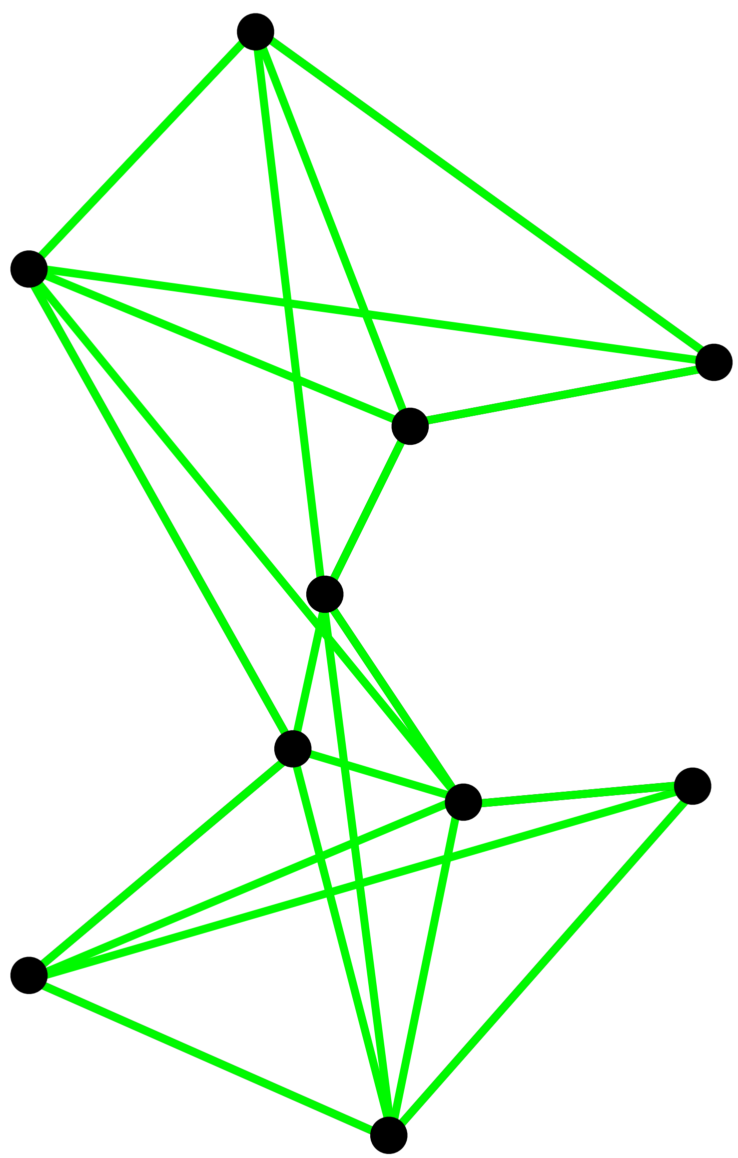


- Vertices are mut. visible: segment is inside polygon
- Visibility graph: edge for every pair of visible verts

  ⇒topological map
- Meeting problem:

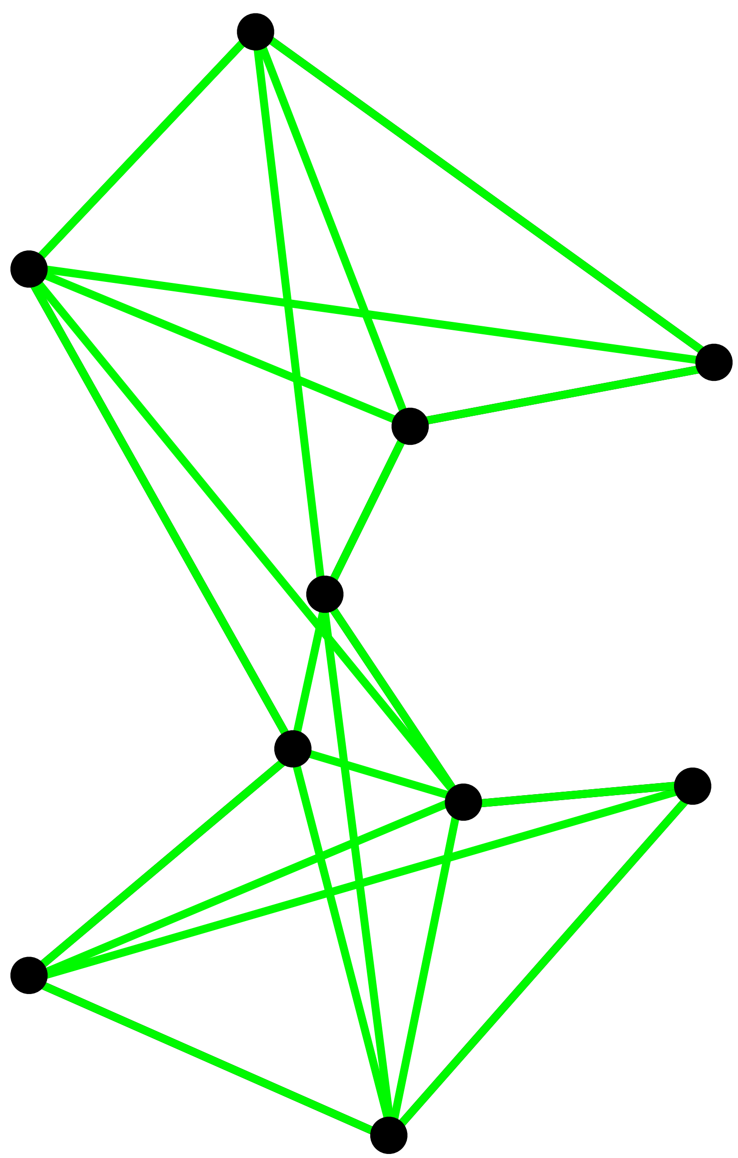
  ⇒robots form a clique

Introduction
visibilities



- Vertices are mut. visible: segment is inside polygon
- Visibility graph: edge for every pair of visible verts
  ⇒topological map
- Meeting problem:
  ⇒robots form a clique
- Mapping problem:

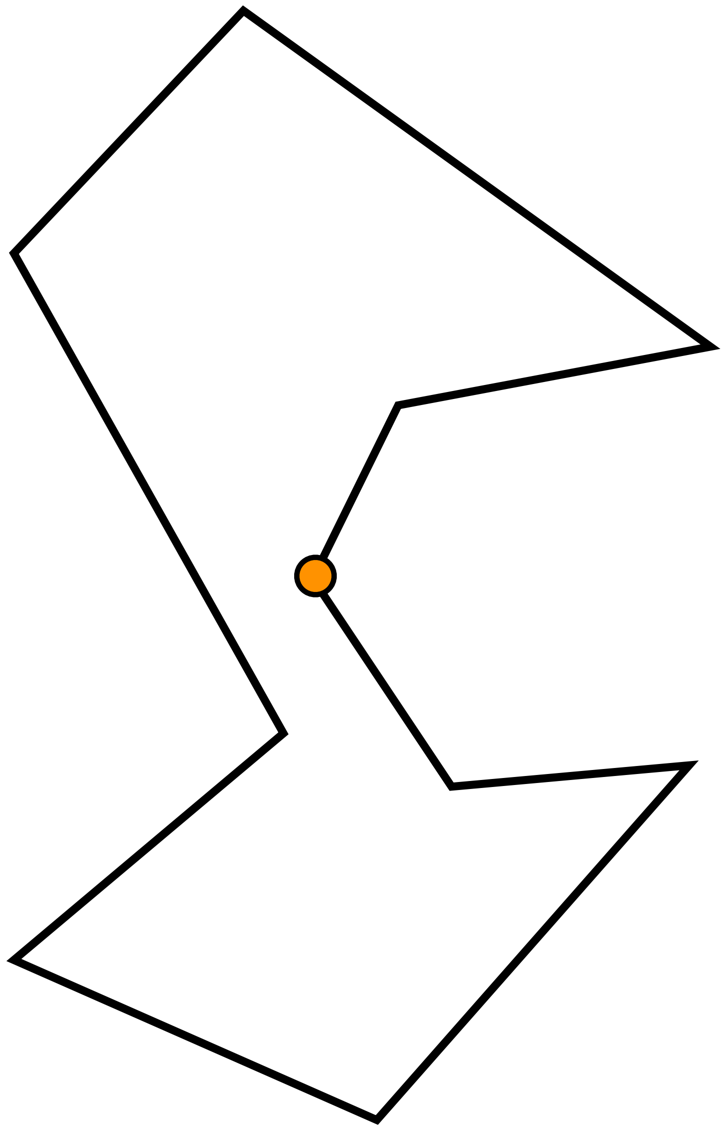
Introduction
visibilities



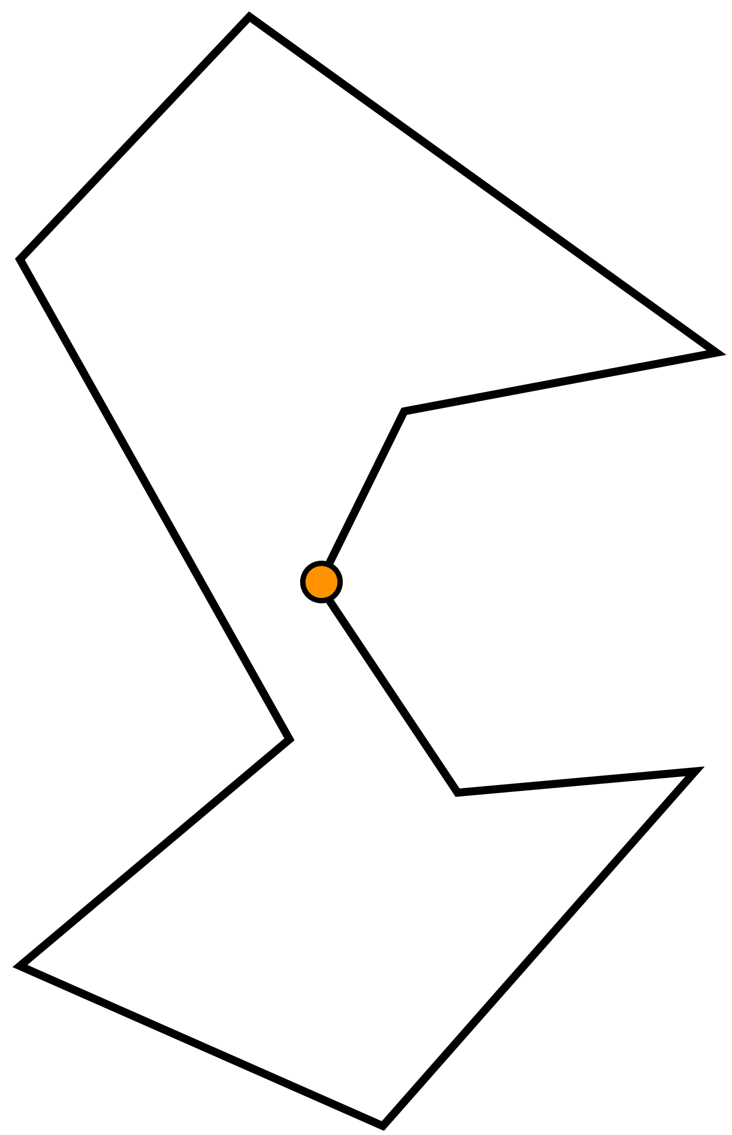
- Vertices are mut. visible: segment is inside polygon
- Visibility graph: edge for every pair of visible verts
  $\Rightarrow$topological map
- Meeting problem:
  $\Rightarrow$robots form a clique
- Mapping problem:
  $\Rightarrow$reconstruct vis. graph

Introduction

robot model

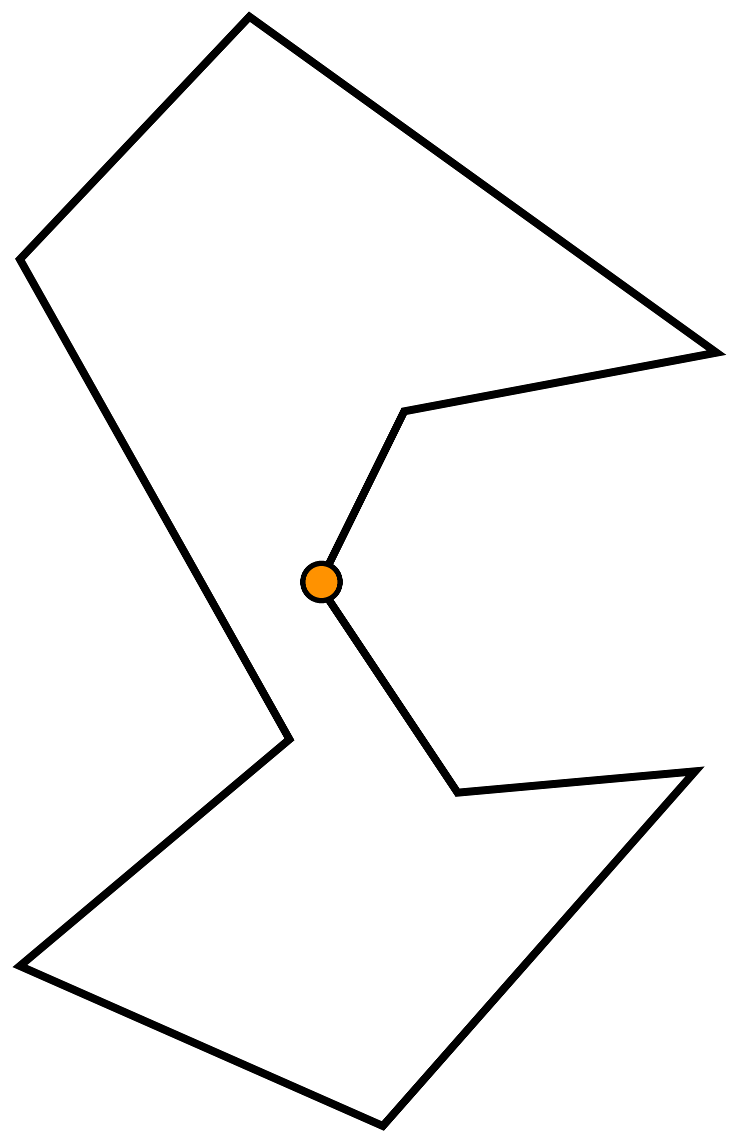


Introduction
robot model



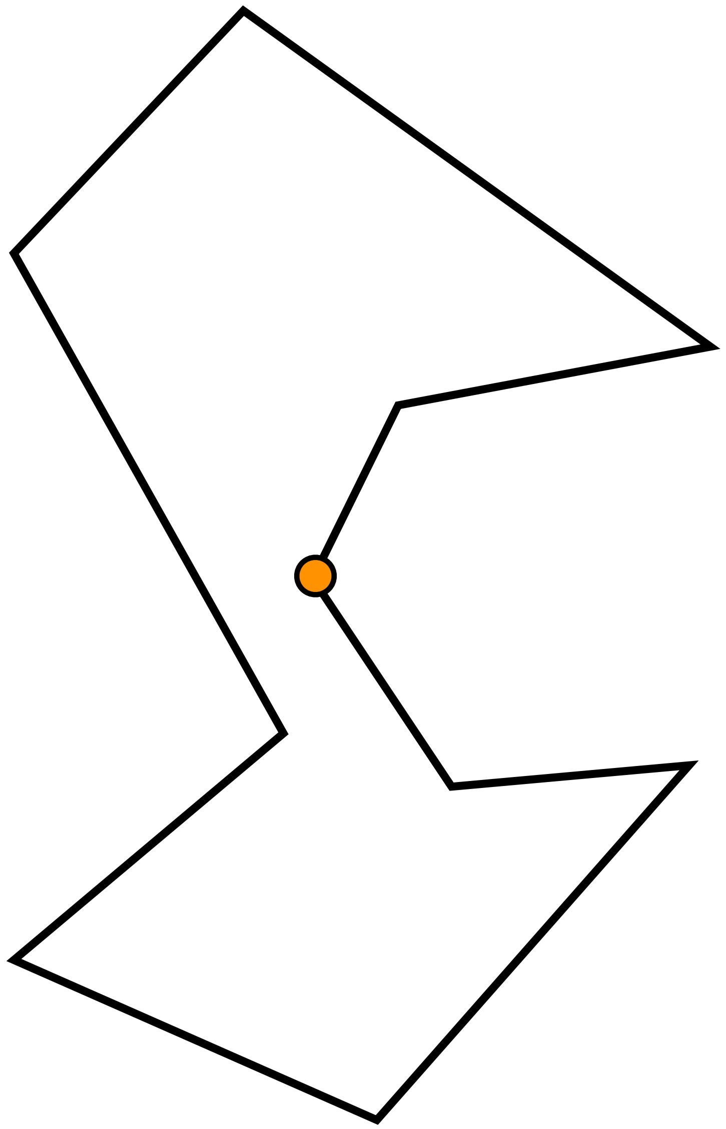
- We assume n is given

Introduction
robot model



- We assume n is given
- We allow the robot to

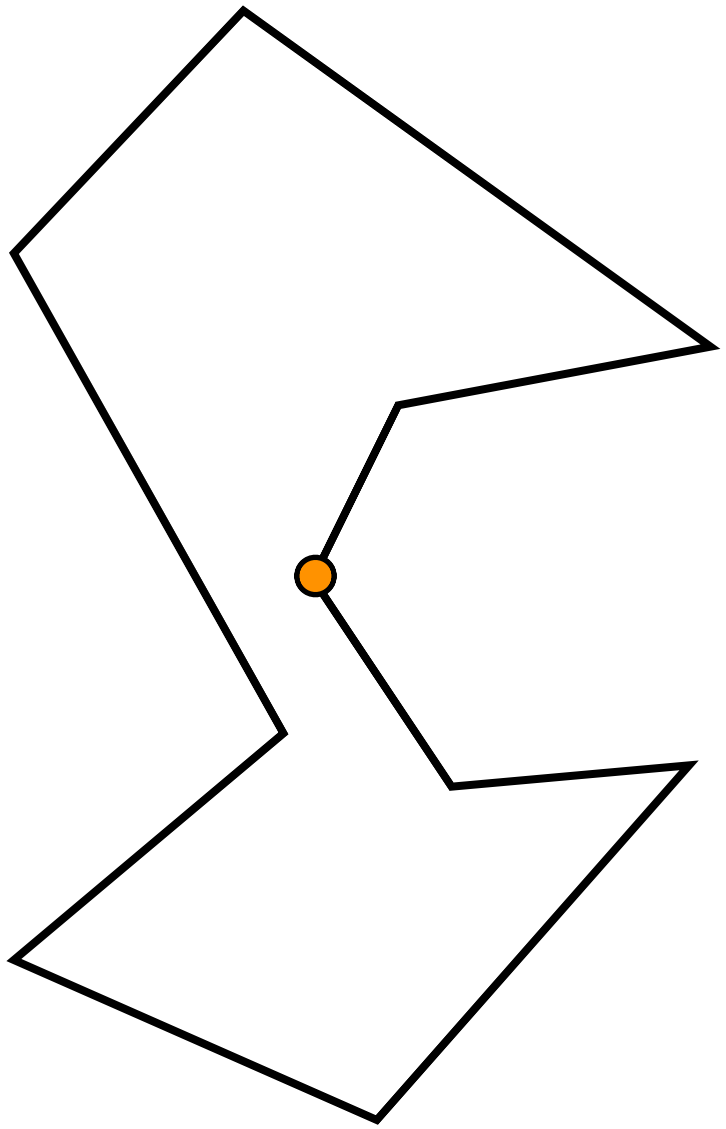
Introduction
robot model



- We assume n is given
- We allow the robot to
  - while at a vertex:

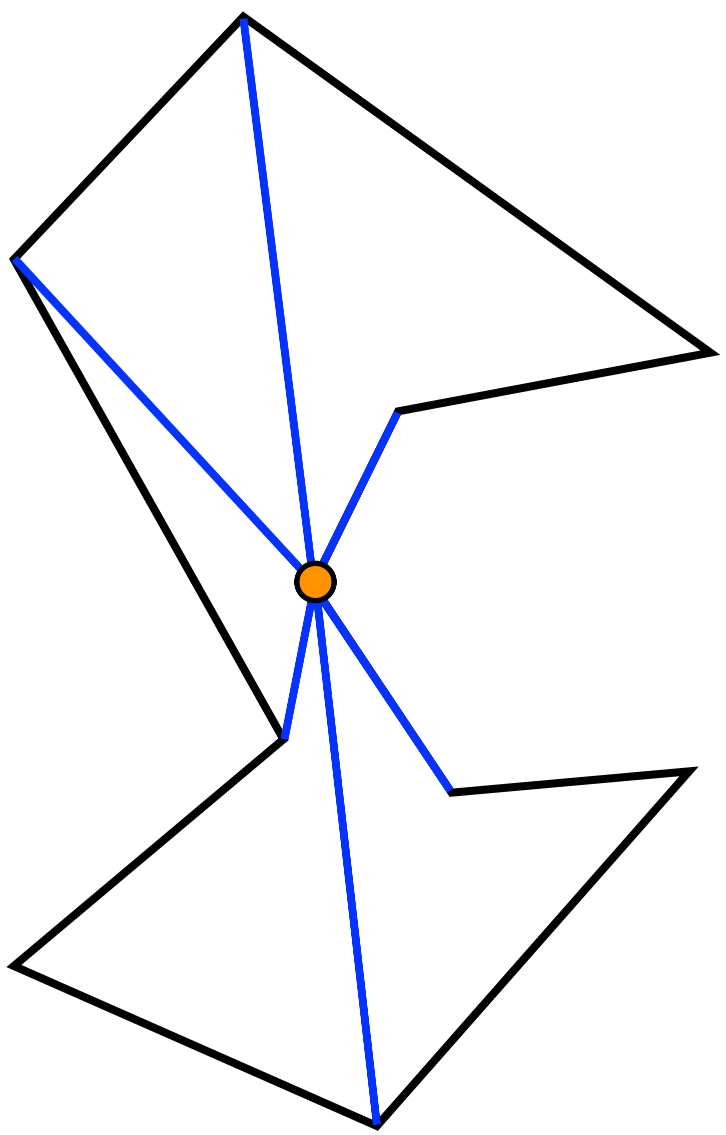
Introduction

robot model



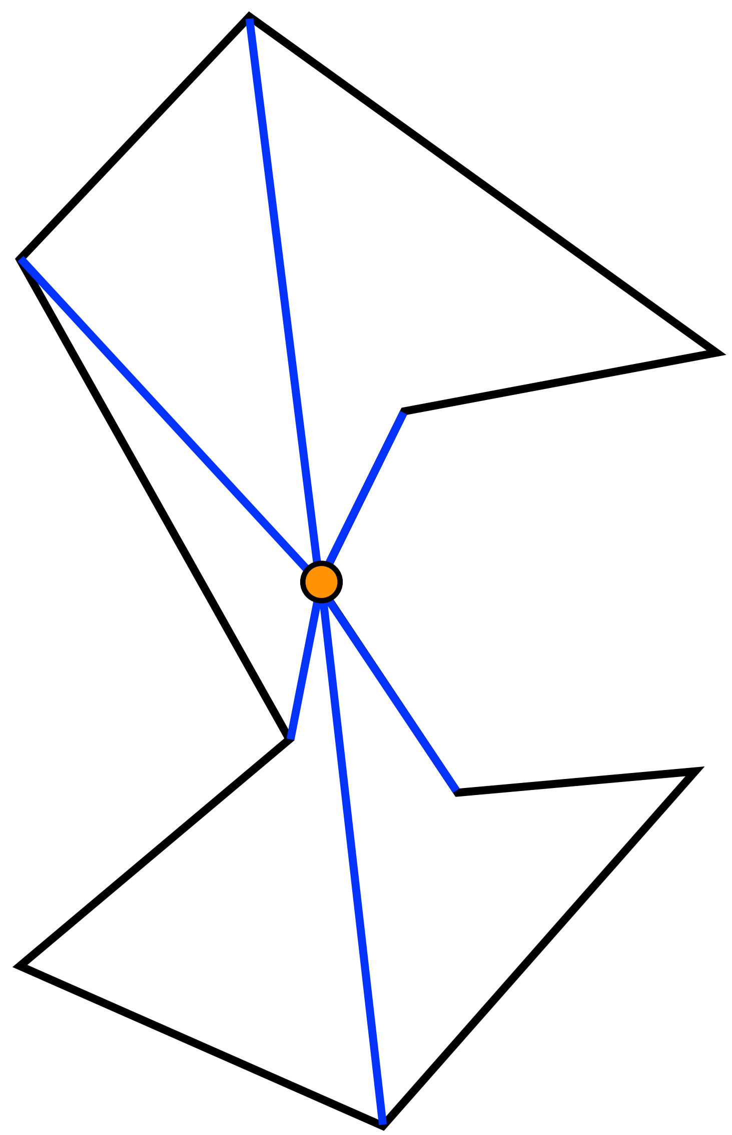
- We assume n is given
- We allow the robot to
  - while at a vertex:
    - see visible vertices

Introduction
robot model



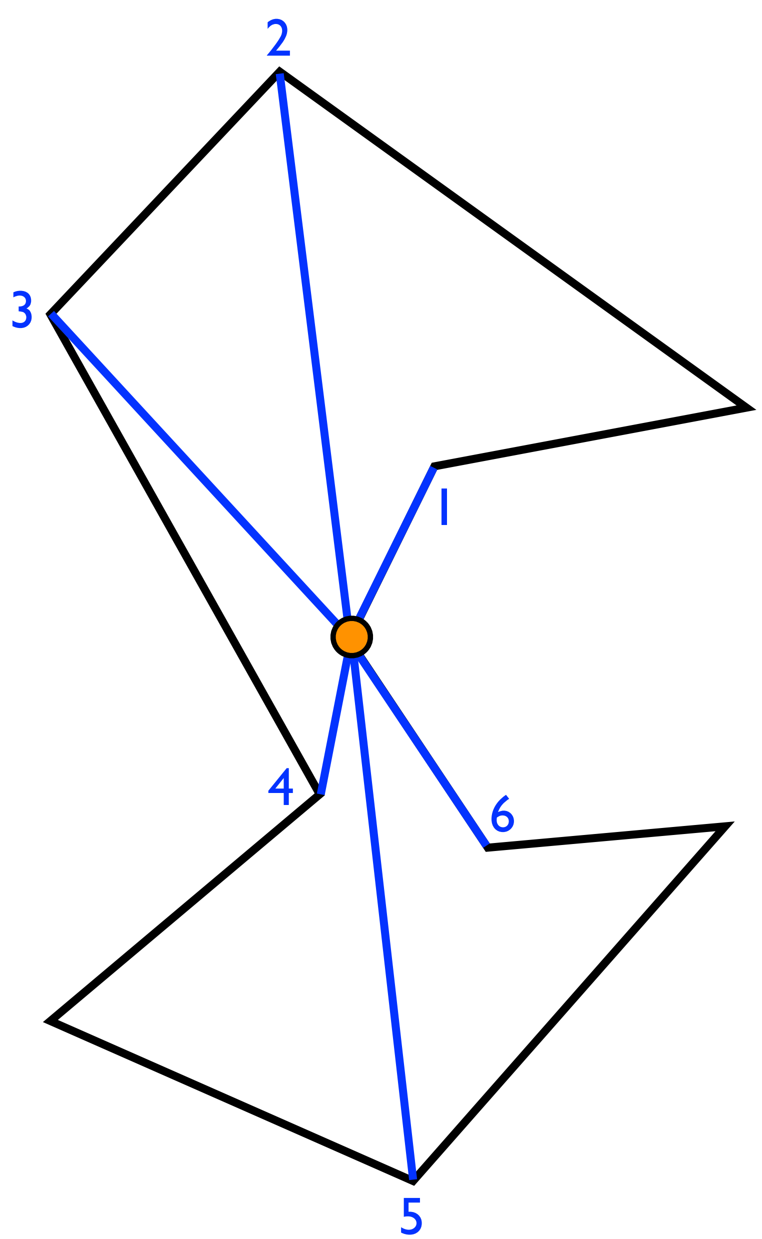
- We assume n is given
- We allow the robot to
  - while at a vertex:
    - see visible vertices

Introduction
robot model



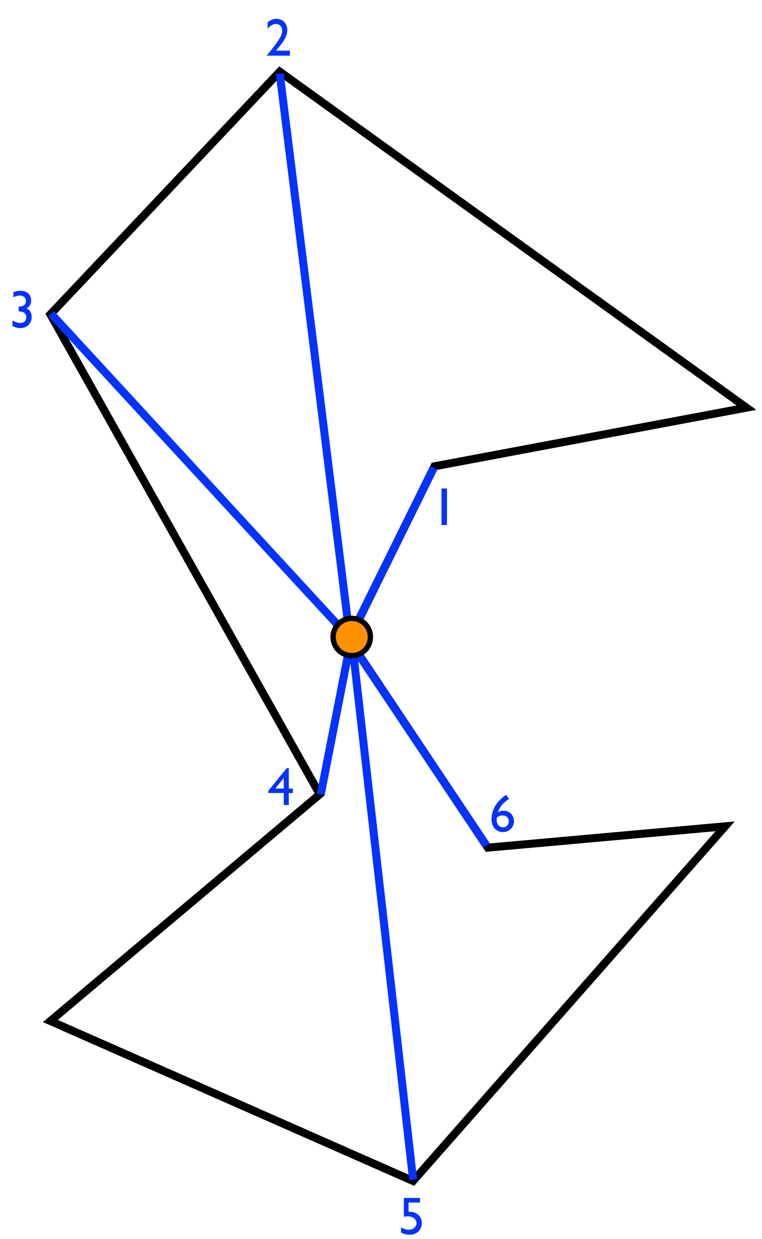
- We assume n is given
- We allow the robot to
  - while at a vertex:
    - see visible vertices
    - order vertices (ccw)

Introduction
robot model



- We assume n is given
- We allow the robot to
  - while at a vertex:
    - see visible vertices
    - order vertices (ccw)

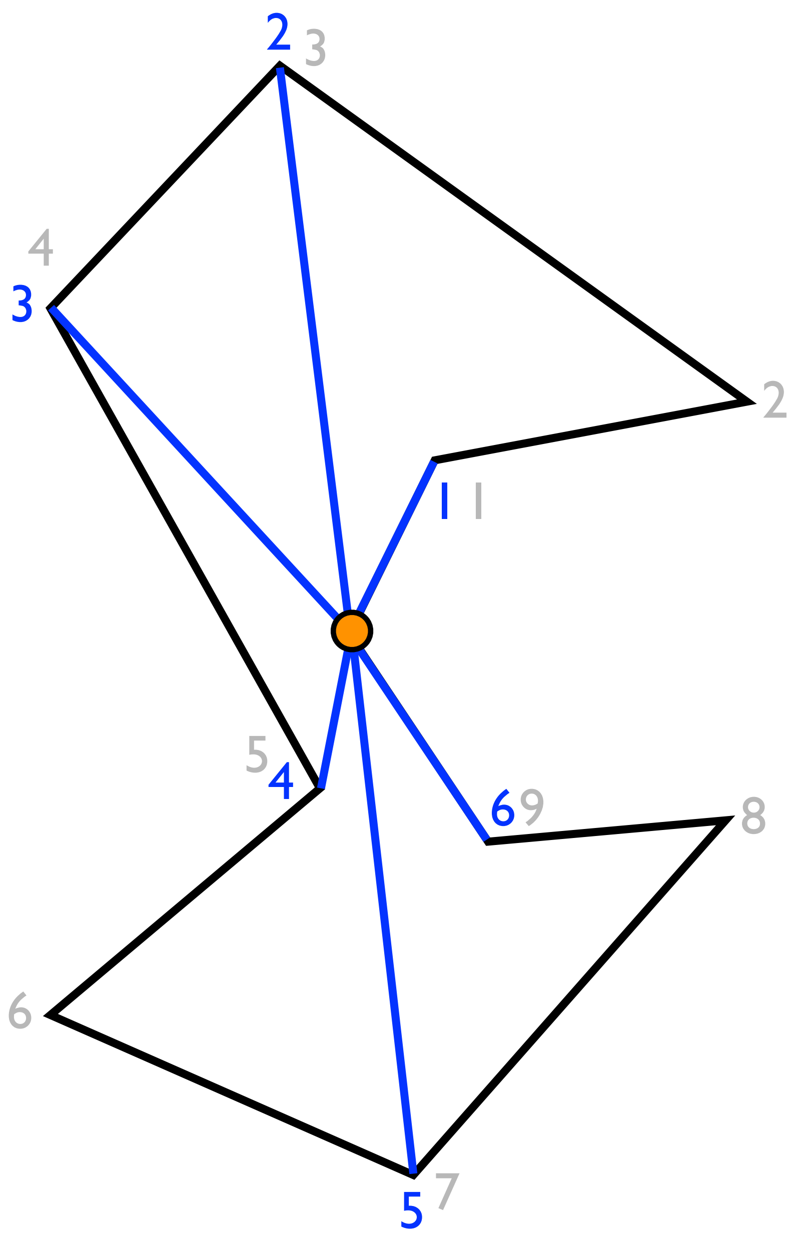
Introduction
robot model



- We assume n is given
- We allow the robot to
  - while at a vertex:
    - see visible vertices
    - order vertices (ccw)

  ⇒ no global ID!

