Graph Visualization:

Topology-Shape-Metrics

Topology-Shape-Metrics

1. Background
    - Graphs and Graph Drawings
    - Planarity
    - Topology

2. The topology-shape-metrics approach
    1. Topology
    2. Shape
    3. Metrics

3. Remarks

Background: **Graphs and Graph Drawings**

A **graph** as an adjacency matrix

	0	1	2	3	4	5
0	0	1	0	1	0	1
1	1	0	1	0	1	0
2	0	1	0	1	0	1
3	1	0	1	0	1	0
4	0	1	0	1	0	1
5	1	0	1	0	1	0

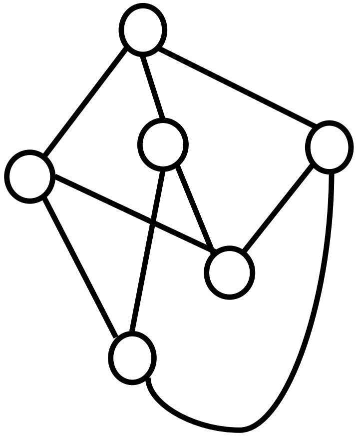
The same graph as an adjacency list

vertex	Adjacent vertices
0	1, 3, 5
1	0, 2, 4
2	1, 3, 5
3	0, 2, 4
4	1, 3, 5
5	0, 2, 4

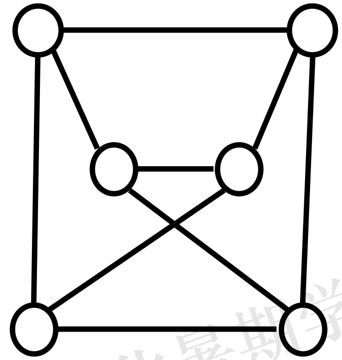
A *graph* has
- no geometry
- no graphics

It is *purely combinatorial* information.

A **graph drawing**



Another drawing of
the same graph



A graph drawing has *layout*

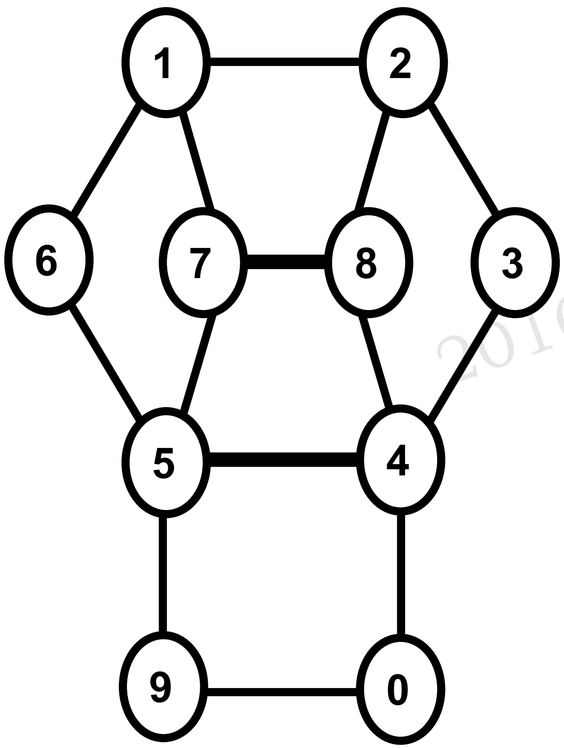
- A position for each vertex
- A route for each edge

It is combinatorial *plus geometric* information.

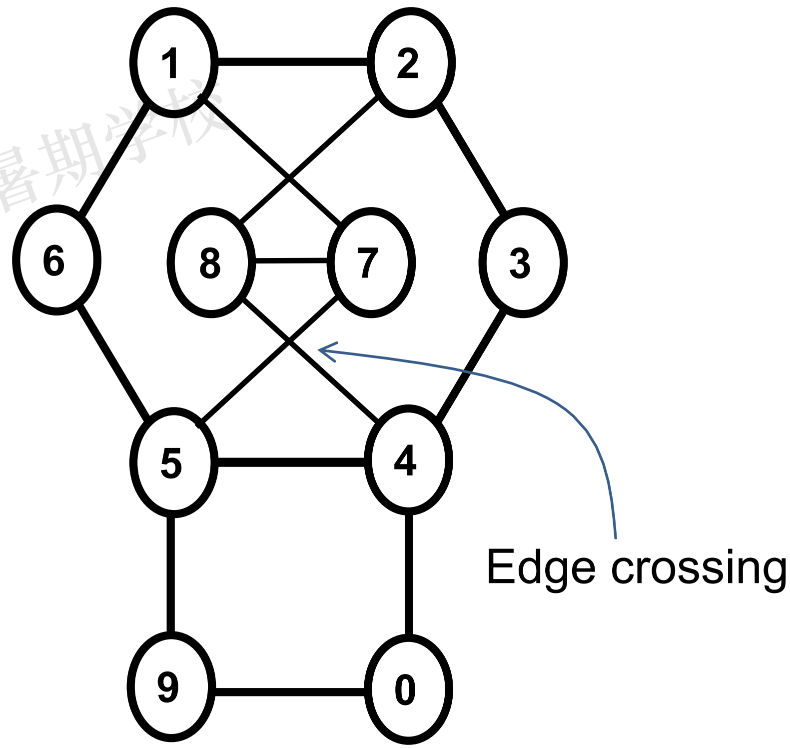
Background: **Planarity**

A graph *drawing*
is *planar* if it has *no edge crossings*.

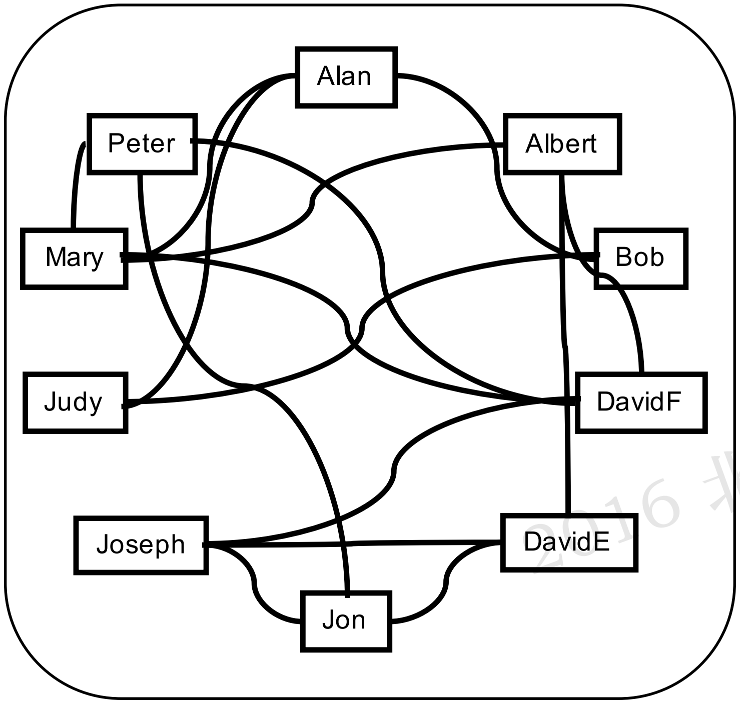
A planar graph drawing



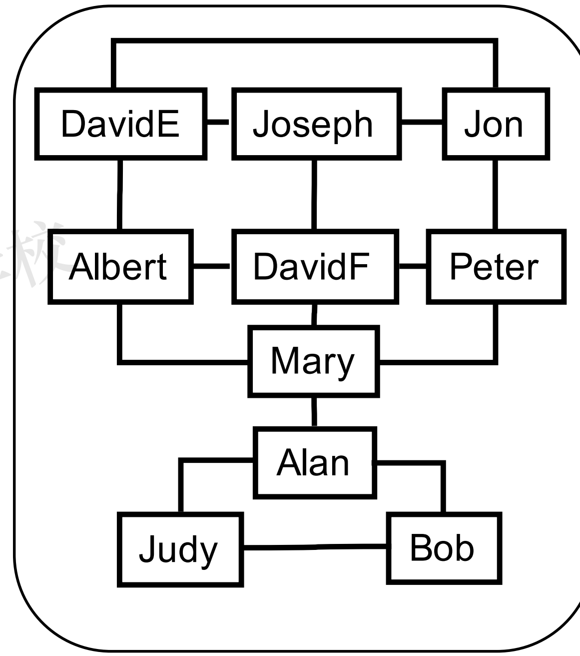
A non-planar graph drawing



Note: Planar drawings make beautiful pictures



Bad picture



Good picture

A *graph*
is *planar*
if it *can be* drawn without edge crossings.

A planar graph G_1

	0	1	2	3	4	5	6	7	8	9
0					1					1
1			1				1	1		
2		1		1					1	
3			1		1					
4	1			1		1			1	
5					1		1	1		1
6		1				1				
7		1				1			1	
8			1		1			1		
9	1					1				

A non-planar graph G_2

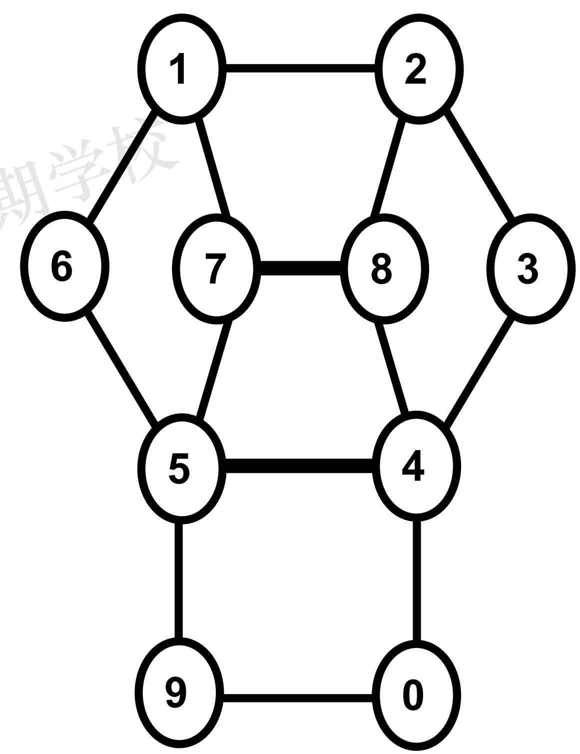
	0	1	2	3	4	5
0		1		1		1
1	1		1		1	
2		1		1		1
3	1		1		1	
4		1		1		1
5	1		1		1	

Lemma:

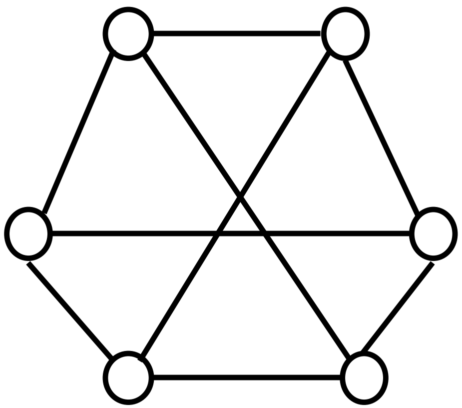
G_1 is planar.

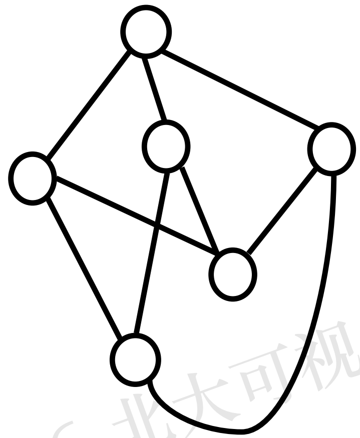
	0	1	2	3	4	5	6	7	8	9
0					1					1
1			1				1	1		
2		1		1					1	
3			1		1					
4	1			1		1			1	
5					1		1	1		1
6		1				1				
7		1				1			1	
8			1		1			1		
9	1					1				

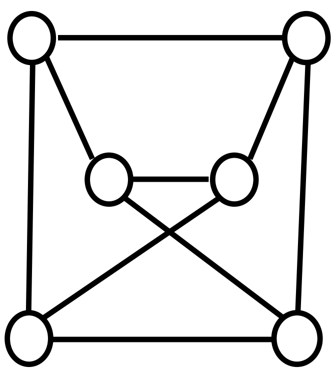
Proof



A graph G is *non-planar*
if *every* drawing of G has edge crossings.







A non-planar graph G_2

	0	1	2	3	4	5
0		1		1		1
1	1		1		1	
2		1		1		1
3	1		1		1	
4		1		1		1
5	1		1		1	

Note: There are many beautiful theorems about planar graphs

Theorem: (from Euler, 1700s) For a planar graph $G = (V, E)$, $|E| \leq 3|V| - 6$. If $|E| = 3|V| - 6$, then G is triconnected and each face of the embedding of G has 3 edges.

Theorem (Kuratowski,1930) A graph is planar if and only if it does not contain a subgraph that is a subdivision of K_5 or $K_{3,3}$.

Theorem: (Steinitz, 1930s) For every triconnected planar graph G, there is a convex polyhedron P in 3D such that the vertex-edge graph of P is isomorphic to G.

Theorem: (Appel-Haken, 1970s) A planar graph can be colored in 4 colors.

Theorem: (Lipton-Tarjan, 1980s) For every planar graph G with n vertices, there is set of $O(\sqrt{n})$ vertices whose removal divides the graph into components of size at most $\frac{2n}{3}$.

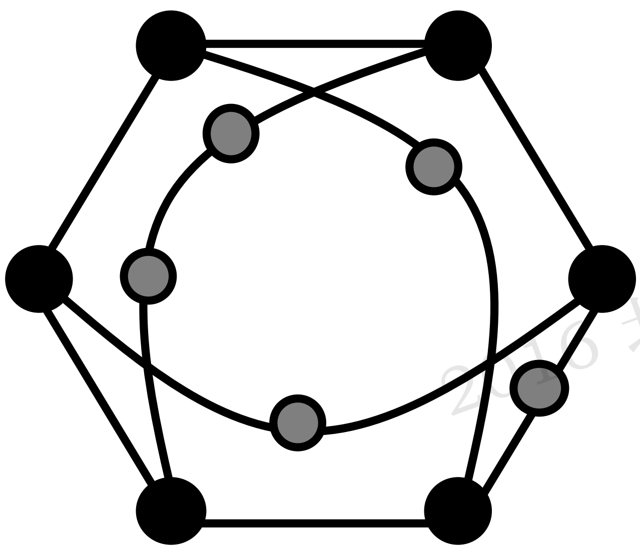
Theorem: (Trivial) A graph is planar if and only if each of its triconnected components is planar.

Graph theorists *love* planar graphs

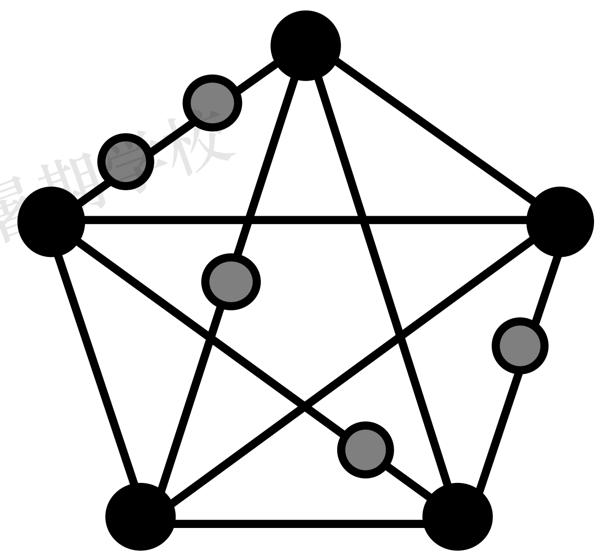
Note: There are many beautiful theorems about planar graphs

<u>Theorem</u> (Kuratowski,1930)

A graph is planar if and only if it does not contain a subgraph that is a subdivision of $\boldsymbol{K_5}$ or $\boldsymbol{K_{3,3}}$.



$\boldsymbol{K_{3,3}}$ (complete bipartite graph on 6 vertices) subdivision



$\boldsymbol{K_5}$ (the complete graph on 5 vertices) subdivision

Theorem (Kuratowski,1930)
A graph is planar if and only if it does not contain a subgraph that is a subdivision of K_5 or $K_{3,3}$.

Corollary
G_2 is non-planar

Proof:
G_2 is $K_{3,3}$.

G_2

	0	1	2	3	4	5
0		1		1		1
1	1		1		1	
2		1		1		1
3	1		1		1	
4		1		1		1
5	1		1		1	

Remarks: ***Planar*** graphs and ***real-world*** graphs

- Most real-world graphs are not planar
- But most are “nearly” planar in some sense:
  - deletion of $\boldsymbol{o}(\boldsymbol{n})$ edges gives a planar graph
  - scale-free networks are locally dense and globally sparse

Planarity testing algorithms

<u>Hopcroft-Tarjan planarity testing algorithm (1974)</u>

- ➢ Tests whether a graph is planar or not, in linear time
- ➢ Very complicated algorithm; implementation difficult
  - First published version incorrect; corrected by Deo (1976)
  - Most implementations incorrect
  - First correct implementation (I believe) 1994.

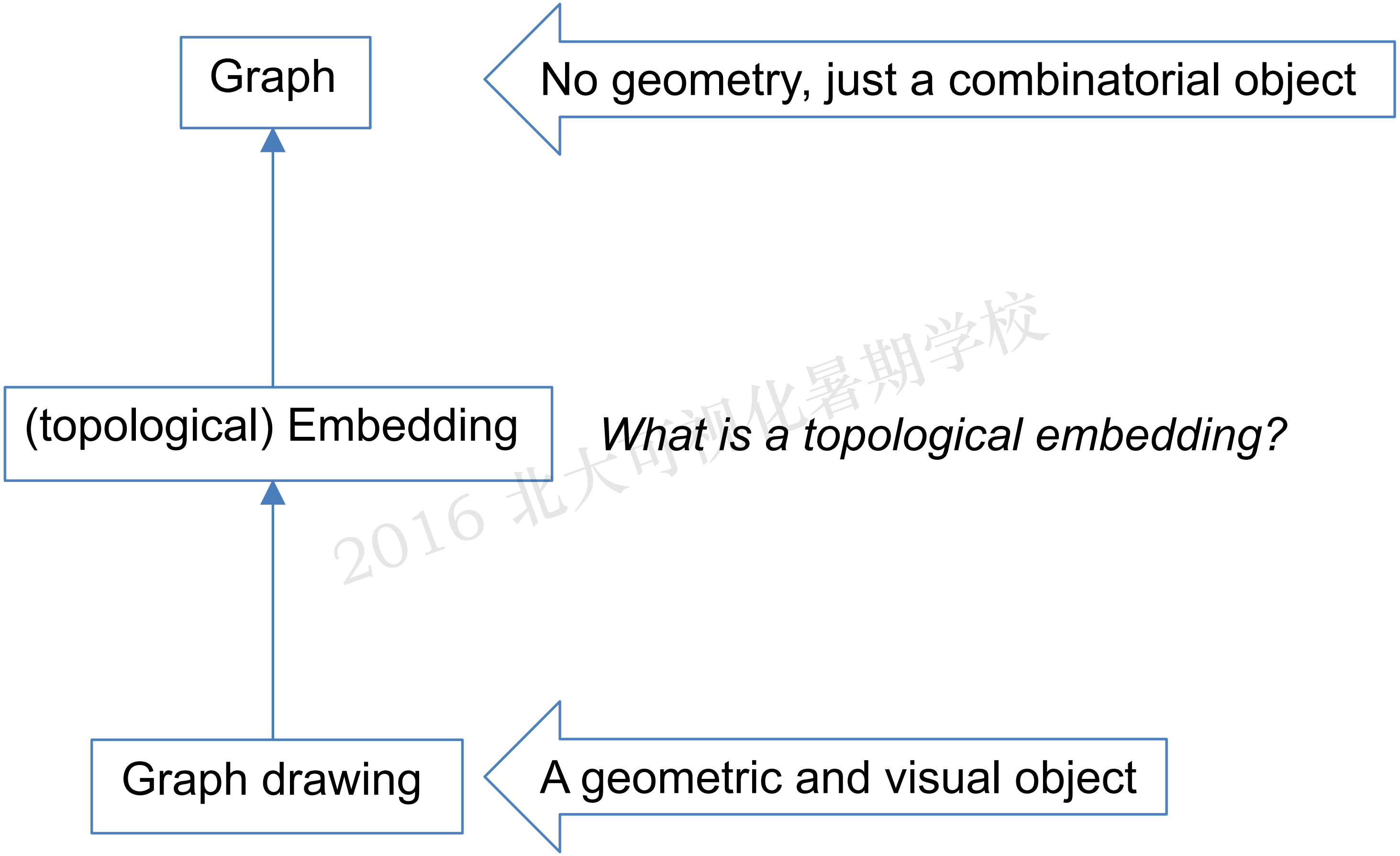
<u>Many subsequent planarity testing algorithms</u>

