Octagonal Drawings of Plane Graphs with Prescribed Face Areas

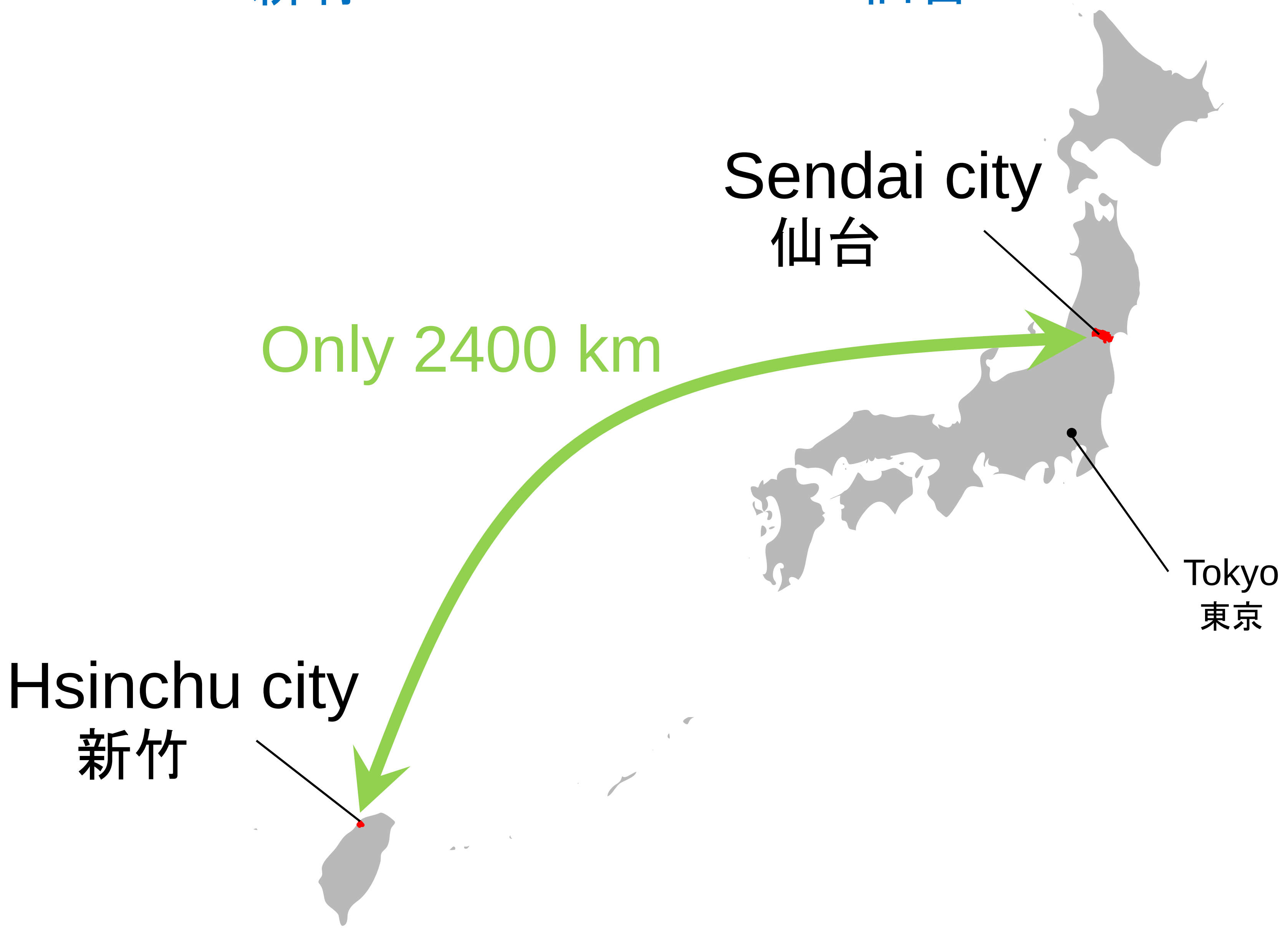
指定面積的平面図的八角形描画

Takao Nishizeki（Tohoku Univ.)

西関　隆夫　（東北大学）

ツディィン　メンジディィ　ピンメントウディ　バージョシン　ミョーワォ

Hsinchu city and Sendai city
新竹　　　　　　　　仙台



Tohoku University(東北大学)

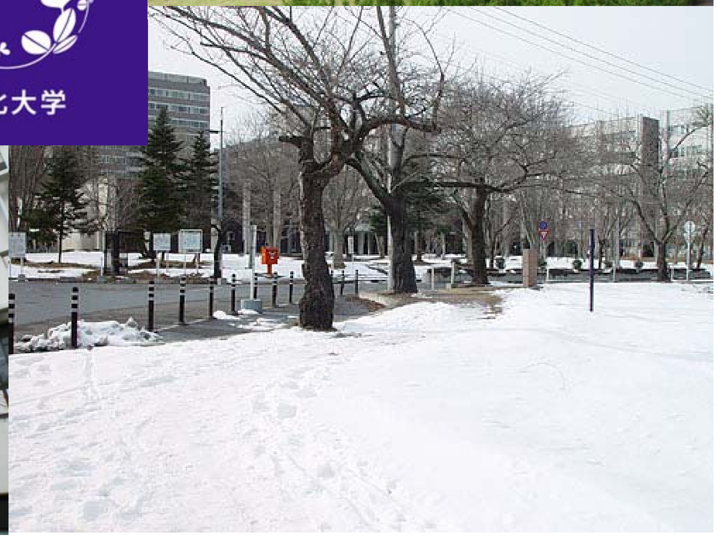
Tohoku University was established in 1907.

春

夏

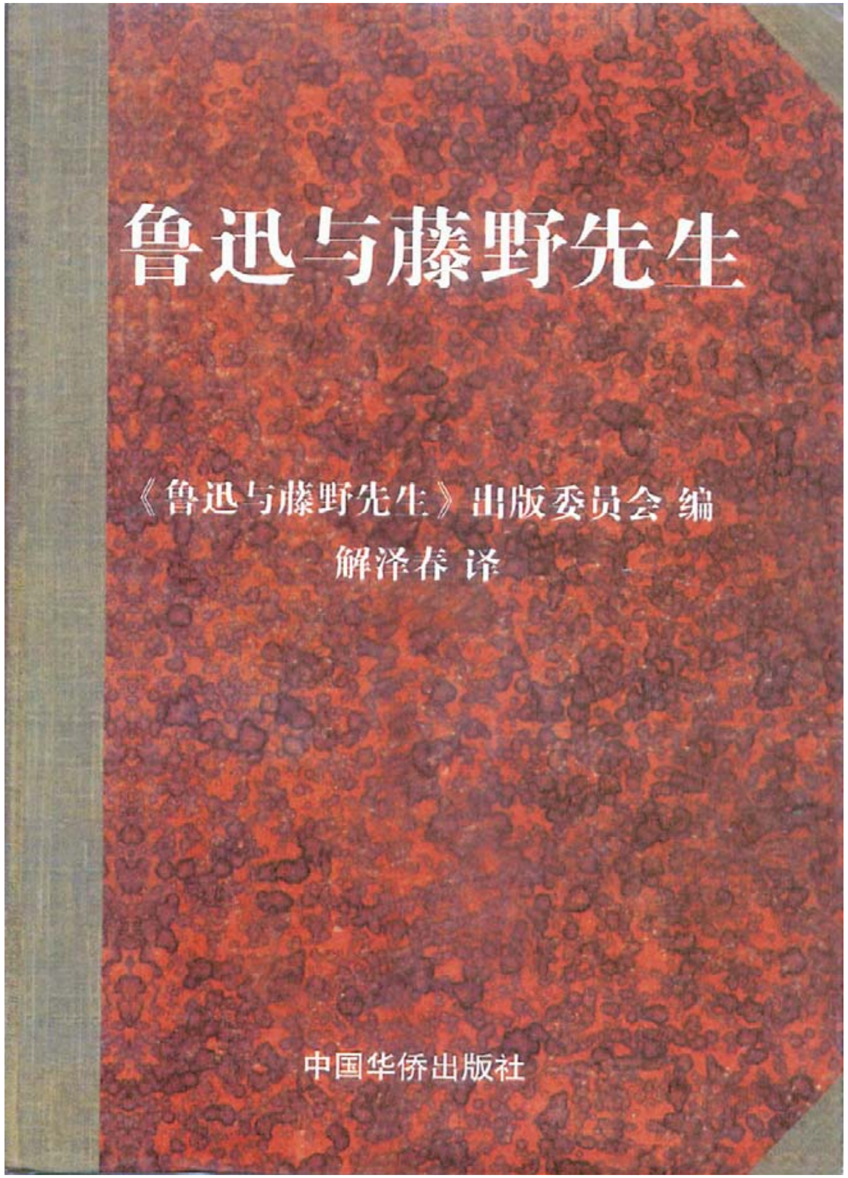
秋

冬



Lu Xian and Prof. Fujino

Book Cover



1st page

GSIS, Tohoku University

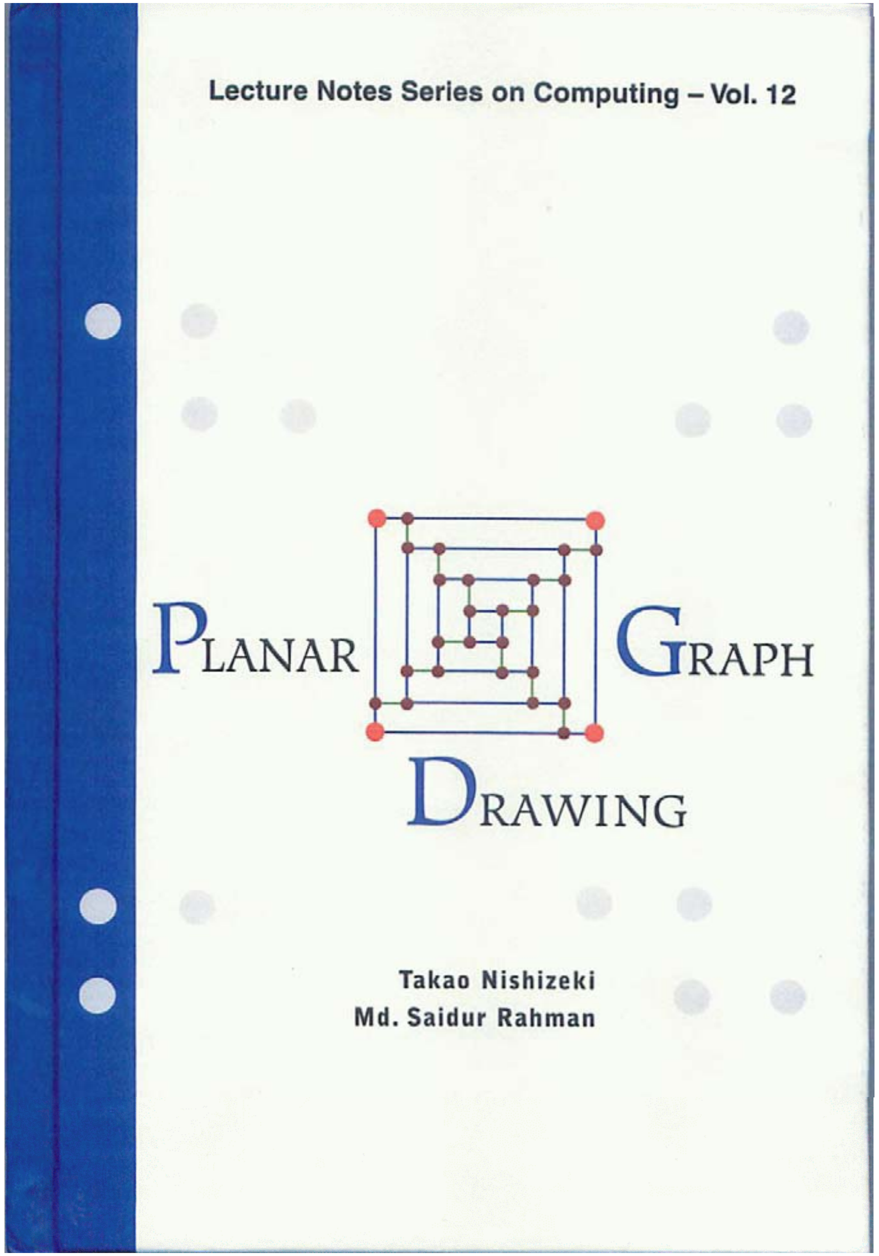
Graduate School of Information Sciences (**GSIS**), Tohoku University, was established in 1993. **情報科学研究科**

- 150 Faculties
- 450 students

- Math.
- Computer Science
- Robotics
- Transportation
- Economics
- Human Social Sciences

Interdisciplinary School

Book


Lecture Notes Series on Computing – Vol. 12
PLANAR GRAPH DRAWING
Takao Nishizeki
Md. Saidur Rahman

Octagonal Drawings of Plane Graphs with Prescribed Face Areas

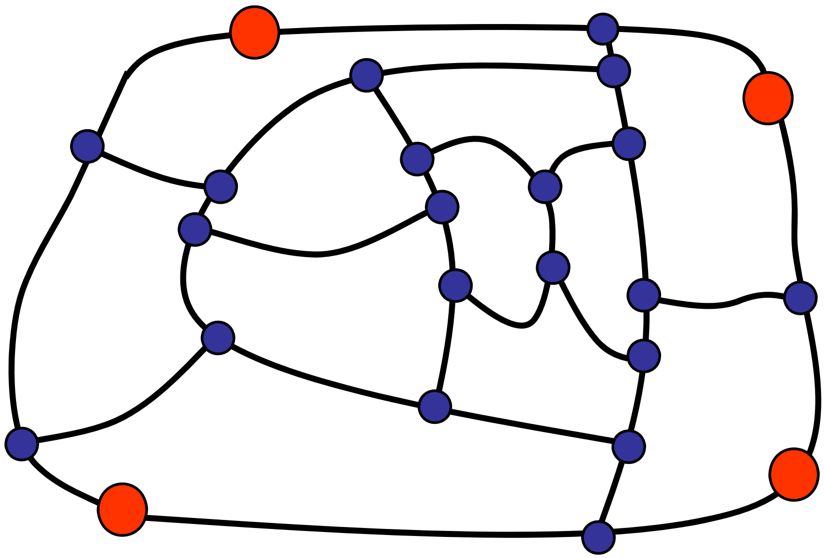
指定面積的平面図的八角形描画

Takao Nishizeki (Tohoku Univ.)

西関　隆夫　（東北大学）

ツディン　メンジディ　ピンメントウディ　バージョシン　ミョーワォ

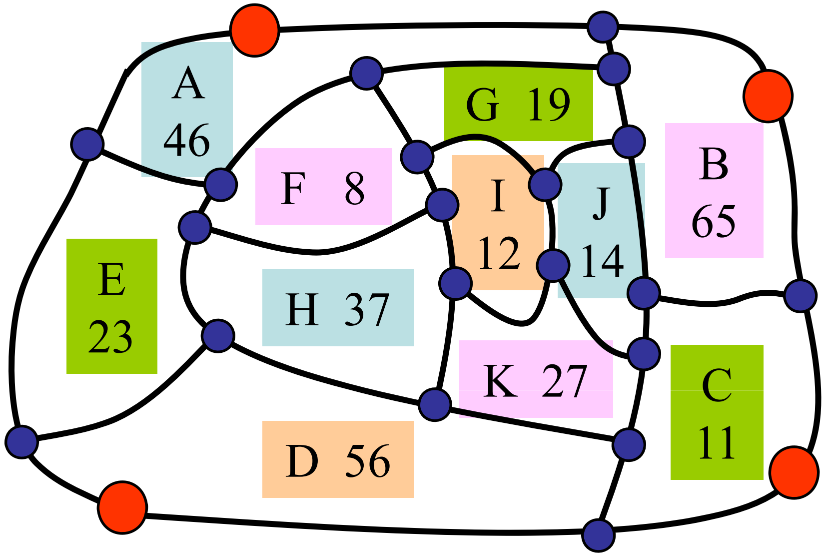
Prescribed-Area Octagonal Drawing



Input

Plane graph

Prescribed-Area Octagonal Drawing

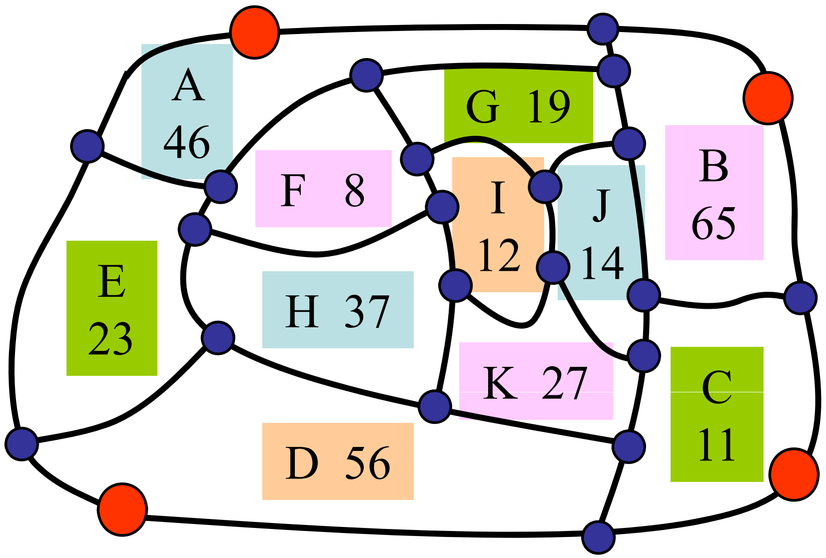


Input

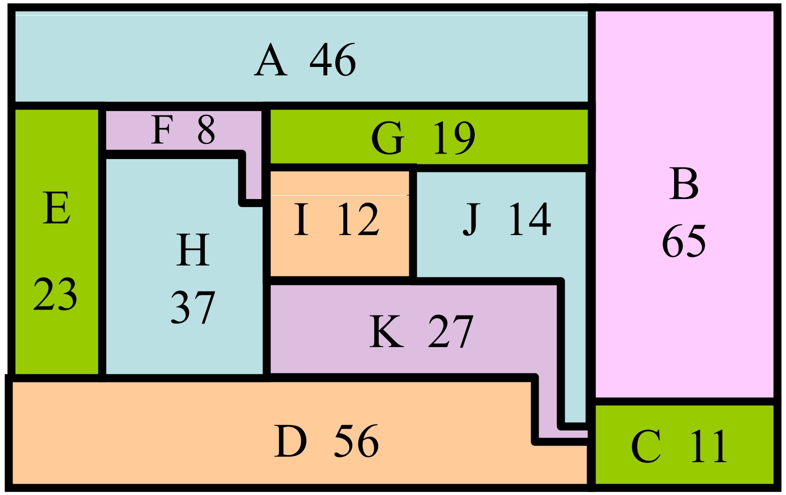
Plane graph

A real number for each inner face

Prescribed-Area Octagonal Drawing



Input

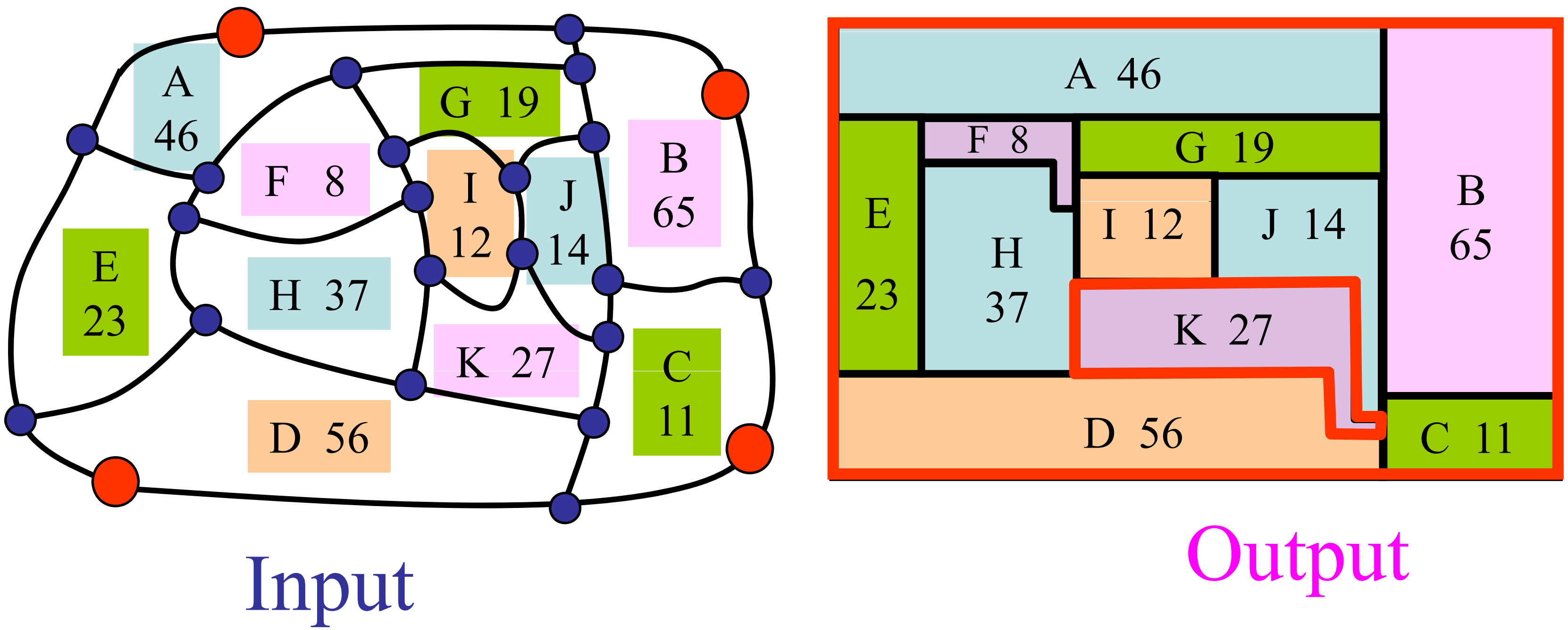


Output

Plane graph
A real number for each inner face

Prescribed-area
octagonal drawing

Prescribed-Area Octagonal Drawing

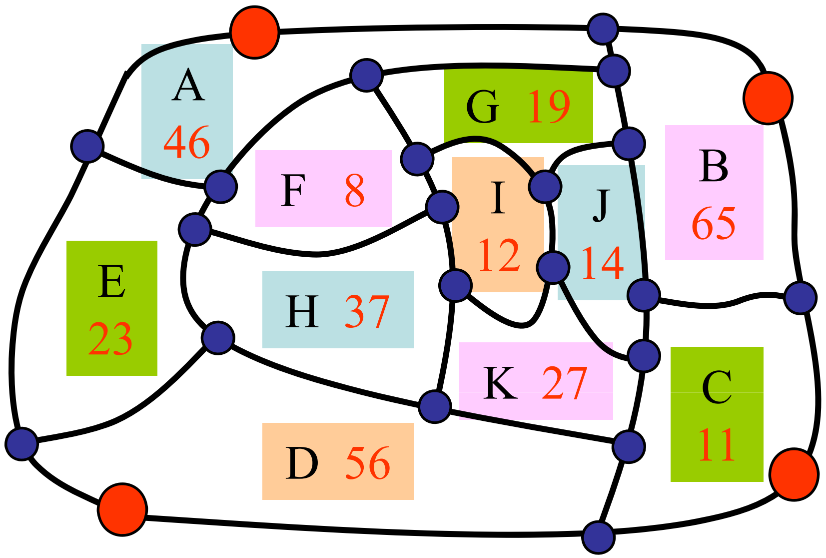


Each inner face is drawn as a rectilinear polygon of at most eight corners.

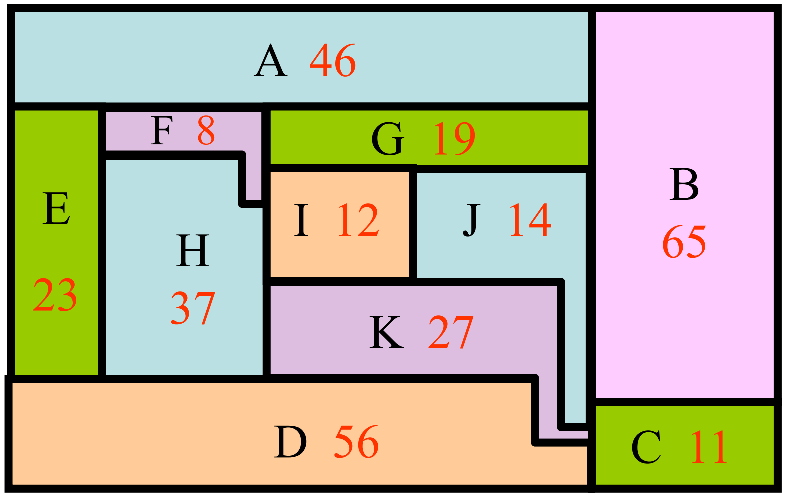
The outer face is drawn as a rectangle.

Each face has its prescribed area.

Prescribed-Area Octagonal Drawing



Input



Output

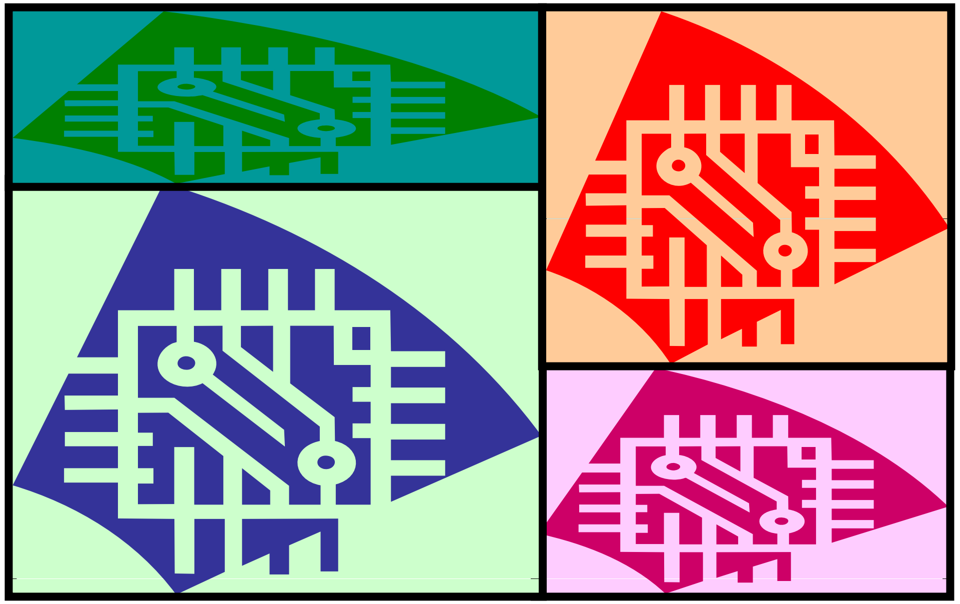
Each inner face is drawn as a rectilinear polygon of at most eight corners.

The outer face is drawn as a rectangle.

Each face has its prescribed area.

Applications

VLSI Floorplanning



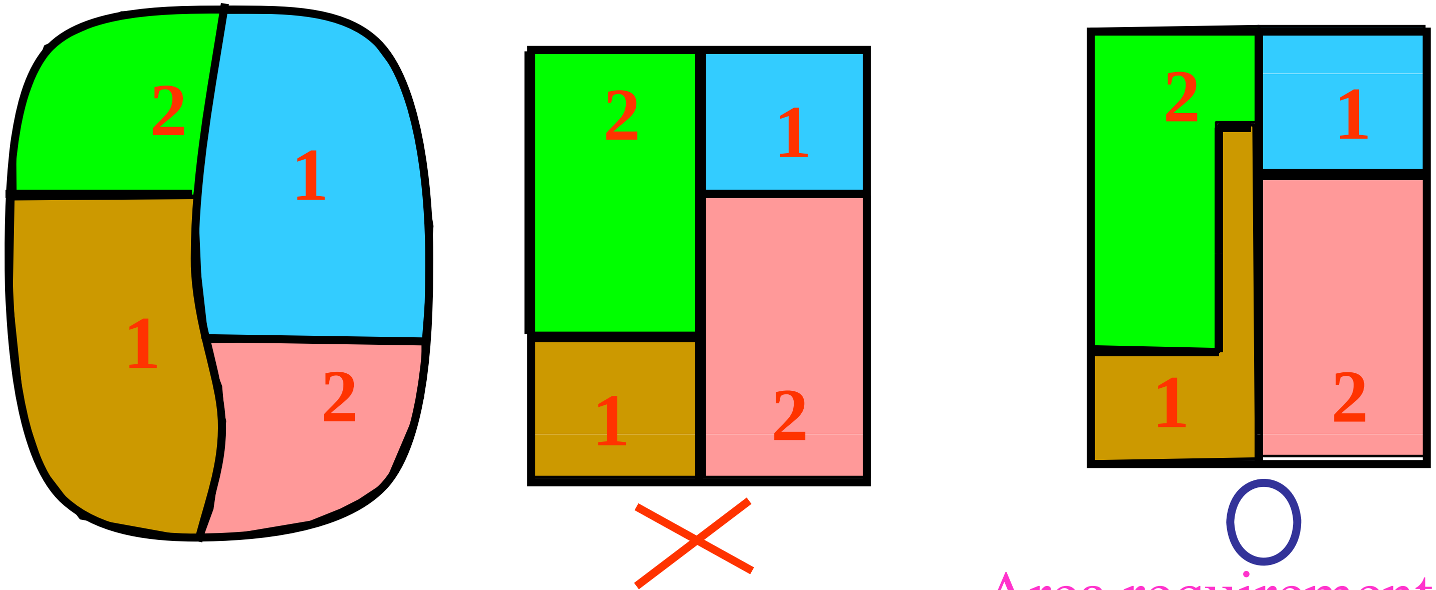
Obtained by subdividing a given rectangle into smaller rectangles.

Each smaller rectangle corresponds to a module.

Each module has **area requirements.**

Applications

VLSI Floorplanning



Area requirements cannot be satisfied if each module is allowed to be only a rectangle.

Area requirements can be satisfied if each module is allowed to be a simple rectilinear polygon.

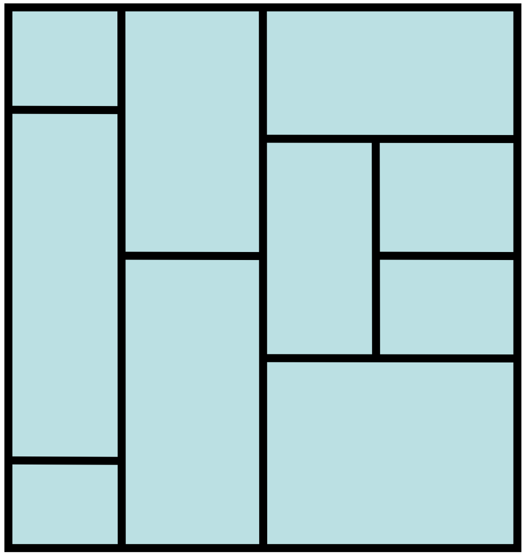
It is desirable to keep the shape of each rectilinear polygon as simple as possible.

Our Results

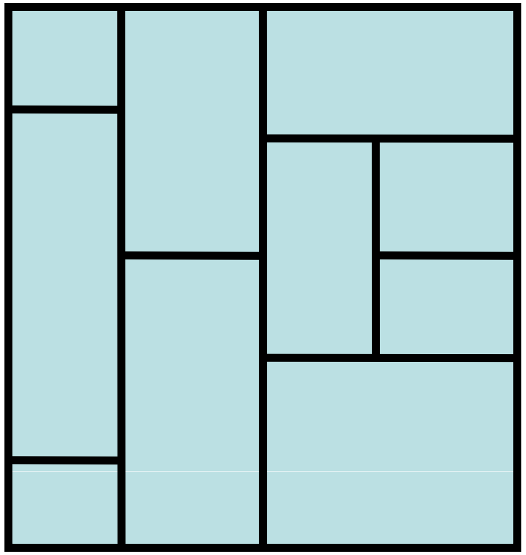
G: a good slicing graph

$O(n)$ time algorithm.

Slicing Floorplan

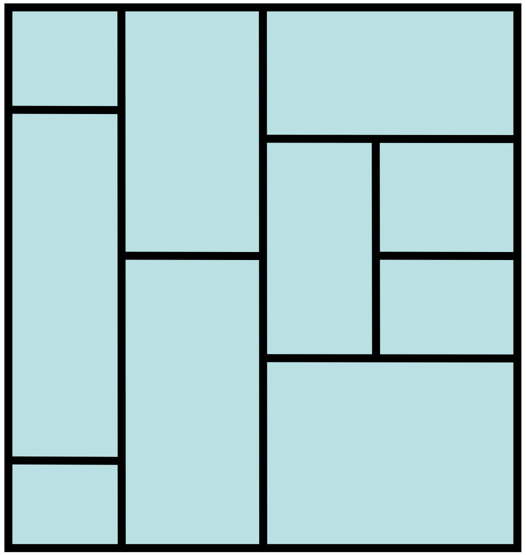


Slicing Floorplan

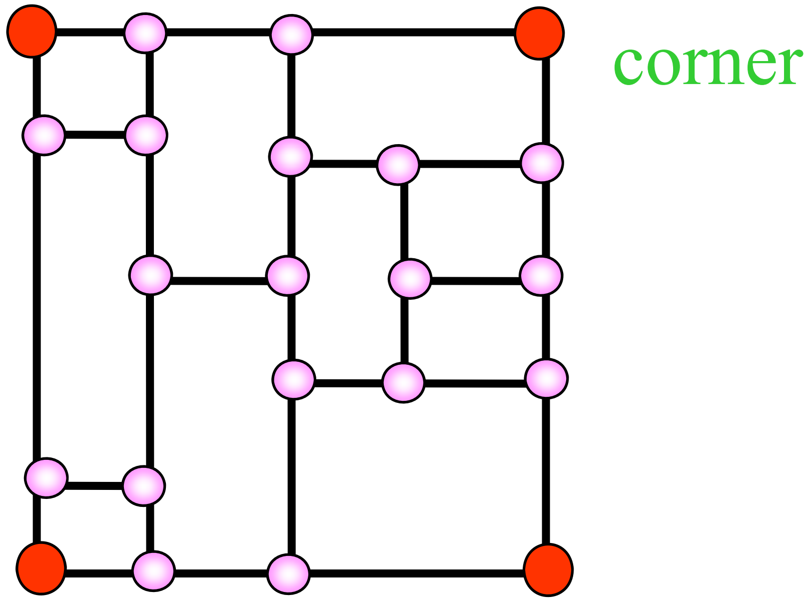


A slicing floorplan can be obtained by repeatedly subdividing rectangles horizontally or vertically.