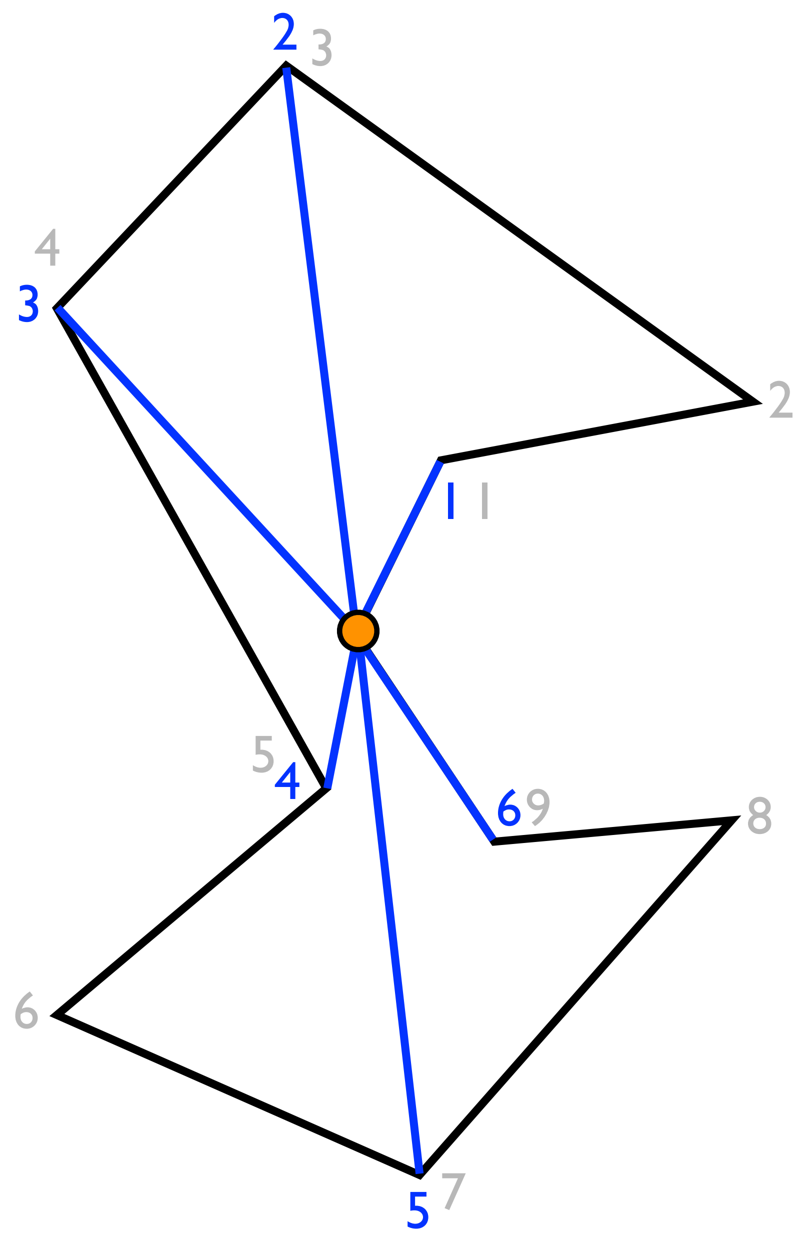
Introduction
robot model



- We assume n is given
- We allow the robot to
  - while at a vertex:
    - see visible vertices
    - order vertices (ccw)

  ⇒ no global ID!

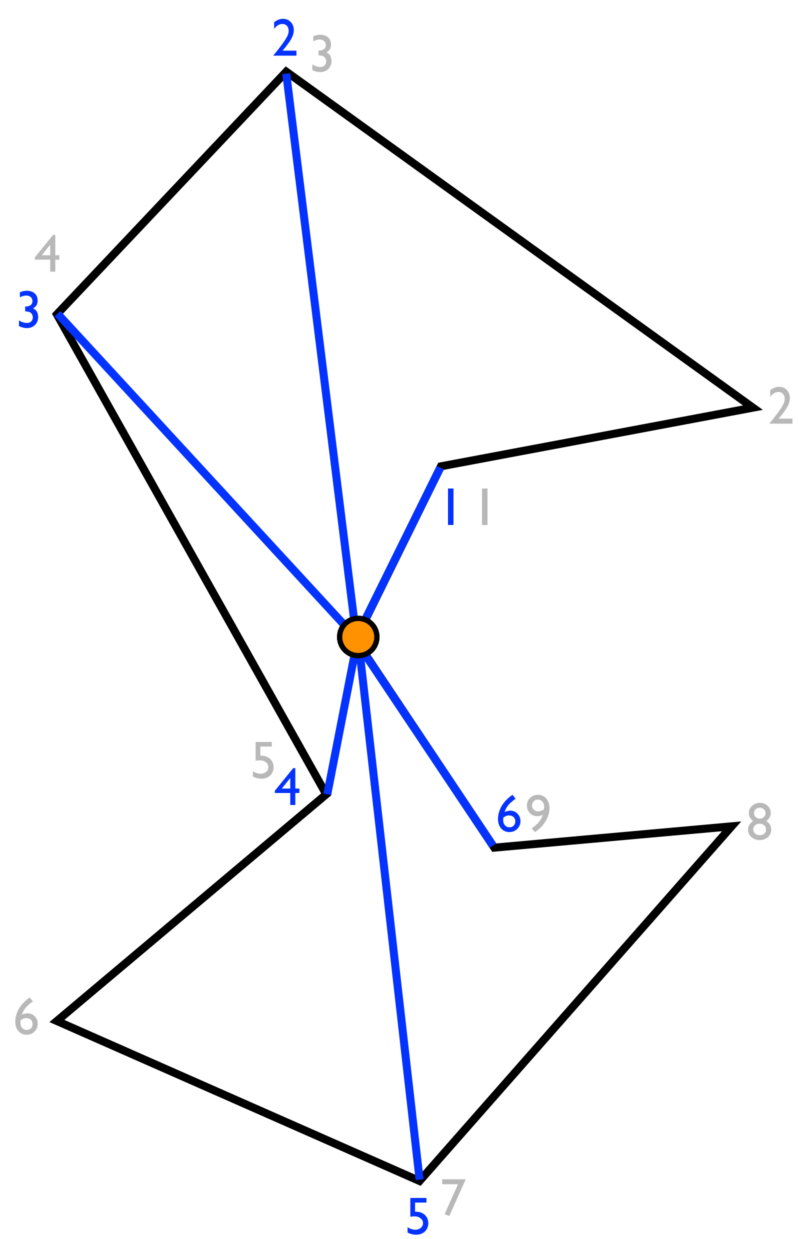
Introduction
robot model



- We assume n is given
- We allow the robot to
  - while at a vertex:
    - see visible vertices
    - order vertices (ccw)

    ⇒ no global ID!
  - have (enough) memory

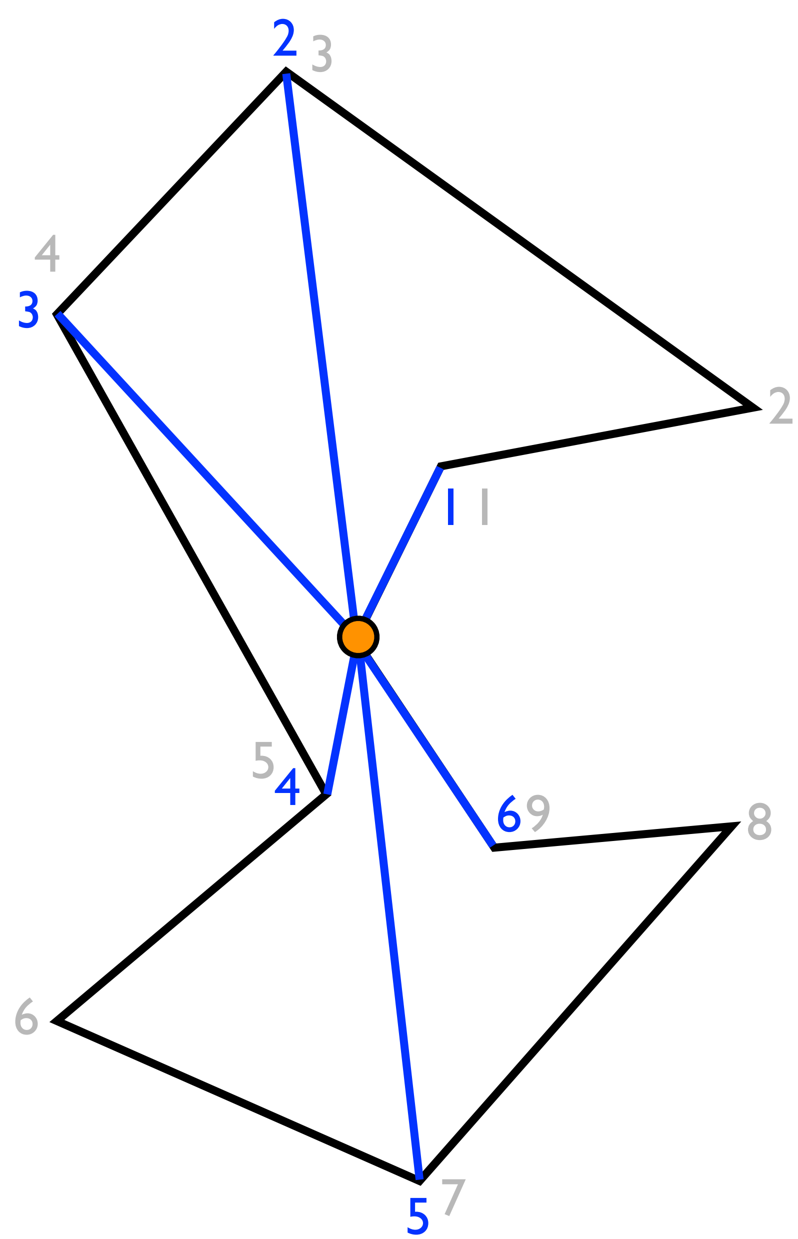
Introduction
robot model



- We assume n is given
- We allow the robot to
  - while at a vertex:
    - see visible vertices
    - order vertices (ccw)

    ⇒ no global ID!
  - have (enough) memory
  - move to visible verts

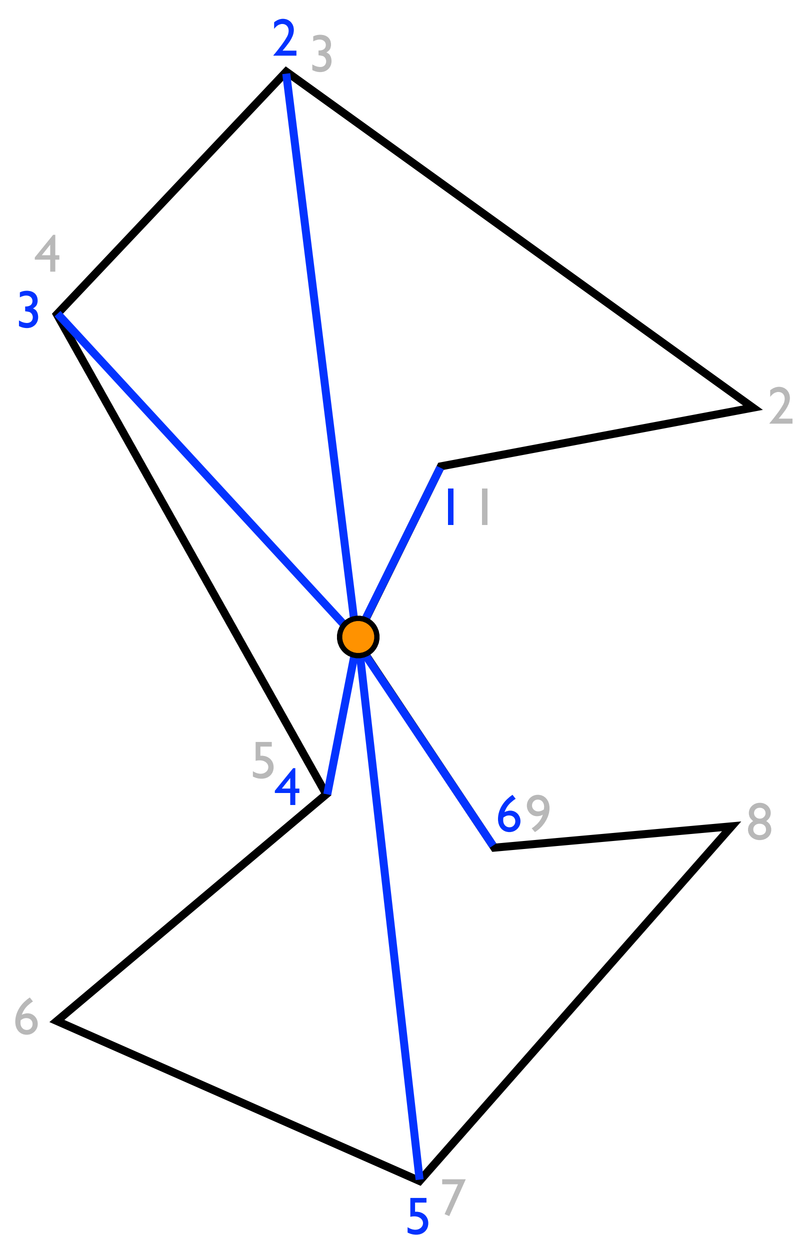
Introduction
robot model



- We assume n is given
- We allow the robot to
  - while at a vertex:
    - see visible vertices
    - order vertices (ccw)

    ⇒ no global ID!
  - have (enough) memory
  - move to visible verts

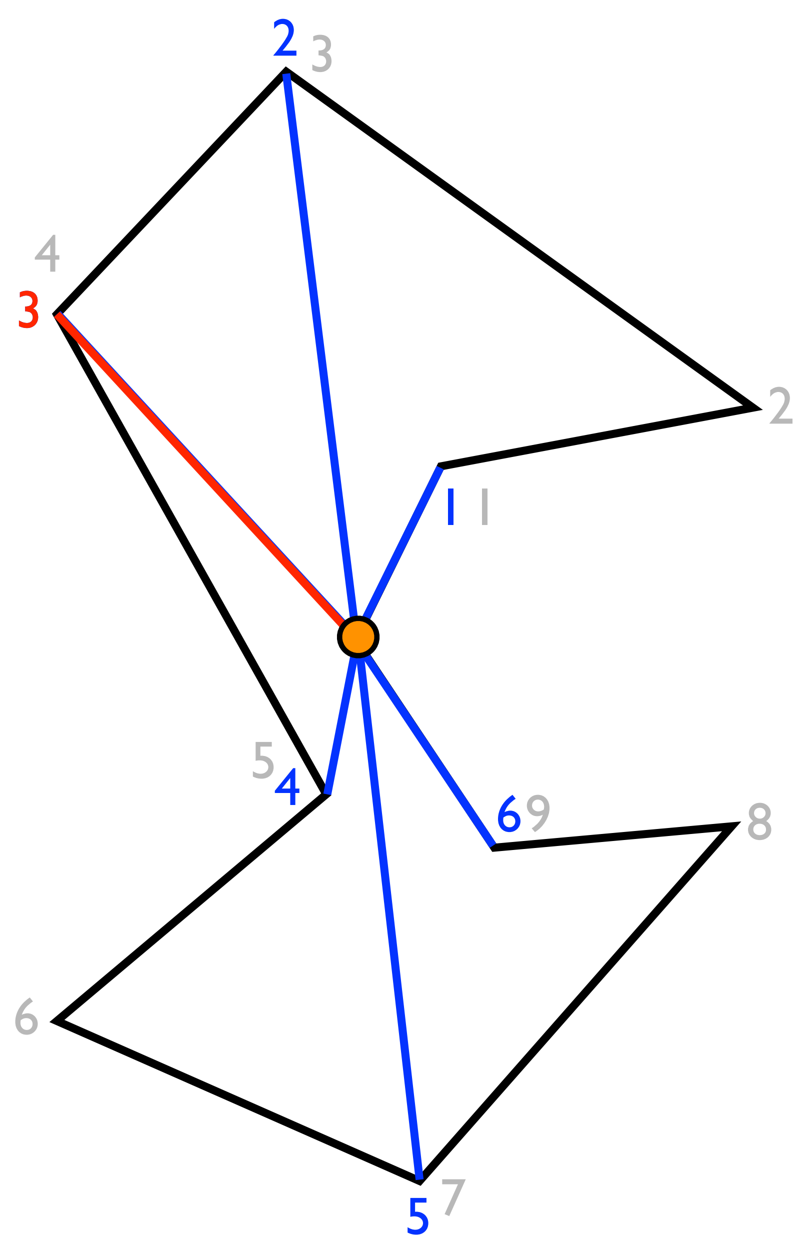
Introduction
robot model



- We assume n is given
- We allow the robot to
  - while at a vertex:
    - see visible vertices
    - order vertices (ccw)

    ⇒ no global ID!
  - have (enough) memory
  - move to visible verts
  - look-back

Introduction
robot model



- We assume n is given
- We allow the robot to
  - while at a vertex:
    - see visible vertices
    - order vertices (ccw)

    ⇒ no global ID!
  - have (enough) memory
  - move to visible verts
  - look-back

    ⇒ origin of last move

Vertices as Viewpoints

Vertices as Viewpoints

view

Vertices as Viewpoints

view

- Capture information a robot can collect about v

Vertices as Viewpoints

view

- Capture information a robot can collect about v

⇒*view* from v

Vertices as Viewpoints

view

- Capture information a robot can collect about v

⇒*view* from v

⇒collection of all paths

Vertices as Viewpoints

view

- Capture information a robot can collect about v

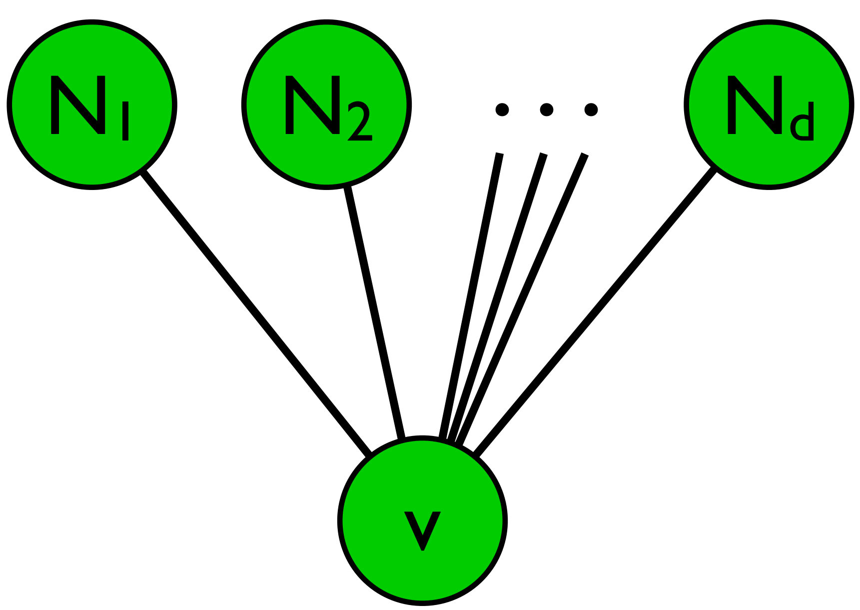
  ⇒ *view* from v

  ⇒ collection of all paths

- level-1-view:

Vertices as Viewpoints

view



- Capture information a robot can collect about v

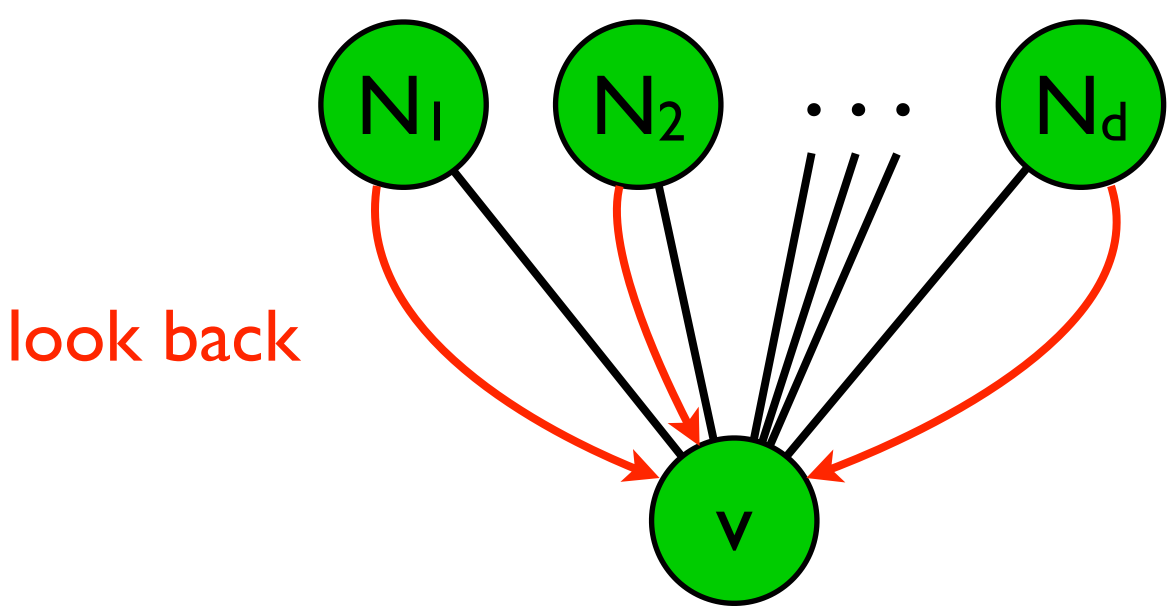
  ⇒ *view* from v

  ⇒ collection of all paths

- level-1-view:

Vertices as Viewpoints

view



- Capture information a robot can collect about v

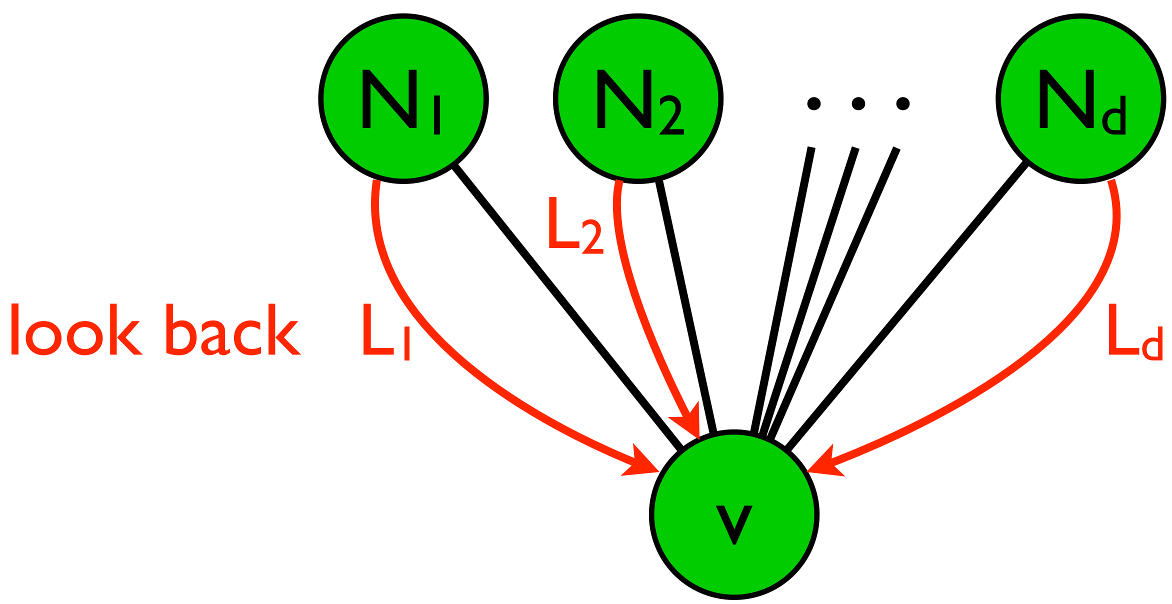
  ⇒ *view* from v

  ⇒ collection of all paths

- level-1-view:

Vertices as Viewpoints

view



- Capture information a robot can collect about v

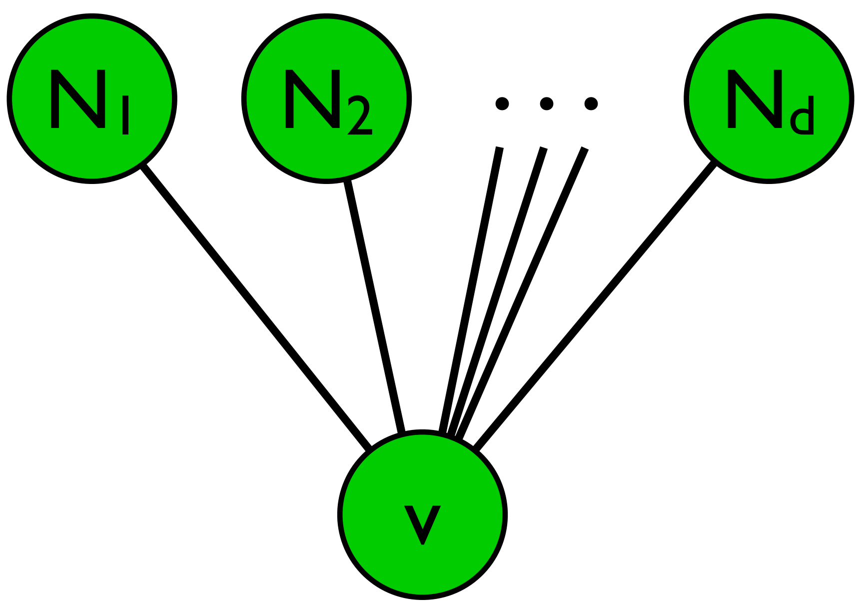
  ⇒ *view* from v

  ⇒ collection of all paths

- level-1-view:
  $v^1 = (L_1, L_2, ..., L_d)$

Vertices as Viewpoints

view



- Capture information a robot can collect about v

  ⇒ *view* from v

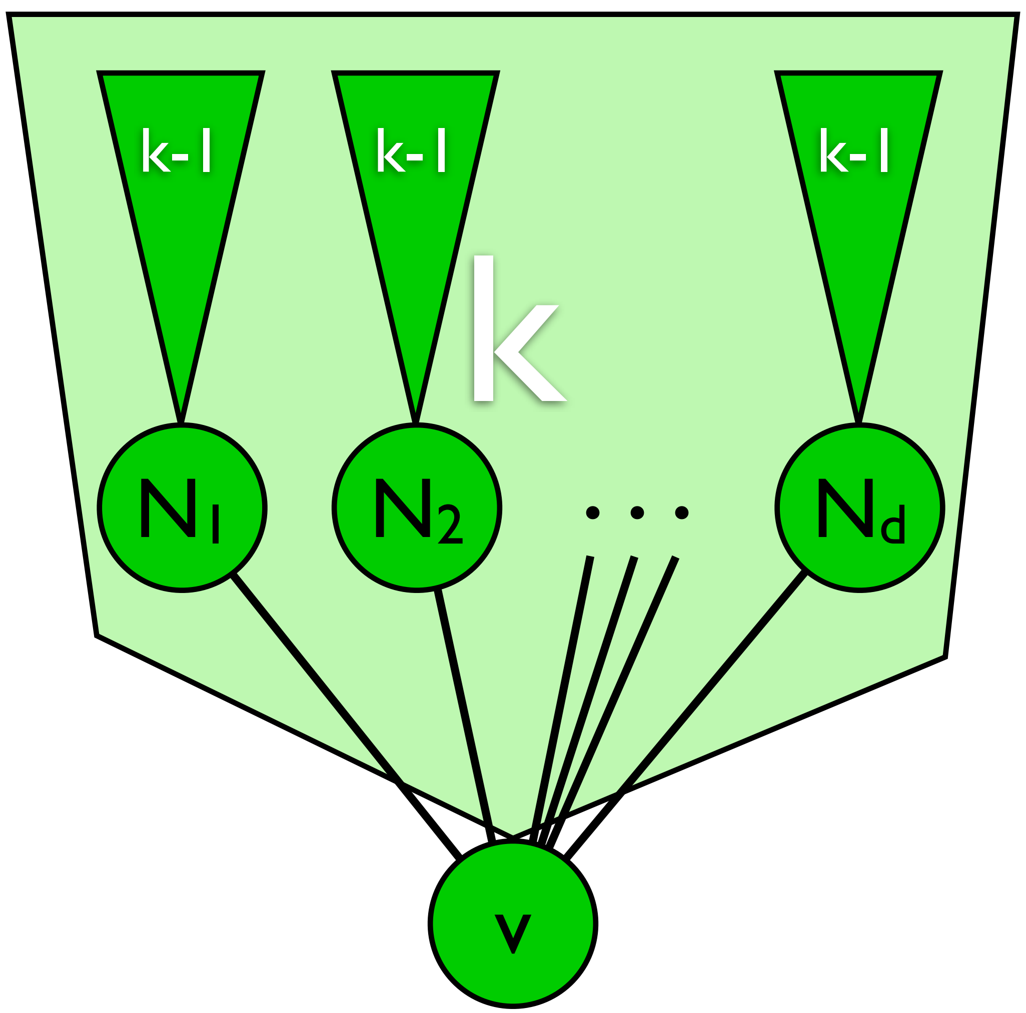
  ⇒ collection of all paths

- level-1-view:
  $v^1 = (L_1, L_2, ..., L_d)$

- level-k-view:

Vertices as Viewpoints

view



- Capture information a robot can collect about v

  ⇒ *view* from v

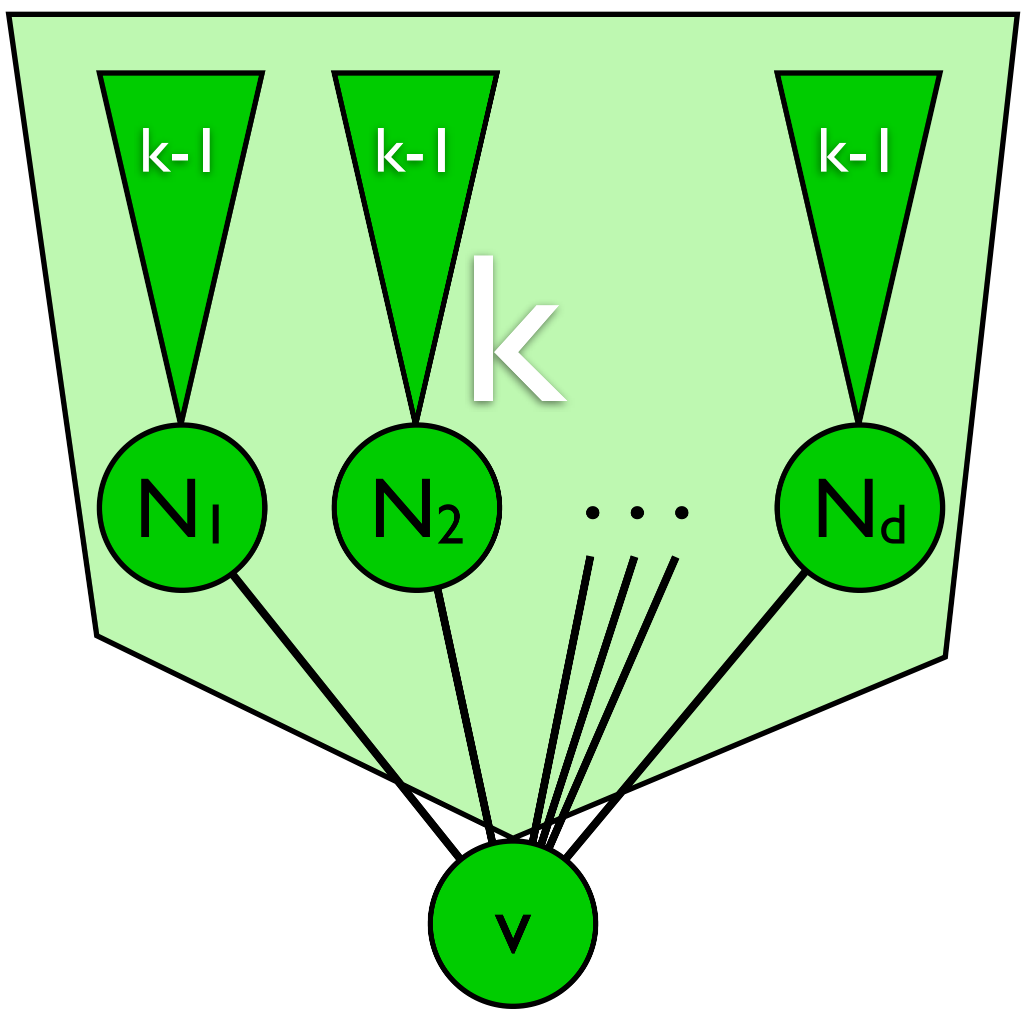
  ⇒ collection of all paths

- level-1-view:
  $v^1 = (L_1, L_2, ..., L_d)$

- level-k-view:

Vertices as Viewpoints

view



- Capture information a robot can collect about v

  ⇒ *view* from v

  ⇒ collection of all paths

- level-1-view:
  $v^1 = (L_1, L_2, \ldots, L_d)$

- level-k-view:
  $v^k = (N_1^{k-1}, N_2^{k-1}, \ldots, N_d^{k-1})$

Vertices as Viewpoints

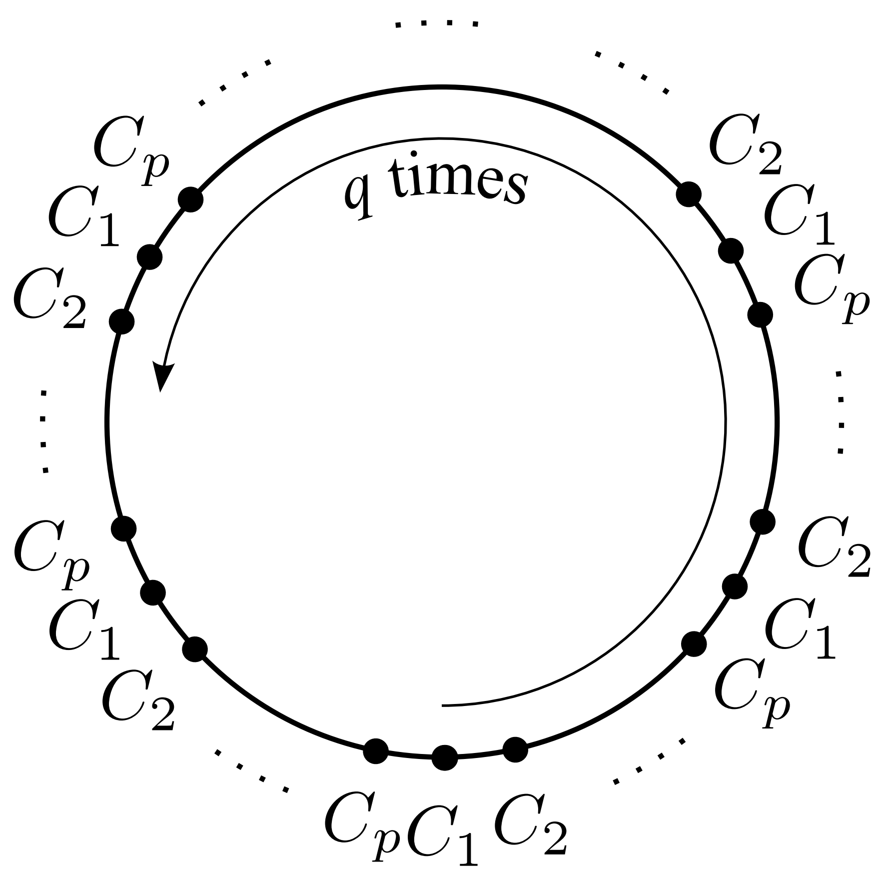
classes

Vertices as Viewpoints

classes

- group all vertices with same v^∞ into classes C_i

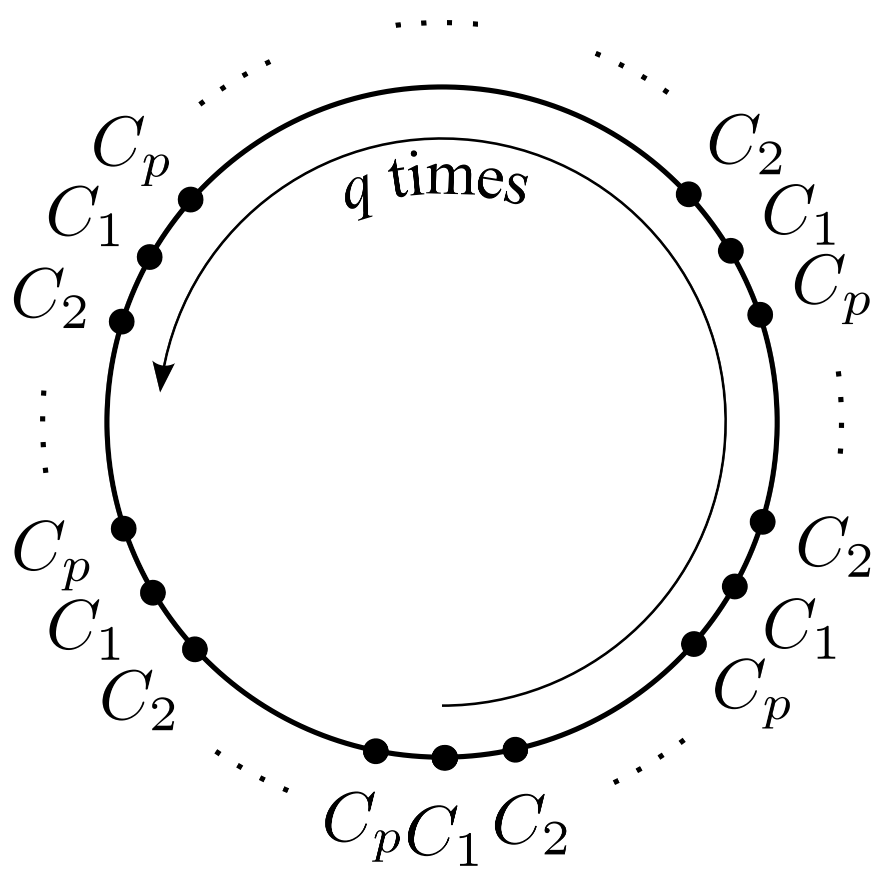
Vertices as Viewpoints
classes



- group all vertices with same v^{∞} into classes C_i
  ⇒periodic on boundary

Vertices as Viewpoints

classes



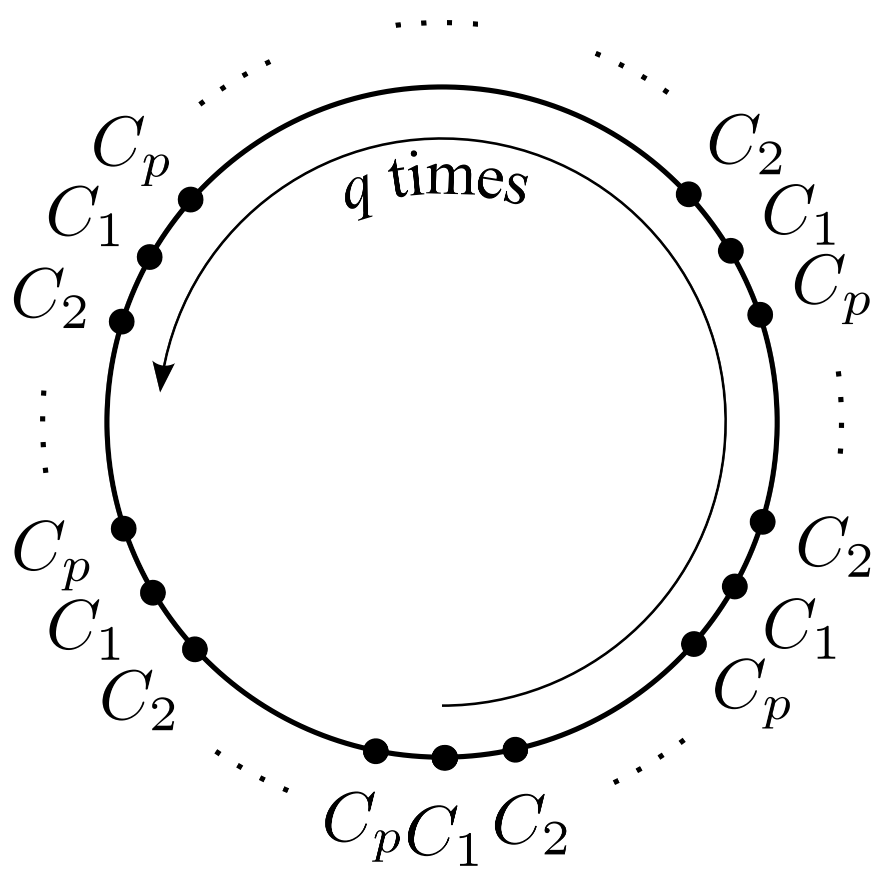
- group all vertices with same v^∞ into classes C_i

  $\Rightarrow$ periodic on boundary

  $\Rightarrow |C_i| = |C_j| \ \forall i,j$

Vertices as Viewpoints

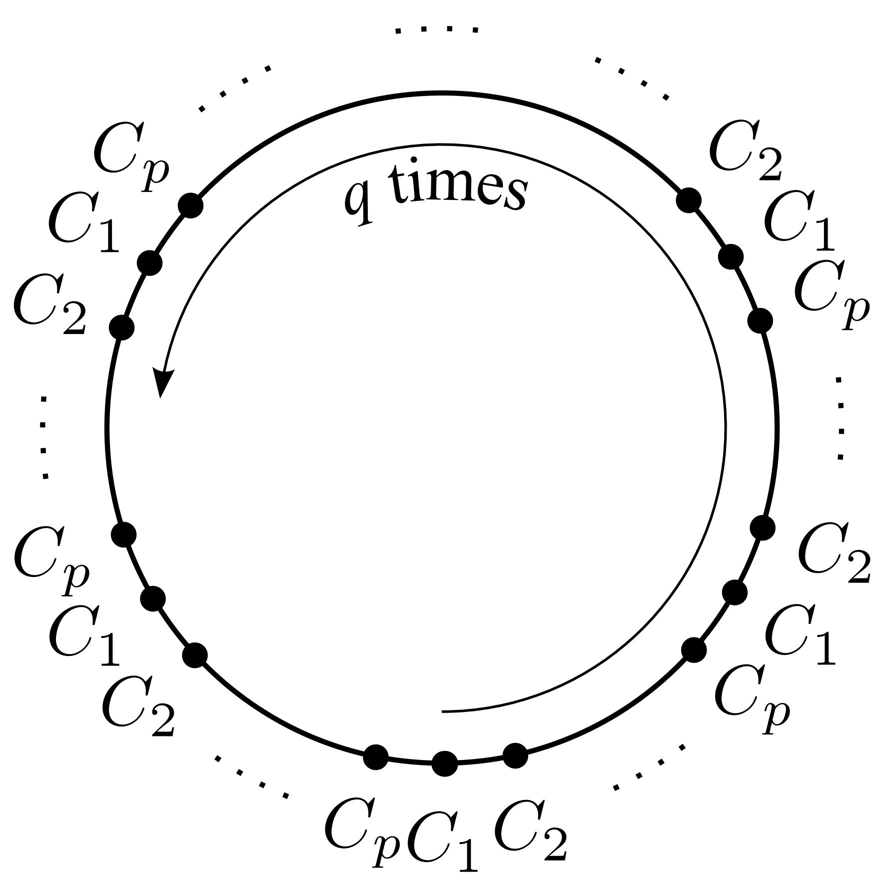
classes



- group all vertices with same v^{∞} into classes C_i
  ⇒periodic on boundary
  ⇒$|C_i| = |C_j|\ \forall i,j$
- $|C_i| = 1$: distinguishable ✓

Vertices as Viewpoints

classes

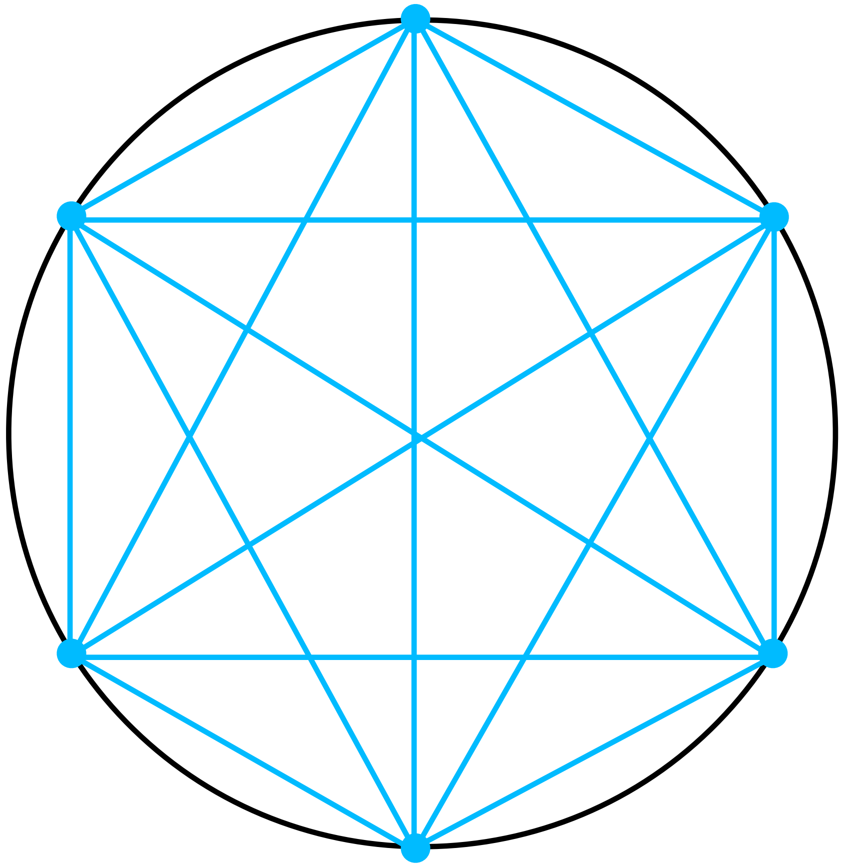


- group all vertices with same v^∞ into classes C_i
  $\Rightarrow$ periodic on boundary
  $\Rightarrow |C_i| = |C_j| \; \forall i,j$
- $|C_i| = 1$: distinguishable ✓
- Norris95: v^{n-1} is enough! (same resulting classes)

The Class C^*

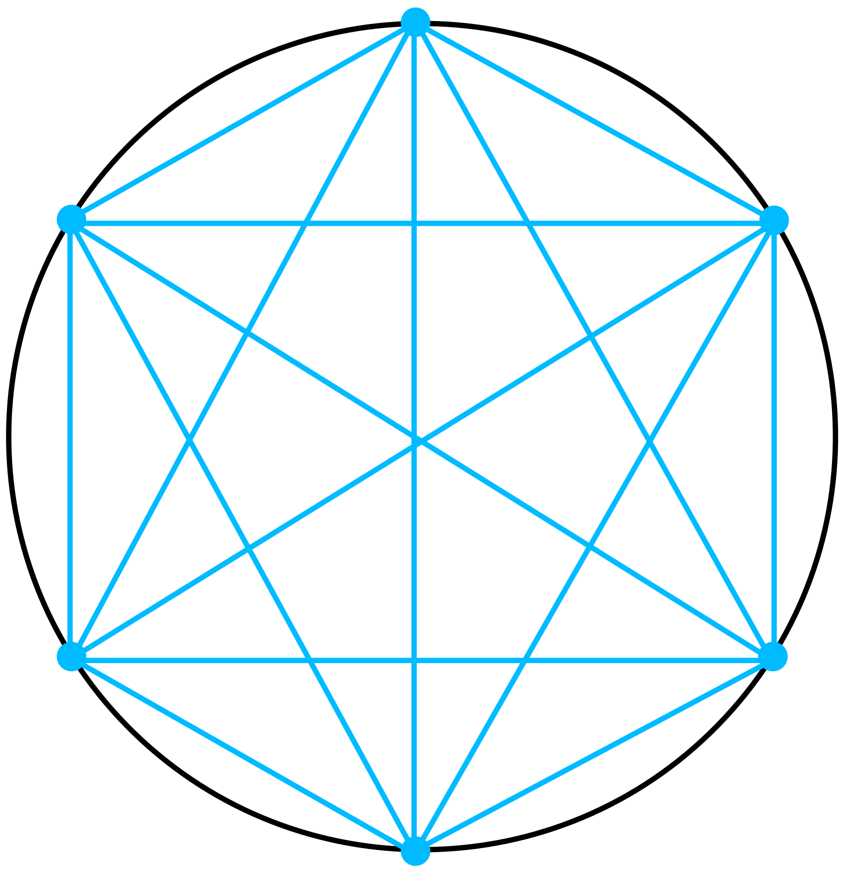
The Class C*

definition



The Class C^*

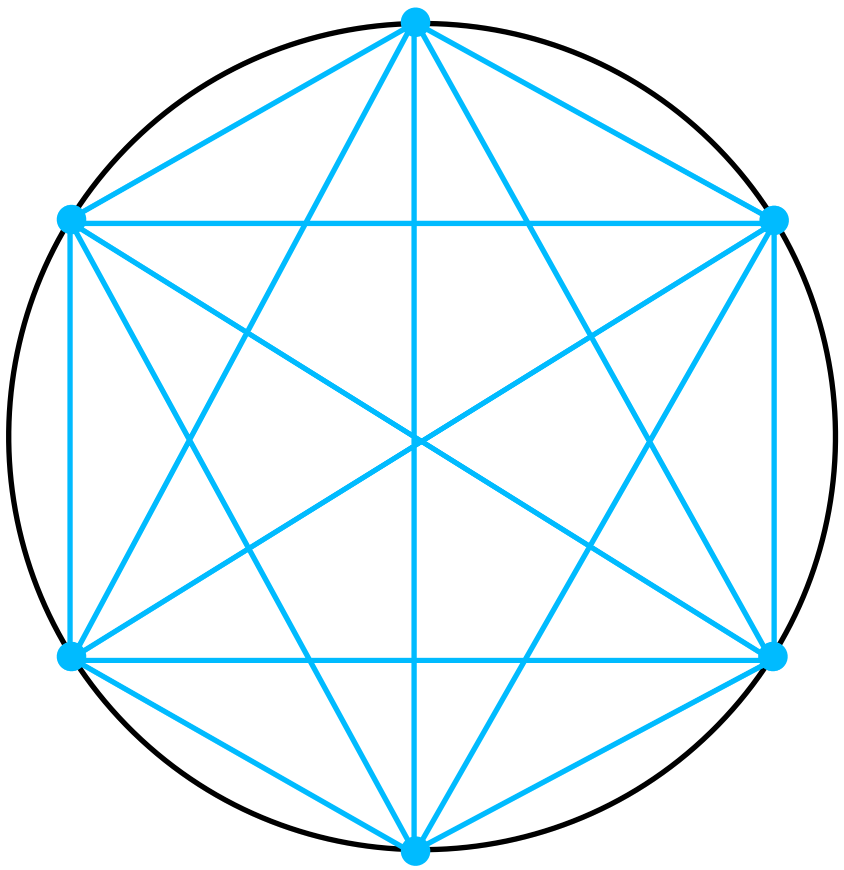
definition



- C^* is the lexicographically smallest class that forms a clique

The Class C^*

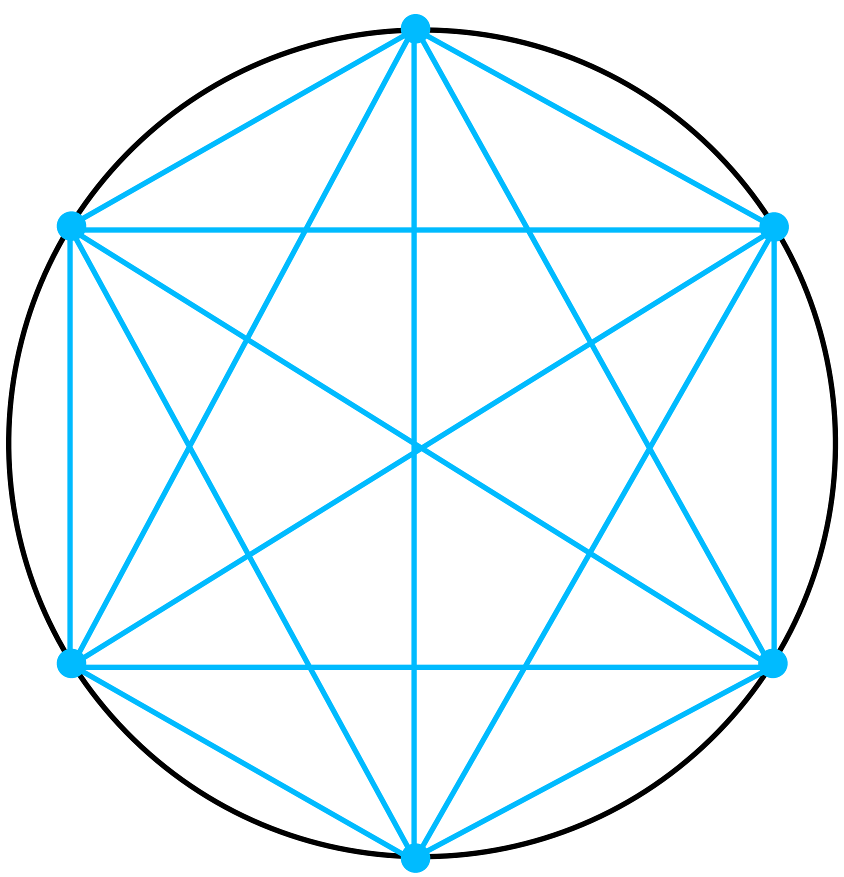
definition



- C^* is the lexicographically smallest class that forms a clique
- Will show:
  Every polygon has a class that forms a clique (!)

The Class C^*

definition

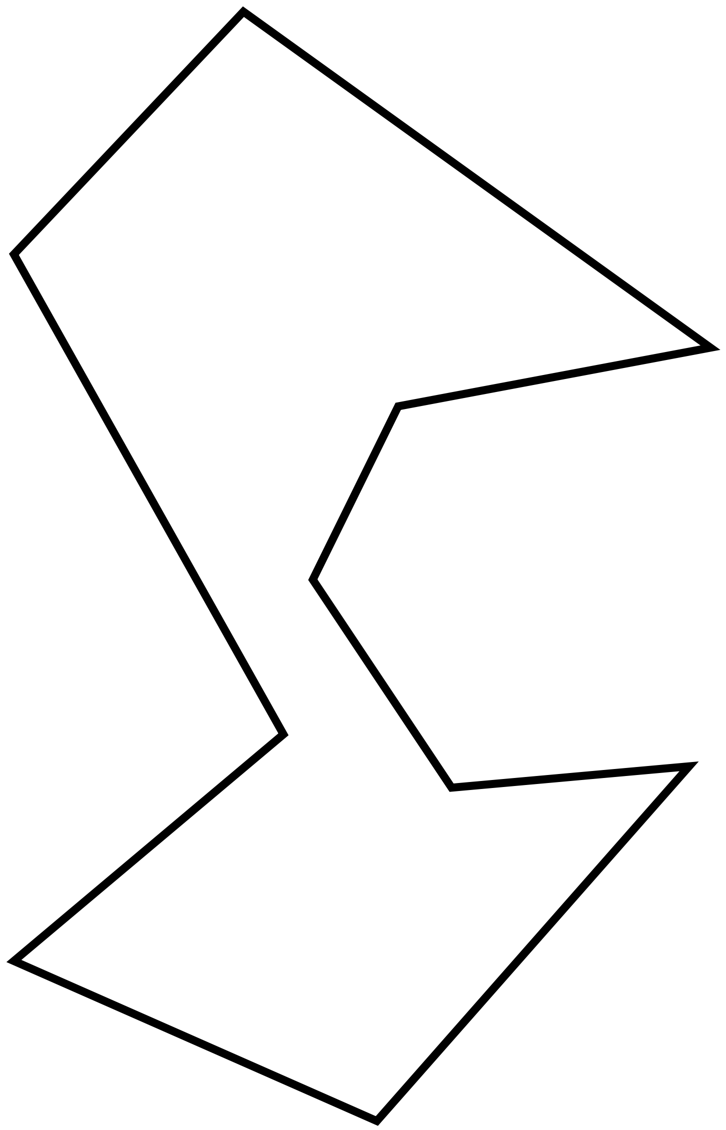


- C^* is the lexicographically smallest class that forms a clique
- Will show:
  Every polygon has a class that forms a clique (!)

$\Rightarrow C^*$ is well defined, unique

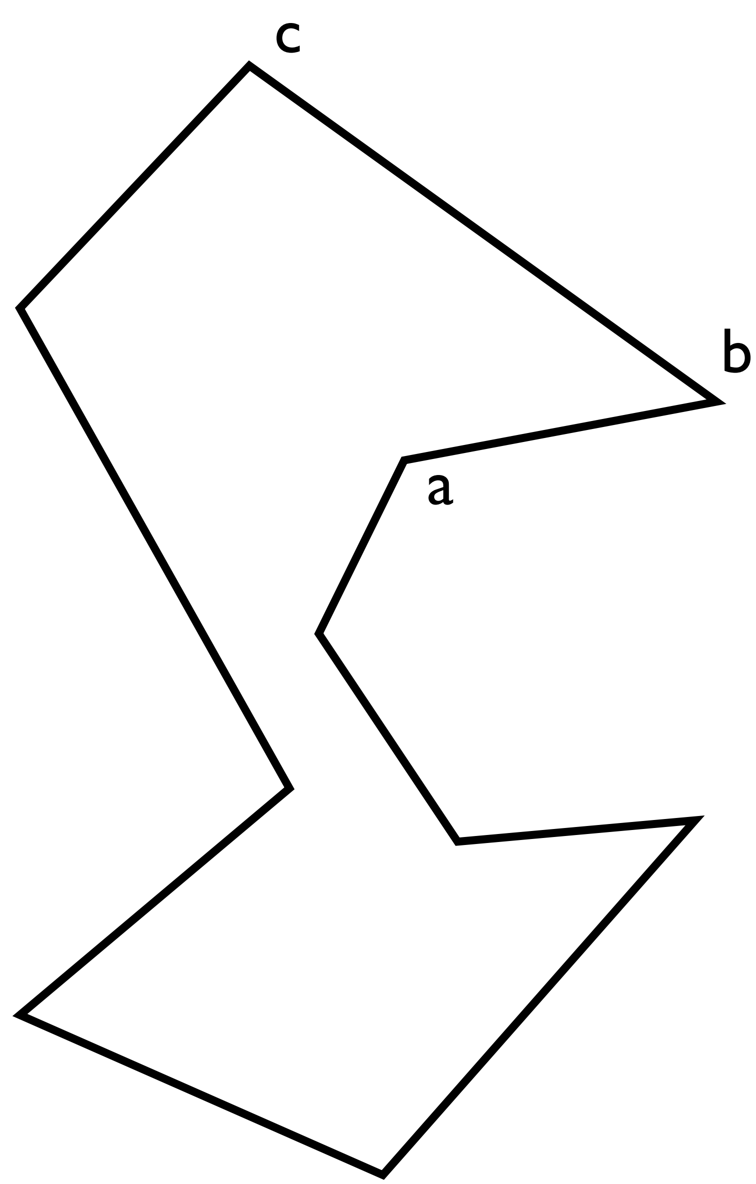
The Class C^*

ears



The Class C^*

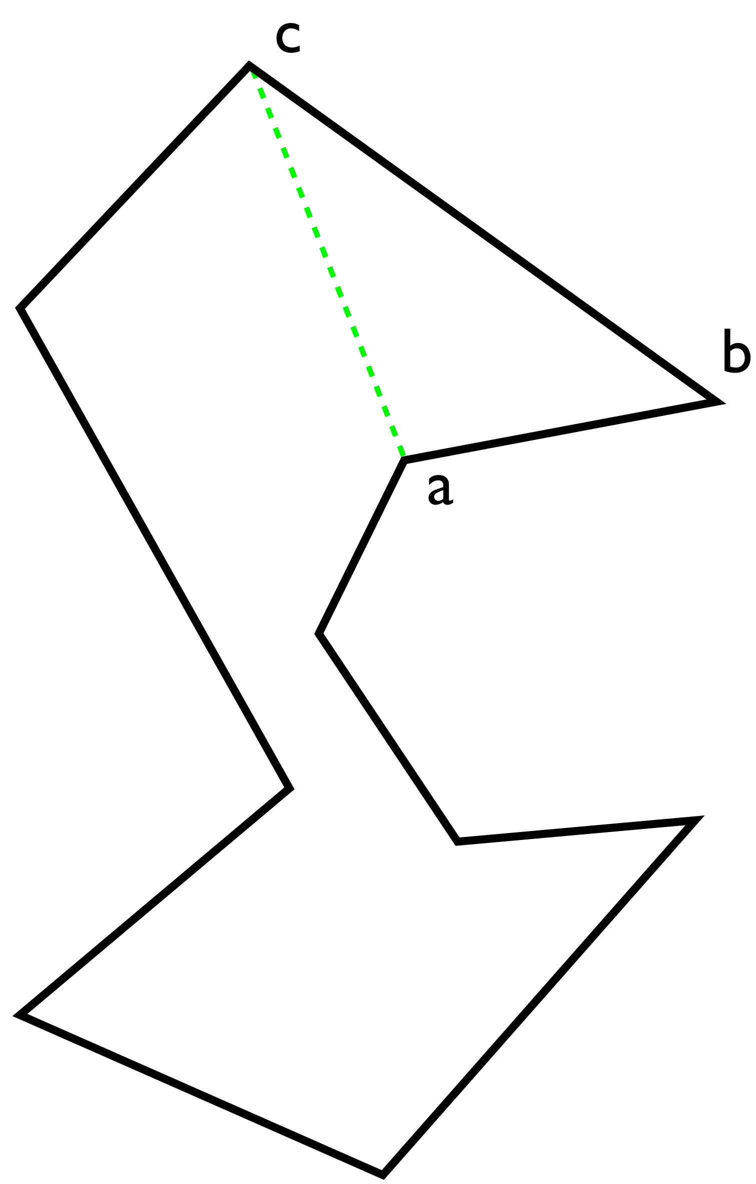
ears



- Let a,b,c be a sequence of vertices on the boundary

The Class C^*

ears

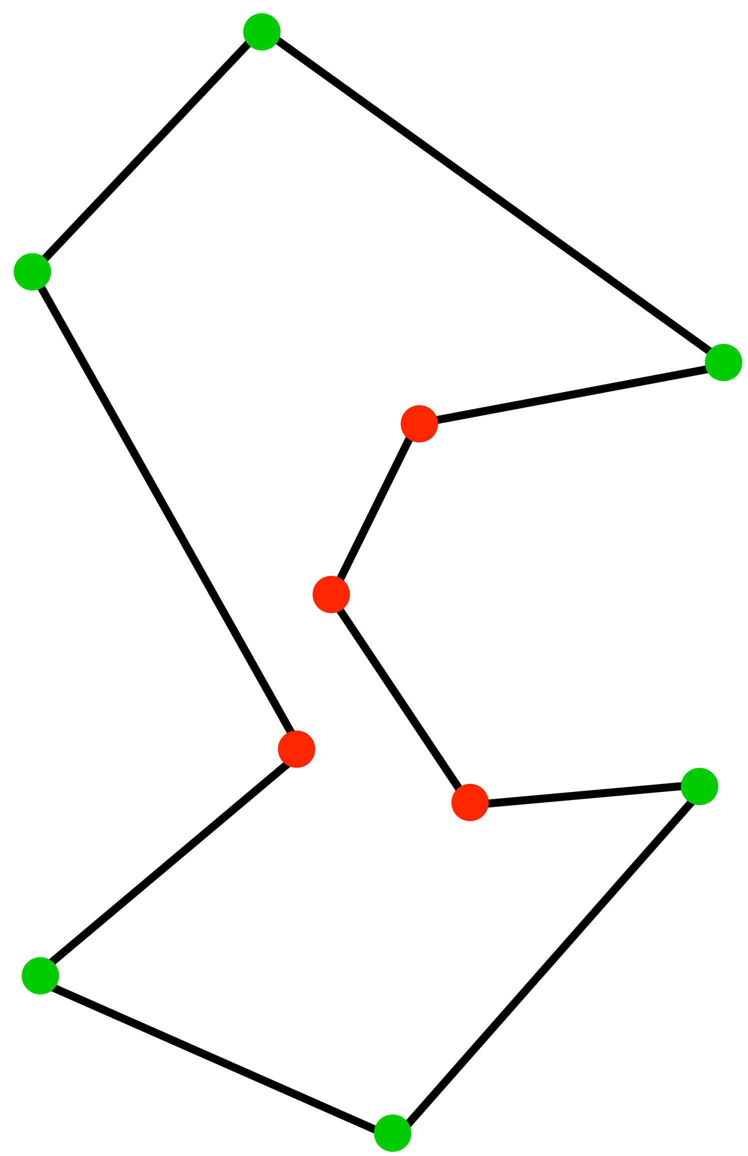


- Let a,b,c be a sequence of vertices on the boundary

  ⇒b is an *ear*, iff a sees c

The Class C^*

ears

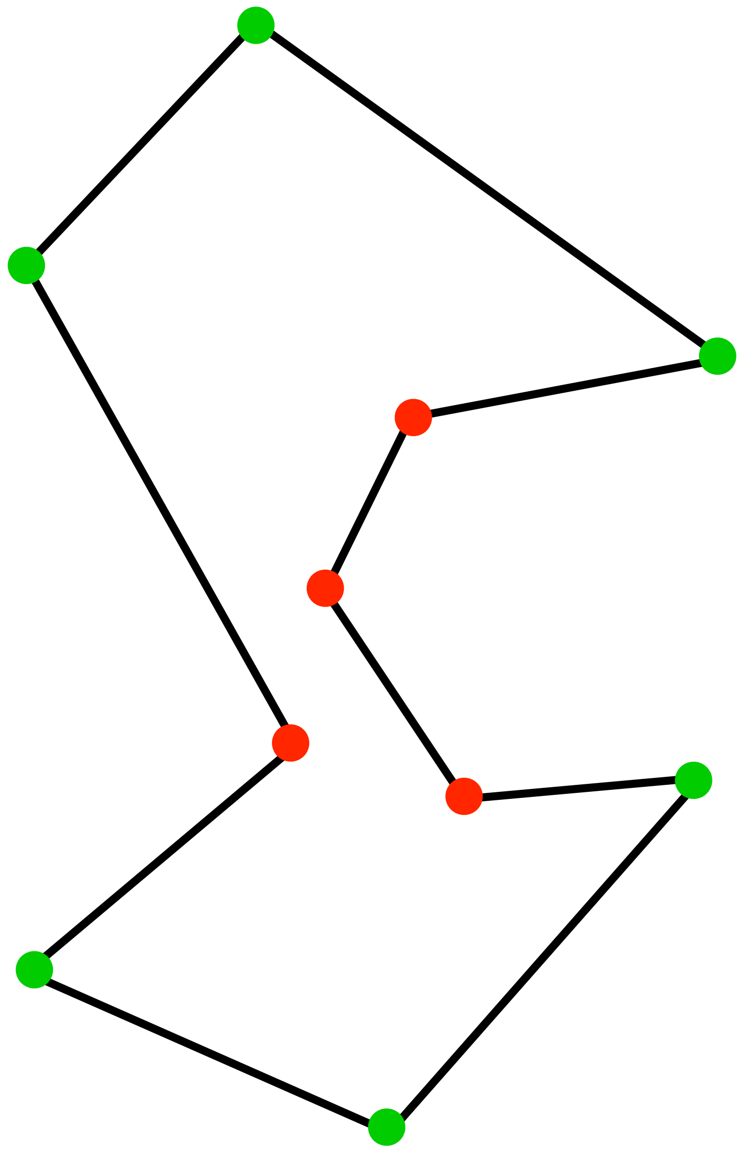


- Let a,b,c be a sequence of vertices on the boundary

  ⇒b is an *ear*, iff a sees c

The Class C*

ears



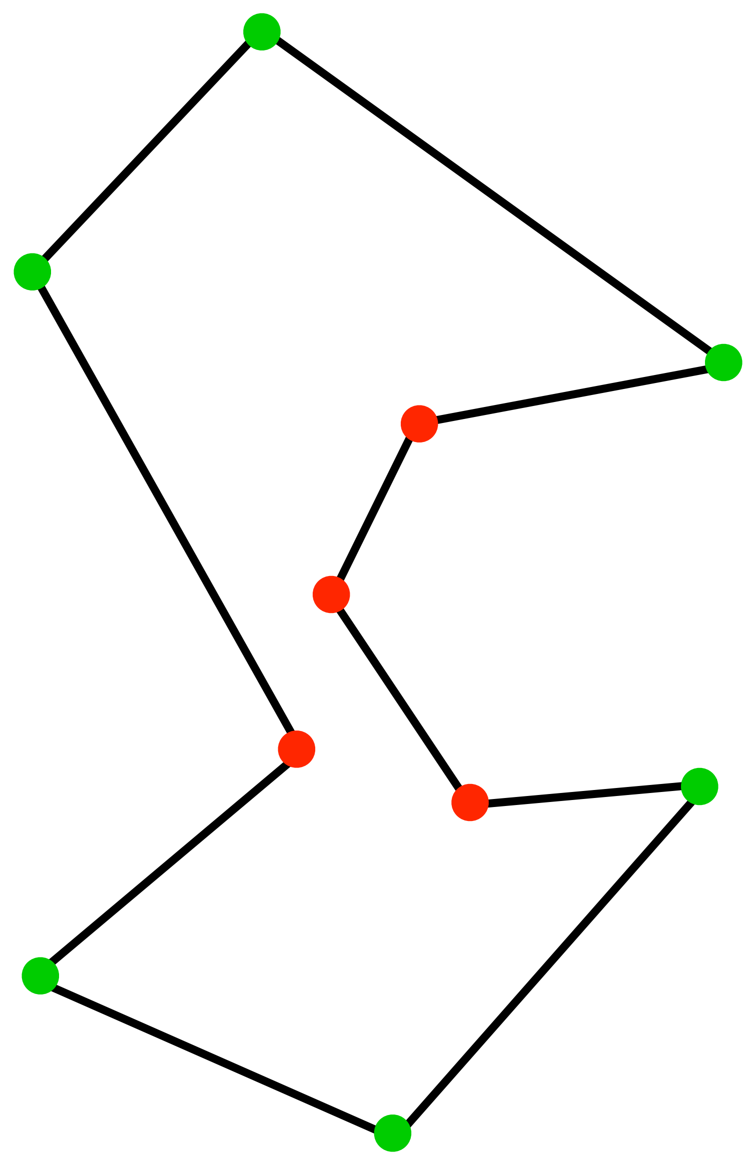
- Let a,b,c be a sequence of vertices on the boundary

  ⇒b is an *ear*, iff a sees c

- b is an ear, iff the move -1, 2, look back yields "-2"