- ➢ Lempel-Even-Cederbaum 1966
- ➢ Booth-Lueker 1976
- ➢ Rosensthiel-de Frayssieux 1990
- ➢ Hsu/Boyer-Myvold 2000

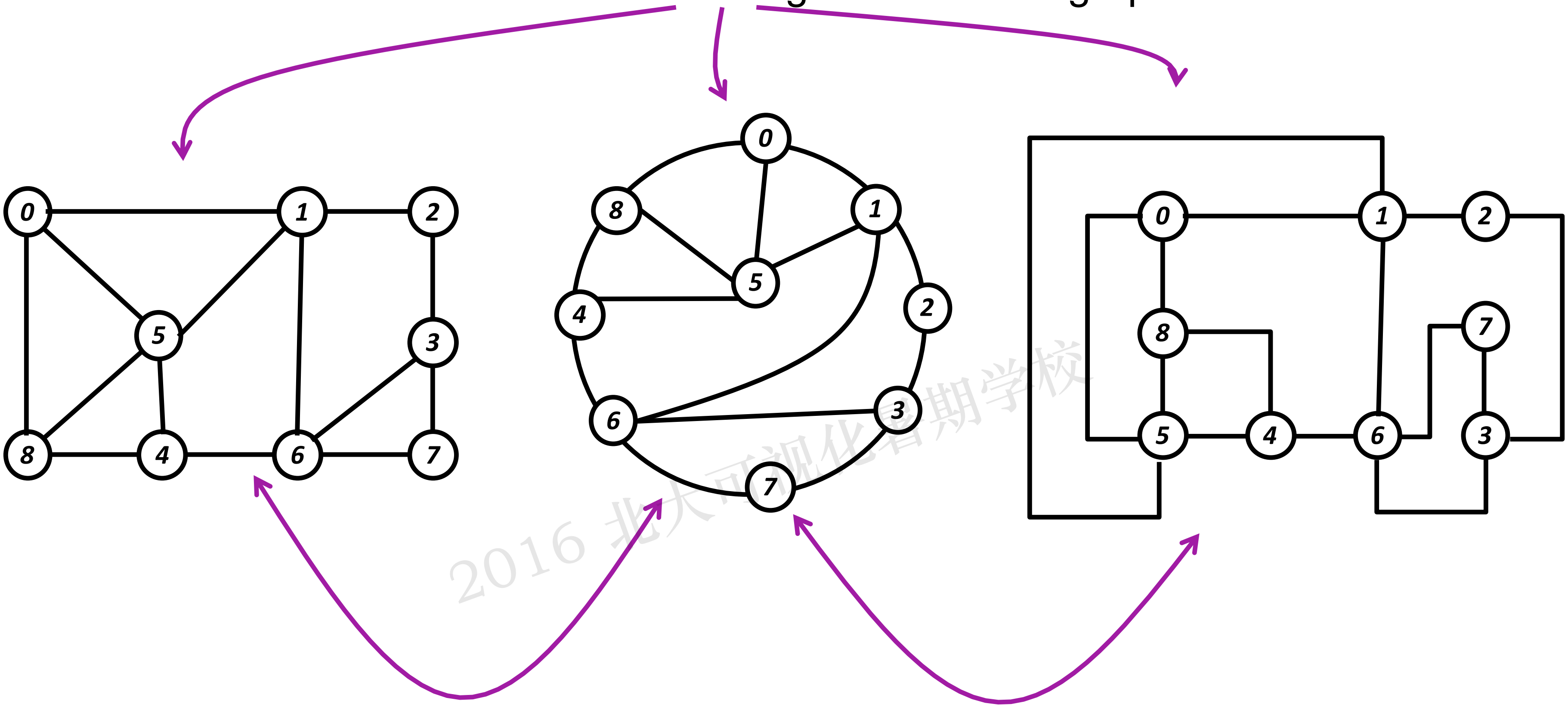
Note:

- ➢ All these planarity testing algorithms are <u>***efficient***</u> and <u>***effective***</u>, but none is <u>***elegant***</u>.
- ➢ Finding an <u>*elegant*</u> linear time planarity testing algorithm is still an unsolved problem.

Background: **Topology**


Graph
No geometry, just a combinatorial object
(topological) Embedding
What is a topological embedding?
Graph drawing
A geometric and visual object

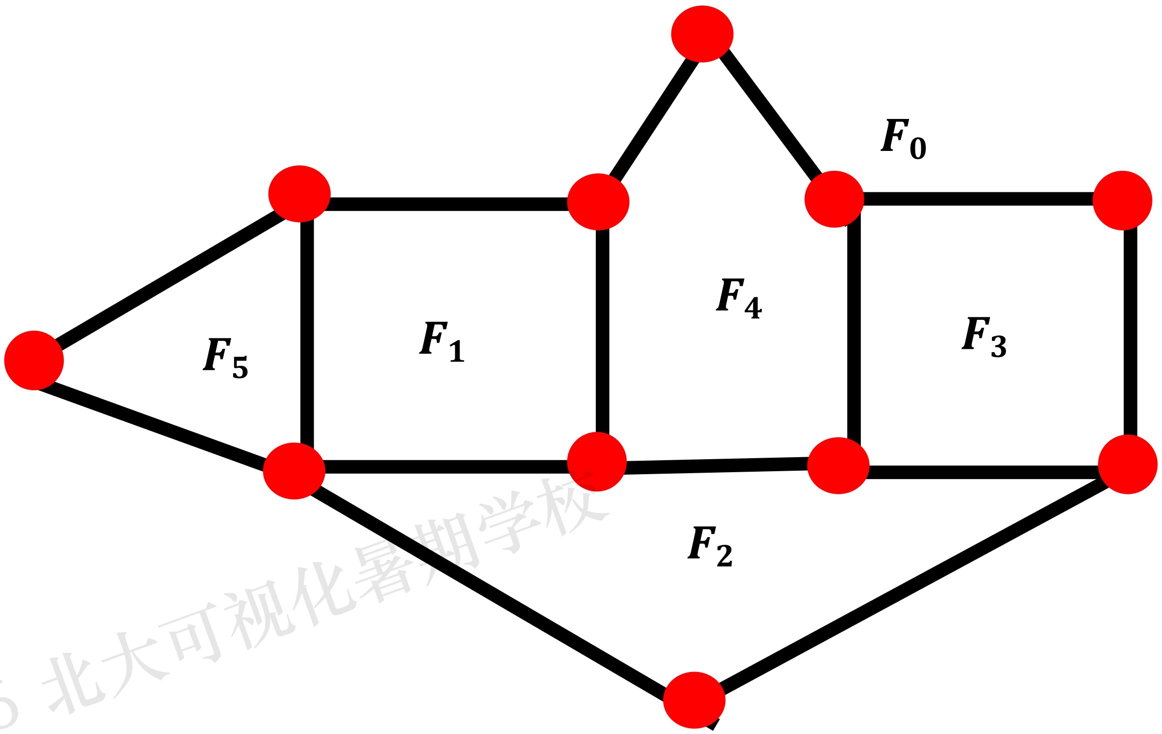
Three different drawings of the same graph

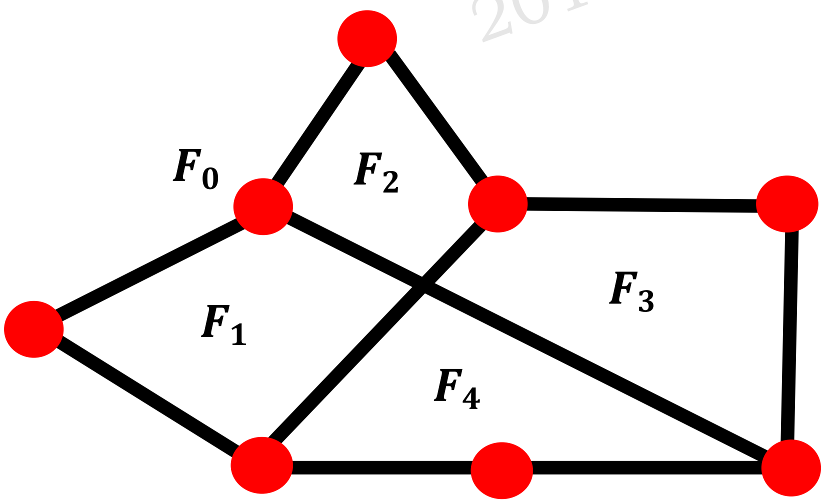


The same topological embedding

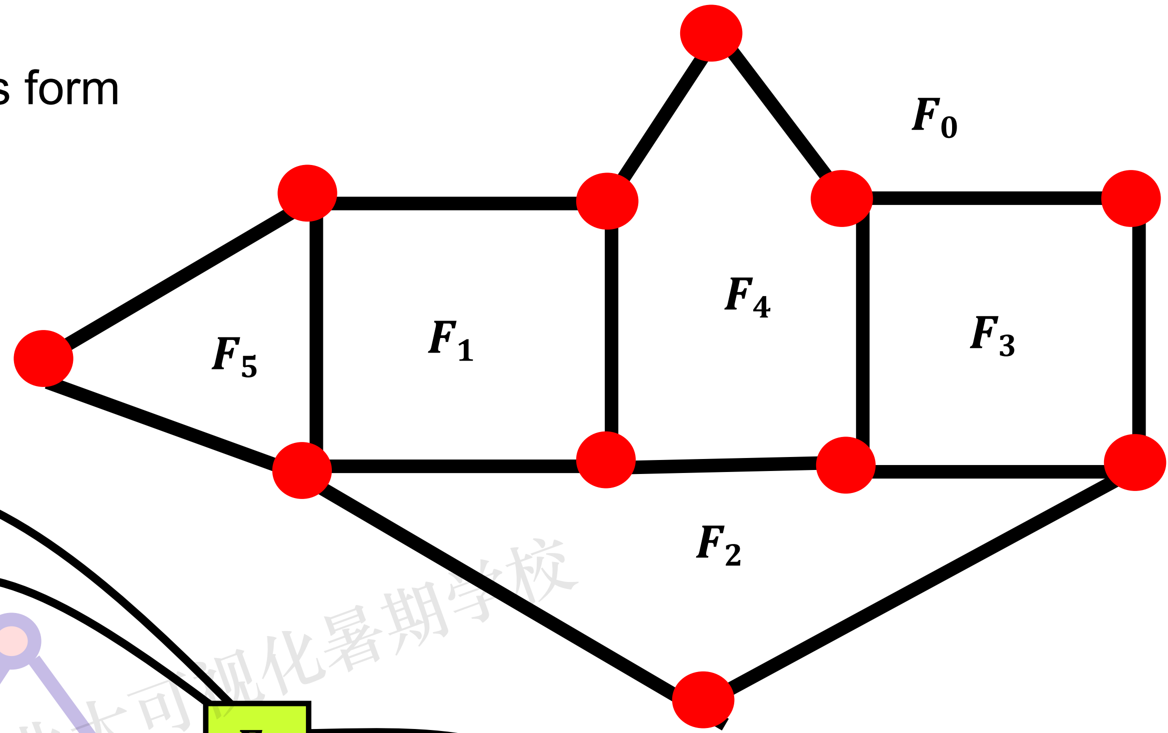
Different topological embeddings

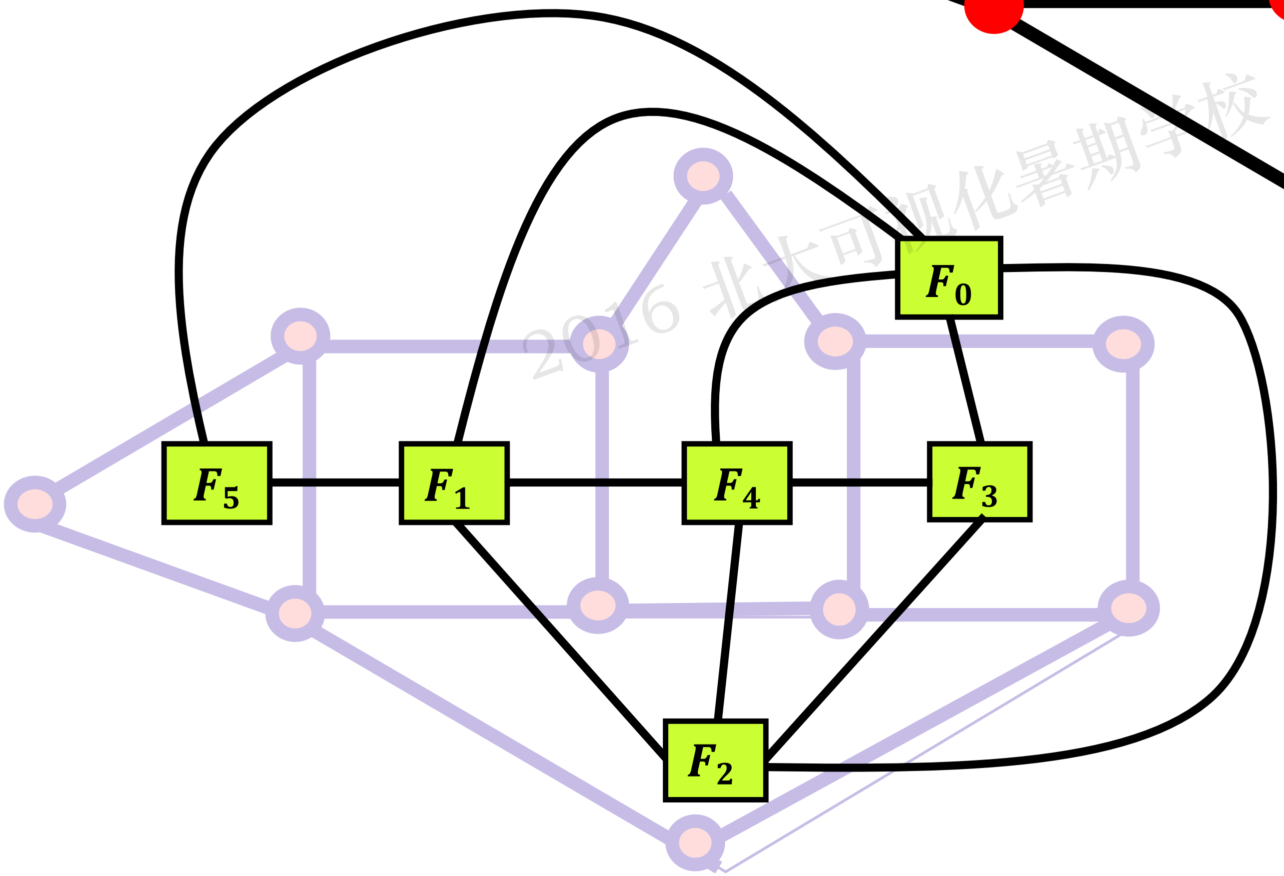
A graph drawing divides the plane into regions called *faces*.





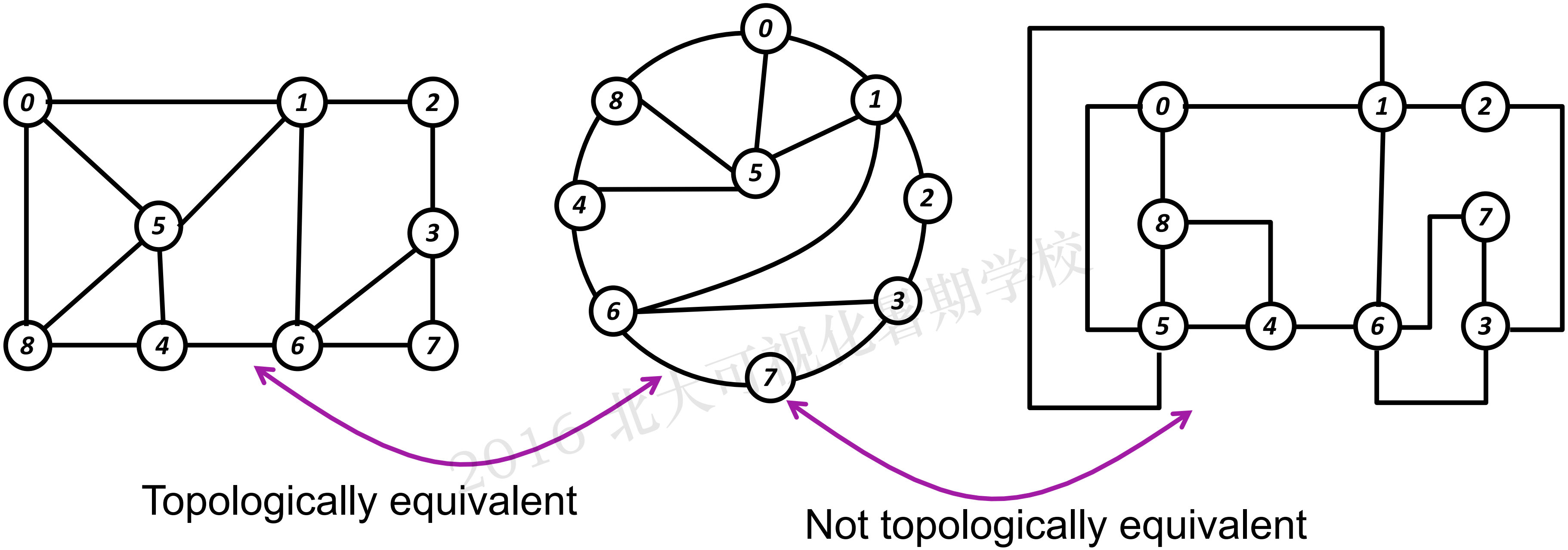
The faces plus their adjacencies form the *dual graph* of the drawing.





Definition:

Two drawings of a graph are *topologically equivalent* if there is a homeomorphism of the plane/sphere that maps one to the other.



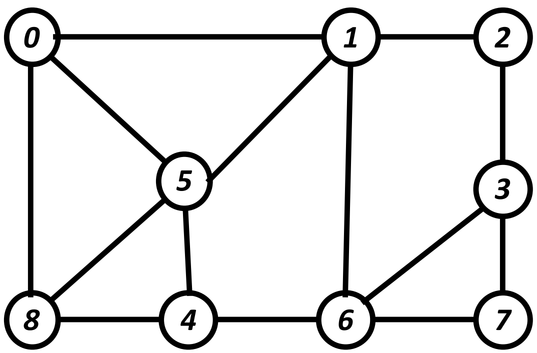
Alternative definitions:

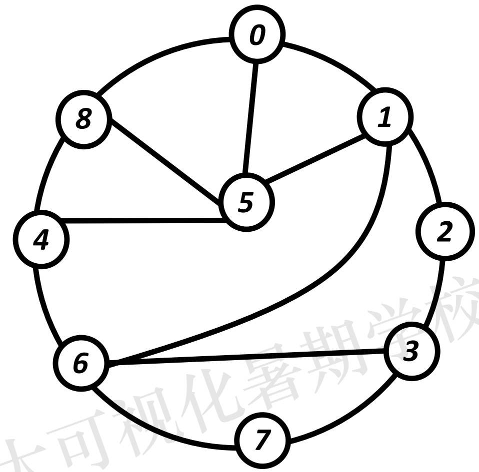
Two drawings of a graph are topologically equivalent if

1. they have the “same” dual graph, or alternatively
2. they have the same clockwise circular ordering of edges around each vertex.

Definition:

Two drawings of a graph are *topologically equivalent* if there is a homeomorphism of the plane/sphere that maps one to the other.





0	8,1,5
1	5,0,2,6
2	1,3
3	6,2,7
4	8,5,6
5	8,0,1,4
6	4,1,3,7
7	6,3
8	0,5,4

Same clockwise circular orderings of edges around each vertex

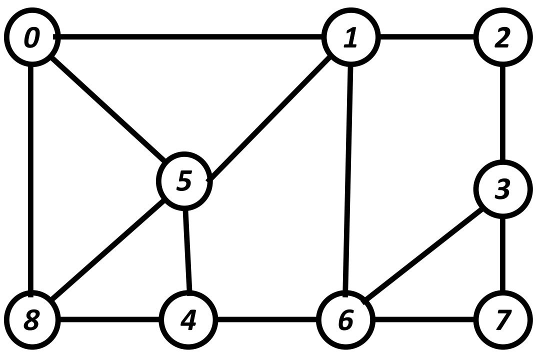
↔

Same topological embedding

0	5,8,1
1	6,5,0,2
2	3,1
3	7,6,2
4	8,5,6
5	4,8,0,1
6	4,1,3,7
7	6,3
8	4,0,5

Definition:

A *(topological) embedding* of a graph G is an equivalence class of drawings of G under topological equivalence.