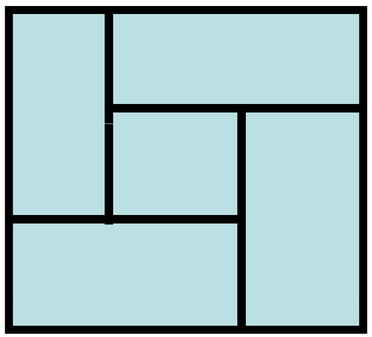
Slicing Floorplan and Slicing Graph



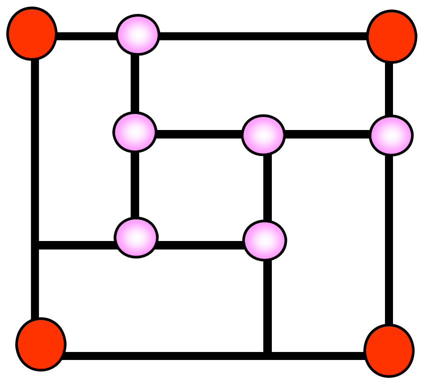
Slicing Floorplan



Slicing Graph

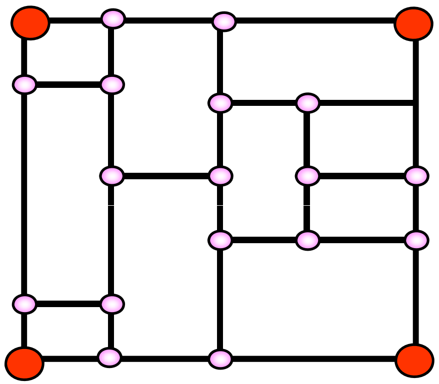


Not a Slicing Floorplan

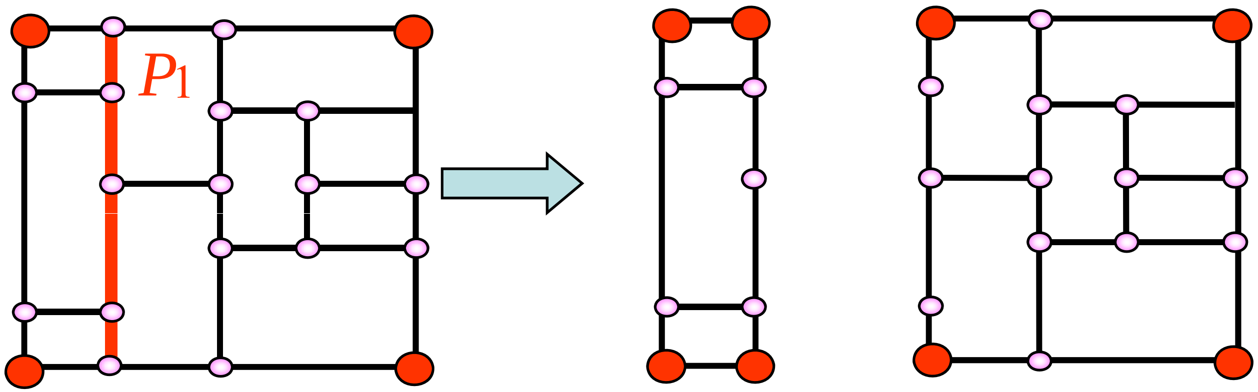


Not a Slicing Graph

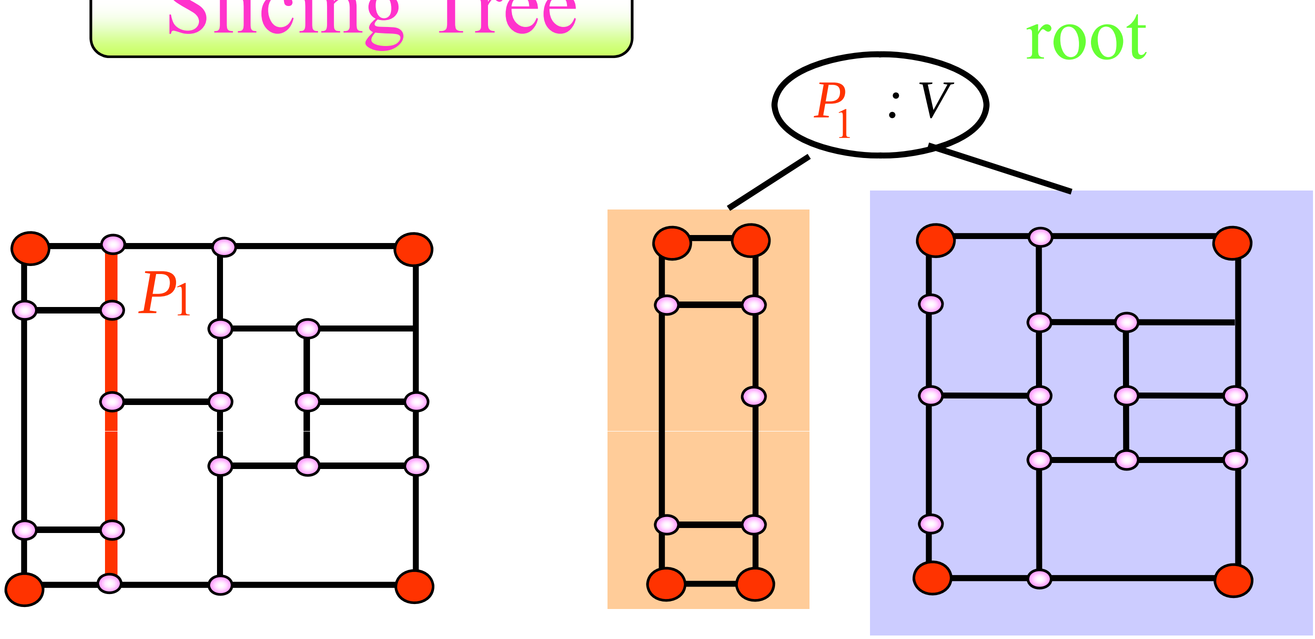
Slicing Tree



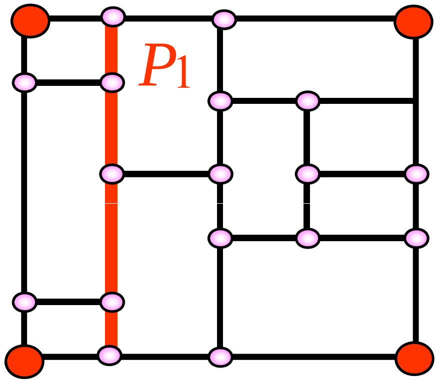
Slicing Tree

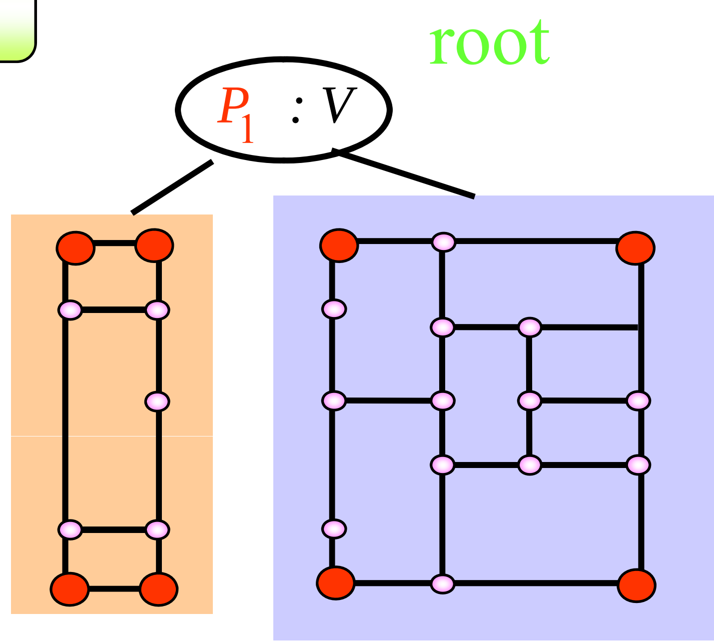


Slicing Tree



Slicing Tree

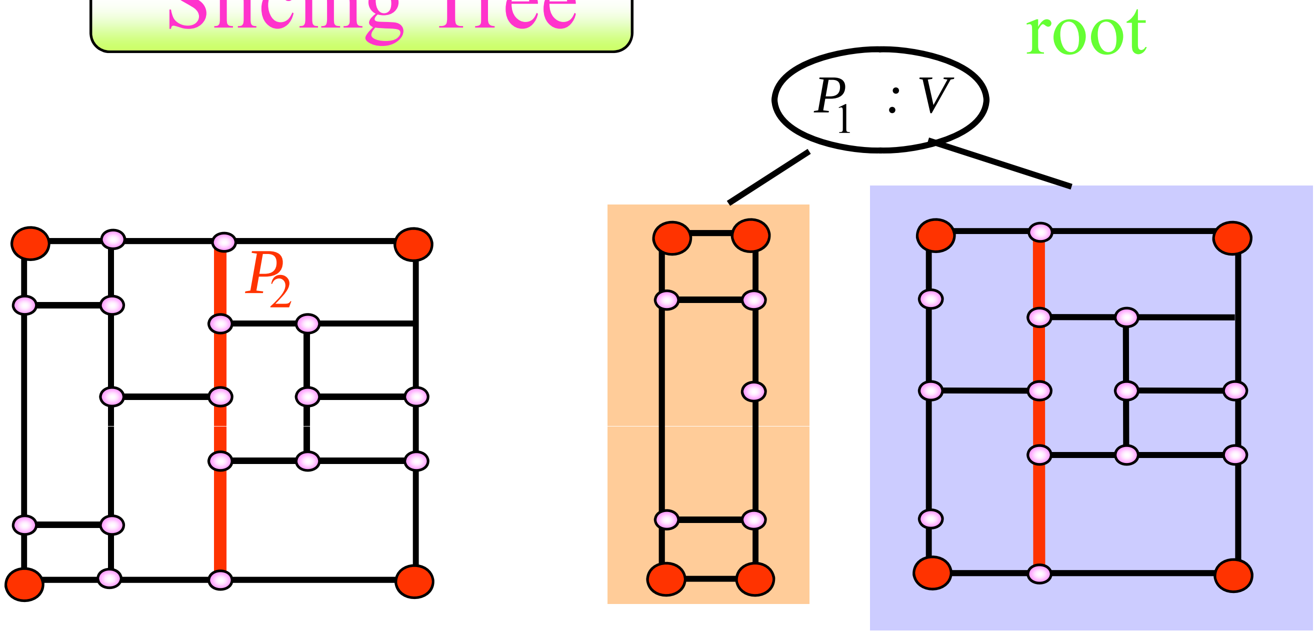




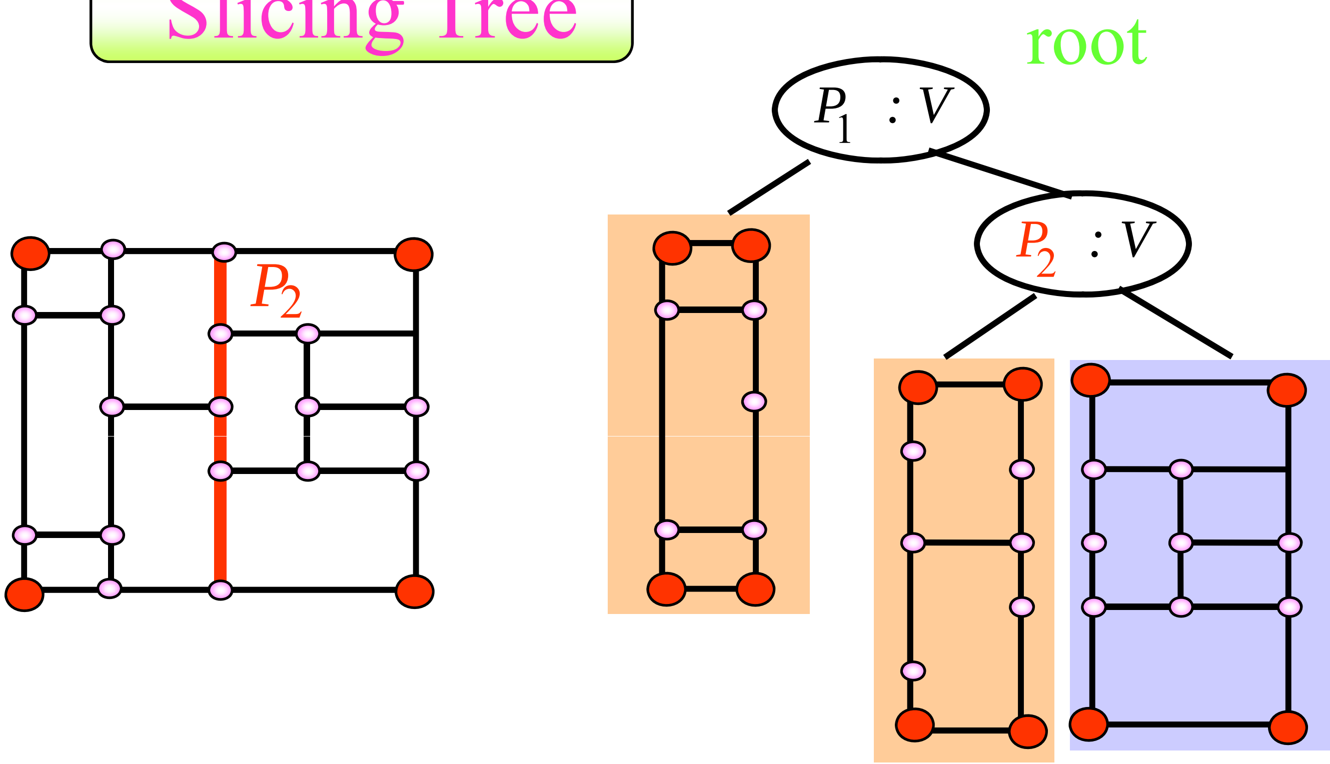
Right subgraph becomes right subtree

Left subgraph becomes left subtree

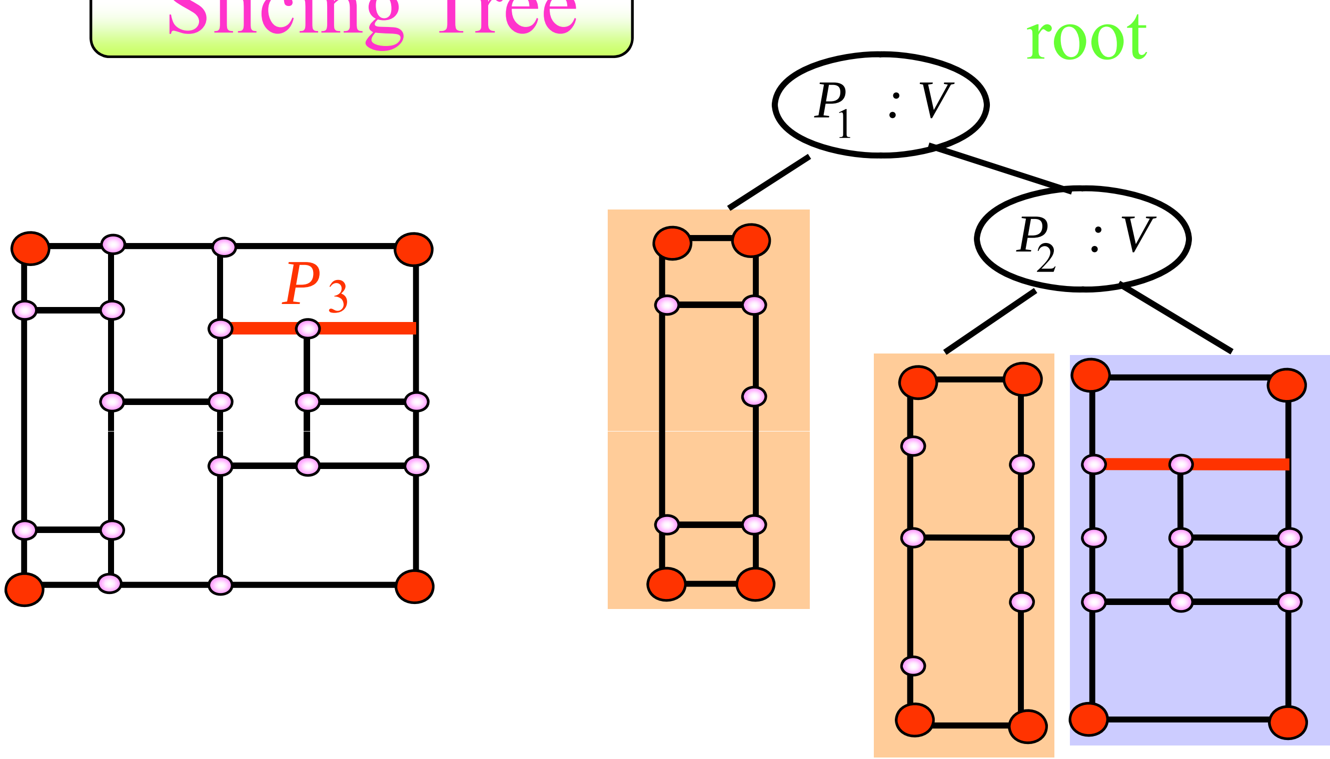
Slicing Tree



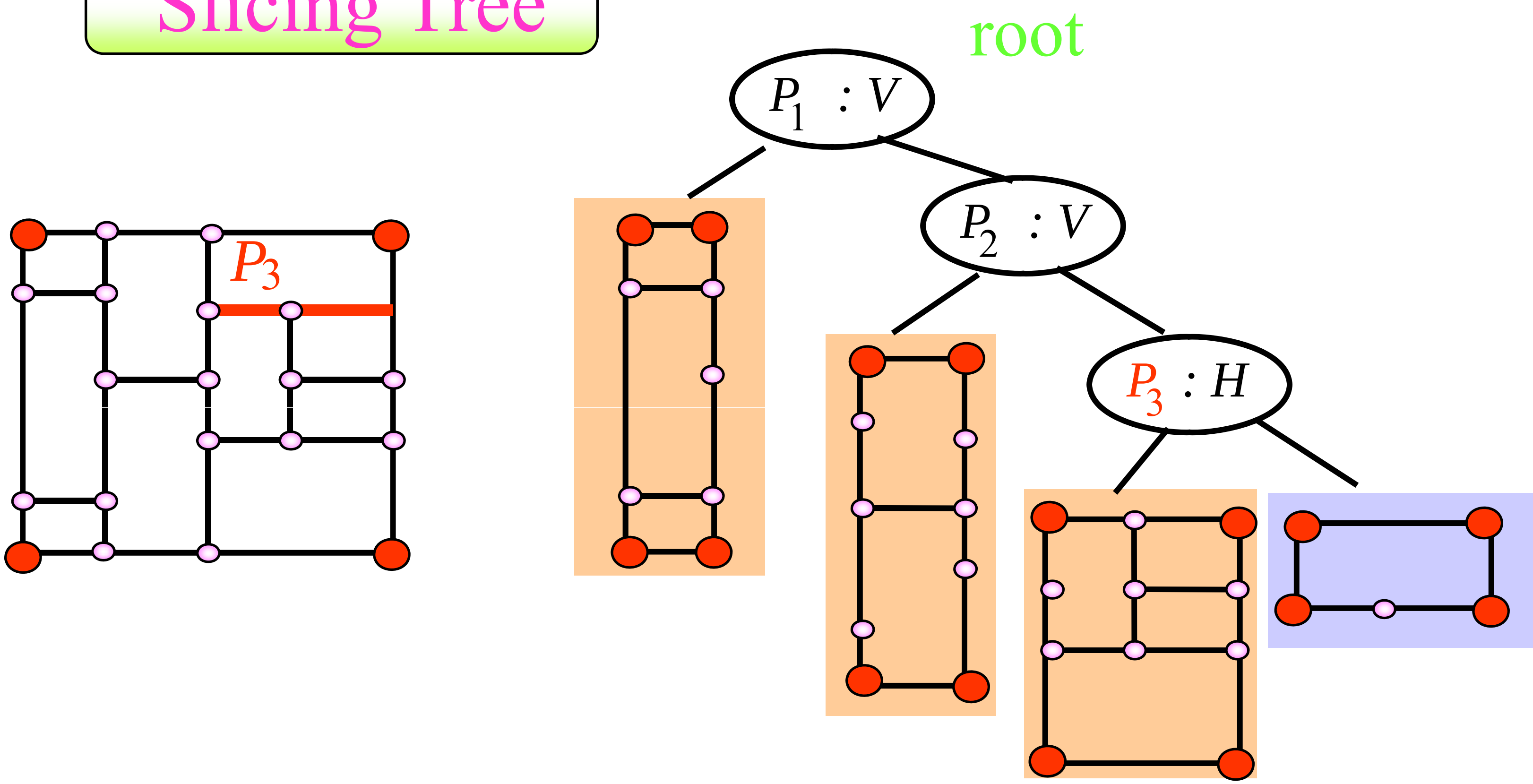
Slicing Tree



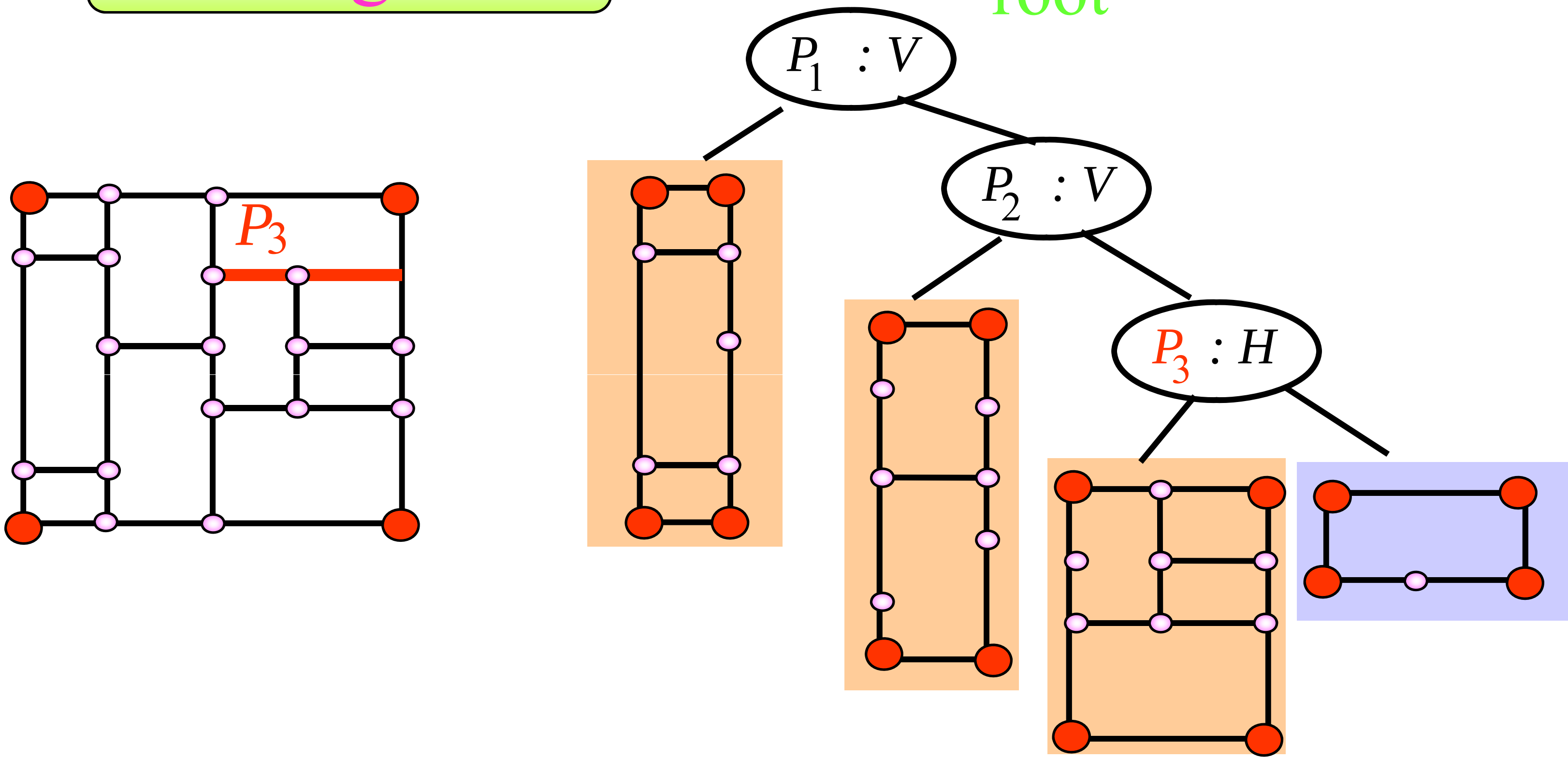
Slicing Tree



Slicing Tree



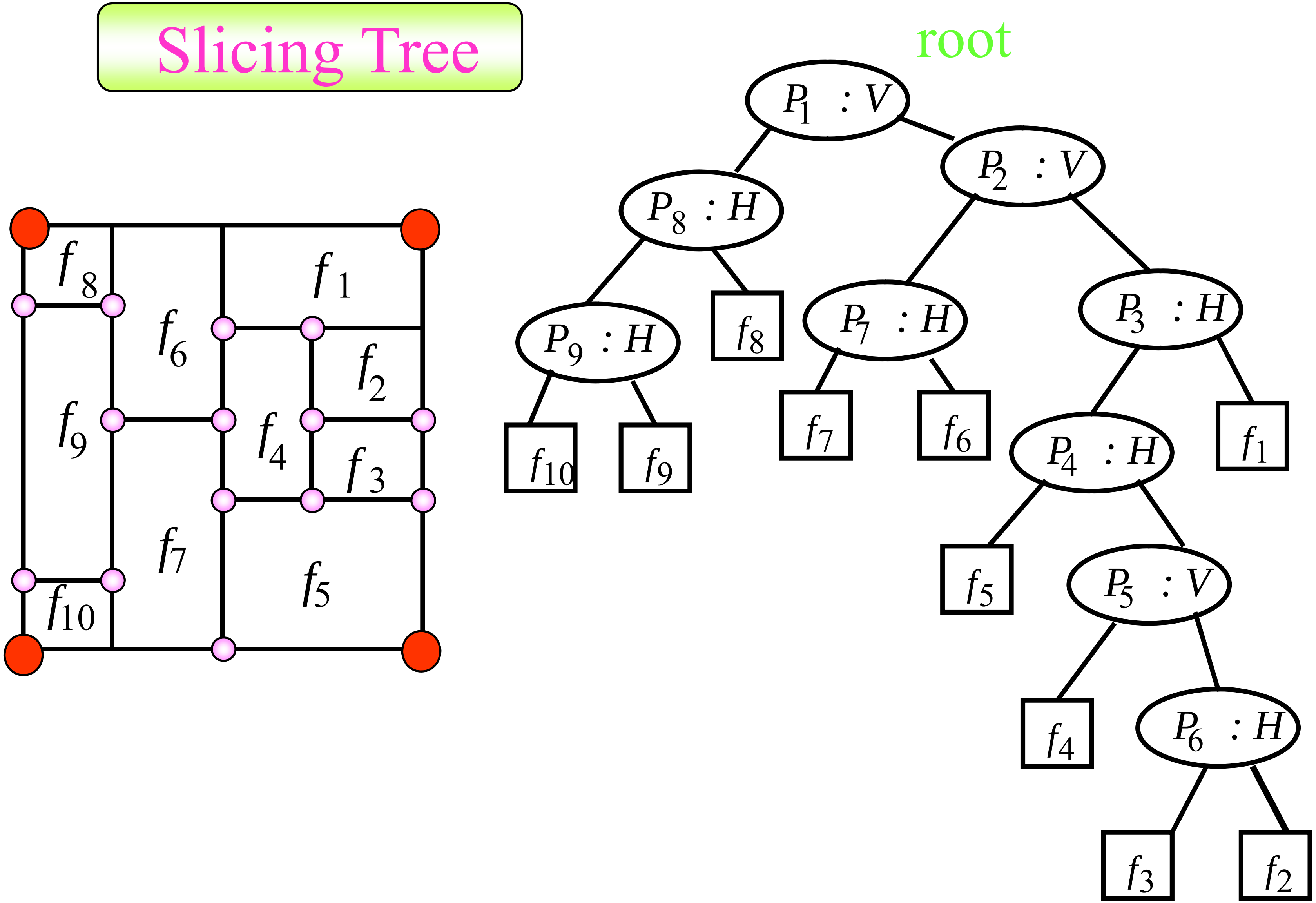
Slicing Tree



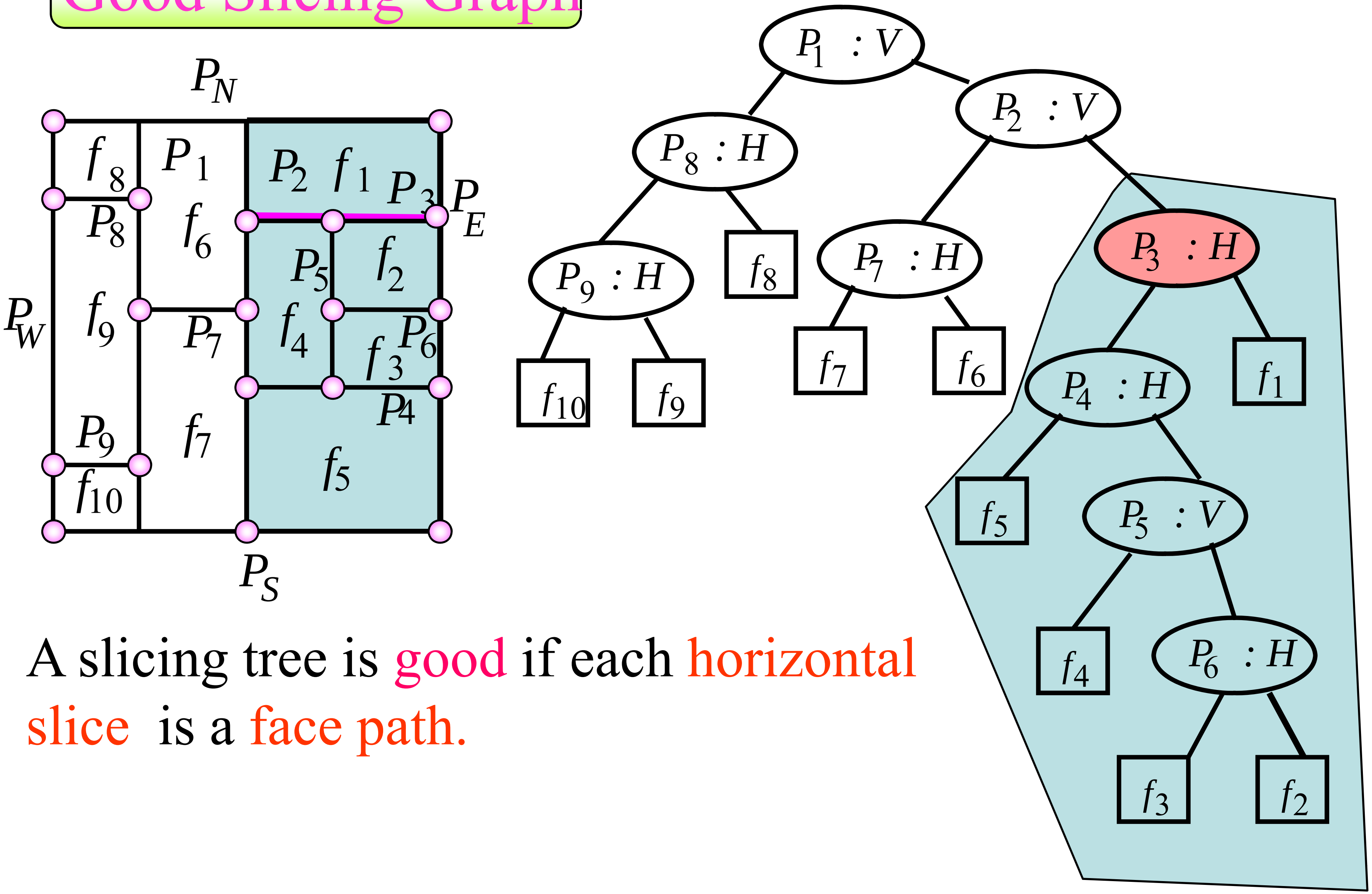
Upper subgraph becomes right subtree

Lower subgraph becomes left subtree

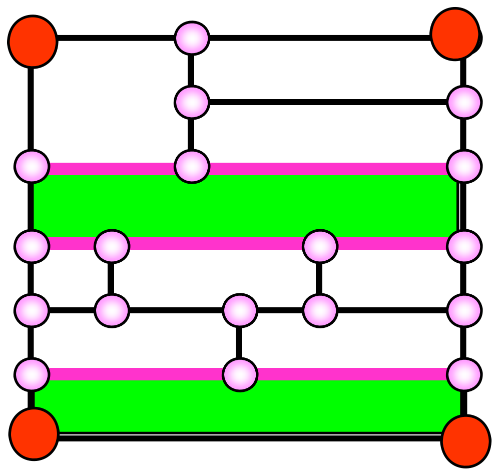
Slicing Tree



Good Slicing Graph

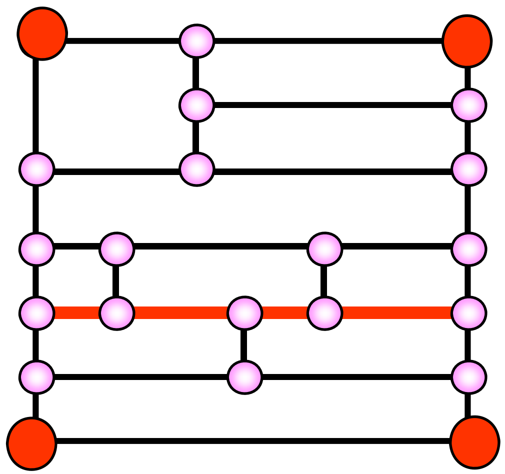


A slicing tree is good if each horizontal slice is a face path.



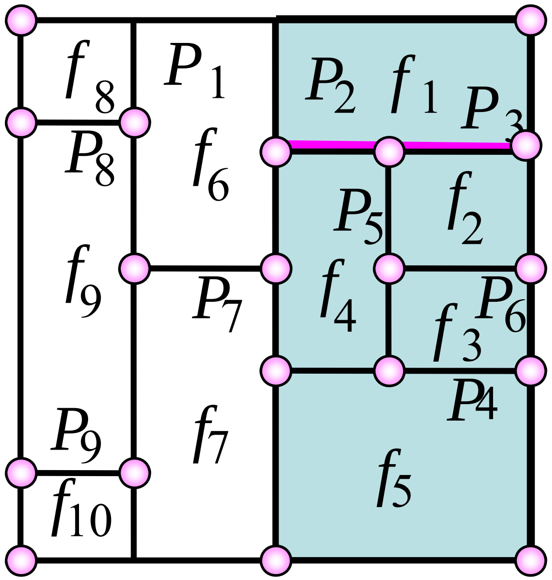
Three face paths

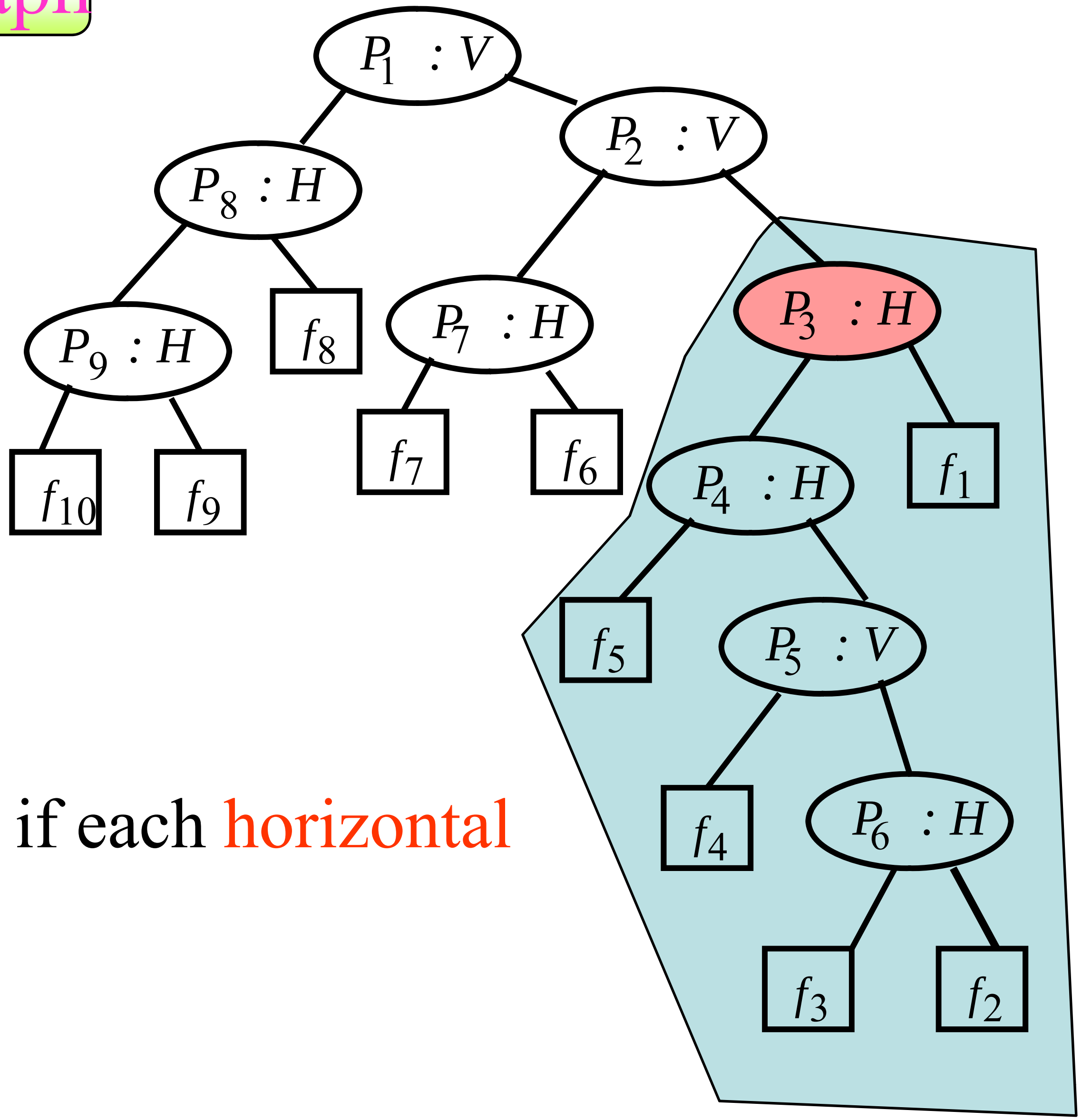
On a boundary of a single face



Not a face path

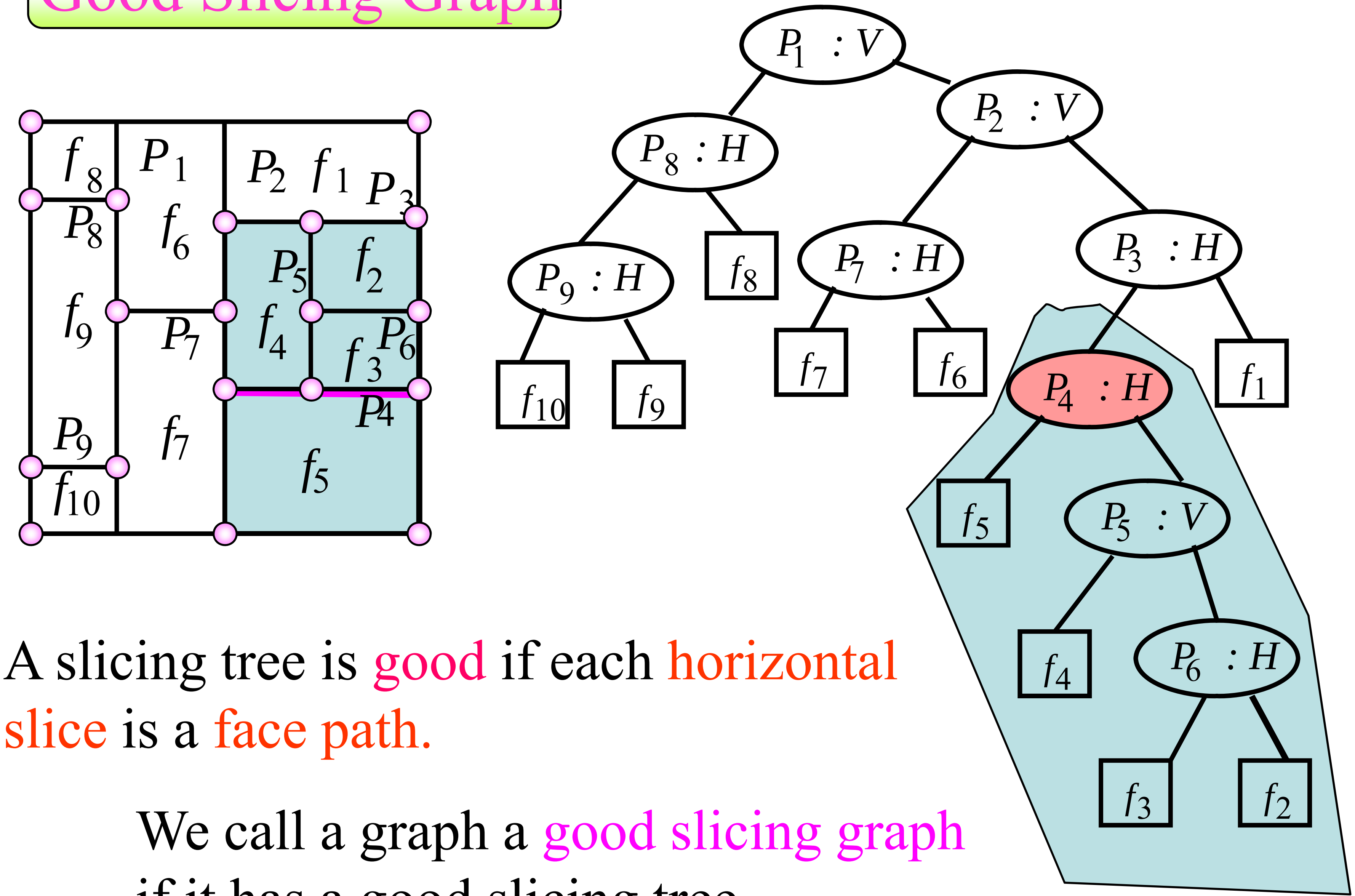
Good Slicing Graph





A slicing tree is good if each horizontal slice is a face path.

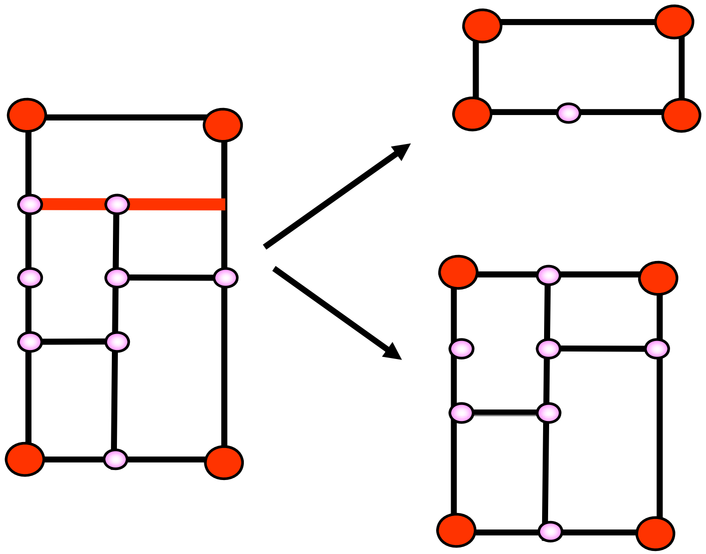
Good Slicing Graph



A slicing tree is good if each horizontal slice is a face path.

We call a graph a good slicing graph if it has a good slicing tree.

Good Slicing Graph

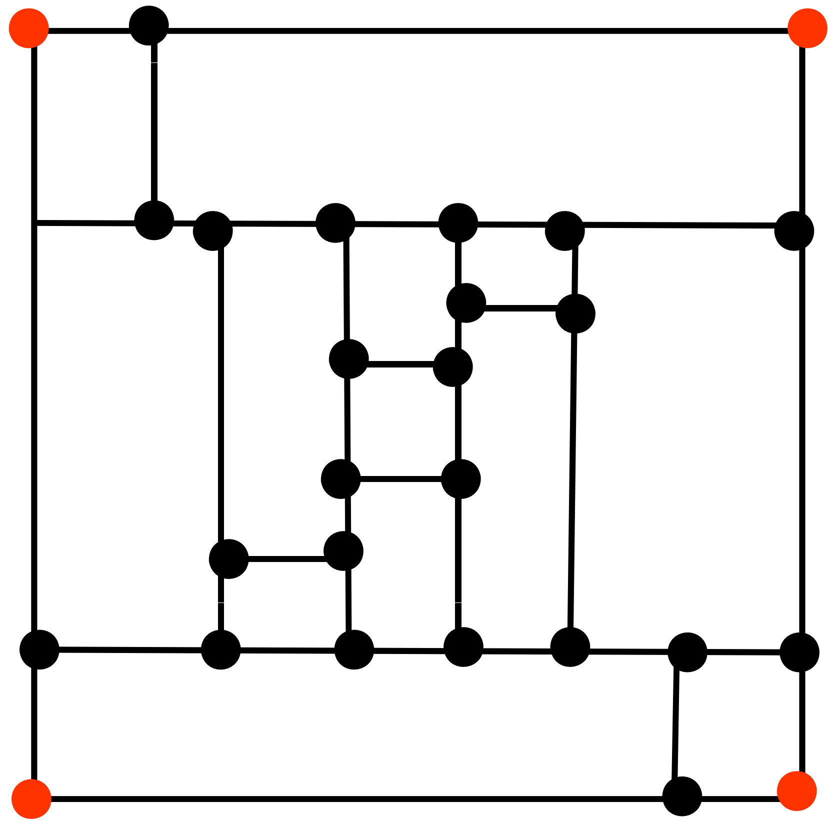


Cannot be vertically sliced

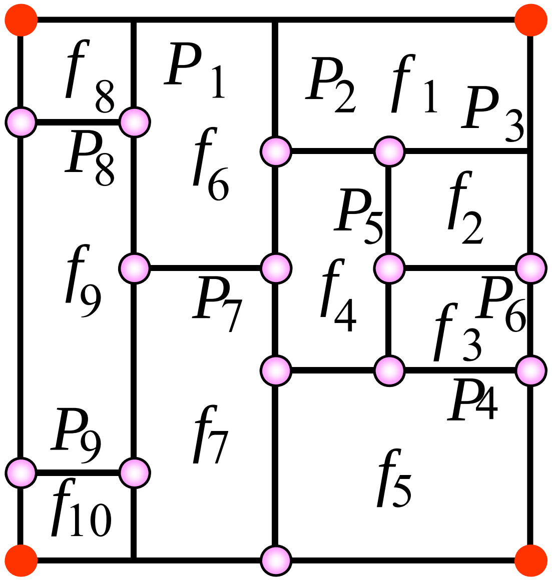
Can be vertically sliced

For a horizontal slice, at least one of the upper subgraph and the lower subgraph cannot be vertically sliced.

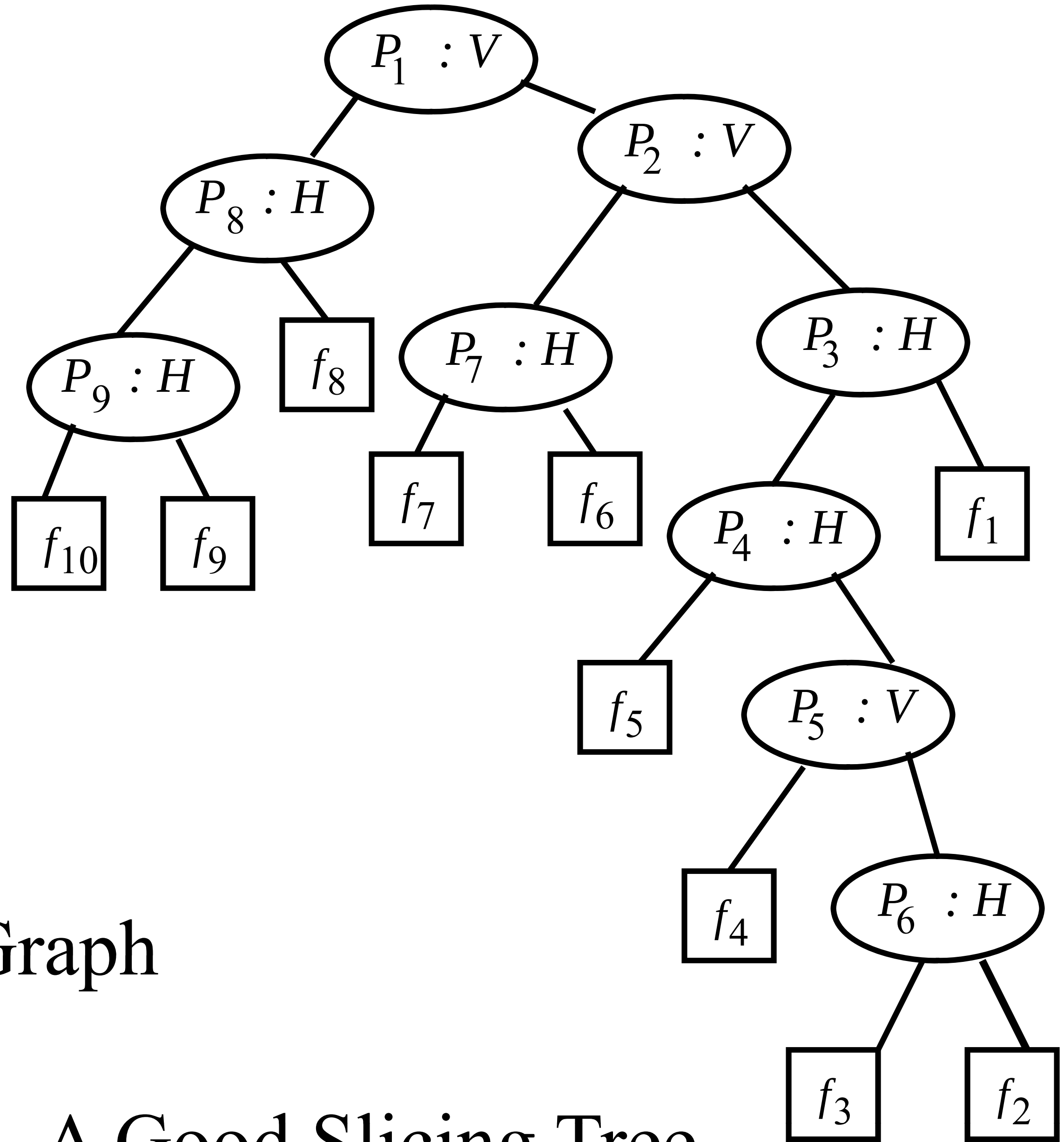
Not every slicing graph is a good slicing graph.



Input



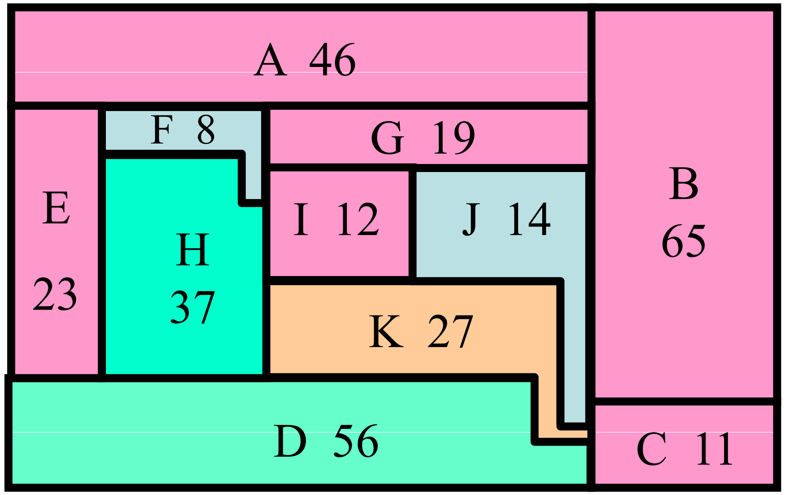
A Good Slicing Graph



A Good Slicing Tree

Output

Prescribed-area
Octagonal drawing

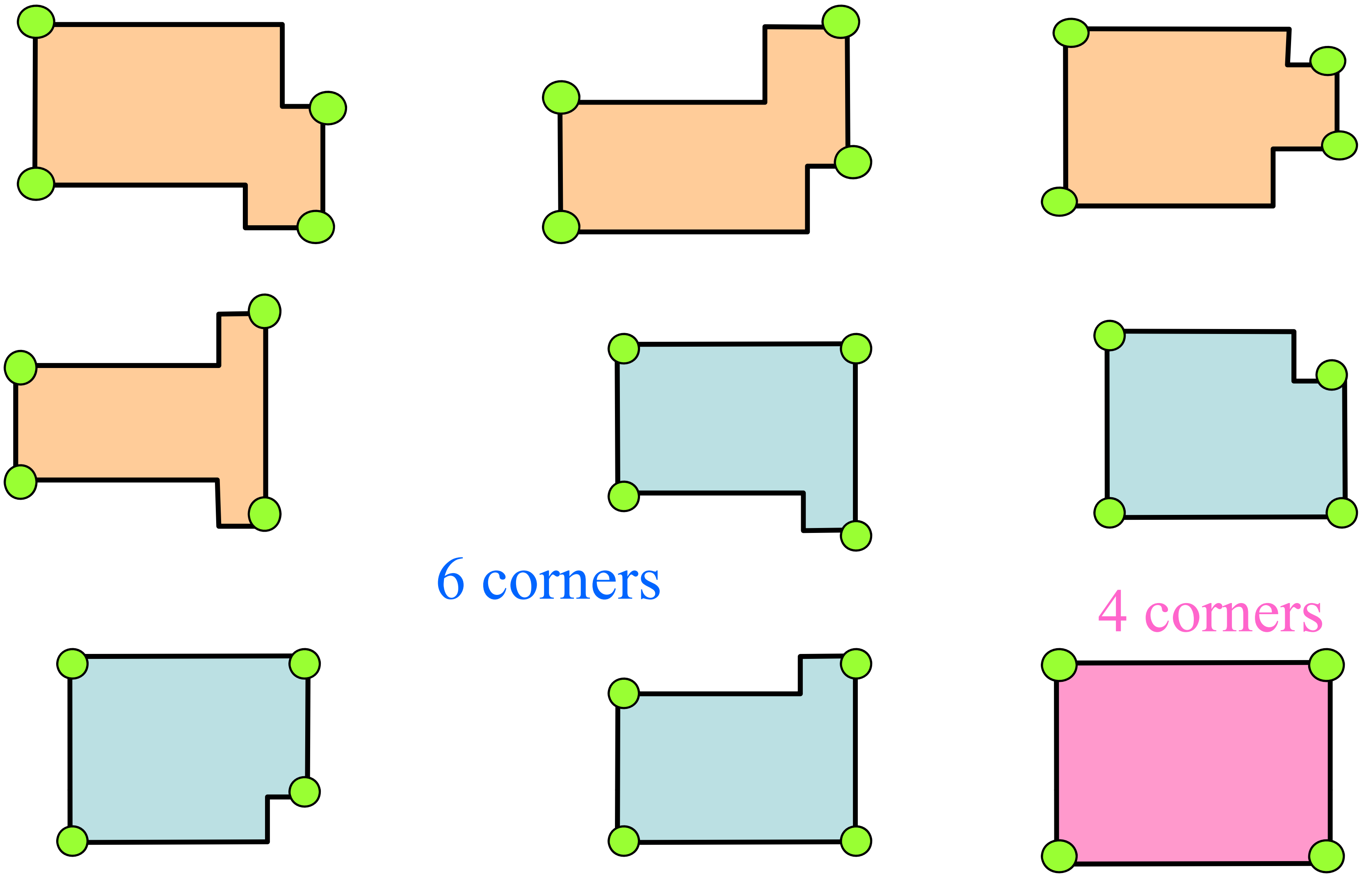


Each inner face is drawn as a rectilinear polygon with at most eight corners.

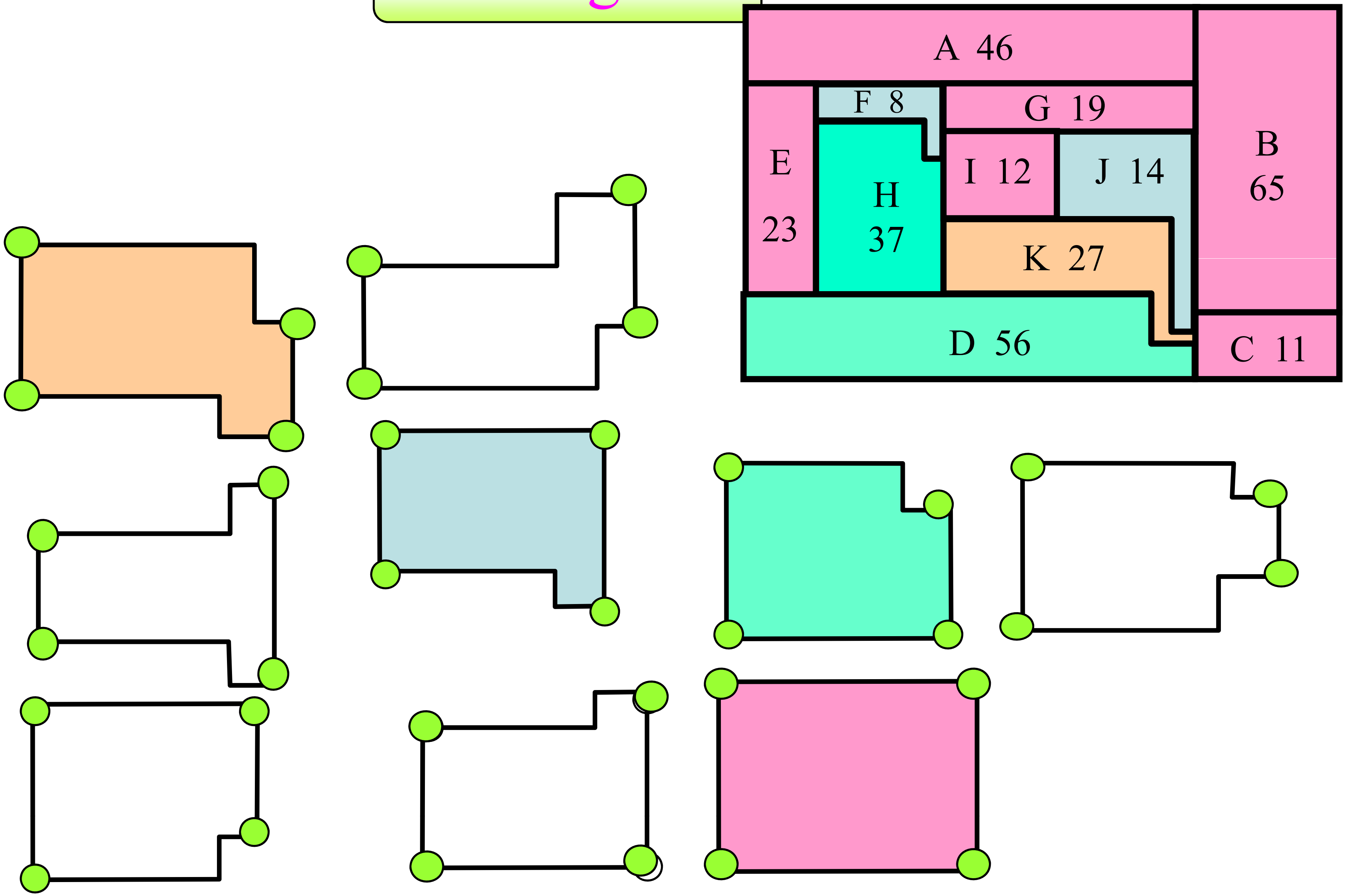
Particularly, each inner face is drawn as a rectilinear polygon of the following nine shapes.