The Class C*

ears



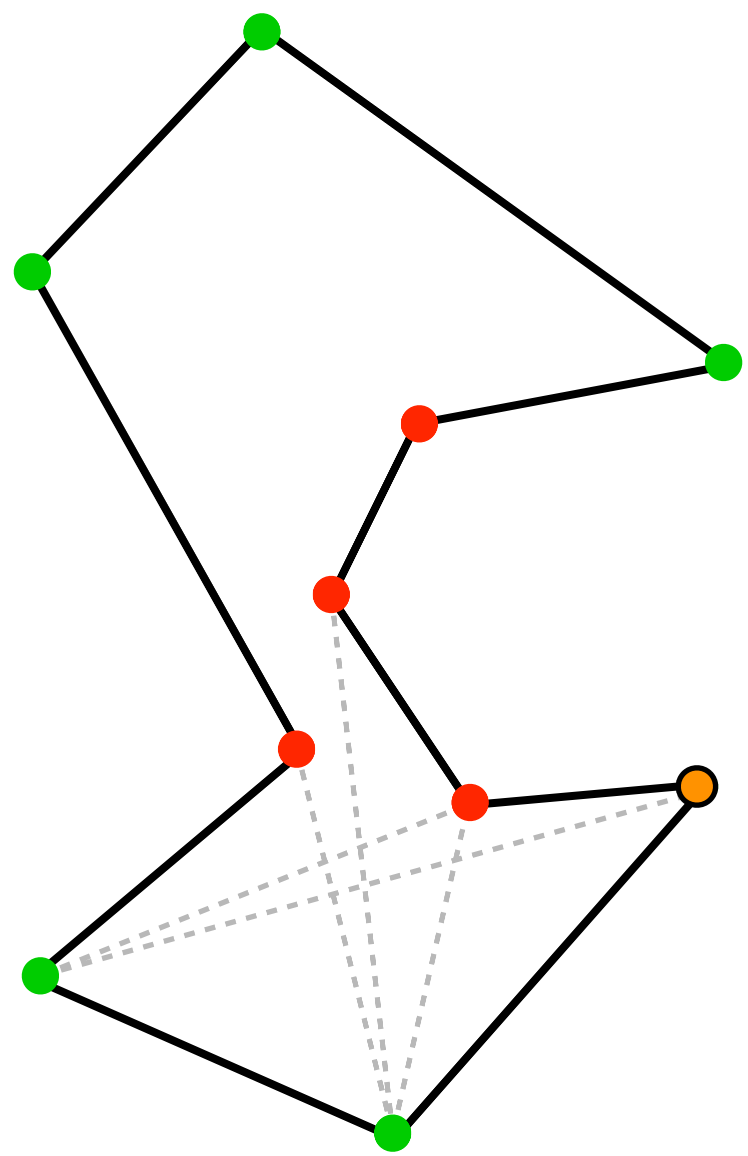
- Let a,b,c be a sequence of vertices on the boundary

  ⇒b is an *ear*, iff a sees c

- b is an ear, iff the move -1, 2, look back yields "-2"

The Class C^*

ears



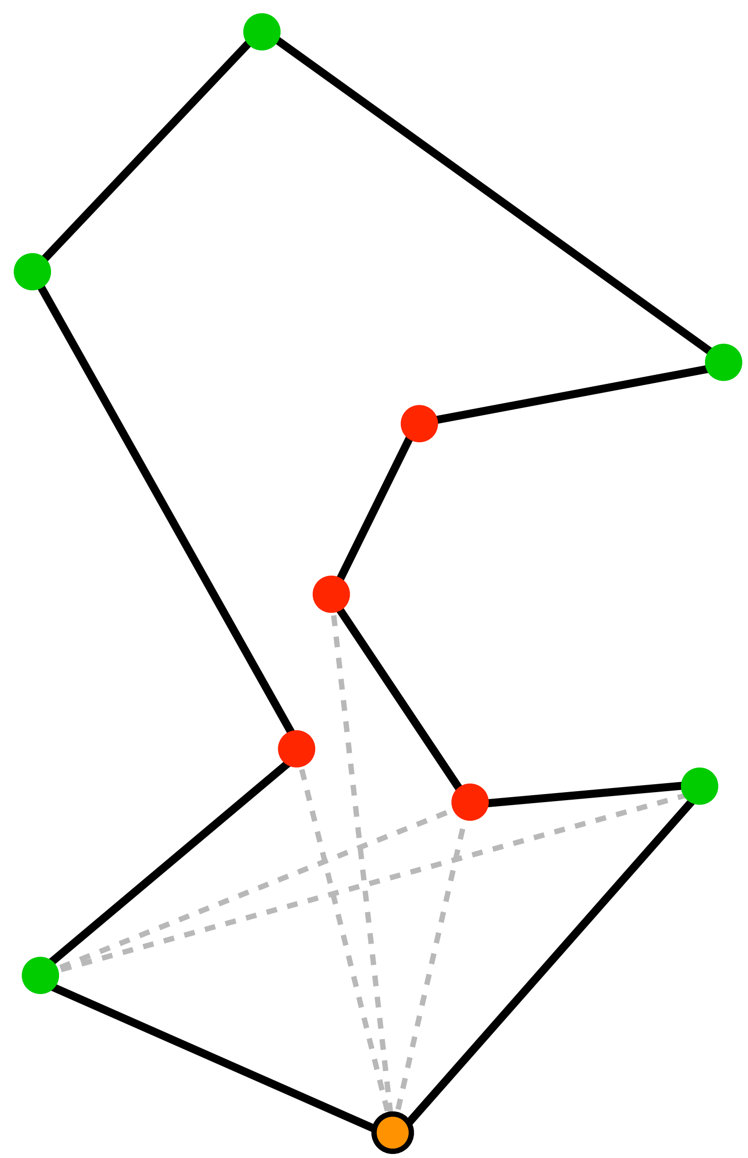
- Let a,b,c be a sequence of vertices on the boundary

  ⇒b is an *ear*, iff a sees c

- b is an ear, iff the move -1, 2, look back yields "-2"

The Class C*

ears



- Let a,b,c be a sequence of vertices on the boundary

  ⇒b is an *ear*, iff a sees c

- b is an ear, iff the move -1, 2, look back yields '-2'

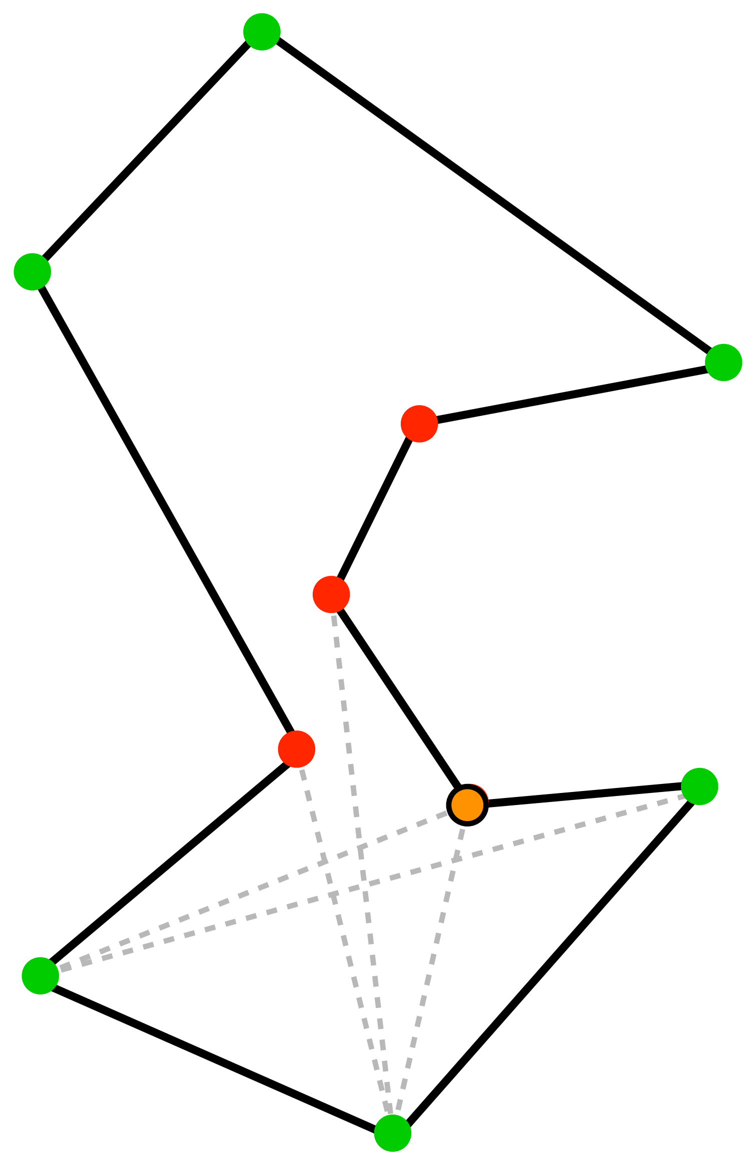
first left neighbor

second right neighbor

second left neighbor

The Class C*

ears



- Let a,b,c be a sequence of vertices on the boundary

  ⇒b is an *ear*, iff a sees c

- b is an ear, iff the move -1, 2, look back yields "-2"

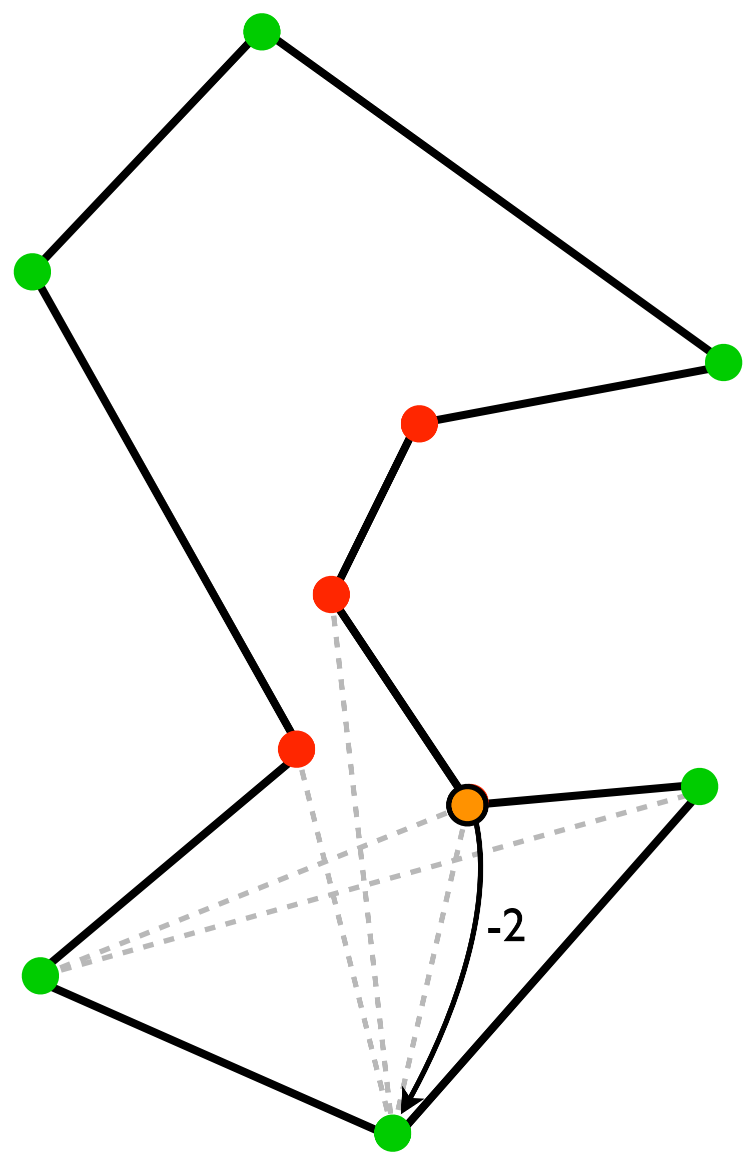
first left neighbor

second right neighbor

second left neighbor

The Class C*

ears



- Let a,b,c be a sequence of vertices on the boundary

  ⇒b is an *ear*, iff a sees c

- b is an ear, iff the move -1, 2, look back yields "-2"

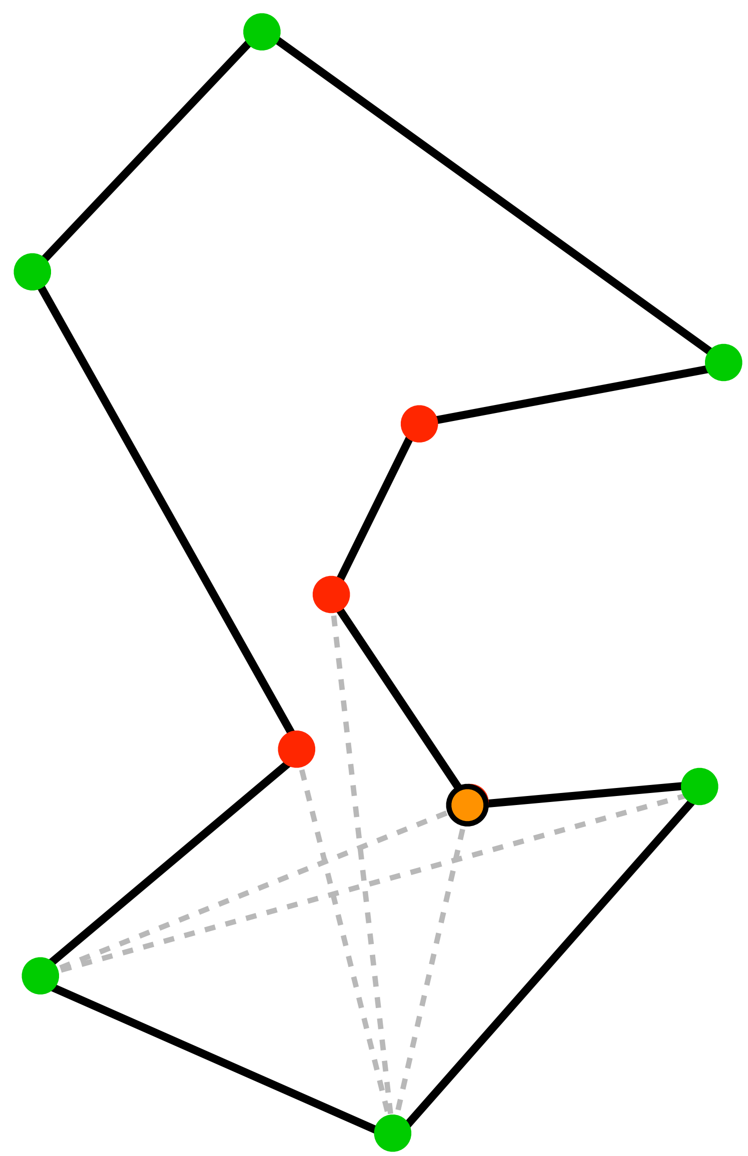
first left neighbor

second right neighbor

second left neighbor

The Class C*

ears



- Let a,b,c be a sequence of vertices on the boundary

  ⇒b is an *ear*, iff a sees c

- b is an ear, iff the move -1, 2, look back yields "-2"

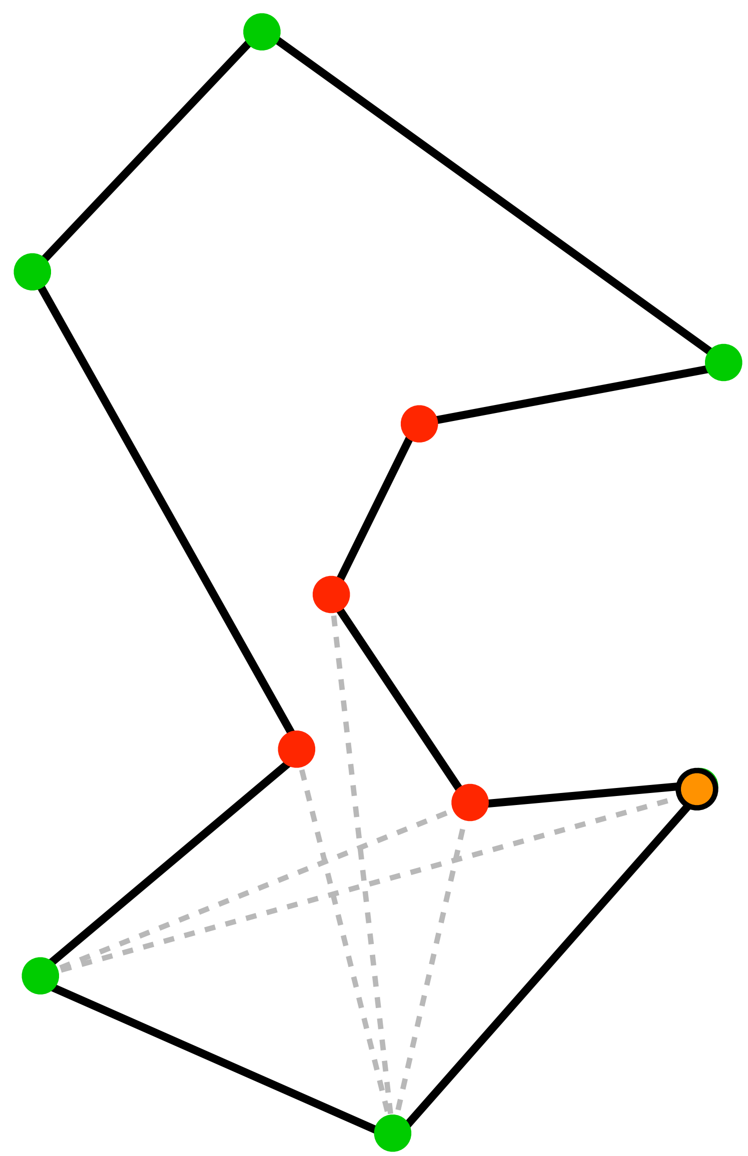
first left neighbor

second right neighbor

second left neighbor

The Class C^*

ears



- Let a,b,c be a sequence of vertices on the boundary

  ⇒b is an *ear*, iff a sees c

- b is an ear, iff the move -1, 2, look back yields "-2"

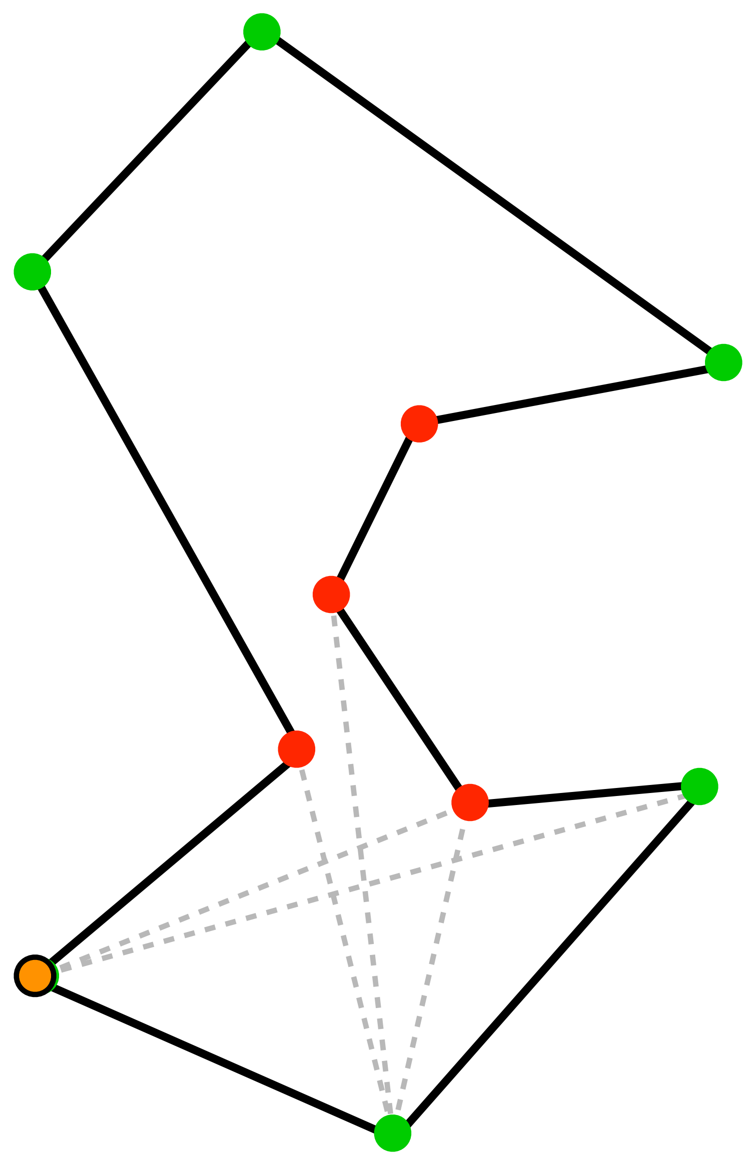
first left neighbor

second right neighbor

second left neighbor

The Class C^*

ears



- Let a,b,c be a sequence of vertices on the boundary

  ⇒b is an *ear*, iff a sees c

- b is an ear, iff the move -1, 2, look back yields "-2"

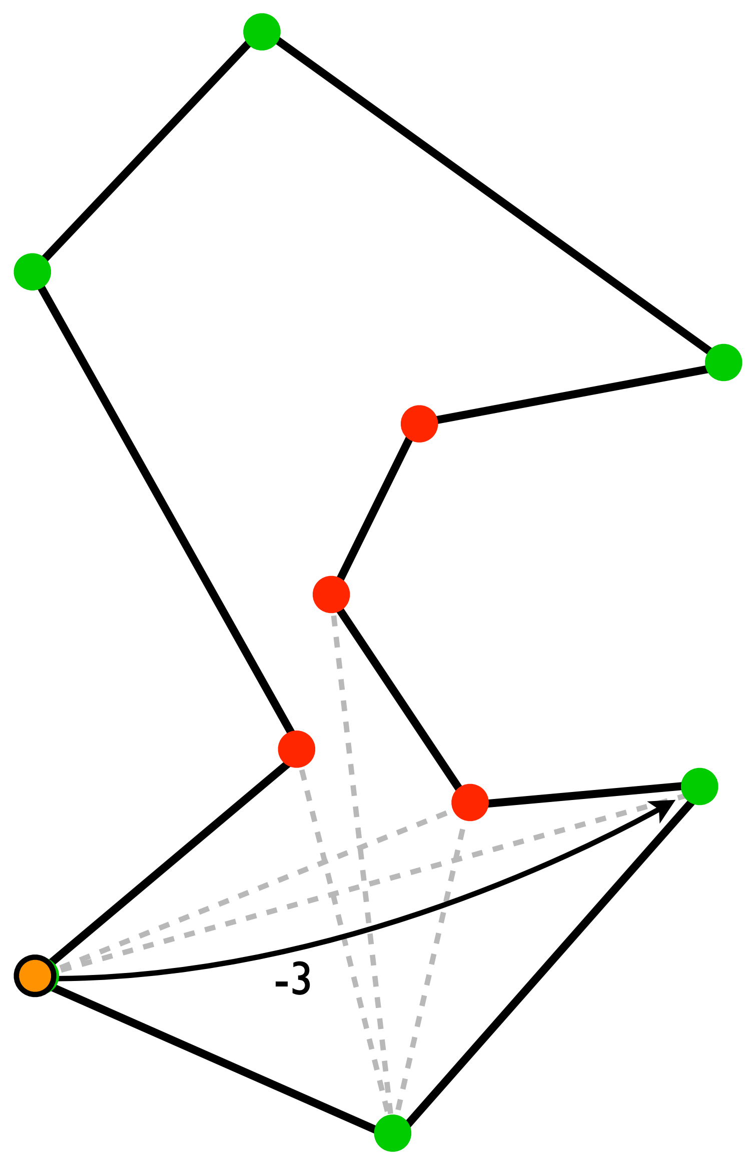
first left neighbor

second right neighbor

second left neighbor

The Class C*

ears



- Let a,b,c be a sequence of vertices on the boundary

  ⇒b is an *ear*, iff a sees c

- b is an ear, iff the move -1, 2, look back yields "-2"

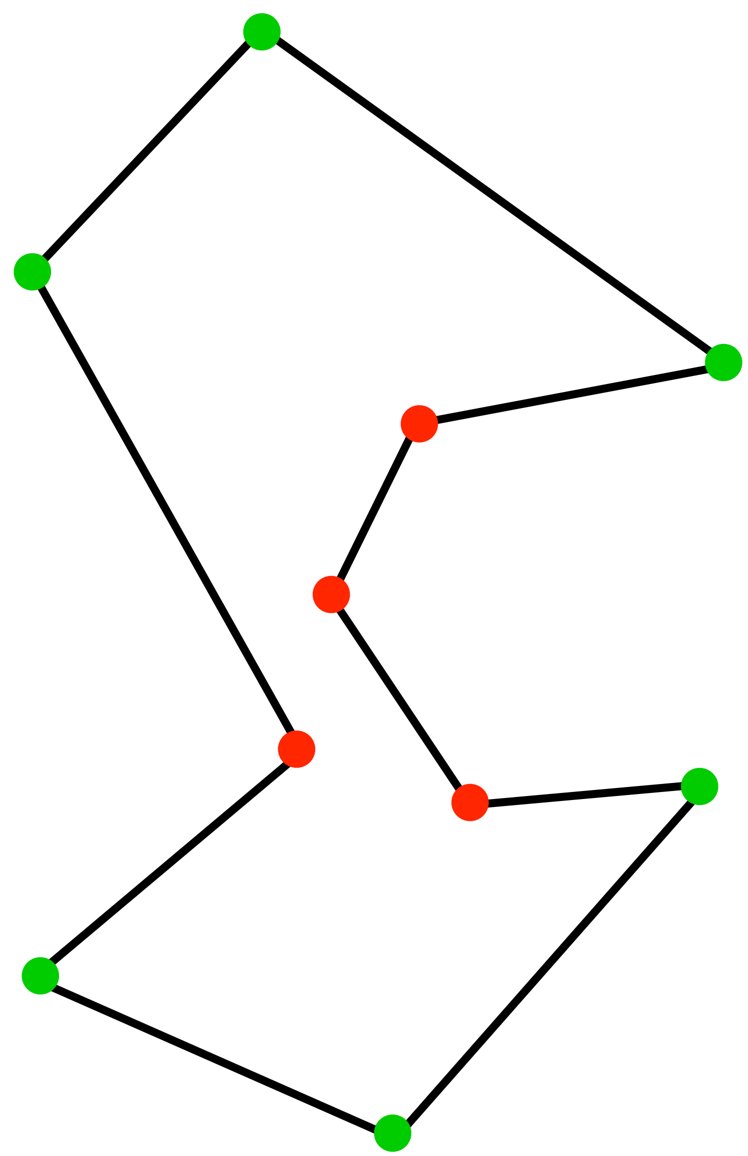
first left neighbor

second right neighbor

second left neighbor

The Class C*

ears



- Let a,b,c be a sequence of vertices on the boundary

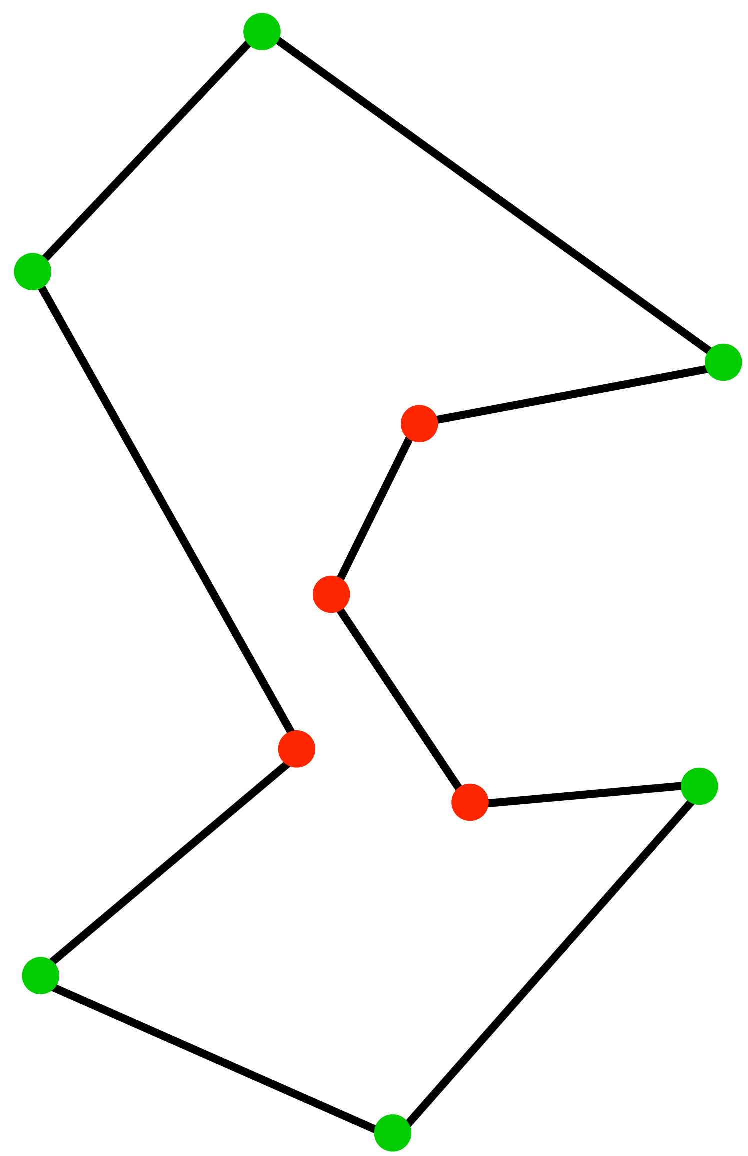
  ⇒b is an *ear*, iff a sees c

- b is an ear, iff the move -1, 2, look back yields "-2"

  ⇒vertices in the same class as an ear are ears

The Class C*

ears



- Let a,b,c be a sequence of vertices on the boundary

  ⇒b is an *ear*, iff a sees c

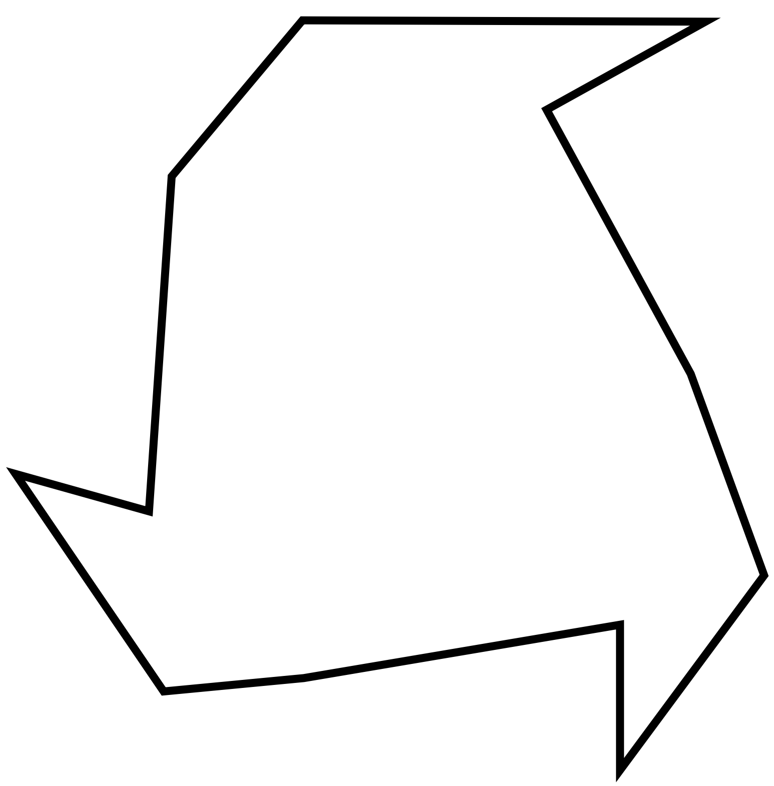
- b is an ear, iff the move -1, 2, look back yields "-2"

  ⇒vertices in the same class as an ear are ears

- Every polygon has an ear

The Class C^*

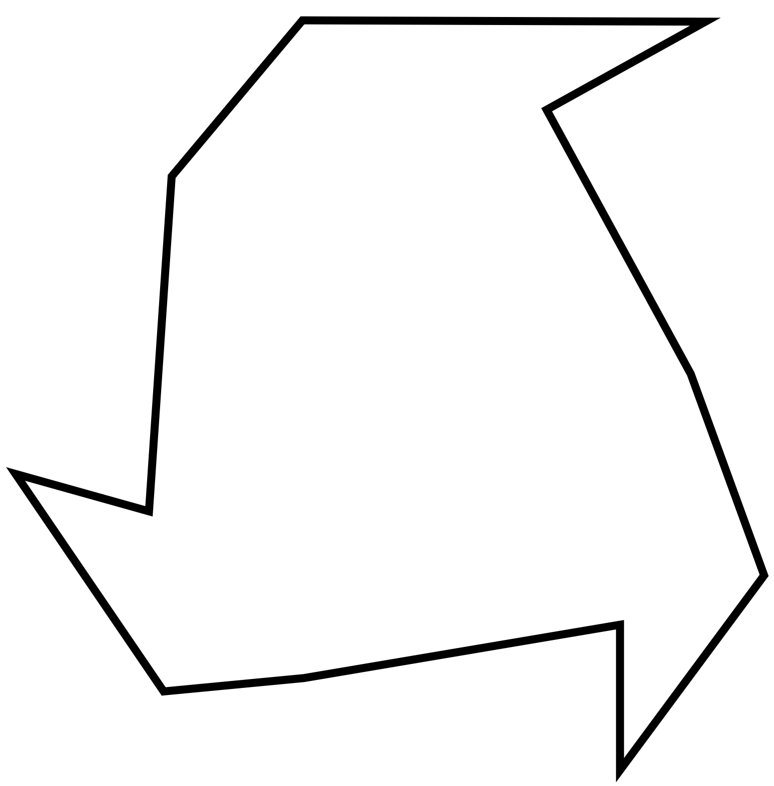
existence of a clique



The Class C*

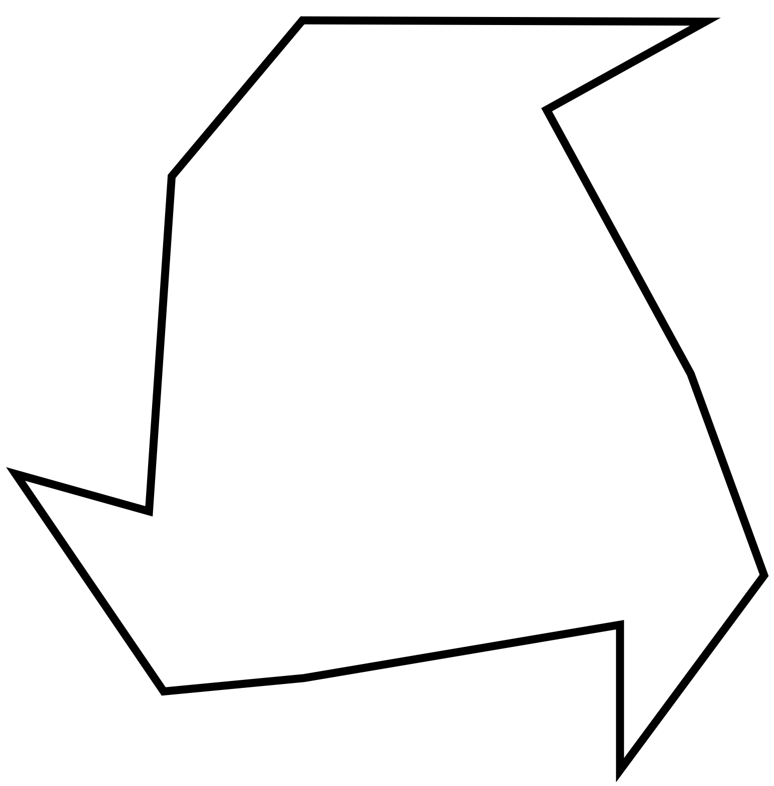
existence of a clique

- Cut ears repeatedly...