0	8,1,5
1	5,0,2,6
2	1,3
3	6,2,7
4	8,5,6
5	8,0,1,4
6	4,1,3,7
7	6,3
8	0,5,4

Alternative definitions:

1. An embedding consists of a graph, plus the dual graph.
2. An embedding consists of a graph, plus a clockwise circular ordering of edges around each vertex.

A data structure for planar embeddings:

a) List of vertices.
b) For each vertex u, a circular list of vertices v adjacent to u.

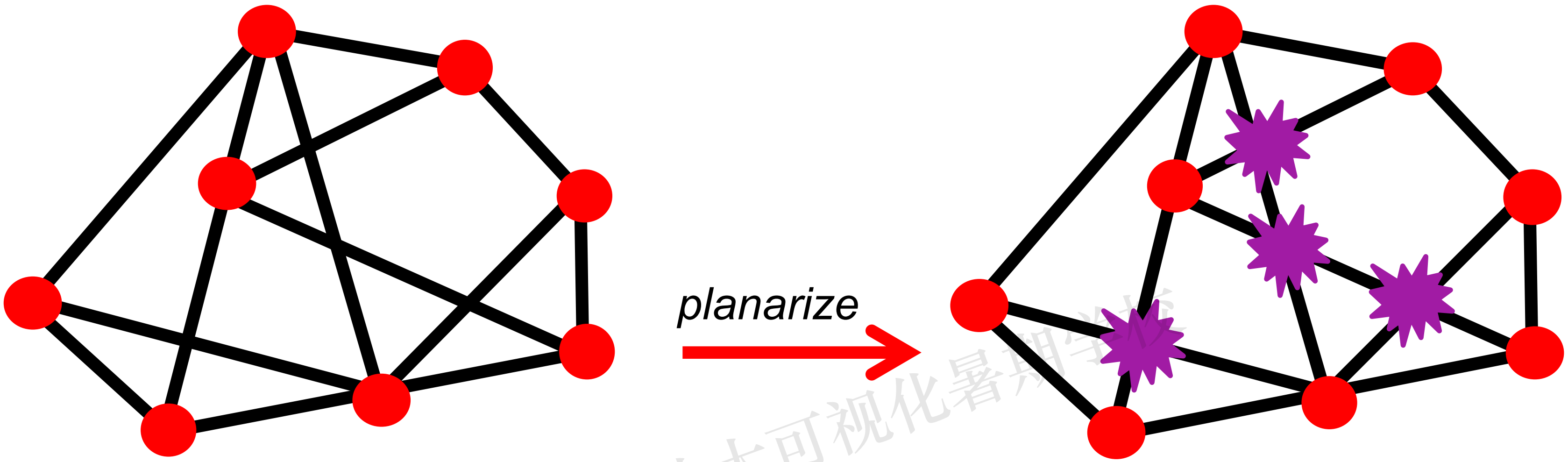
Planar embedding algorithms

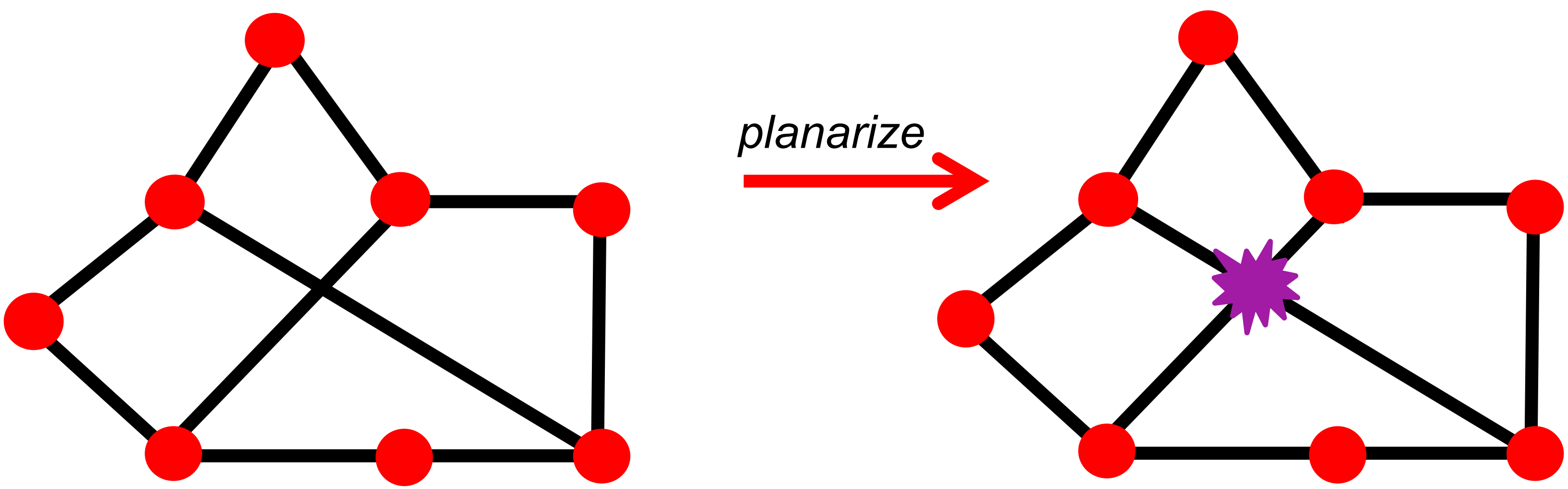
- Most planarity testing algorithms "can be adjusted" to output a planar embedding of a planar graph in linear time.
- All these algorithms are efficient and effective; none is elegant.

Background: **Planarization**

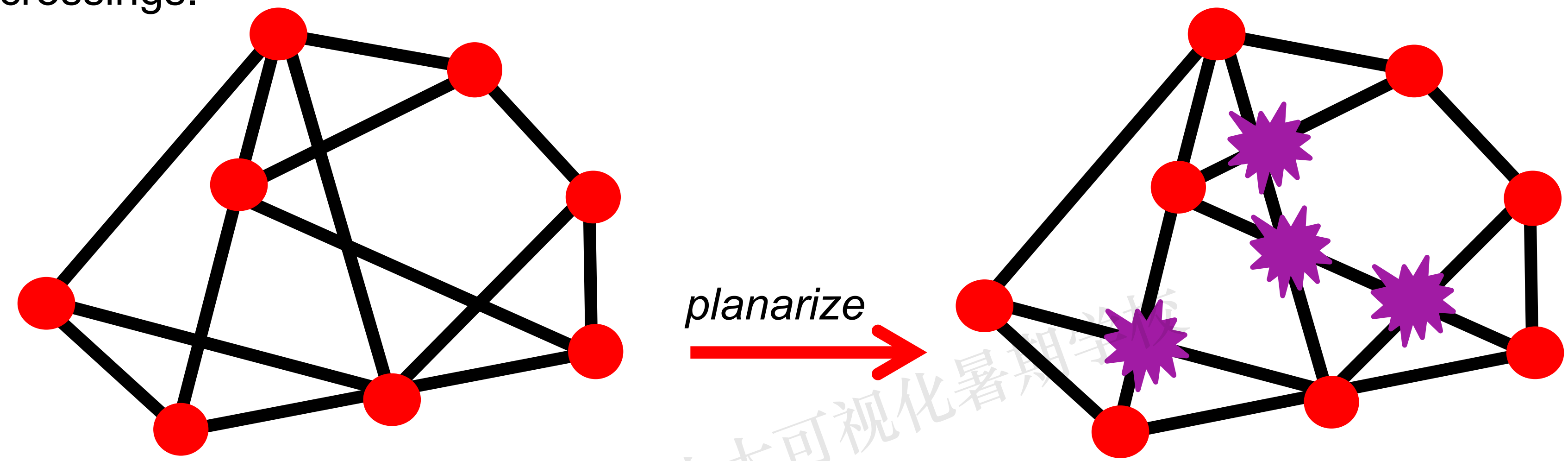
a) Planarize a topological embedding
b) Planarize a graph

A *non-planar topological embedding* can be *planarized* by placing *dummy vertices* at the edge crossings.





A *non-planar embedding* can be *planarized* by placing *dummy vertices* at the edge crossings.



Note:

- Planarization of a topological embedding can be done in linear time.
- The concept of planarization allows us to apply terminology, definitions, and data structures about planar embeddings to non-planar embeddings.
- If the number of crossing points is small, then planarization might not increase asymptotic time complexity of algorithms.
- Normally, we use planarization to give a data structure for a non-planar embedding.

A *non-planar graph* can be *planarized* by placing *gluing* independent edges together.

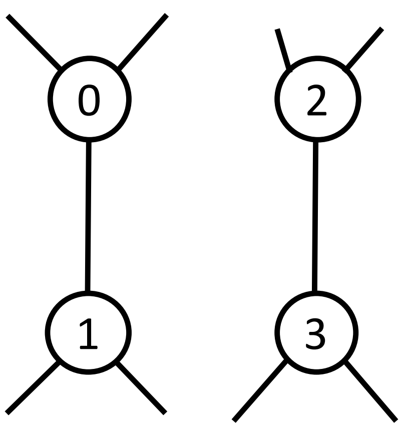
A non-planar graph

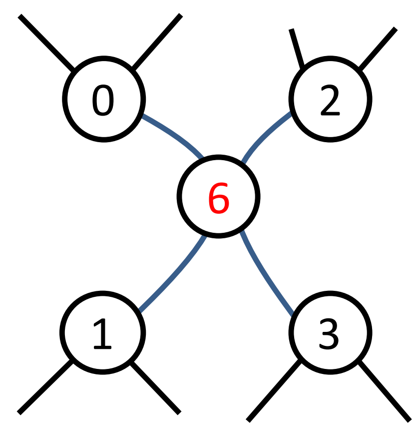
vertex	Adjacent vertices
0	1, 3, 5
1	0, 2, 4
2	1, 3, 5
3	0, 2, 4
4	1, 3, 5
5	0, 2, 4

Glue edges (0,1) and (2,3) →

A planar graph

vertex	Adjacent vertices
0	6, 3, 5
1	6, 2, 4
2	1, 6, 5
3	0, 6, 4
4	1, 3, 5
5	0, 2, 4
6	0,1,2,3

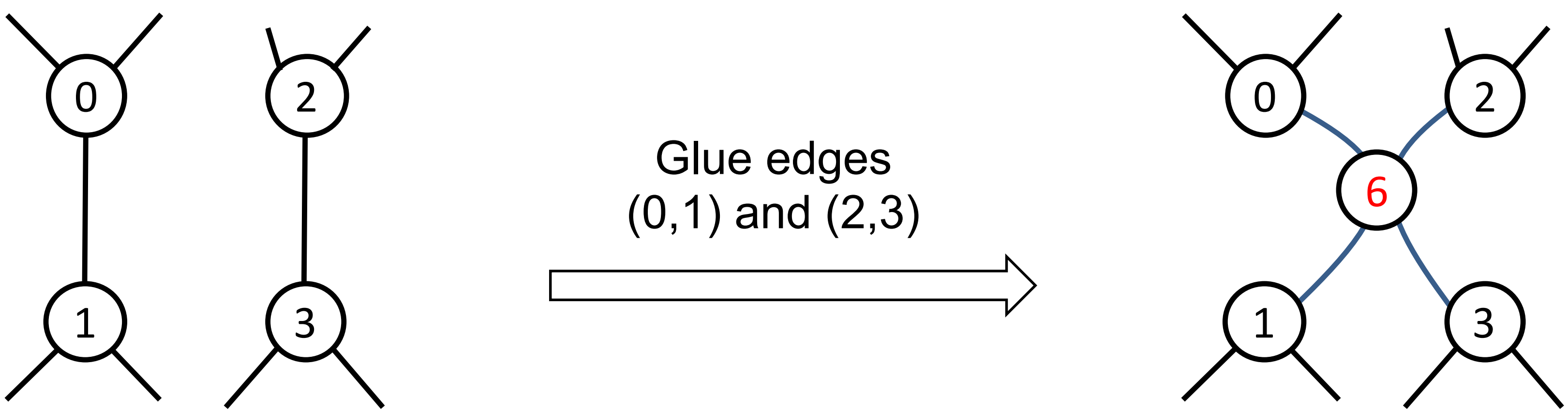




A *non-planar graph* can be *planarized* by placing *gluing* independent edges together.

Theorem: For every graph G, there is a sequence of edge-gluings that makes G planar.

Theorem: Finding a minimum sequence of edge-gluings that makes a graph planar is an NP-complete problem.



3. Graph drawings

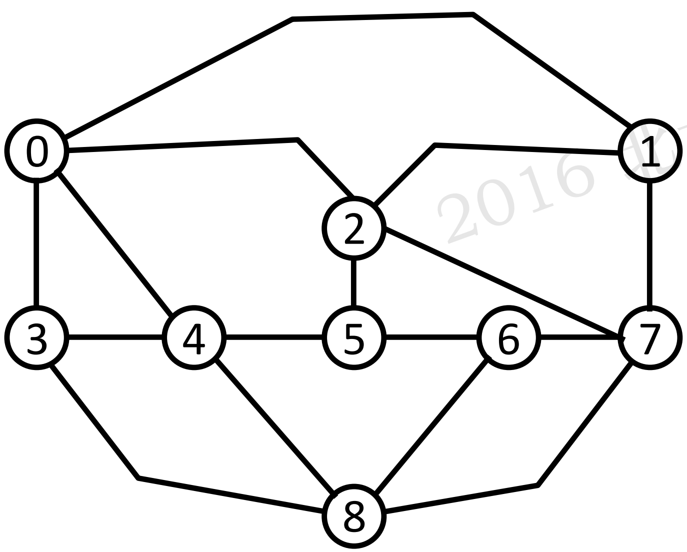
A *drawing* of a graph $G = (V, E)$ consists of

- a location $p(u)$ for each vertex u, and
- a Jordan arc $c(u, v)$ for each edge (u, v) such that the endpoints of $c(u, v)$ are $p(u)$ and $p(v)$.
- (plus a lot of non-degeneracy conditions)

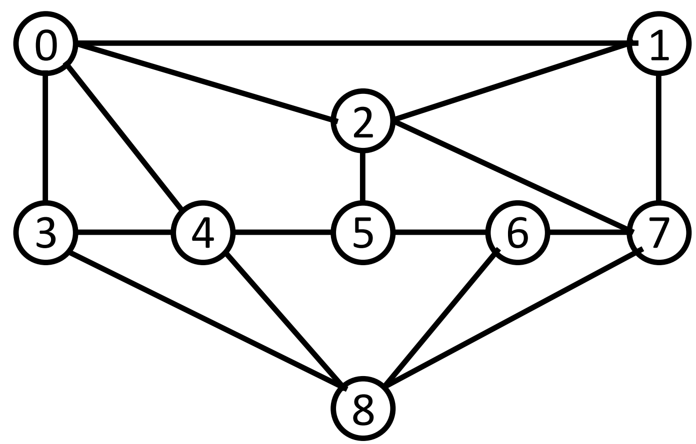
There are many kinds of graph drawings

- *Grid drawing*: vertices (and edge bends?) are located at integer grid points
- *Polyline drawing*: edges are polylines
- *Straight-line drawing*: edges are straight line segments
- *Orthogonal drawing*: edges are polylines made up of vertical and horizontal line segments
- ….
- …..

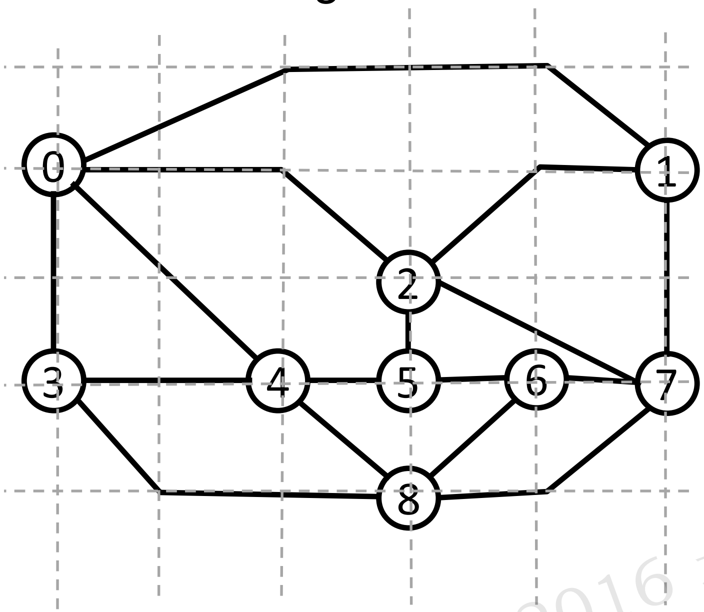
Polyline drawing



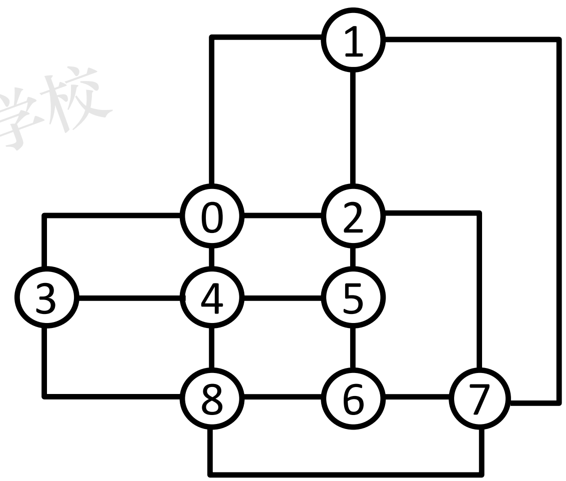
Straight-line drawing



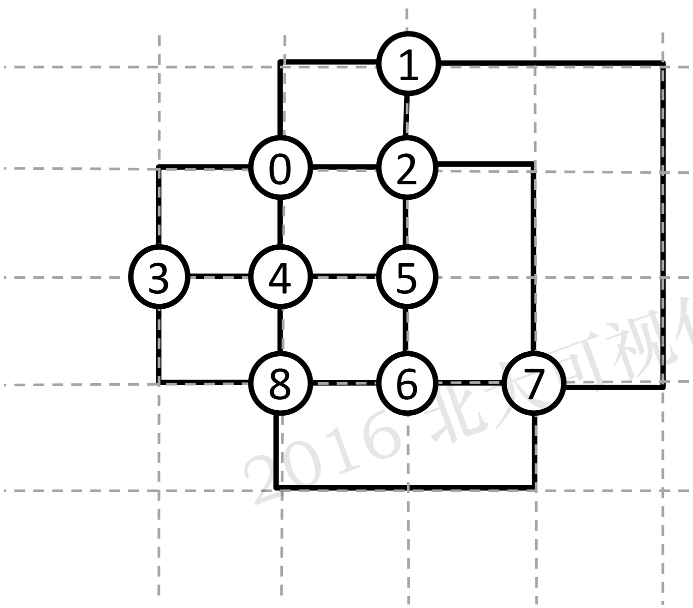
Grid drawing

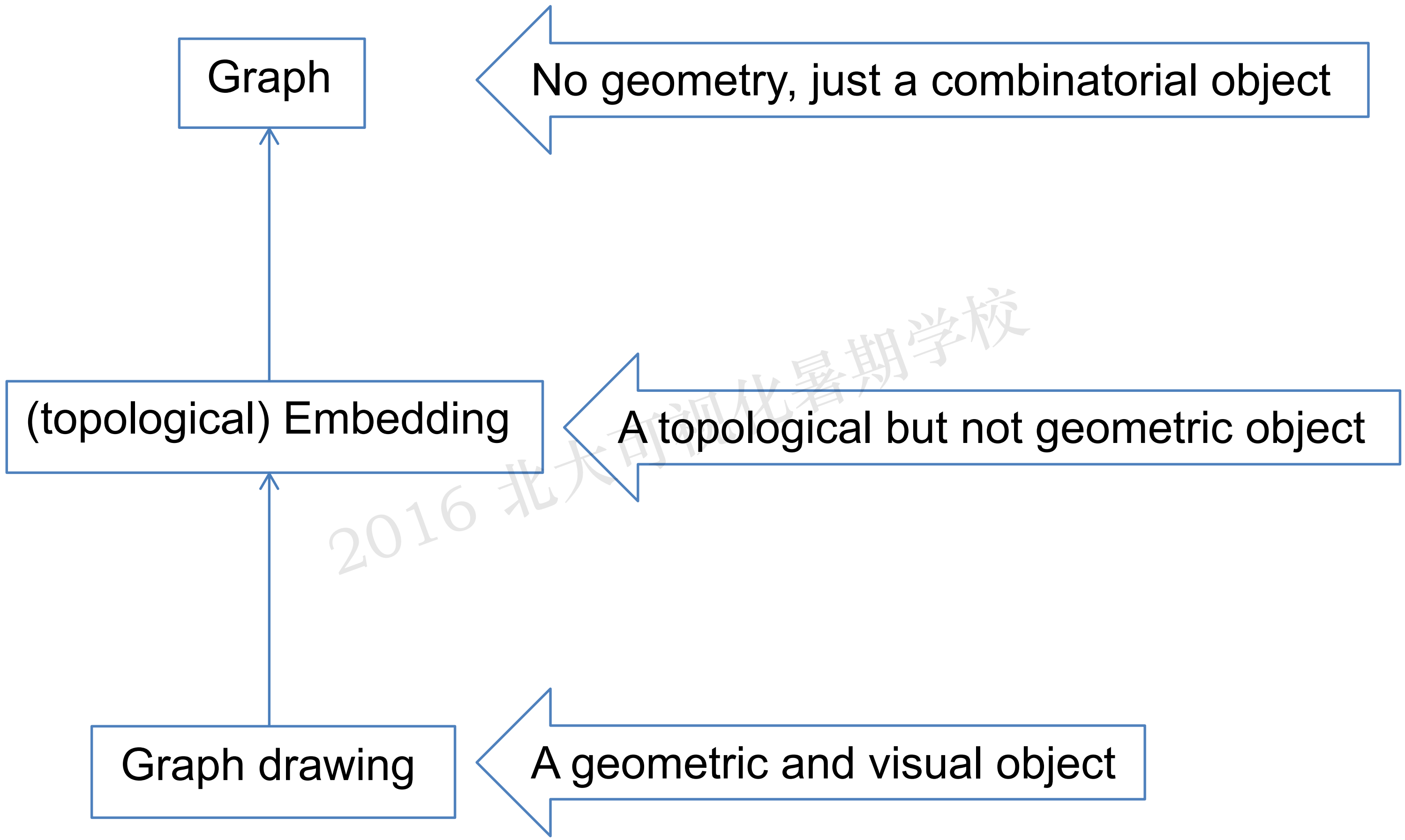


Orthogonal drawing



Orthogonal grid drawing




Graph
No geometry, just a combinatorial object
(topological) Embedding
A topological but not geometric object
Graph drawing
A geometric and visual object