Octagons

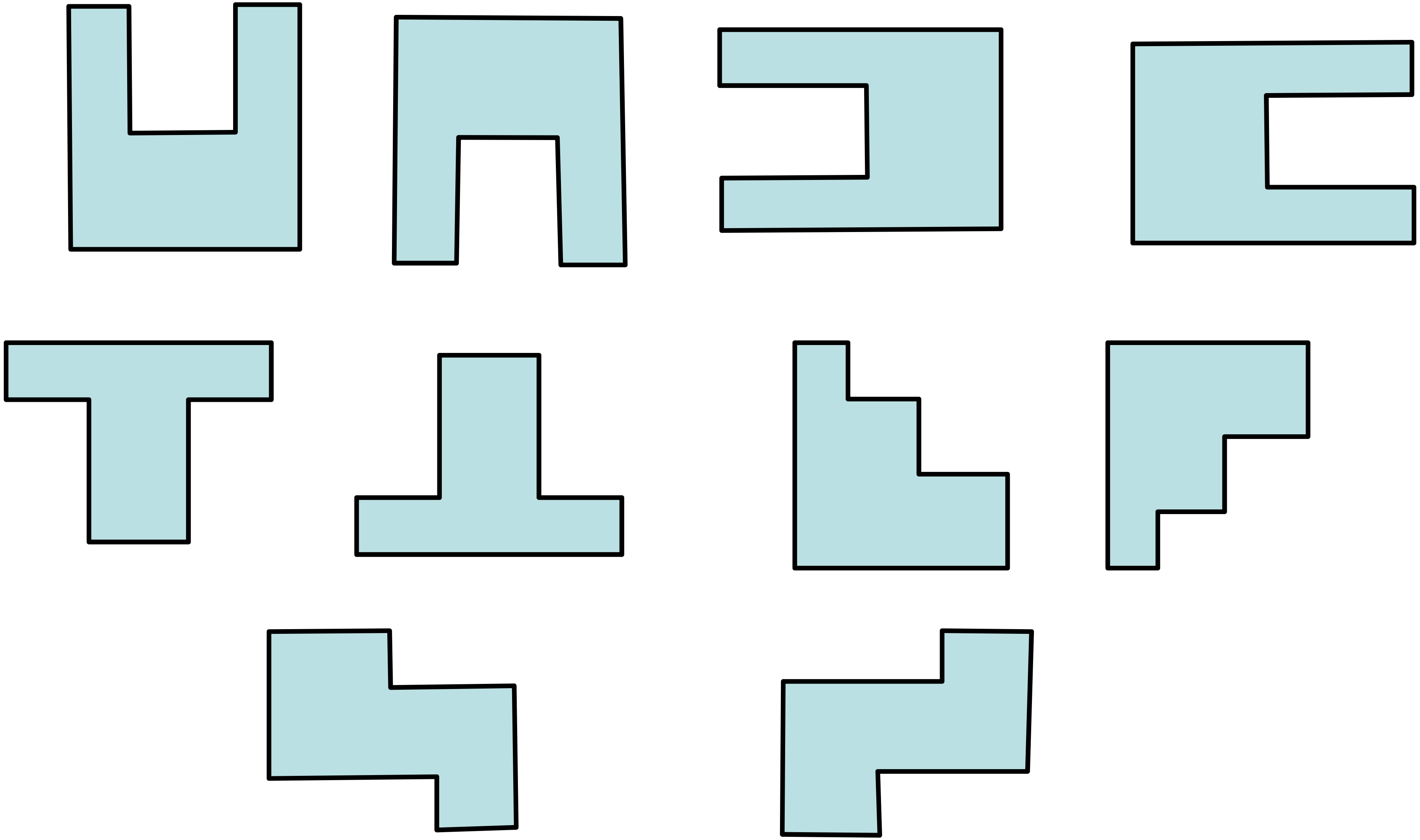
8 corners



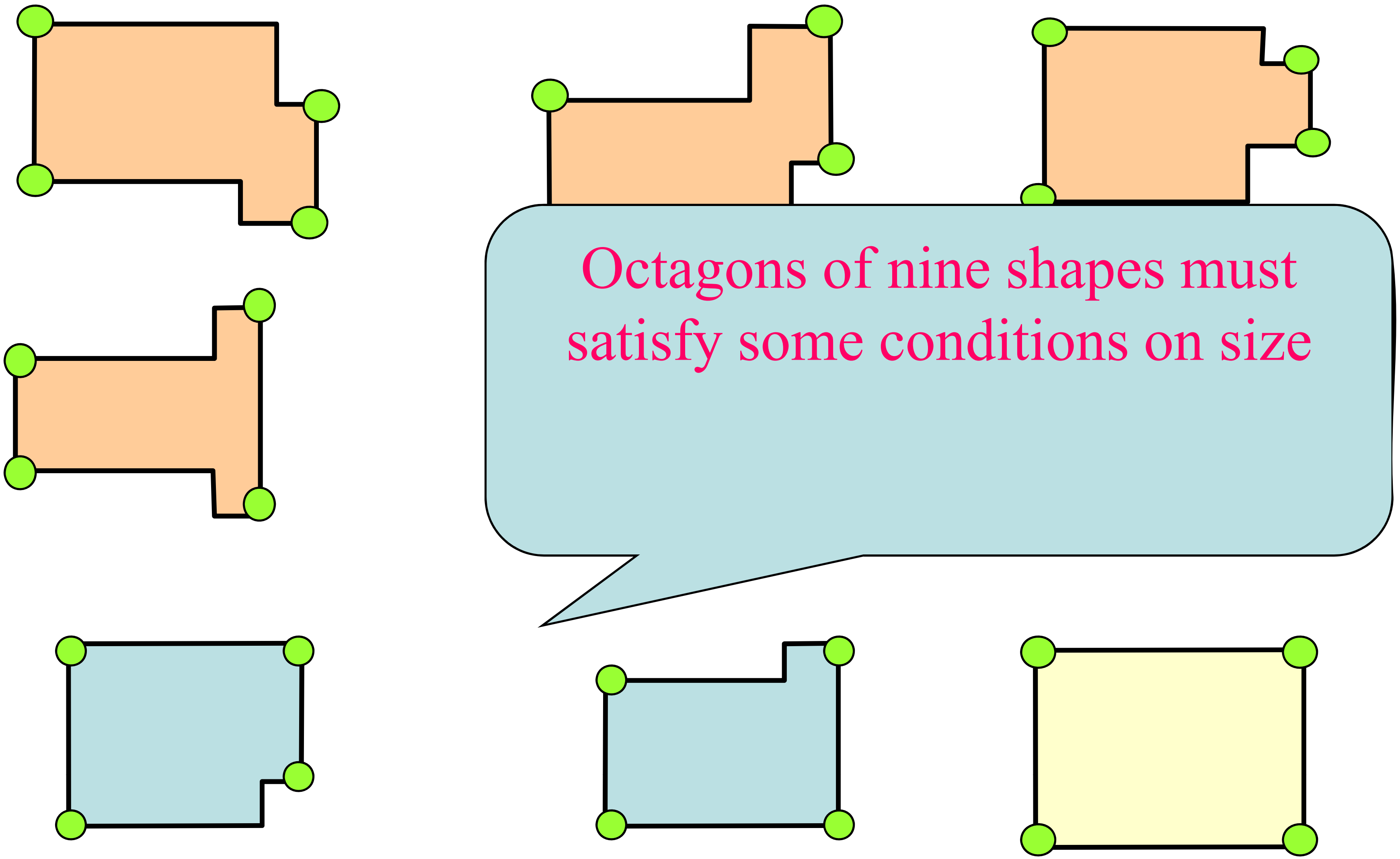
Octagons



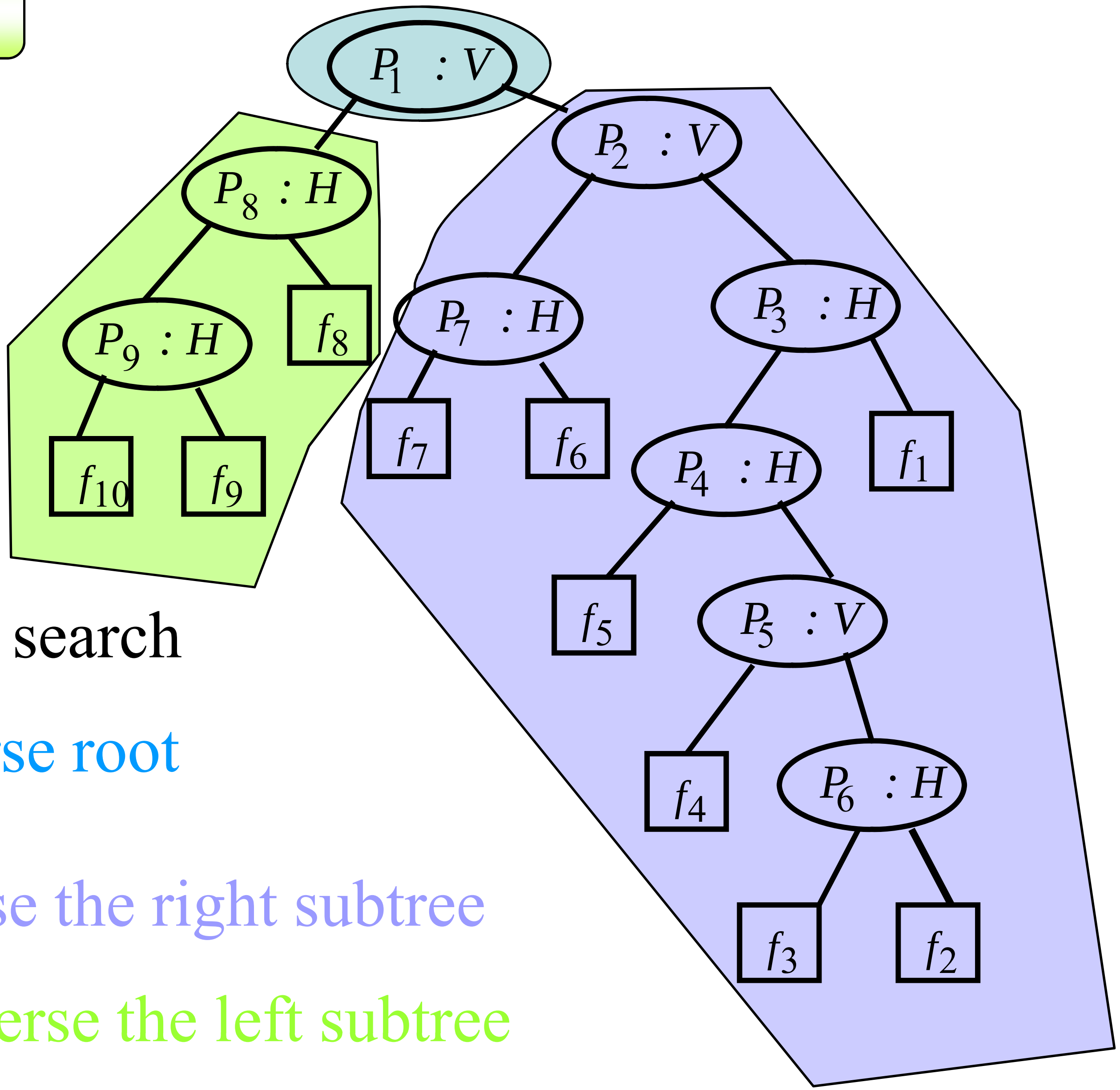
Do not appear



Feasible Octagons



Algorithm



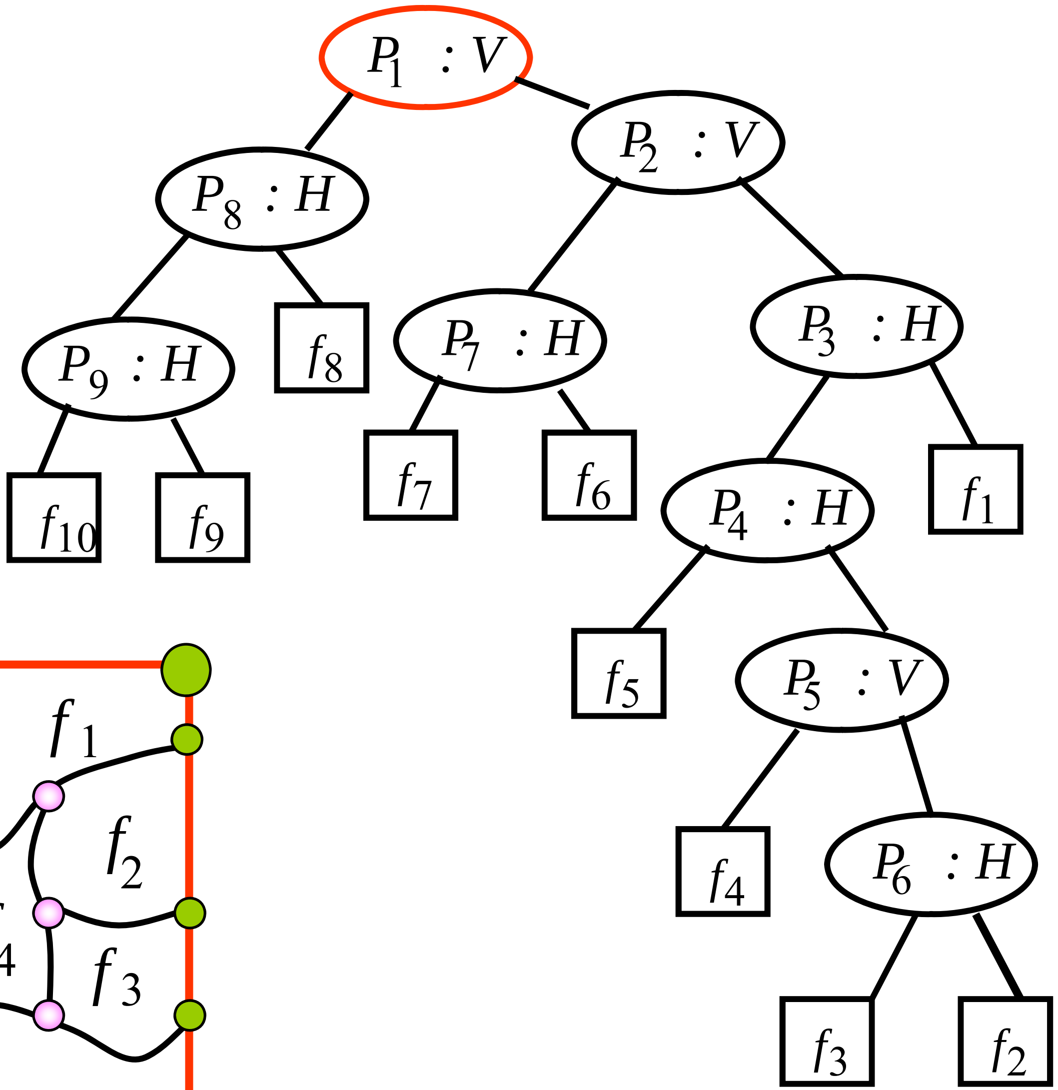
Depth-first search

First traverse root

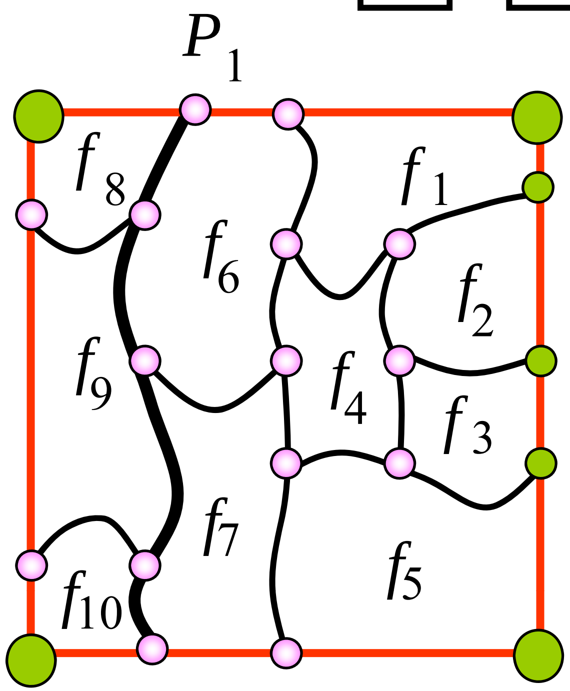
Then traverse the right subtree

Finally, traverse the left subtree

Algorithm

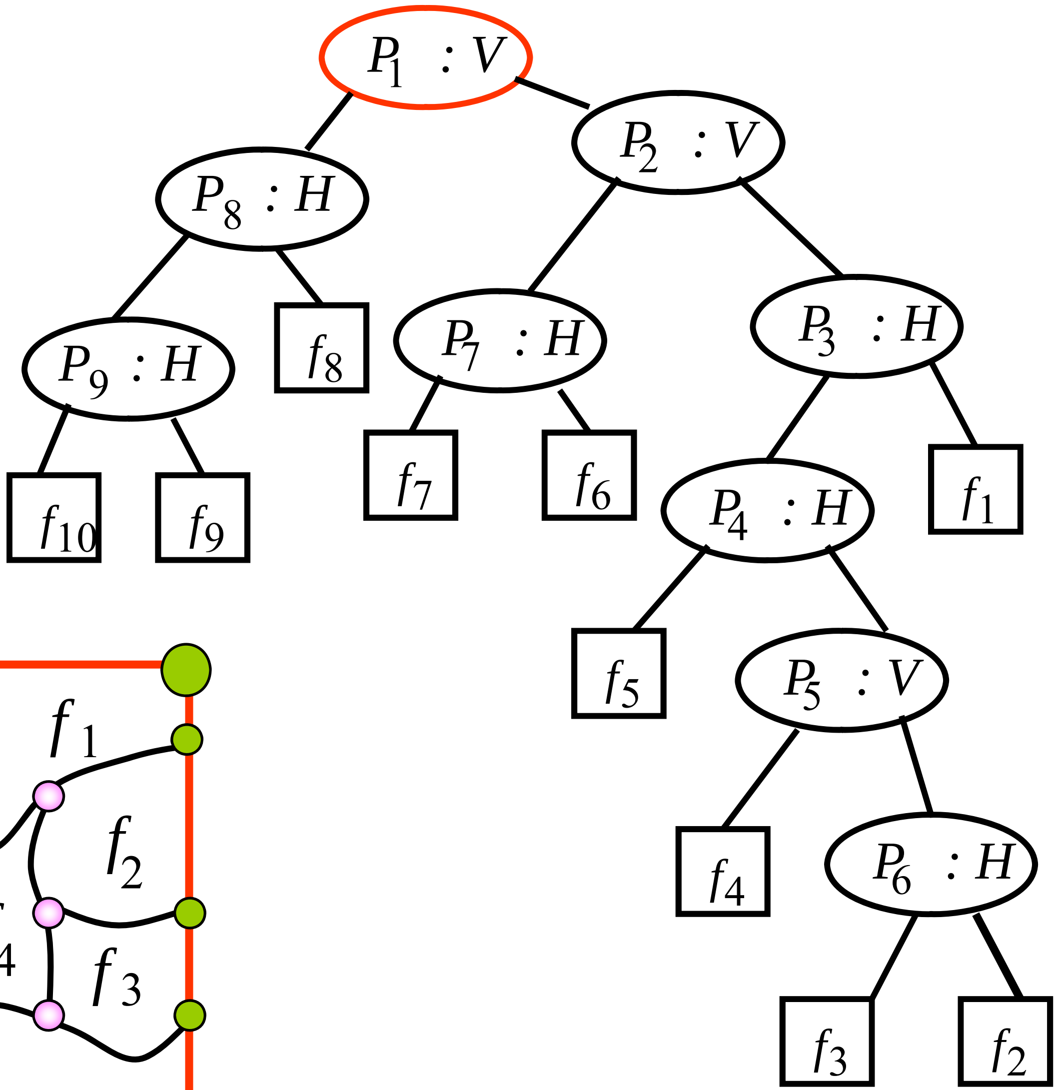


$P_1 : V$
$P_8 : H$
$P_2 : V$
$P_9 : H$
f_8
$P_7 : H$
$P_3 : H$
f_{10}
f_9
f_7
f_6
$P_4 : H$
f_1
f_5
$P_5 : V$
f_4
$P_6 : H$
f_3
f_2

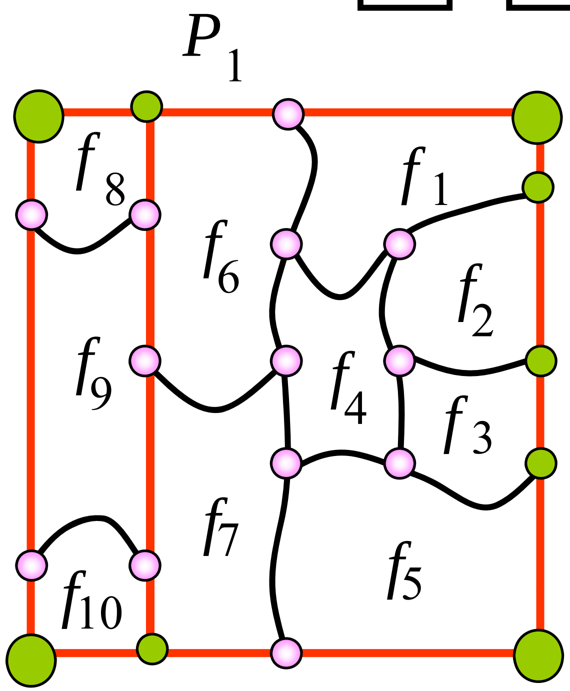


P_1
f_8
f_6
f_1
f_2
f_9
f_4
f_3
f_7
f_5
f_{10}

Algorithm

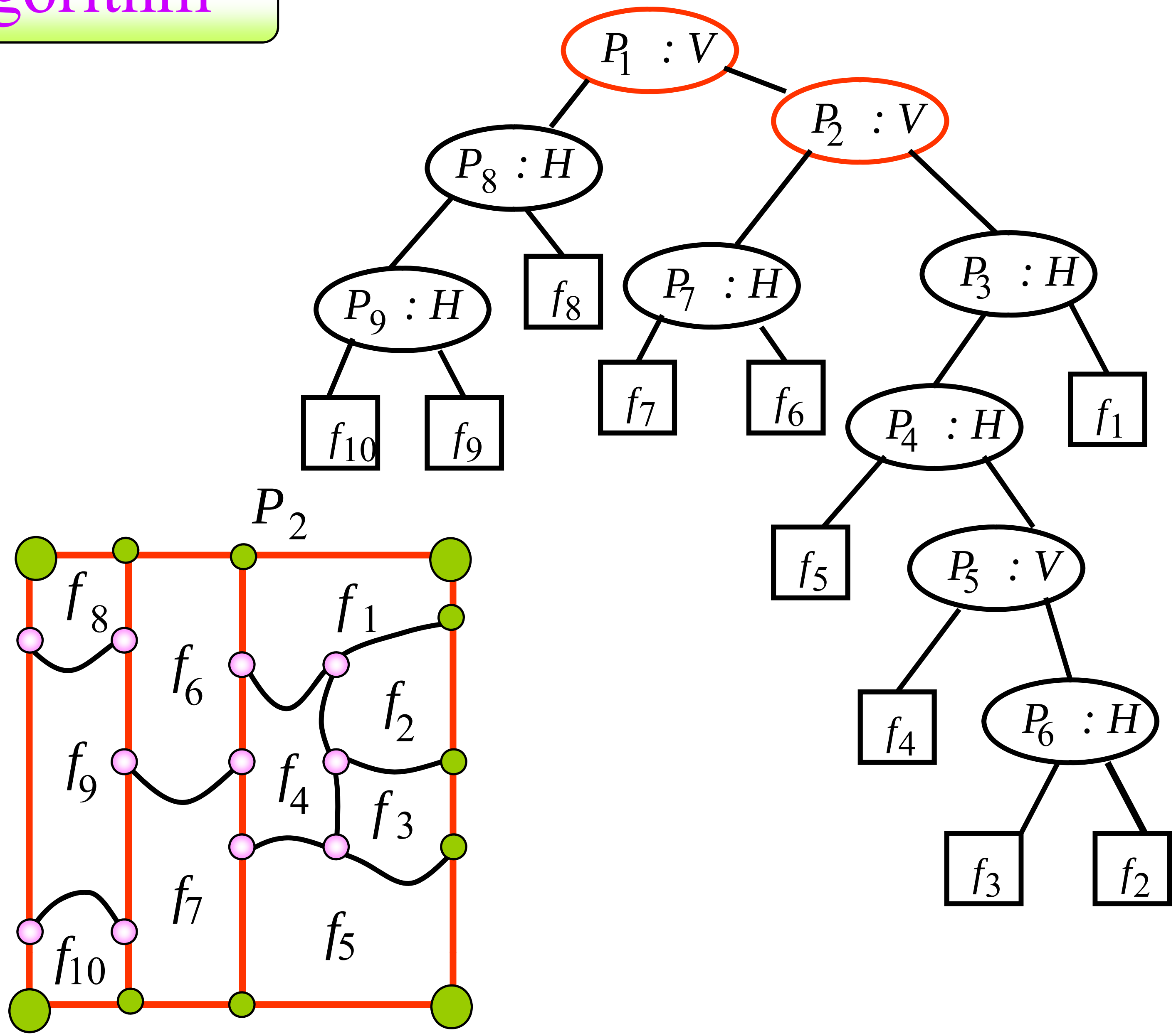


$P_1 : V$
$P_8 : H$
$P_2 : V$
$P_9 : H$
f_8
$P_7 : H$
$P_3 : H$
f_{10}
f_9
f_7
f_6
$P_4 : H$
f_1
f_5
$P_5 : V$
f_4
$P_6 : H$
f_3
f_2



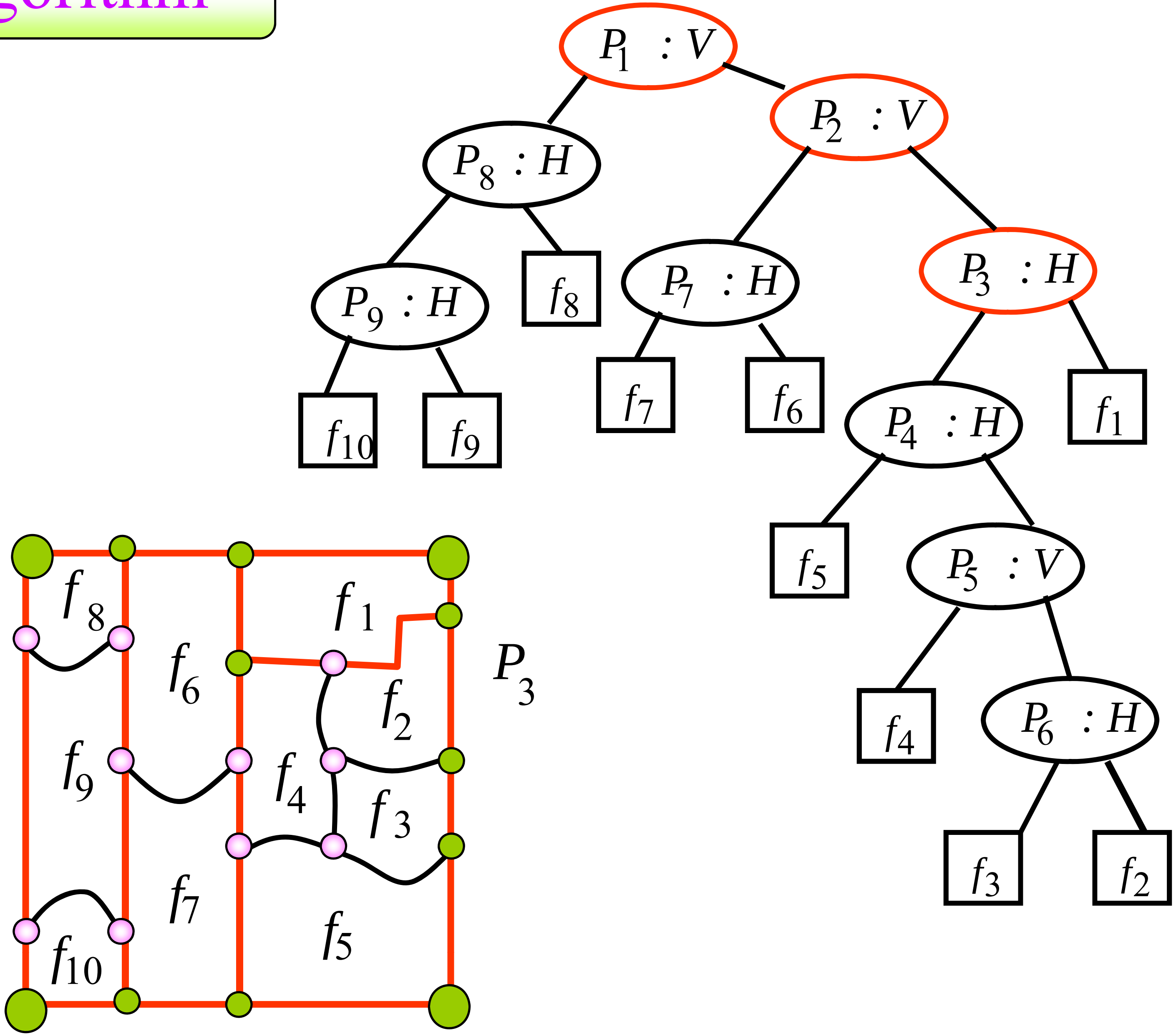
P_1
f_8
f_1
f_6
f_2
f_9
f_4
f_3
f_7
f_5
f_{10}

Algorithm



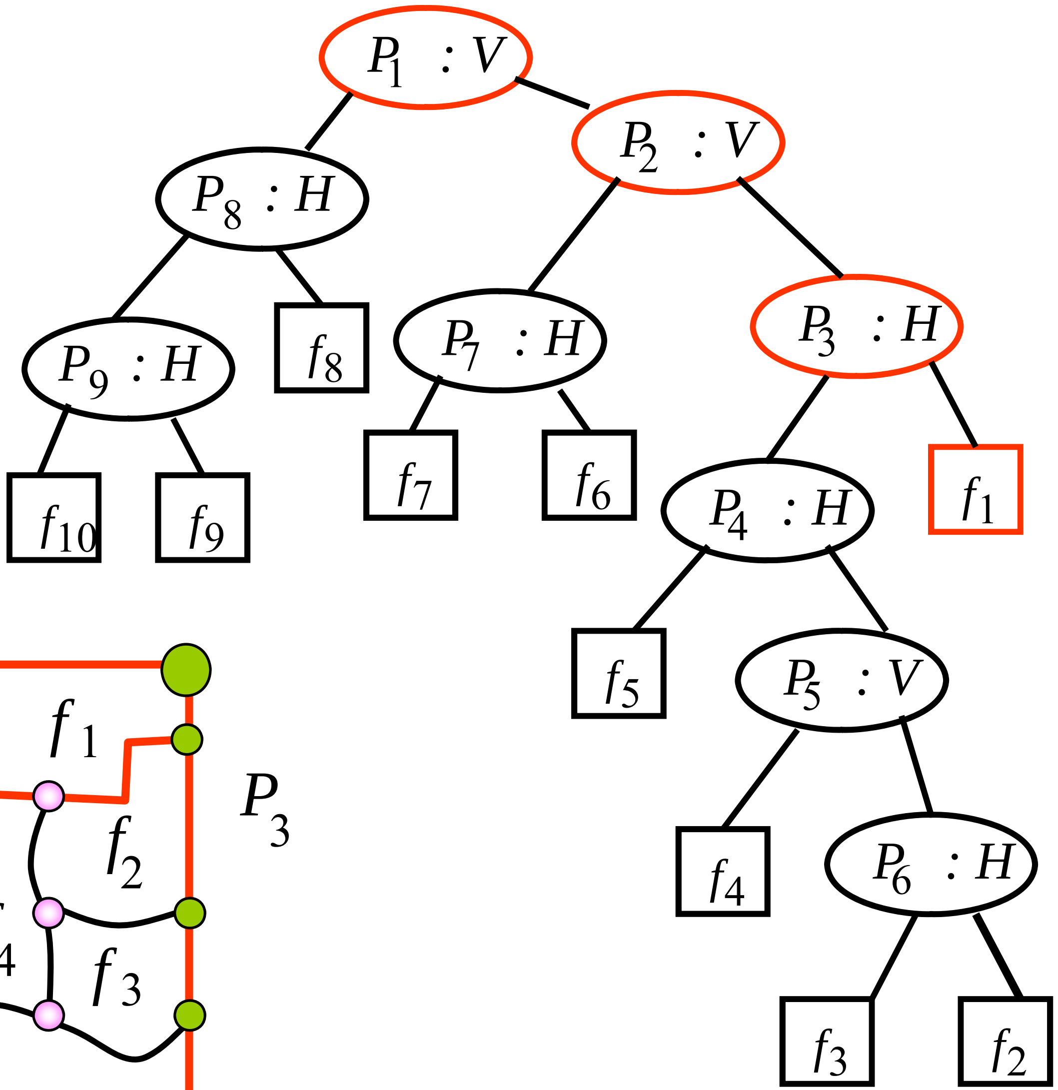
$P_1 : V$
$P_2 : V$
$P_8 : H$
$P_9 : H$
f_8
$P_7 : H$
$P_3 : H$
f_{10}
f_9
f_7
f_6
$P_4 : H$
f_1
f_5
$P_5 : V$
f_4
$P_6 : H$
f_3
f_2
P_2
f_8
f_1
f_6
f_2
f_9
f_4
f_3
f_7
f_{10}
f_5

Algorithm

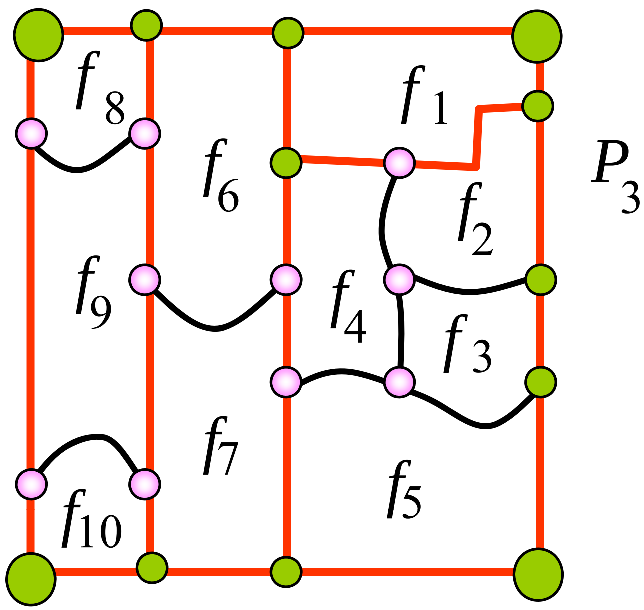


P1 : V
P2 : V
P8 : H
P9 : H
f8
P7 : H
P3 : H
f10
f9
f7
f6
P4 : H
f1
f5
P5 : V
f4
P6 : H
f3
f2
f8
f1
f6
f2
P3
f9
f4
f3
f7
f5
f10

Algorithm

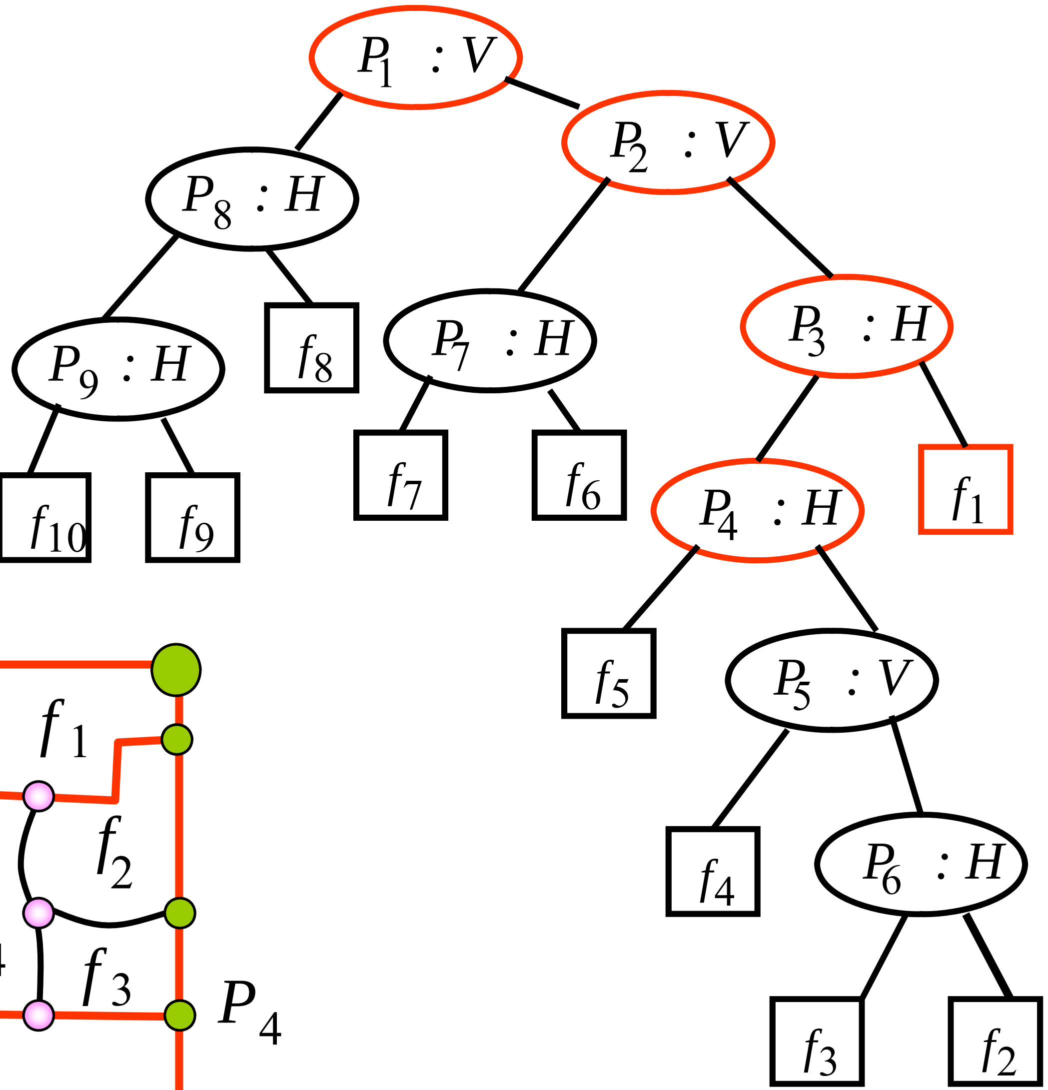


$P_1 : V$
$P_2 : V$
$P_8 : H$
$P_9 : H$
$P_7 : H$
$P_3 : H$
$P_4 : H$
$P_5 : V$
$P_6 : H$
f_1
f_2
f_3
f_4
f_5
f_6
f_7
f_8
f_9
f_{10}

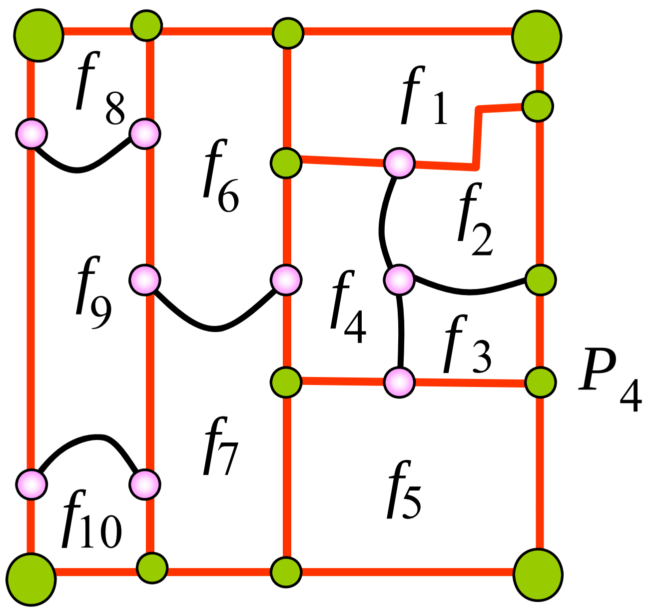


f_8
f_6
f_1
P_3
f_2
f_9
f_4
f_3
f_7
f_5
f_{10}

Algorithm

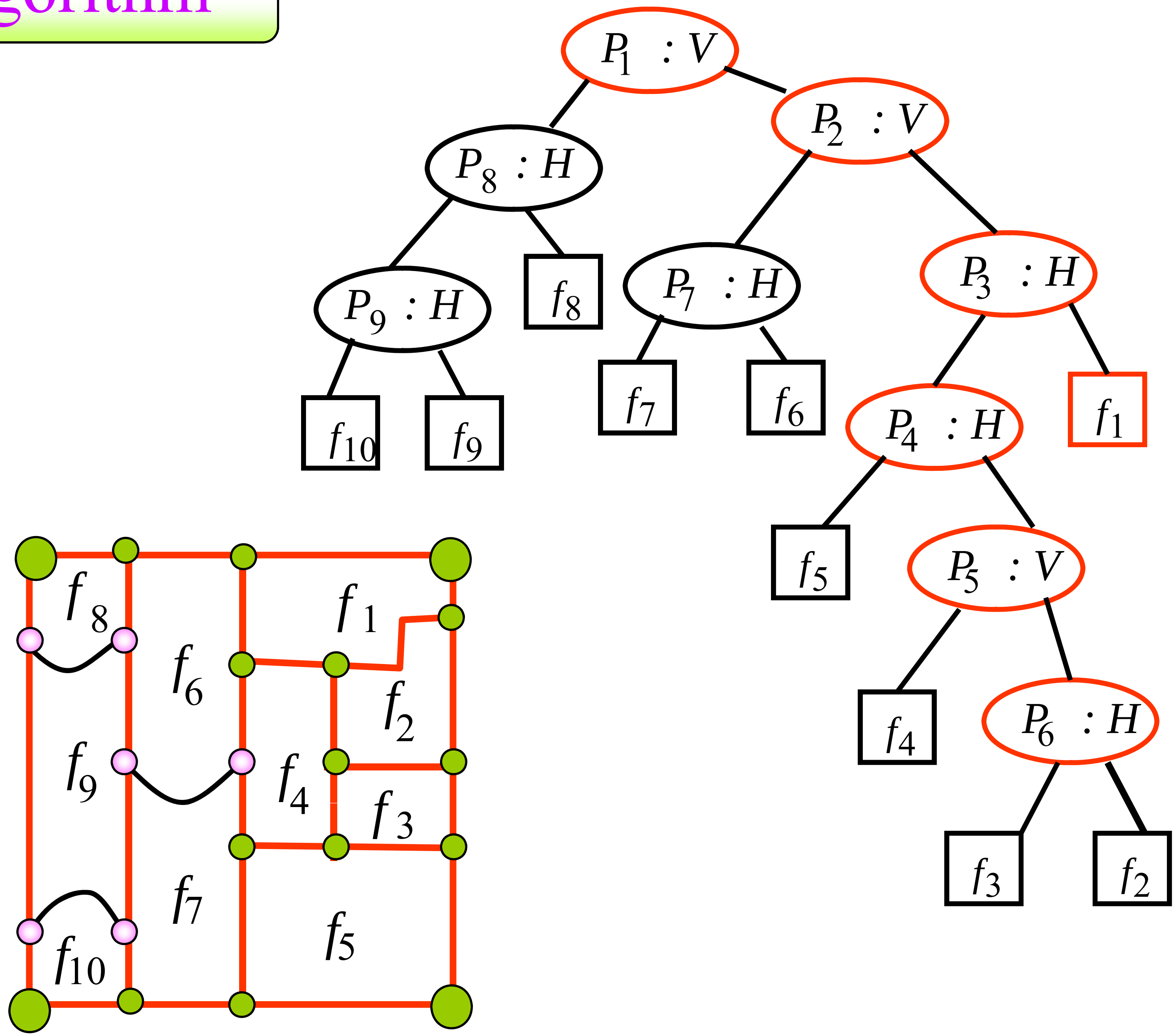


P1 : V
P2 : V
P8 : H
P7 : H
P3 : H
P9 : H
f8
f10
f9
f7
f6
P4 : H
f1
f5
P5 : V
f4
P6 : H
f3
f2



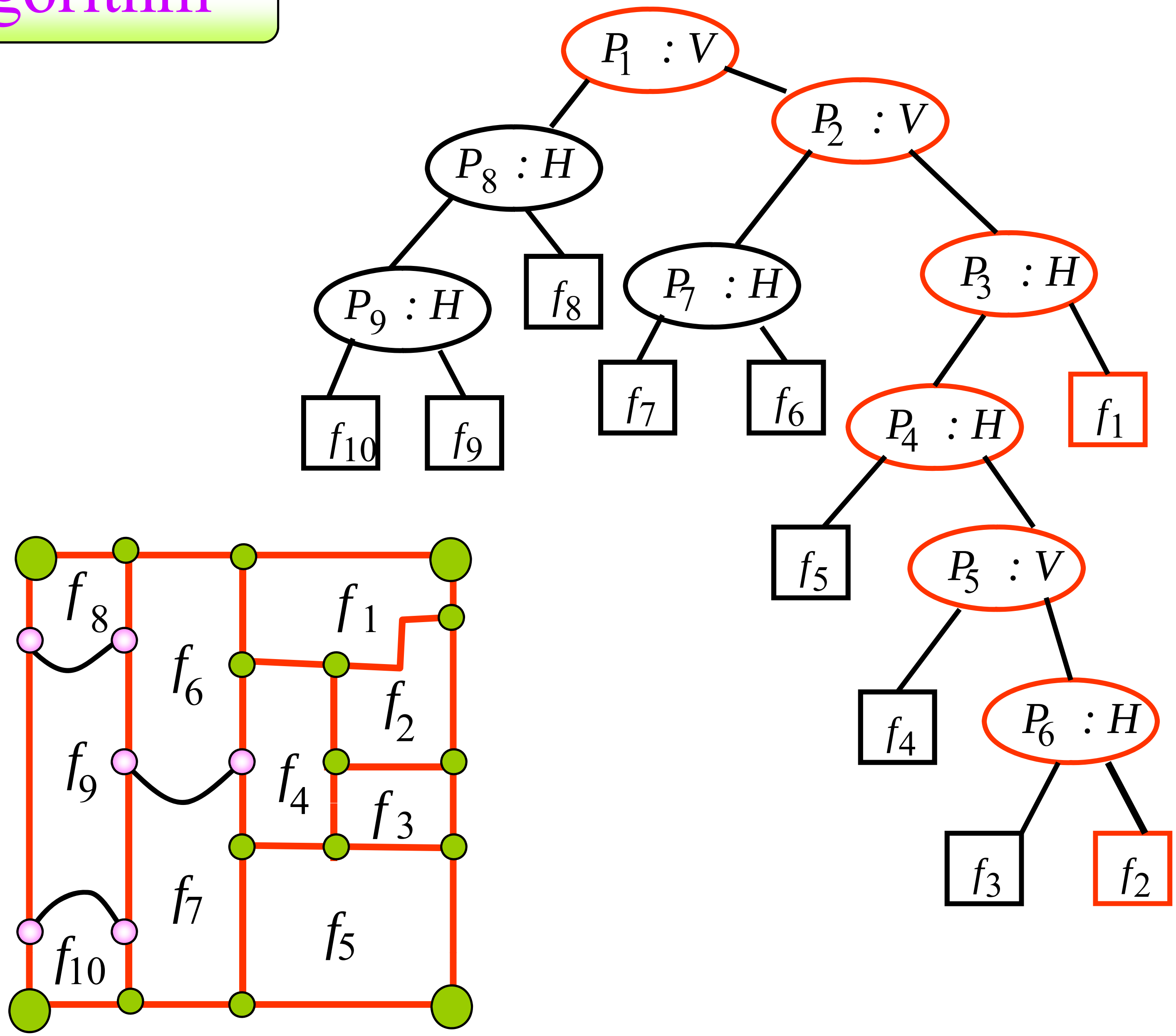
f8
f1
f6
f2
f9
f4
f3
P4
f7
f5
f10

Algorithm

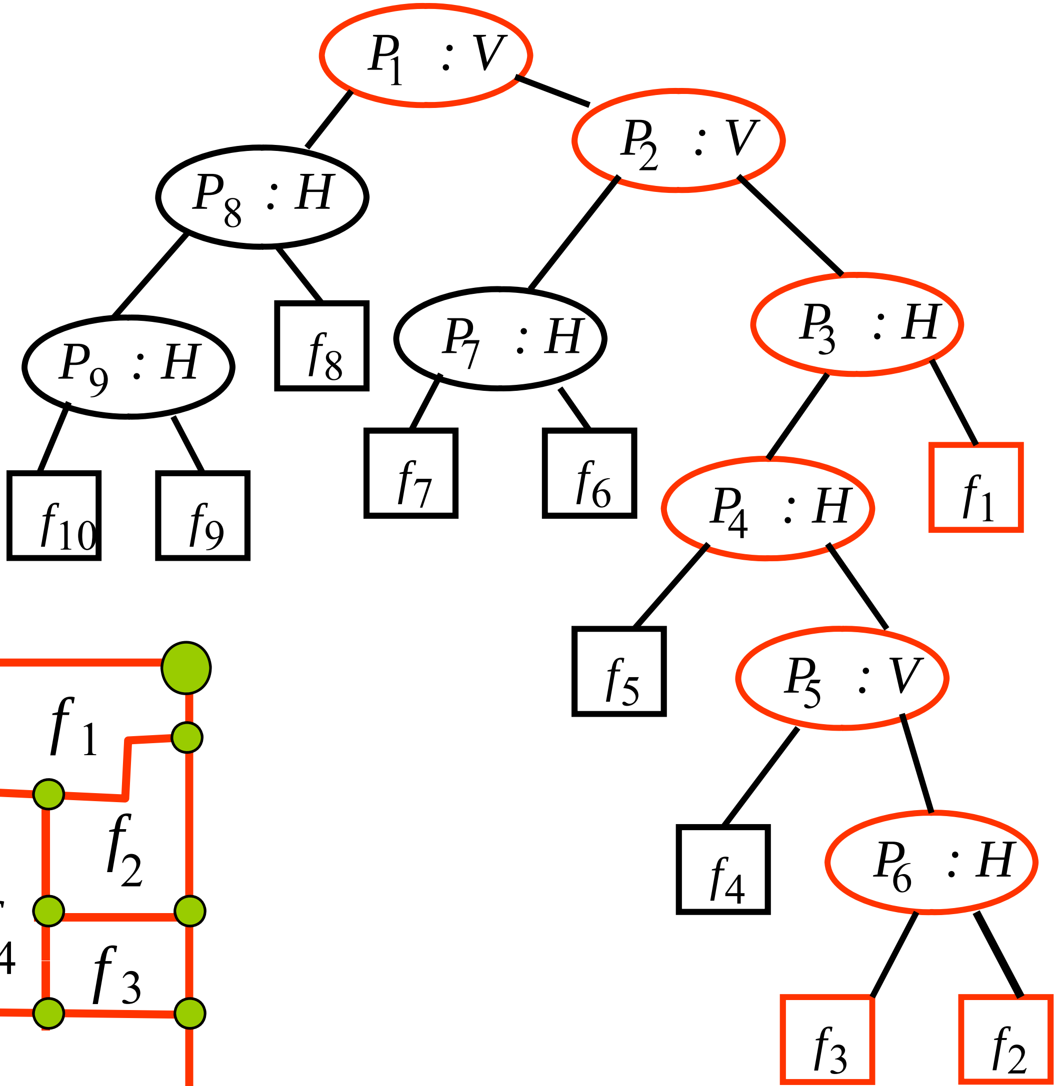


$P_1 : V$
$P_8 : H$
$P_2 : V$
$P_9 : H$
f_8
$P_7 : H$
$P_3 : H$
f_{10}
f_9
f_7
f_6
$P_4 : H$
f_1
f_5
$P_5 : V$
f_4
$P_6 : H$
f_3
f_2
f_8
f_1
f_6
f_2
f_9
f_4
f_3
f_7
f_5
f_{10}

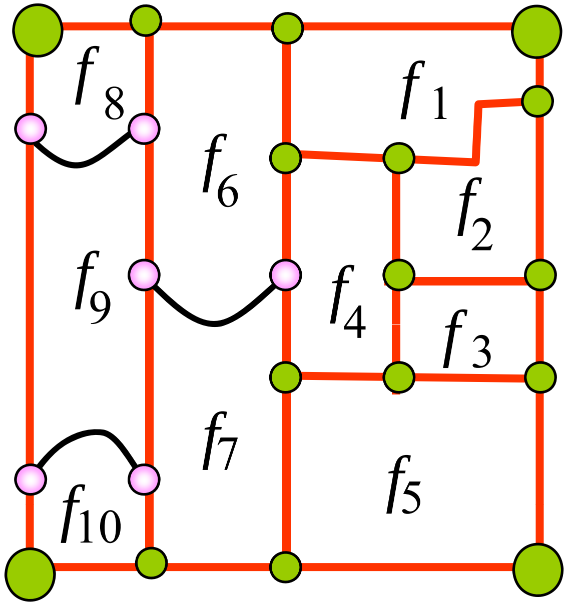
Algorithm



Algorithm

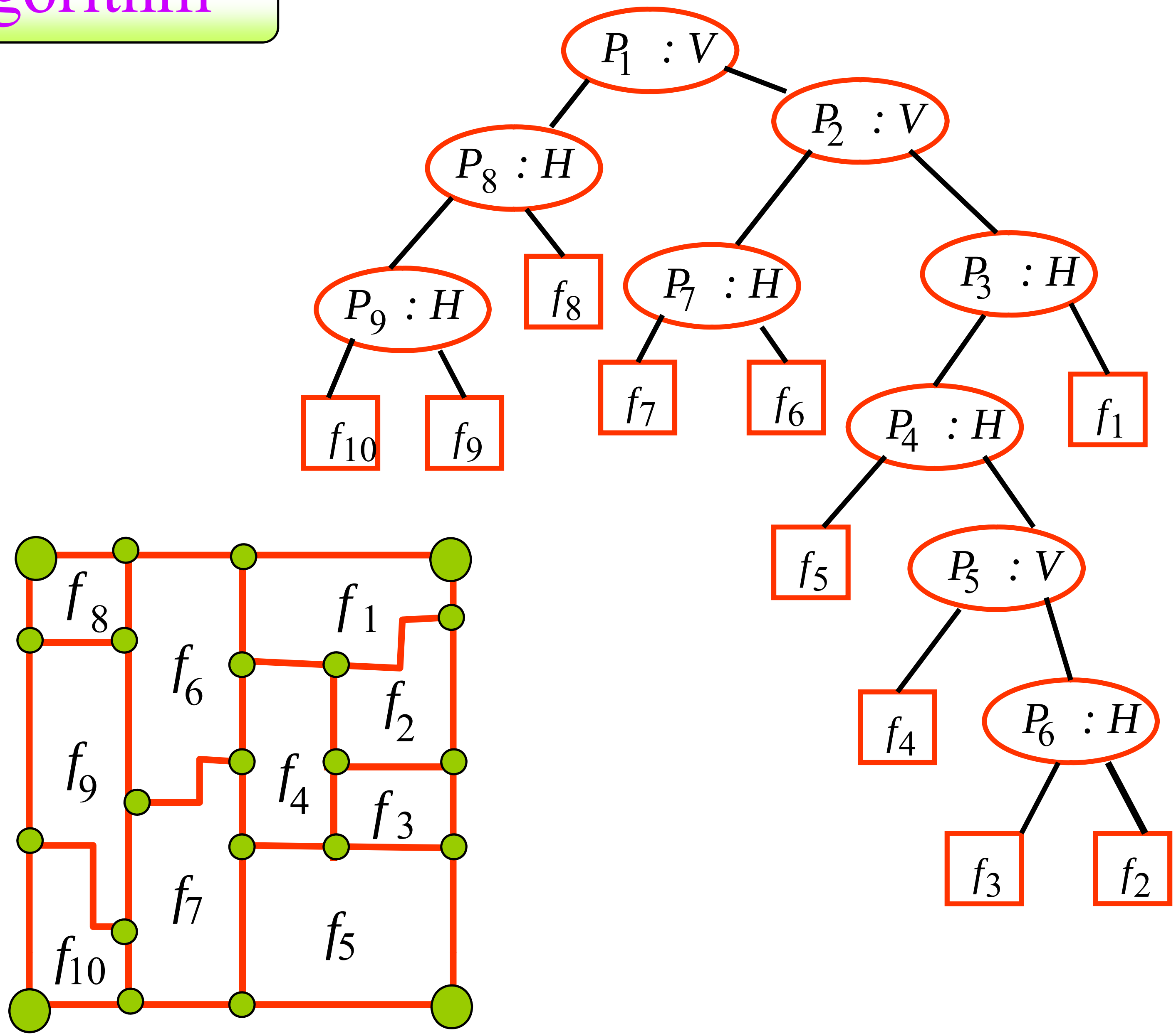


$P_1 : V$
$P_8 : H$
$P_2 : V$
$P_9 : H$
f_8
$P_7 : H$
$P_3 : H$
f_{10}
f_9
f_7
f_6
$P_4 : H$
f_1
f_5
$P_5 : V$
f_4
$P_6 : H$
f_3
f_2



f_8
f_1
f_6
f_2
f_9
f_4
f_3
f_7
f_5
f_{10}

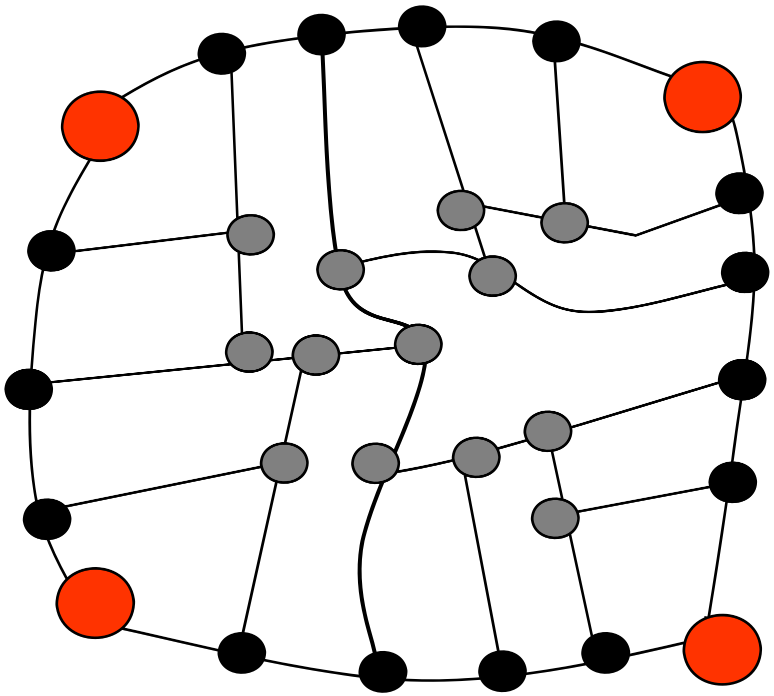
Algorithm



P_1 : V
P_8 : H
P_2 : V
P_9 : H
f_8
P_7 : H
P_3 : H
f_{10}
f_9
f_7
f_6
P_4 : H
f_1
f_5
P_5 : V
f_4
P_6 : H
f_3
f_2
f_8
f_1
f_6
f_2
f_9
f_4
f_3
f_7
f_5
f_{10}

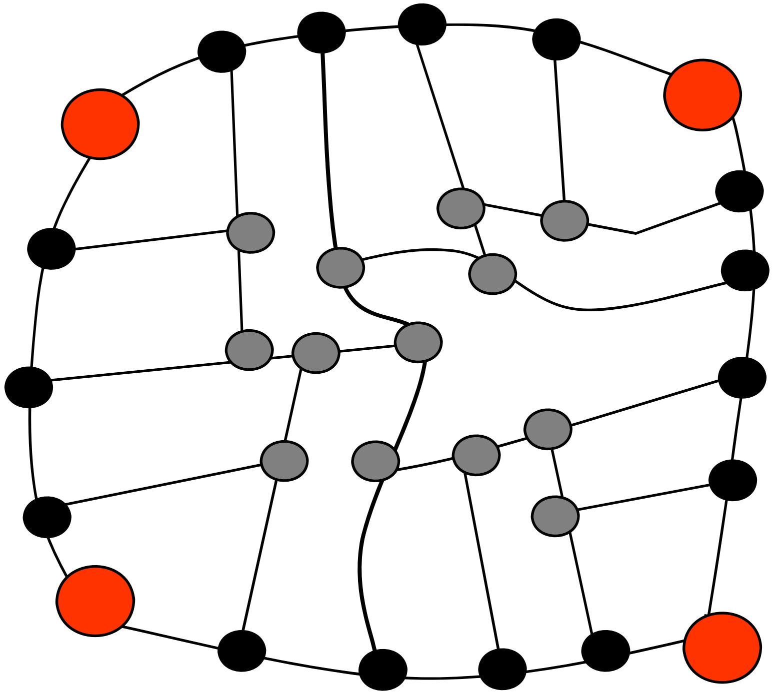
Algorithm

Initialization at root



Algorithm

Initialization at root

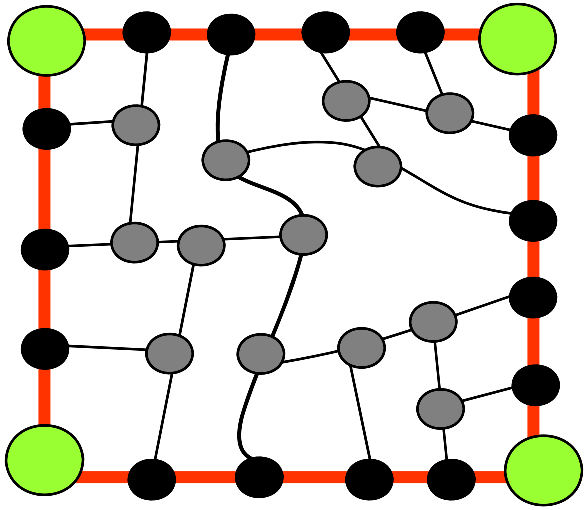


Draw the outer cycle as an arbitrary rectangle of area $A(G)$.

$A(G)$: sum of the prescribed areas of all inner faces in G.

Algorithm

Initialization at root

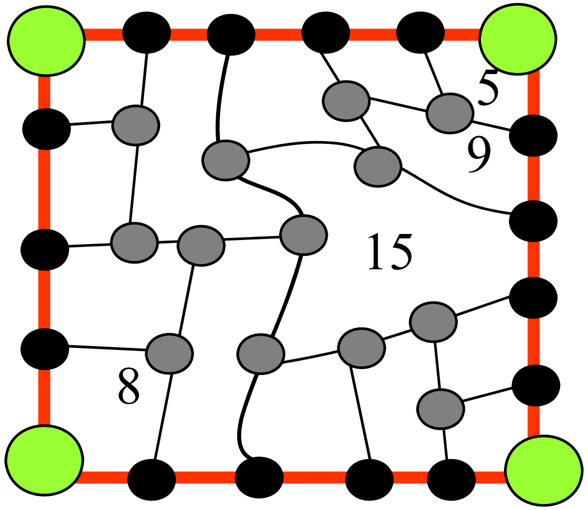


Draw the outer cycle as an arbitrary rectangle of area $A(G)$.

$A(G)$: sum of the prescribed areas of all inner faces in G.

Algorithm

Initialization at root



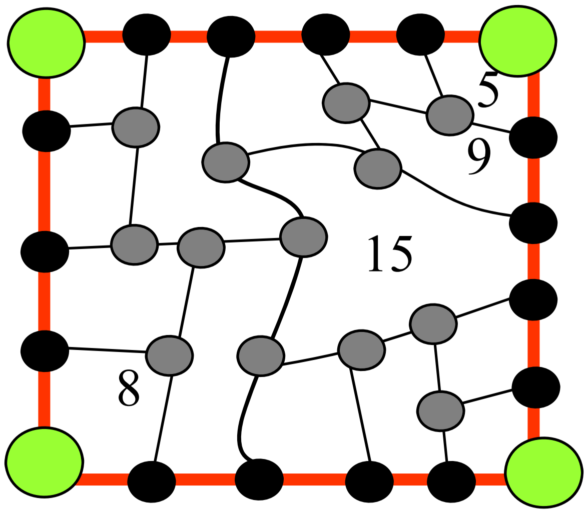
Draw the outer cycle as an arbitrary rectangle of area $A(G)$.