The Class C^*

existence of a clique

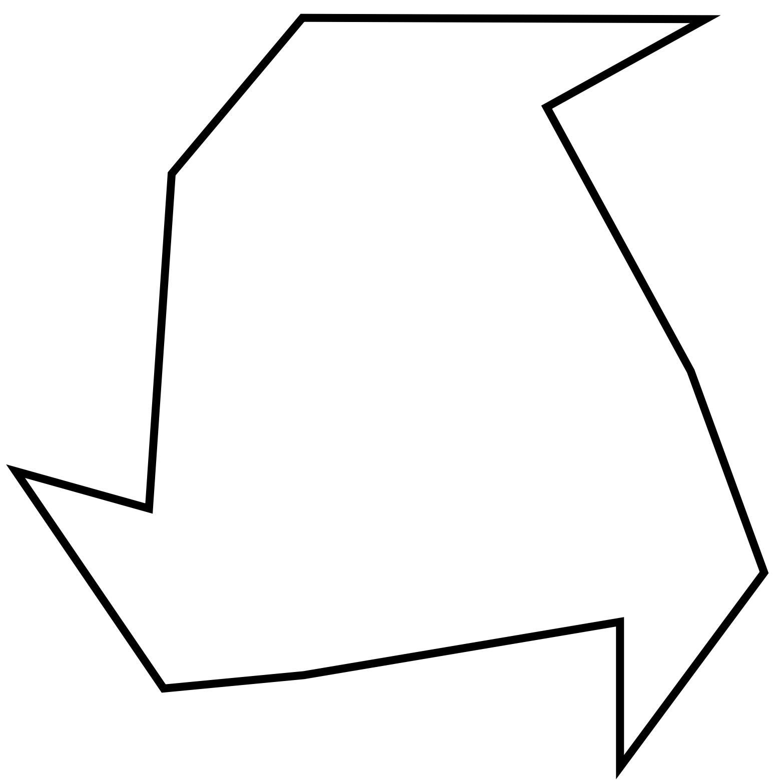


- Cut ears repeatedly...

  $\Rightarrow$cut the entire class

The Class C^*

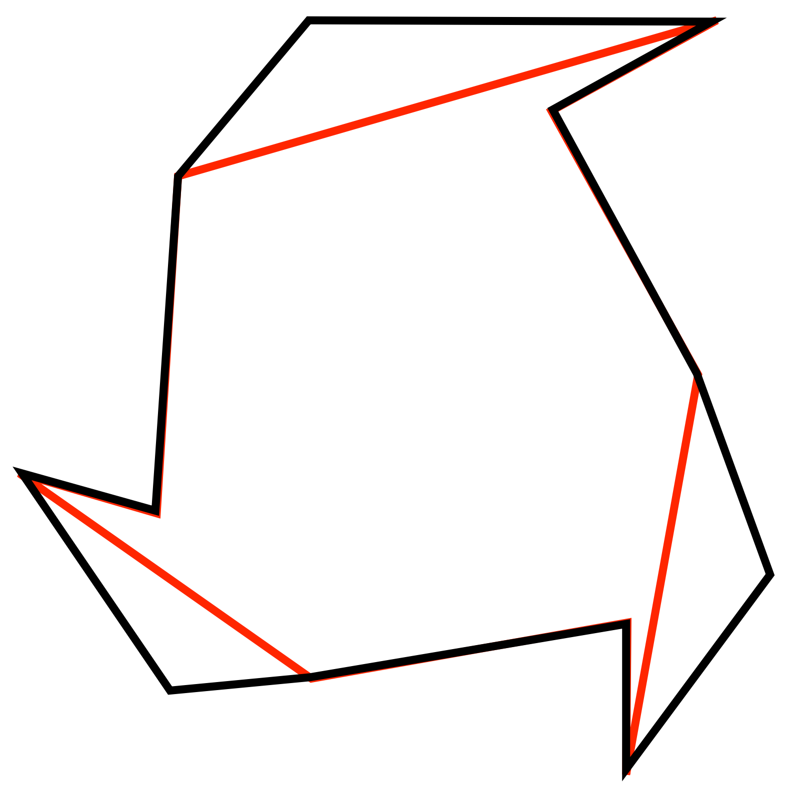
existence of a clique



- Cut ears repeatedly...
  - ⇒cut the entire class
  - ⇒no class will split!

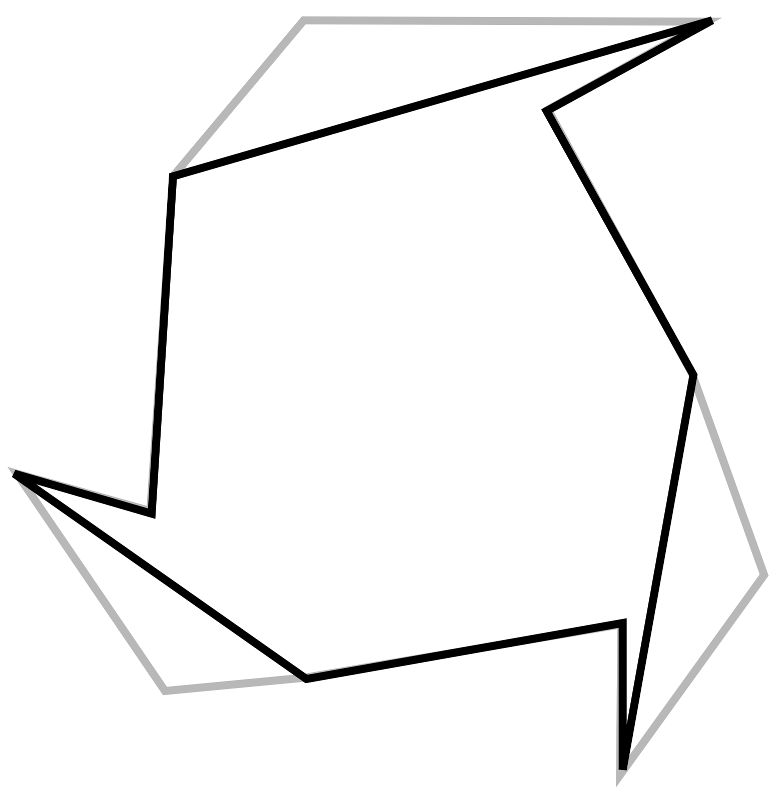
The Class C^*

existence of a clique



- Cut ears repeatedly...
  - ⇒cut the entire class
  - ⇒no class will split!

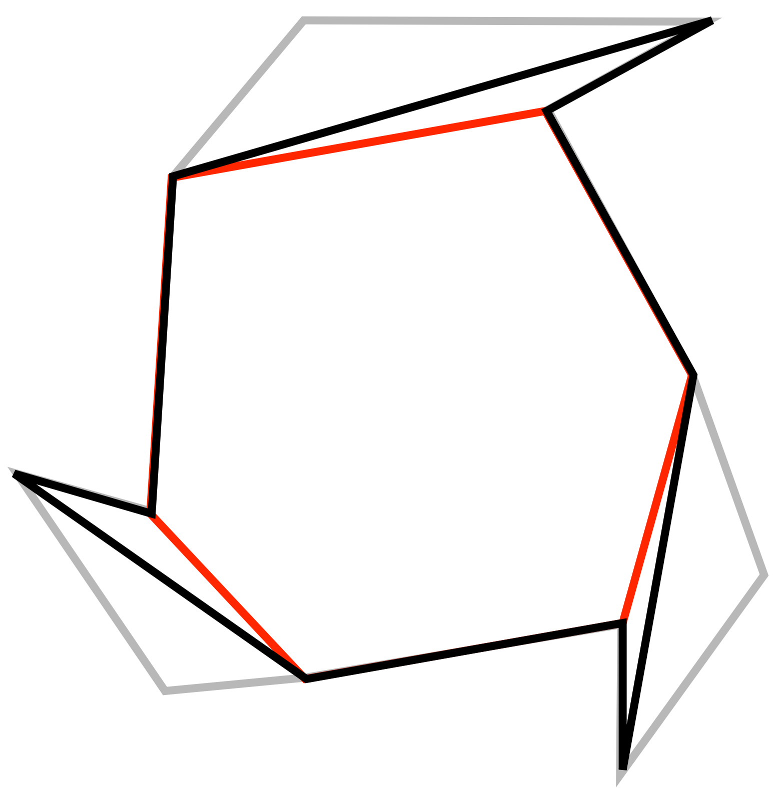
The Class C^*
existence of a clique



- Cut ears repeatedly...
  ⇒cut the entire class
  ⇒no class will split!

The Class C^*

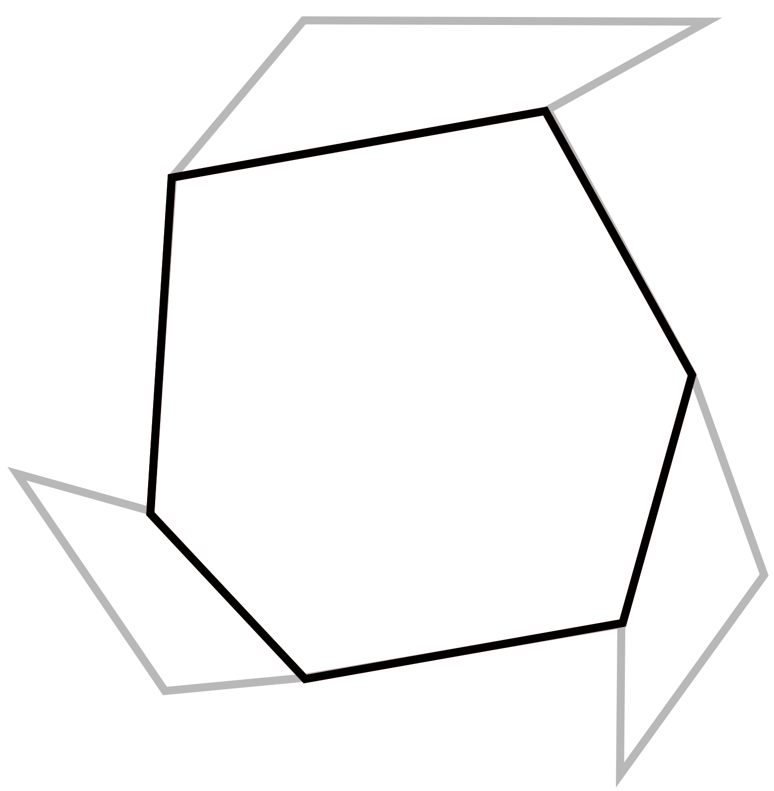
existence of a clique



- Cut ears repeatedly...
  - ⇒cut the entire class
  - ⇒no class will split!

The Class C^*

existence of a clique



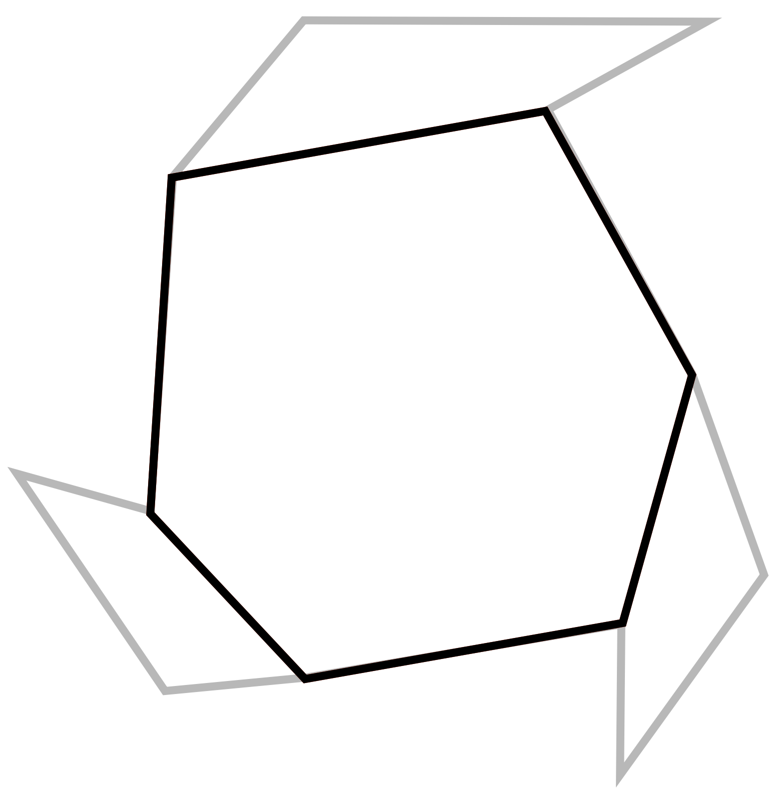
- Cut ears repeatedly...

  ⇒cut the entire class

  ⇒no class will split!

The Class C^*

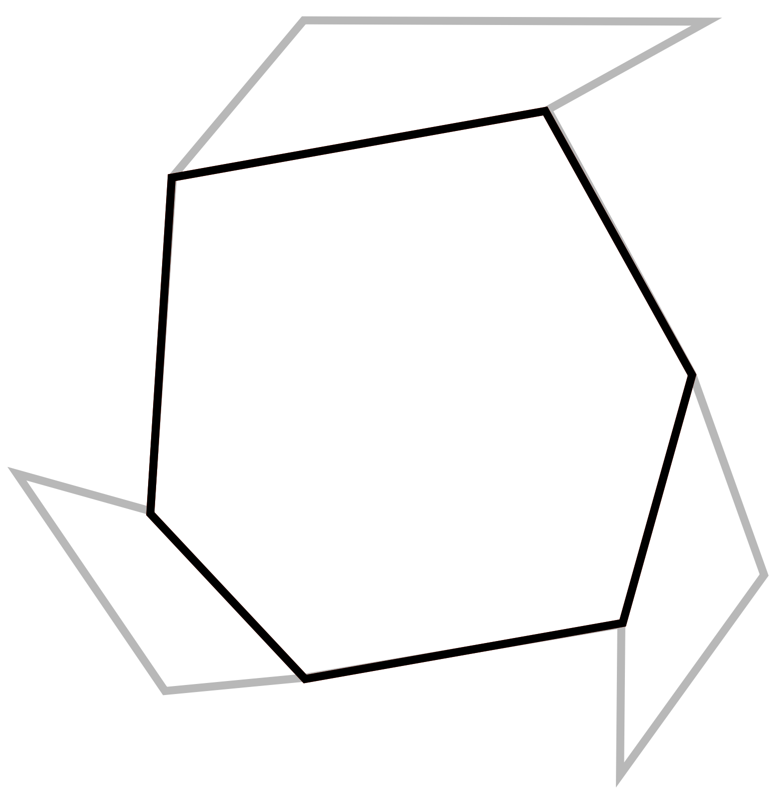
existence of a clique



- Cut ears repeatedly...
  ⇒cut the entire class
  ⇒no class will split!
- ... until only one remains

The Class C^*

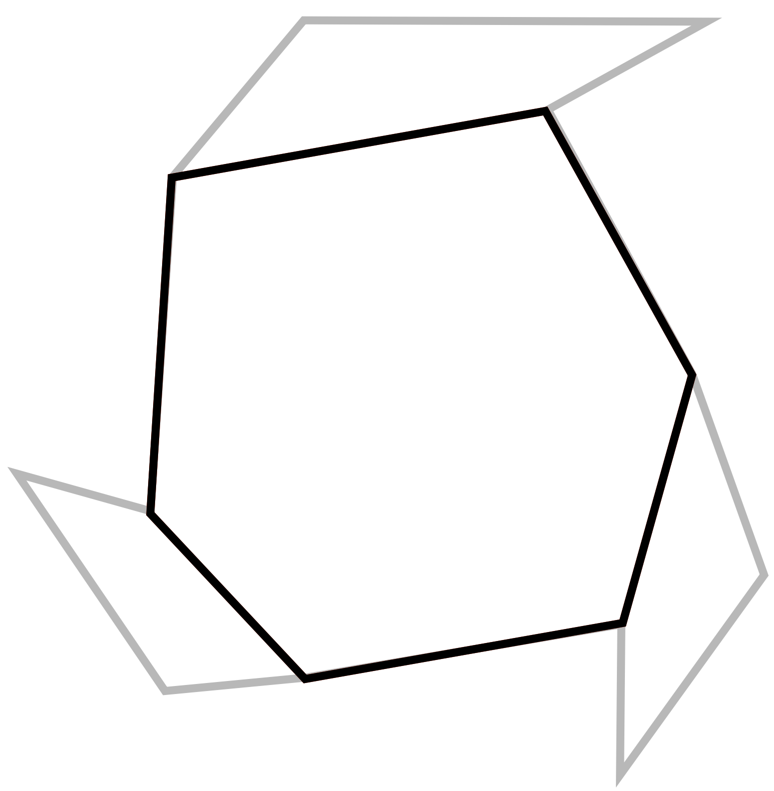
existence of a clique



- Cut ears repeatedly...
  ⇒cut the entire class
  ⇒no class will split!
- ... until only one remains
  ⇒must be a clique!

The Class C^*

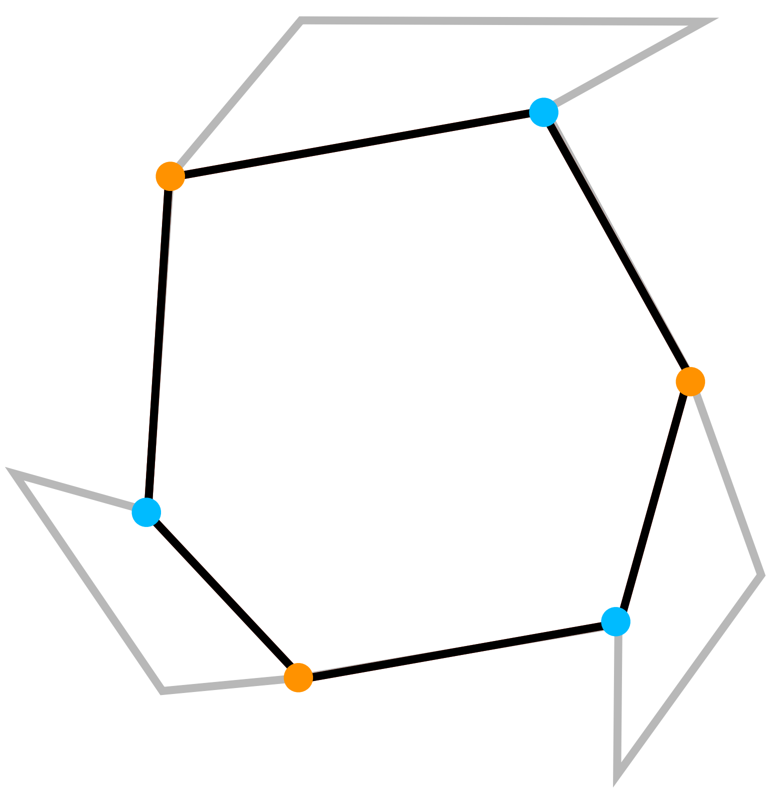
existence of a clique



- Cut ears repeatedly...
  - ⇒cut the entire class
  - ⇒no class will split!
- ... until only one remains
  - ⇒must be a clique!
  - ⇒contains all vertices of some original class

The Class C^*

existence of a clique



- Cut ears repeatedly...

  ⇒cut the entire class

  ⇒no class will split!

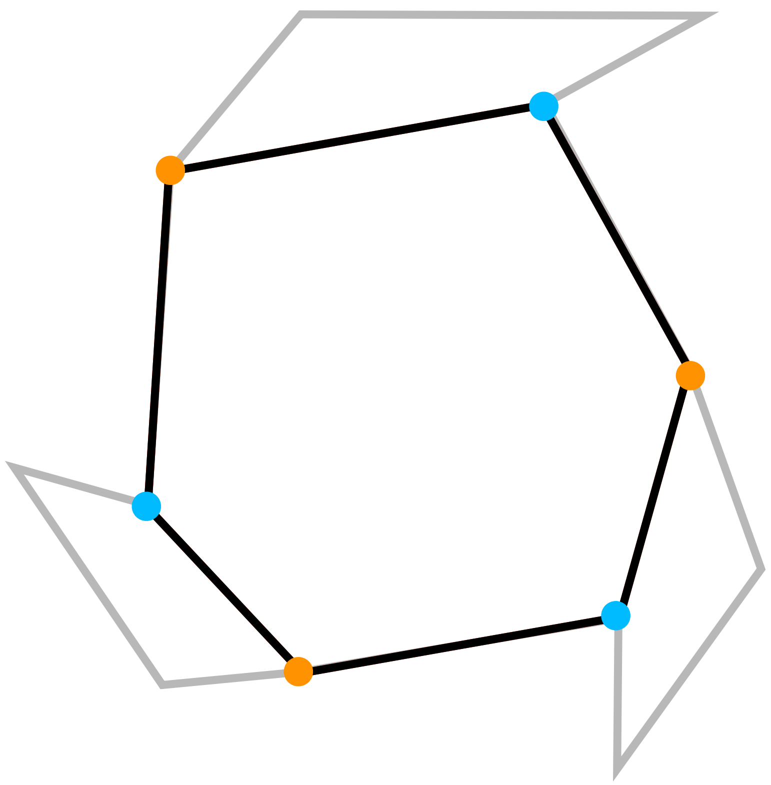
- ... until only one remains

  ⇒must be a clique!

  ⇒contains all vertices of some original class

The Class C*

existence of a clique



- Cut ears repeatedly...

  ⇒ cut the entire class

  ⇒ no class will split!

- ... until only one remains

  ⇒ must be a clique!

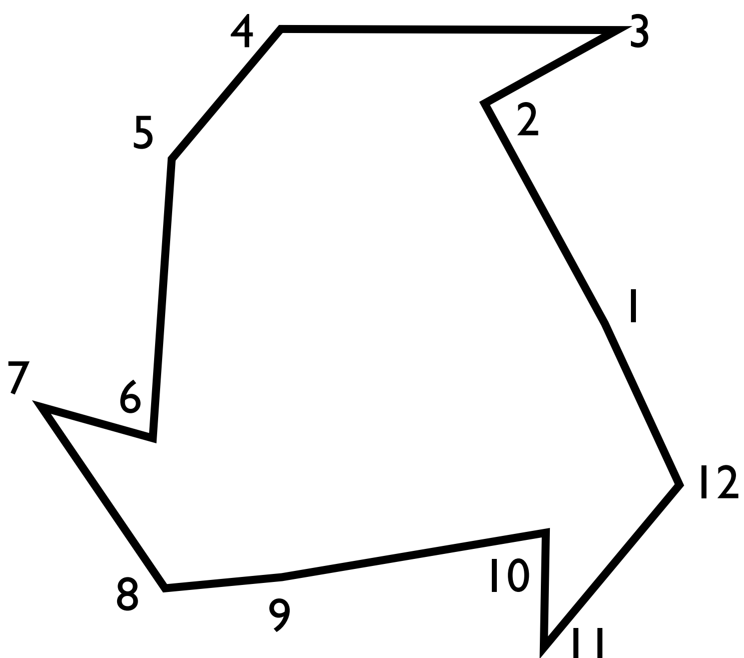
  ⇒ contains all vertices of some original class

⇒ Every polygon has a class that is a clique!

Meeting and Mapping

Meeting and Mapping

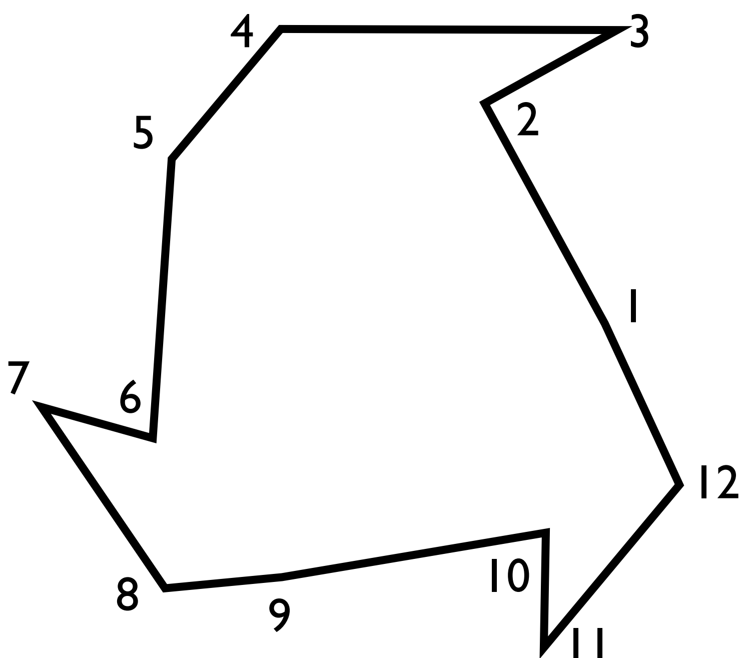
problem re-definition



vert	class	neighbors
1	C_1	C_2,C_4,C_1,C_2,C_4,C_1,C_2,C_3,C_4
2	C_2	C_3,C_4,C_1,C_2,C_4,C_1,C_2,C_4, C_1
3	C_3	C_4,C_1,C_2
4	C_4	C_1,C_2,C_4,C_1,C_2,C_4, C_1,C_2,C_3
5	C_1	C_2,C_4,C_1,C_2,C_4,C_1,C_2,C_3,C_4
6	C_2	C_3,C_4,C_1,C_2,C_4,C_1,C_2,C_4, C_1
7	C_3	C_4,C_1,C_2
8	C_4	C_1,C_2,C_4,C_1,C_2,C_4, C_1,C_2,C_3
9	C_1	C_2,C_4,C_1,C_2,C_4,C_1,C_2,C_3,C_4
10	C_2	C_3,C_4,C_1,C_2,C_4,C_1,C_2,C_4, C_1
11	C_3	C_4,C_1,C_2
12	C_4	C_1,C_2,C_4,C_1,C_2,C_4, C_1,C_2,C_3

Meeting and Mapping

problem re-definition

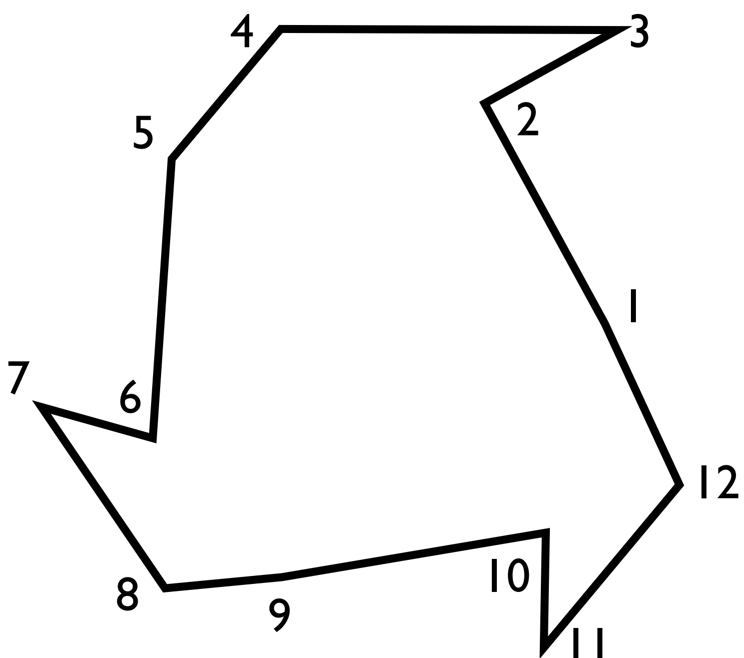


vert	class	neighbors
1	C_1	C_2,C_4,C_1,C_2,C_4,C_1,C_2,C_3,C_4
2	C_2	C_3,C_4,C_1,C_2,C_4,C_1,C_2,C_4, C_1
3	C_3	C_4,C_1,C_2
4	C_4	C_1,C_2,C_4,C_1,C_2,C_4, C_1,C_2,C_3
5	C_1	C_2,C_4,C_1,C_2,C_4,C_1,C_2,C_3,C_4
6	C_2	C_3,C_4,C_1,C_2,C_4,C_1,C_2,C_4, C_1
7	C_3	C_4,C_1,C_2
8	C_4	C_1,C_2,C_4,C_1,C_2,C_4, C_1,C_2,C_3
9	C_1	C_2,C_4,C_1,C_2,C_4,C_1,C_2,C_3,C_4
10	C_2	C_3,C_4,C_1,C_2,C_4,C_1,C_2,C_4, C_1
11	C_3	C_4,C_1,C_2
12	C_4	C_1,C_2,C_4,C_1,C_2,C_4, C_1,C_2,C_3

- Views of level n-1 are sufficient to infer classes

Meeting and Mapping

problem re-definition



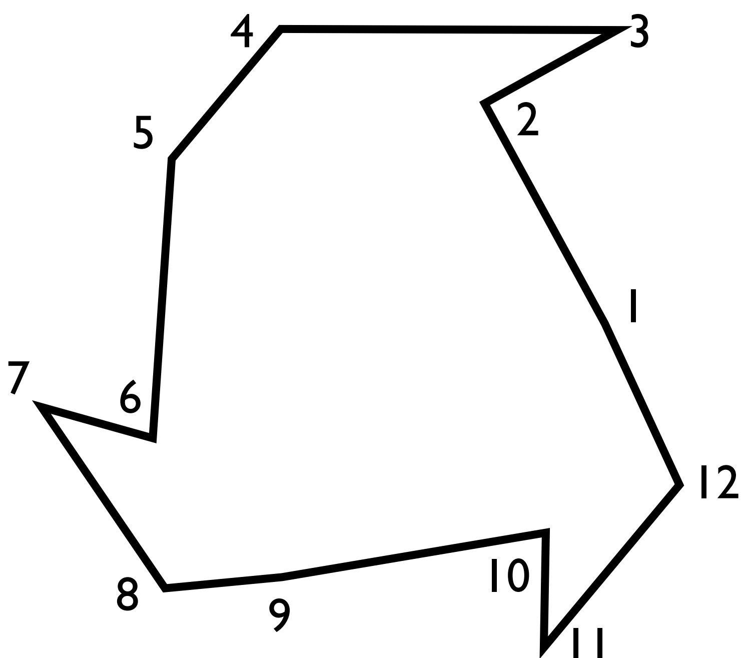
vert	class	neighbors
1	C_1	$C_2,C_4,C_1,C_2,C_4,C_1,C_2,C_3,C_4$
2	C_2	$C_3,C_4,C_1,C_2,C_4,C_1,C_2,C_4, C_1$
3	C_3	C_4,C_1,C_2
4	C_4	$C_1,C_2,C_4,C_1,C_2,C_4, C_1,C_2,C_3$
5	C_1	$C_2,C_4,C_1,C_2,C_4,C_1,C_2,C_3,C_4$
6	C_2	$C_3,C_4,C_1,C_2,C_4,C_1,C_2,C_4, C_1$
7	C_3	C_4,C_1,C_2
8	C_4	$C_1,C_2,C_4,C_1,C_2,C_4, C_1,C_2,C_3$
9	C_1	$C_2,C_4,C_1,C_2,C_4,C_1,C_2,C_3,C_4$
10	C_2	$C_3,C_4,C_1,C_2,C_4,C_1,C_2,C_4, C_1$
11	C_3	C_4,C_1,C_2
12	C_4	$C_1,C_2,C_4,C_1,C_2,C_4, C_1,C_2,C_3$

- Views of level n-1 are sufficient to infer classes

  $\Rightarrow$ task in terms of classes

Meeting and Mapping

problem re-definition



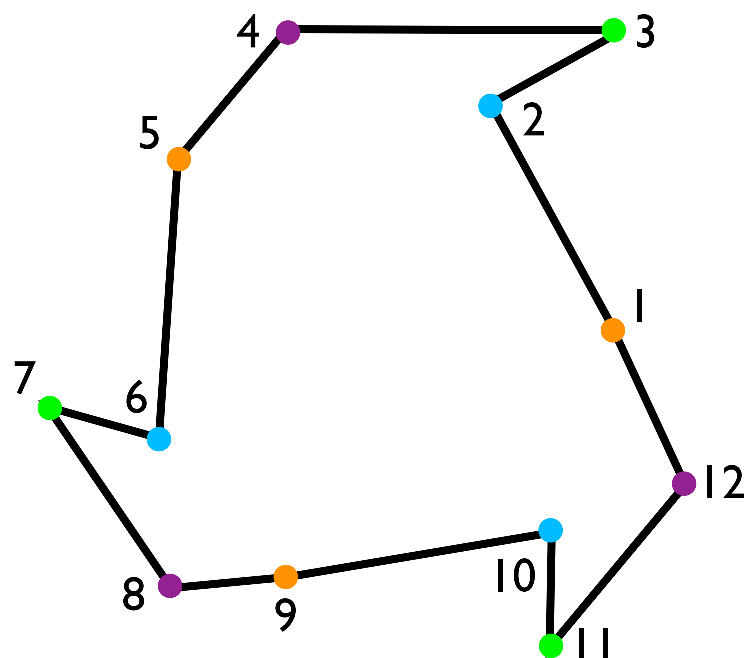
vert	class	neighbors
1	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$
2	C_2	$C_3, C_4, C_1, C_2, C_4, C_1, C_2, C_4, C_1$
3	C_3	C_4, C_1, C_2
4	C_4	$C_1, C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3$
5	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$
6	C_2	$C_3, C_4, C_1, C_2, C_4, C_1, C_2, C_4, C_1$
7	C_3	C_4, C_1, C_2
8	C_4	$C_1, C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3$
9	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$
10	C_2	$C_3, C_4, C_1, C_2, C_4, C_1, C_2, C_4, C_1$
11	C_3	C_4, C_1, C_2
12	C_4	$C_1, C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3$

- Views of level n-1 are sufficient to infer classes

  ⇒task in terms of classes

- Given:

Meeting and Mapping
problem re-definition



vert	class
1	C_1
2	C_2
3	C_3
4	C_4
5	C_1
6	C_2
7	C_3
8	C_4
9	C_1
10	C_2
11	C_3
12	C_4

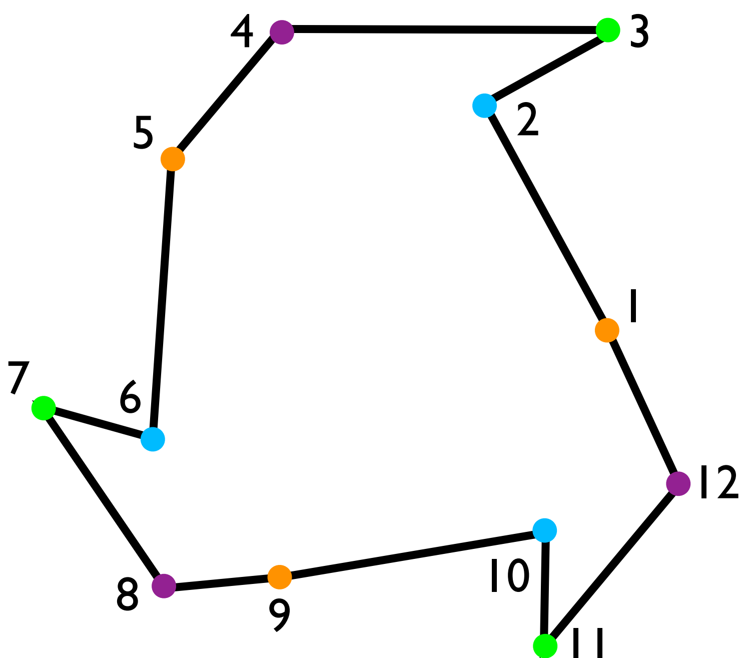
- Views of level n-1 are sufficient to infer classes