Topology-shape-metrics approach

Topology-shape-metrics method:

Input: a graph $\boldsymbol{G}$

Algorithm:

1. *<u>Topology</u>*: Compute a good topological *<u>embedding</u>* of $\boldsymbol{G}$
2. *<u>Shape</u>*: Compute a good orthogonal *<u>shape</u>* for this topological embedding
3. *<u>Metrics</u>*: Compute a good orthogonal grid drawing of $\boldsymbol{G}$

Output: an *orthogonal grid* drawing of $\boldsymbol{G}$

<u>Aim</u>: produce a topological embedding with few edge crossings

1. Compute a good topological *embedding* of G

Input: a graph $G = (V, E)$

a) Compute a planar subgraph $G' = (V, E')$, where E' is a subset of E, such that $|E'|$ is as large as possible.

b) Compute a planar embedding G'' of G'.

c) Insert the edges of $E - E'$ into G'', creating as few crossings as possible, to create an embedding G''' of G.

Output: an embedding of G with few crossings

Use a maximum-planar-subgraph method

We need solutions for a difficult problem:

Maximum Planar Subgraph (MPS)

Input: a graph $\boldsymbol{G}$

Output: a planar subgraph of $\boldsymbol{G}$ with a maximum number of edges.

Note

- The Maximum Planar Subgraph problem is NP-complete
- Many heuristic approaches have been investigated, implemented, and tested over at least the last 30 years

One successful approach to MPS so far is *integer linear programming*

Integer Linear Program for the Maximum Planar Subgraph problem

Given a graph $\boldsymbol{G = (V, E)}$:

Variables

$\boldsymbol{x_e}$ for each edge $\boldsymbol{e \in E}$

Objective

Maximize $\sum_{e \in E} x_e$

Constraints

a) $\boldsymbol{x_e \in \{0, 1\}}$

b) For each Kuratowski subgraph $\boldsymbol{K = (V_K, E_K)}$ of $\boldsymbol{G}$:

$$\sum_{e \in E_K} x_e < |E_K|$$

Interpetation:

$$x_e = \begin{cases} 1 & \text{if } e \in E' \\ 0 & \text{otherwise} \end{cases}$$

Algorithm:

- Use a traditional “branch&cut” approach, with cutting planes from the many theorems on planar graphs.

1. Compute a good topological *embedding* of G

Input: a graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$

a) Compute a planar subgraph $\boldsymbol{G}' = (\boldsymbol{V}, \boldsymbol{E}')$, where $\boldsymbol{E}'$ is a subset of $\boldsymbol{E}$, such that $|\boldsymbol{E}'|$ is as large as possible.

b) Compute a planar embedding $\boldsymbol{G}''$ of $\boldsymbol{G}'$.

c) Insert the edges of $\boldsymbol{E} - \boldsymbol{E}'$ into $\boldsymbol{G}''$, creating as few crossings as possible, to create an embedding $\boldsymbol{G}'''$ of $\boldsymbol{G}$.

Output: an embedding of $\boldsymbol{G}$ with few crossings

Use planar embedding algorithms

- For example, a variation on the Hopcroft-Tarjan planarity algorithm

1. Compute a good topological *embedding* of G

Input: a graph $G = (V, E)$

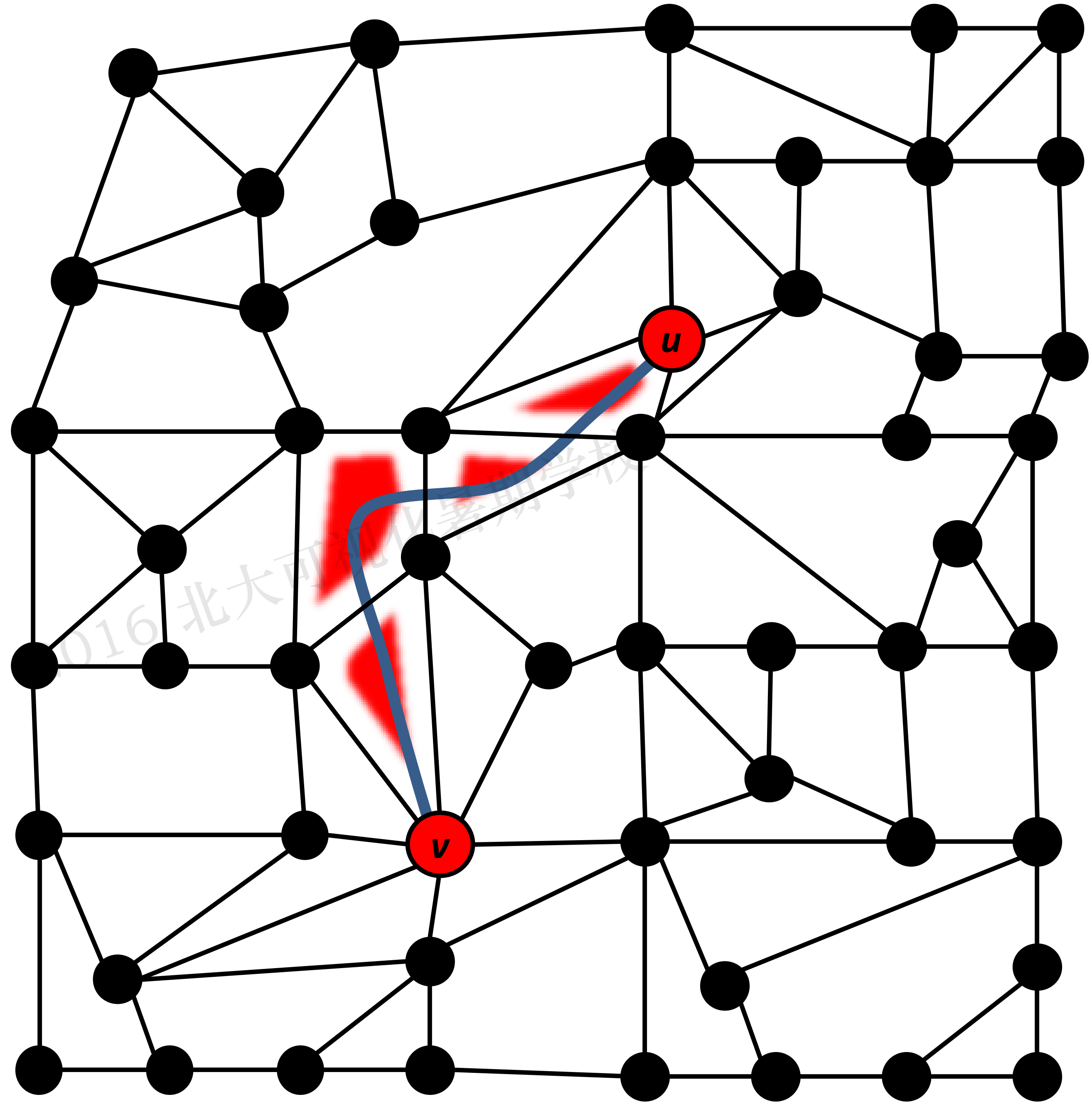
a) Compute a planar subgraph $G' = (V, E')$, where E' is a subset of E, such that $|E'|$ is as large as possible.
b) Compute a planar embedding G'' of G'.
c) Insert the edges of $E - E'$ into G'', creating as few crossings as possible, to create an embedding G''' of G.

Output: an embedding of G with few crossings

Use planar shortest path in the dual

To insert an edge $(u,v) \in E - E'$ into G'':

- Construct the dual graph of G''
- Let f_u be the set of faces containing u
- Let f_v be the set of faces containing v
- Route (u,v) via a shortest path from f_u to f_v.



Topology-shape-metrics approach:

Input: a graph G

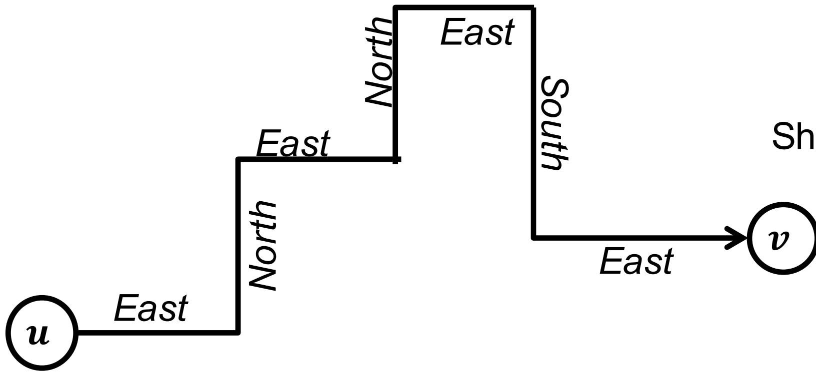
Algorithm:

1. *Topology*: Compute a good topological *embedding* of G
2. *Shape*: Compute a good orthogonal *shape* for this topological embedding
3. *Metrics*: Compute a good orthogonal grid drawing of G

Output: an *orthogonal grid* drawing of G

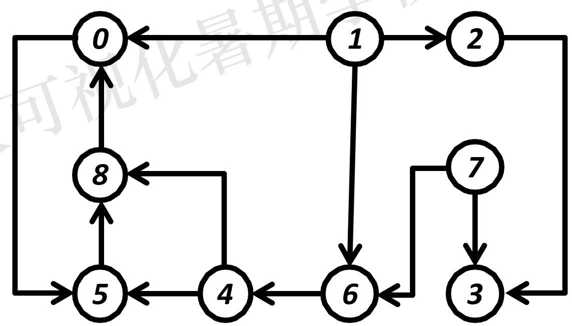
Aim: give shape with a small number of edge bends.

The *shape of a directed orthogonal edge* is the sequence of North/South/East/West turns.



Shape of (u, v): $ENENESE$

The *shape of an orthogonal drawing* consists of the shape of each edge (after directing edges arbitrarily)



0→5	WSE
1→0	W
1→2	E
1→6	S
2→3	ESW
4→5	W
4→8	NW
5→8	N
6→4	W
7→3	S
7→6	WSW
8→0	N

2. *Shape*:

- Compute a good orthogonal shape for the topological embedding output from the topology step.
- We want a small number of bends

Minimum Bends Problem

Input: An embedding $\boldsymbol{G}$

Output: A shape for $\boldsymbol{G}$ with a minimum number of bends.

Surprising result

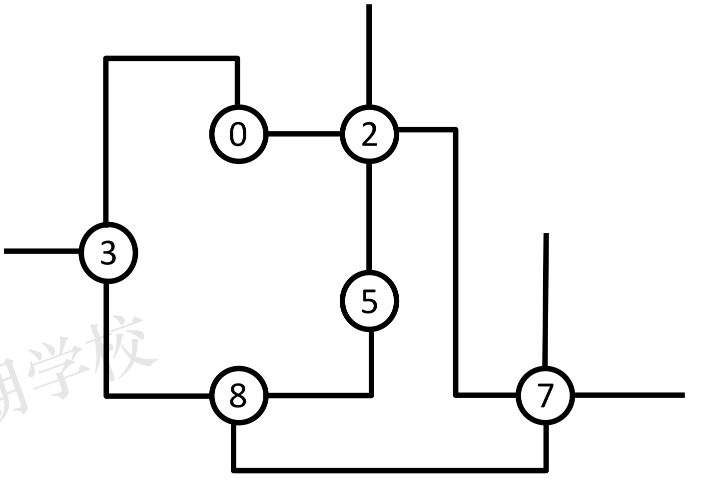
Theorem (Tamassia, ~1987)

The Minimum Bends Problem can be solved in polynomial time.

$O(n^{1.75} \log n)$

Tamassia's algorithm to give a shape with a minimum total number of bends

- Note that $\frac{\pi}{2}$ angles in a drawing satisfy some linear constraints
  - The sum of angles around a face is $\mathbf{2(a+b-4)\frac{\pi}{2}}$, where $\boldsymbol{a}$ is the number of vertices and $\boldsymbol{b}$ is the number of bends in the face.
  - The sum of angles around a vertex is $\mathbf{4\frac{\pi}{2}}$.
  - … plus other constraints from theorems on planar graphs
  - …
  - …

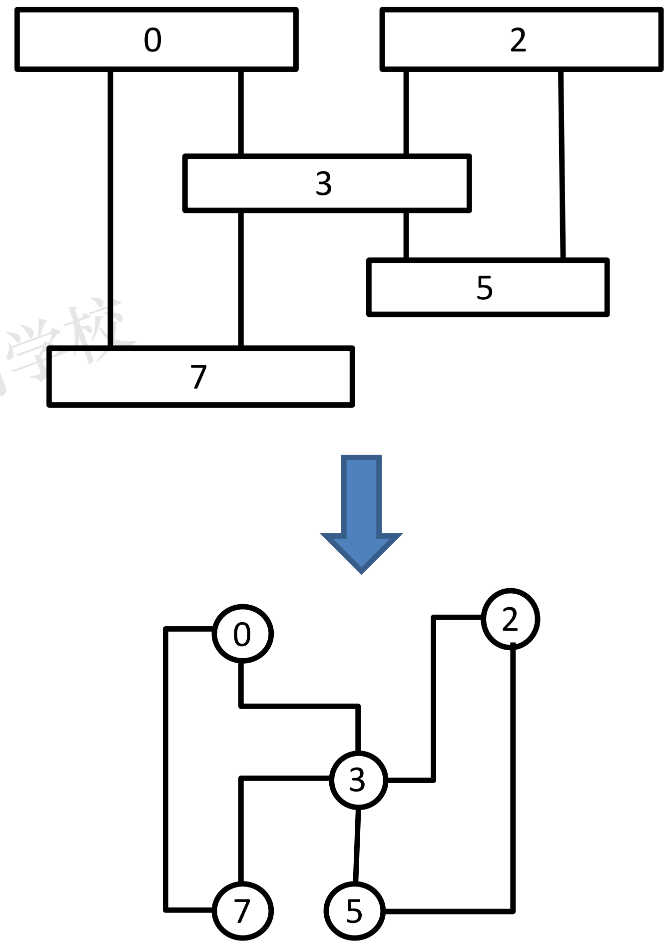


- These constraints can be modelled as a minimum flow problem
- We can solve the minimum flow problem in polynomial time.

Visibility Algorithm

Alternative method to give a shape with a small number of bends

1. Create a *visibility representation* of the input embedding
2. Adjust the visibility representation to give an orthogonal shape



- Runs in linear time
- Relatively elegant
- Does *not* give a minimum total number of bends
- But guarantees that the number of bends on an edge is at most 4.

Visibility algorithm

Input: 2-connected topological embedding $G = (V, E)$

Output: Visibility drawing of G

1. Construct an st-numbering y of G, and direct G according to y.
2. Construct the directed dual D, and topologically sort the nodes of D, to give an x coordinate $x(f)$ for each face f of G.
3. For each edge $e = (u, v) \in E$:

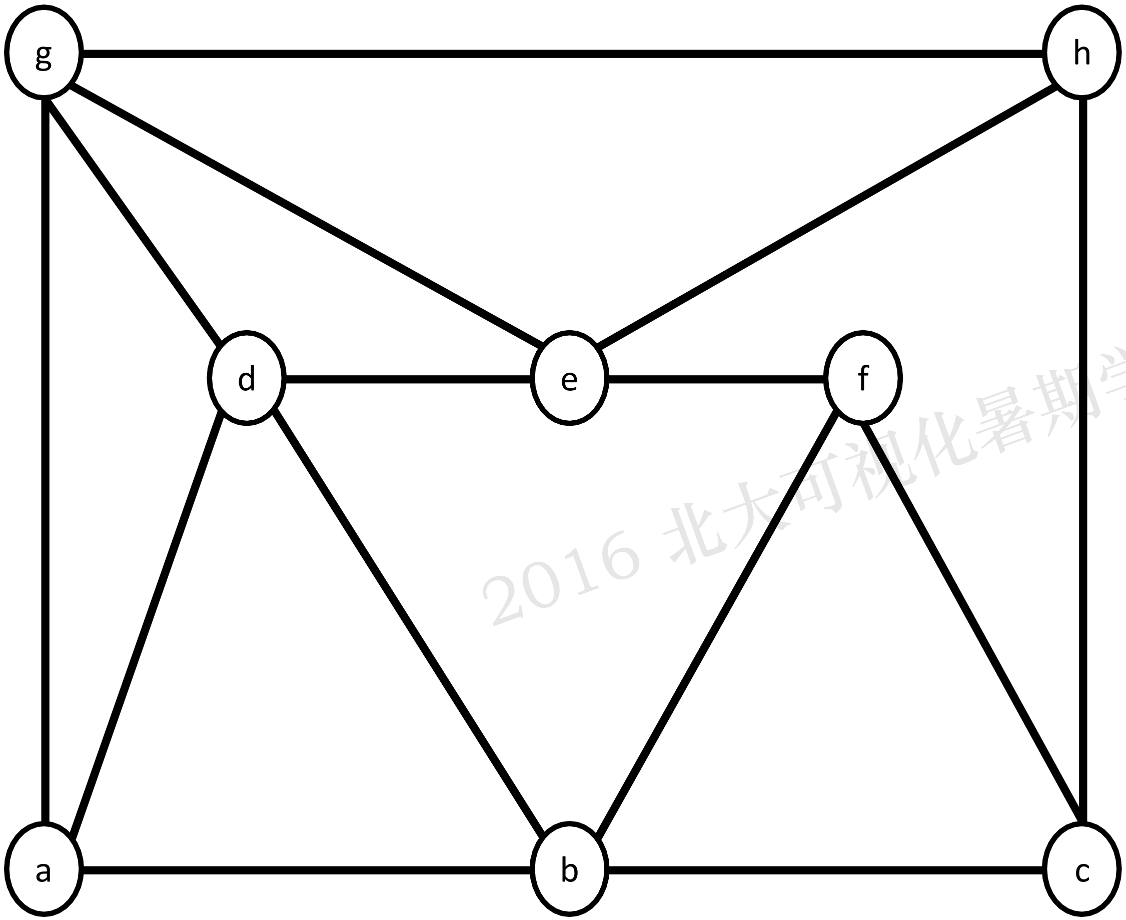
   Let f_e be the face to the left of e

   Draw e as a vertical line segment from $(x(f_e), y(u))$ to $(x(f_e), y(v))$.
4. For each vertex u in V:

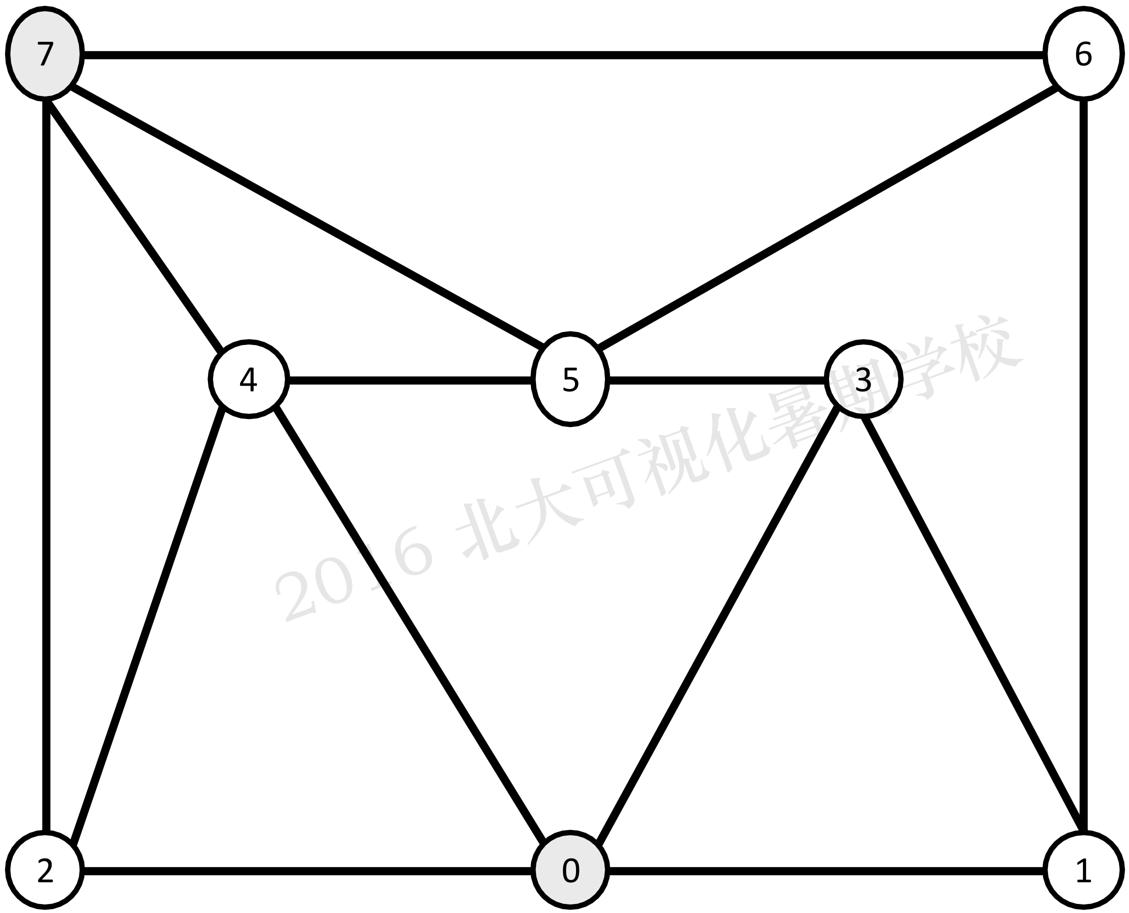
   Let $x_{min}(u) = \min_e(x(f_e))$ over all edges e incident to u

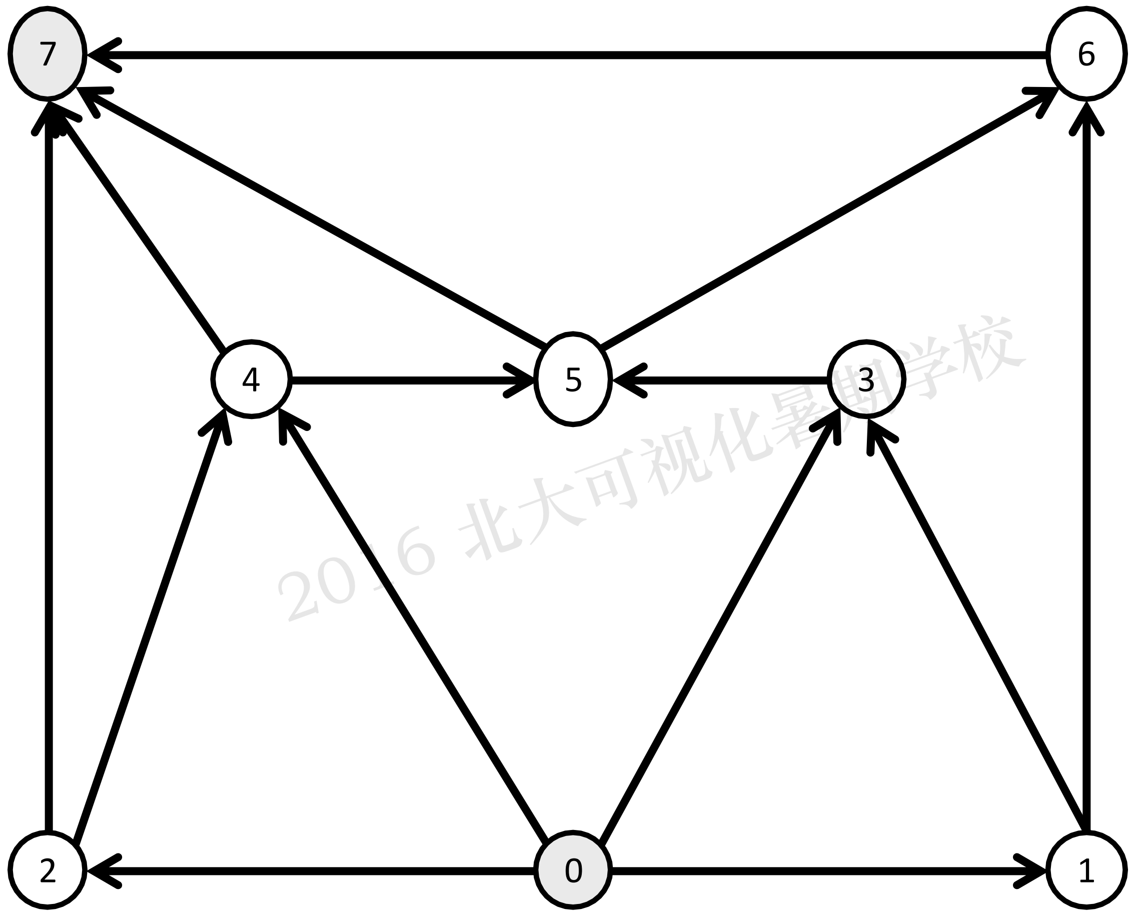
   Let $x_{max}(u) = \max_e(x(f_e))$ over all edges e incident to u

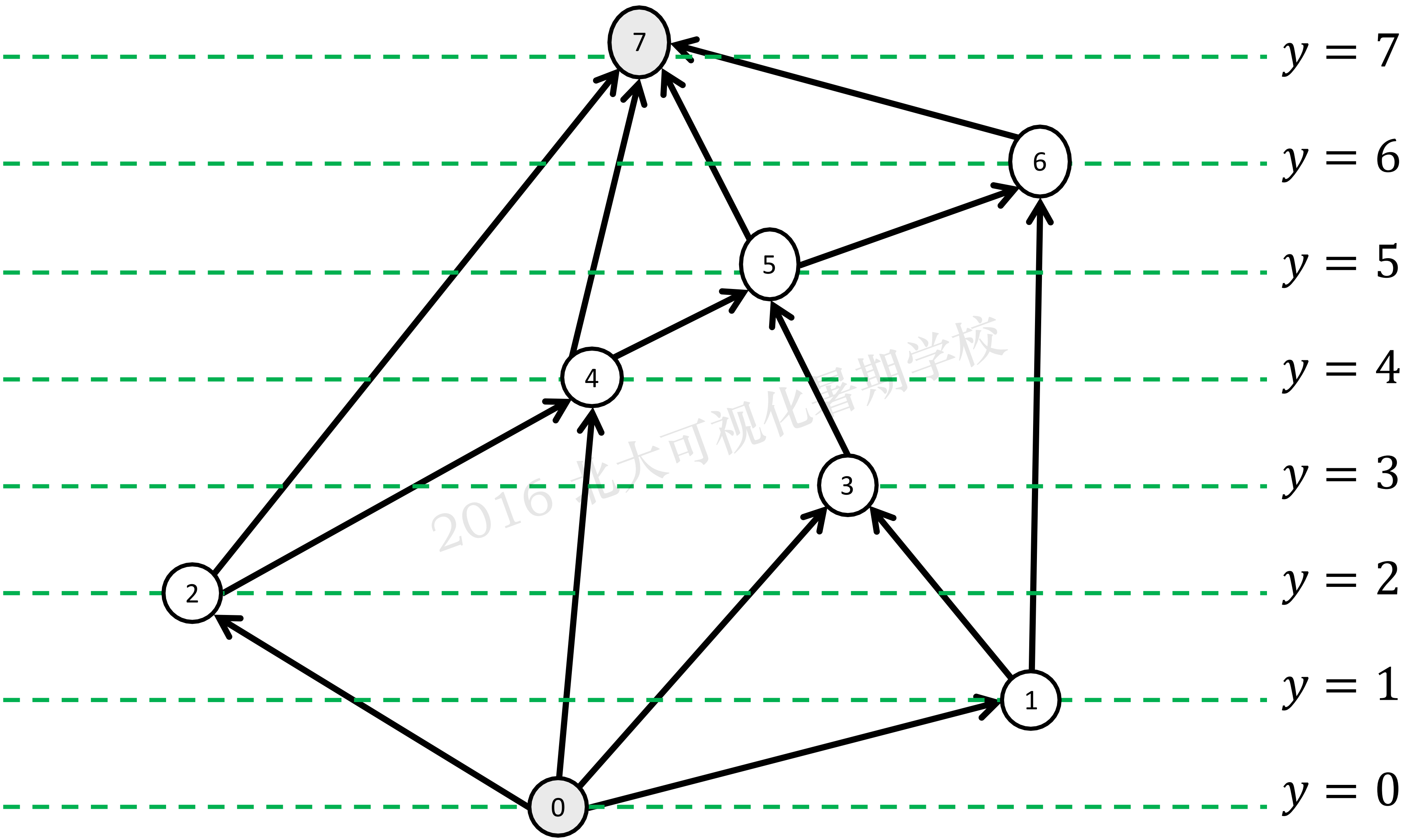
   Draw u as a horizontal line segment from $(x_{min}(u), y(u))$ to $(x_{max}(u), y(u))$.
5. Convert the visibility drawing to an orthogonal drawing.



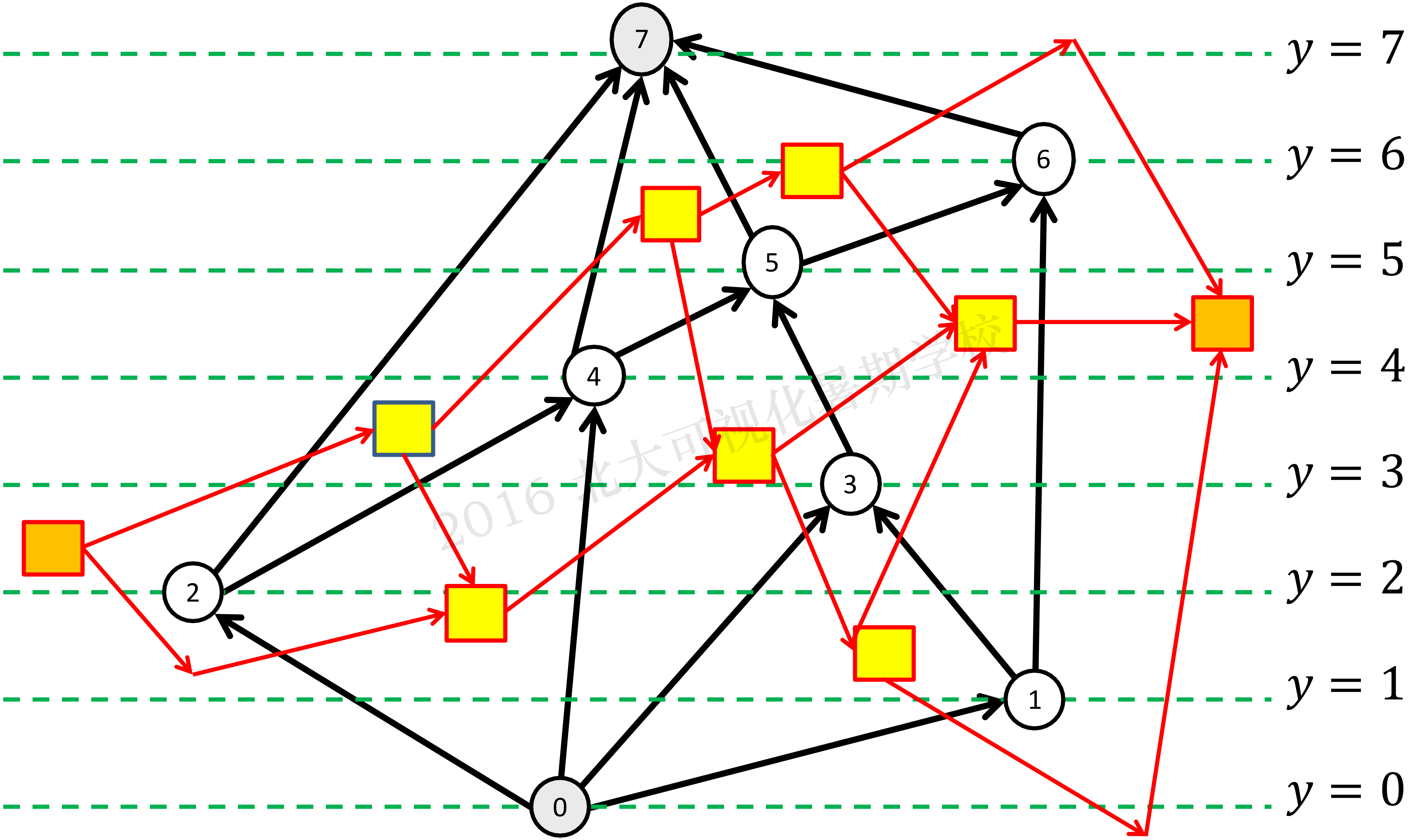
a	b,g,d
b	a,d,f,c
c	b,f,h
d	a,g,e,b
e	d,g,h,f
f	c,b,e
g	a,h,e,d
h	c,e,g


7
6
4
5
3
2
0
1

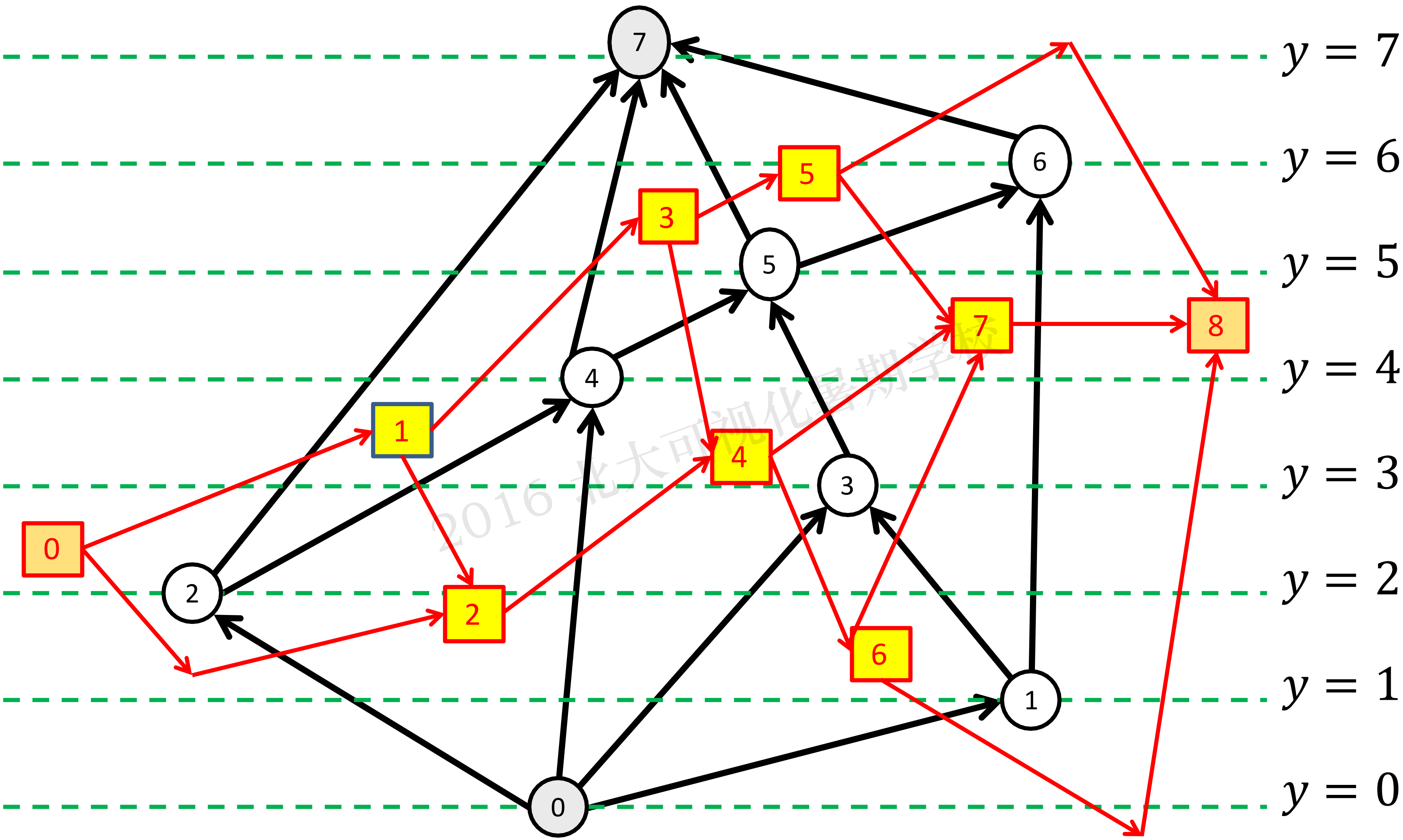

7
6
4
5
3
2
0
1



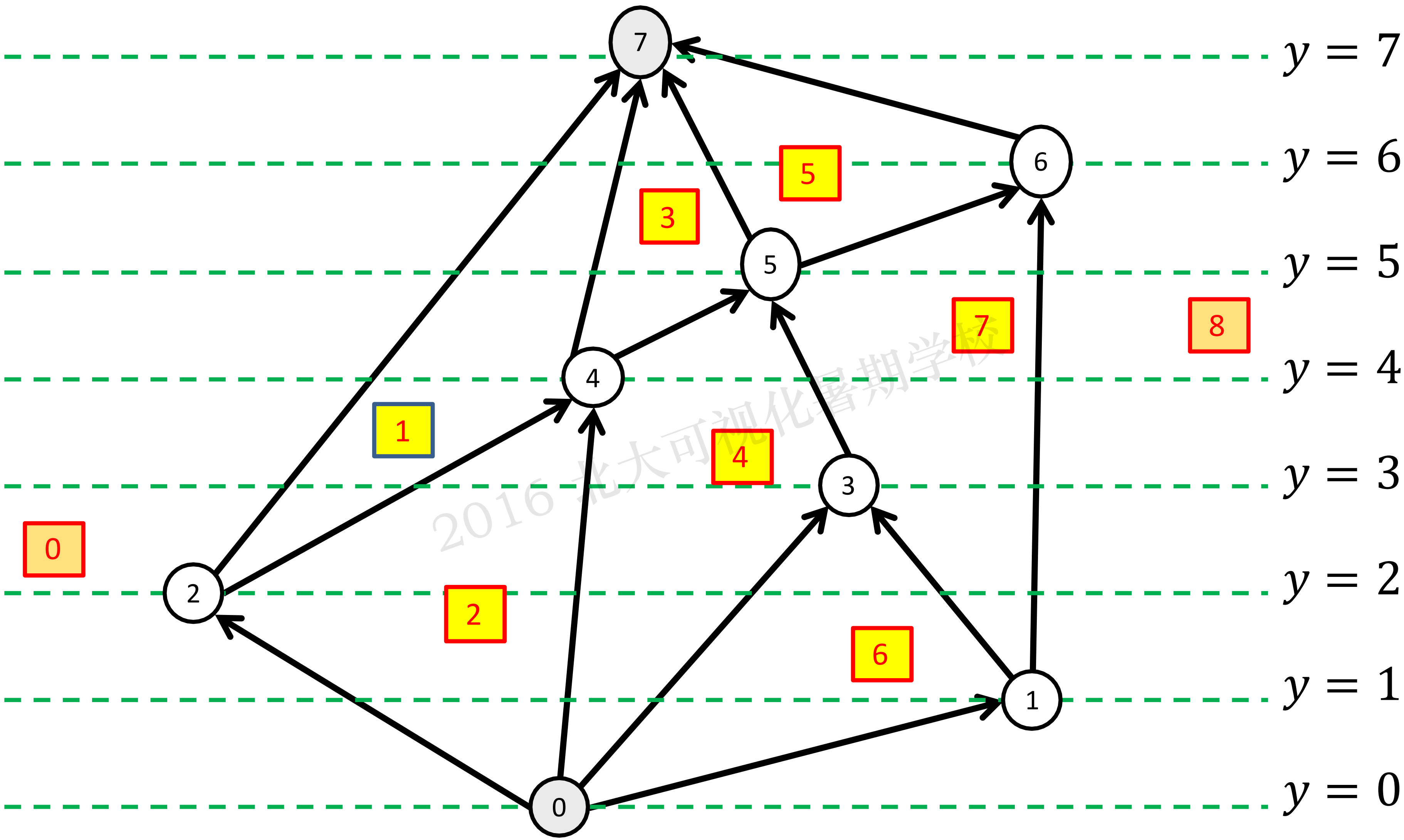
7
6
5
4
3
2
1
0
y = 7
y = 6
y = 5
y = 4
y = 3
y = 2
y = 1
y = 0