$A(G)$: sum of the prescribed areas of all inner faces in G.

$$A(G) = 5 + 9 + 15 + \cdots + 8 = 460$$

Algorithm

Initialization at root



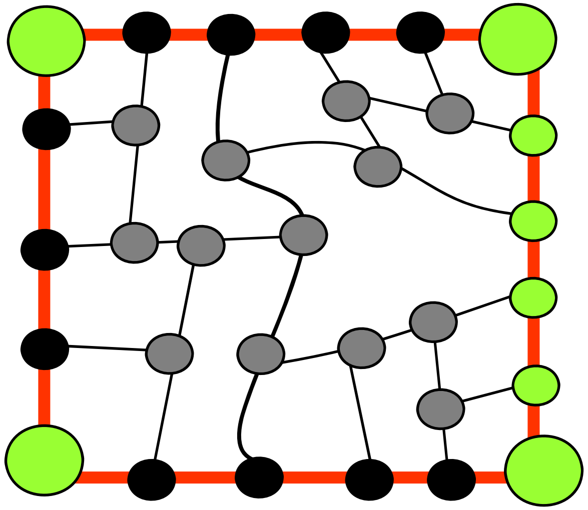
Draw the outer cycle as an arbitrary rectangle of area $A(G)$.

$A(G)$: sum of the prescribed areas of all inner faces in G.

$$A(G) = 5 + 9 + 15 + \cdots + 8 = 460 \quad = 23 \times 20$$

Algorithm

Initialization at root



Draw the outer cycle as an arbitrary rectangle of area $A(G)$.

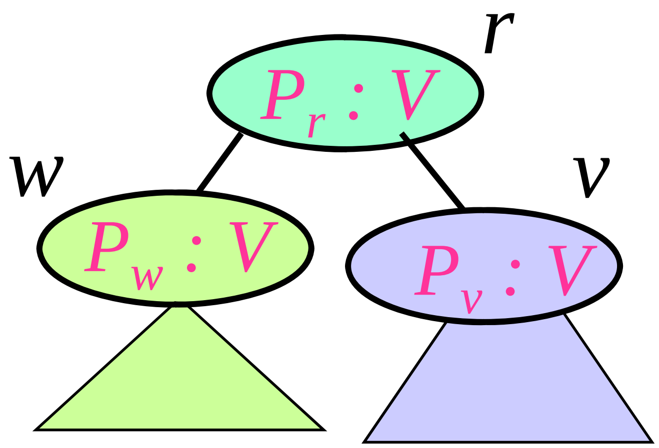
$A(G)$: sum of the prescribed areas of all inner faces in G.

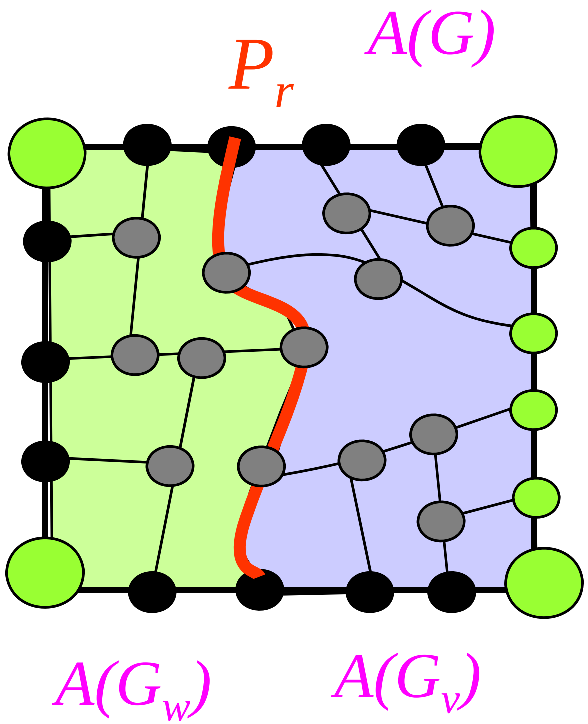
Fix arbitrarily the position of vertices on the right side of the rectangle preserving their relative positions.

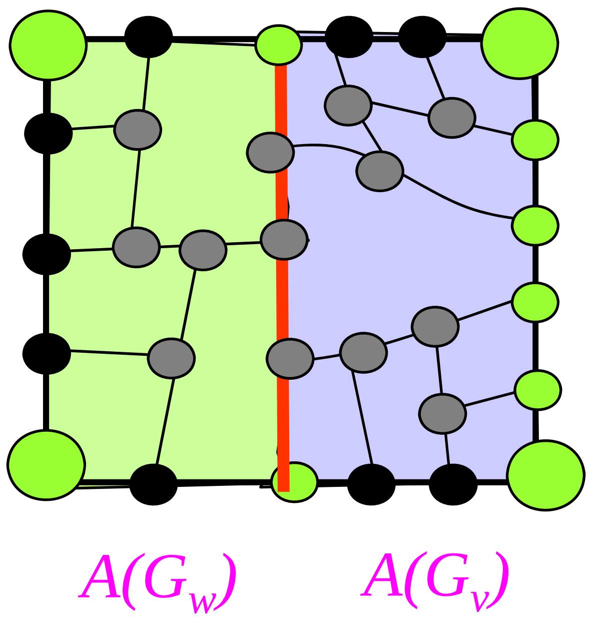
Algorithm

Operation at root

Root : vertical slice



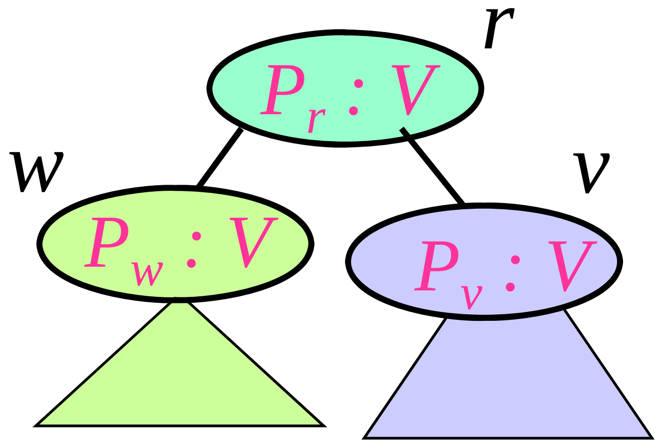


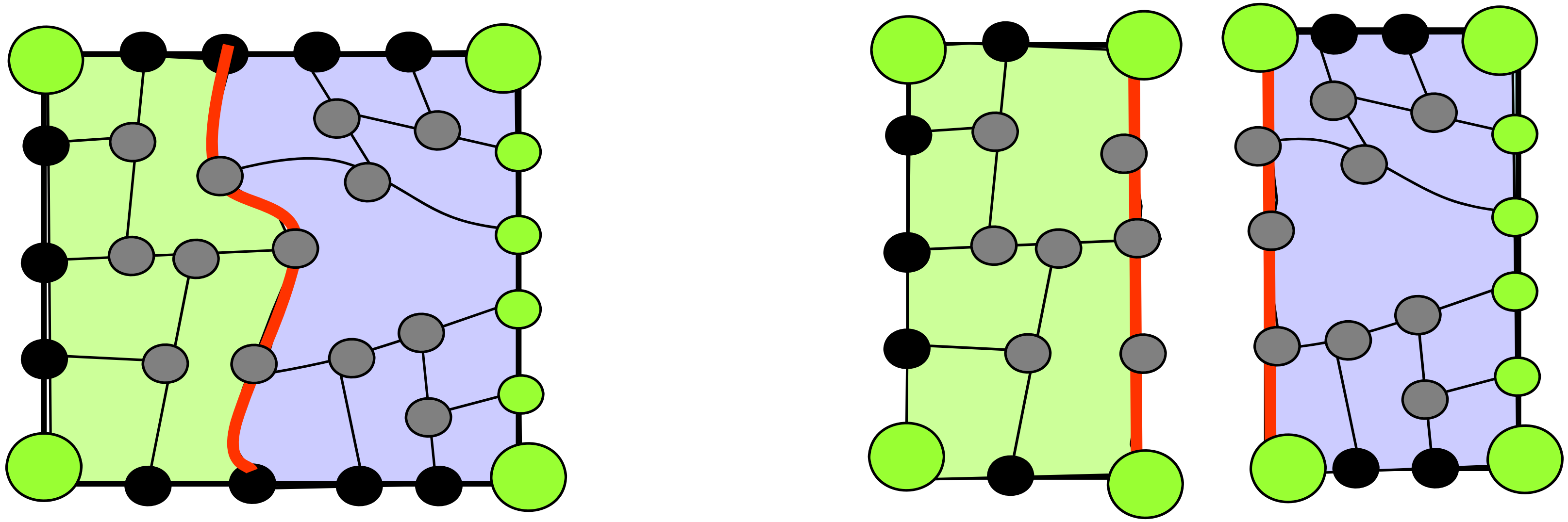


Algorithm

Operation at root

Root : vertical slice

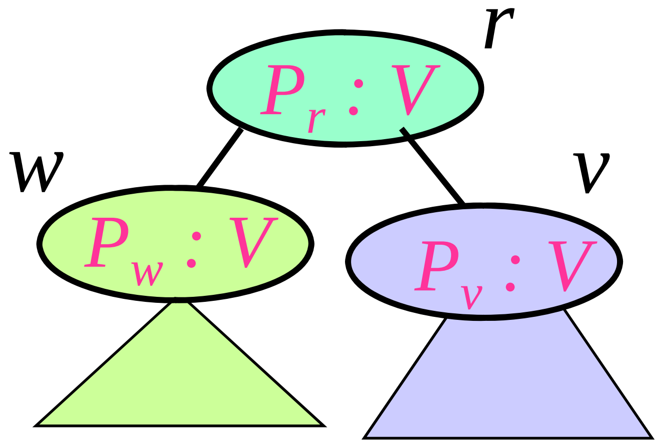


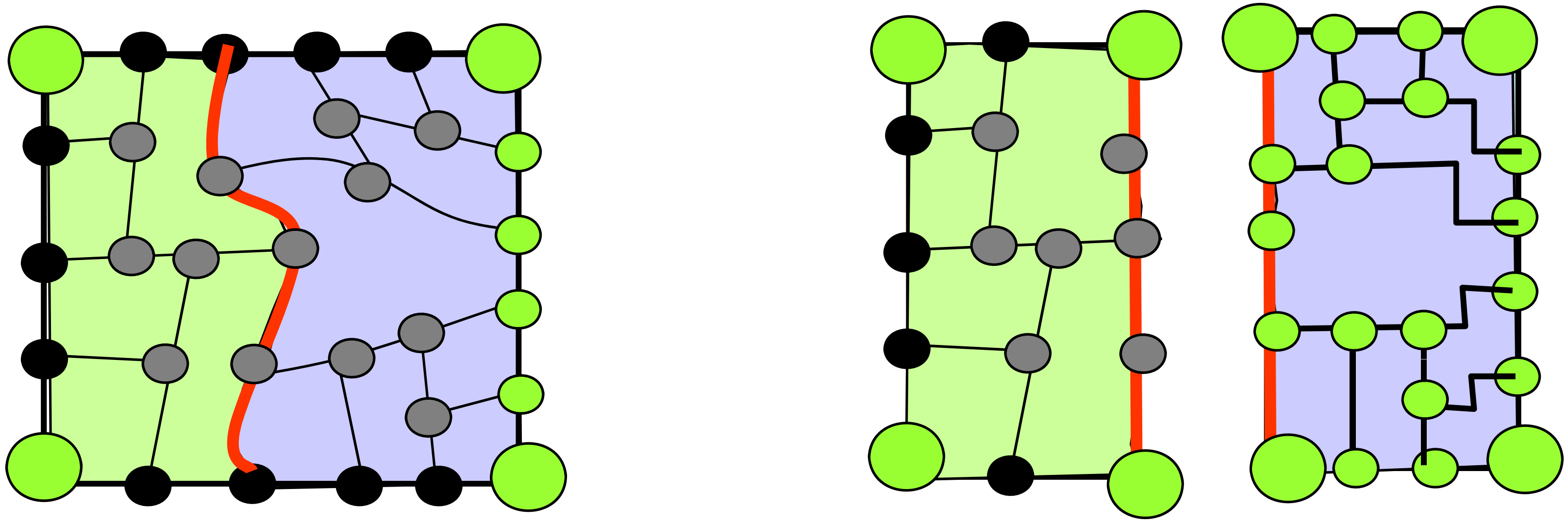


Algorithm

Operation at root

Root : vertical slice

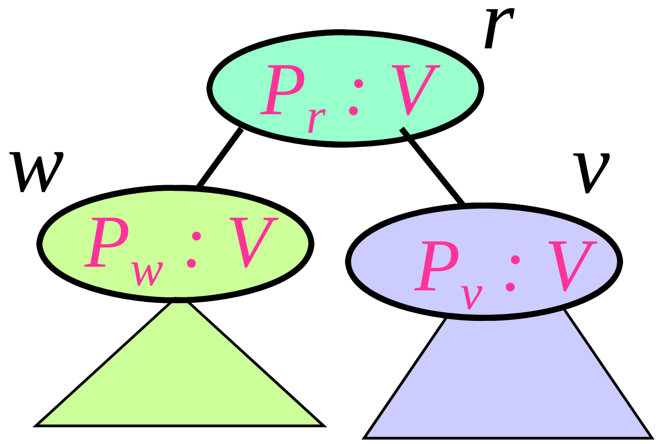


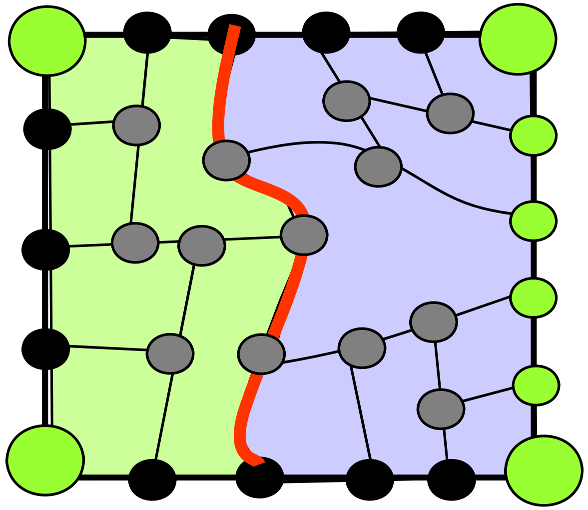


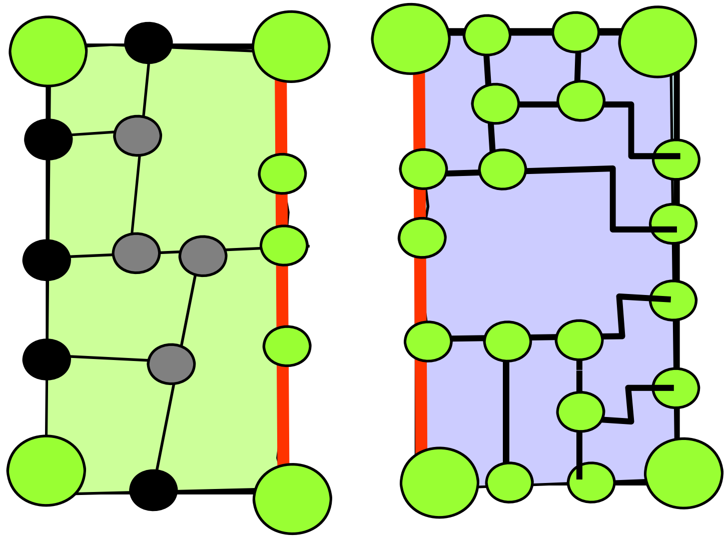
Algorithm

Operation at root

Root : vertical slice



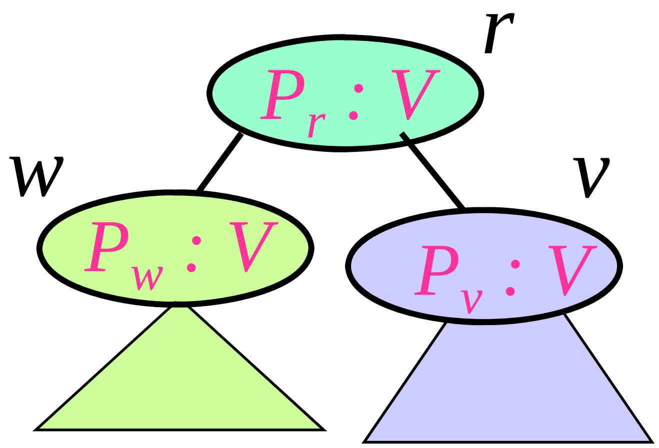


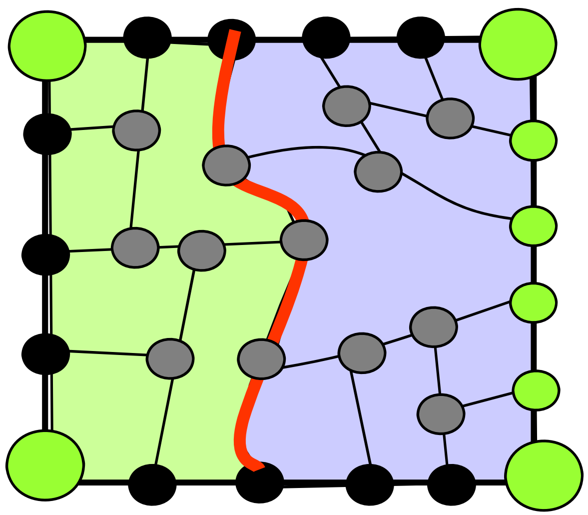


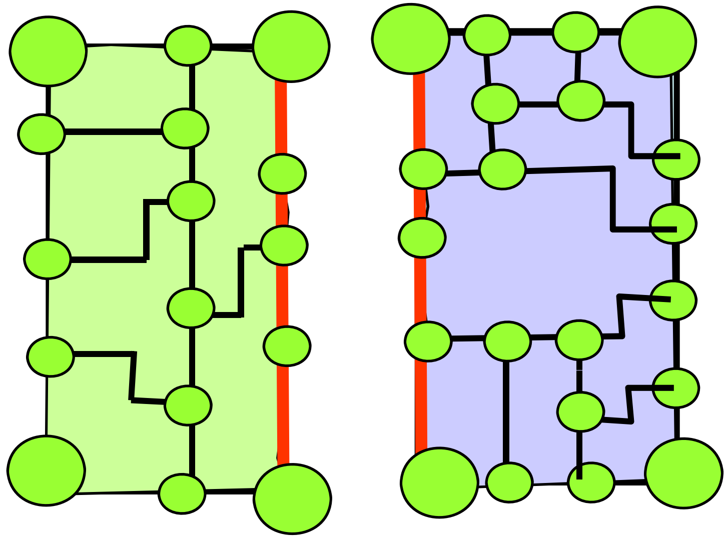
Algorithm

Operation at root

Root : vertical slice



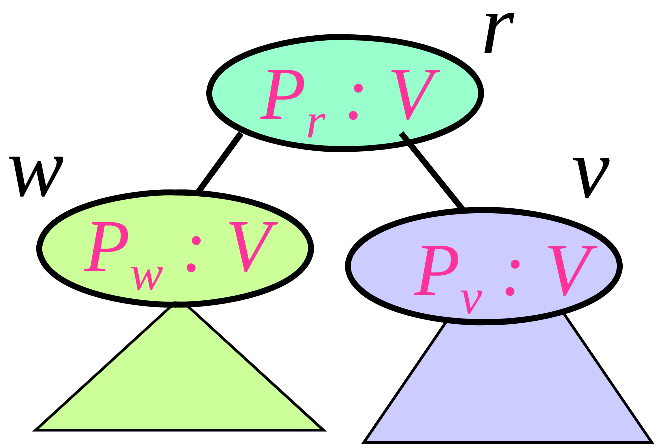


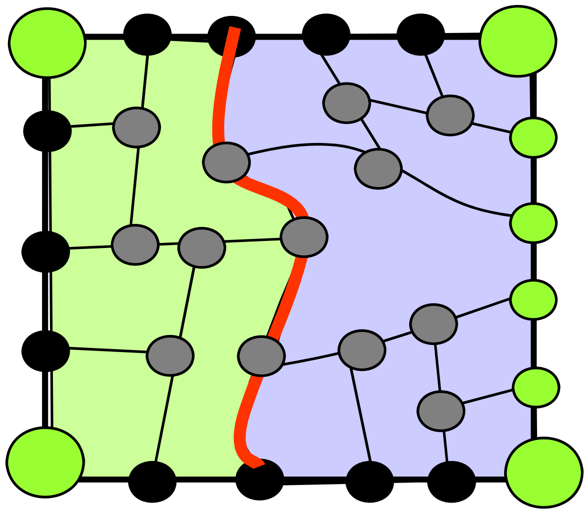


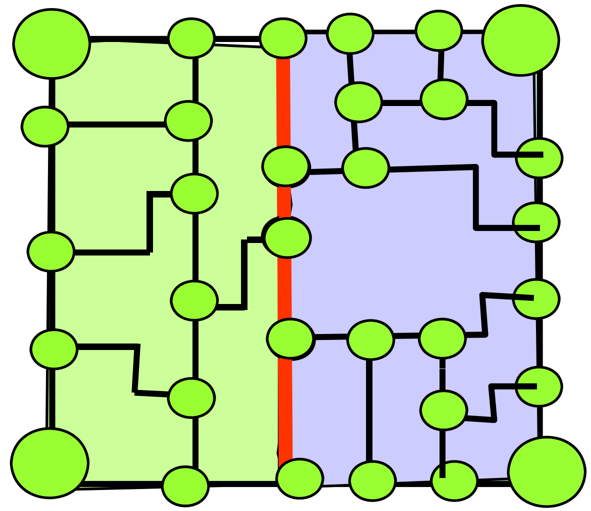
Algorithm

Operation at root

Root : vertical slice



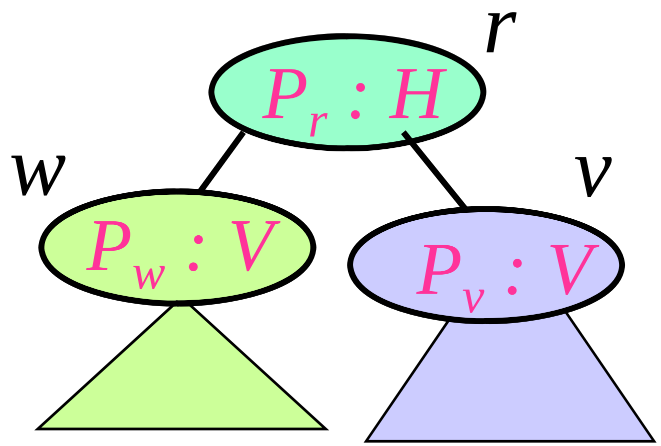


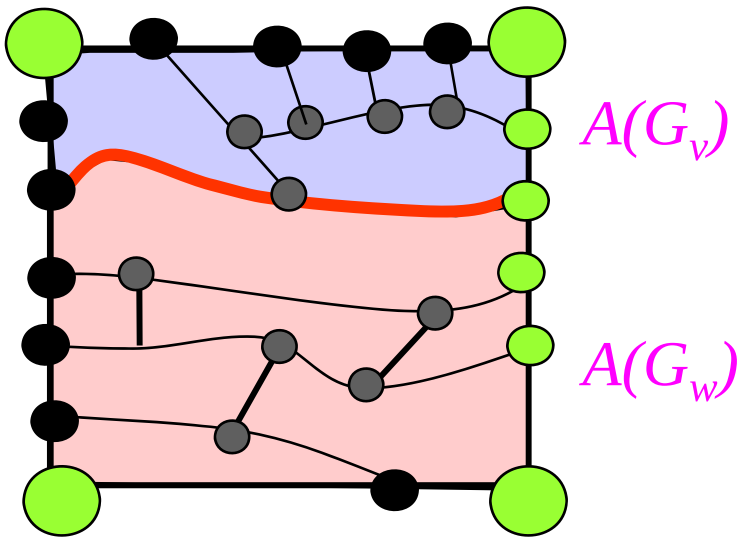


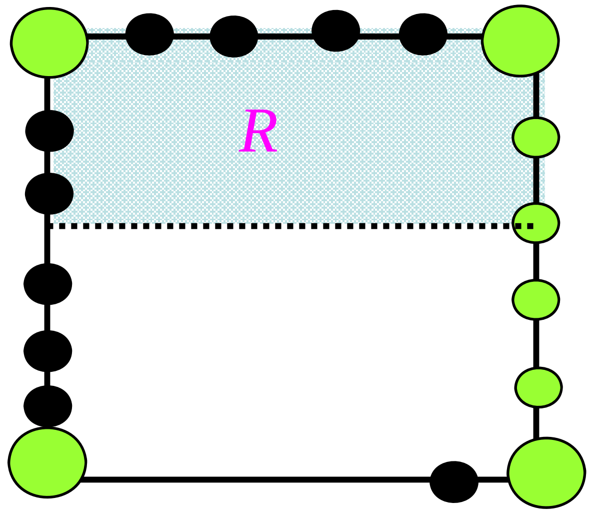
Algorithm

Operation at root

Root : horizontal slice



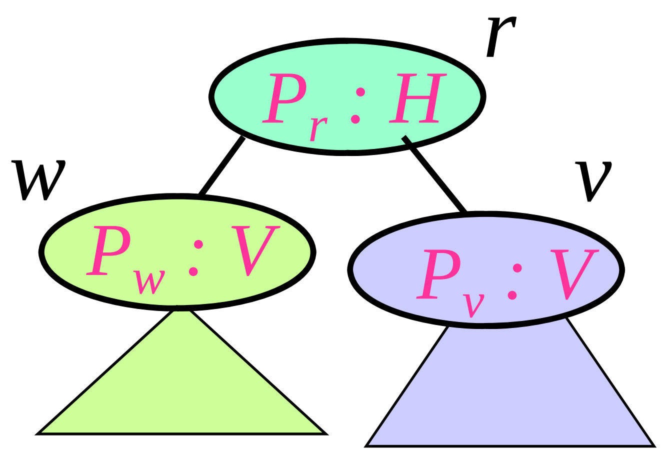


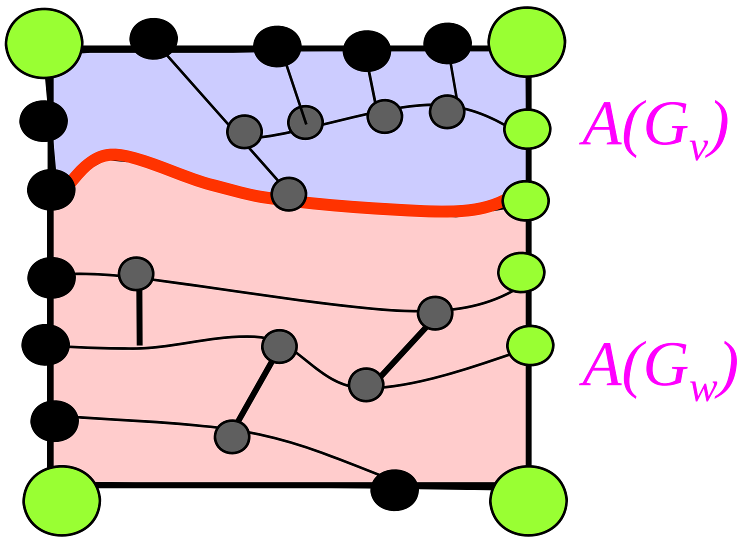


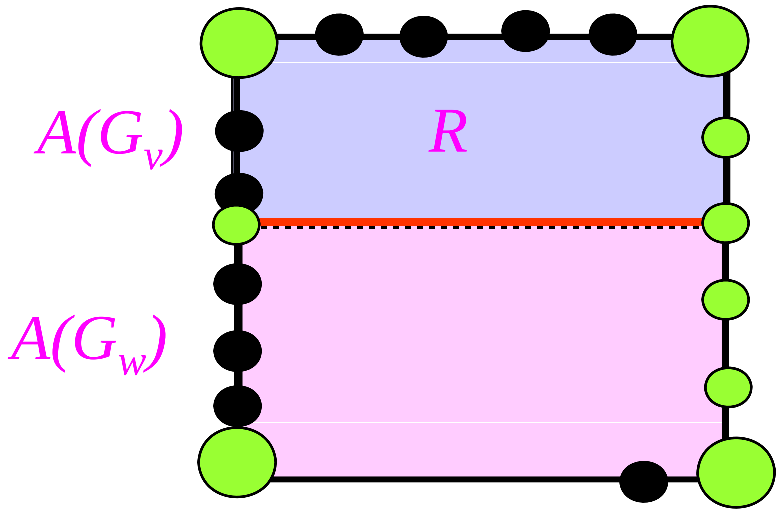
Algorithm

Operation at root

Root : horizontal slice





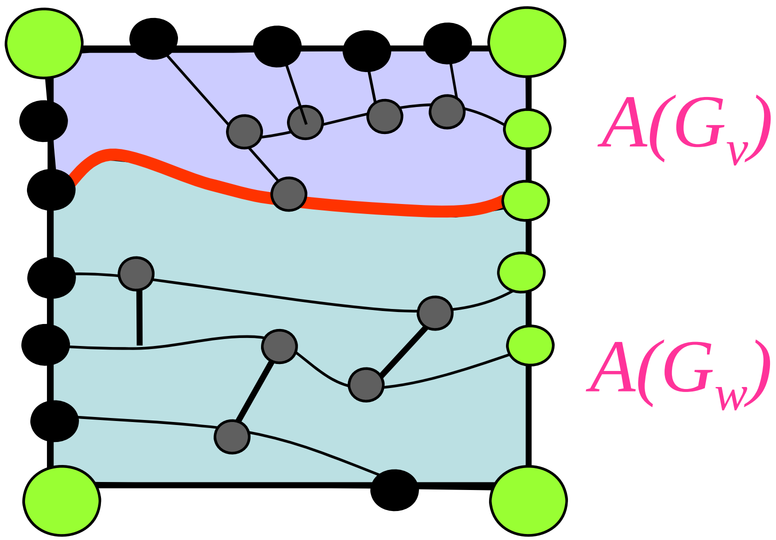


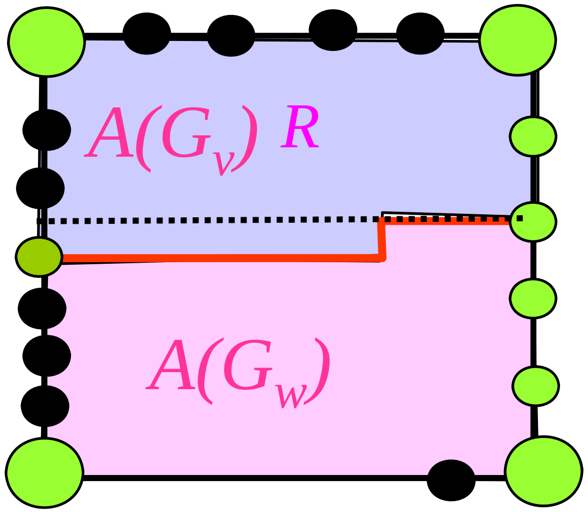
Case 1: $A(G_v) = A(R)$

Algorithm

Operation at root

Root : horizontal slice



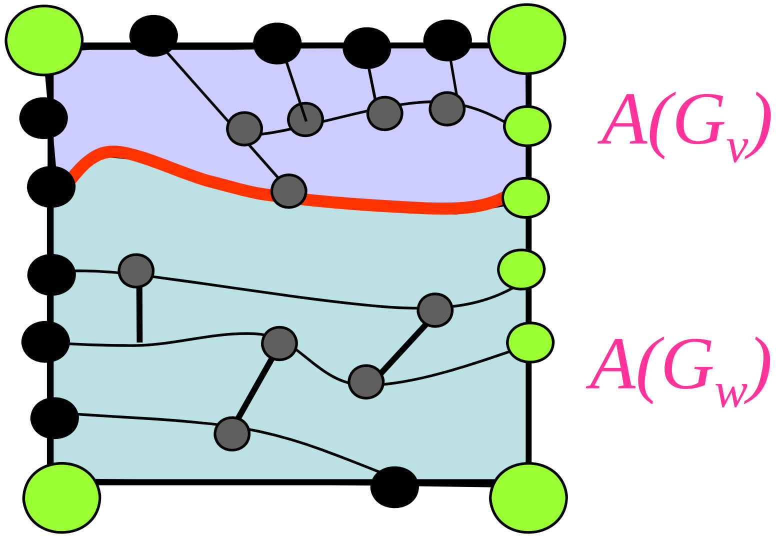


Case 2: $A(G_v) > A(R)$

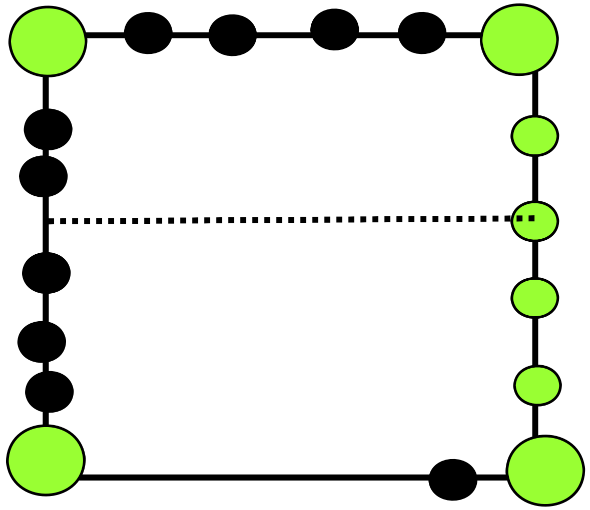
Algorithm

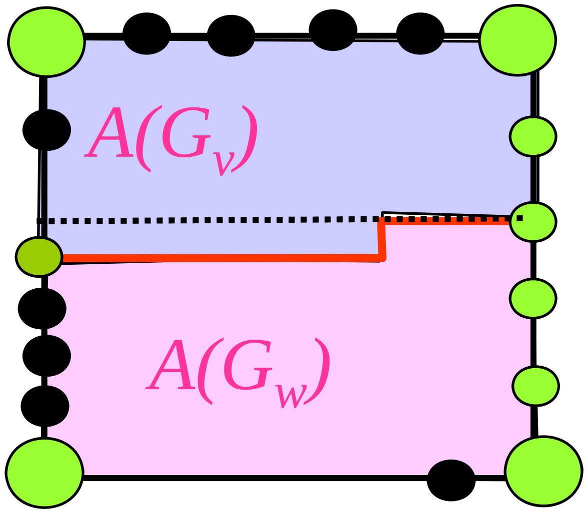
Operation at root

Root : horizontal slice



Case 3: $A(G_v) < A(R)$





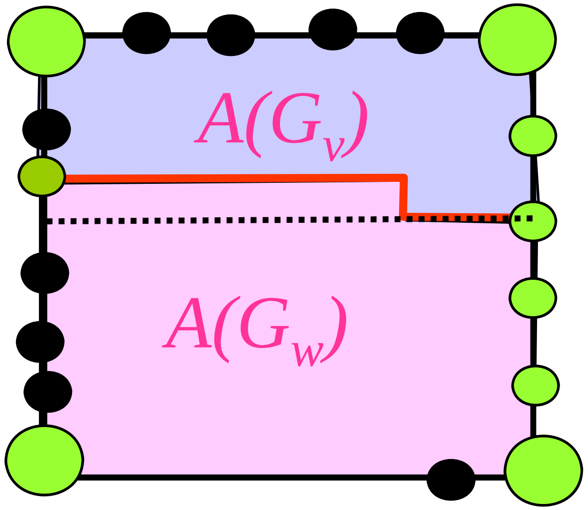
Case 2: $A(G_v) > A(R)$

Algorithm

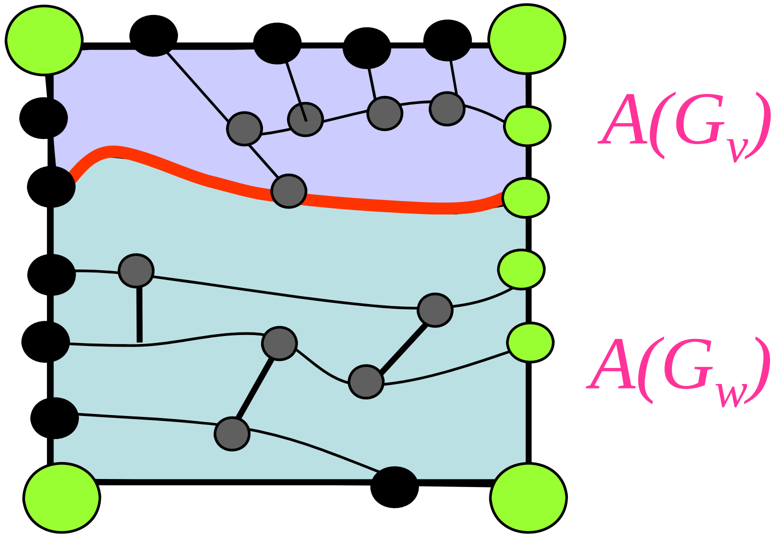
Operation at root

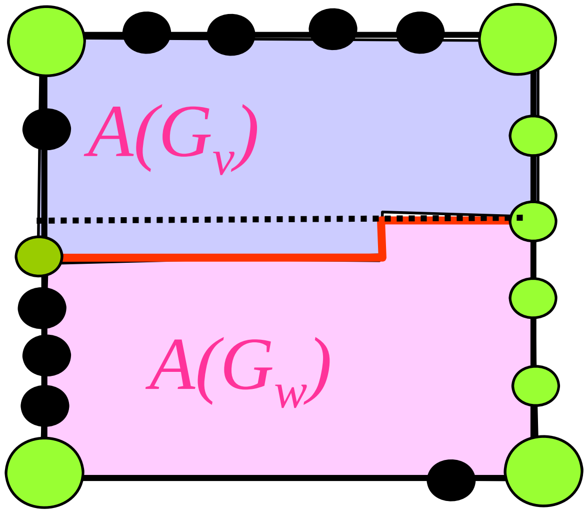
Root : horizontal slice

$Case\ 3:\ \ A(G_v) < A(R)$

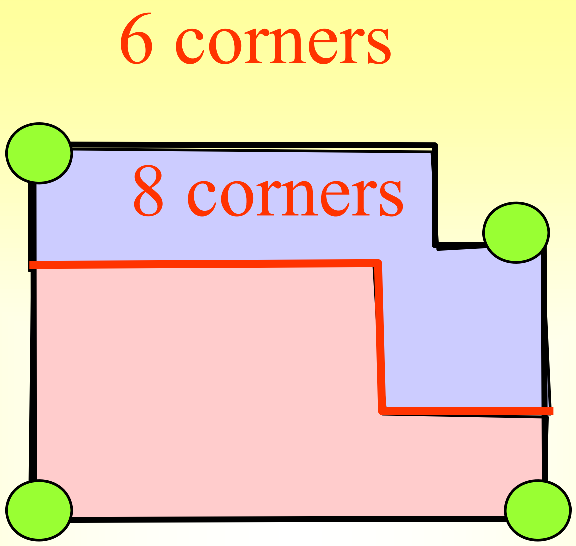


6 corners





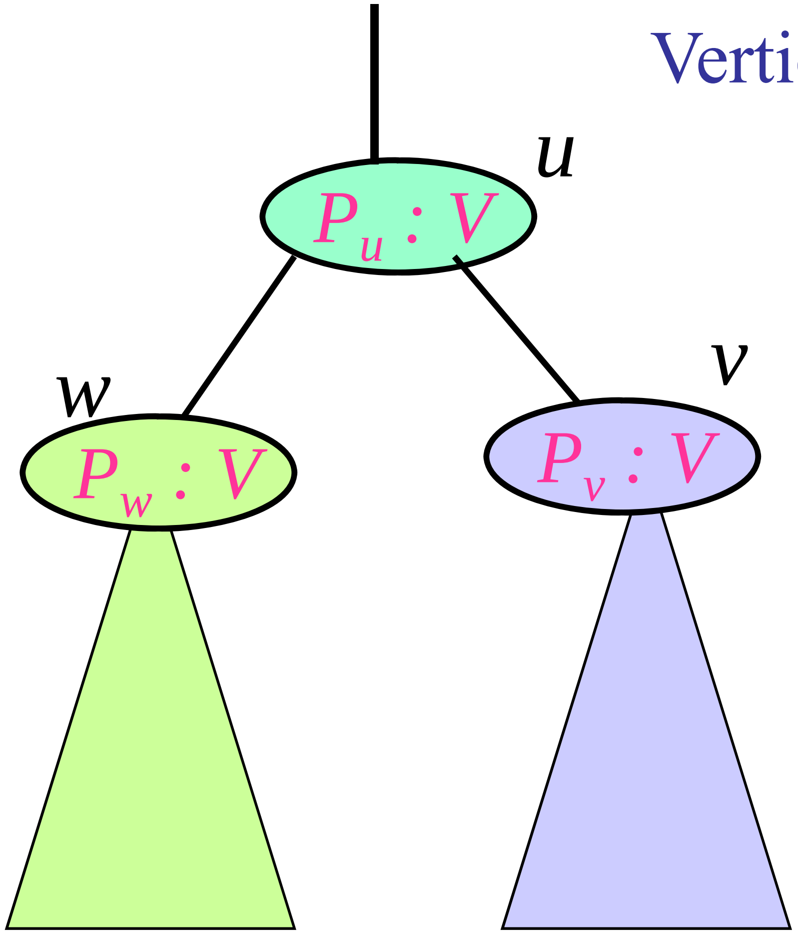
$Case\ 2:\ \ A(G_v) > A(R)$

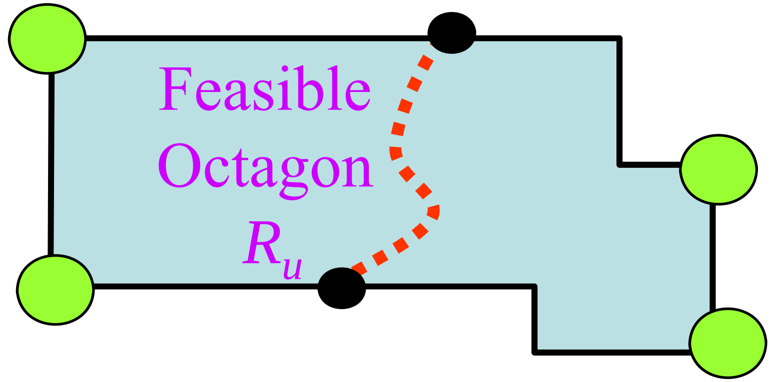


Polygon of exactly 8 corners may appear when a horizontal slice is embedded inside a polygon of 6 corners.