  ⇒task in terms of classes

- Given:
  - classes along boundary

Meeting and Mapping

problem re-definition



vert	class	neighbors
1	C_1	$C_2,C_4,C_1,C_2,C_4,C_1,C_2,C_3,C_4$
2	C_2	$C_3,C_4,C_1,C_2,C_4,C_1,C_2,C_4, C_1$
3	C_3	C_4,C_1,C_2
4	C_4	$C_1,C_2,C_4,C_1,C_2,C_4, C_1,C_2,C_3$
5	C_1	$C_2,C_4,C_1,C_2,C_4,C_1,C_2,C_3,C_4$
6	C_2	$C_3,C_4,C_1,C_2,C_4,C_1,C_2,C_4, C_1$
7	C_3	C_4,C_1,C_2
8	C_4	$C_1,C_2,C_4,C_1,C_2,C_4, C_1,C_2,C_3$
9	C_1	$C_2,C_4,C_1,C_2,C_4,C_1,C_2,C_3,C_4$
10	C_2	$C_3,C_4,C_1,C_2,C_4,C_1,C_2,C_4, C_1$
11	C_3	C_4,C_1,C_2
12	C_4	$C_1,C_2,C_4,C_1,C_2,C_4, C_1,C_2,C_3$

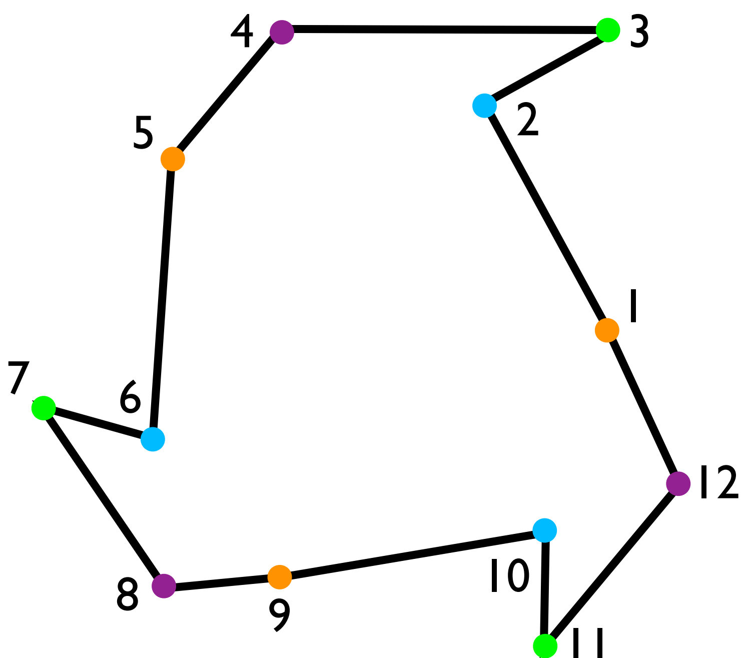
- Views of level n-1 are sufficient to infer classes

  ⇒task in terms of classes

- Given:
  - classes along boundary
  - classes of neighbors

Meeting and Mapping

problem re-definition



vert	class	neighbors
1	C_1	$C_2,C_4,C_1,C_2,C_4,C_1,C_2,C_3,C_4$
2	C_2	$C_3,C_4,C_1,C_2,C_4,C_1,C_2,C_4, C_1$
3	C_3	C_4,C_1,C_2
4	C_4	$C_1,C_2,C_4,C_1,C_2,C_4, C_1,C_2,C_3$
5	C_1	$C_2,C_4,C_1,C_2,C_4,C_1,C_2,C_3,C_4$
6	C_2	$C_3,C_4,C_1,C_2,C_4,C_1,C_2,C_4, C_1$
7	C_3	C_4,C_1,C_2
8	C_4	$C_1,C_2,C_4,C_1,C_2,C_4, C_1,C_2,C_3$
9	C_1	$C_2,C_4,C_1,C_2,C_4,C_1,C_2,C_3,C_4$
10	C_2	$C_3,C_4,C_1,C_2,C_4,C_1,C_2,C_4, C_1$
11	C_3	C_4,C_1,C_2
12	C_4	$C_1,C_2,C_4,C_1,C_2,C_4, C_1,C_2,C_3$

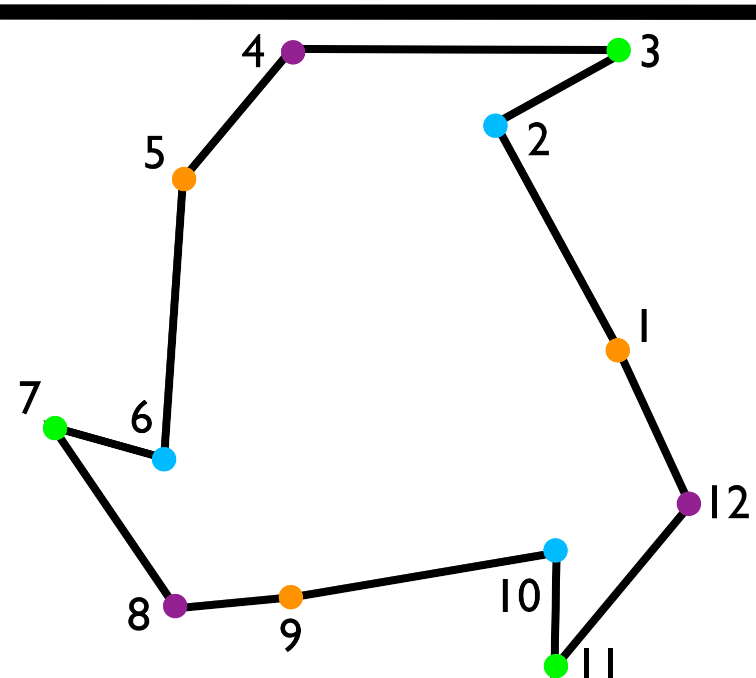
- Views of level n-1 are sufficient to infer classes

  ⇒task in terms of classes

- Given:
  - classes along boundary
  - classes of neighbors
- Tasks:

Meeting and Mapping

problem re-definition



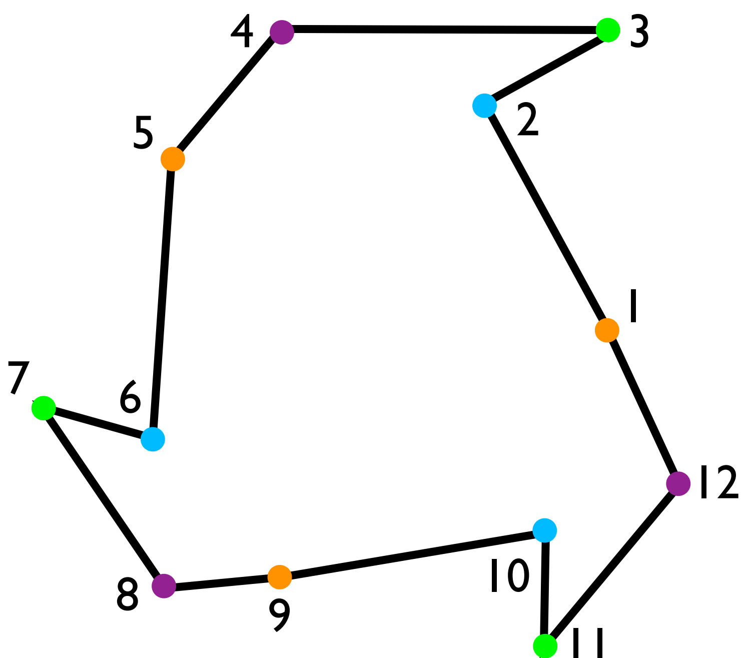
vert	class	neighbors
1	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$
2	C_2	$C_3, C_4, C_1, C_2, C_4, C_1, C_2, C_4, C_1$
3	C_3	C_4, C_1, C_2
4	C_4	$C_1, C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3$
5	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$
6	C_2	$C_3, C_4, C_1, C_2, C_4, C_1, C_2, C_4, C_1$
7	C_3	C_4, C_1, C_2
8	C_4	$C_1, C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3$
9	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$
10	C_2	$C_3, C_4, C_1, C_2, C_4, C_1, C_2, C_4, C_1$
11	C_3	C_4, C_1, C_2
12	C_4	$C_1, C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3$

- Views of level n-1 are sufficient to infer classes

  ⇒task in terms of classes
- Given:
  - classes along boundary
  - classes of neighbors
- Tasks:
  - meet other robots

Meeting and Mapping

problem re-definition

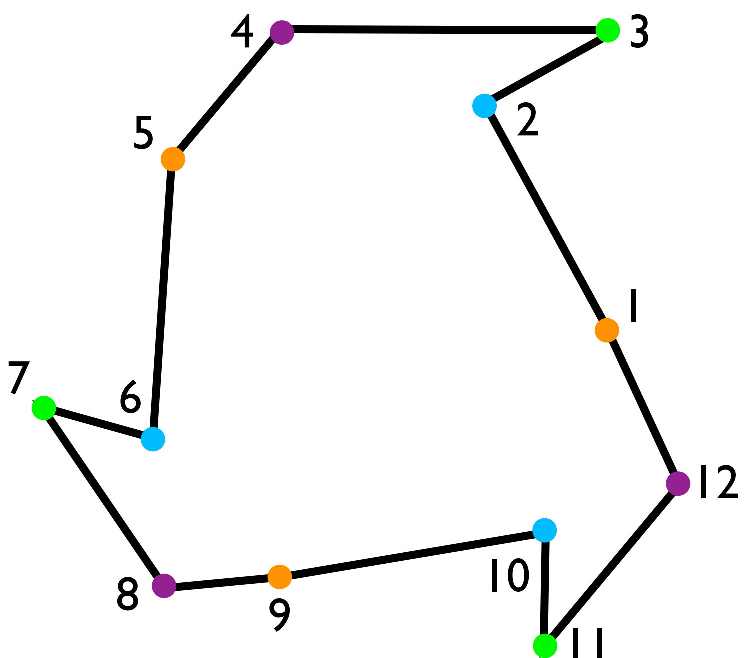


vert	class	neighbors
1	C_1	$C_2,C_4,C_1,C_2,C_4,C_1,C_2,C_3,C_4$
2	C_2	$C_3,C_4,C_1,C_2,C_4,C_1,C_2,C_4, C_1$
3	C_3	C_4,C_1,C_2
4	C_4	$C_1,C_2,C_4,C_1,C_2,C_4, C_1,C_2,C_3$
5	C_1	$C_2,C_4,C_1,C_2,C_4,C_1,C_2,C_3,C_4$
6	C_2	$C_3,C_4,C_1,C_2,C_4,C_1,C_2,C_4, C_1$
7	C_3	C_4,C_1,C_2
8	C_4	$C_1,C_2,C_4,C_1,C_2,C_4, C_1,C_2,C_3$
9	C_1	$C_2,C_4,C_1,C_2,C_4,C_1,C_2,C_3,C_4$
10	C_2	$C_3,C_4,C_1,C_2,C_4,C_1,C_2,C_4, C_1$
11	C_3	C_4,C_1,C_2
12	C_4	$C_1,C_2,C_4,C_1,C_2,C_4, C_1,C_2,C_3$

- Views of level n-1 are sufficient to infer classes

  ⇒task in terms of classes
- Given:
  - classes along boundary
  - classes of neighbors
- Tasks:
  - meet other robots
  - infer visibility graph

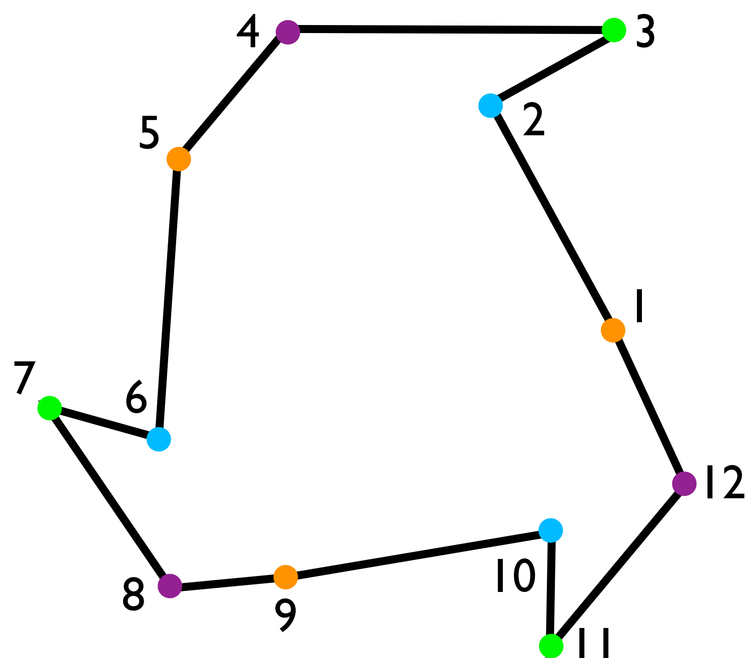
Meeting and Mapping
meeting



vert	class	neighbors
1	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$
2	C_2	$C_3, C_4, C_1, C_2, C_4, C_1, C_2, C_4, C_1$
3	C_3	C_4, C_1, C_2
4	C_4	$C_1, C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3$
5	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$
6	C_2	$C_3, C_4, C_1, C_2, C_4, C_1, C_2, C_4, C_1$
7	C_3	C_4, C_1, C_2
8	C_4	$C_1, C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3$
9	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$
10	C_2	$C_3, C_4, C_1, C_2, C_4, C_1, C_2, C_4, C_1$
11	C_3	C_4, C_1, C_2
12	C_4	$C_1, C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3$

Meeting and Mapping

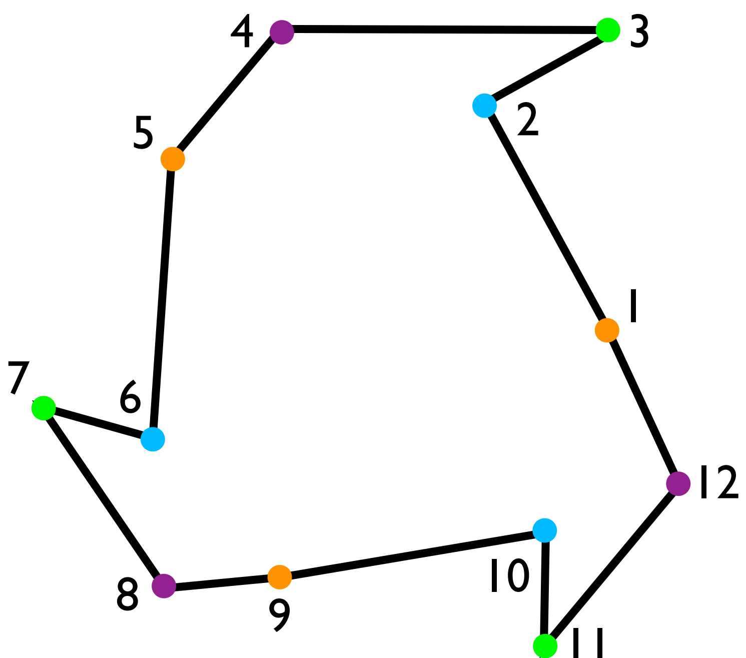
meeting



vert	class	neighbors
1	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$
2	C_2	$C_3, C_4, C_1, C_2, C_4, C_1, C_2, C_4, C_1$
3	C_3	C_4, C_1, C_2
4	C_4	$C_1, C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3$
5	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$
6	C_2	$C_3, C_4, C_1, C_2, C_4, C_1, C_2, C_4, C_1$
7	C_3	C_4, C_1, C_2
8	C_4	$C_1, C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3$
9	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$
10	C_2	$C_3, C_4, C_1, C_2, C_4, C_1, C_2, C_4, C_1$
11	C_3	C_4, C_1, C_2
12	C_4	$C_1, C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3$

- C^* is unique

Meeting and Mapping
meeting

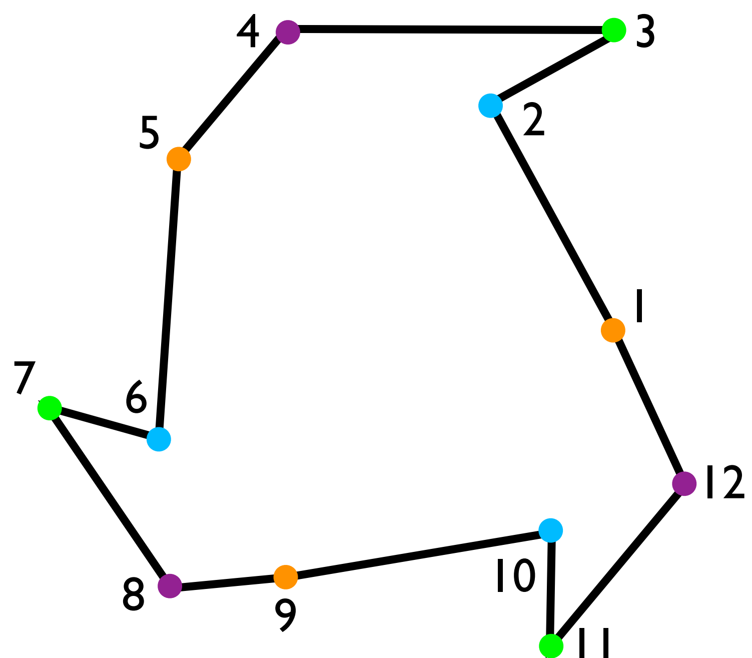


vert	class	neighbors
1	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$
2	C_2	$C_3, C_4, C_1, C_2, C_4, C_1, C_2, C_4, C_1$
3	C_3	C_4, C_1, C_2
4	C_4	$C_1, C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3$
5	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$
6	C_2	$C_3, C_4, C_1, C_2, C_4, C_1, C_2, C_4, C_1$
7	C_3	C_4, C_1, C_2
8	C_4	$C_1, C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3$
9	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$
10	C_2	$C_3, C_4, C_1, C_2, C_4, C_1, C_2, C_4, C_1$
11	C_3	C_4, C_1, C_2
12	C_4	$C_1, C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3$

- C^* is unique

Meeting and Mapping

meeting

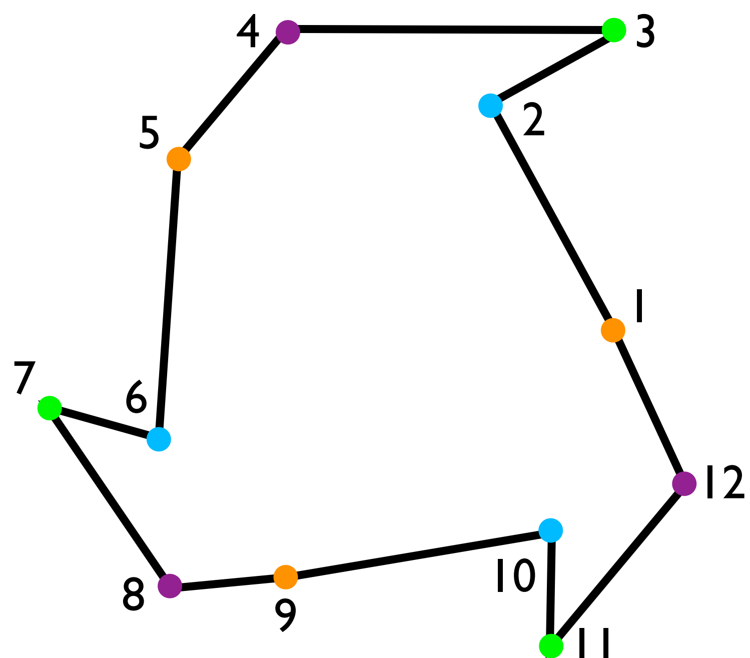


vert	class	neighbors
1	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$
2	C_2	$C_3, C_4, C_1, C_2, C_4, C_1, C_2, C_4, C_1$
3	C_3	C_4, C_1, C_2
4	C_4	$C_1, C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3$
5	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$
6	C_2	$C_3, C_4, C_1, C_2, C_4, C_1, C_2, C_4, C_1$
7	C_3	C_4, C_1, C_2
8	C_4	$C_1, C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3$
9	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$
10	C_2	$C_3, C_4, C_1, C_2, C_4, C_1, C_2, C_4, C_1$
11	C_3	C_4, C_1, C_2
12	C_4	$C_1, C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3$

- C^* is unique
- C^* can be infered

Meeting and Mapping

meeting



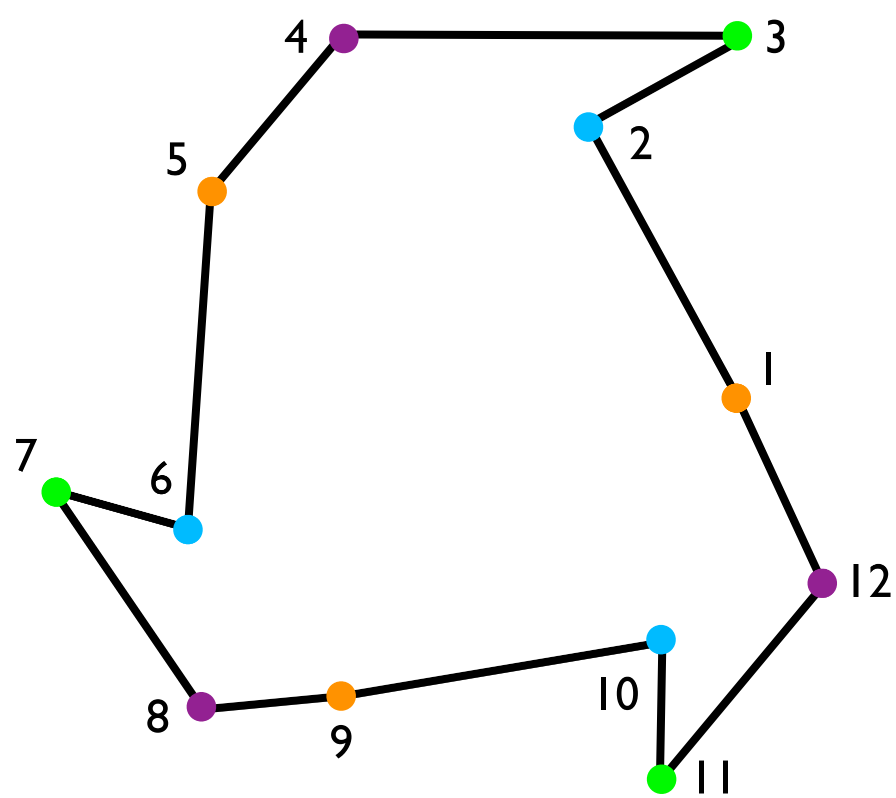
vert	class	neighbors
1	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$
2	C_2	$C_3, C_4, C_1, C_2, C_4, C_1, C_2, C_4, C_1$
3	C_3	C_4, C_1, C_2
4	C_4	$C_1, C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3$
5	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$
6	C_2	$C_3, C_4, C_1, C_2, C_4, C_1, C_2, C_4, C_1$
7	C_3	C_4, C_1, C_2
8	C_4	$C_1, C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3$
9	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$
10	C_2	$C_3, C_4, C_1, C_2, C_4, C_1, C_2, C_4, C_1$
11	C_3	C_4, C_1, C_2
12	C_4	$C_1, C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3$

- C^* is unique
- C^* can be infered

⇒ Meeting is trivial: move along boundary until a vertex in C^*

Meeting and Mapping

visibility graph reconstruction

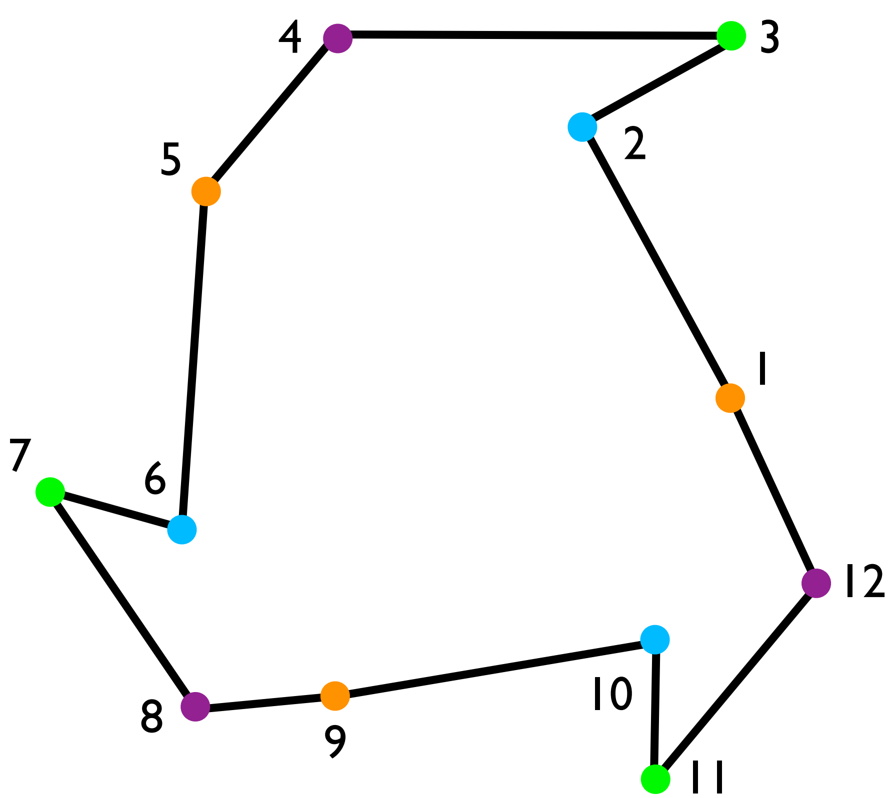


vert	class	neighbors
1	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$

Meeting and Mapping

visibility graph reconstruction

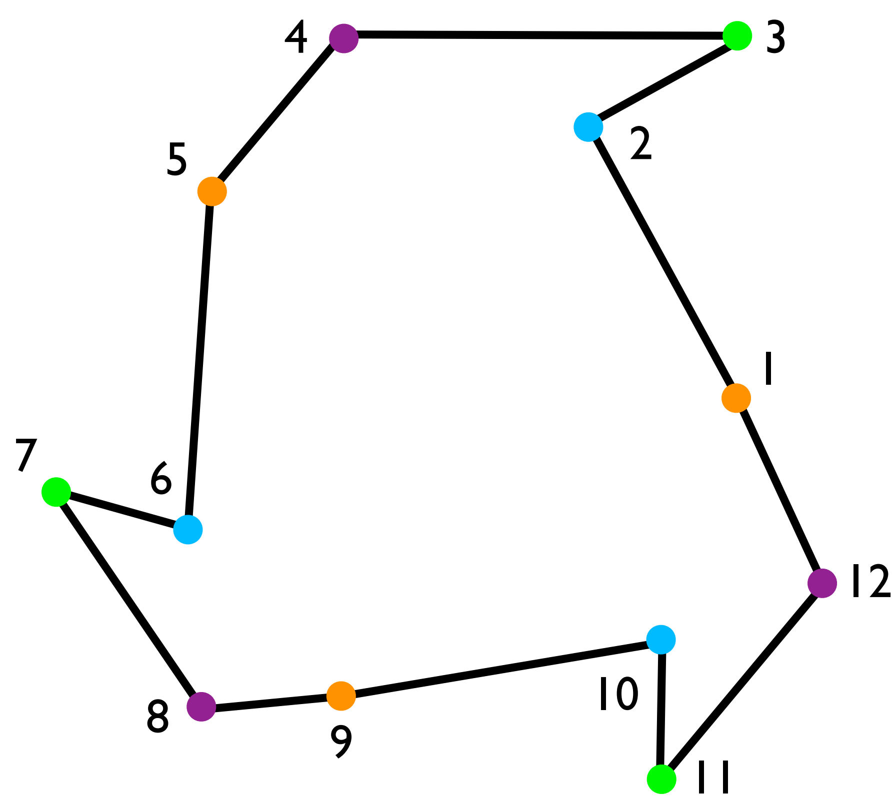
- need to identify vertices in the list of neighbors



vert	class	neighbors
1	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$

? ? ? ? ? ? ? ? ?

Meeting and Mapping
visibility graph reconstruction

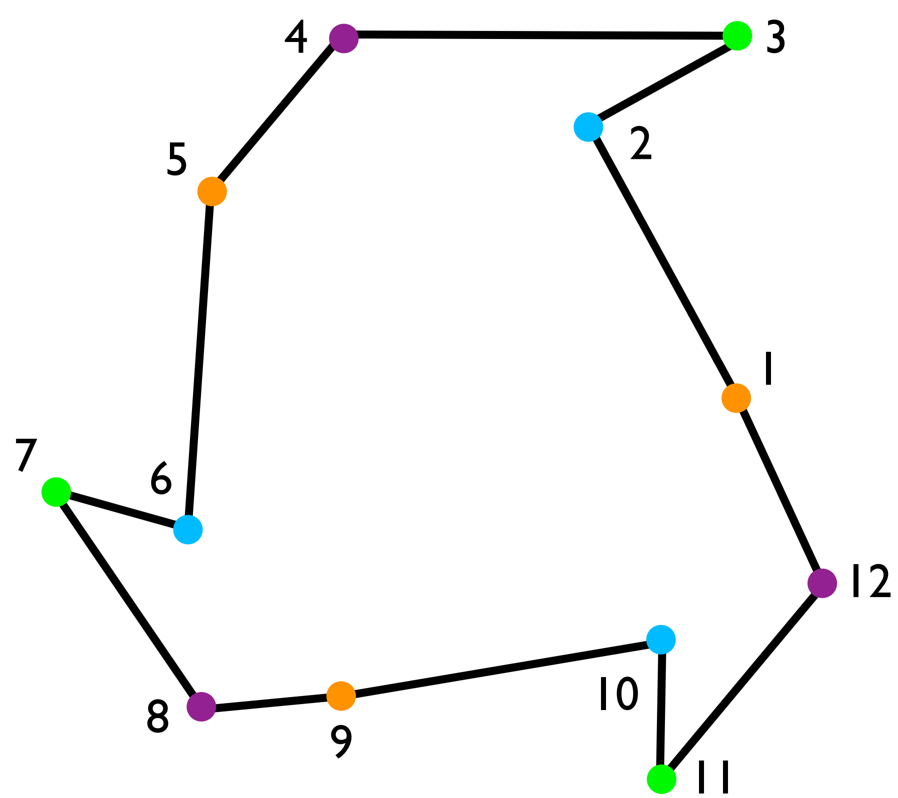


vert	class	neighbors
1	C1	C2,C4,C1,C2,C4,C1,C2,C3,C4
		? ? ? ? ? ? ? ? ?

- need to identify vertices in the list of neighbors
- If own class is a clique:

Meeting and Mapping

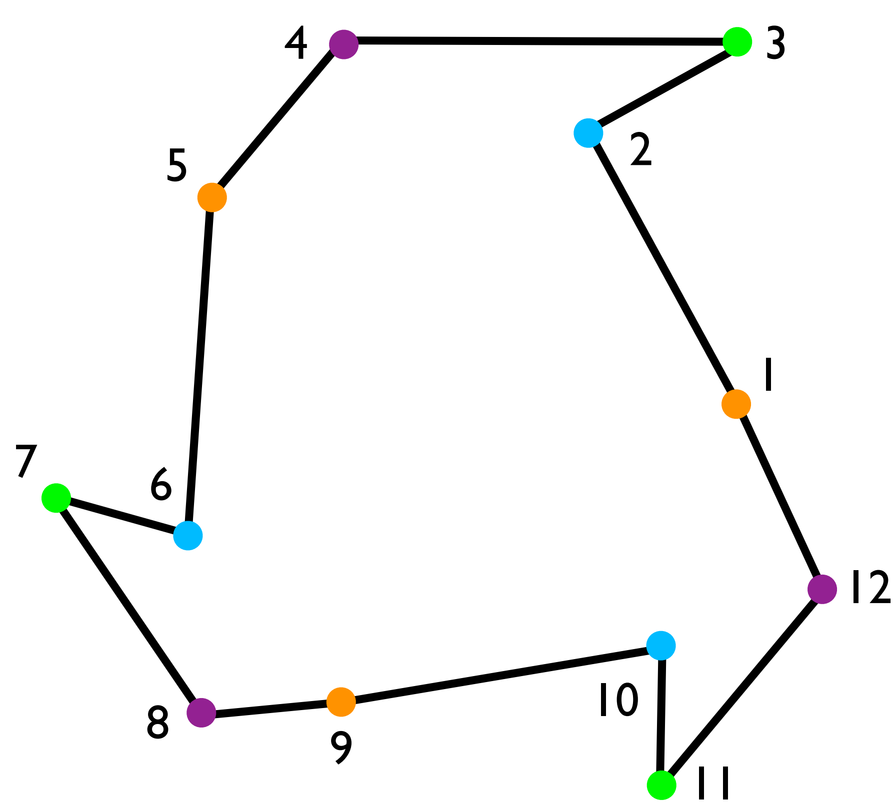
visibility graph reconstruction



vert	class	neighbors
1	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$
		? ? ? ? ? ? ? ? ?

- need to identify vertices in the list of neighbors
- If own class is a clique:
  - classmates are easy

Meeting and Mapping
visibility graph reconstruction

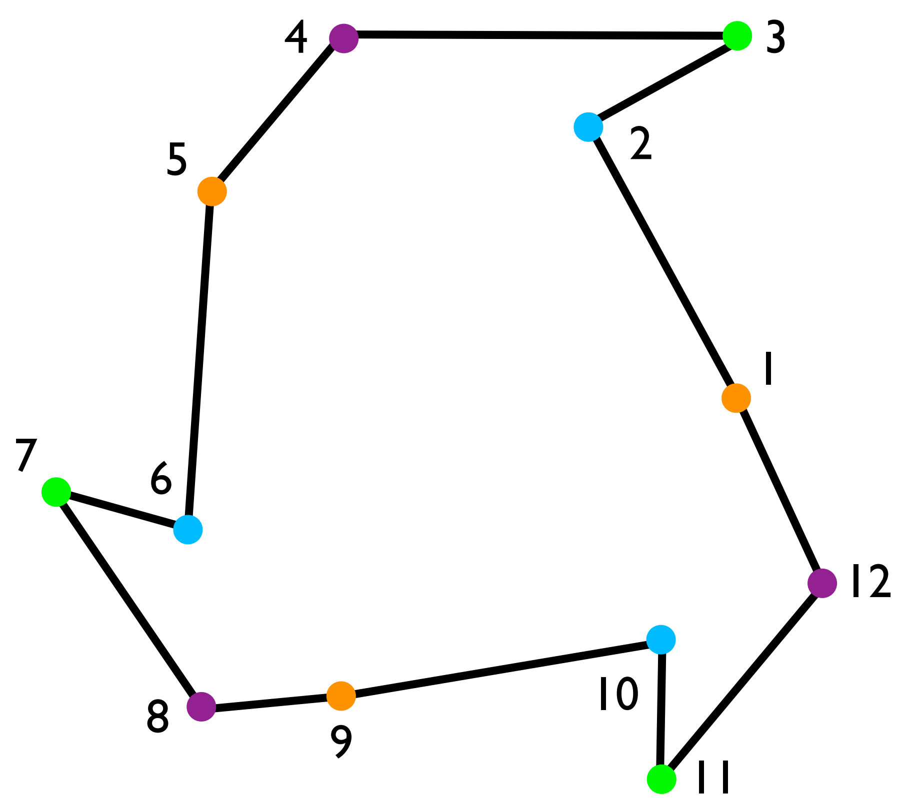


vert	class	neighbors
1	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$
		? ? 5 ? ? 9 ? ? ?