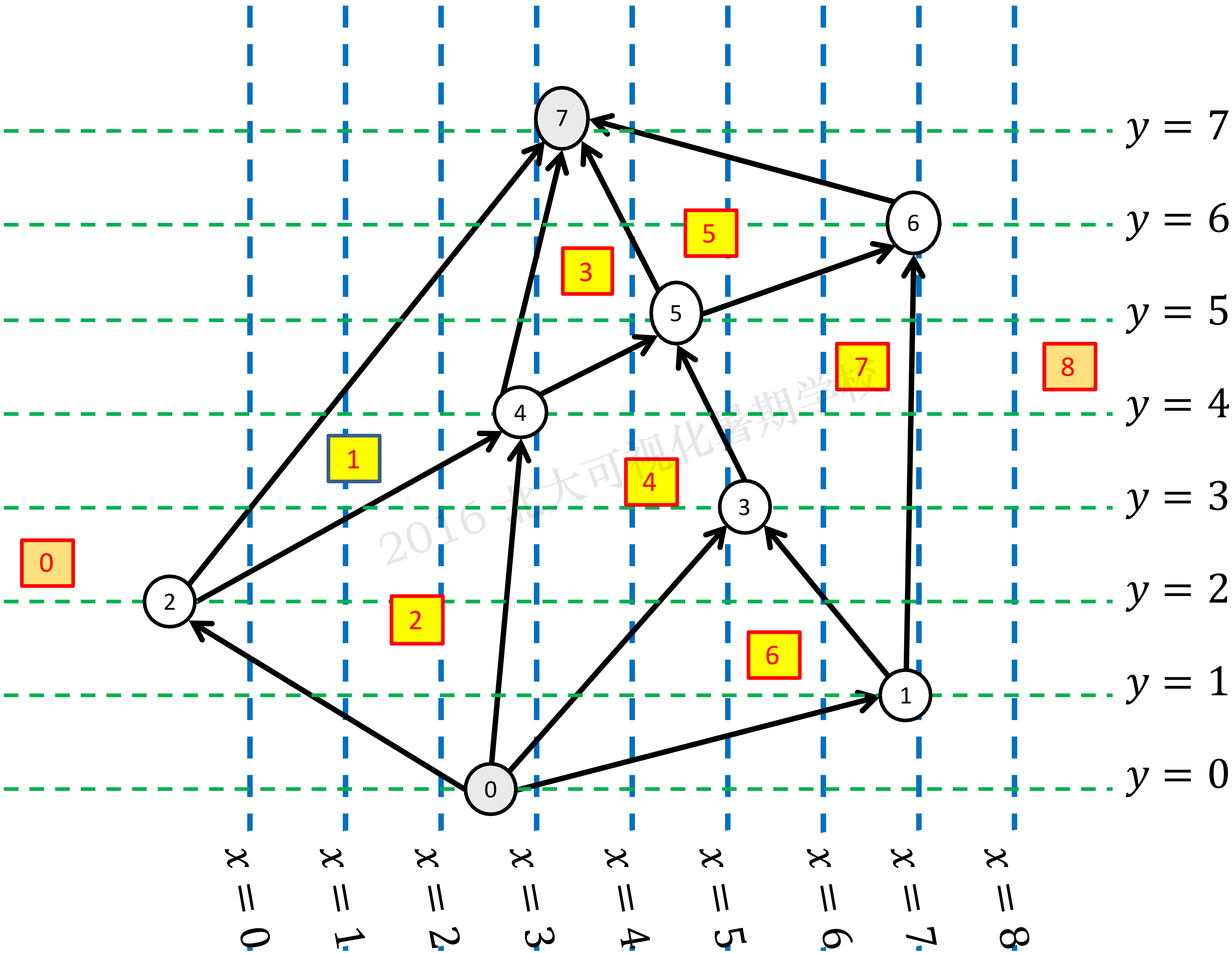
7
6
5
4
3
2
1
0
y = 7
y = 6
y = 5
y = 4
y = 3
y = 2
y = 1
y = 0
2016 北大可视化暑期学校



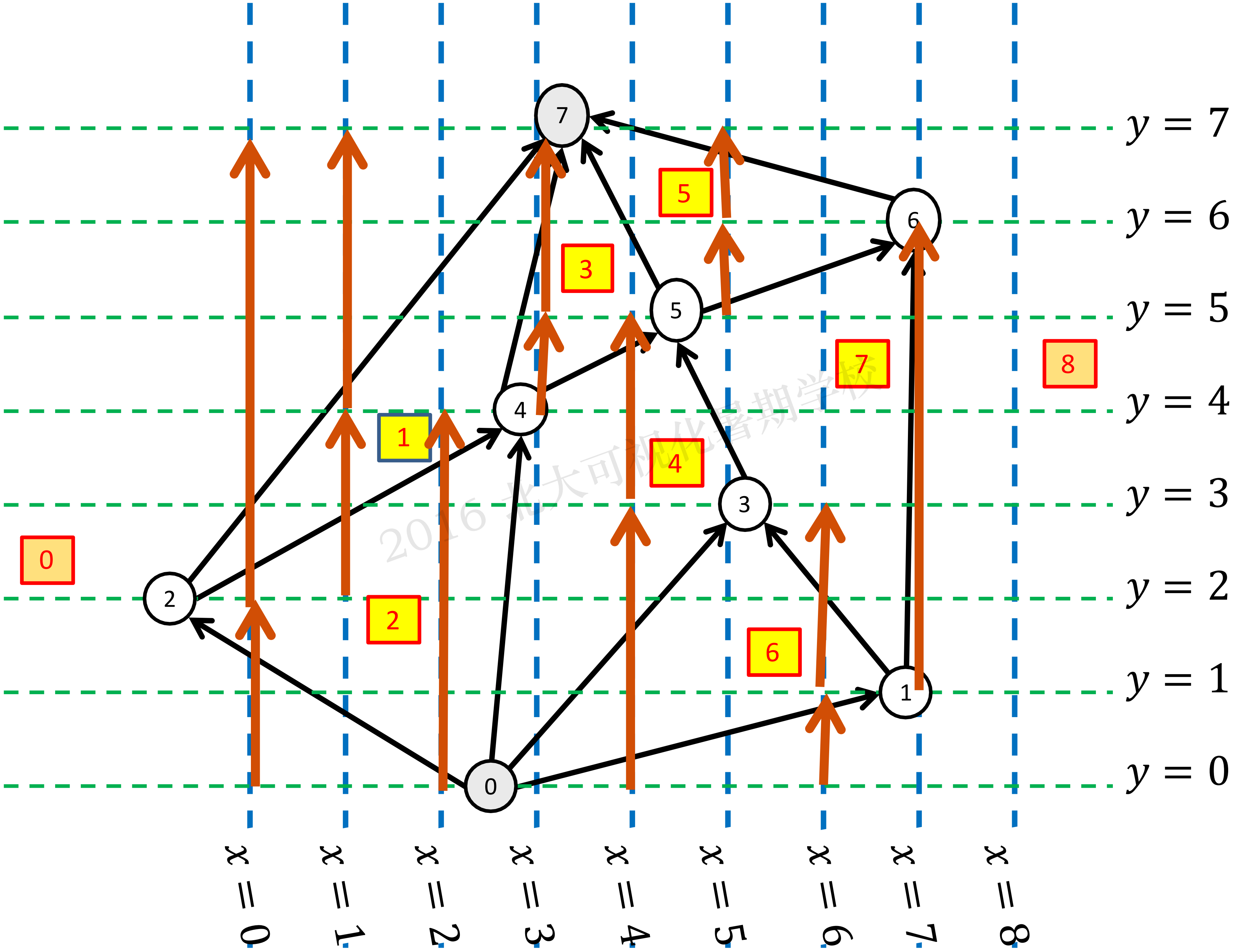
$y = 7$
$y = 6$
$y = 5$
$y = 4$
$y = 3$
$y = 2$
$y = 1$
$y = 0$

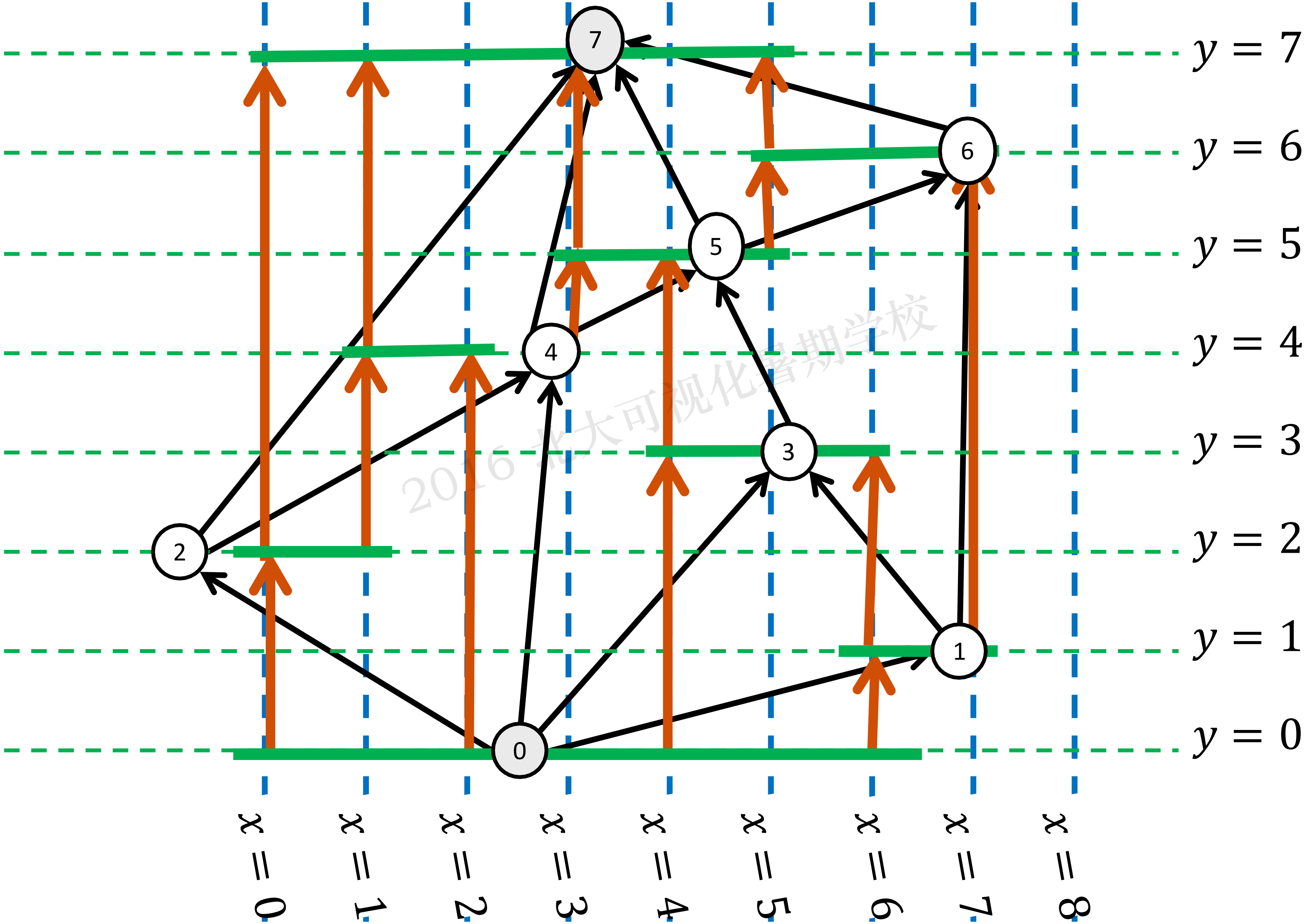


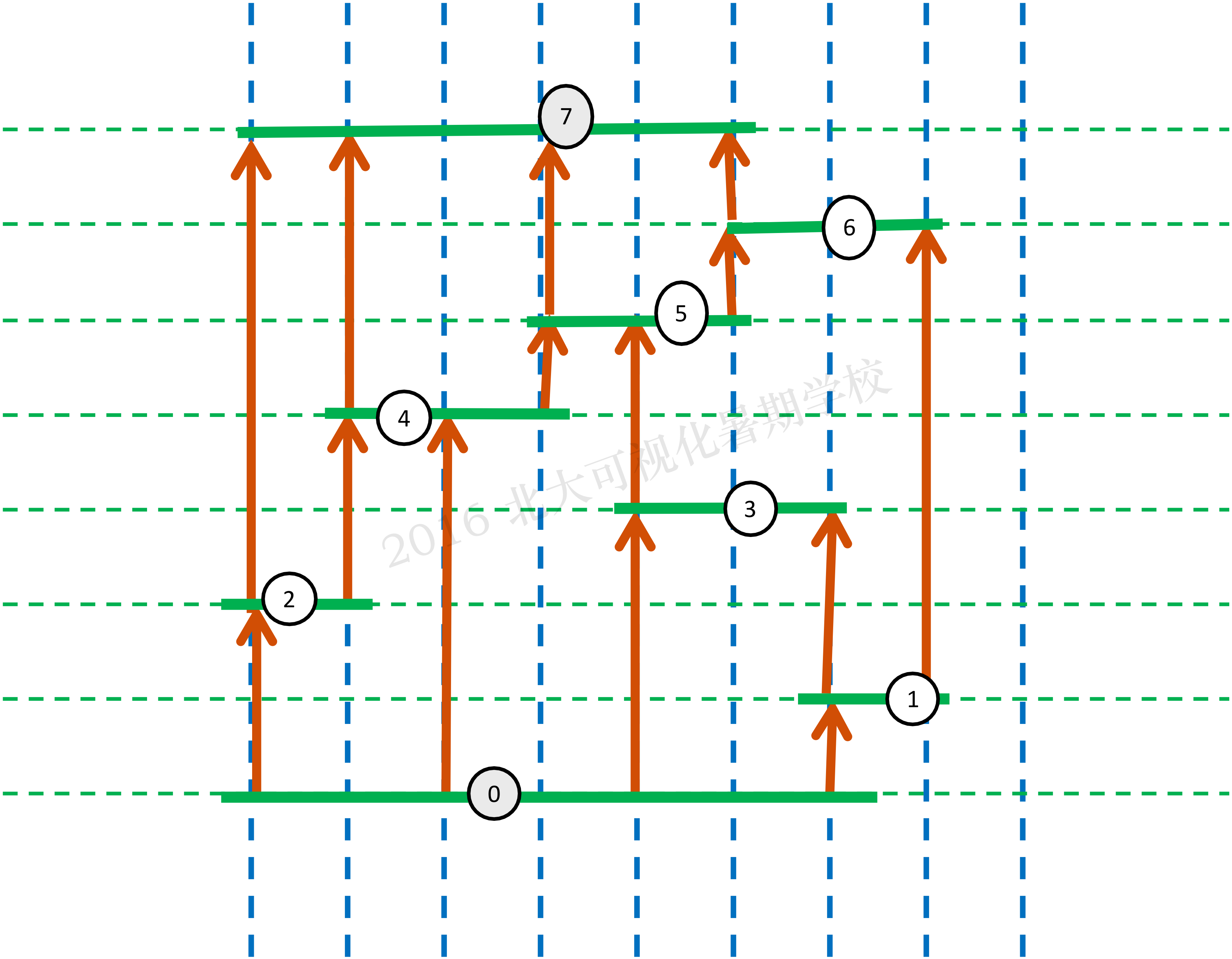
7
6
5
4
3
2
1
0
5
3
7
8
1
4
0
2
6
$y = 7$
$y = 6$
$y = 5$
$y = 4$
$y = 3$
$y = 2$
$y = 1$
$y = 0$
2016 北大可视化暑期学校



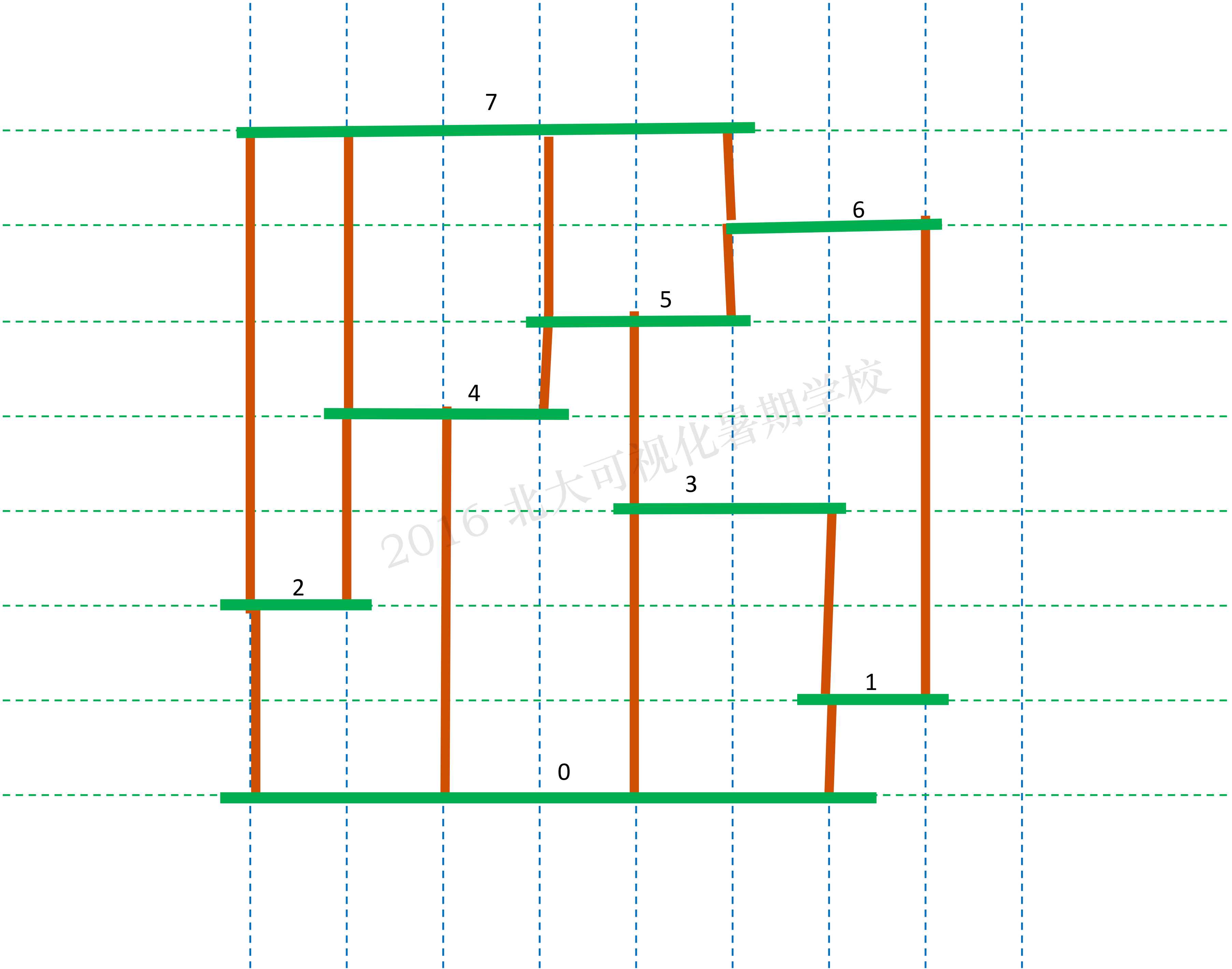
0
1
2
3
4
5
6
7
0
1
2
3
4
5
6
7
8
y = 7
y = 6
y = 5
y = 4
y = 3
y = 2
y = 1
y = 0
x = 0
x = 1
x = 2
x = 3
x = 4
x = 5
x = 6
x = 7
x = 8
2016 北大可视化暑期学校