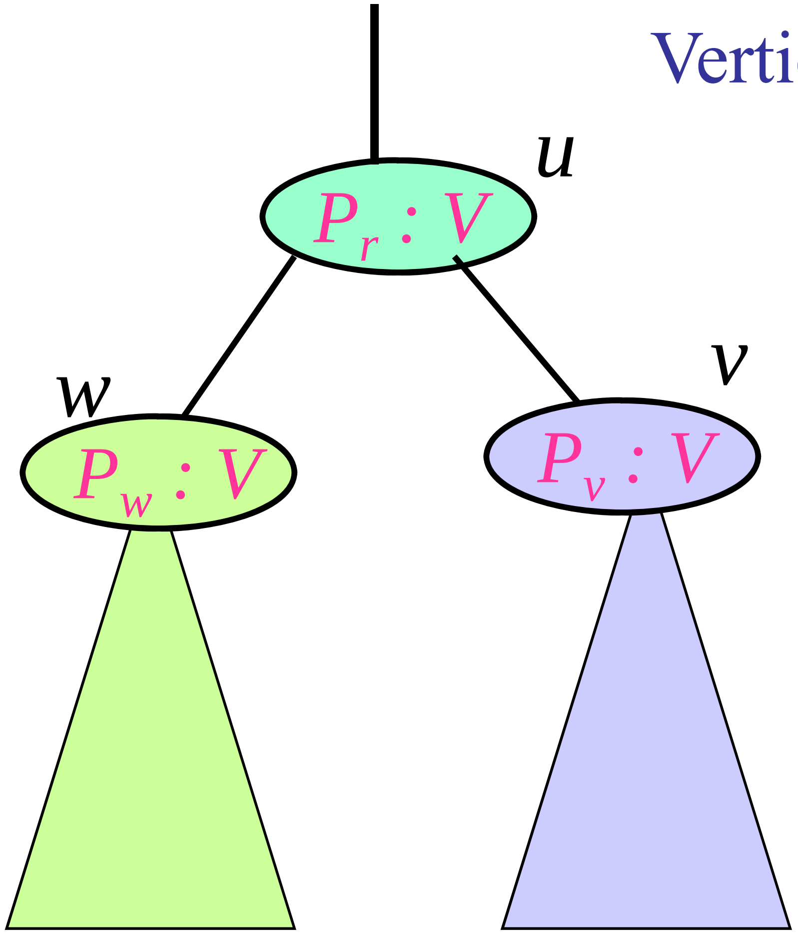
General computation at an internal node

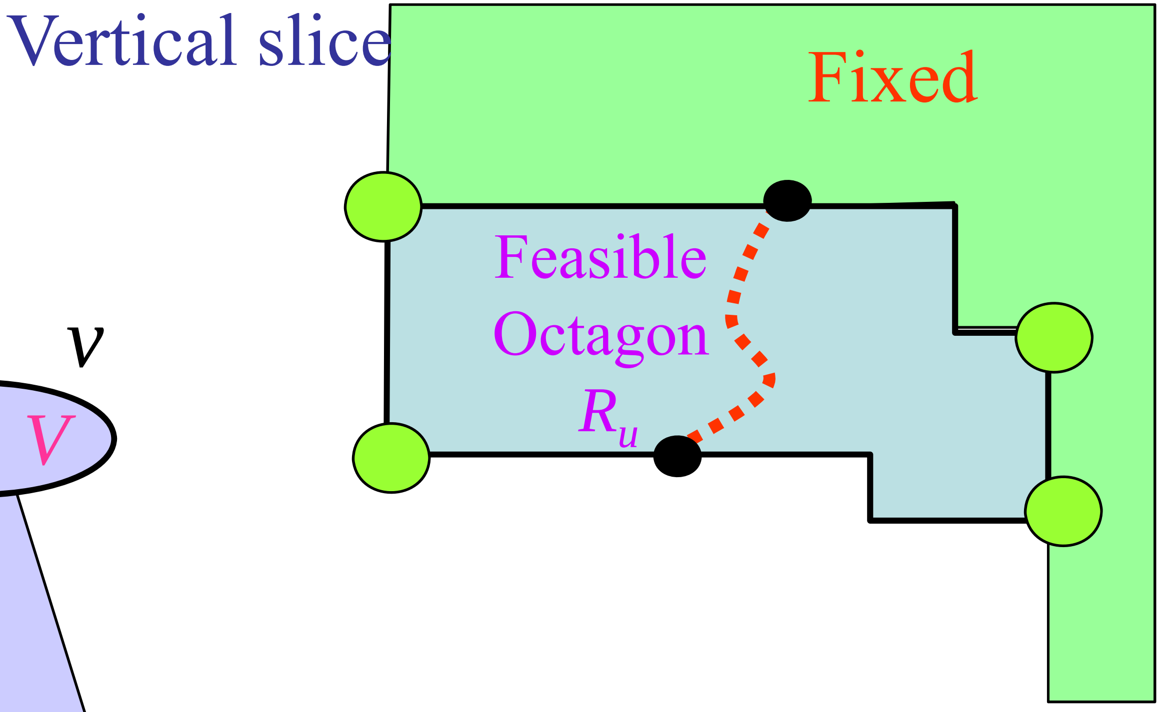
Vertical slice



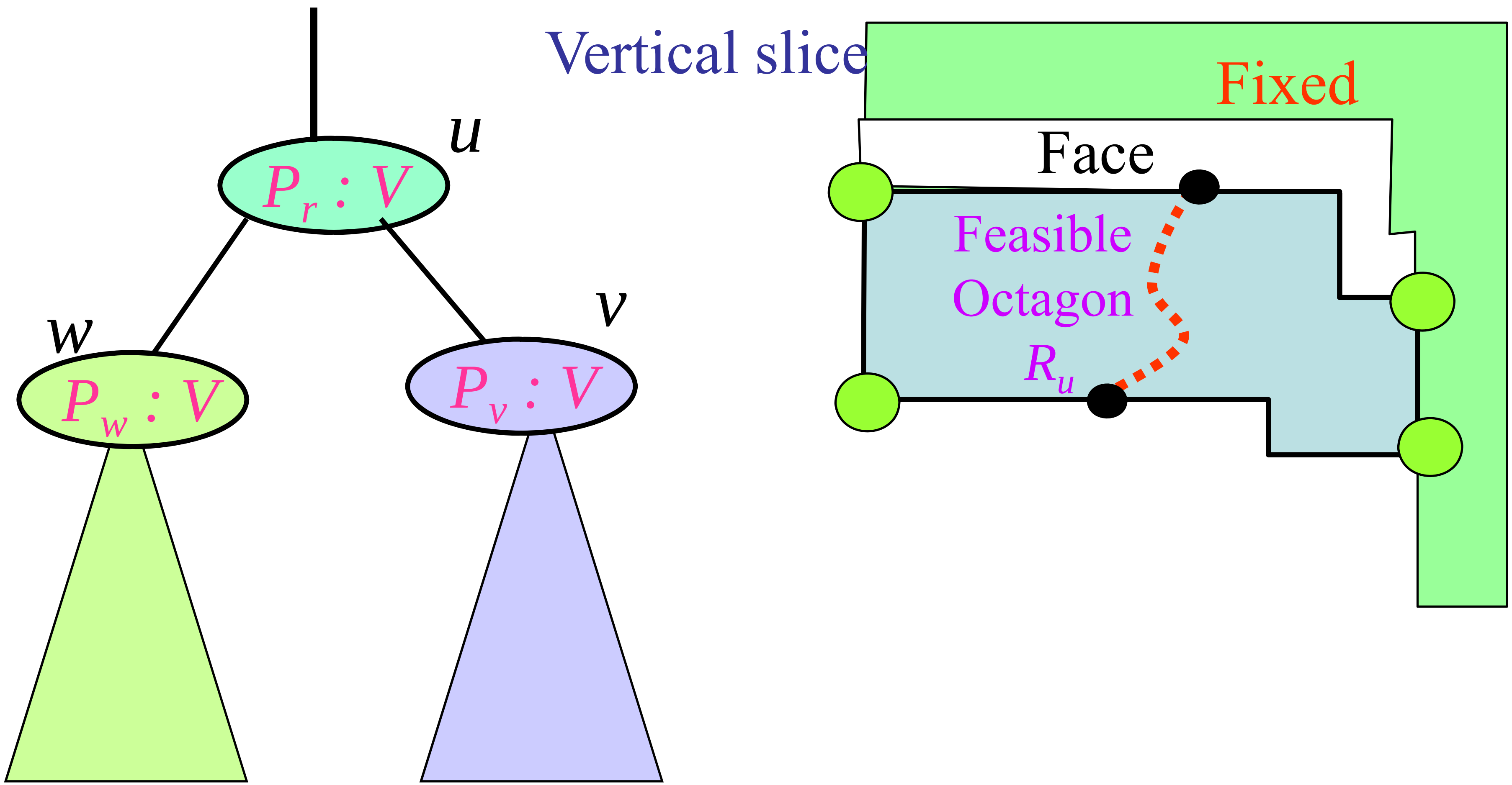


General computation at an internal node



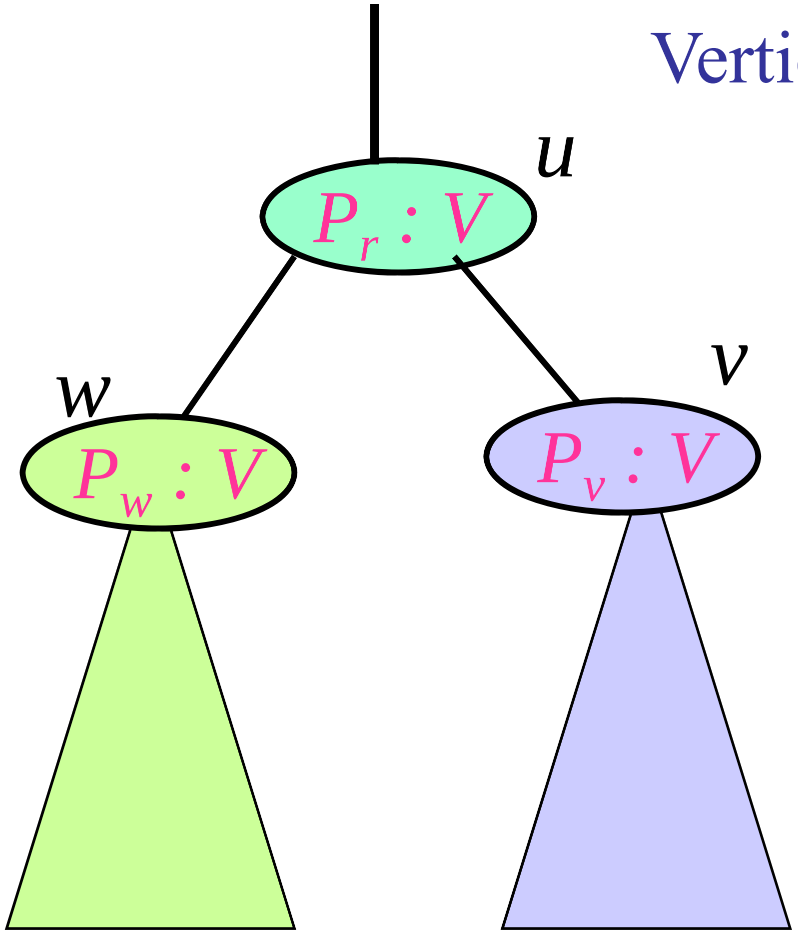


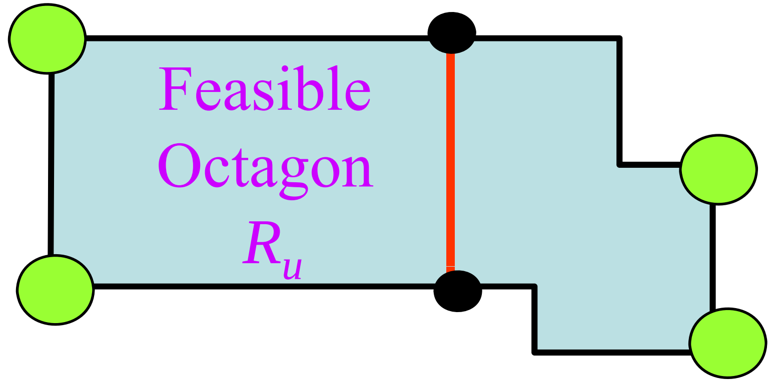
General computation at an internal node



General computation at an internal node

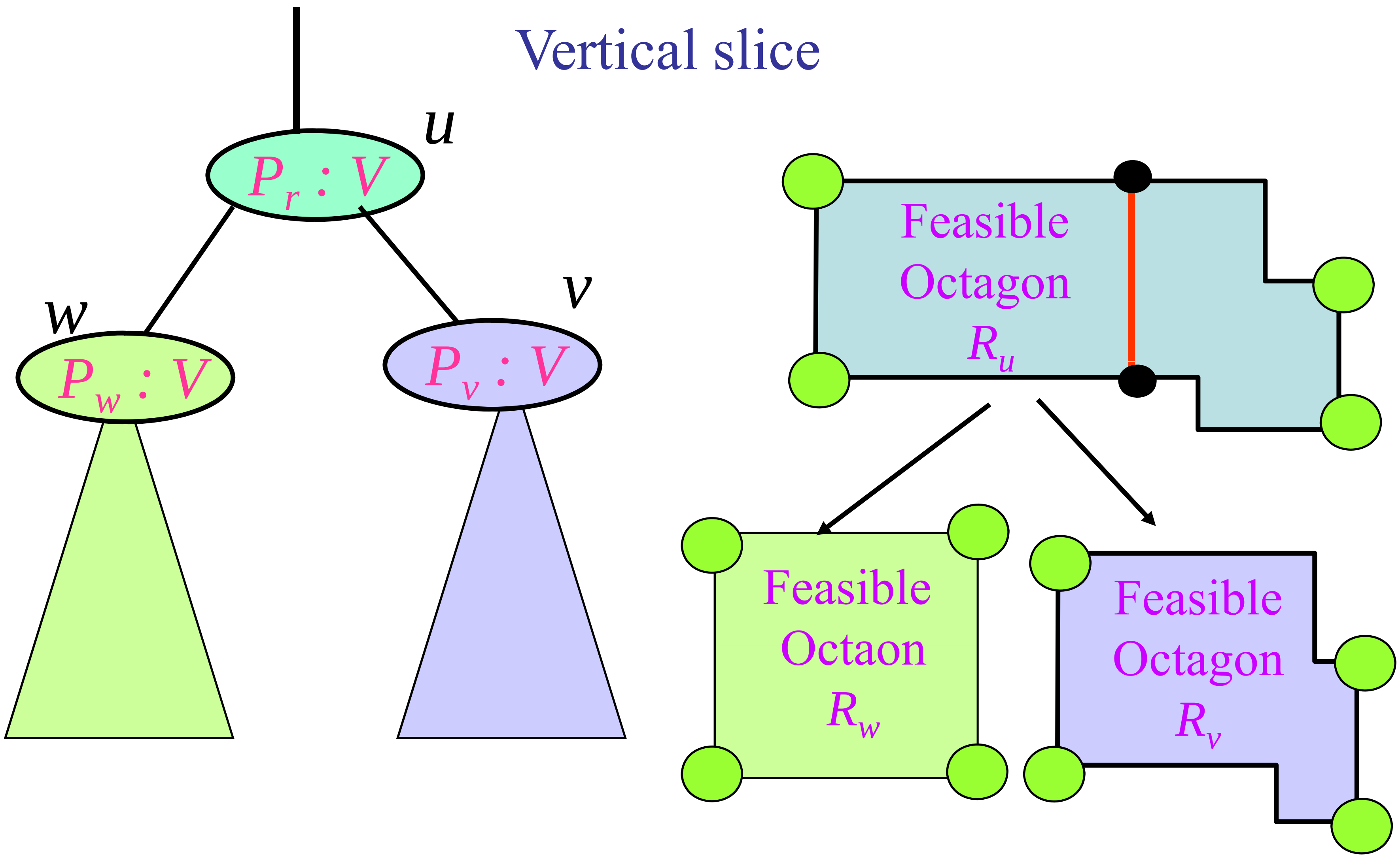
Vertical slice



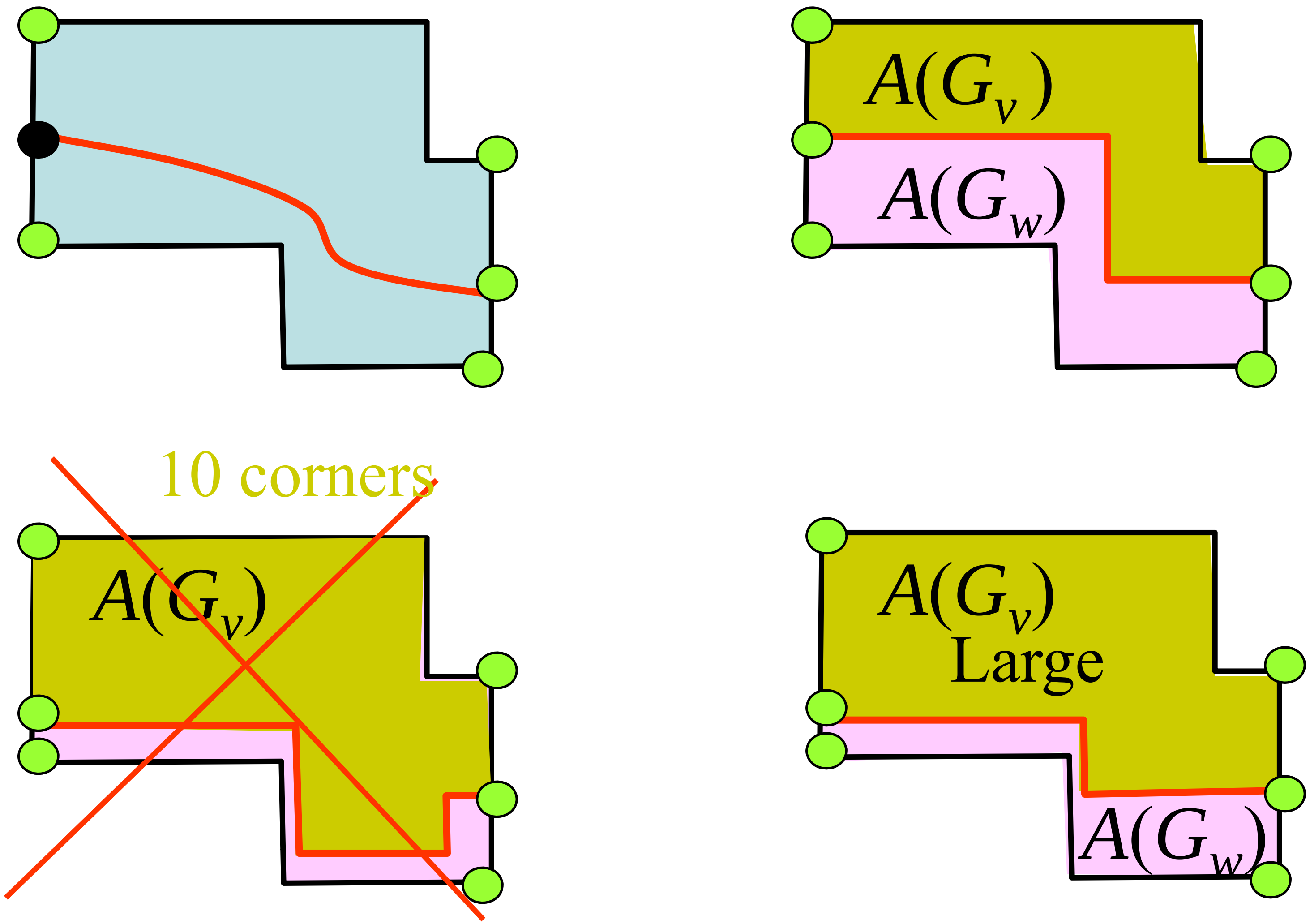


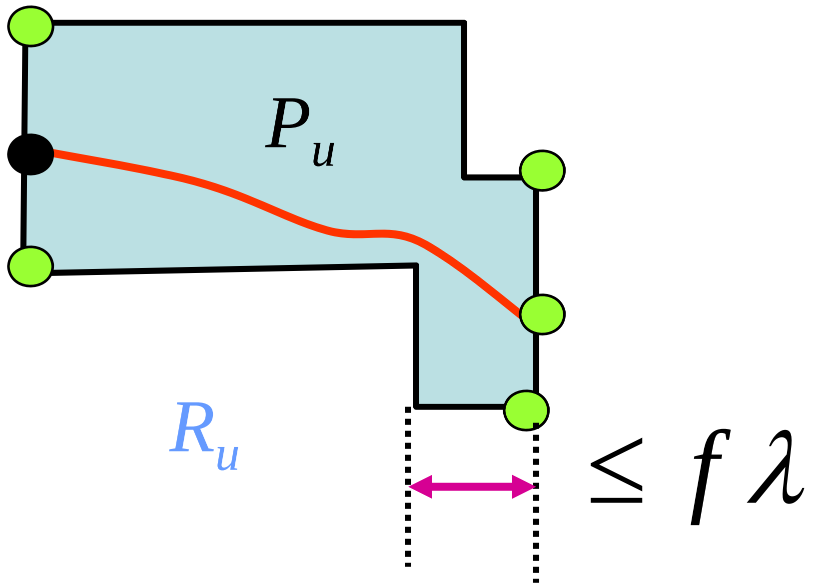
Embed the slicing path in R_u

General computation at an internal node

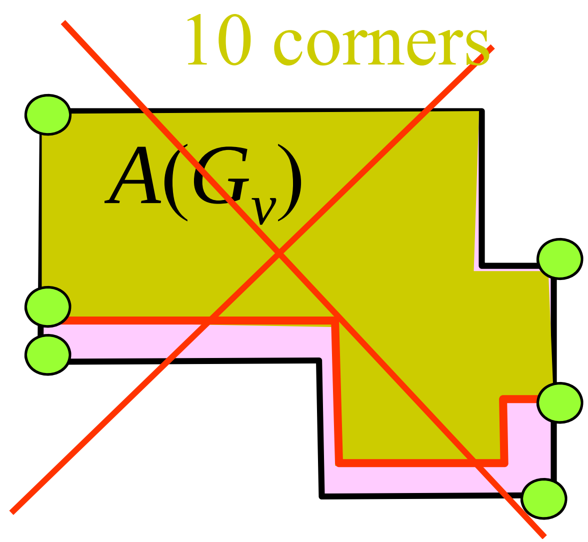


Horizontal slice



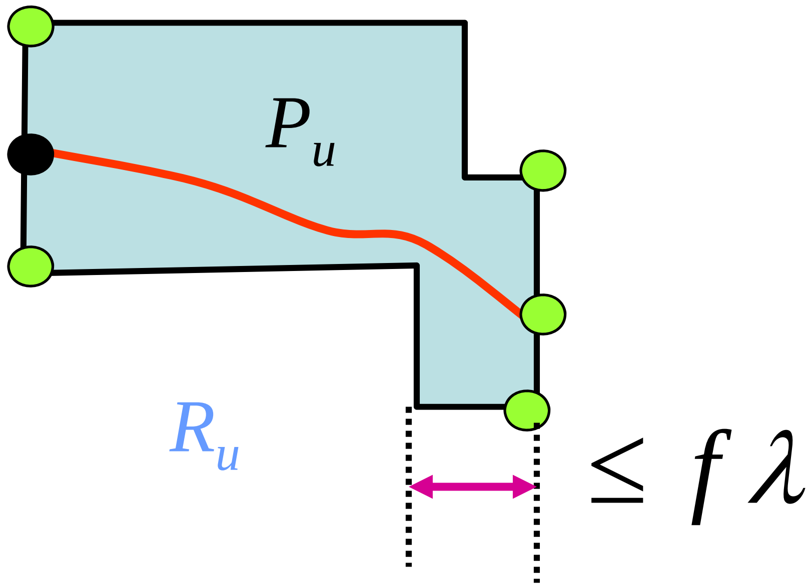


very small foot-lenght



f number of inner faces in G

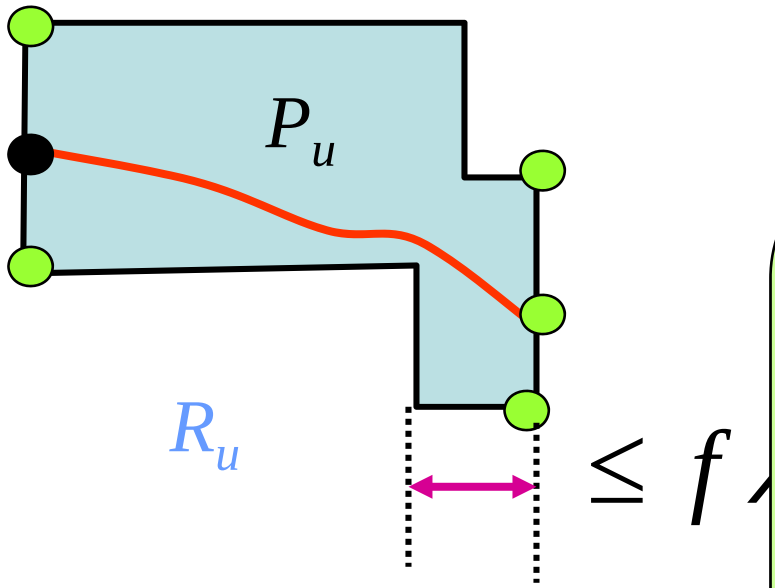
$$\lambda = \frac{A_{\min}}{f\,H}$$



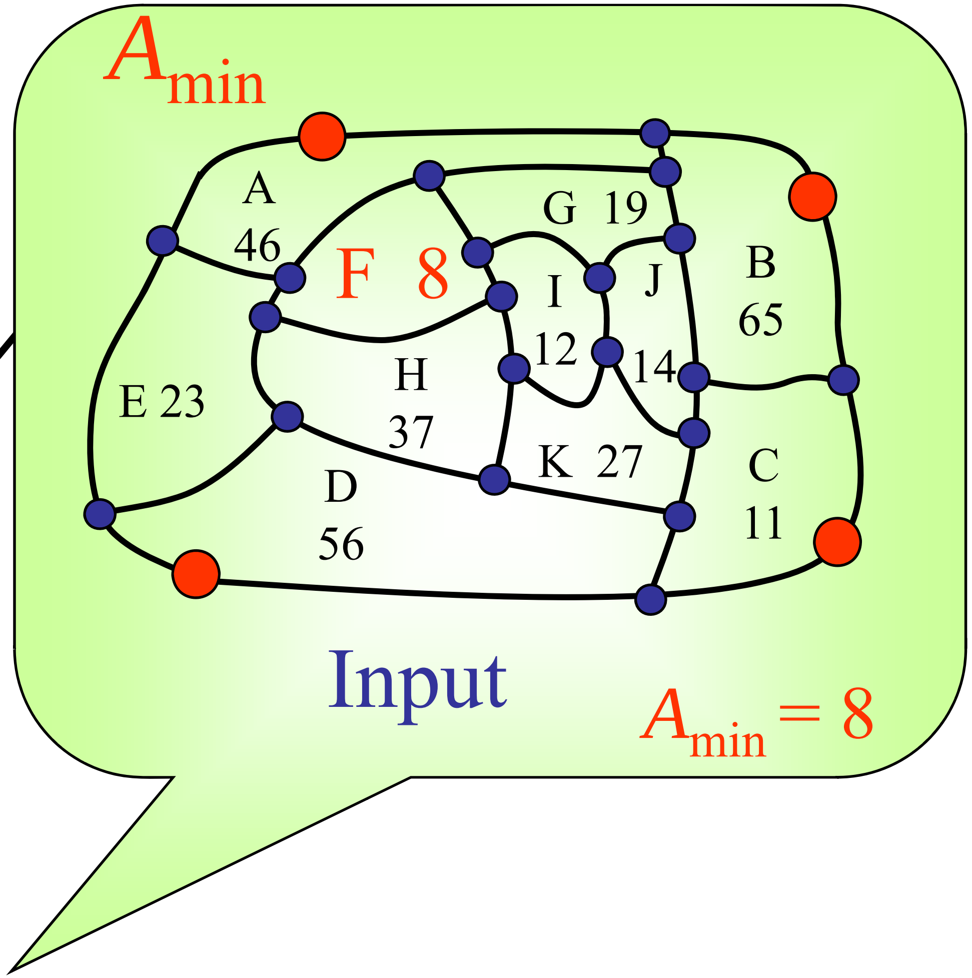
very small foot-lenght

f number of inner faces in G

$$\lambda = \frac{A_{\min}}{f\,H}$$

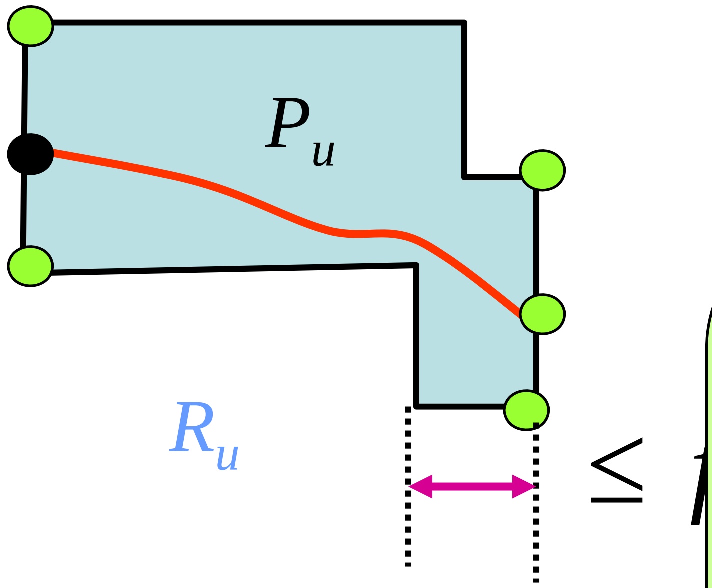


very small foot-lenght



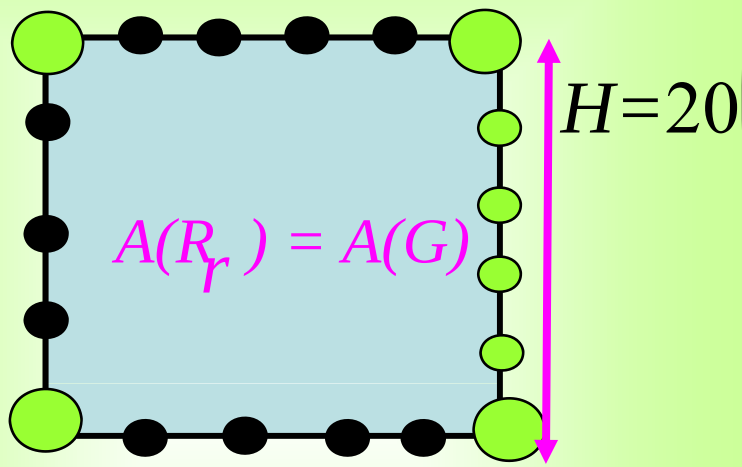
f number of inner faces in G

$$\lambda = \frac{A_{\min}}{f\,H}$$



very small foot-lengh

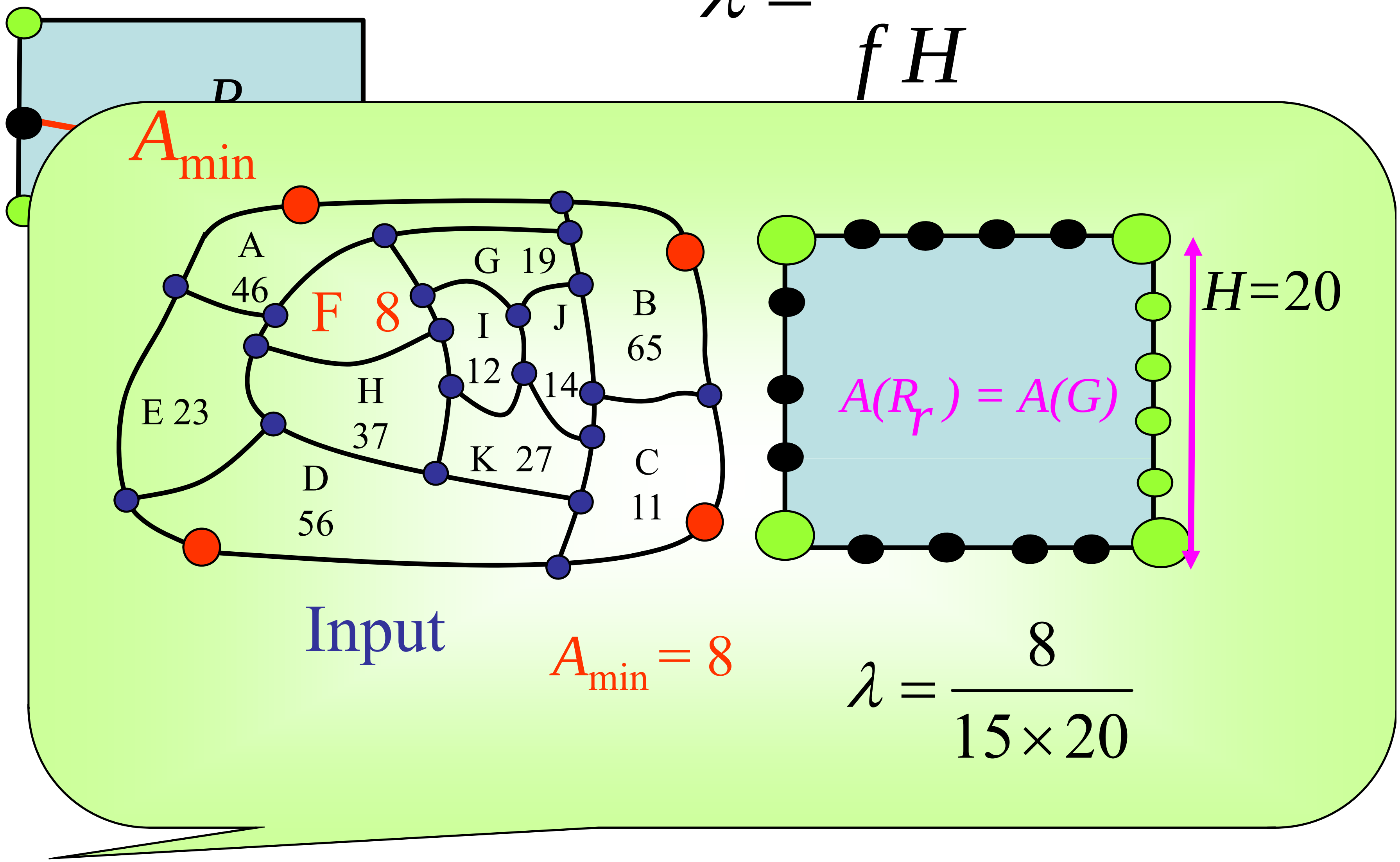
H height of initial rectangle R_r



Input at root R_r

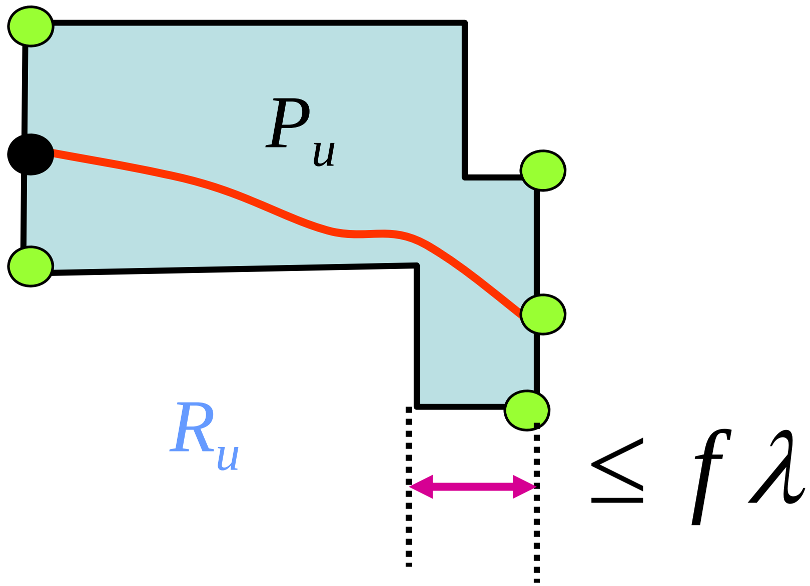
f number of inner faces in G

$$\lambda = \frac{A_{\min}}{f\,H}$$

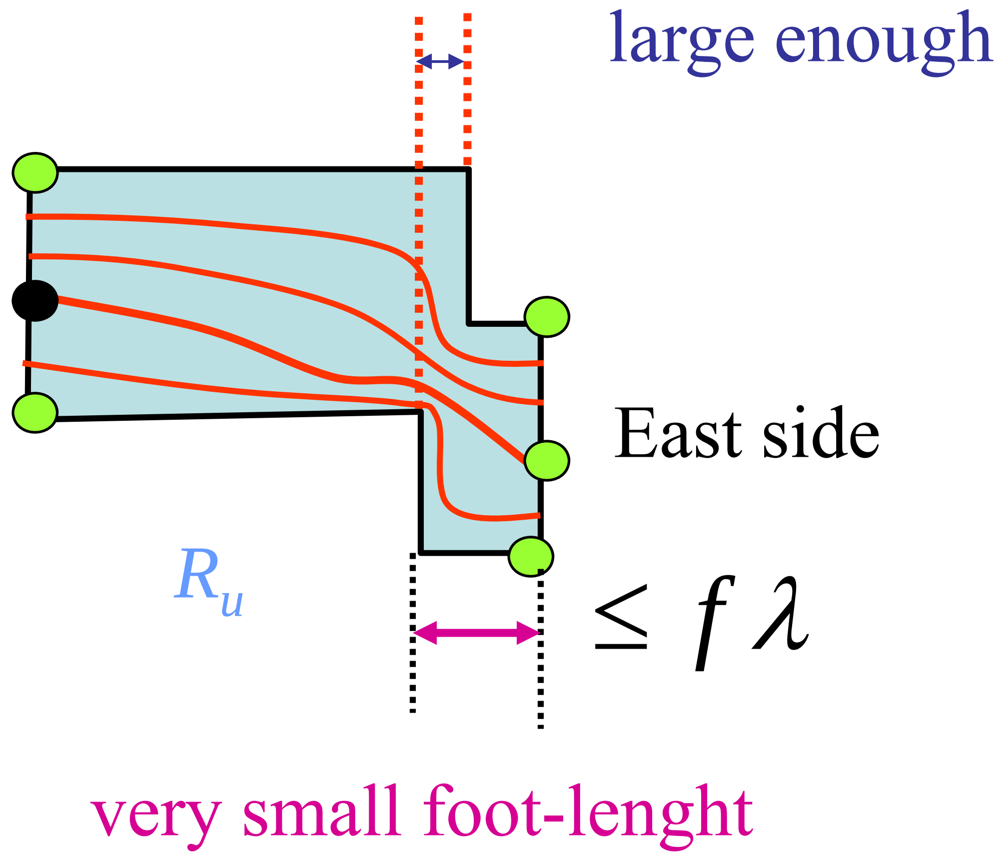


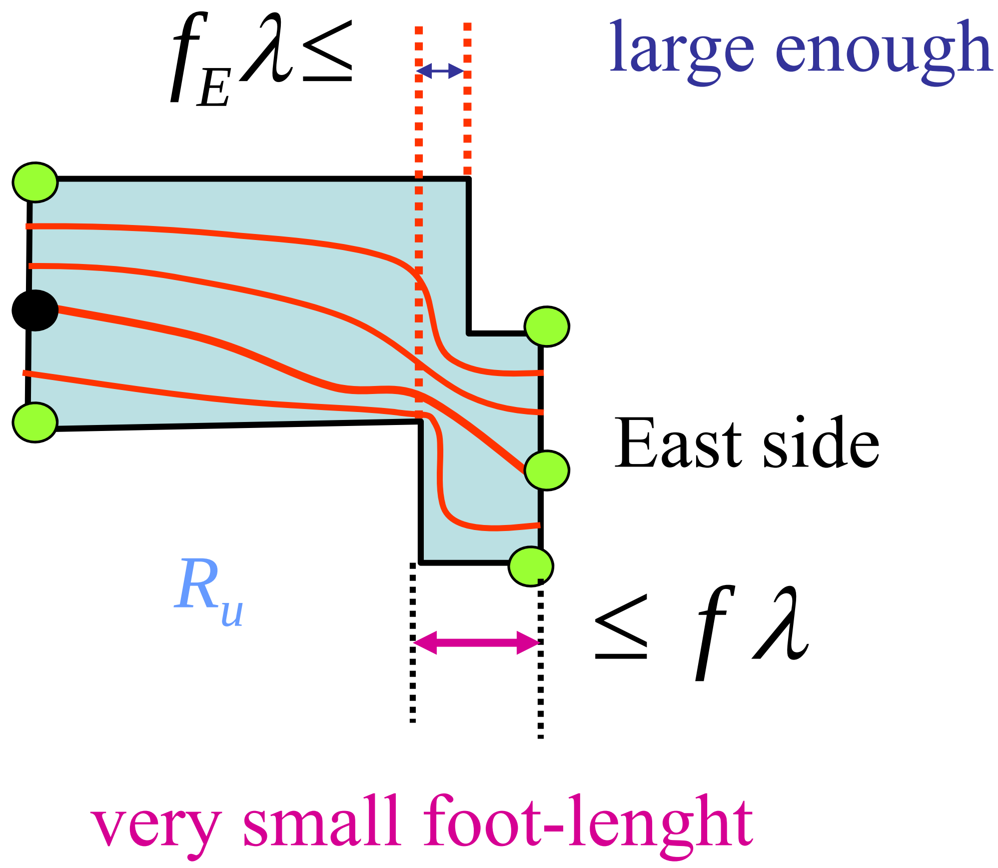
f number of inner faces in G

$$\lambda = \frac{A_{\min}}{f\,H}$$



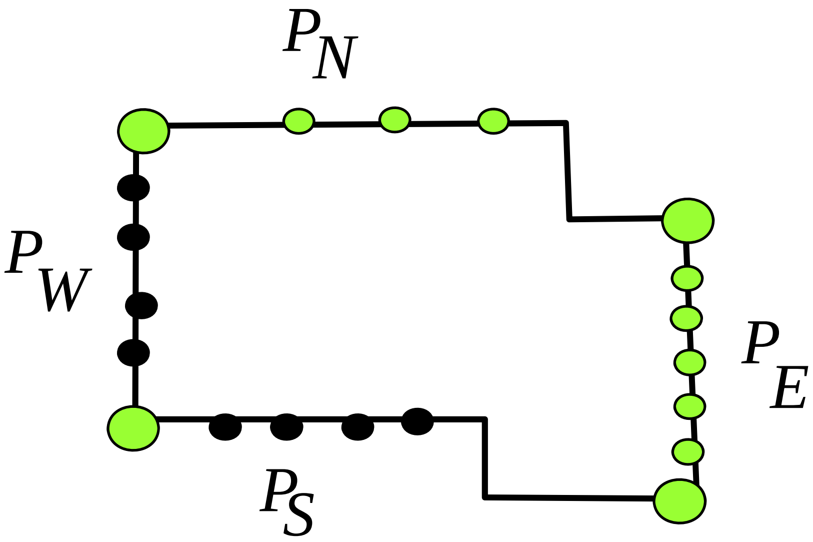
very small foot-lenght

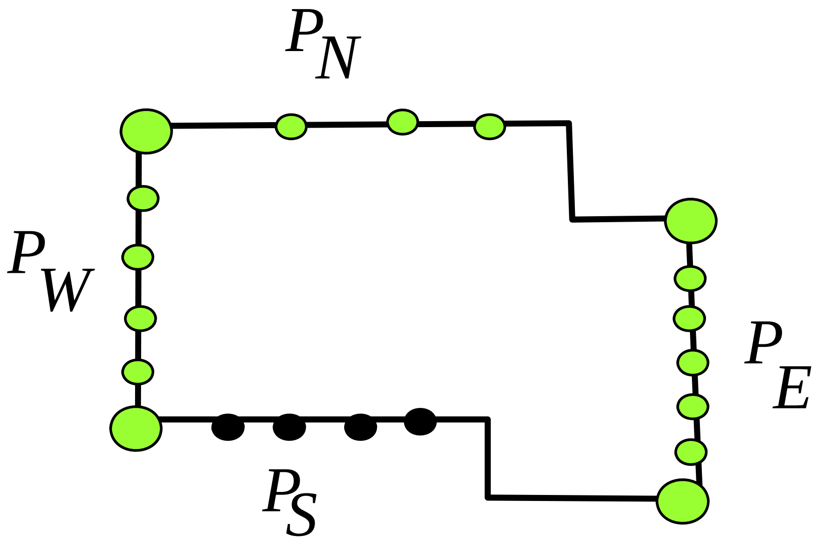

large enough
East side
R_u
$\leq f\lambda$
very small foot-lenght



f_E The number of inner faces each of which has an edge on the east side.

Computation at a leaf node





Time Complexity

Overall time complexity is linear.

Conclusion

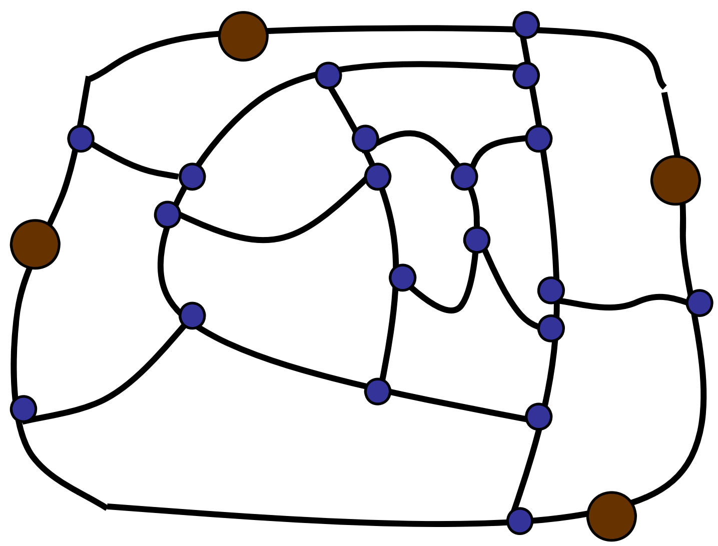
We have presented a linear algorithm for prescribed area octagonal drawings of good slicing graphs.

We also give a sufficient condition for a graph of maximum degree 3 to be a good slicing graph and give a linear-time algorithm to find a good slicing tree of such graphs.

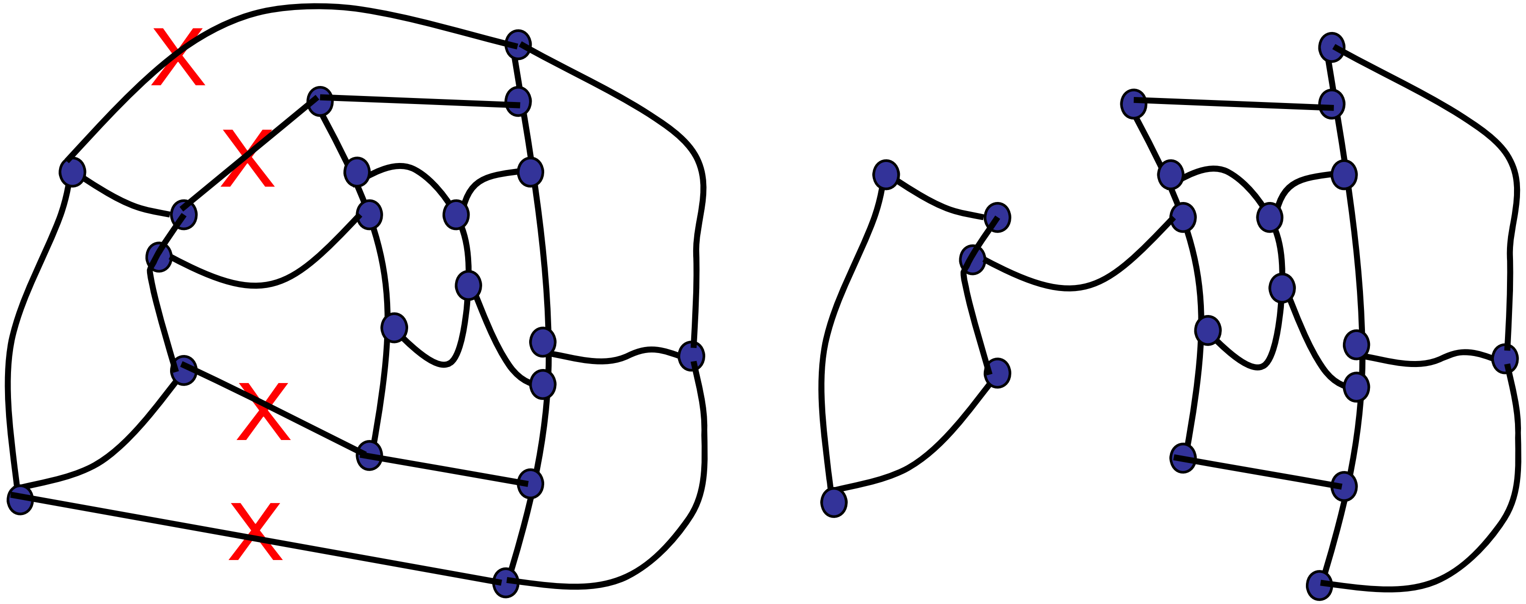
Obtaining such an algorithm for larger classes of graphs is our future work.

A sufficient condition for a good slicing graph

A good slicing tree can be found in linear time for a cyclically 5-edge connected plane cubic graph.

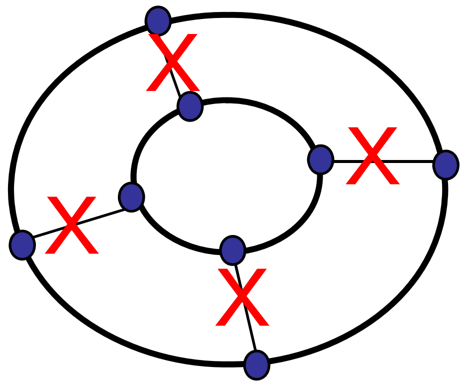


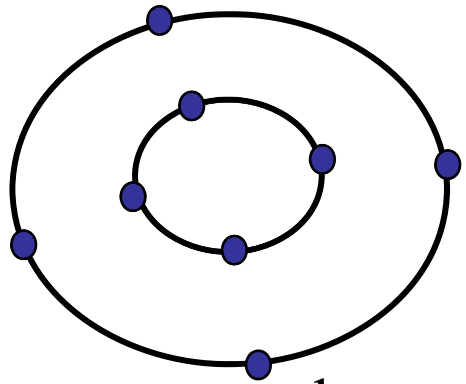
Cyclically 5-edge Connected Cubic Plane Graphs



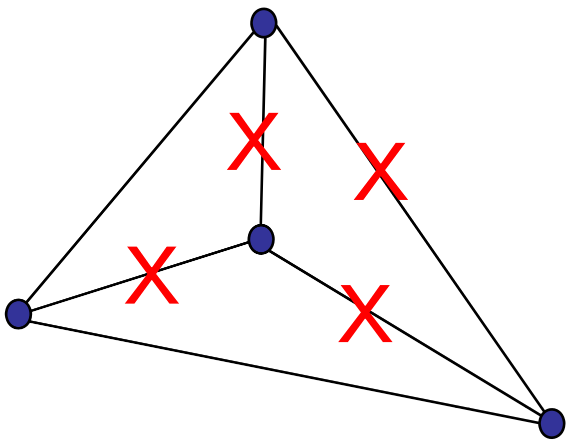
Removal of any set of less than 5-edges leaves a graph such that exactly one of the connected components has a cycle.

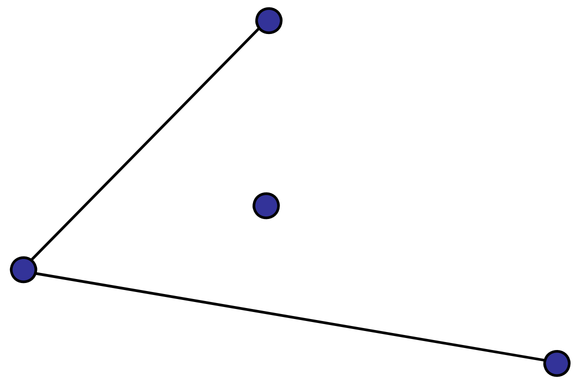
Not cyclically 5-edge connected





Two connected components having cycles.

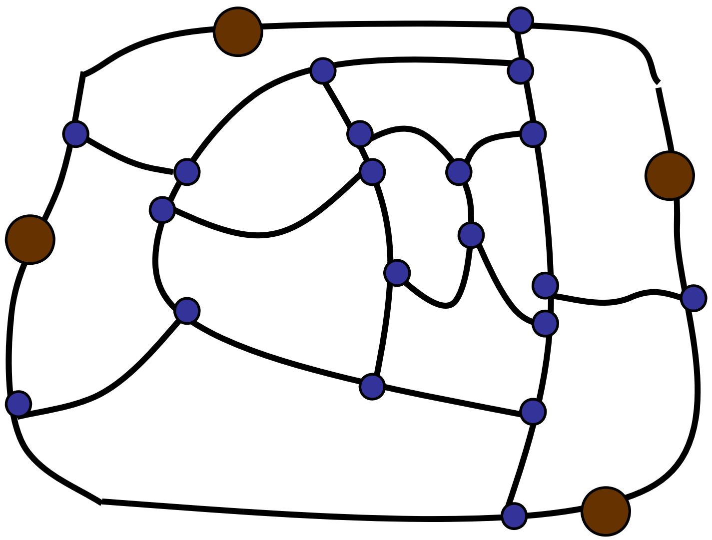




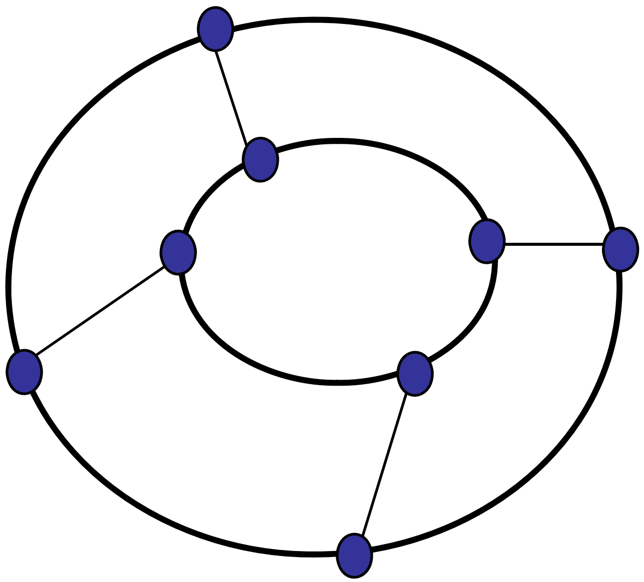
No connected component has a cycle.

A sufficient condition for a good slicing graph

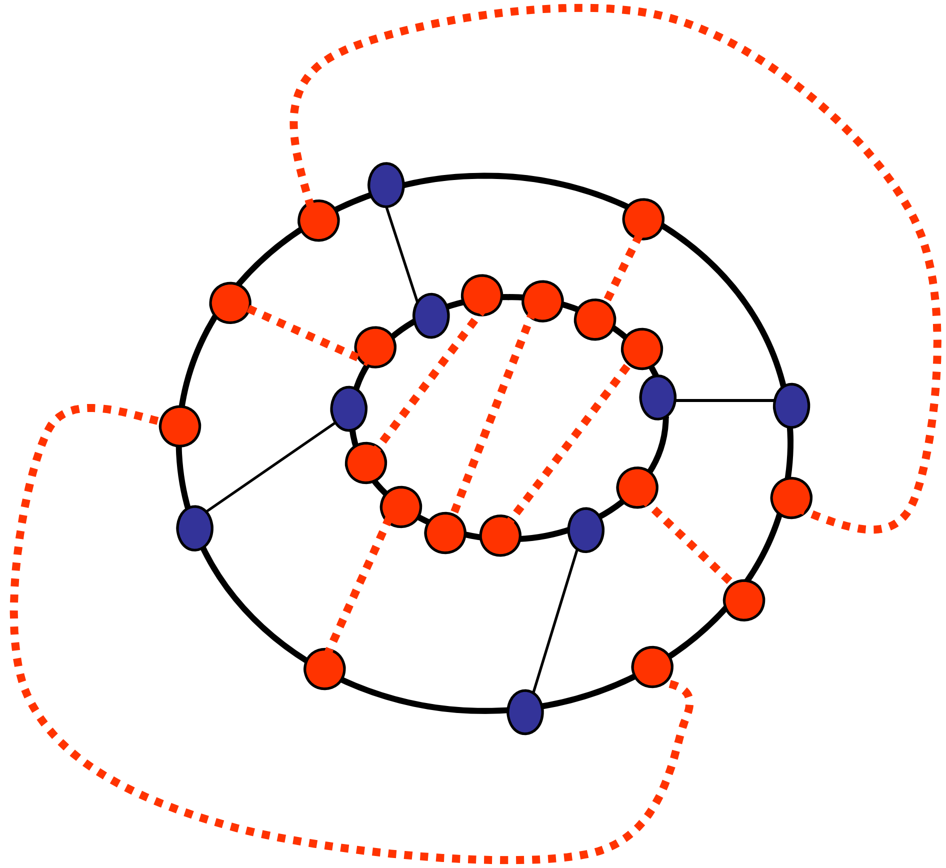
Any graph G obtained from a cyclically 5-edge connected plane cubic graph by inserting four vertices of degree 2 on outer face is a good slicing graph.



Augmentation



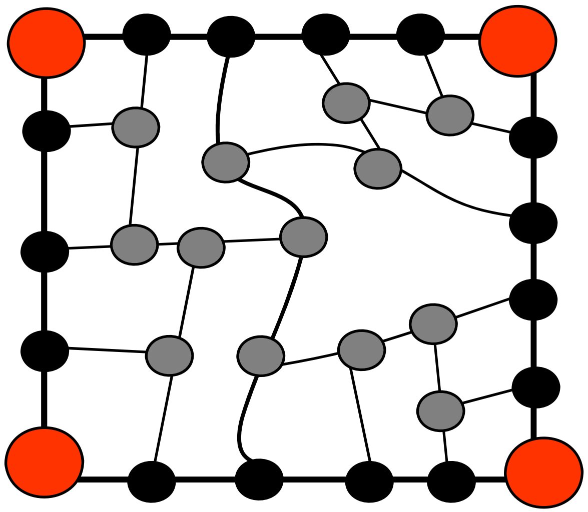
Augmentation

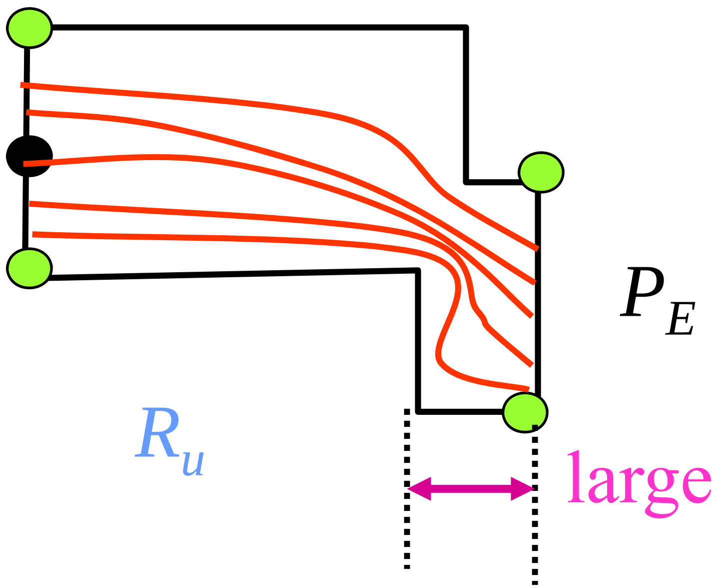


Thank You

Algorithm

Initialization at root

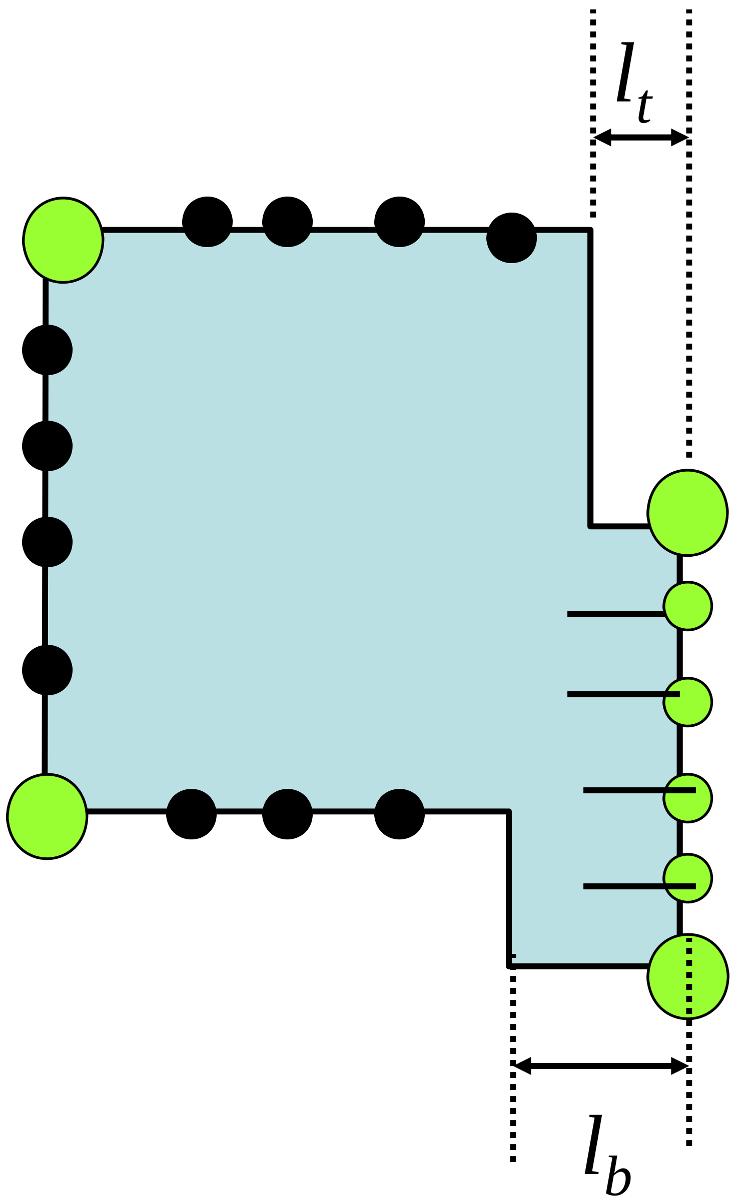




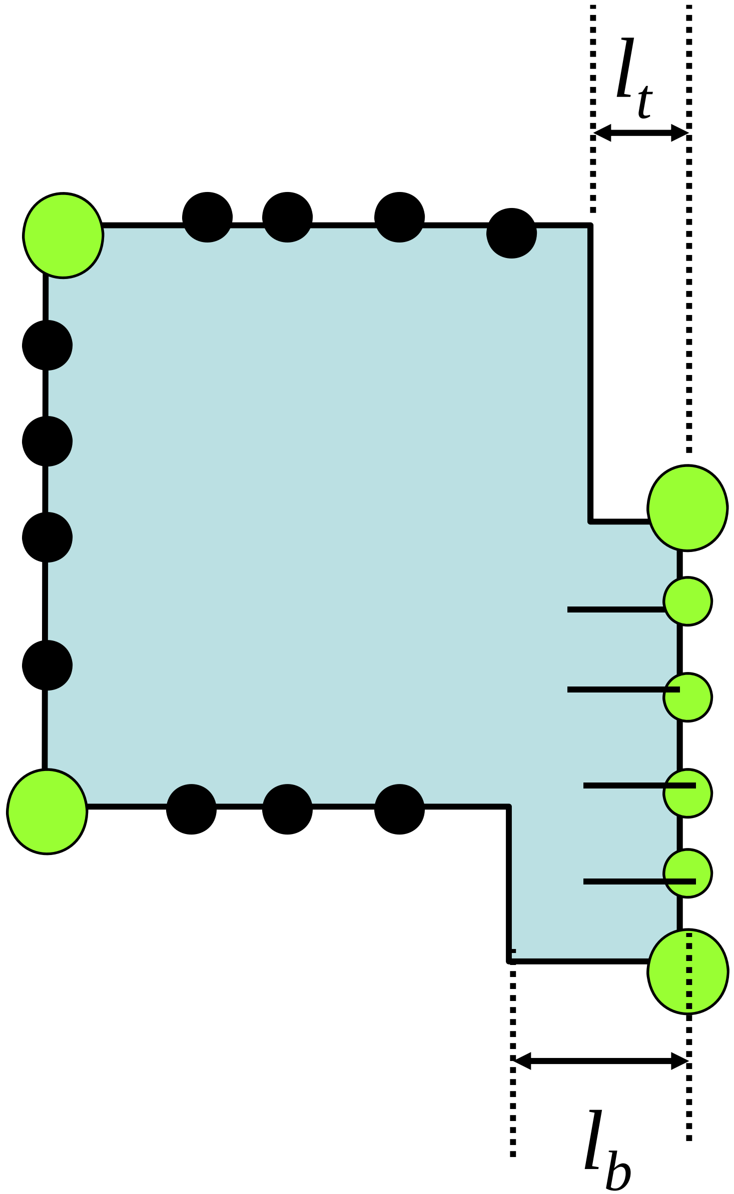
If a large number of inner faces have edges on P_E then foot-lengh should be large enough.