- need to identify vertices in the list of neighbors
- If own class is a clique:
  - classmates are easy

Meeting and Mapping

visibility graph reconstruction

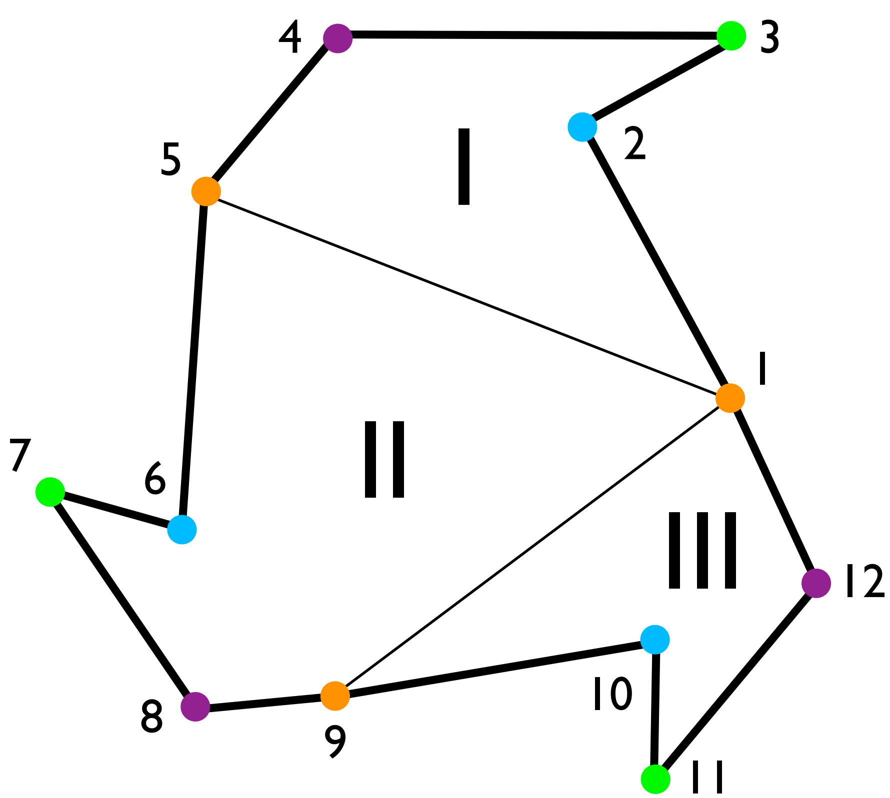


vert	class	neighbors
1	C_1	C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4
		? ? 5 ? ? 9 ? ? ?

- need to identify vertices in the list of neighbors
- If own class is a clique:
  - classmates are easy
  - classmates can be used to segment neighbors

Meeting and Mapping

visibility graph reconstruction

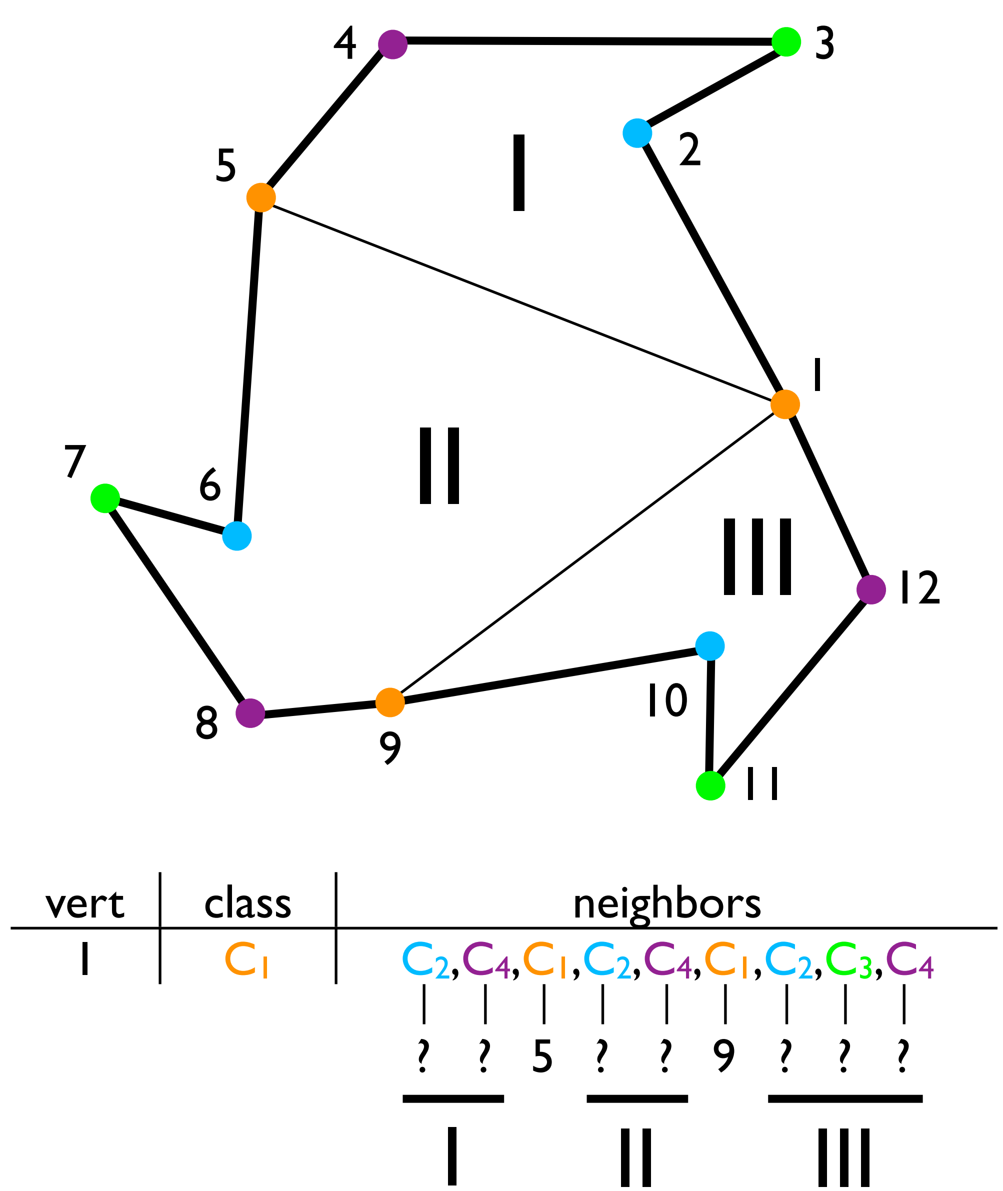


vert	class	neighbors
1	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$
		? ? 5 ? ? 9 ? ? ?
		I II III

- need to identify vertices in the list of neighbors
- If own class is a clique:
  - classmates are easy
  - classmates can be used to segment neighbors

Meeting and Mapping

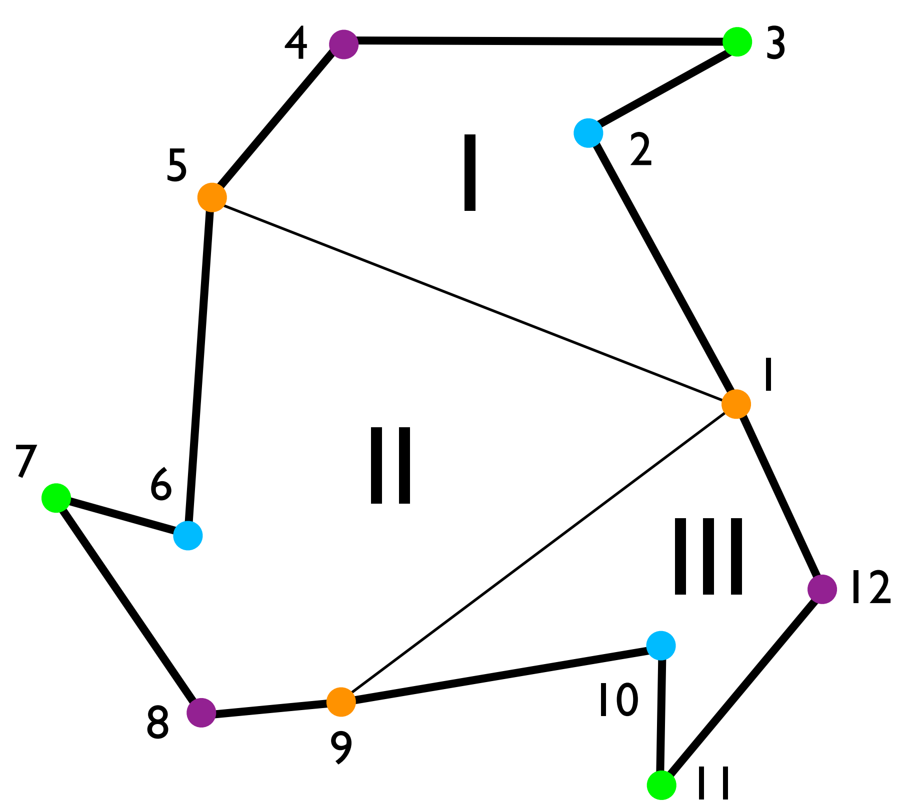
visibility graph reconstruction



- need to identify vertices in the list of neighbors
- If own class is a clique:
  - classmates are easy
  - classmates can be used to segment neighbors
- Classes are periodic

Meeting and Mapping

visibility graph reconstruction



vert	class	neighbors
1	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$

? ? 5 ? ? 9 ? 11 ?

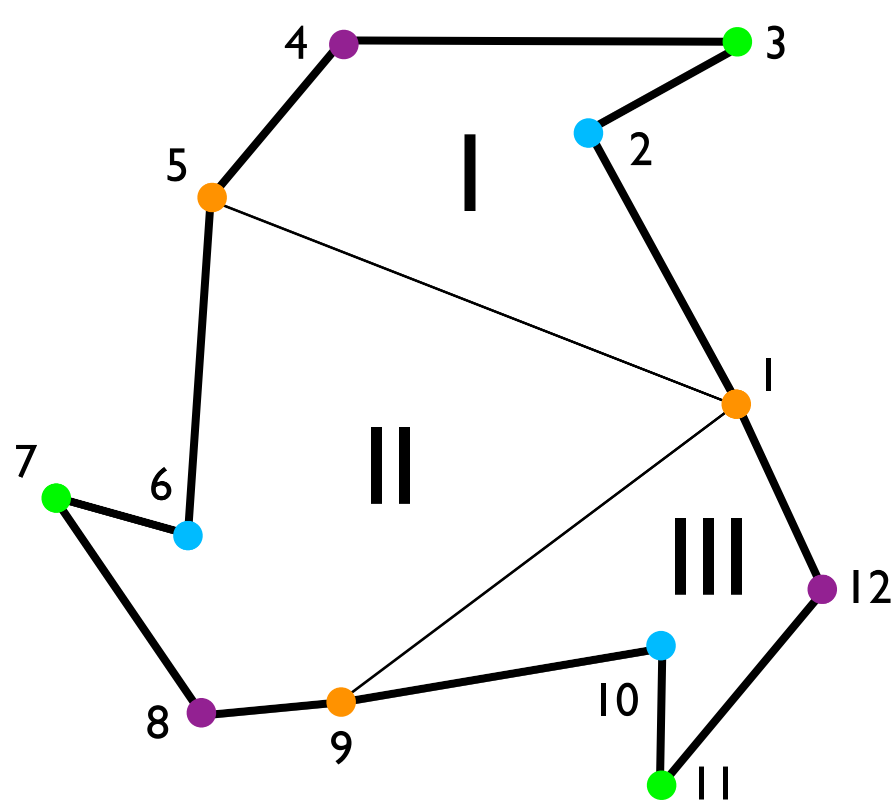
I II III

- need to identify vertices in the list of neighbors
- If own class is a clique:
  - classmates are easy
  - classmates can be used to segment neighbors
- Classes are periodic

  ⇒ segment + class → ID

Meeting and Mapping

visibility graph reconstruction



vert	class	neighbors
1	C_1	$C_2, C_4, C_1, C_2, C_4, C_1, C_2, C_3, C_4$
		? ? 5 ? ? 9 ? 11 ?
		I II III

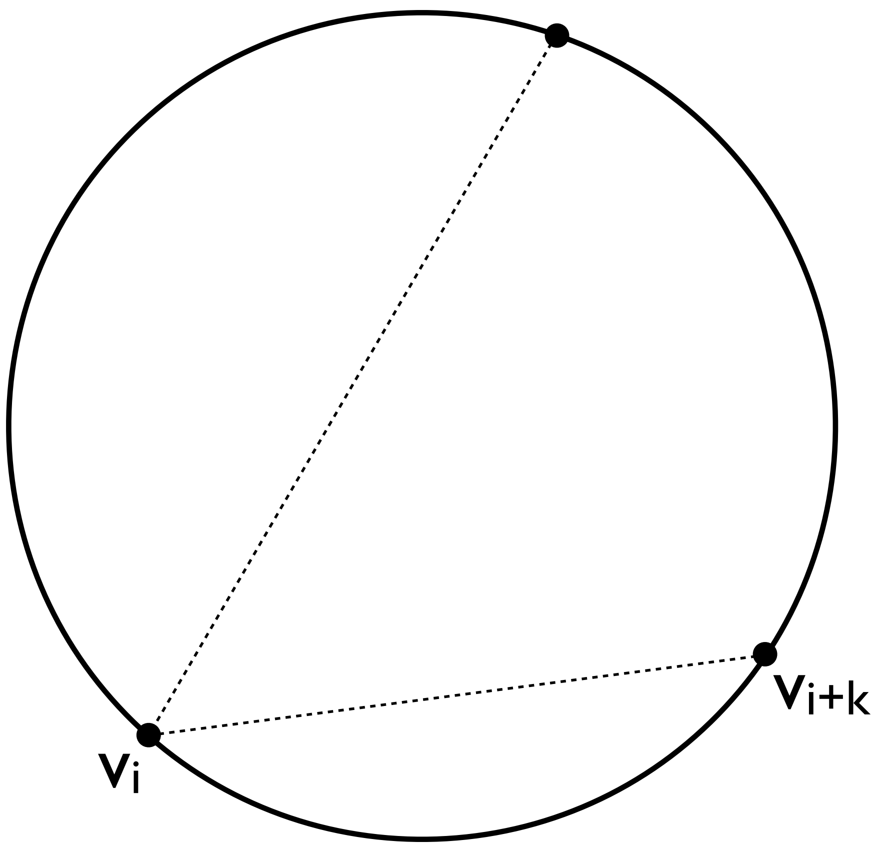
- need to identify vertices in the list of neighbors
- If own class is a clique:
  - classmates are easy
  - classmates can be used to segment neighbors
- Classes are periodic

  ⇒ segment + class → ID

⇒ C^* vertices can be done

Meeting and Mapping

visibility graph reconstruction II

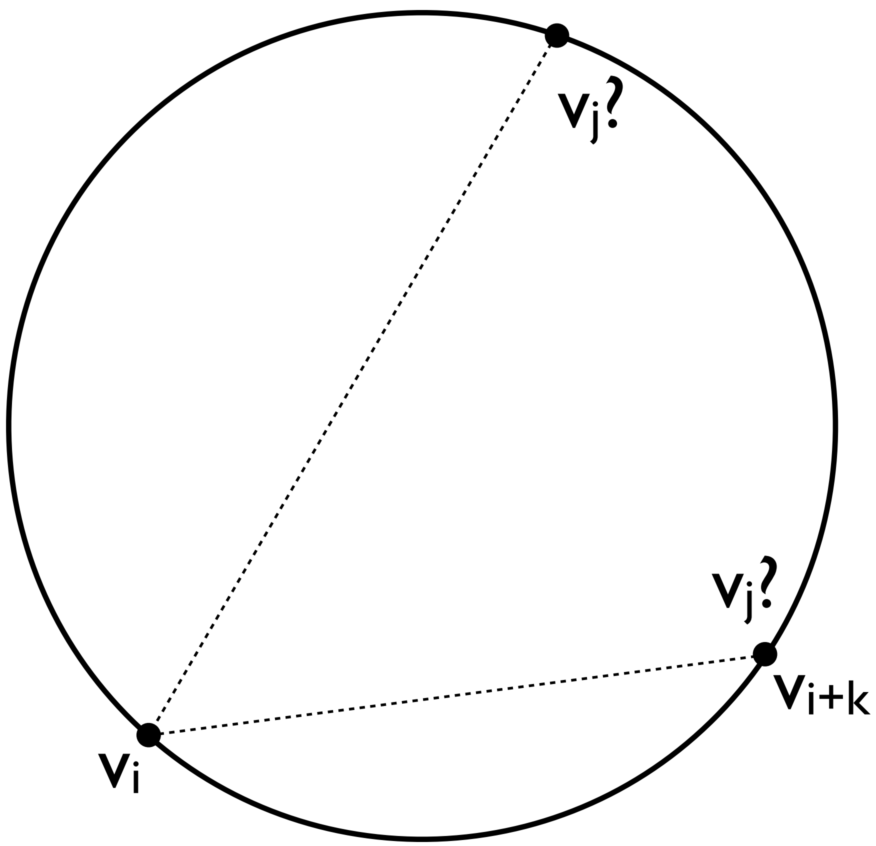


vert	class	neighbors
i	C	$C_A, C_B, C_C, \ldots, C_L, \ldots, C_M, \ldots, C_X, C_Y, C_Z$

- Identify edges (v_i, v_{i+k}) of increasing distances k

Meeting and Mapping

visibility graph reconstruction II

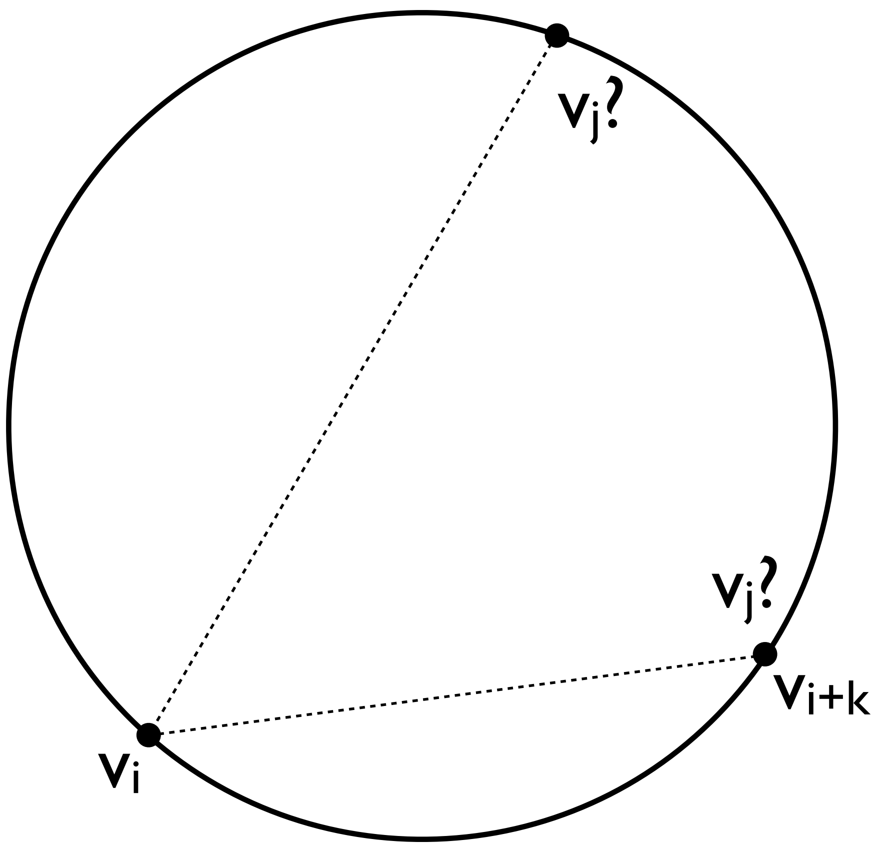


vert	class	neighbors
i	C	$C_A, C_B, C_C, \ldots, C_L, \ldots, C_M, \ldots, C_X, C_Y, C_Z$

✓ ? ✓

- Identify edges (v_i, v_{i+k}) of increasing distances k
- Is the next unidentified vertex $v_j = v_{i+k}$ or not?

Meeting and Mapping
visibility graph reconstruction II



vert	class	neighbors
i	C	$C_A, C_B, C_C, \ldots, C_L, \ldots, C_M, \ldots, C_X, C_Y, C_Z$

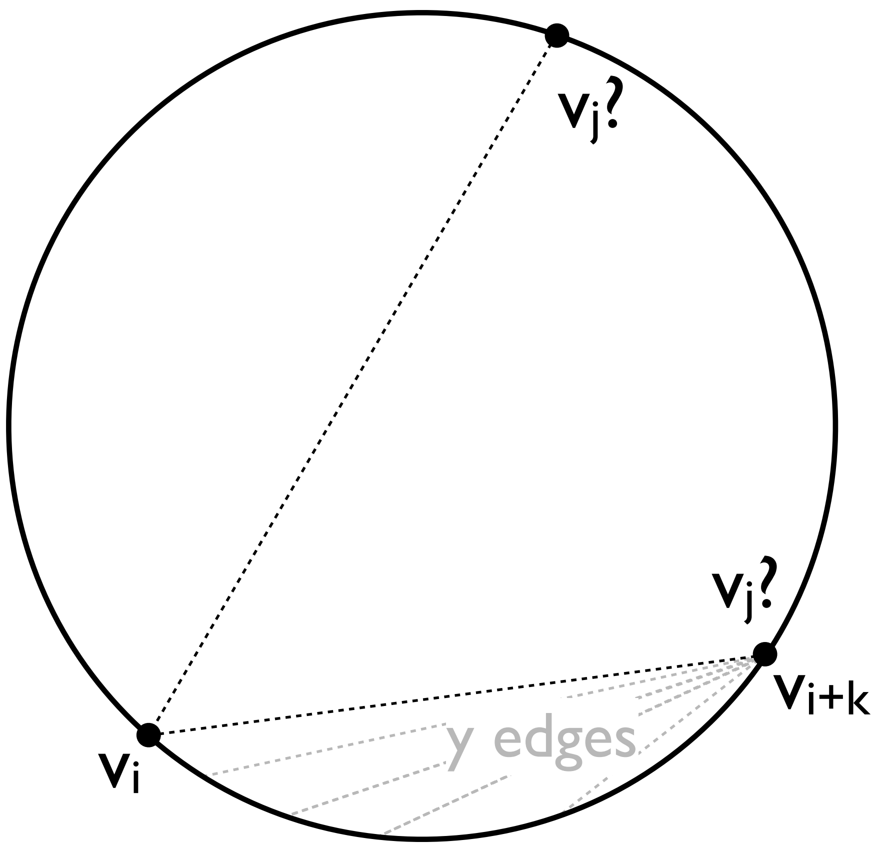
✓ ? ✓

- Identify edges (v_i, v_{i+k}) of increasing distances k
- Is the next unidentified vertex $v_j = v_{i+k}$ or not?

  $\Rightarrow$ easy, if of different class

Meeting and Mapping

visibility graph reconstruction II



- Identify edges (v_i, v_{i+k}) of increasing distances k
- Is the next unidentified vertex $v_j = v_{i+k}$ or not?

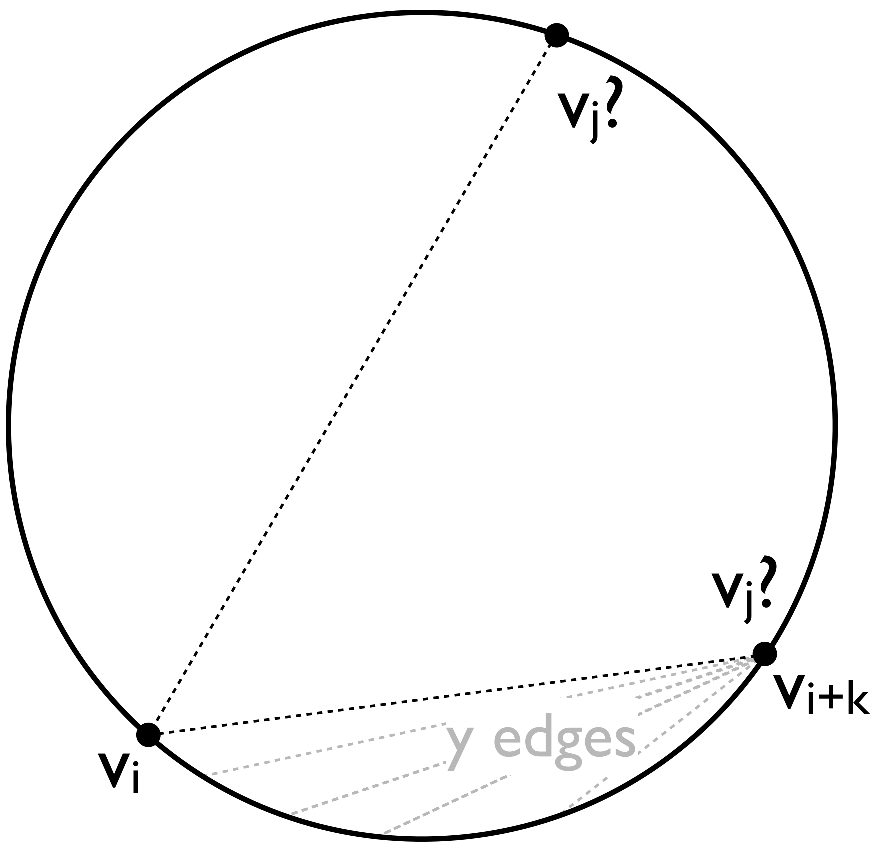
  ⇒ easy, if of different class
- y is number of dist. k-1 backward-edges of v_{i+k}

vert	class	neighbors
i	C	$C_A, C_B, C_C, \ldots, C_L, \ldots, C_M, \ldots, C_X, C_Y, C_Z$

✓ ? ✓

Meeting and Mapping

visibility graph reconstruction II



vert	class	neighbors
i	C	$C_A, C_B, C_C, \ldots, C_L, \ldots, C_M, \ldots, C_X, C_Y, C_Z$

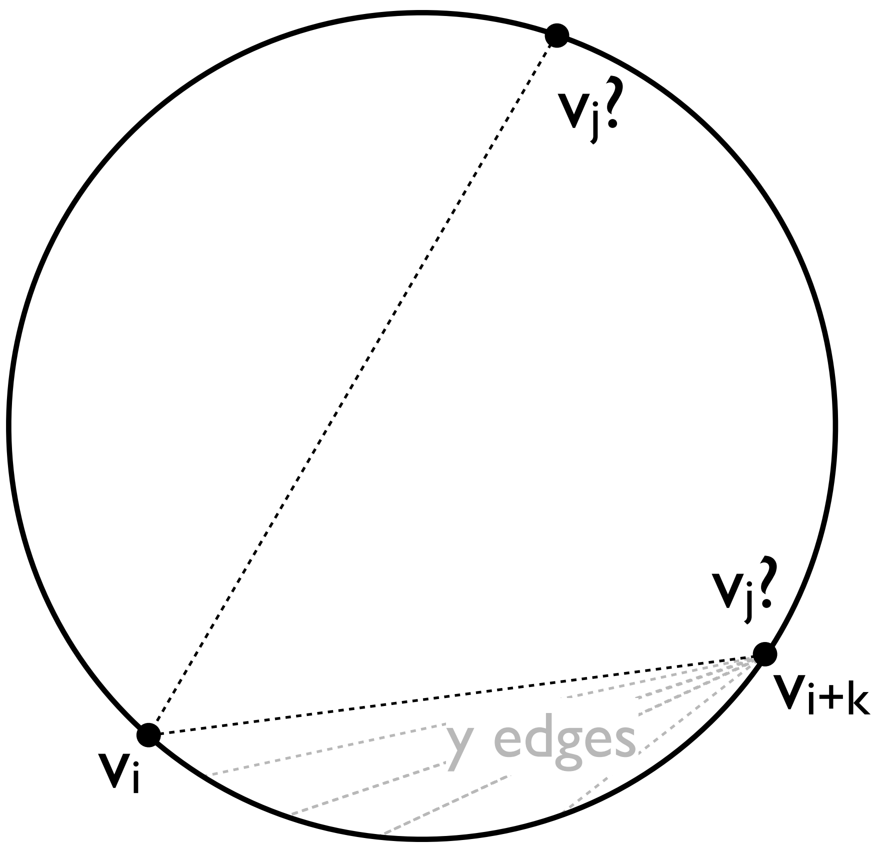
✓ ? ✓

- Identify edges (v_i, v_{i+k}) of increasing distances k
- Is the next unidentified vertex $v_j = v_{i+k}$ or not?

  $\Rightarrow$ easy, if of different class
- y is number of dist. k-1 backward-edges of v_{i+k}
- Move to v_j and look back

Meeting and Mapping

visibility graph reconstruction II

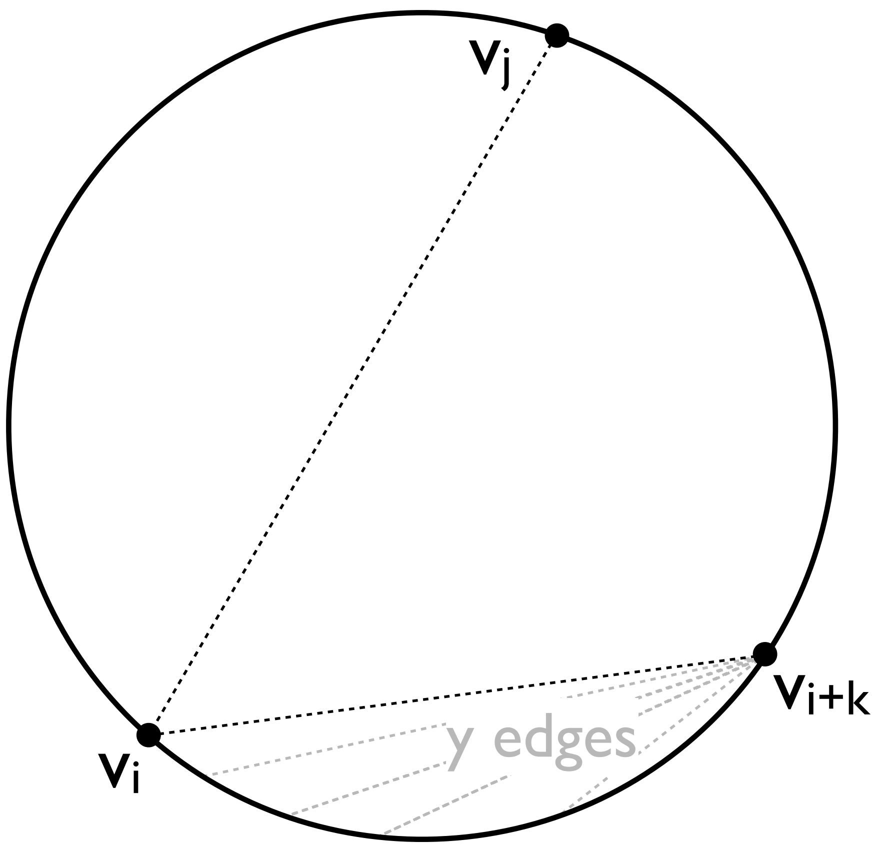


vert	class	neighbors
i	C	$C_A, C_B, C_C, \ldots, C_L, \ldots, C_M, \ldots, C_X, C_Y, C_Z$

- Identify edges (v_i, v_{i+k}) of increasing distances k
- Is the next unidentified vertex $v_j = v_{i+k}$ or not?