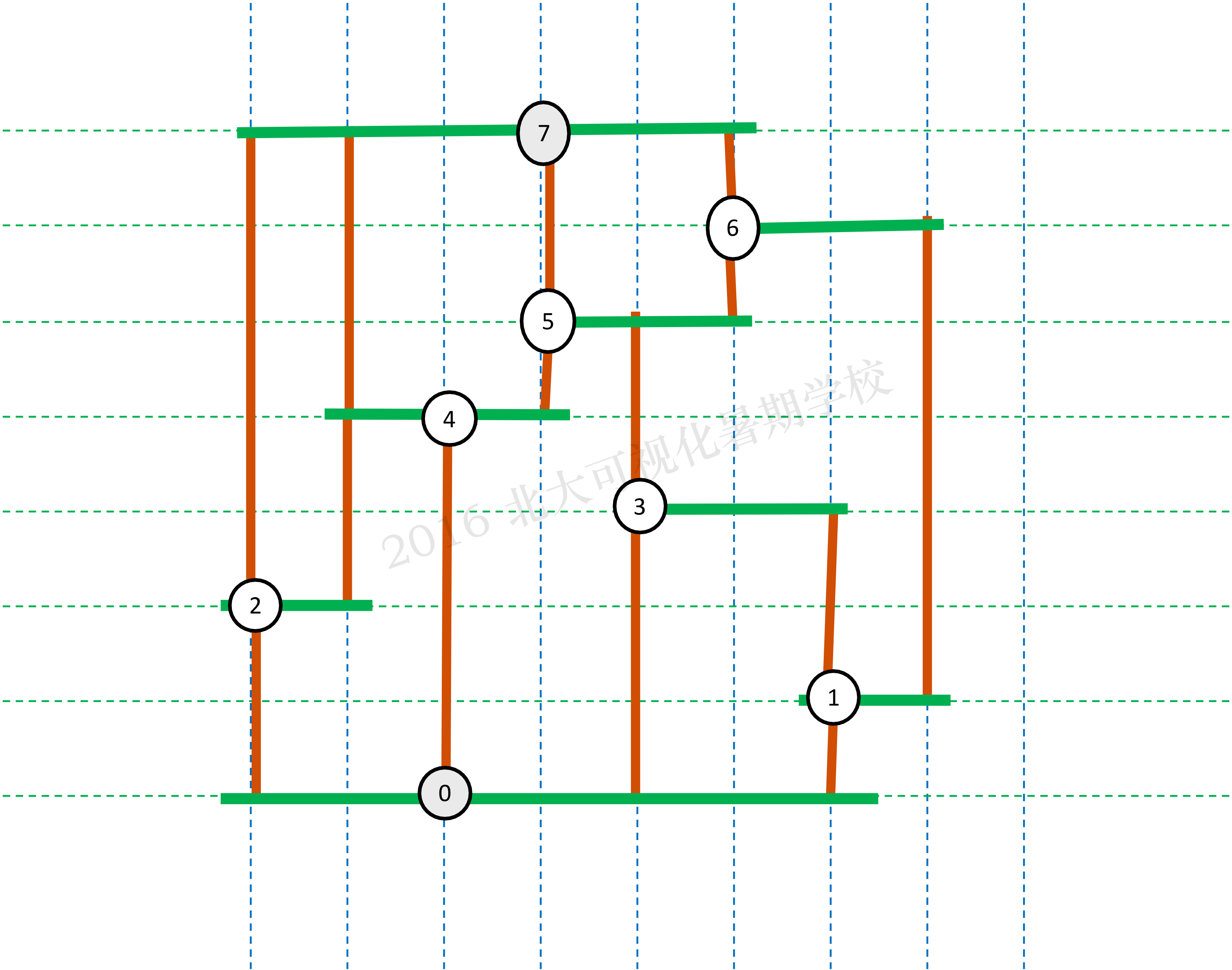

y = 7
y = 6
y = 5
y = 4
y = 3
y = 2
y = 1
y = 0
x = 0
x = 1
x = 2
x = 3
x = 4
x = 5
x = 6
x = 7
x = 8
0
1
2
3
4
5
6
7
8
2016 北大可视化暑期学校


7
6
5
4
3
2
1
0
y = 7
y = 6
y = 5
y = 4
y = 3
y = 2
y = 1
y = 0
x = 0
x = 1
x = 2
x = 3
x = 4
x = 5
x = 6
x = 7
x = 8
2016 北大可视化暑期学校

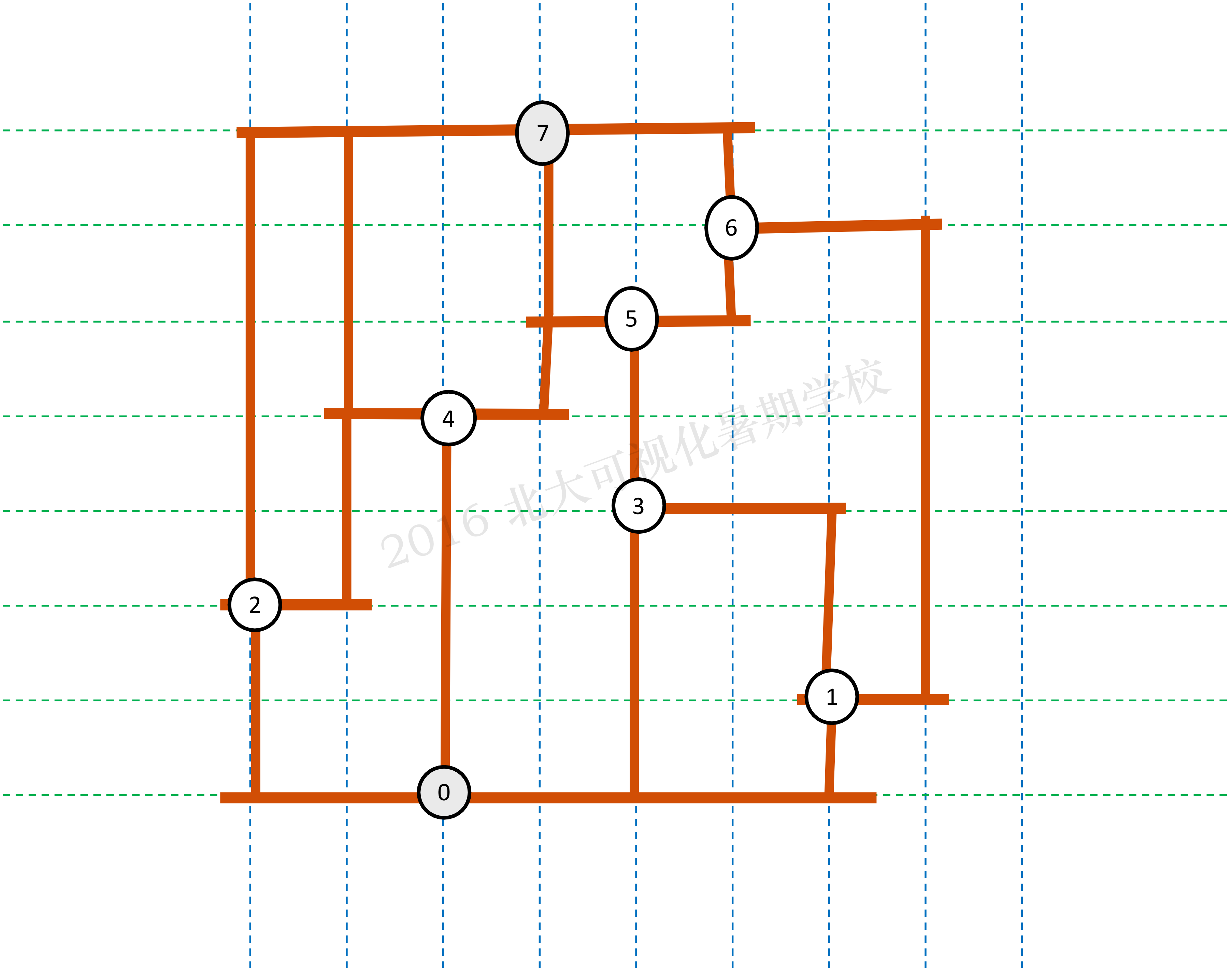


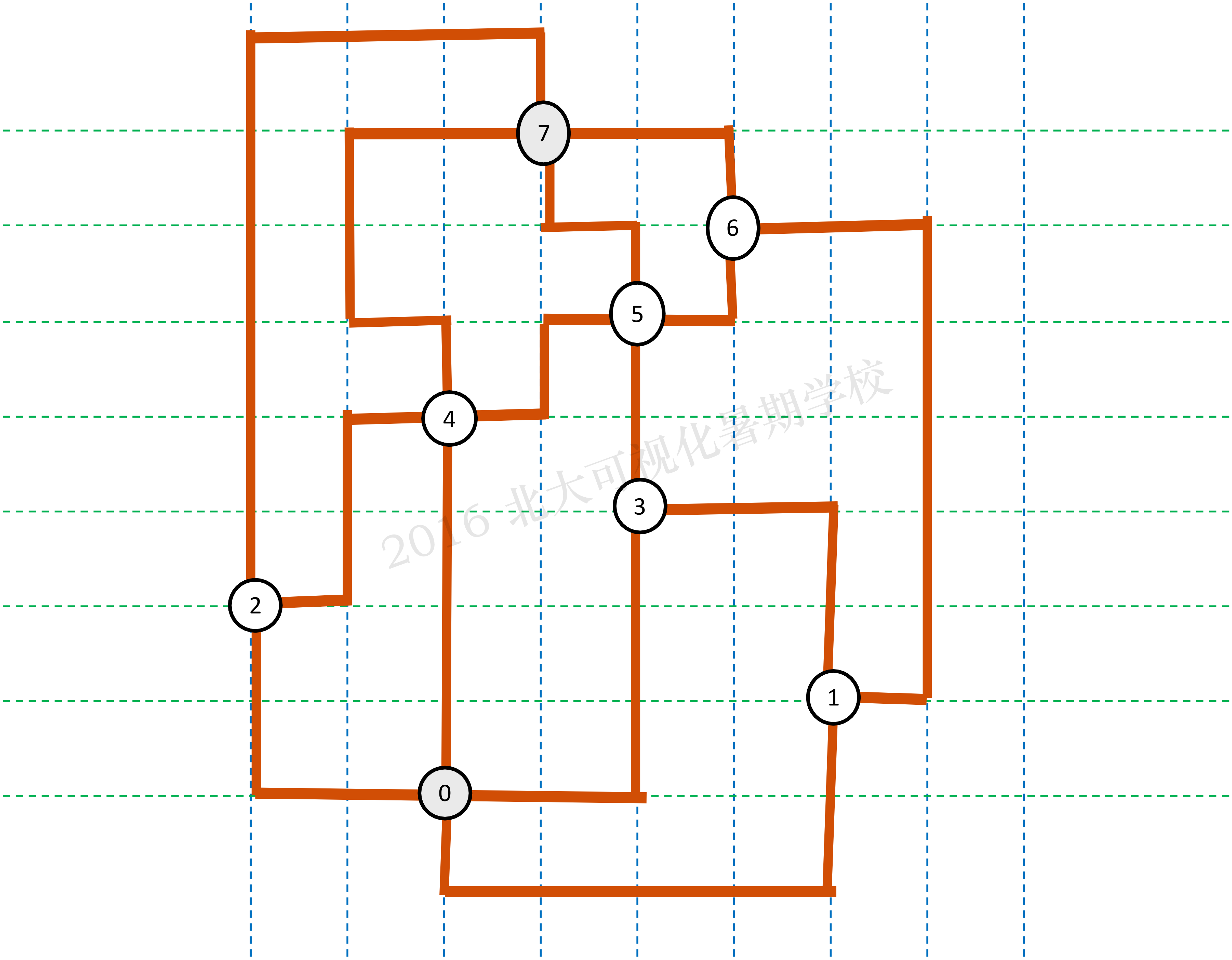
7
6
5
4
3
2
1
0

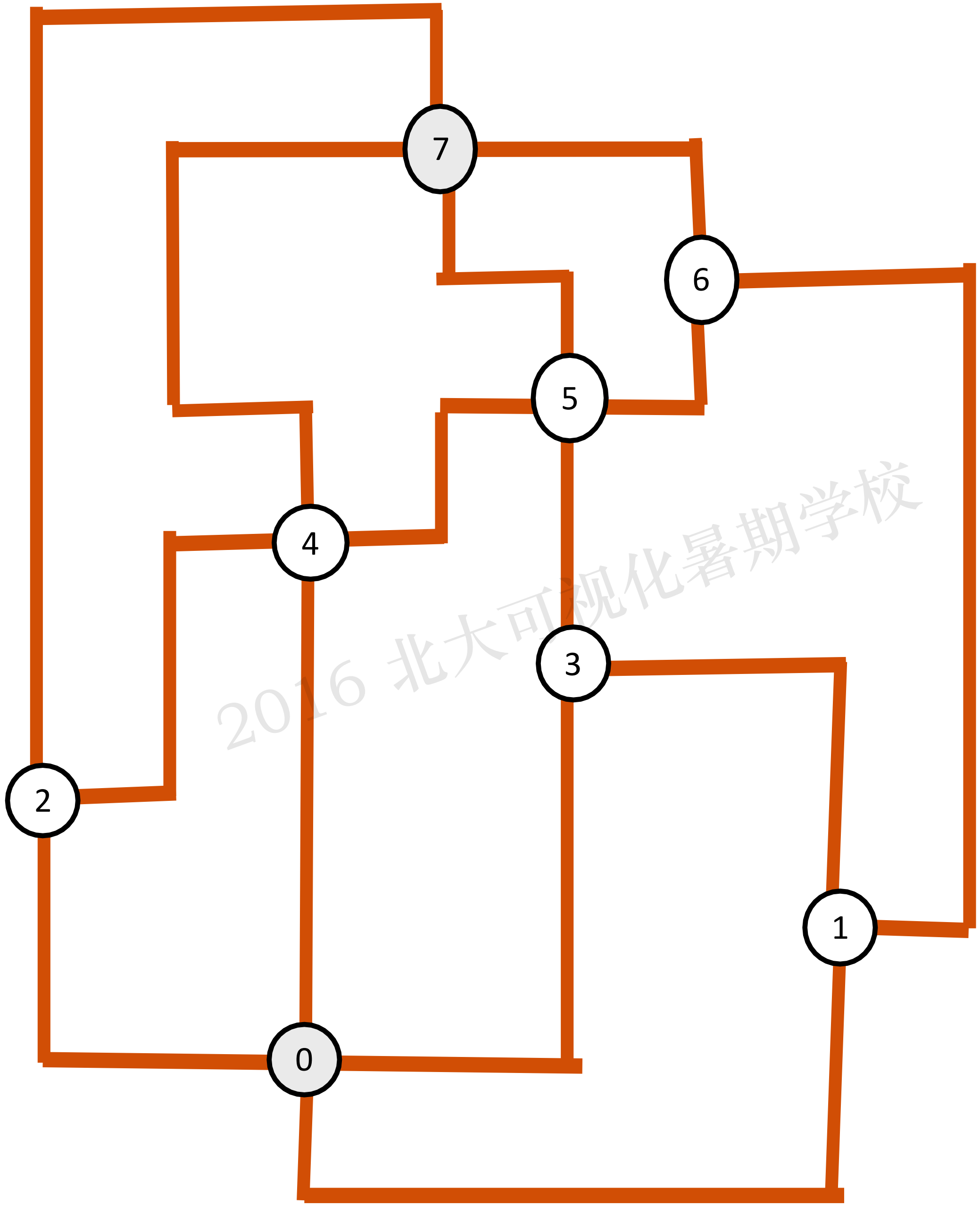

7
6
5
4
3
2
1
0



7
6
5
4
3
2
1
0


7
6
5
4
3
2
1
0


7
6
5
4
3
2
1
0
2016 北大可视化暑期学校


7
6
5
4
3
2
1
0

Topology-shape-metrics approach:

Input: a graph G

Algorithm:

1. *Topology*: Compute a good topological *embedding* of G
2. *Shape*: Compute a good orthogonal *shape* for this topological embedding
3. *Metrics*: Compute a good orthogonal grid drawing of G

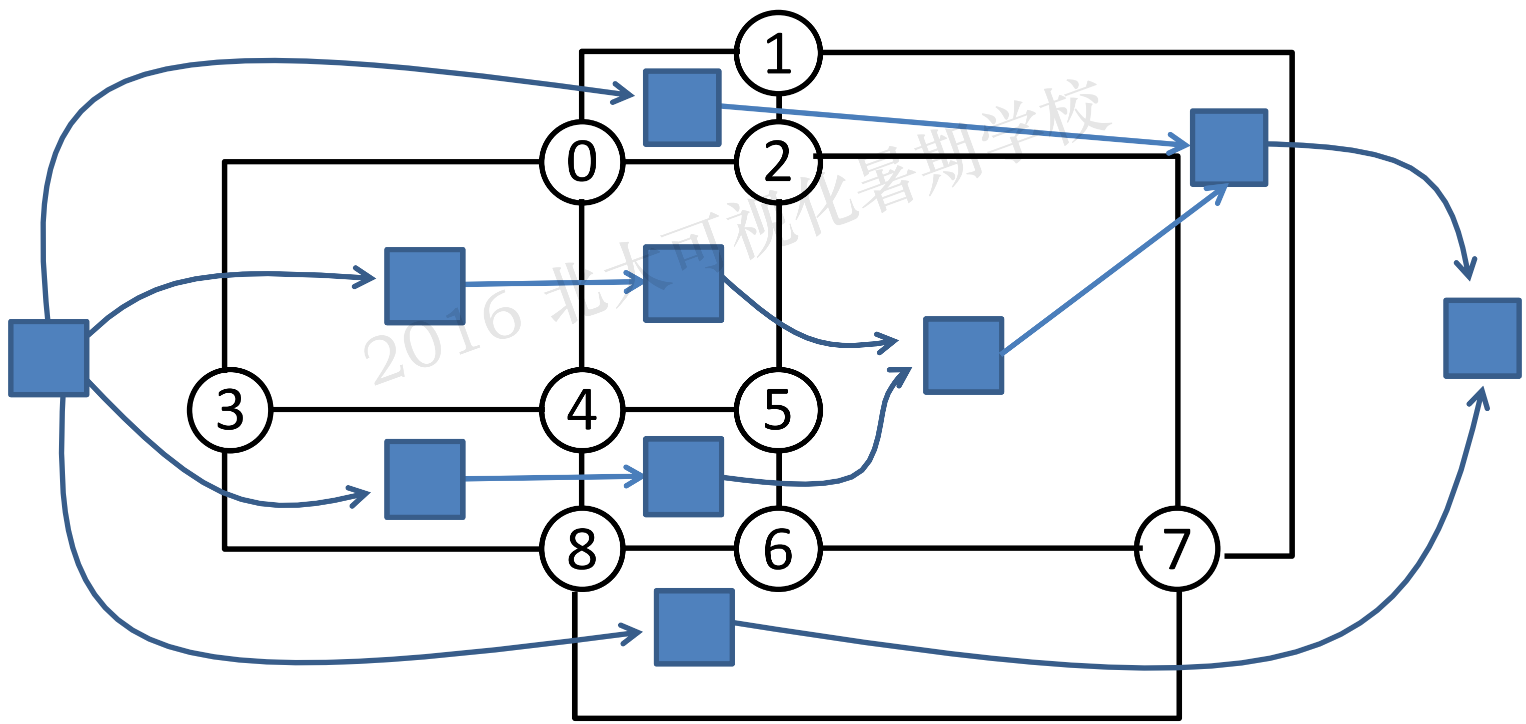
Output: an *orthogonal grid* drawing of G

Aim: give a drawing with good vertex resolution

Metrics step: Use VLSI-inspired compaction methods to get a drawing on a small grid

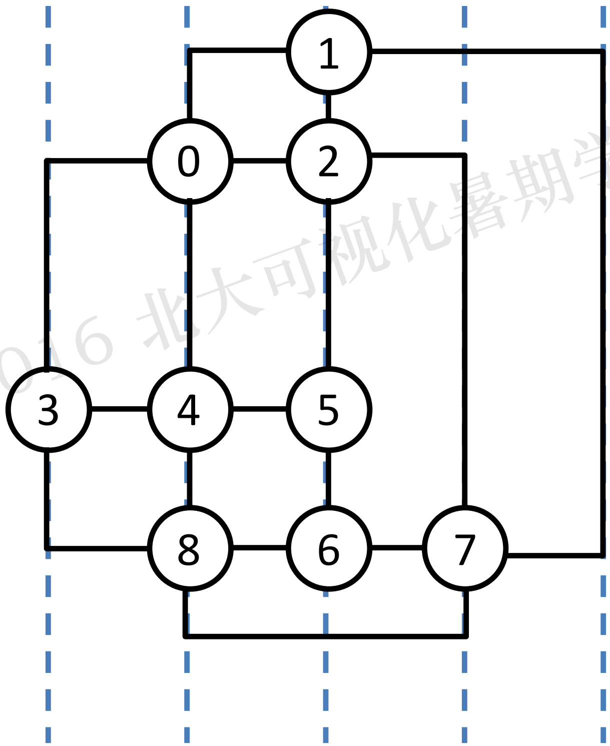
Compaction in the x direction

1. Construct a directed visibility graph H on the dual with source at the left and sink at the right.
2. For each vertex u in H, find a longest path in H from the source to u.
3. Assign y-coordinates using path-length from the source.