Dimensions of R_u play crucial roles.

Dimensions of a Feasible Octagon ?



l_t
l_b



$$\boxed{l_t + f_E\,\delta \le l_b < f\,\delta}$$

f number of inner faces in G

f_E The number of inner faces each of which has an edge on the east side.

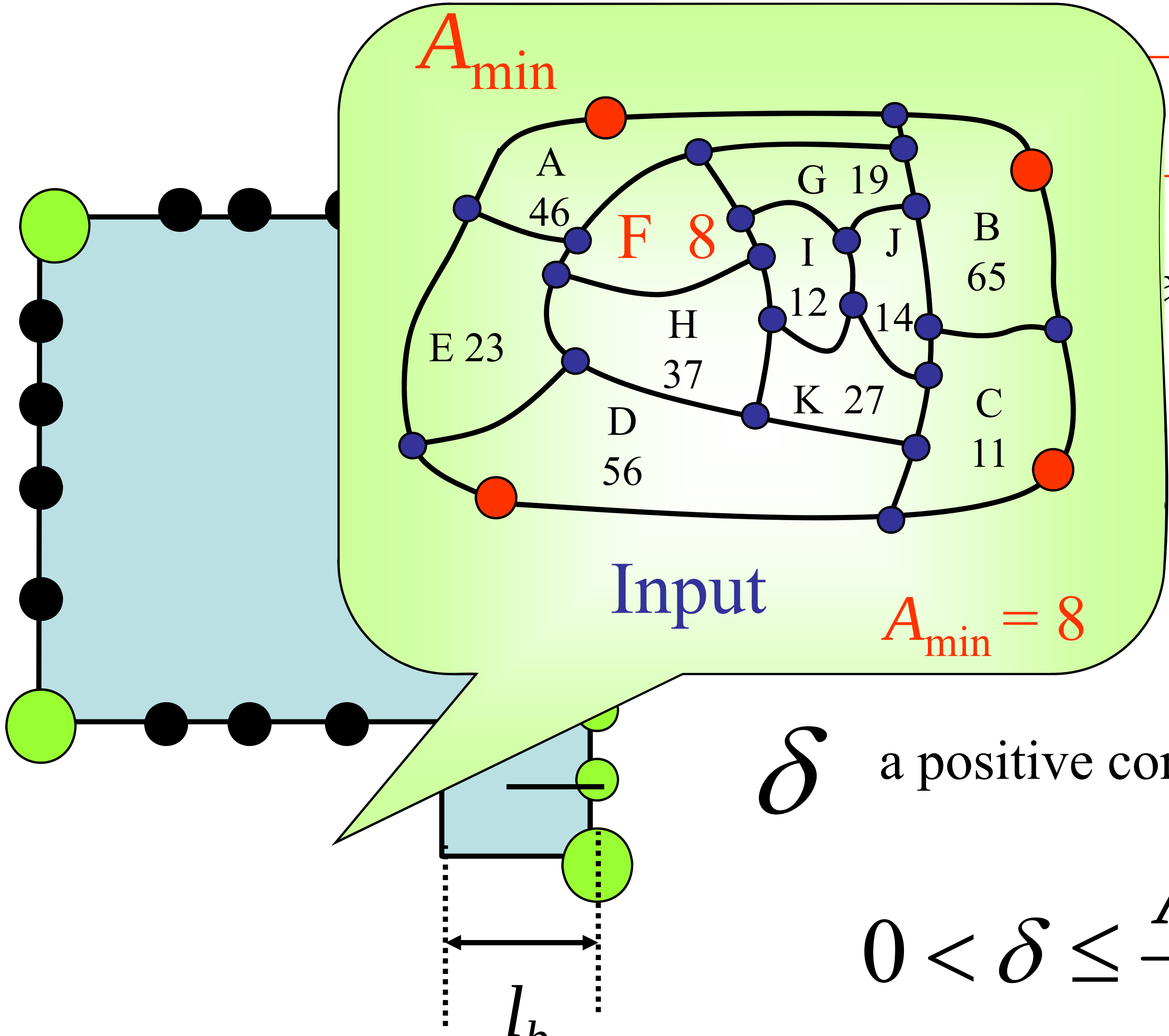
δ a positive constant.

$$0 < \delta \le \frac{A_{\min}}{f\,H}$$

$f\,\delta$

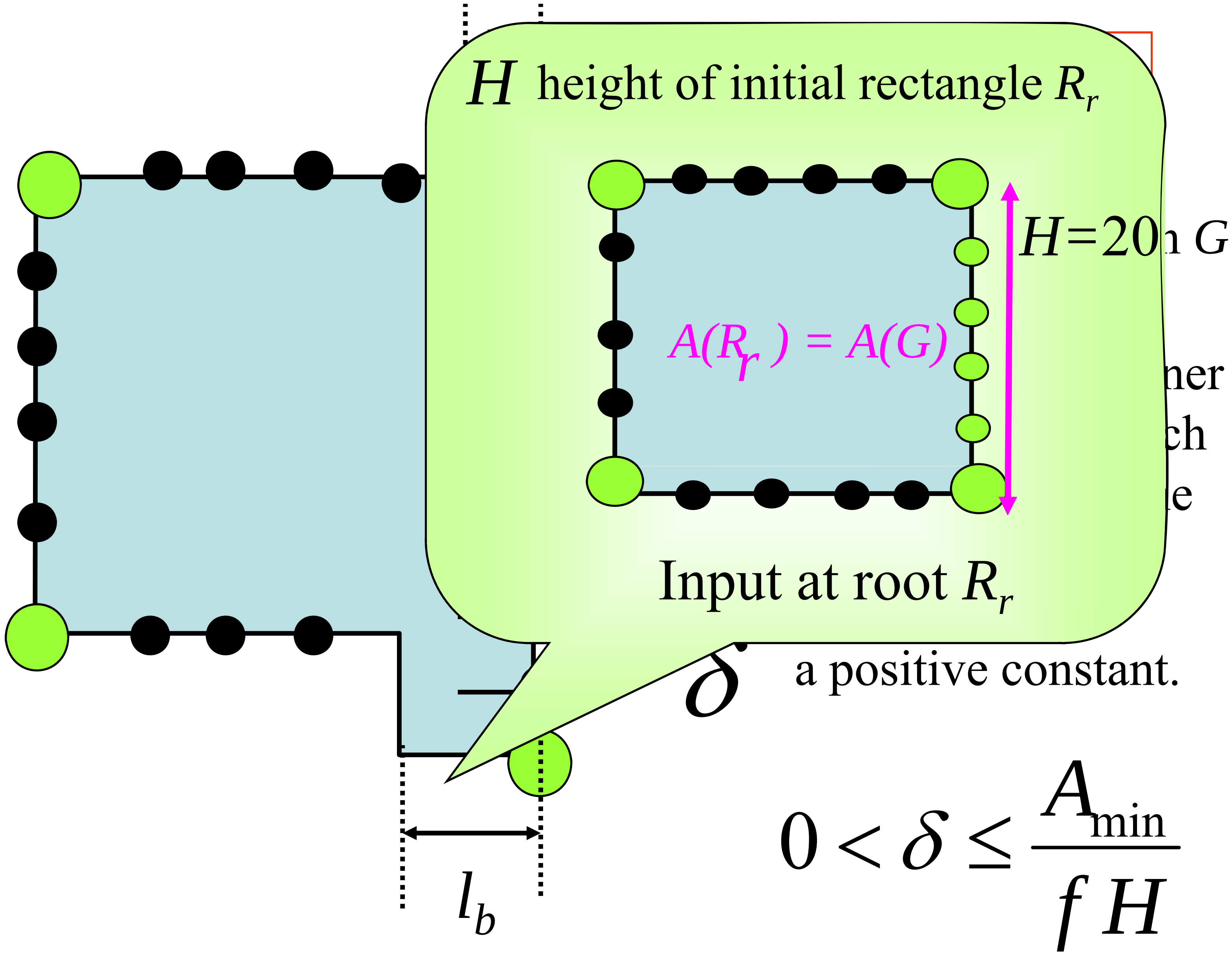
aces in G

of inner
f which
on the

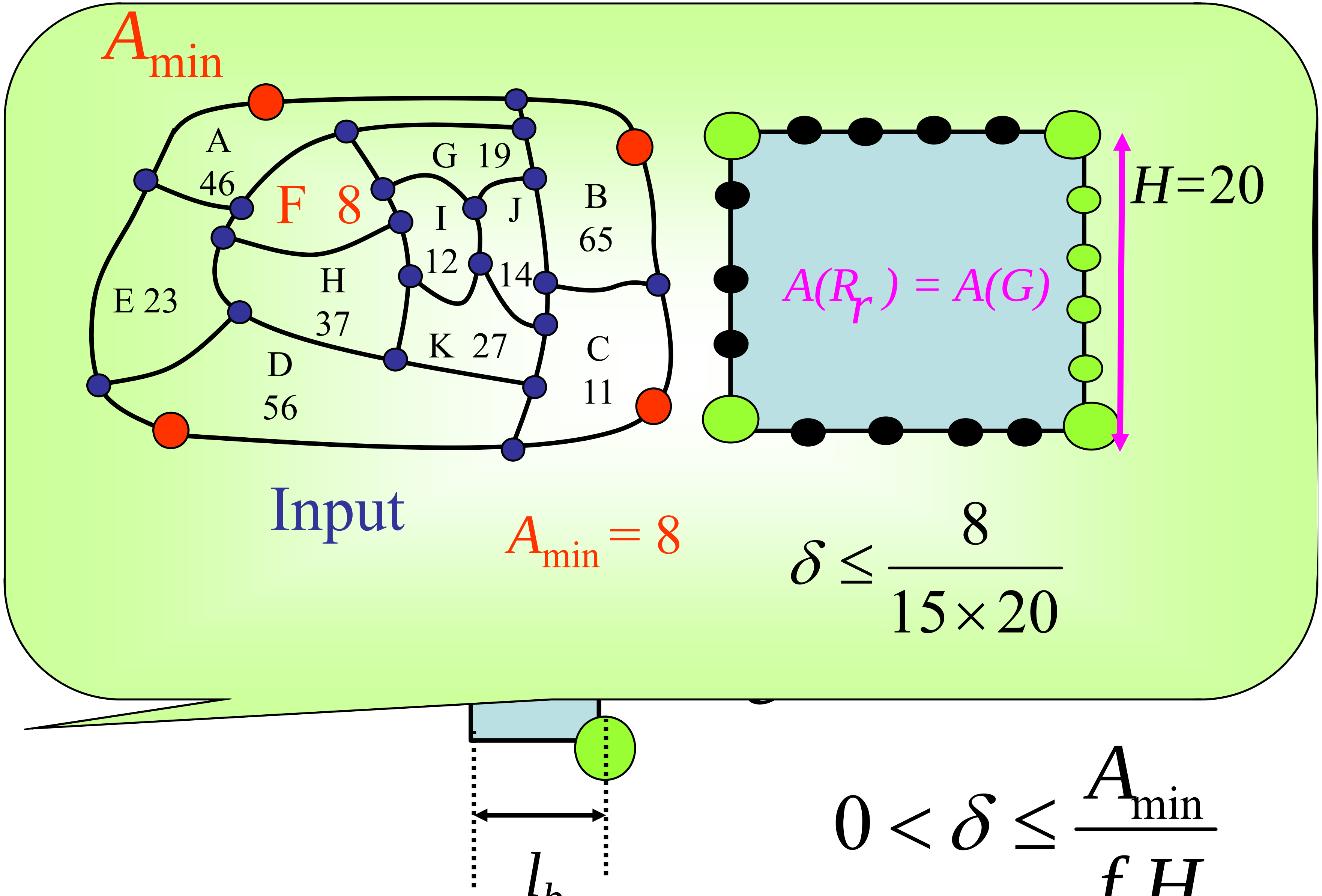


δ a positive constant.

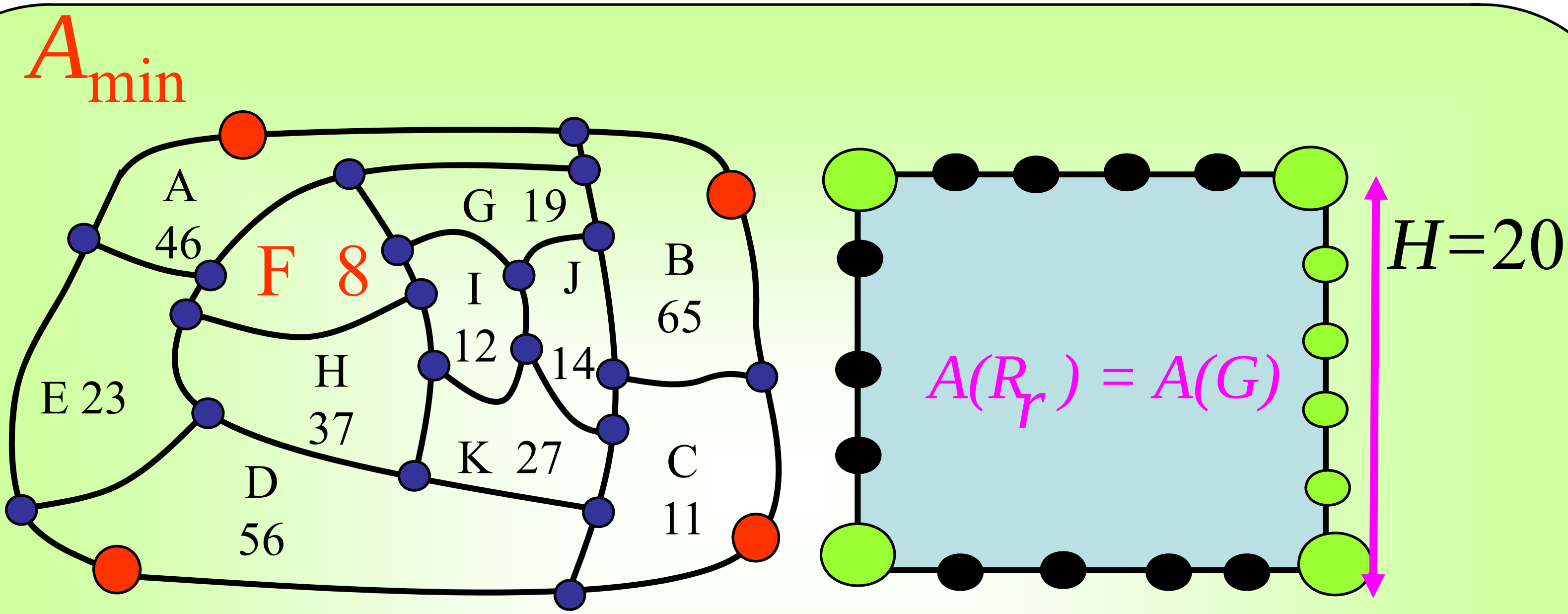
$$0 < \delta \le \frac{A_{\min}}{f\,H}$$



H height of initial rectangle R_r
H=20
G
A(R_r) = A(G)
Input at root R_r
δ
a positive constant.
l_b
0 < δ ≤ A_min / f H



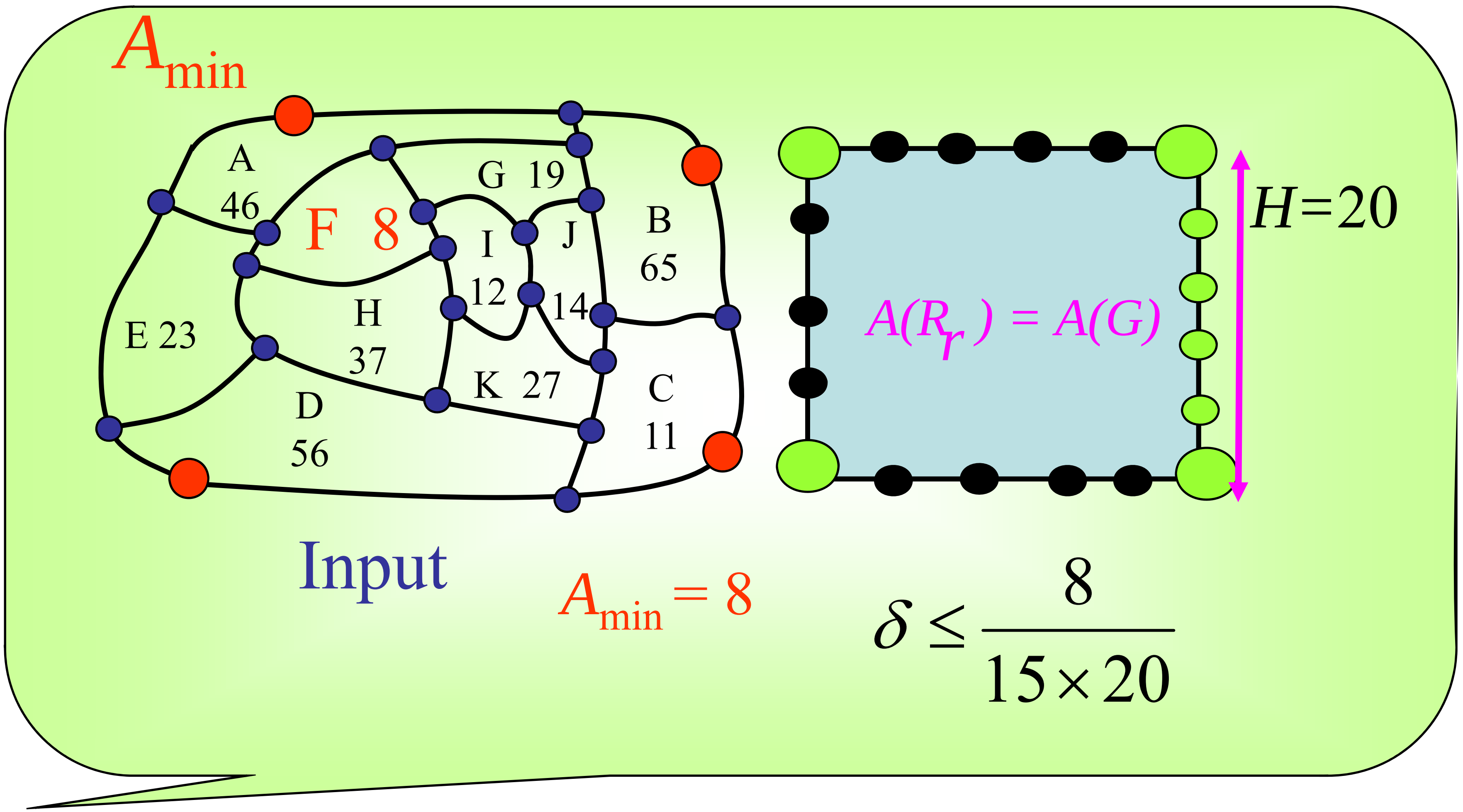
$$0 < \delta \leq \frac{A_{min}}{f\,H}$$



Input

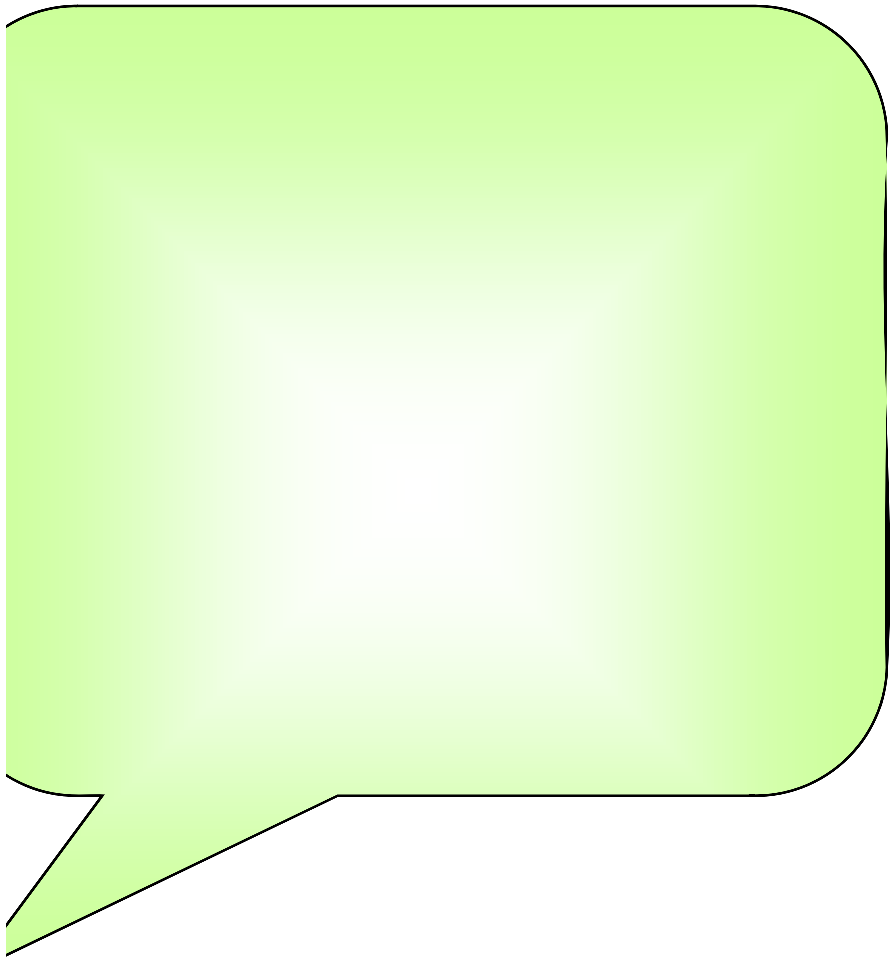
$A_{\min} = 8$

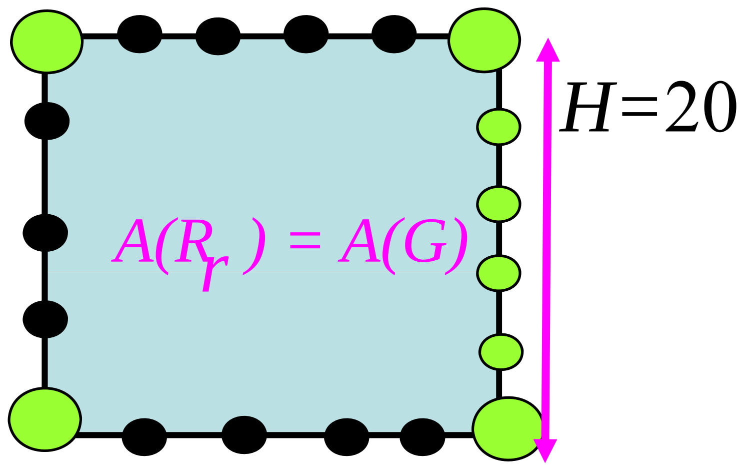
$$\delta \le \frac{8}{15 \times 20}$$



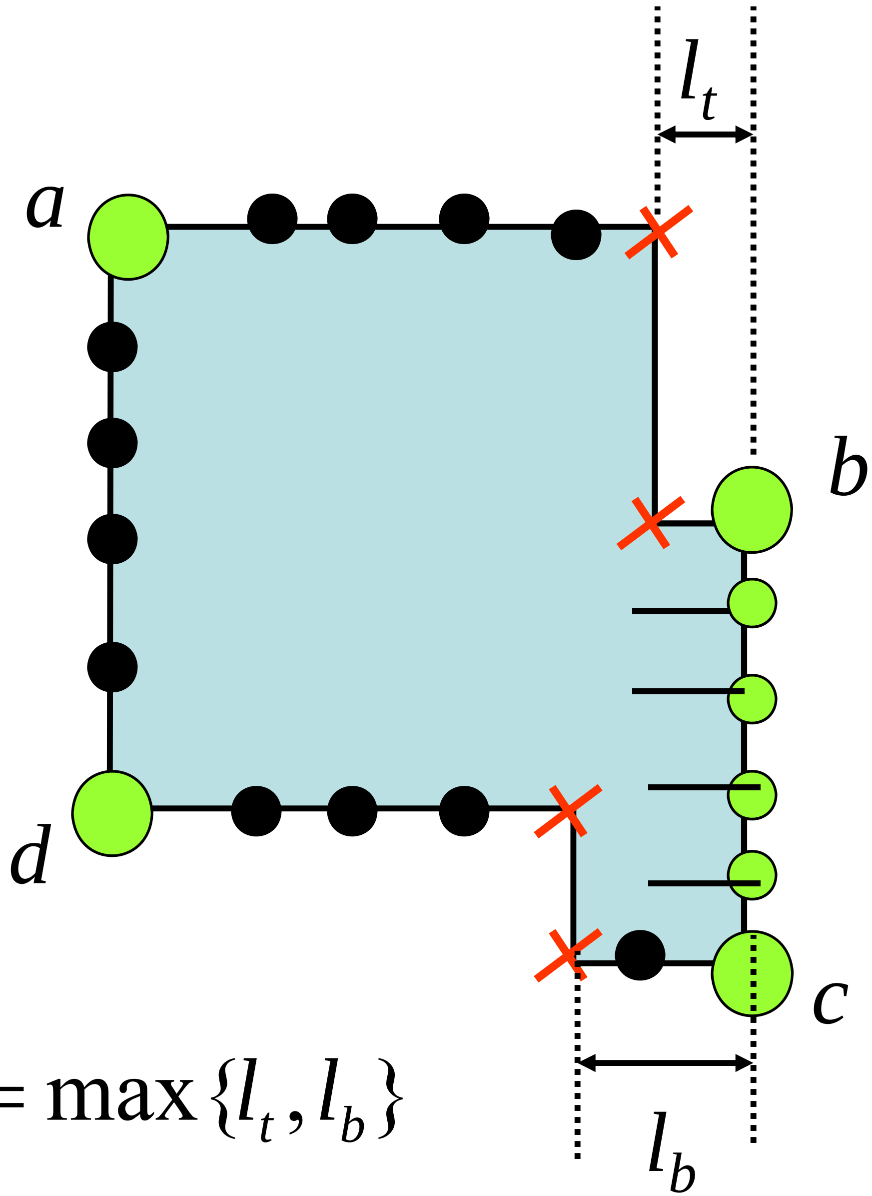
A_{min}
A 46
F 8
G 19
I 12
J
B 65
E 23
H 37
14
K 27
C 11
D 56
$A(R_r) = A(G)$
$H=20$
Input
$A_{min} = 8$
$\delta \le \frac{8}{15 \times 20}$

H height of initial rectangle R_r





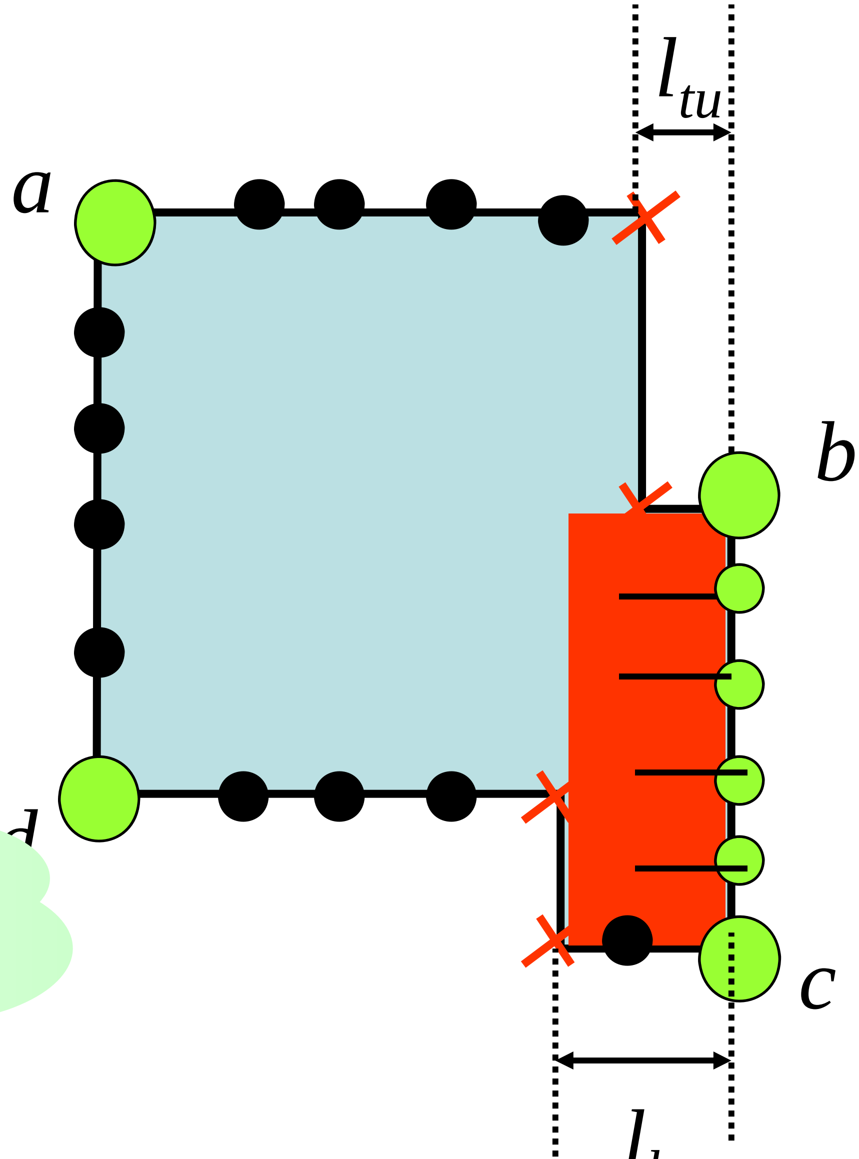
Input at root R_r



$$l = \max\{l_t, l_b\}$$

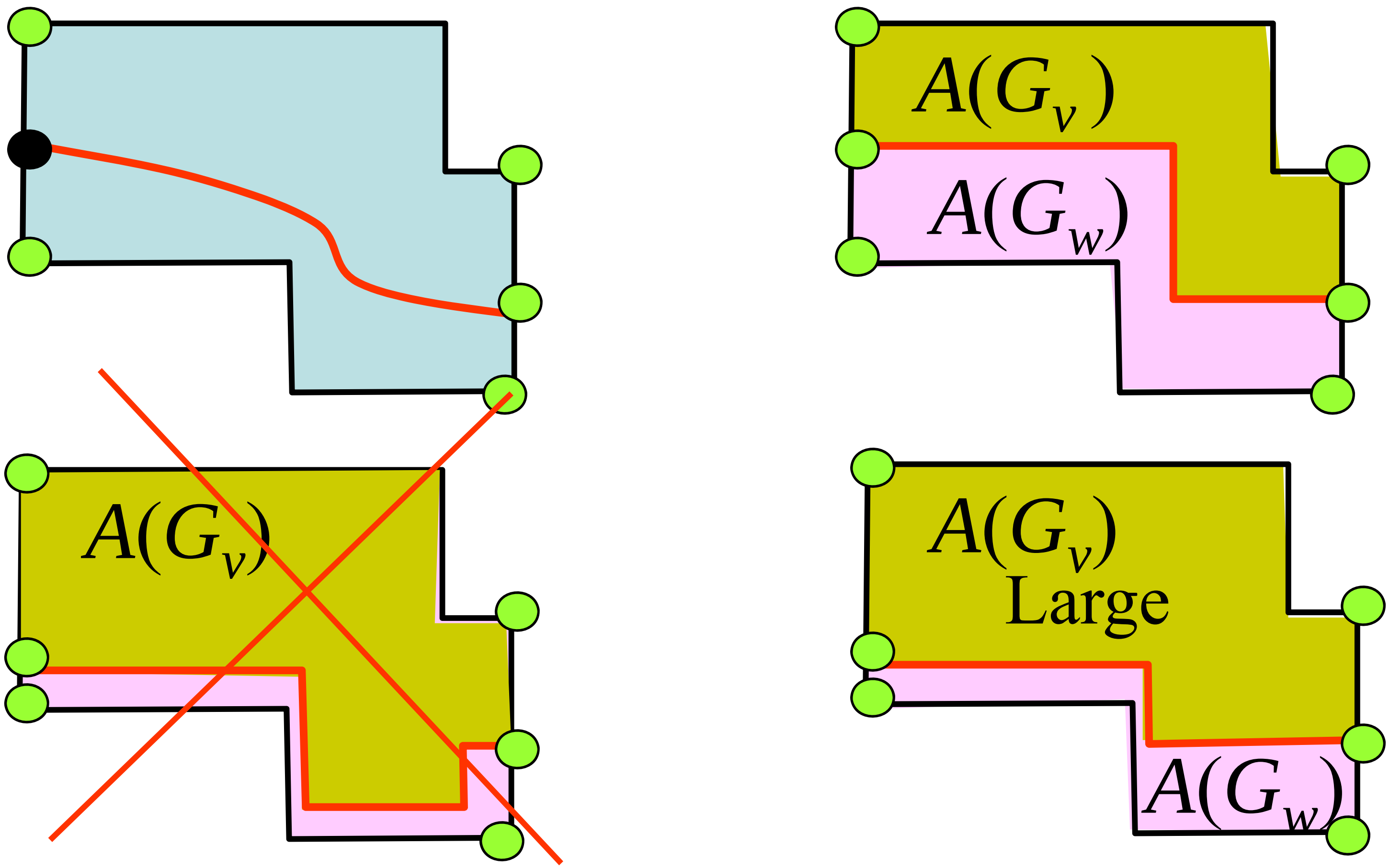
$$\boxed{l < f\delta}$$

$$l < f\,\delta$$



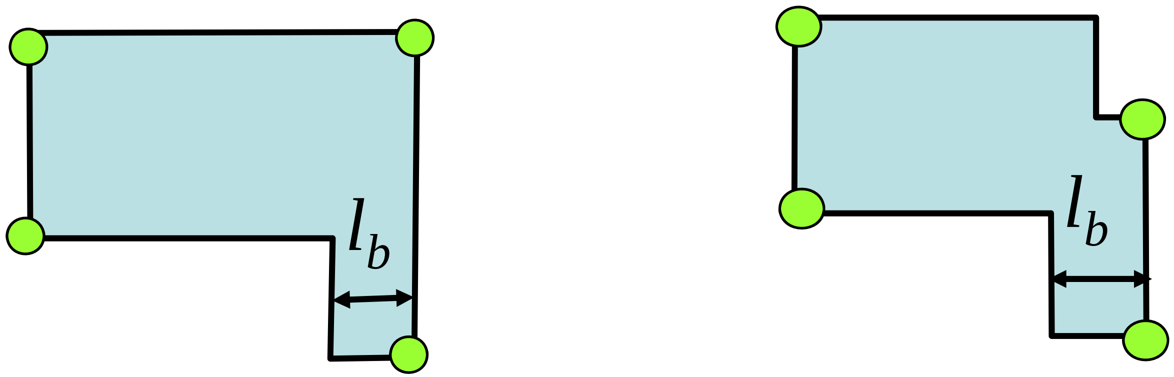
$$\text{Red area} < l_u H < f\,\delta H < A_{\min}$$

$$l_u = \max\{l_{tu}, l_{bu}\}$$

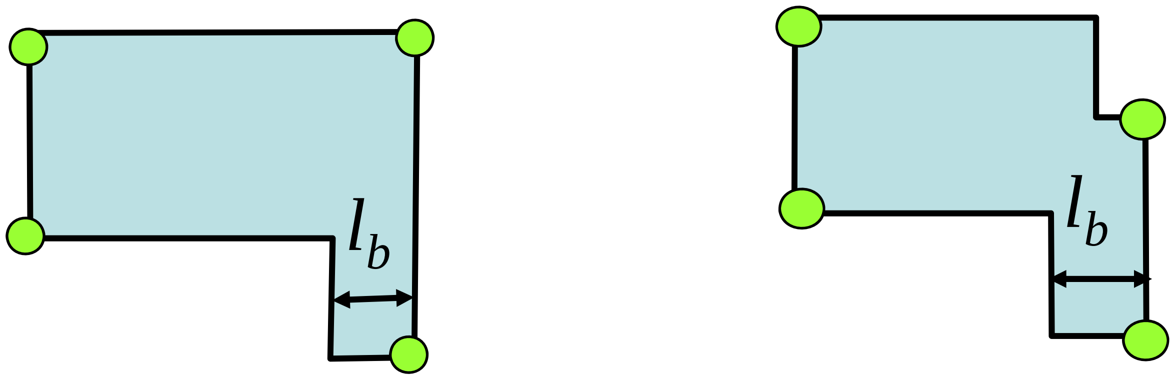


This situation does not occur.

$$l_b \geq f_E \delta$$



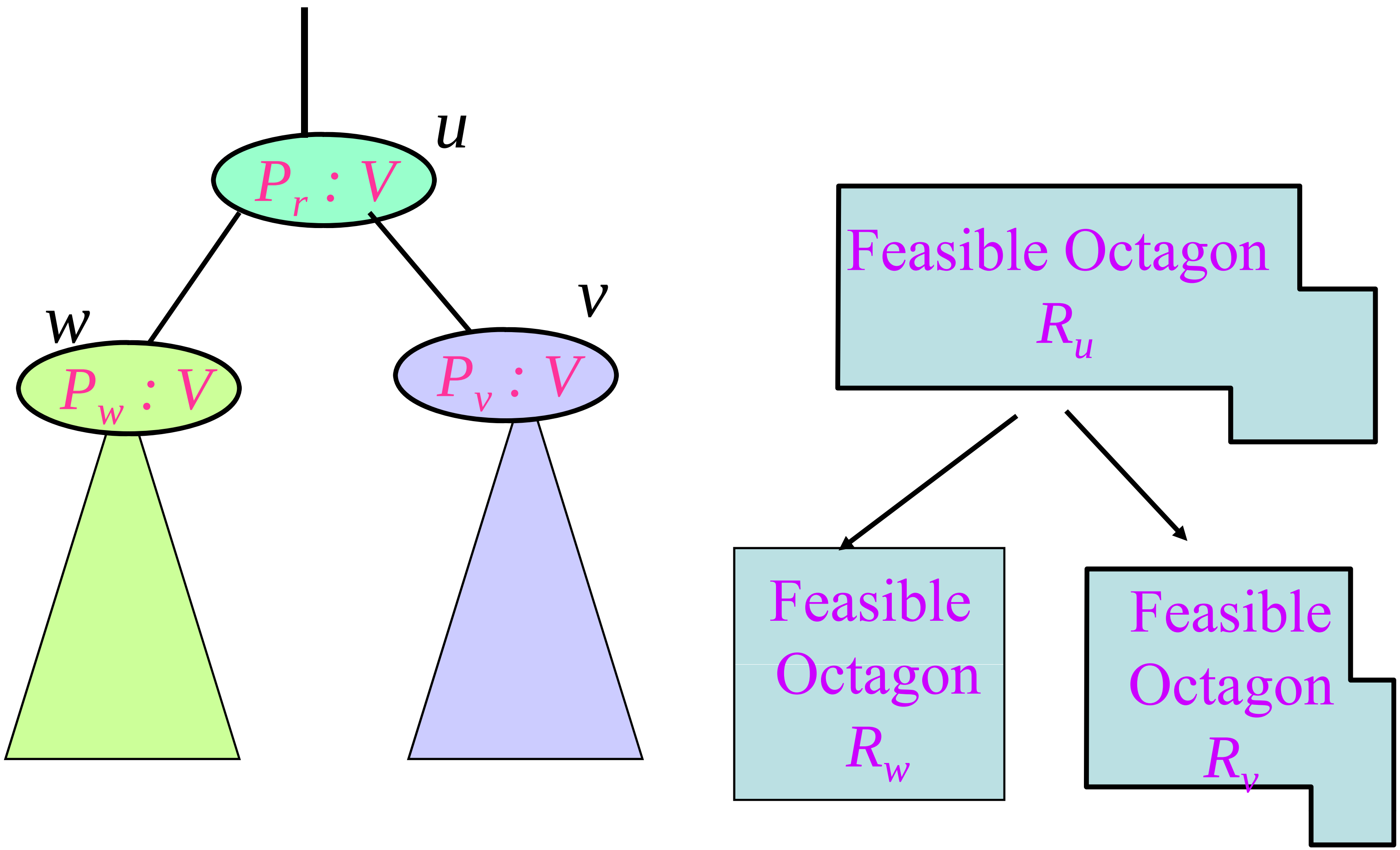
$$l_b \geq f_E \delta$$



$$l_{bu} \geq \delta$$

for a facial octagon whose x_{S1} is convex.

General computation at an internal node



Time Complexity

- Using a bottom-up computation on slicing tree, area of subgraphs for all internal nodes can be computed in linear time.

- With an O(n) time preprocessing, embedding of the slicing path at each internal node takes constant time.

- Computation time at a leaf node is proportional to the number of non-corner vertices on the west side of the face.

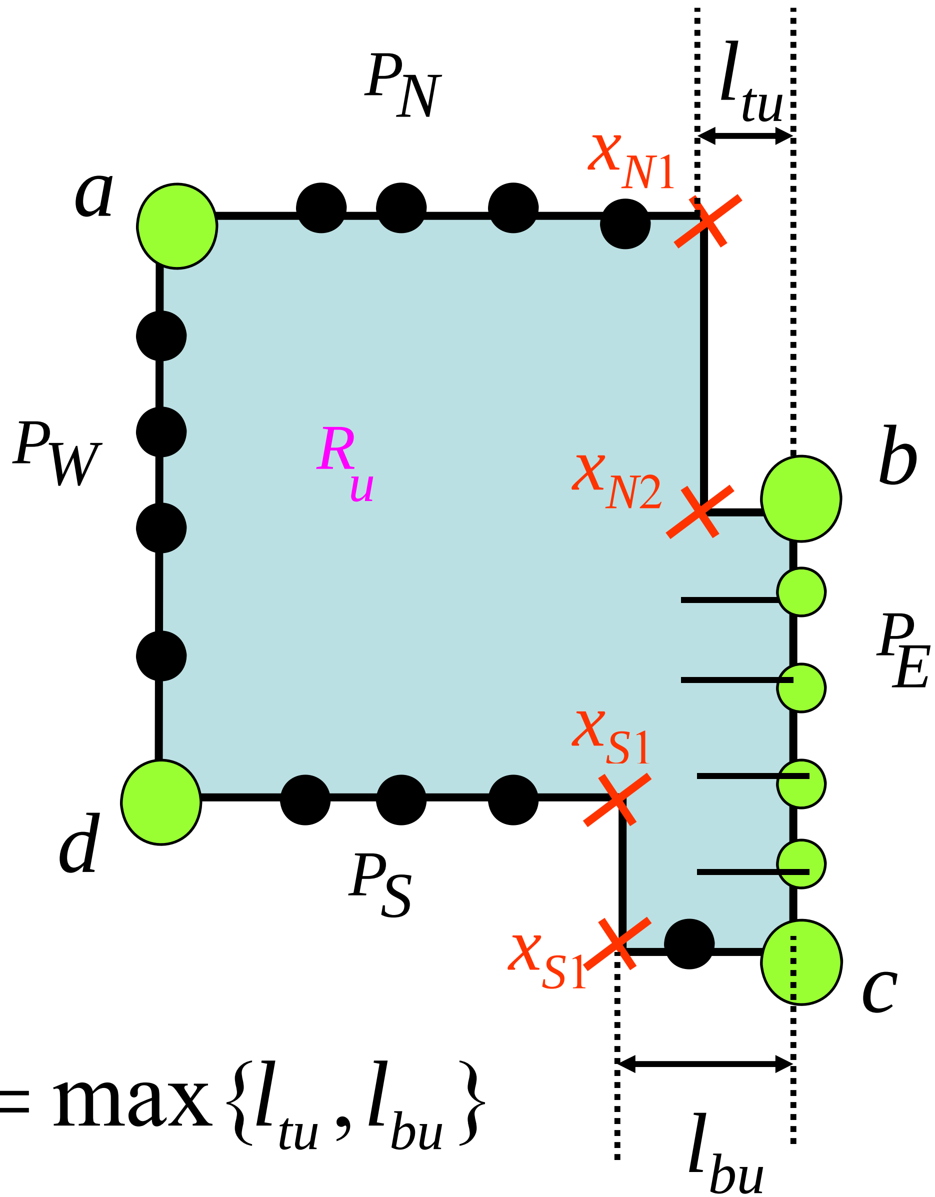
- Overall time complexity is linear.

Conclusion

We have presented a linear algorithm for prescribed area octagonal drawings of good slicing graphs.

Obtaining such an algorithm for larger classes of graphs is our future works.

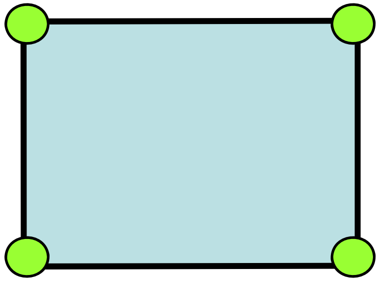
$l < f\,\delta$



$$l_u = \max\{l_{tu}, l_{bu}\}$$

If the octagon is a rectangle

$l_{tu} = 0$



$l_{bu} = 0$

$l_u = 0$

Feasible Octagon

R_u is a feasible octagon if R_u satisfies the following eight conditions

(i) $A(R_u) = A(G_u)$

(ii) $l_u < f\delta$

(iii) if x_{N2} is a convex corner then $l_{tu} \geq f_E^u \delta$

(iv) if x_{S1} is a convex corner then $l_{bu} \geq f_E^u \delta$

(v) if both x_{N2} and x_S then $l_{bu} - l_{tu} \geq f_E^u \delta$

(vi) if both x_{N1} and x_{S1} are concave corners

then $l_{tu} - l_{bu} \geq f_E^u \delta$

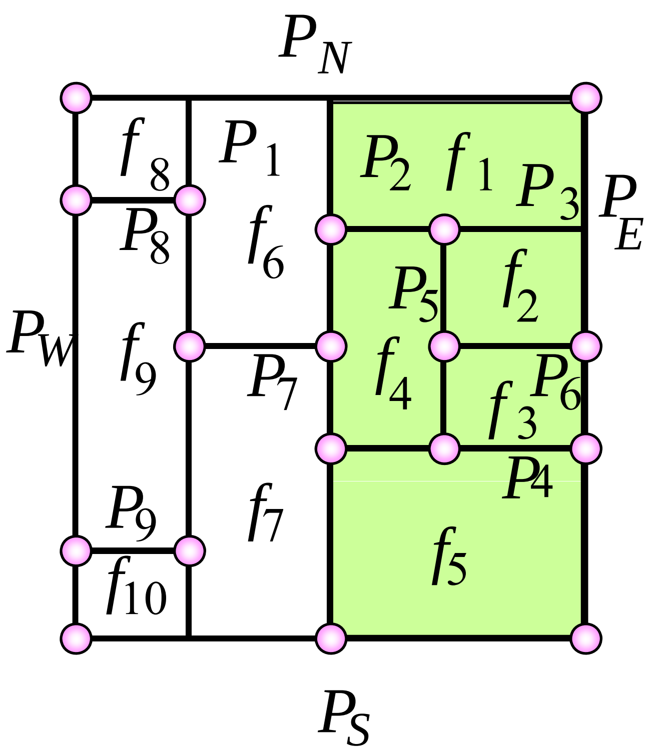
(vii) if x_{N2} is a concave corner

then $l_{tu} < (f - f_E^u)\delta$

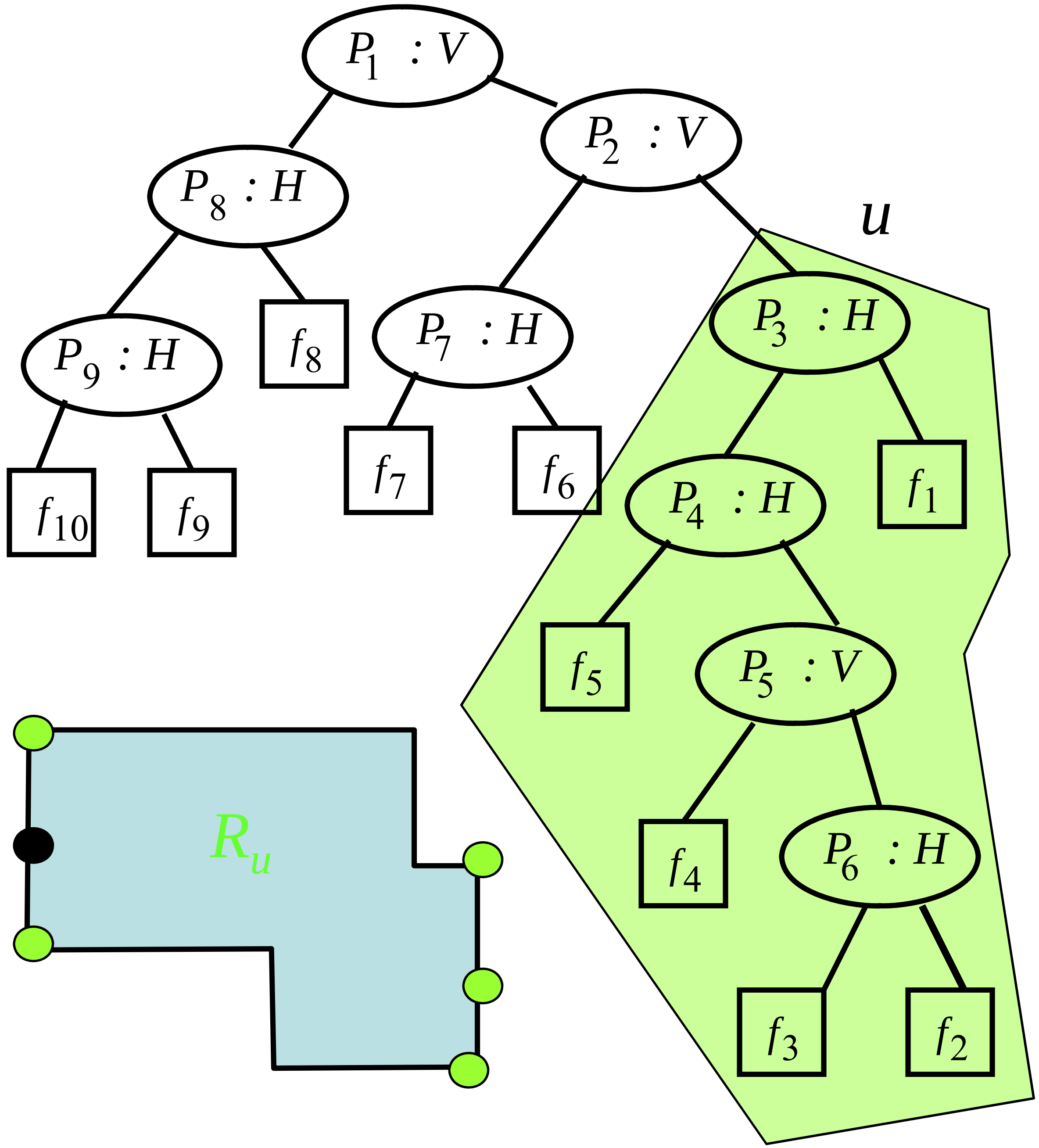
(viii) if x_{S1} is a concave corner

then $l_{bu} < (f - f_E^u)\delta$

(i) $A(R_u) = A(G_u)$



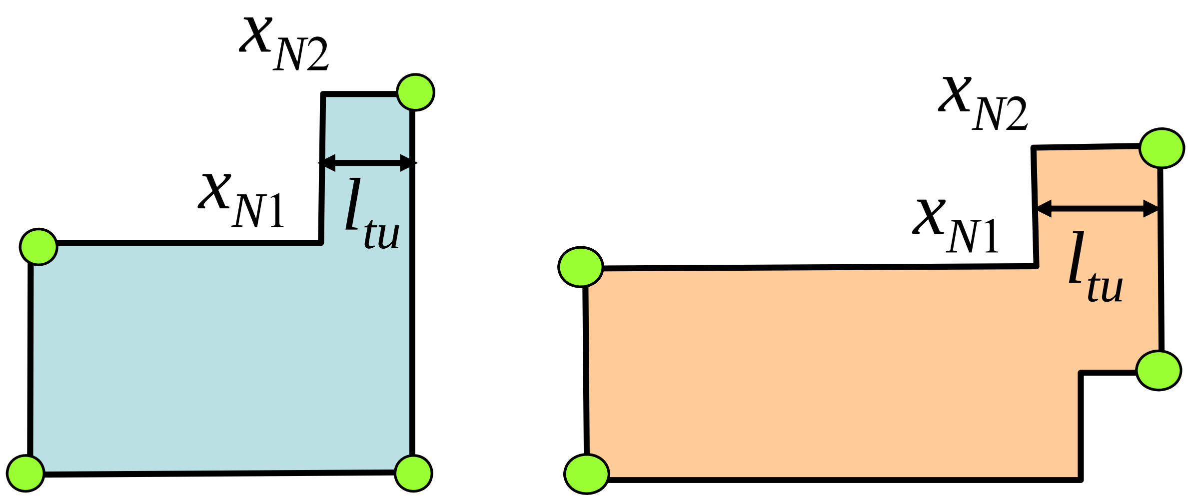
P_N
P_S
P_W
P_E
f_8 P_1 P_2 f_1 P_3
P_8 f_6
P_5 f_2
f_9 P_7 f_4 P_6
f_3
P_4
P_9 f_7
f_5
f_{10}



$P_1 : V$
$P_8 : H$
$P_2 : V$
u
$P_9 : H$
f_8
$P_7 : H$
$P_3 : H$
f_{10} f_9 f_7 f_6
$P_4 : H$
f_1
f_5
$P_5 : V$
f_4
$P_6 : H$
f_3 f_2
R_u

(iii) if x_{N2} is a convex corner

then $$l_{tu} \geq f_E^u \delta$$

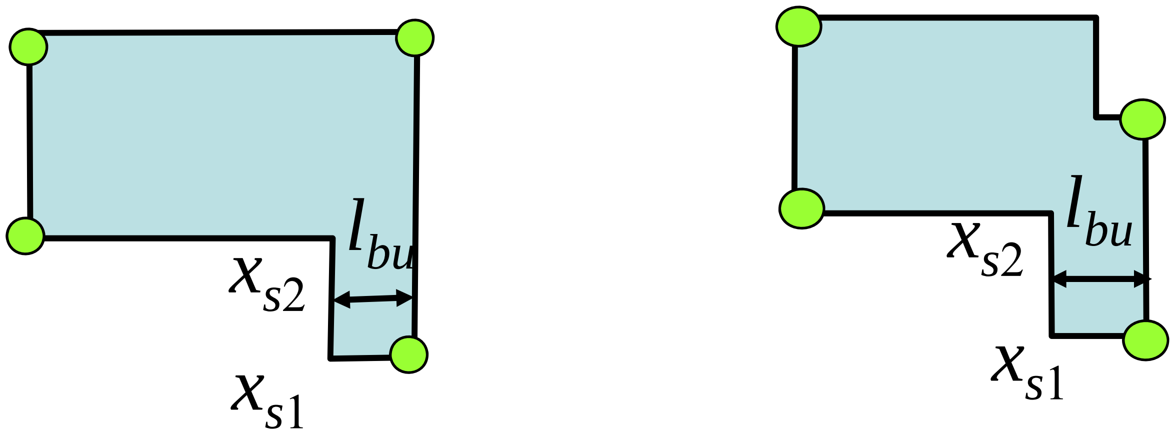


$$l_{tu} \geq \delta$$

for a facial octagon whose x_{N2} is convex.