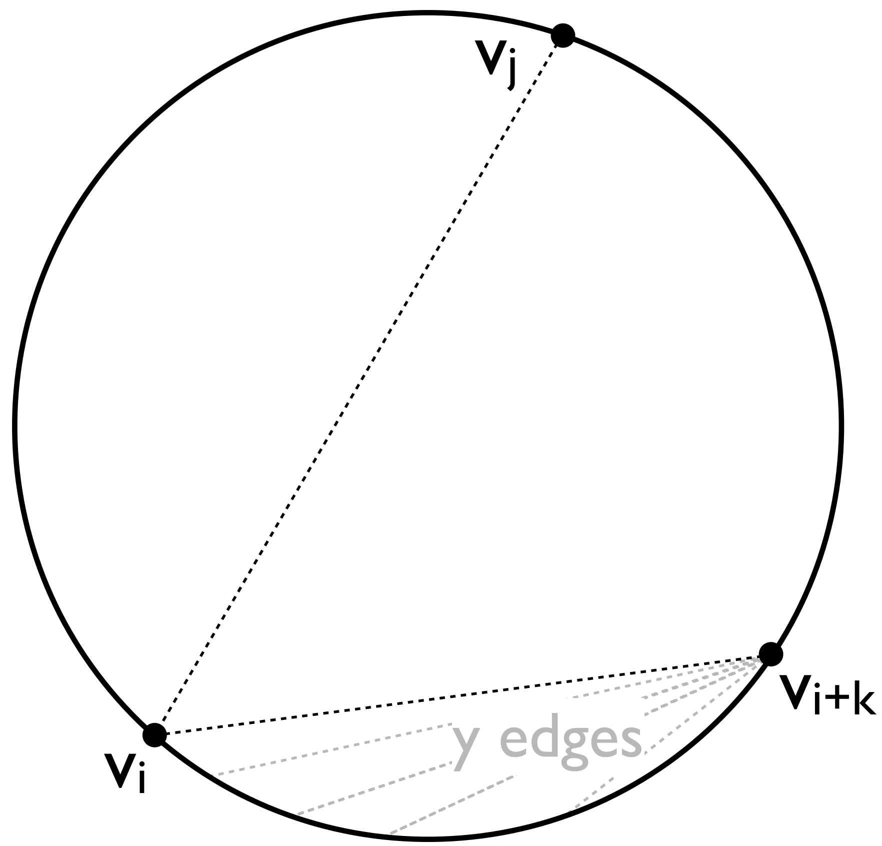
  ⇒ easy, if of different class
- y is number of dist. k-1 backward-edges of v_{i+k}
- Move to v_j and look back
  - $v_j = v_{i+k} \Rightarrow LB = -(y+1)$

Meeting and Mapping
visibility graph reconstruction III



vert	class	neighbors
i	C	$C_A, C_B, C_C, \ldots, C_L, \ldots, C_M, \ldots, C_X, C_Y, C_Z$

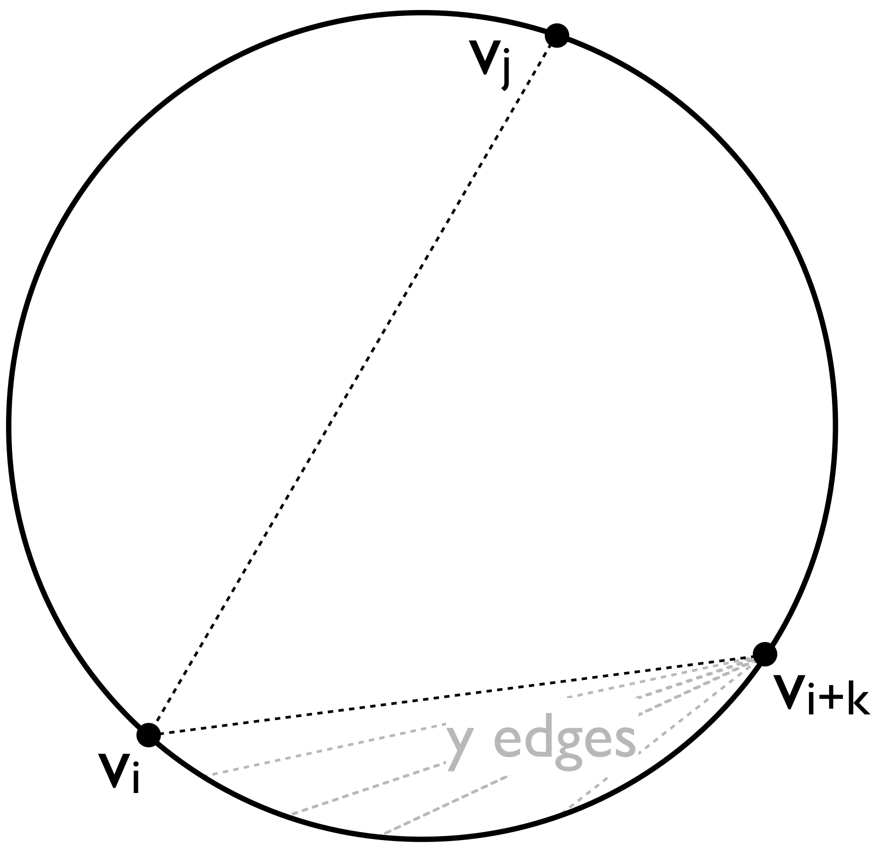
Meeting and Mapping
visibility graph reconstruction III



vert	class	neighbors
i	C	$C_A, C_B, C_C, \ldots, C_L, \ldots, C_M, \ldots, C_X, C_Y, C_Z$

- We show:
  $LB = -(y+1) \Rightarrow v_j = v_{i+k}$

Meeting and Mapping
visibility graph reconstruction III

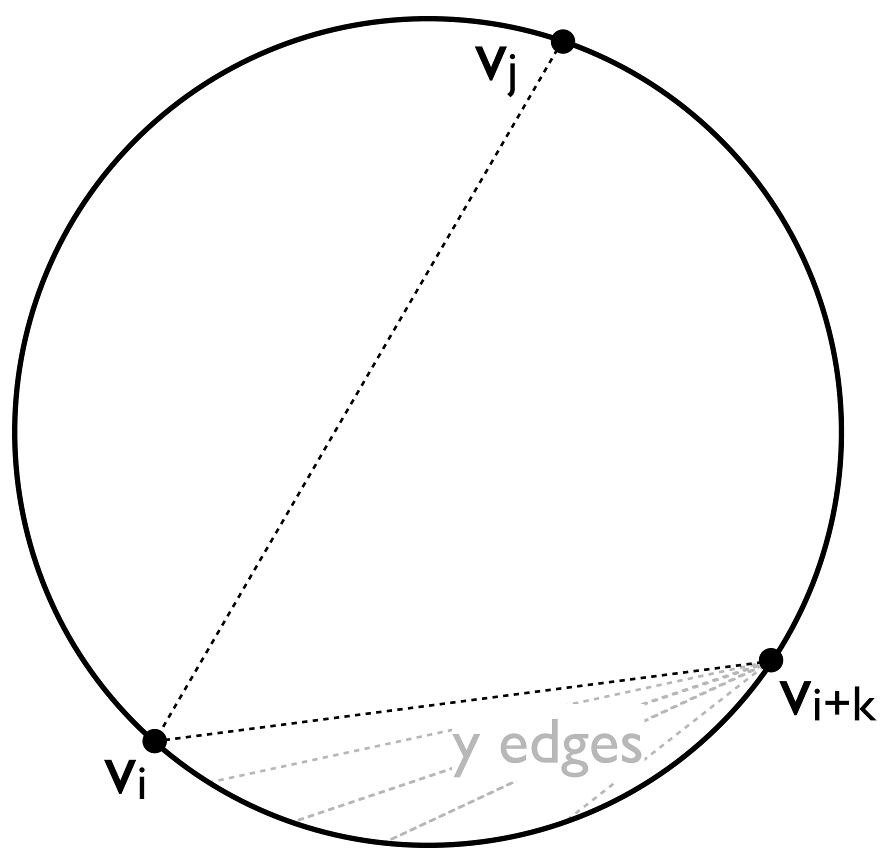


vert	class	neighbors
i	C	$C_A, C_B, C_C, \ldots, C_L, \ldots, C_M, \ldots, C_X, C_Y, C_Z$

- We show:
  $LB = -(y+1) \Rightarrow v_j = v_{i+k}$
- Assume $v_j \neq v_{i+k}$
  but $LB = -(y+1)$

Meeting and Mapping

visibility graph reconstruction III

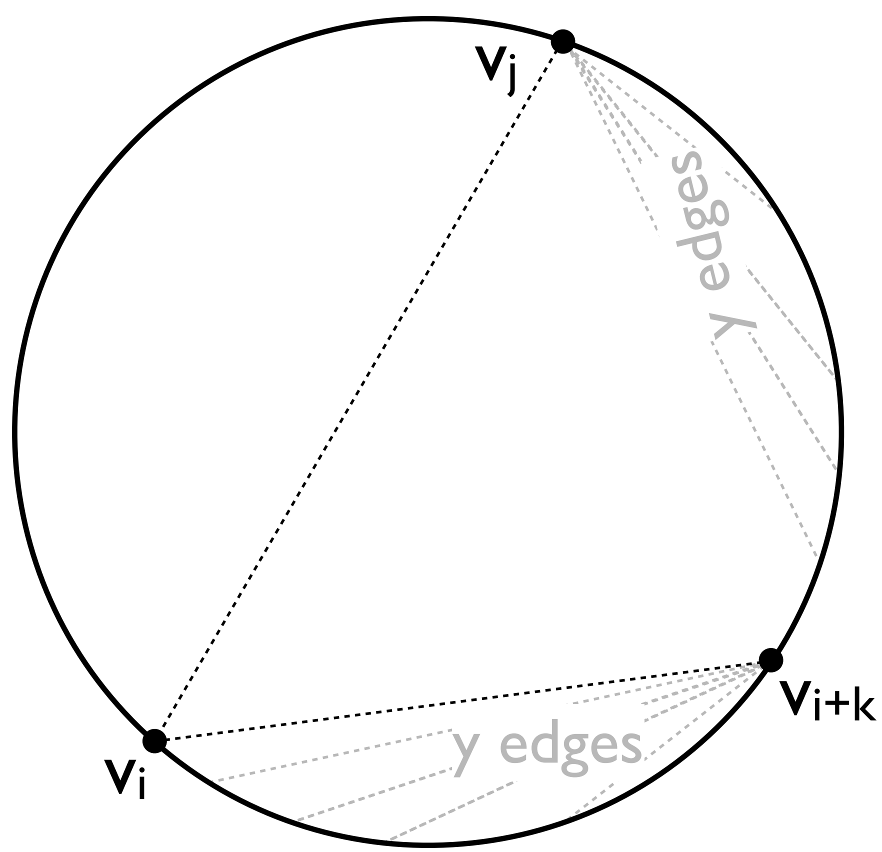


vert	class	neighbors
i	C	$C_A, C_B, C_C, \ldots, C_L, \ldots, C_M, \ldots, C_X, C_Y, C_Z$

- We show:
  LB = -(y+1) $\Rightarrow v_j = v_{i+k}$
- Assume $v_j \neq v_{i+k}$
  but LB = -(y+1)
- v_{i+k} and v_j are same class

Meeting and Mapping

visibility graph reconstruction III

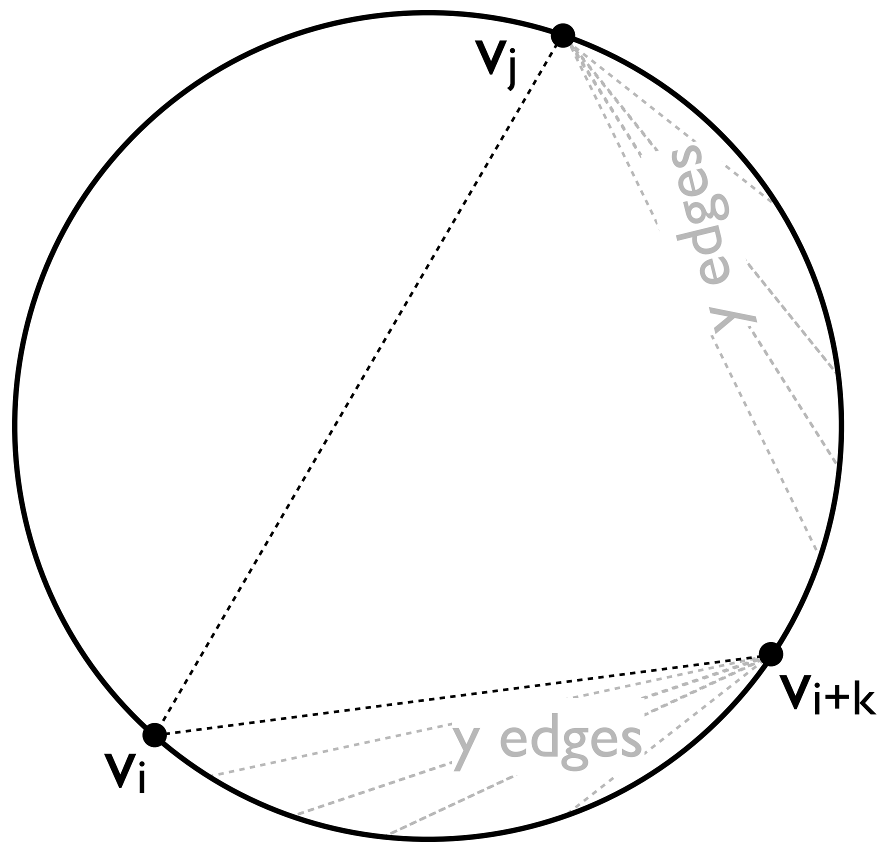


vert	class	neighbors
i	C	$C_A, C_B, C_C, \ldots, C_L, \ldots, C_M, \ldots, C_X, C_Y, C_Z$

- We show:
  LB = -(y+1) $\Rightarrow v_j = v_{i+k}$
- Assume $v_j \neq v_{i+k}$
  but LB = -(y+1)
- v_{i+k} and v_j are same class
  $\Rightarrow v_j$ has y back-edges

Meeting and Mapping

visibility graph reconstruction III

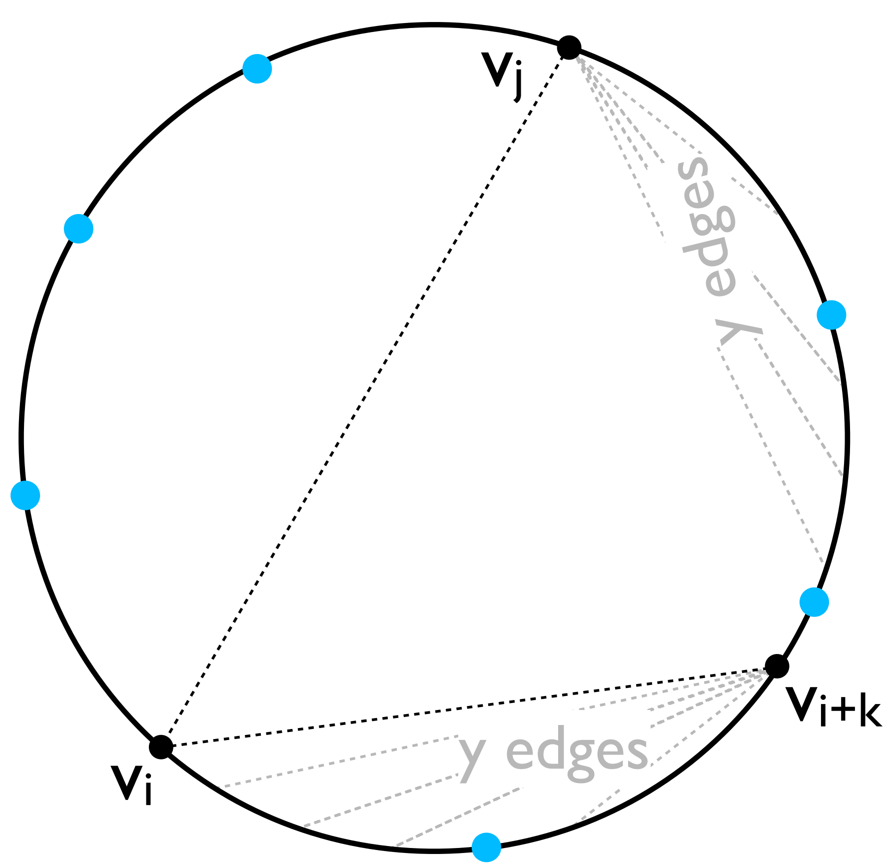


vert	class	neighbors
i	C	$C_A, C_B, C_C, \ldots, C_L, \ldots, C_M, \ldots, C_X, C_Y, C_Z$

- We show:
  LB = -(y+1) $\Rightarrow v_j = v_{i+k}$
- Assume $v_j \neq v_{i+k}$
  but LB = -(y+1)
- v_{i+k} and v_j are same class
  $\Rightarrow v_j$ has y back-edges
  $\Rightarrow$ all back-edges identified

Meeting and Mapping

visibility graph reconstruction III

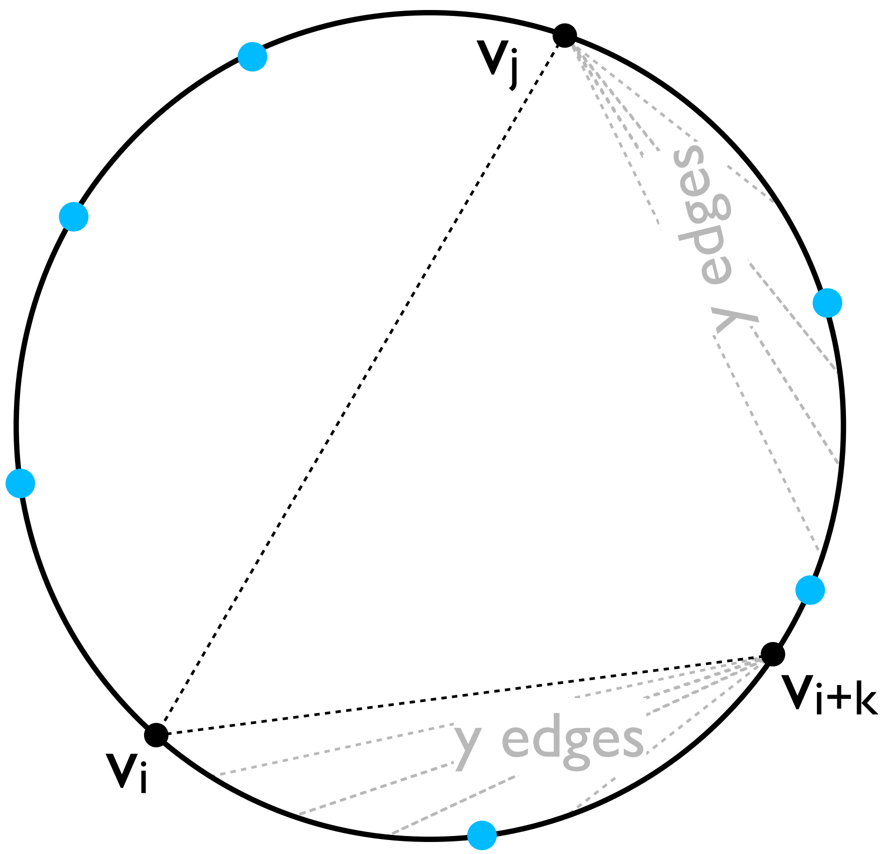


vert	class	neighbors
i	C	$C_A, C_B, C_C, \ldots, C_L, \ldots, C_M, \ldots, C_X, C_Y, C_Z$

- We show:
  $LB = -(y+1) \Rightarrow v_j = v_{i+k}$
- Assume $v_j \neq v_{i+k}$
  but $LB = -(y+1)$
- v_{i+k} and v_j are same class
  $\Rightarrow v_j$ has y back-edges
  $\Rightarrow$ all back-edges identified
- Use C^* as a frame

Meeting and Mapping

visibility graph reconstruction III

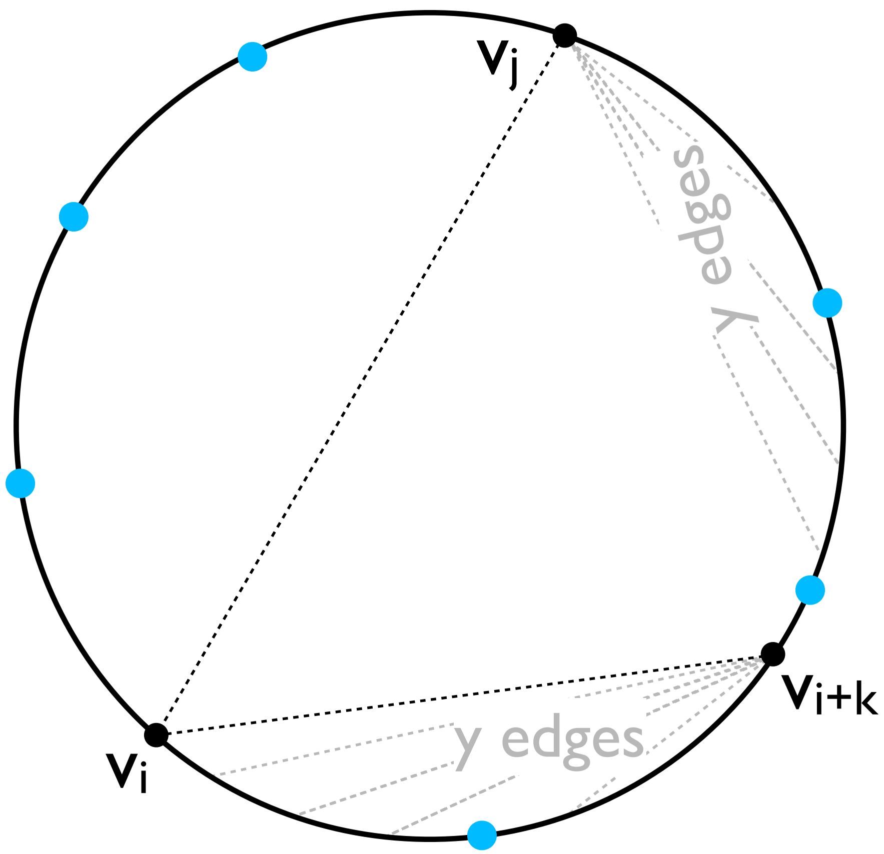


vert	class	neighbors
i	C	$C_A, C_B, C_C, \ldots, C_L, \ldots, C_M, \ldots, C_X, C_Y, C_Z$

- We show:
  LB = -(y+1) $\Rightarrow v_j = v_{i+k}$
- Assume $v_j \neq v_{i+k}$
  but LB = -(y+1)
- v_{i+k} and v_j are same class
  $\Rightarrow v_j$ has y back-edges
  $\Rightarrow$ all back-edges identified
- Use C^* as a frame
  $\Rightarrow$ 2 cases

Meeting and Mapping

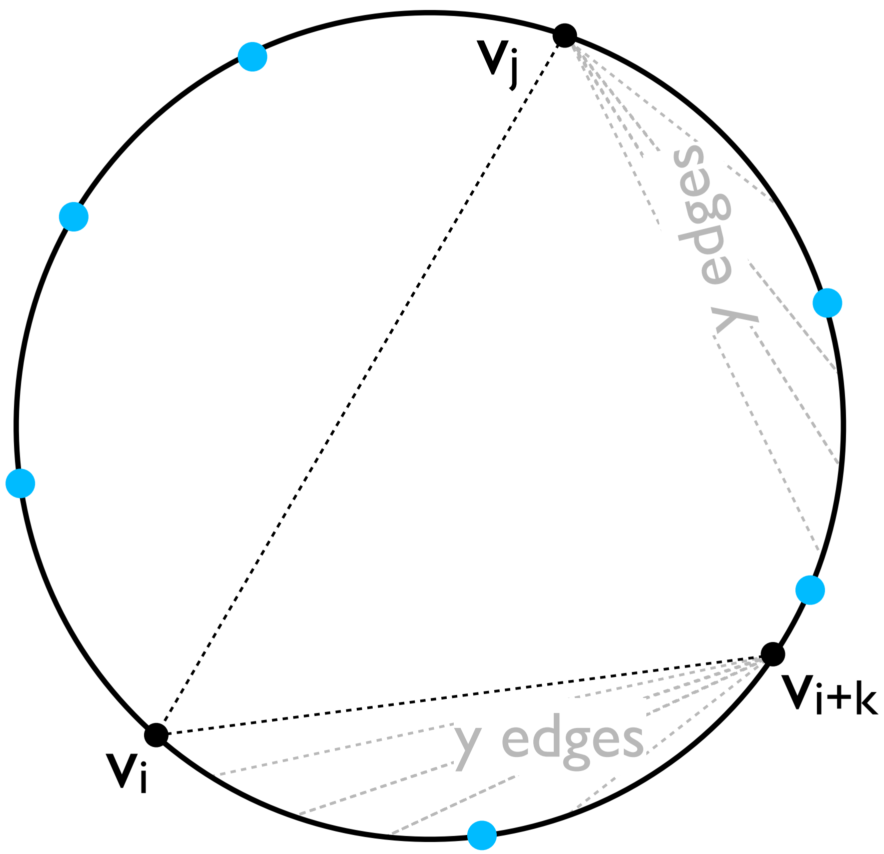
visibility graph reconstruction IV



vert	class	neighbors
i	C	$C_A, C_B, C_C, \ldots, C_L, \ldots, C_M, \ldots, C_X, C_Y, C_Z$

- We show:
  $LB = -(y+1) \Rightarrow v_j = v_{i+k}$
- Use C^* as frame $\Rightarrow$ 2 cases

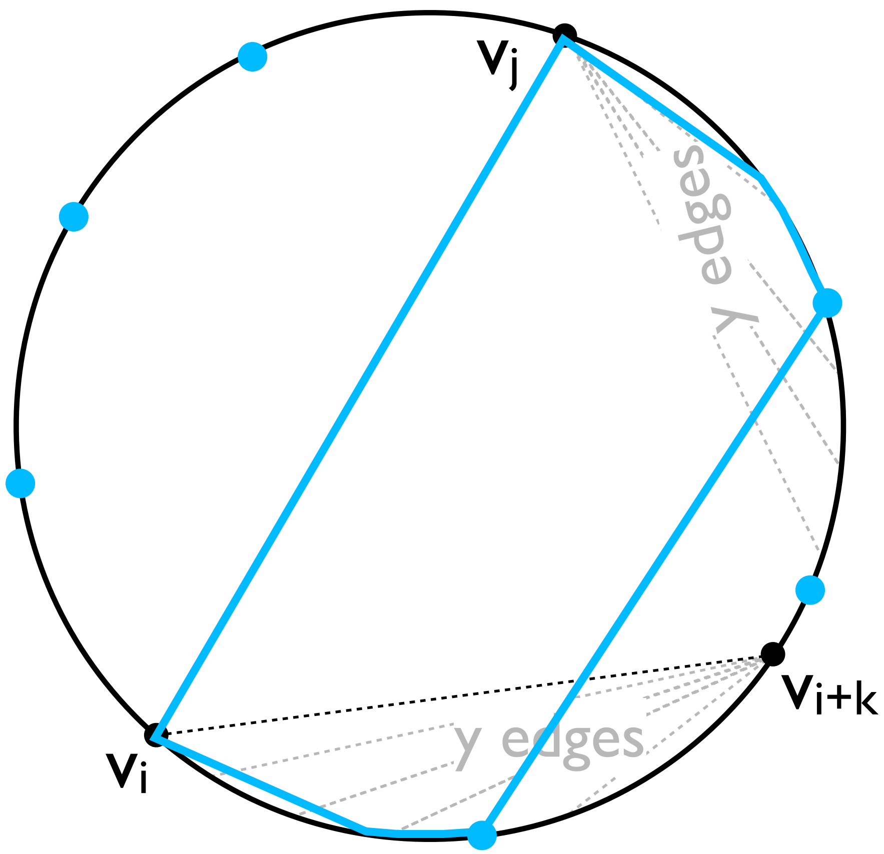
Meeting and Mapping
visibility graph reconstruction IV



vert	class	neighbors
i	C	$C_A, C_B, C_C, \ldots, C_L, \ldots, C_M, \ldots, C_X, C_Y, C_Z$

- We show:
  $LB = -(y+1) \Rightarrow v_j = v_{i+k}$
- Use C^* as frame ⇒2 cases
- Case 1: there are multiple C^* between v_i, v_j

Meeting and Mapping
visibility graph reconstruction IV

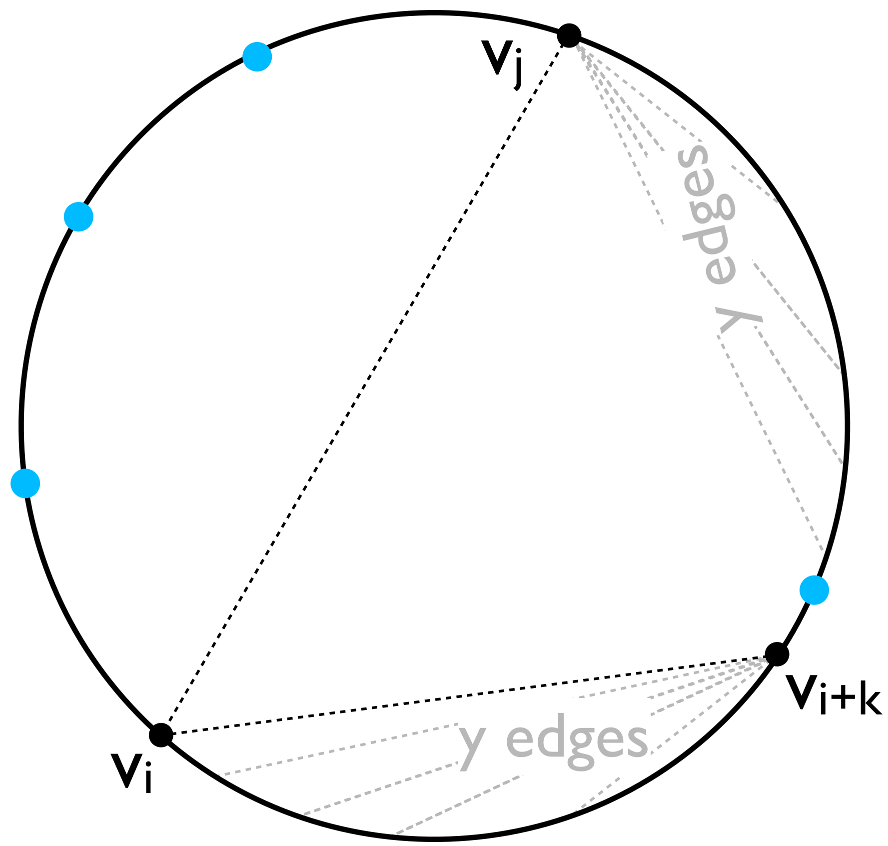


vert	class	neighbors
i	C	$C_A, C_B, C_C, \ldots, C_L, \ldots, C_M, \ldots, C_X, C_Y, C_Z$

- We show:
  LB = -(y+1) $\Rightarrow$ $v_j = v_{i+k}$
- Use C^* as frame $\Rightarrow$2 cases
- Case 1: there are multiple C^* between v_i, v_j
  $\Rightarrow$forbidden polygon ↯

Meeting and Mapping

visibility graph reconstruction IV

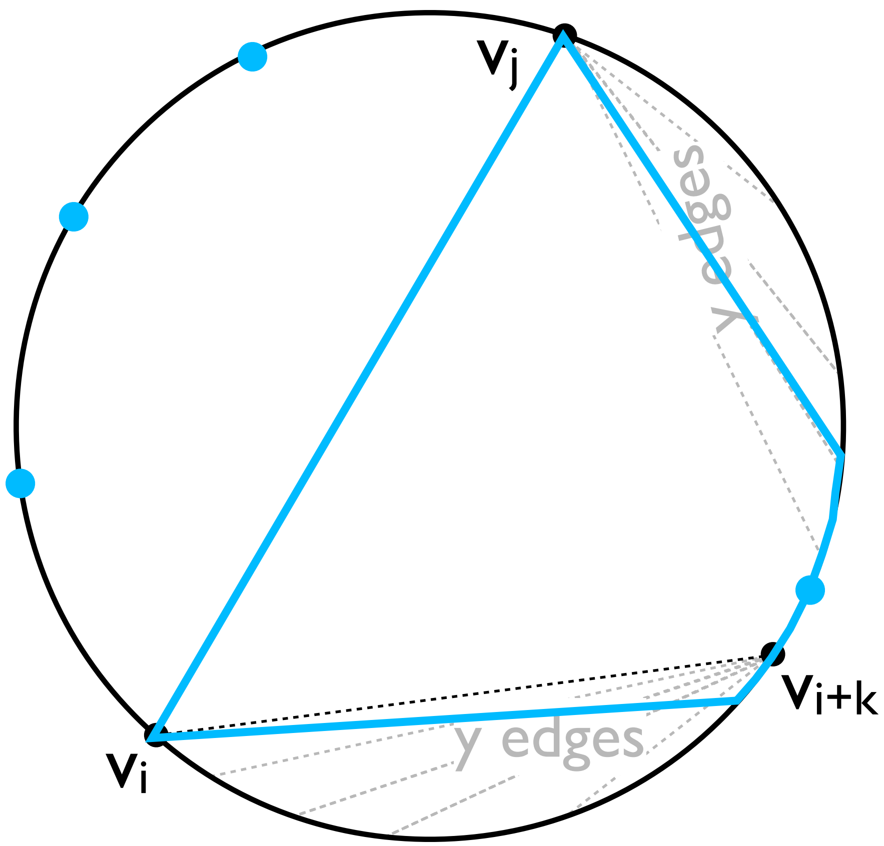


vert	class	neighbors
i	C	$C_A, C_B, C_C, \ldots, C_L, \ldots, C_M, \ldots, C_X, C_Y, C_Z$

- We show:
  $LB = -(y+1) \Rightarrow v_j = v_{i+k}$
- Use C^* as frame ⇒2 cases
- Case I: there are multiple C^* between v_i, v_j
  ⇒forbidden polygon ↯
- Case II: there is one C^*

Meeting and Mapping

visibility graph reconstruction IV

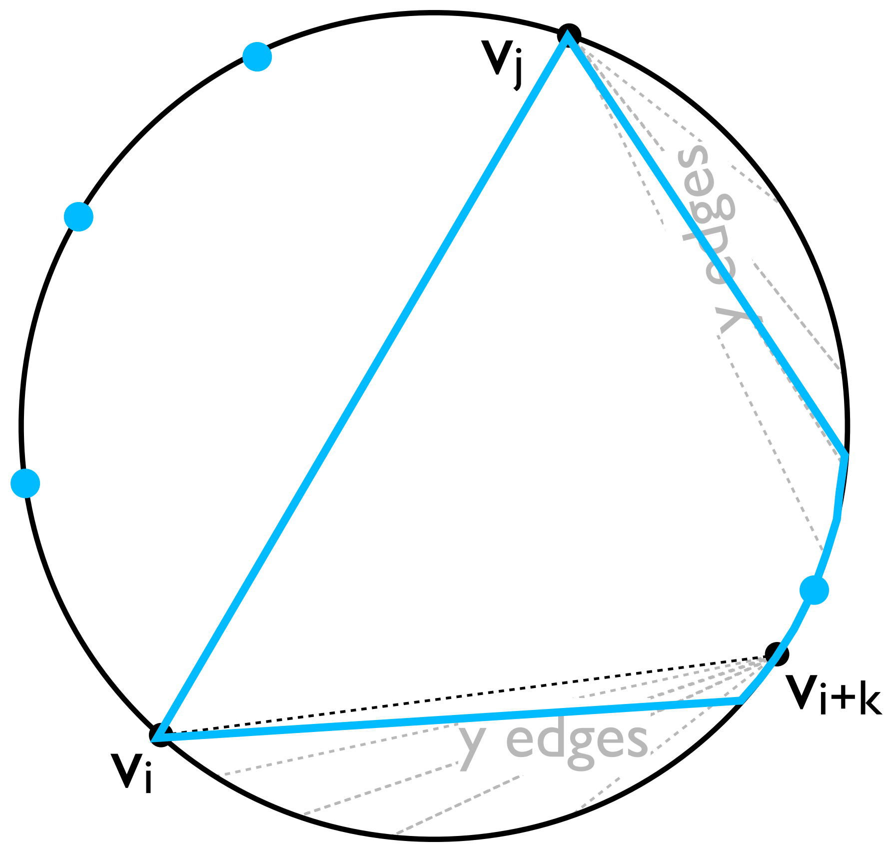


vert	class	neighbors
i	C	$C_A, C_B, C_C, \ldots, C_L, \ldots, C_M, \ldots, C_X, C_Y, C_Z$

- We show:
  $LB = -(y+1) \Rightarrow v_j = v_{i+k}$
- Use C^* as frame ⇒2 cases
- Case 1: there are multiple C^* between v_i, v_j
  ⇒forbidden polygon ↯
- Case II: there is one C^*
  ⇒forbidden polygon ↯

Meeting and Mapping

visibility graph reconstruction IV



vert	class	neighbors
i	C	$C_A, C_B, C_C, \ldots, C_L, \ldots, C_M, \ldots, C_X, C_Y, C_Z$

- We show:
  $LB = -(y+1) \Rightarrow v_j = v_{i+k}$
- Use C^* as frame $\Rightarrow$ 2 cases
- Case I: there are multiple C^* between v_i, v_j
  $\Rightarrow$ forbidden polygon ↯
- Case II: there is one C^*
  $\Rightarrow$ forbidden polygon ↯

$\Rightarrow$ Criterion for deciding $v_j = v_{i+k}$

Summary

Summary

Summary

- The class of a vertex can be determined in finite time by a look-back robot

Summary

- The class of a vertex can be determined in finite time by a look-back robot
- Every polygon has a class C^* that forms a clique

Summary

- The class of a vertex can be determined in finite time by a look-back robot
- Every polygon has a class C^* that forms a clique
- Because of this, robots can always meet "easily"

Summary

- The class of a vertex can be determined in finite time by a look-back robot
- Every polygon has a class C^* that forms a clique
- Because of this, robots can always meet "easily"
- C^* can be used as frame to infer the visibility graph

Thank you!