Compaction in the x direction

1. Construct a directed visibility graph H on the dual with source at the left and sink at the right.
2. For each vertex u in H, find a longest path in H from the source to u.
3. Assign y-coordinates using path-length from the source.



Similarly compact in the y-direction.

Remarks

Topology-shape-metrics method:

Input: a graph $\boldsymbol{G}$

Algorithm:

1. *Topology*: Compute a good topological *embedding* of $\boldsymbol{G}$
2. *Shape*: Compute a good orthogonal *shape* for this topological embedding
3. *Metrics*: Compute a good orthogonal grid drawing of $\boldsymbol{G}$

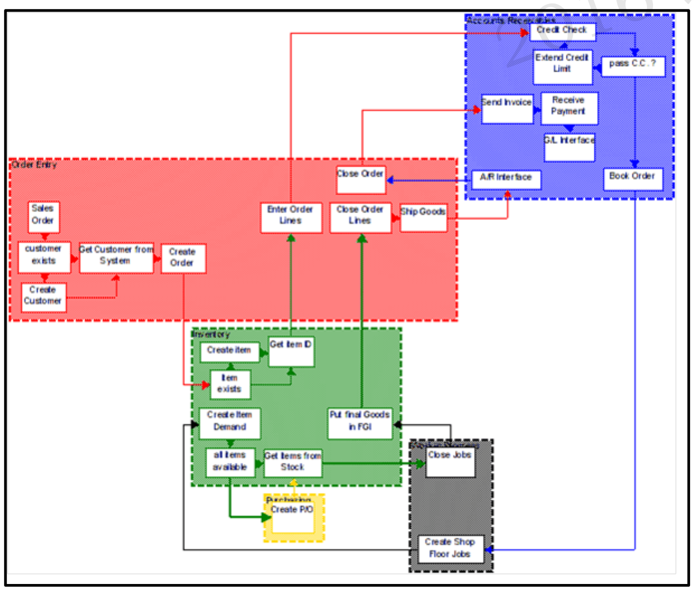
Output: an *orthogonal grid* drawing of $\boldsymbol{G}$

Is this method any good?

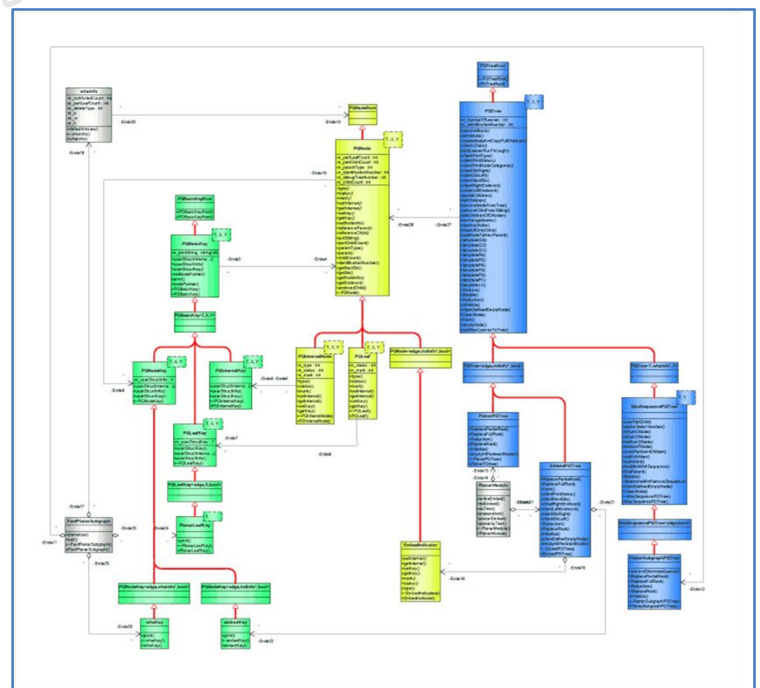
Topology-shape-metrics approach

Good things

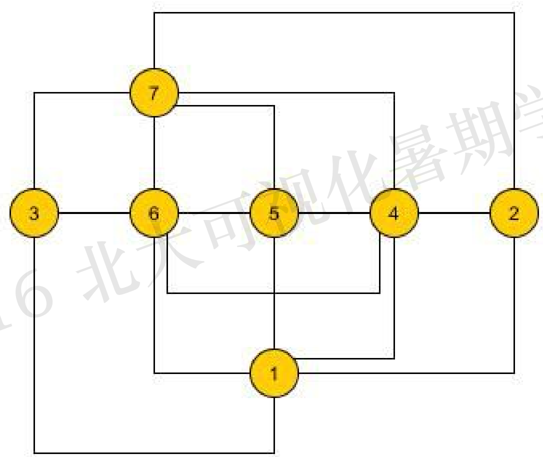
- Works well on small graphs
- Relatively fast (varies from $O(n)$ to $O(n^2 \log n)$)
- Validated readability
- Can be adjusted to handle vertices of large degree and large size
- Can be adjusted for some constraints

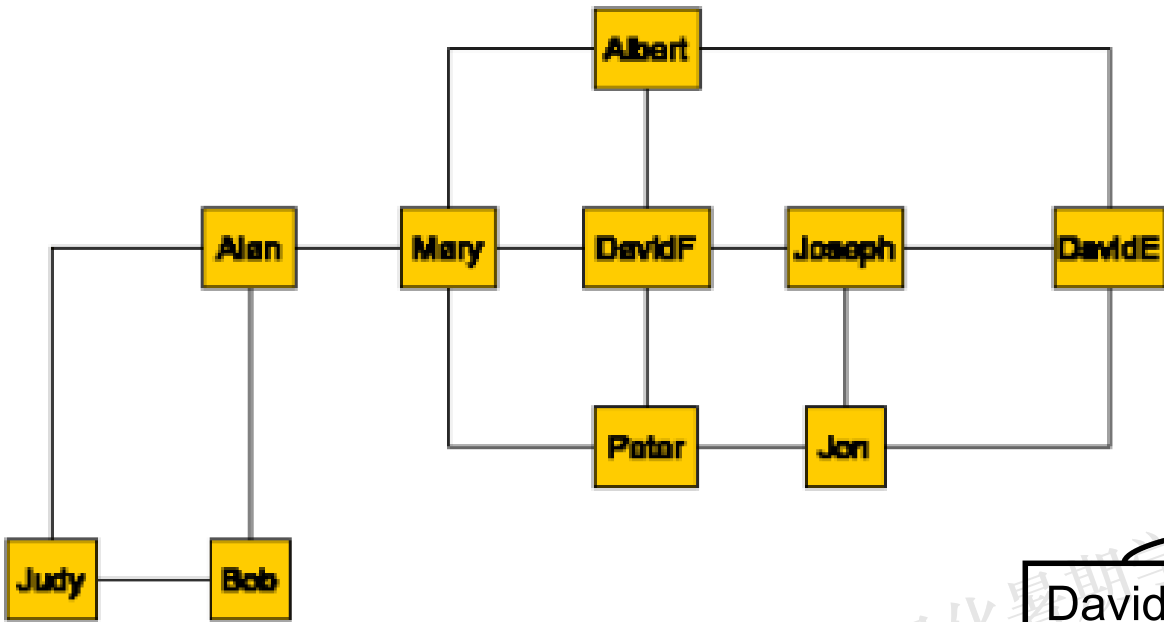


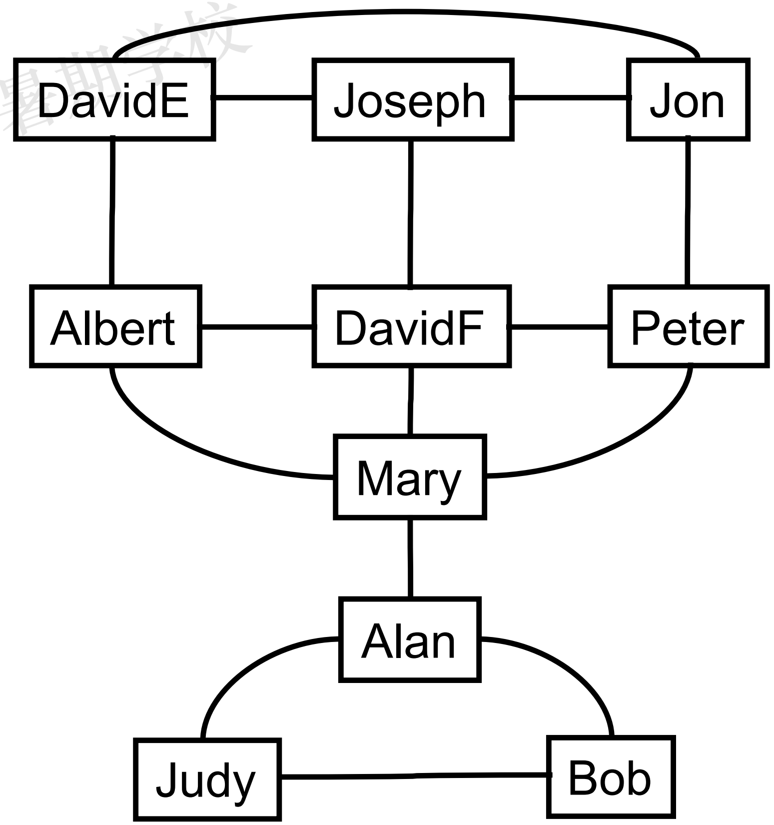
Klein

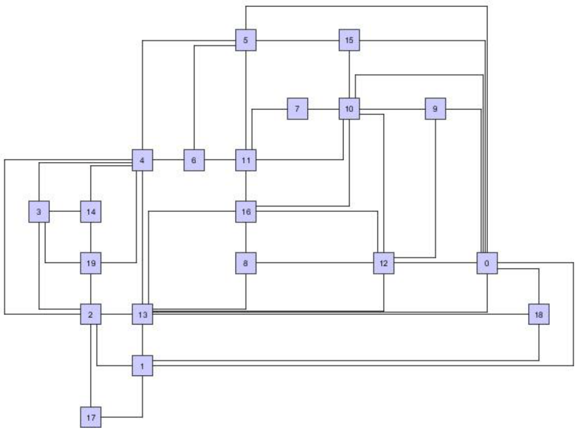


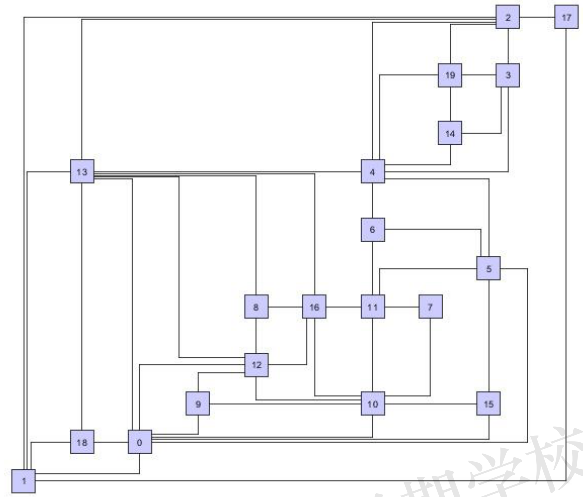
Klein


7
3
6
5
4
2
1

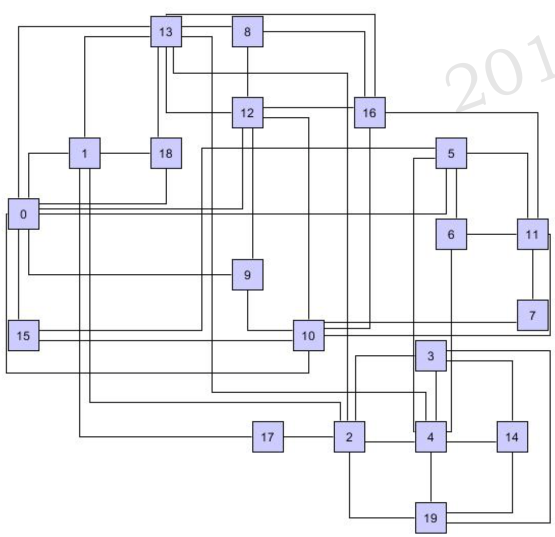

Albert
Alan
Mary
DavidF
Joseph
DavidE
Peter
Jon
Judy
Bob

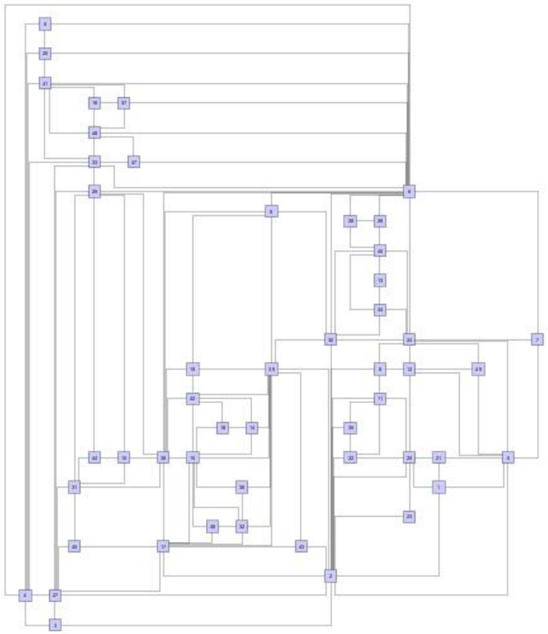

DavidE
Joseph
Jon
Albert
DavidF
Peter
Mary
Alan
Judy
Bob

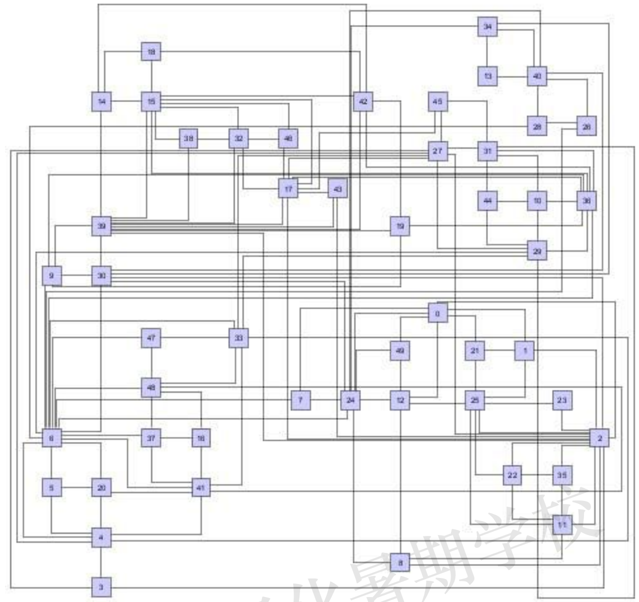

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19

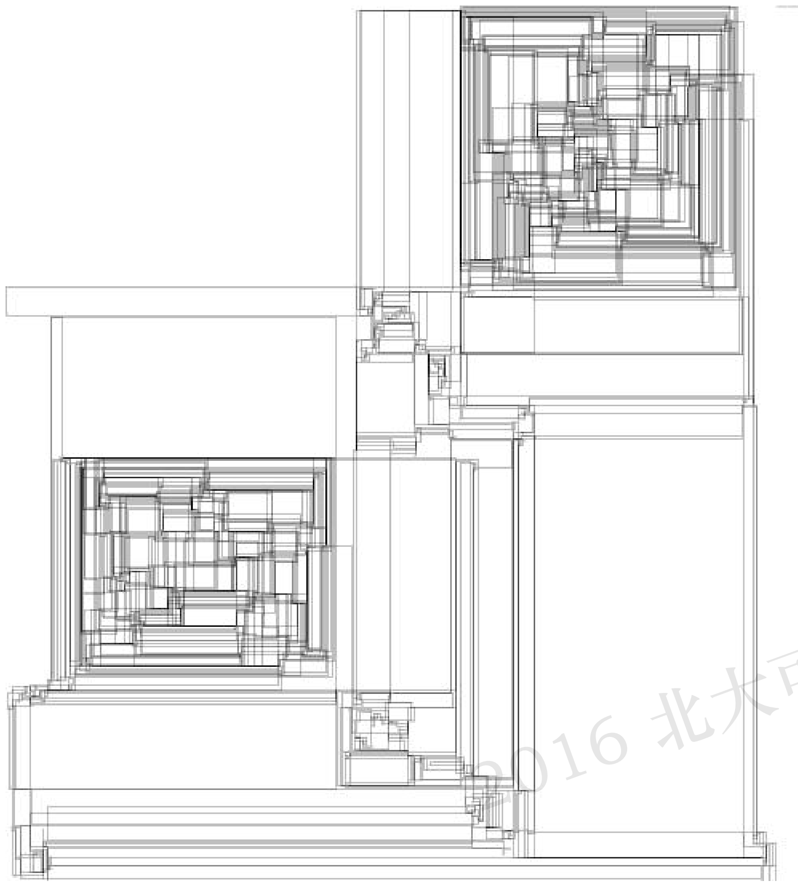
$n = 20$


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19



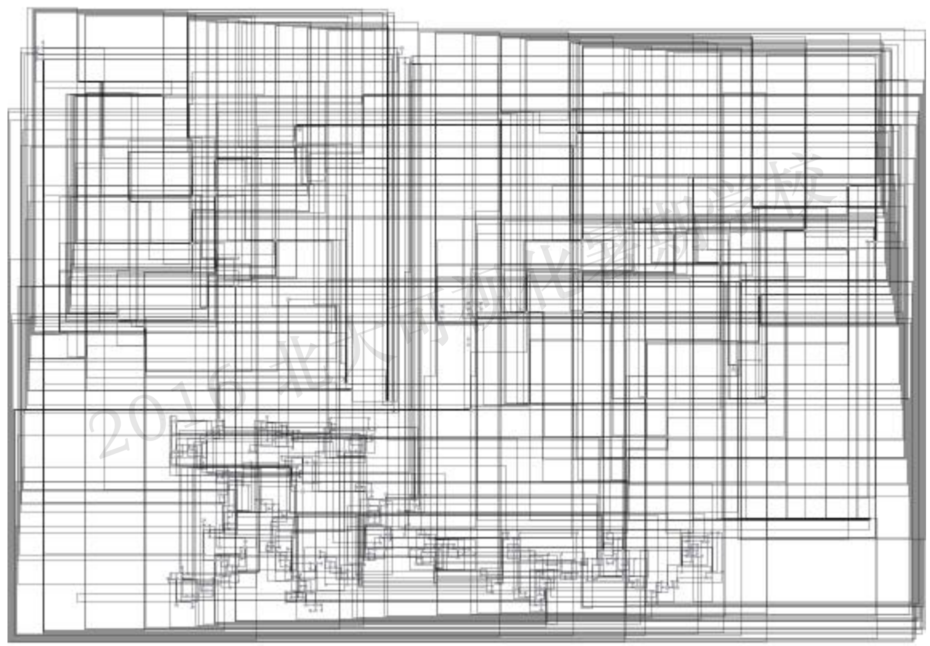


$n = 50$



$n = 500$

$n = 500$



Topology-shape-metrics approach

Bad things

- Large drawings often look bad (poor faithfulness?)
- Very difficult to code