(vi) if x_{S1} is a convex corner

then $$l_{bu} \geq f_E^u \delta$$

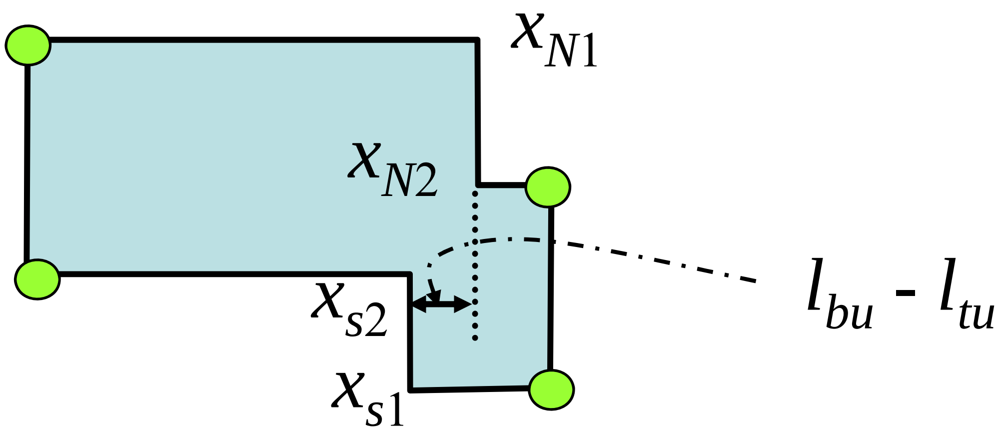


$$l_{bu} \geq \delta$$

for a facial octagon whose x_{S1} is convex.

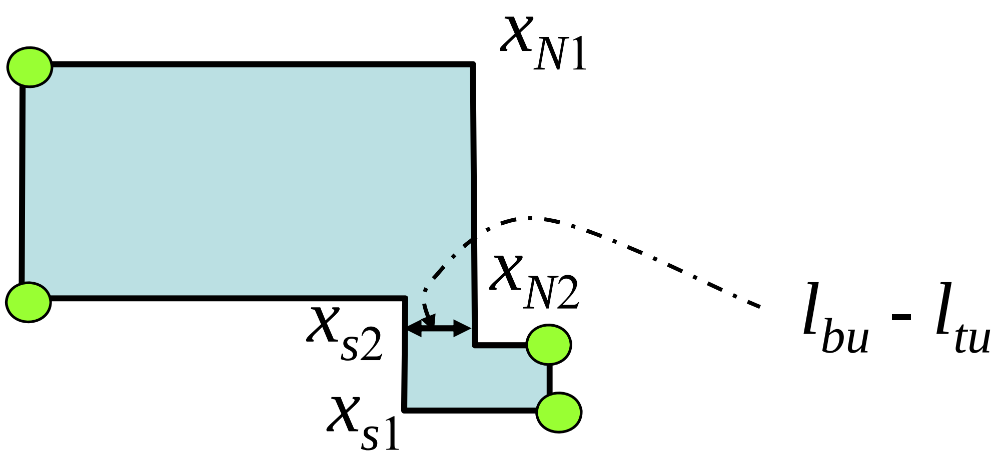
(v) if both x_{N2} and x_{S2} are concave corners

then $l_{bu} - l_{tu} \geq f_E^u \delta$



(v) if both x_{N2} and x_{S2} are concave corners

then $$l_{bu} - l_{tu} \geq f_E^u \delta$$

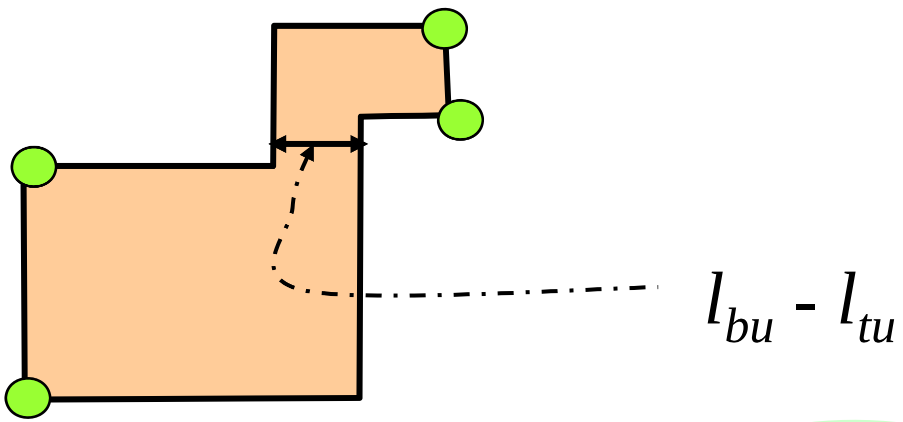


$$l_{bu} - l_{tu} \geq \delta$$

for a facial octagon whose x_{N2} and x_{S2} are concave

(vi) if both x_{N1} and x_{S1} are concave corners

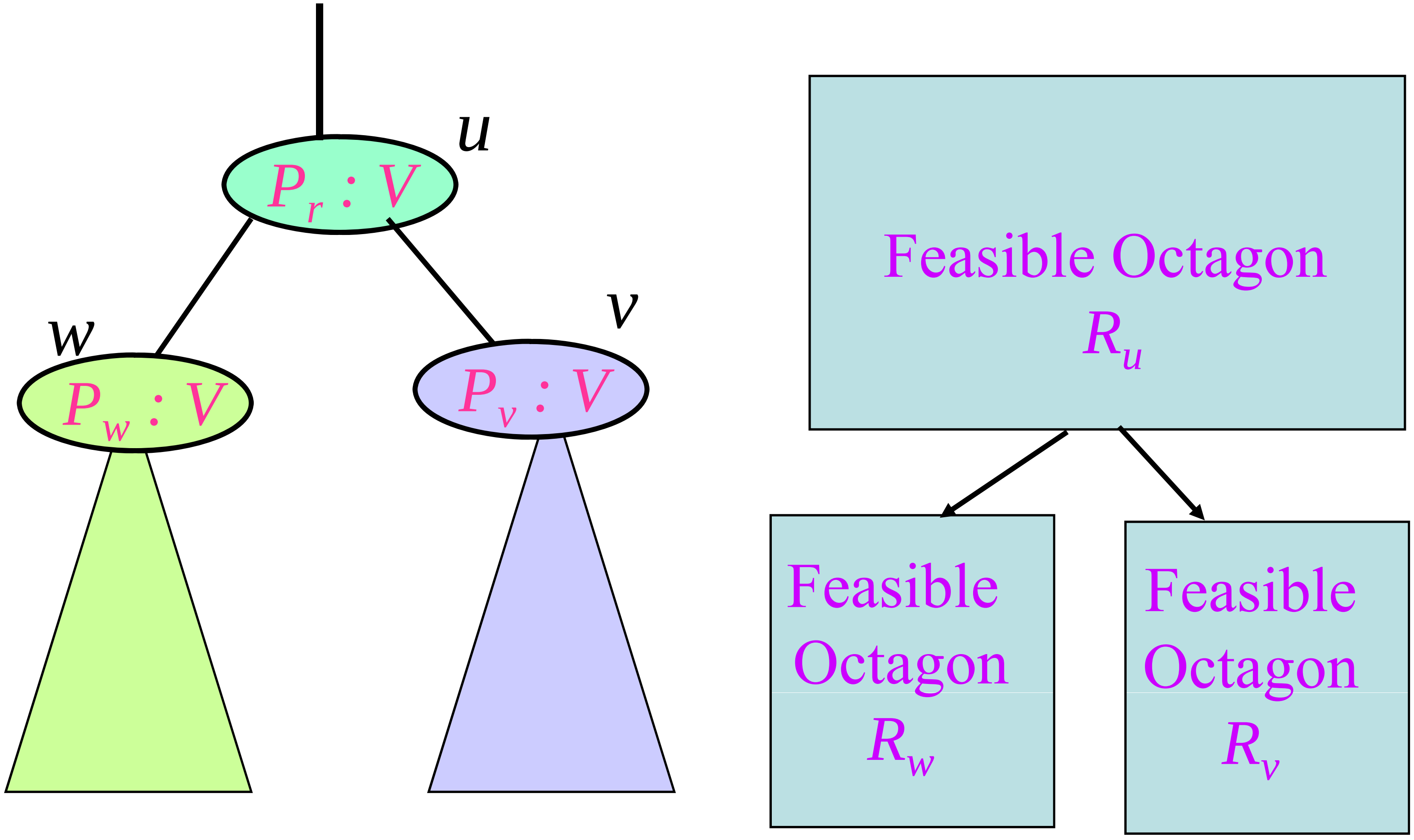
then $l_{tu} - l_{bu} \geq f_E^u \delta$



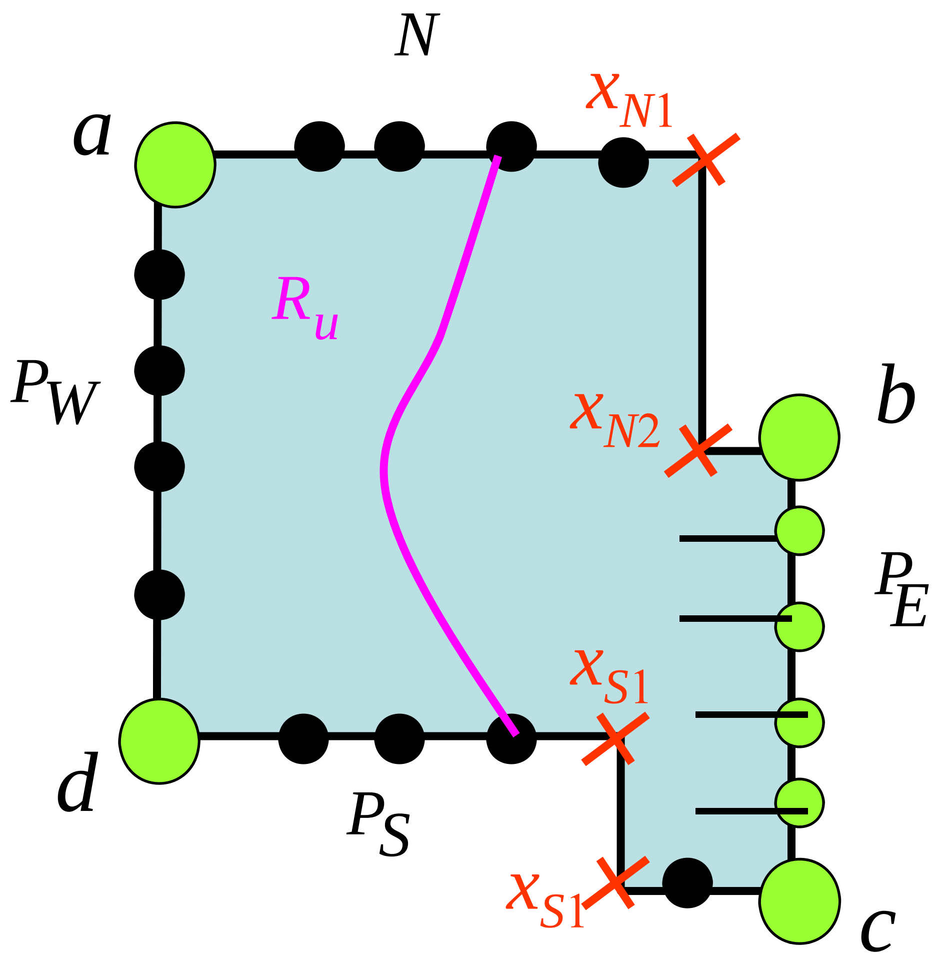
$l_{tu} - l_{bu} \geq \delta$

for a facial octagon whose x_{N1} and x_{S1} are concave

General computation at an internal node



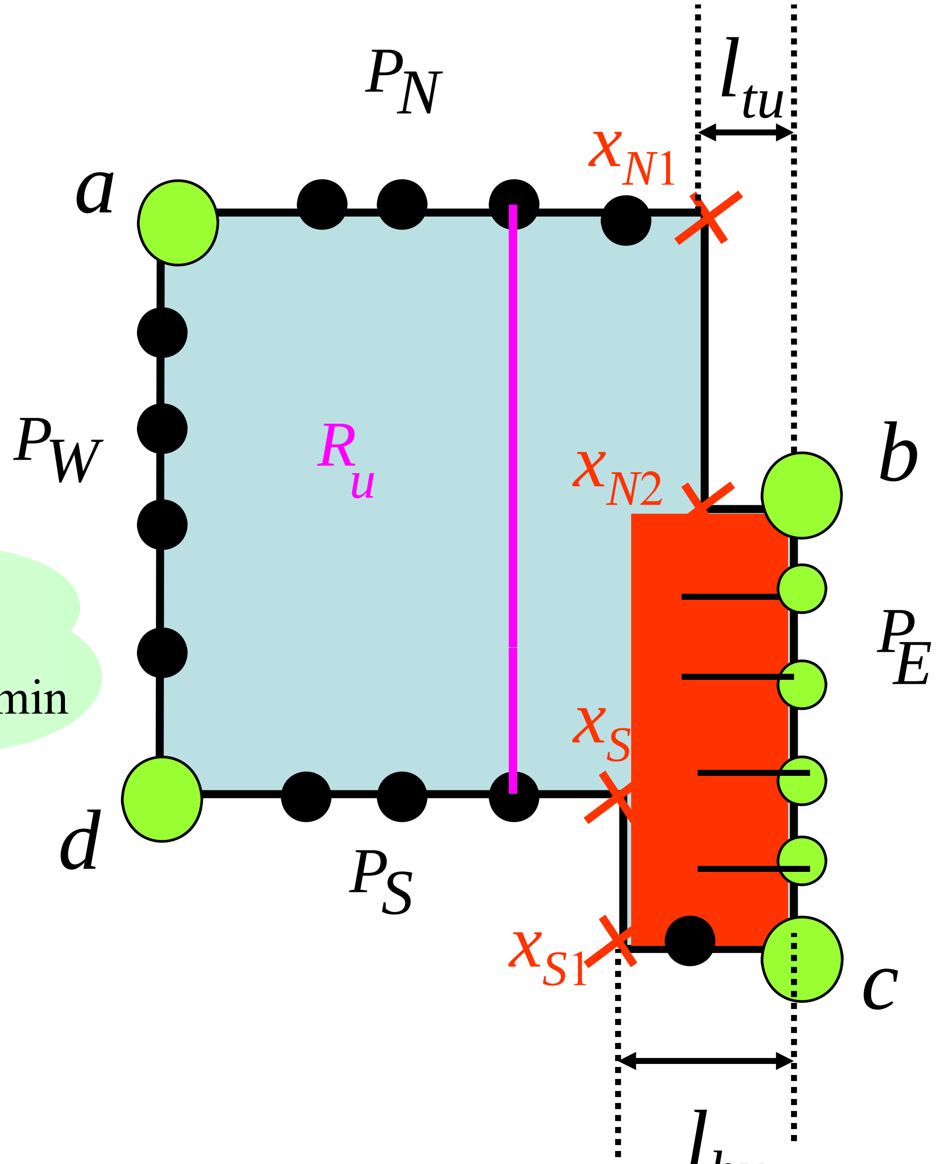
Embedding of a vertical slicing path



(ii) $l_u < f\,\delta$

Red area $< l_u H < f\,\delta H < A_{\min}$

The vertical slicing path is always embedded as a vertical line segment.

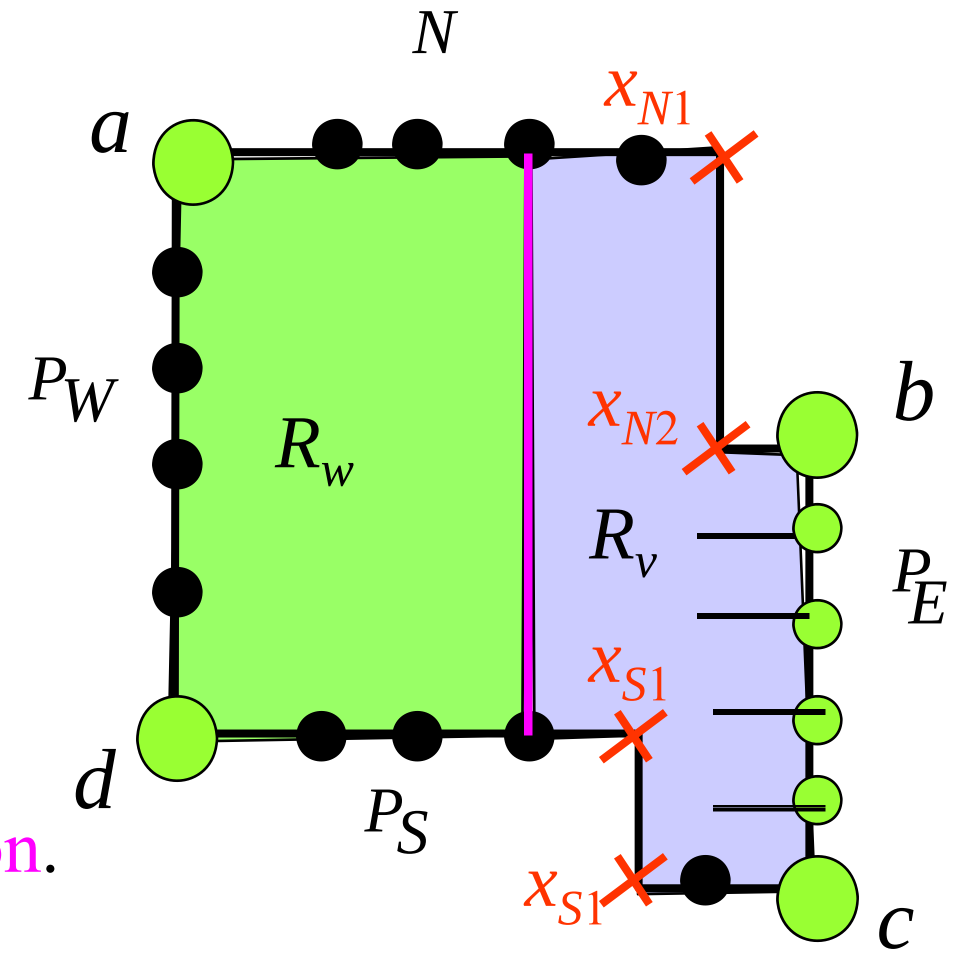


$$l_u = \max\{l_{tu}, l_{bu}\}$$

Embedding of a vertical slicing path

R_v is a feasible octagon since R_u is a feasible octagon.

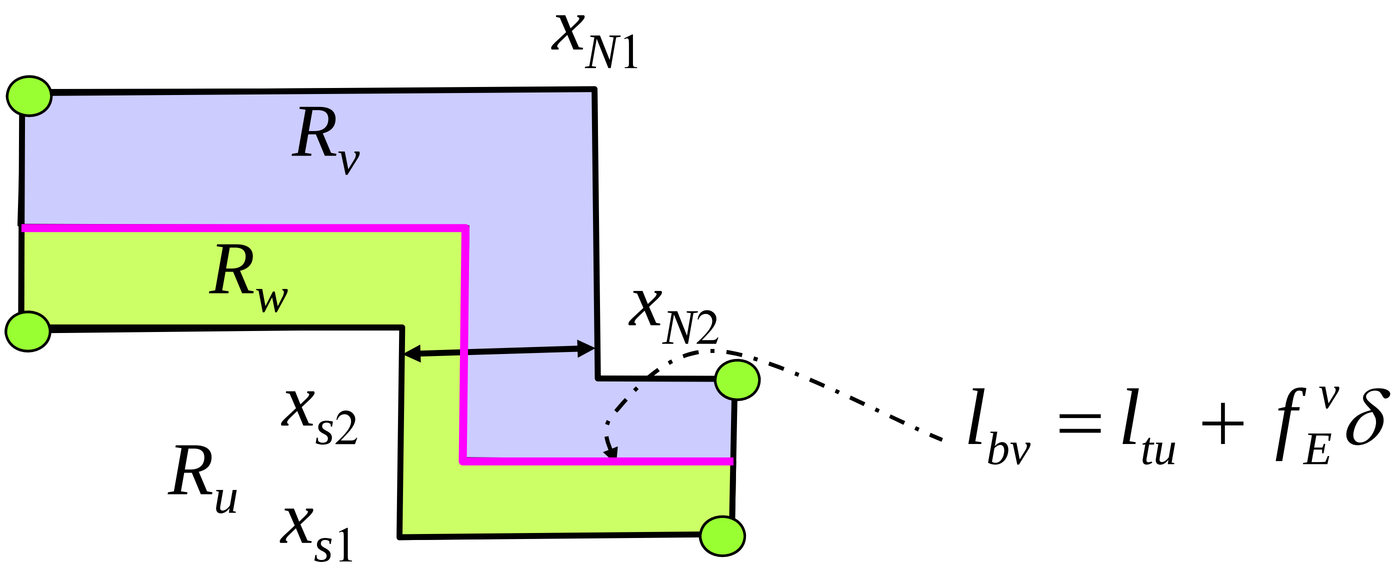
R_w is a rectange which is a feasible octagon.



Embedding of a horizontal slicing path

(v) if both x_{N2} and x_{S2} are concave corners

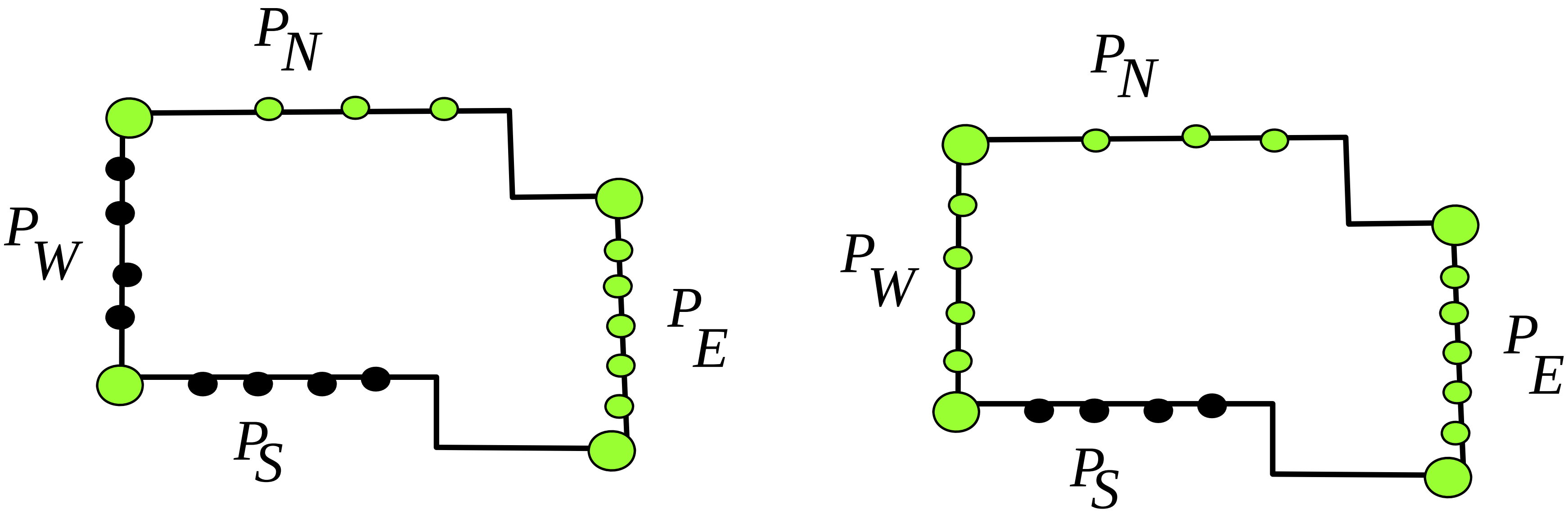
then $l_{bu} - l_{tu} \geq f_E^u \delta$



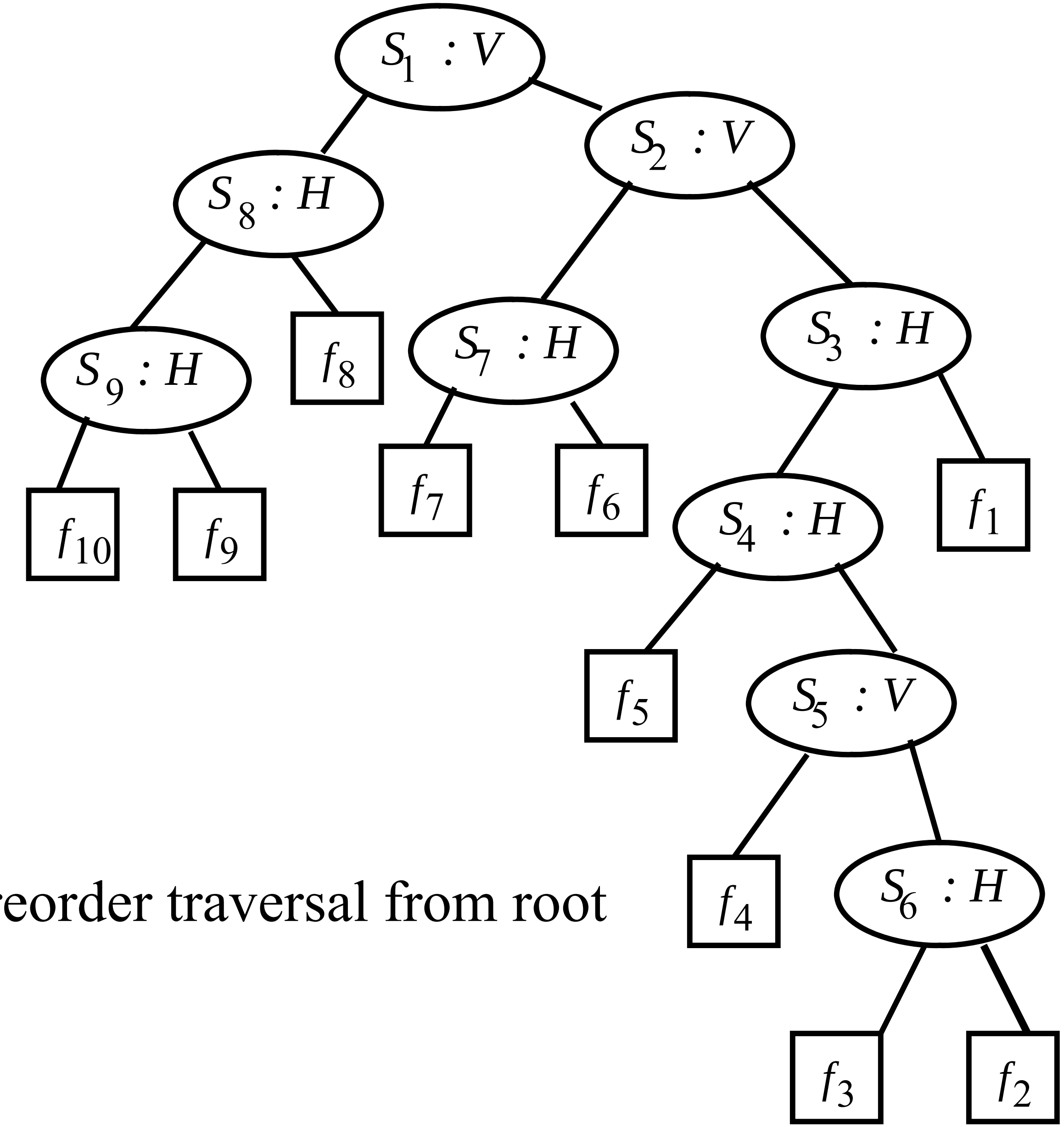
Both R_v and R_w are feasible octagons.

We can prove for other cases.

Computation at a leaf node x

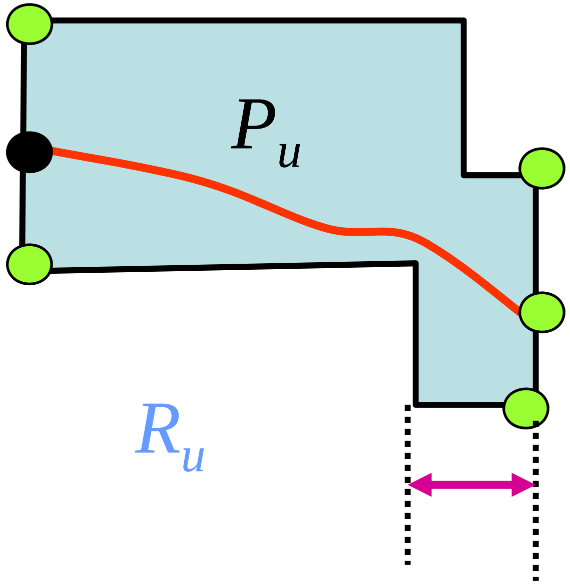


Algorithm Octagonal-Draw



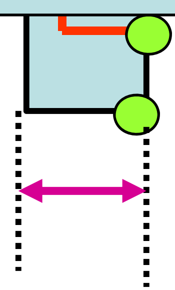
Reverse preorder traversal from root

Intuitively



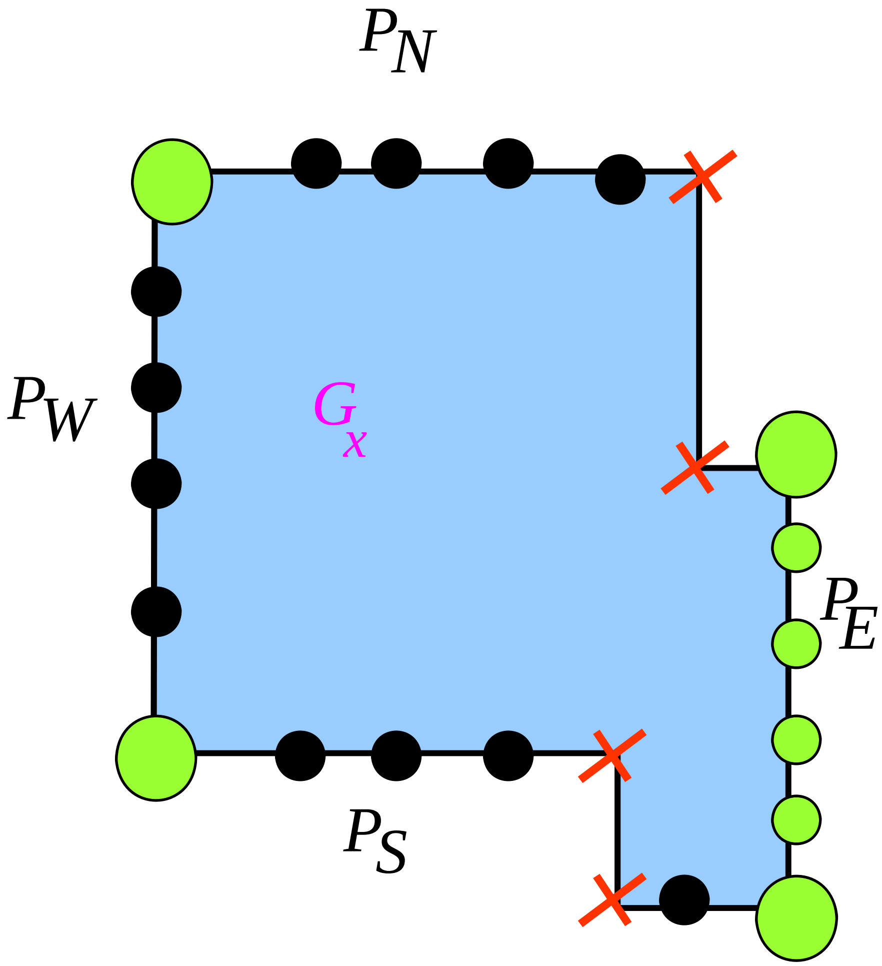
very small

We call R_u a **feasible octagon** if P_u can embedded successfully irrespective of size of $A(G_v)$.

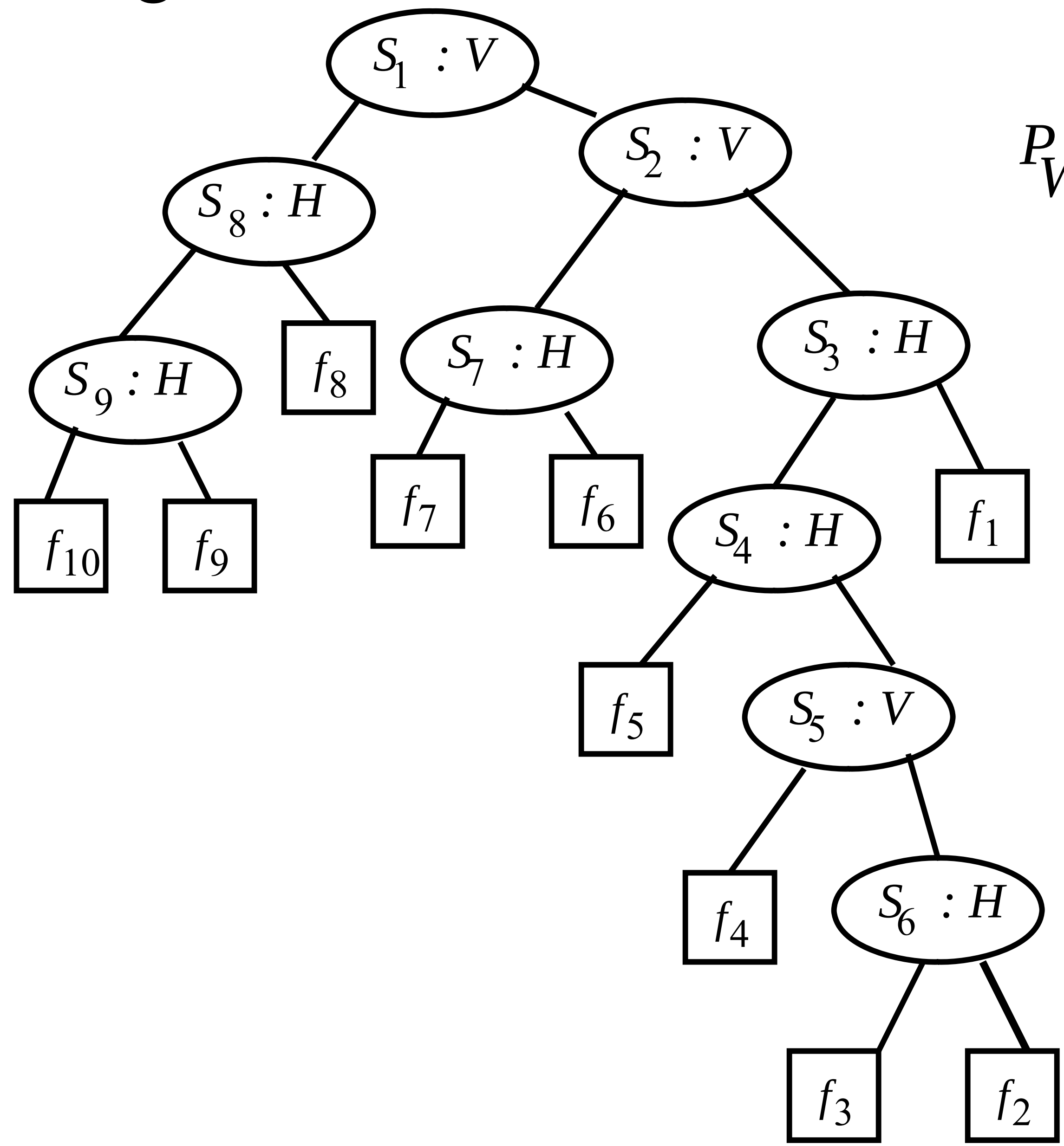


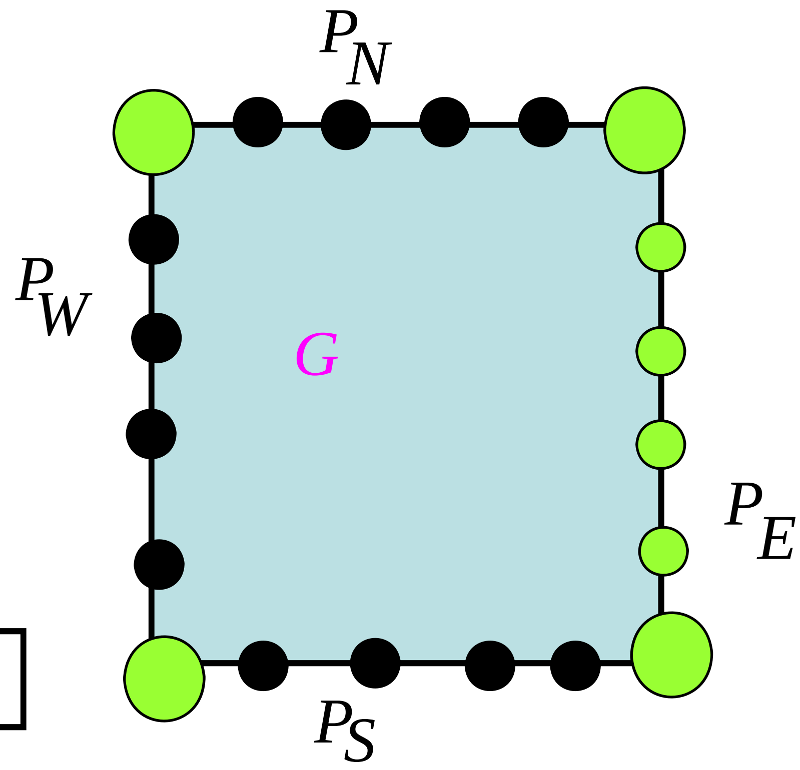
P_u can be embedded successfully although $A(G_v)$ is very large.

Dimensions of R_u play crucial roles.


P_N
P_W
G_x
P_E
P_S

Algorithm

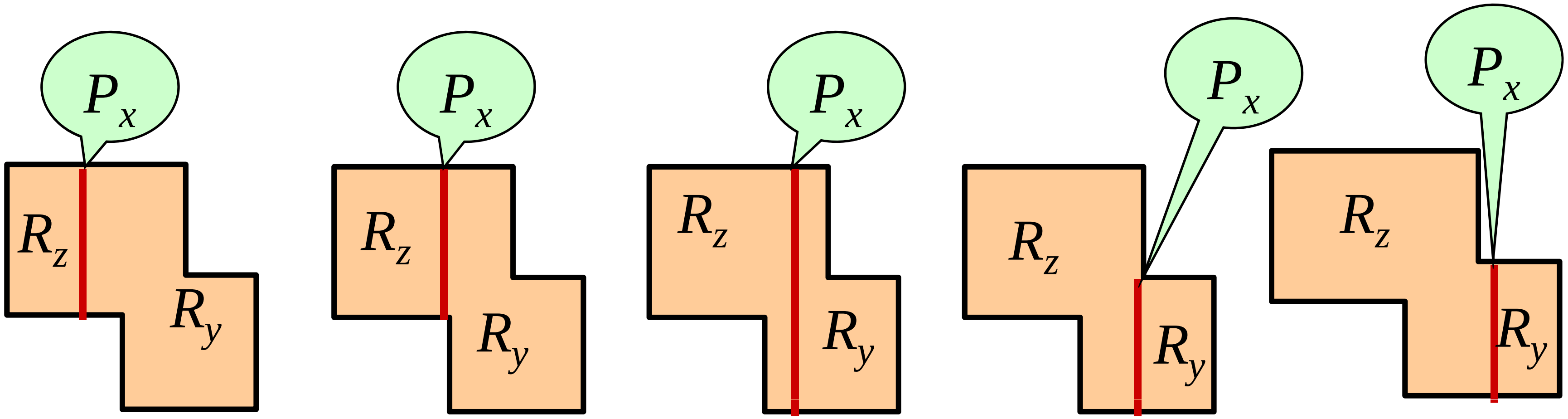




Input at root

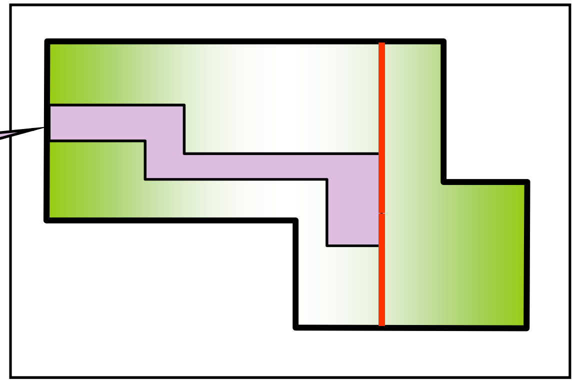
Computation at a V-node x

Let y be the right child of x and z be left child of x.

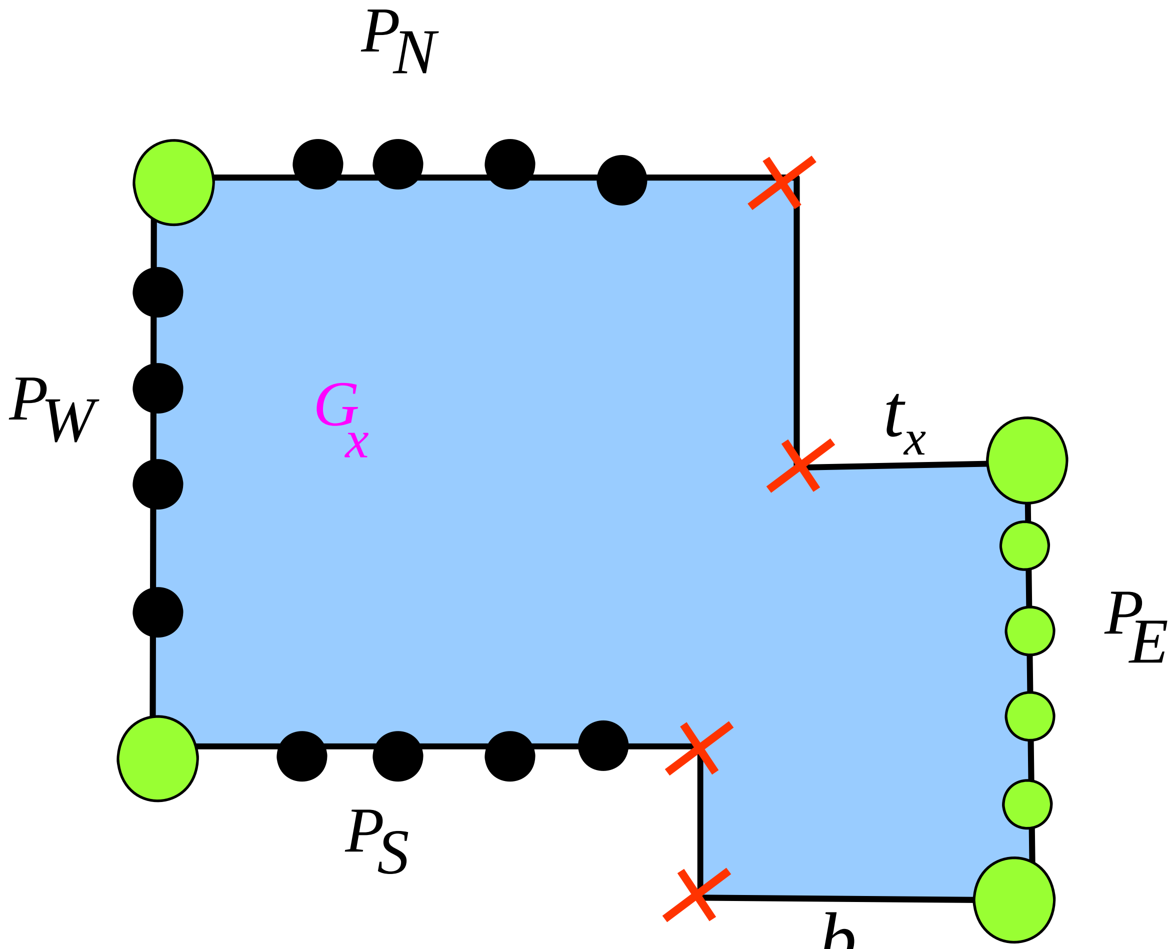


(a) (b) (c) (d) (e)

A face in R_z may need to be drawn with more than eight corners.

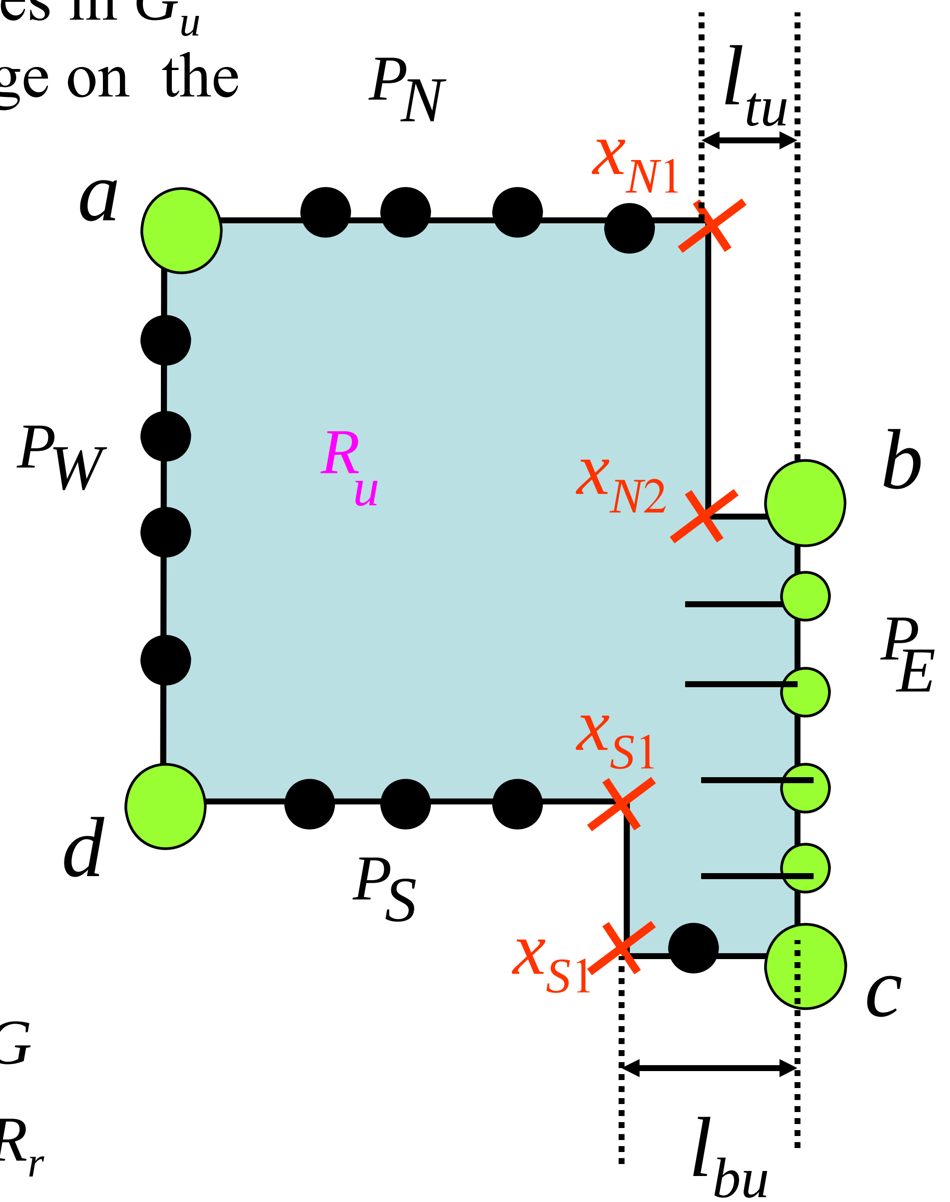


Foot Length l_x of an Octagon R_x



$$l_x = max\{t_x, b_x\}$$

f_E^f The number of inner faces in G_u each of which has an edge on the east side.



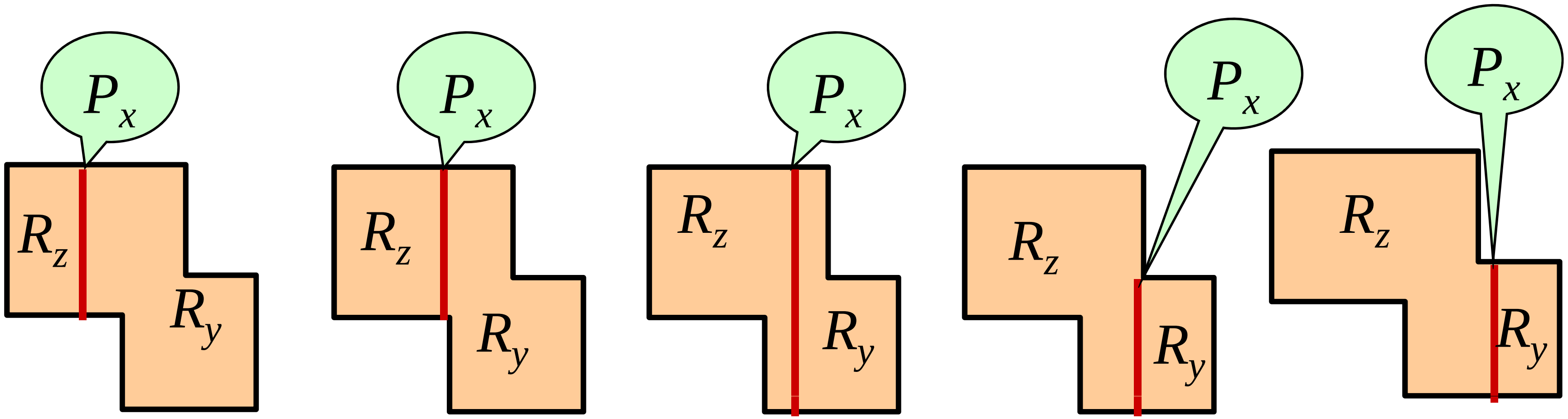
δ a positive constant.

$$0 \leq \delta \leq \frac{A_{\min}}{fH}$$

f number of inner faces in G

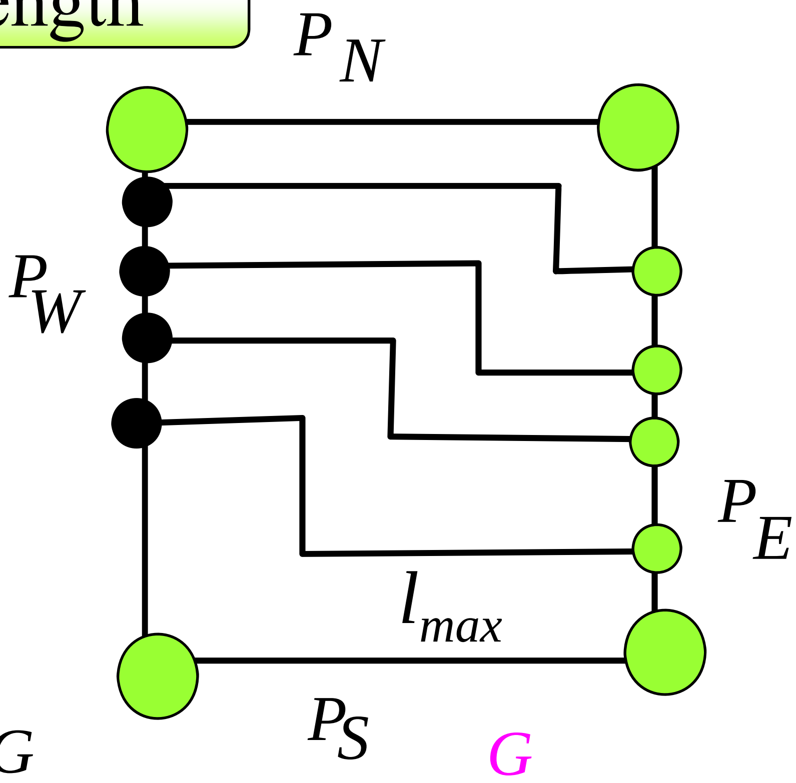
H height of initial rectangle R_r

If the footlength of R_x is resonably small then R_z will always be drawn as a rectngle.



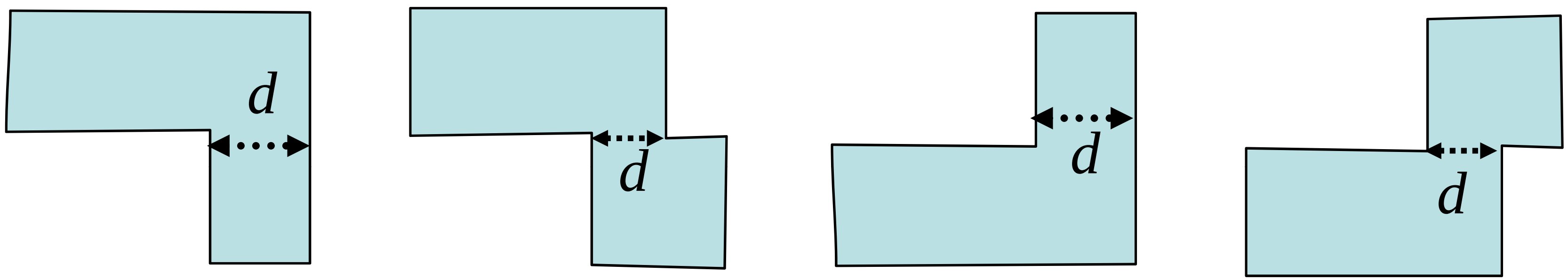
Maximum foot length

$$l_{\max} < fd$$



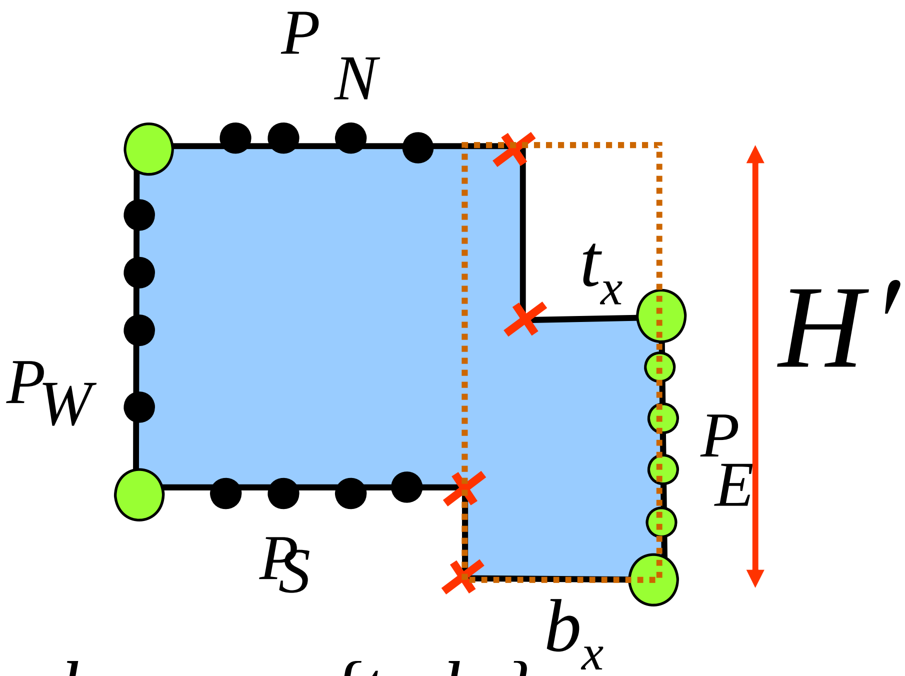
f : number of faces in G

Neck Length d of a Facial Octagon



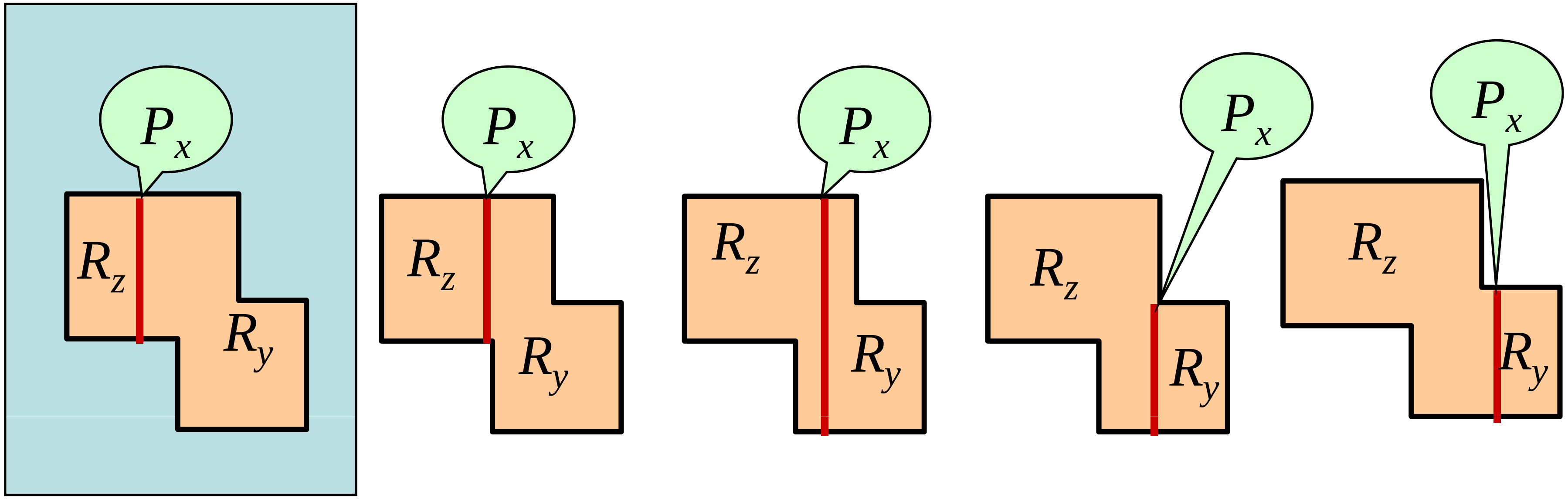
Estimating d

We fix d such that $$fdH \leq A_{\min}$$ holds.



$l_x = max\,\{t_x, b_x\}$

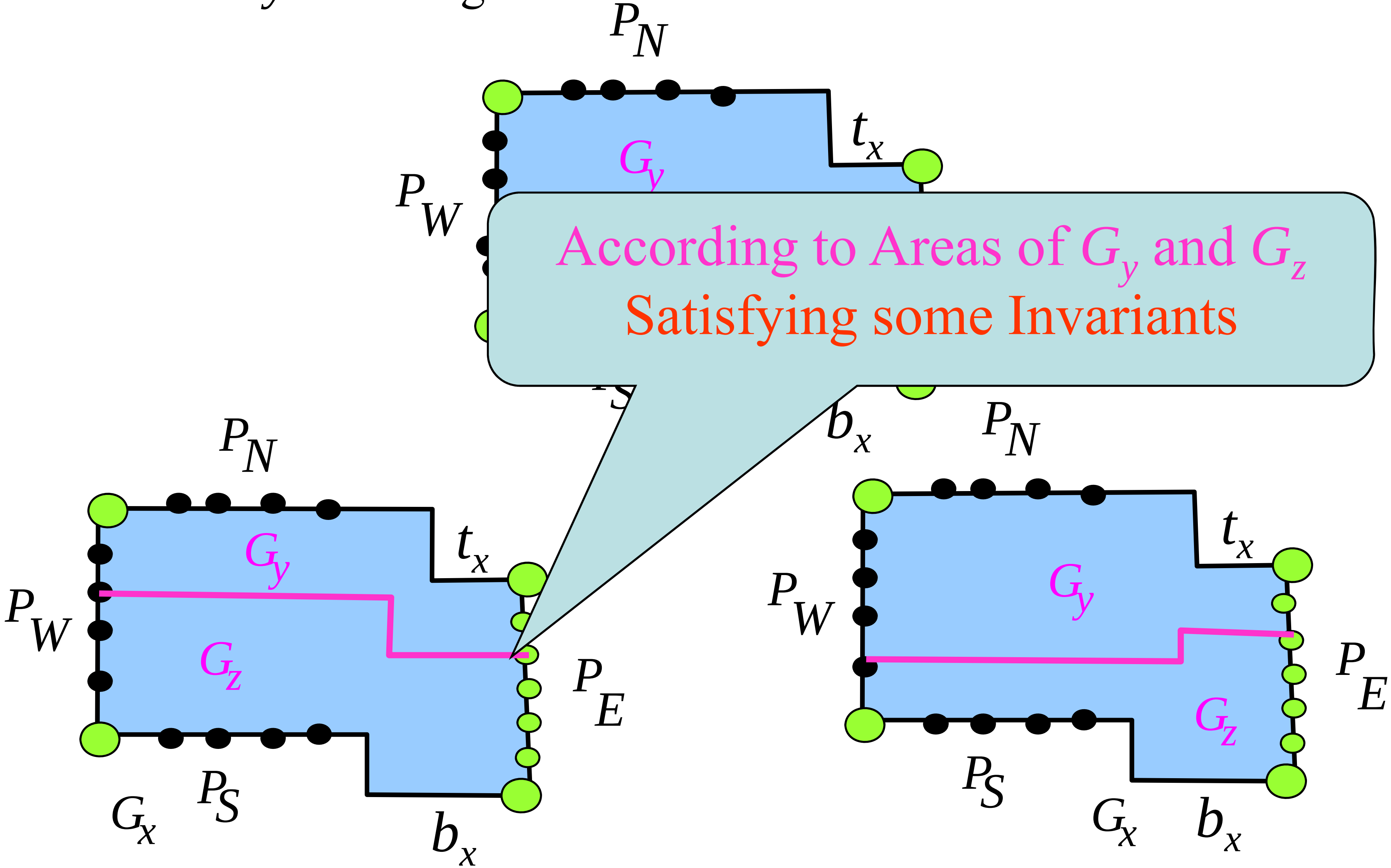
$$l_x H' < fdH' < fdH < A_{\min}$$



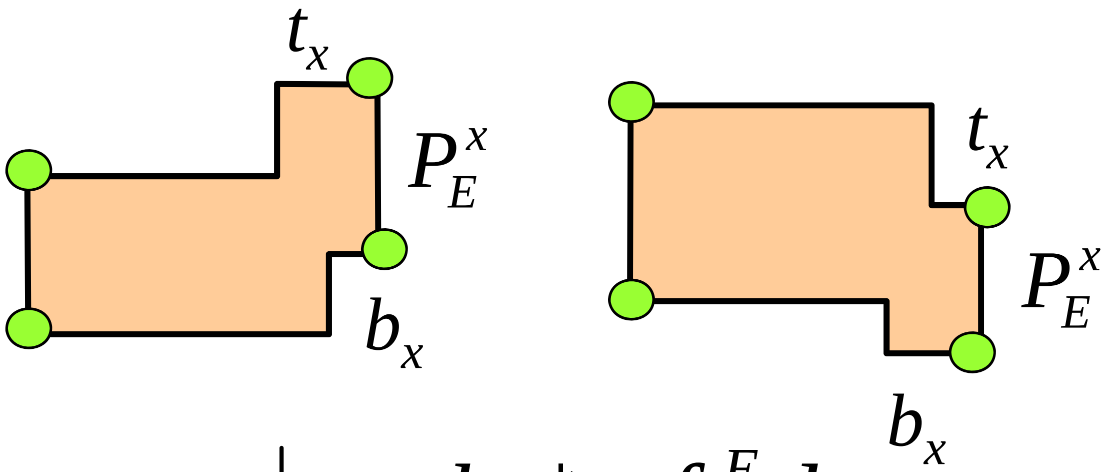
R_z is always a rectangle.

Computation at a H-node x

Let y be the right child of x and z be left child of x.



Invariants

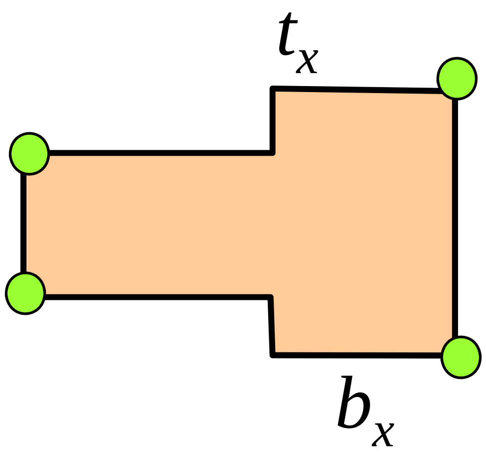


$$|t_x - b_x| \geq f_x^E d$$

f_x^E number of faces in G_x having an edge on P_E^x

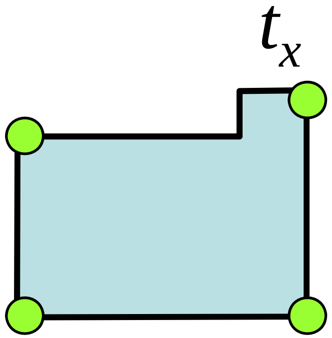
Invariants

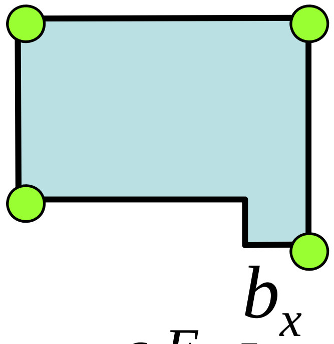
$$t_x \geq f_x^E d$$



$$b_x \geq f_x^E d$$

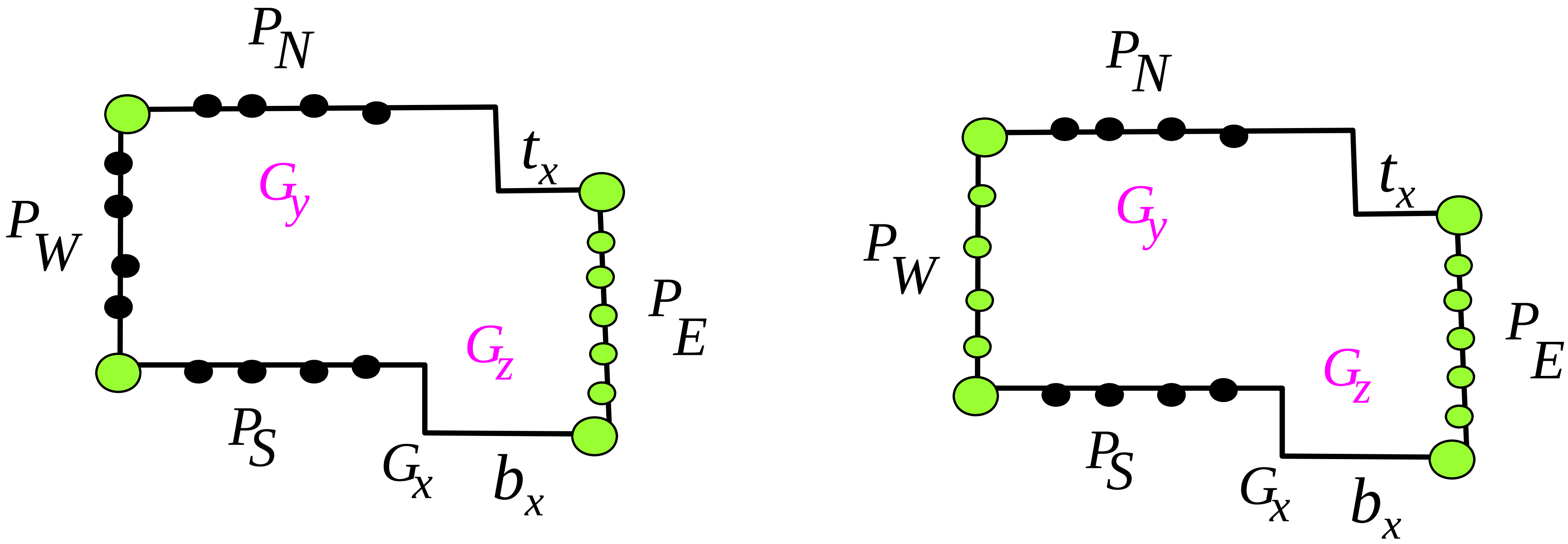
$$t_x \geq f_x^E d$$





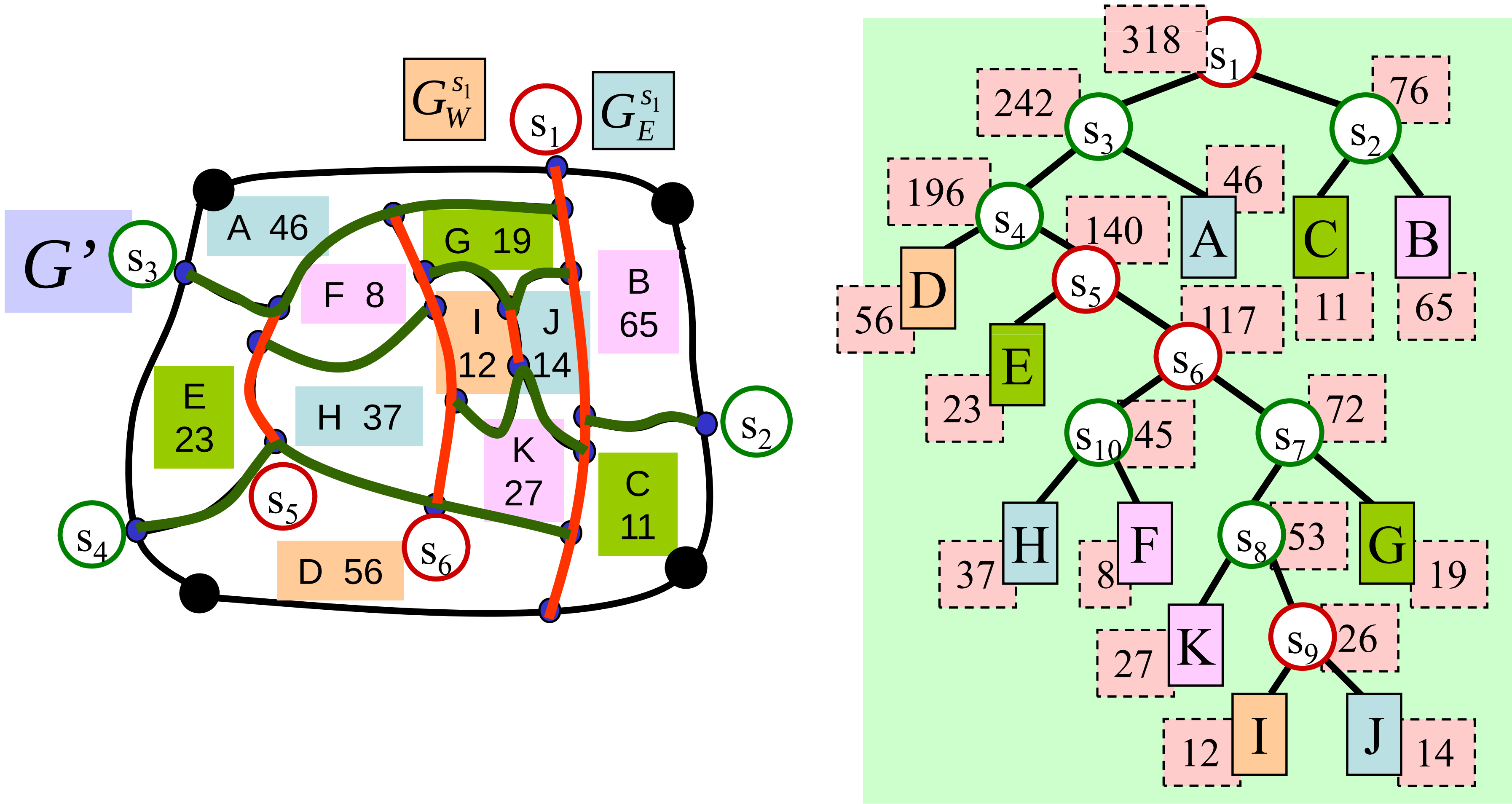
$$b_x \geq f_x^E d$$

Computation at a leaf node x



Time Complexity

- Using a bottom-up computation on slicing tree, area of subgraphs for all internal nodes can be computed in linear time.

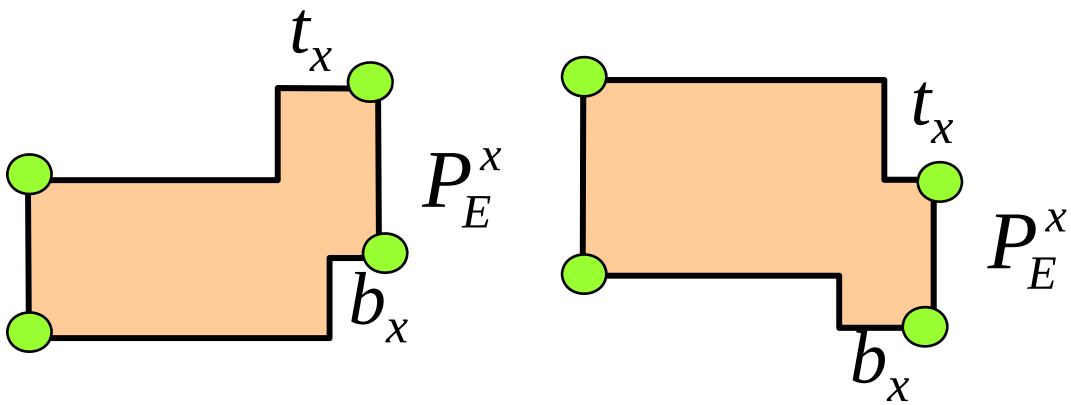


G'
$G_W^{s_1}$
$G_E^{s_1}$
s_1
s_2
s_3
s_4
s_5
s_6
A 46
F 8
G 19
B 65
I 12
J 14
E 23
H 37
K 27
C 11
D 56
318
242
76
196
46
140
117
11
65
56
23
45
72
37
8
53
19
27
26
12
14
s_7
s_8
s_9
s_{10}

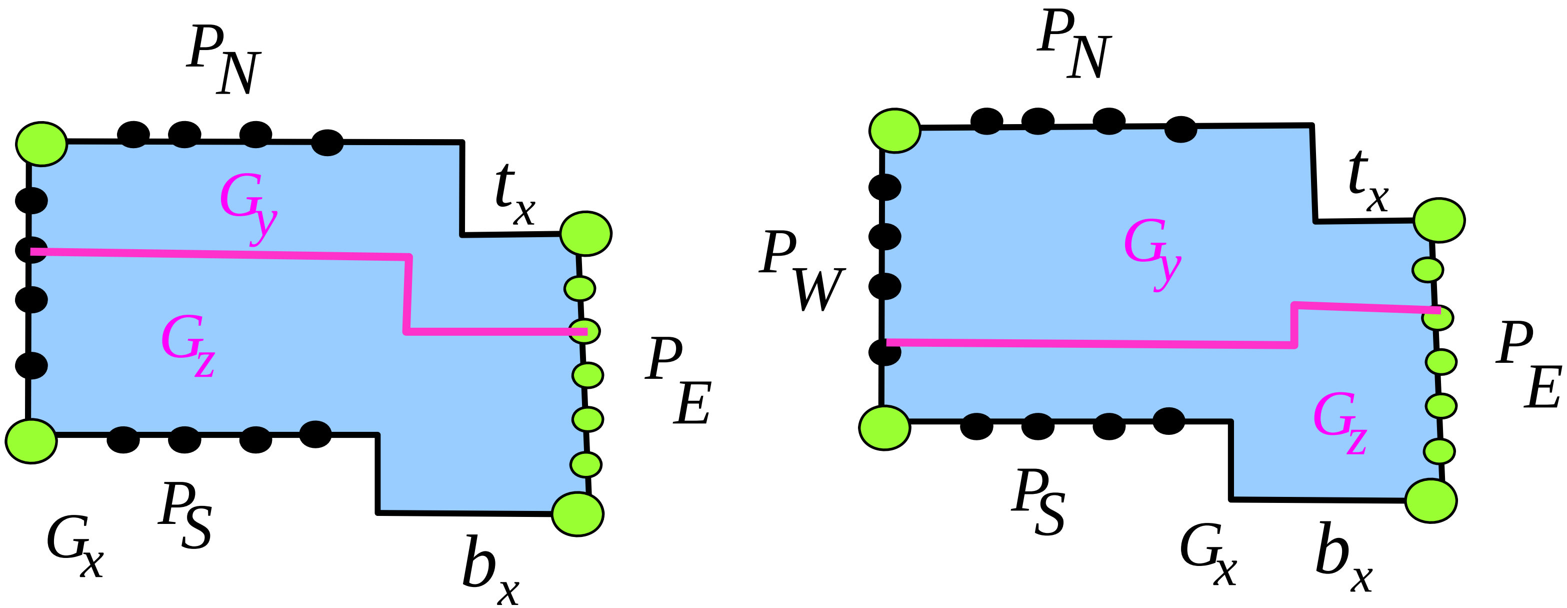
Time Complexity

- Using a bottom-up computation on slicing tree, area of subgraphs for all internal nodes can be computed in linear time.

- With an O(n) time preprocessing, embedding of the slicing path at each internal node takes constant time.



f_x^{E} number of faces in G_x having an edge on P_E^x



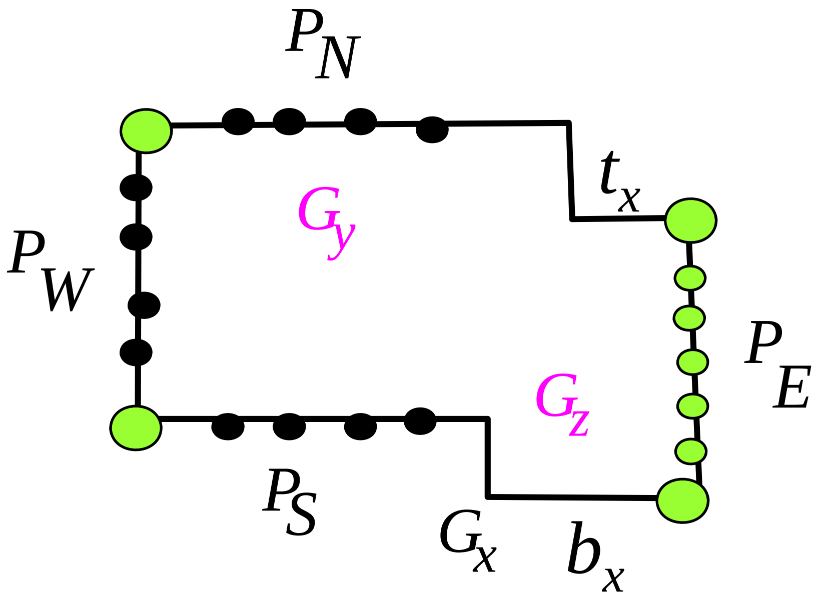
Time Complexity

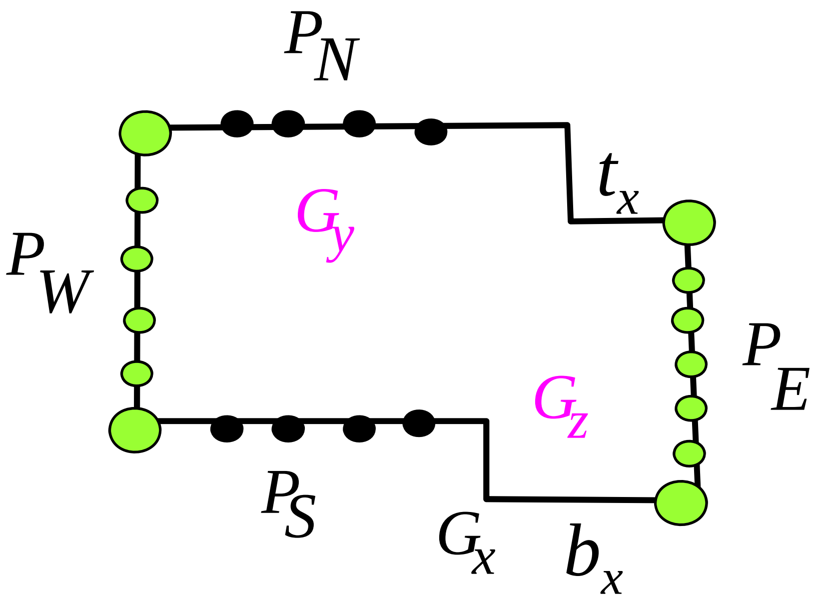
- Using a bottom-up computation on slicing tree, area of subgraphs for all internal nodes can be computed in linear time.

- With an O(n) time preprocessing, embedding of the slicing path at each internal node takes constant time.

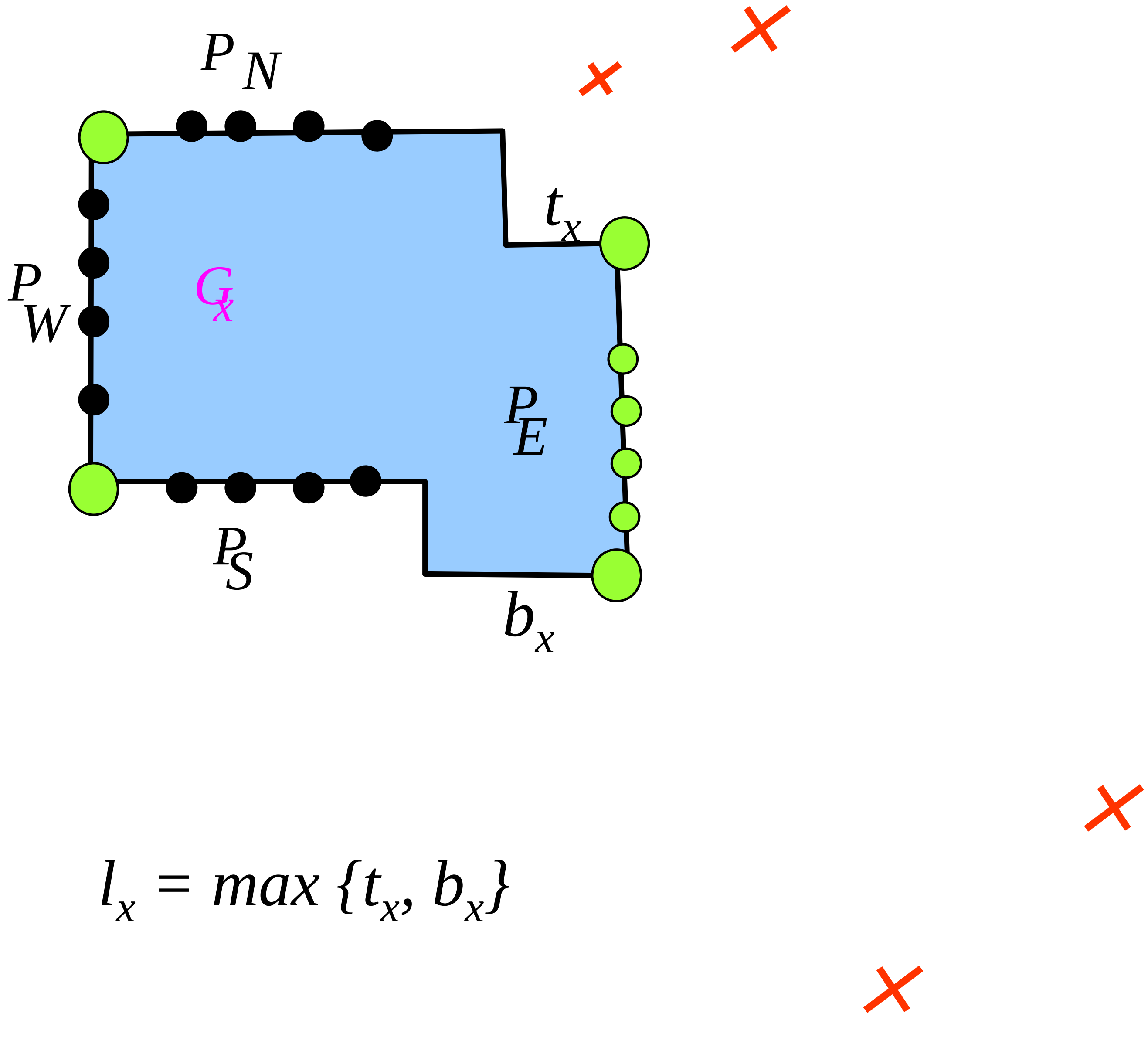
- Computation time at a leaf node is proportional to the number of non-corner vertices on the west side of the face.

Computation at a leaf node x

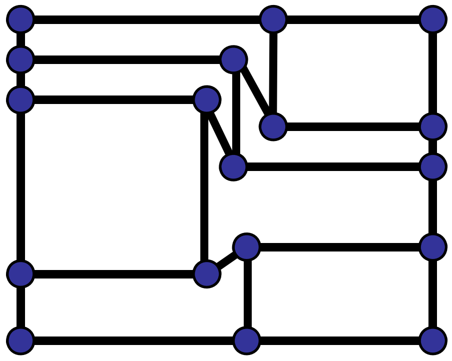


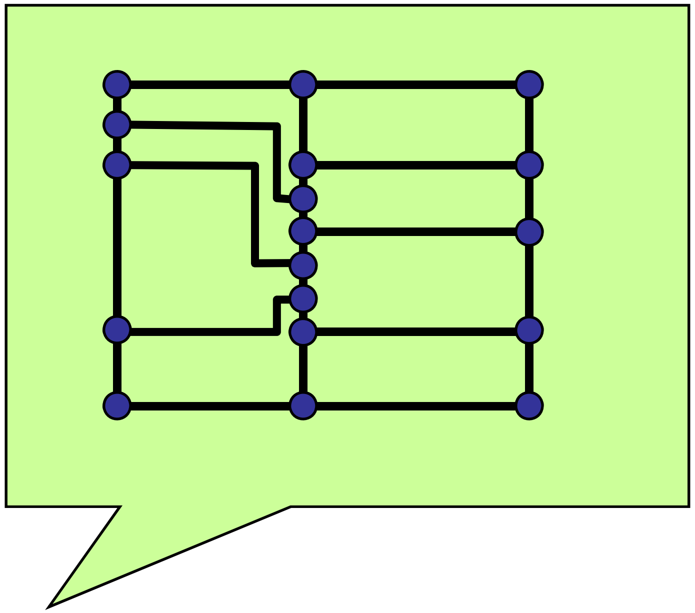


Foot Length l_x of an Octagon R_x



$$l_x = max\,\{t_x, b_x\}$$





According to Areas of G_y and G_z

Satisfying Invariant

A Linear Algorithm for Prescribed-Area Octagonal Drawings of Plane Graphs

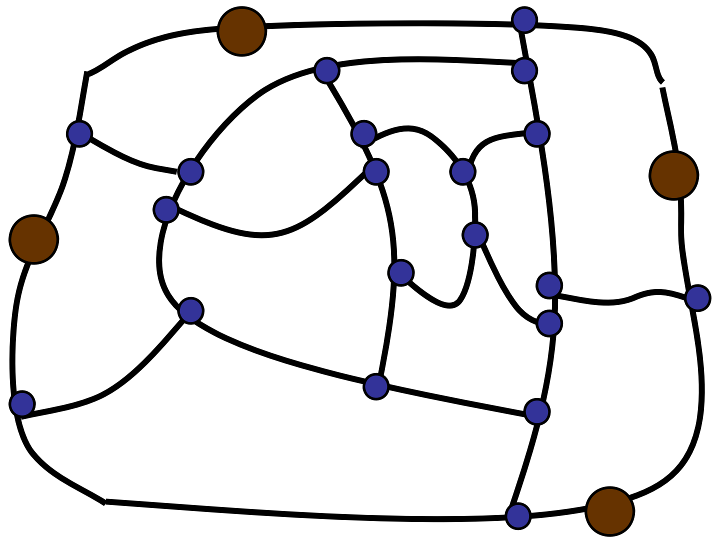
Md. Saidur Rahman
Kazuyuki Miura
Takao Nishizeki

TOHOKU UNIVERSITY

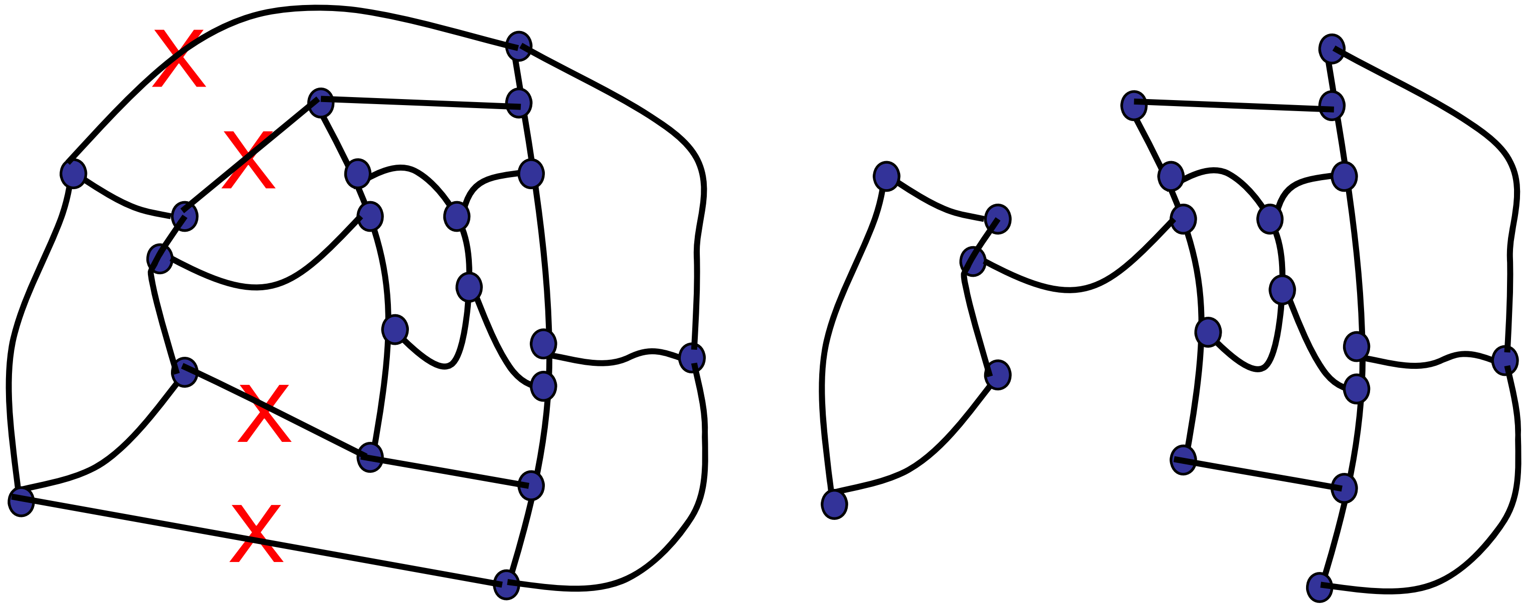
Previous works on prescribed-area drawing

Thomassen, 1992

G: Obtained from cyclically 5-edge connected plane cubic graphs by inserting four vertices of degree 2 on outer face.

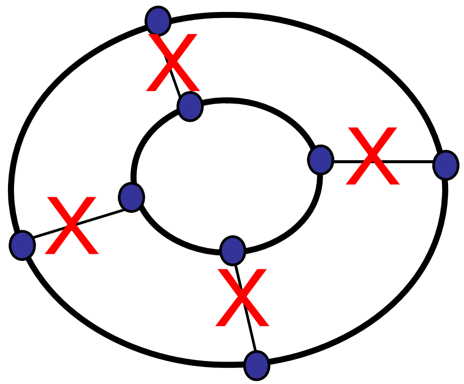


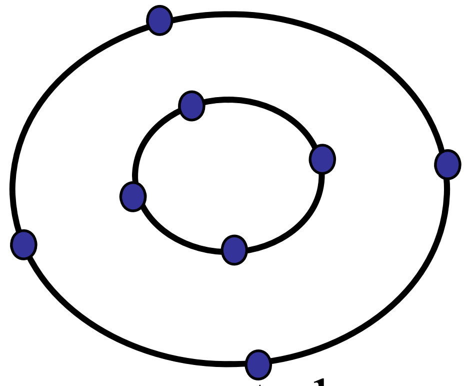
Cyclically 5-edge Connected Cubic Plane Graphs



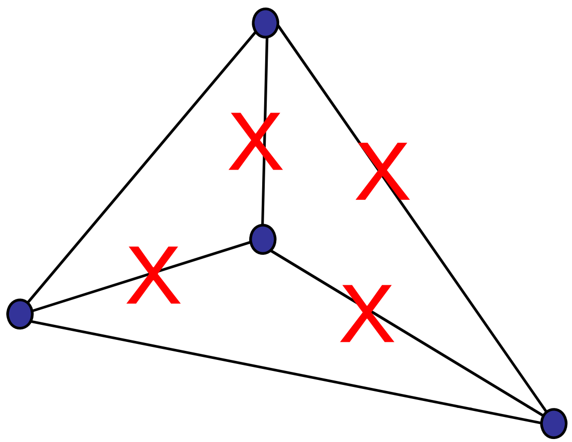
Removal of any set of less than 5-edges leaves a graph such that exactly one of the connected components has a cycle.

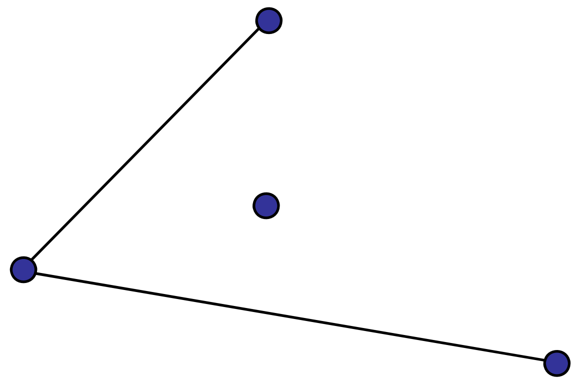
Not cyclically 5-edge connected





Two connected components having cycles.



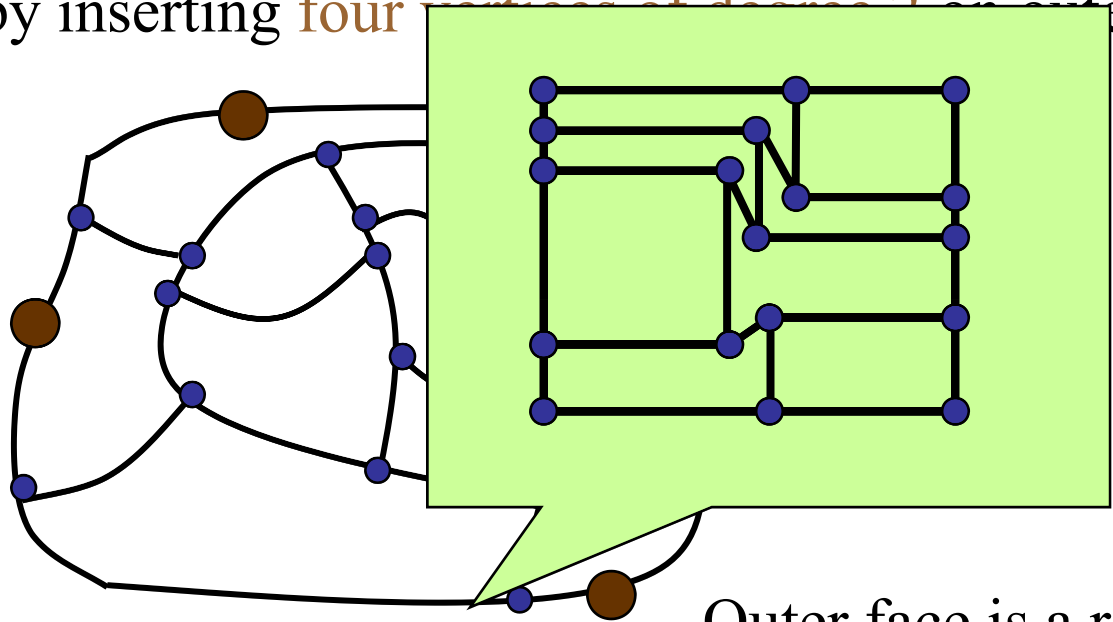


No connected component has a cycle.

Previous works on prescribed area drawing

Thomassen, 1992

G: Obtained from cyclically 5-edge connected plane cubic graphs by inserting four vertices of degree 2 on outer face.



Outer face is a rectangle

Inner faces are arbitrary polygons

G has a prescribed area straight-line drawing.

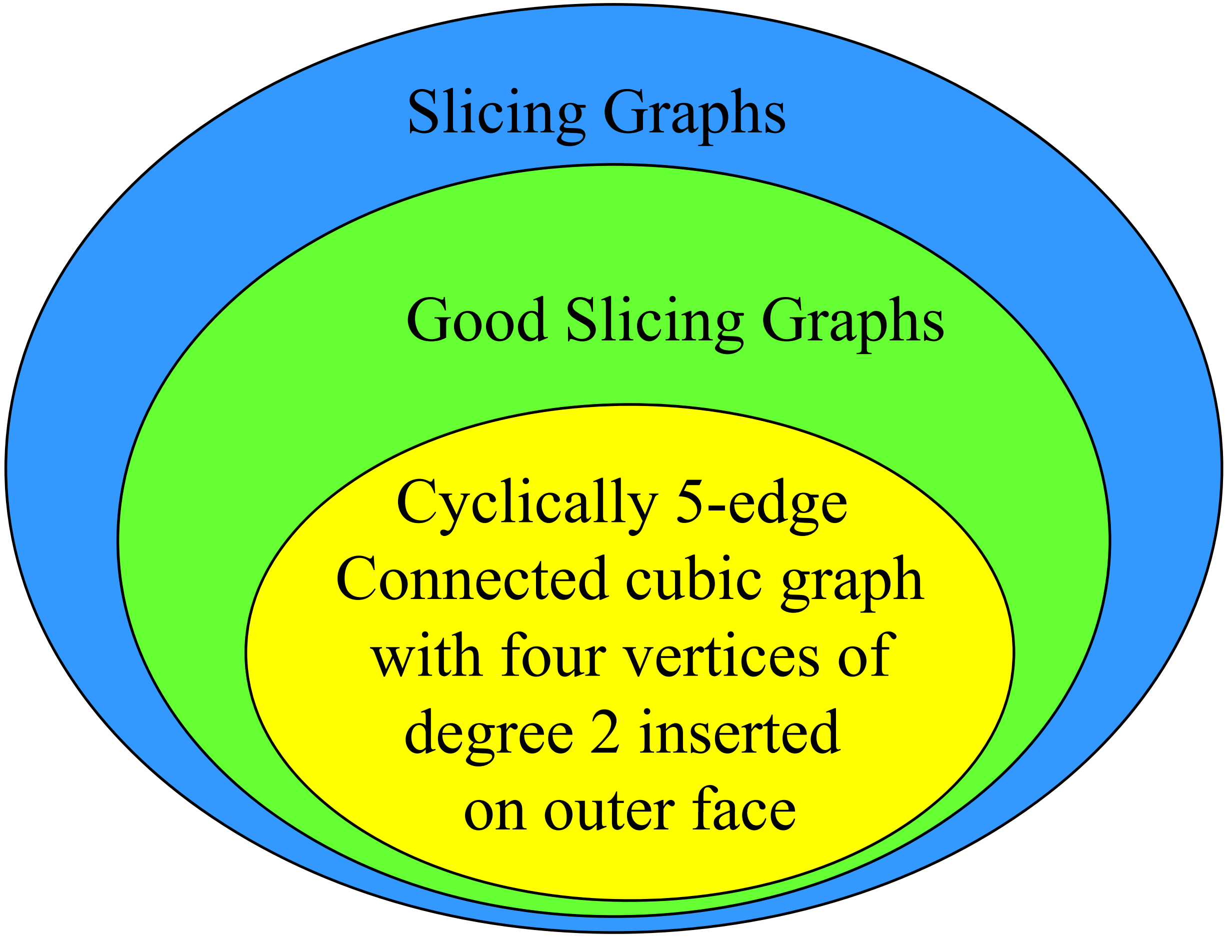
An inner face is not always drawn as a rectilinear polygon.

$O(n^3)$ time algorithm.

Theorem

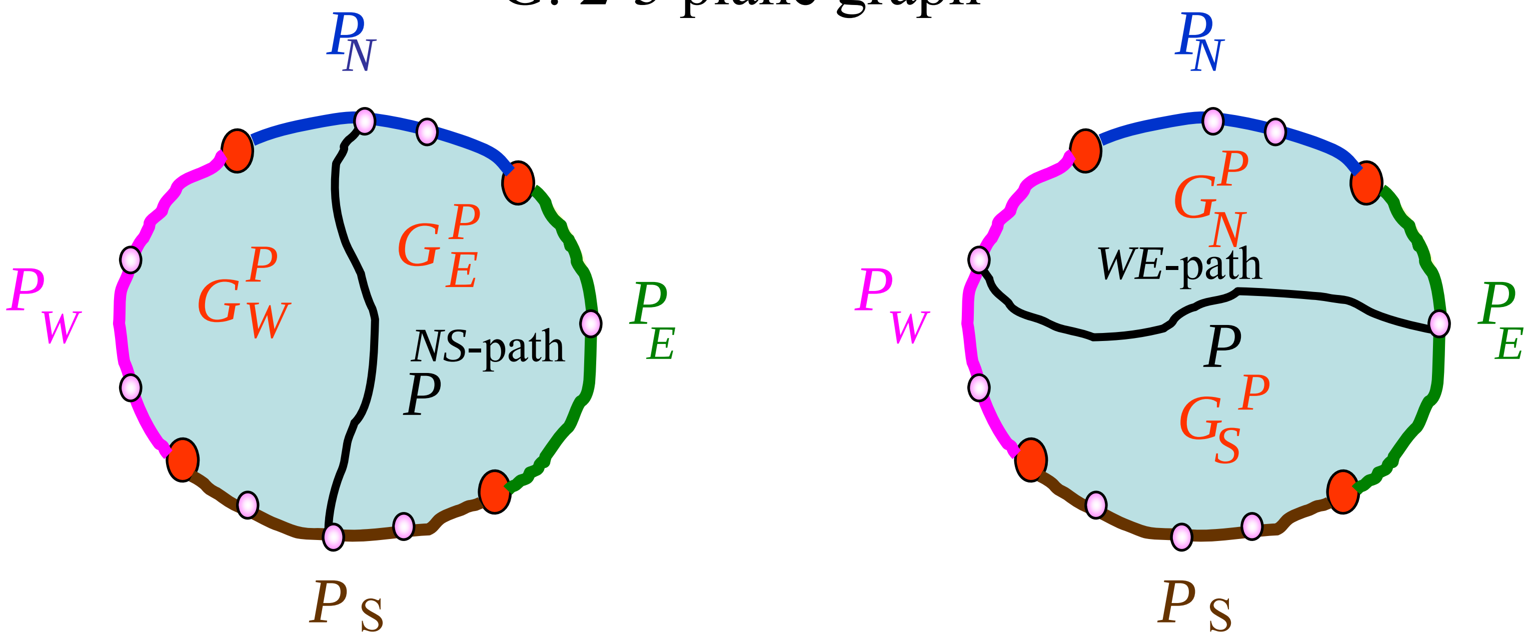
Let G be a 2-3 plane graph obtained from a cyclically 5-edge connected plane cubic graph by inserting four vertices of degree 2 on four distinct edges. Then G is a good slicing graph.

That is, Thomassen's graph is a good slicing graph.


Slicing Graphs
Good Slicing Graphs
Cyclically 5-edge
Connected cubic graph
with four vertices of
degree 2 inserted
on outer face

Slicing Graph

G: 2-3 plane graph



G is a slicing graph if either it has exactly one inner face or it has an *NS*-path or a *WE*-path *P* such that both the subgraphs corresponding to *P* are slicing graphs.

P is called a slicing path.