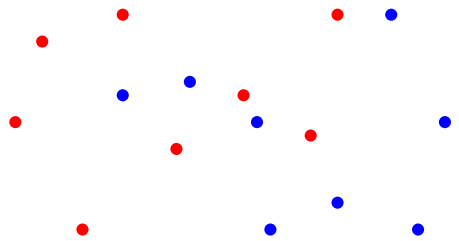
On Embeddability of Buses in Point Sets

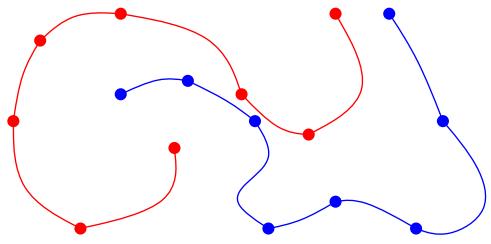
Till Bruckdorfer, Michael Kaufmann, Stephen Kobourov, Sergey Pupyrev

26-09-2015

Motivation: Set Membership Visualization

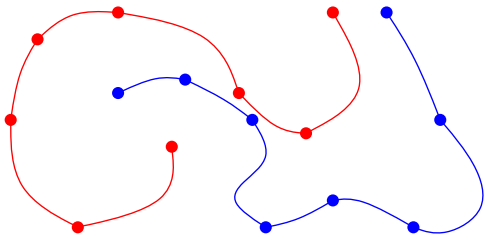


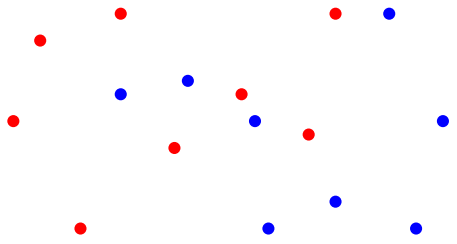
Motivation: Set Membership Visualization



Motivation: Set Membership Visualization

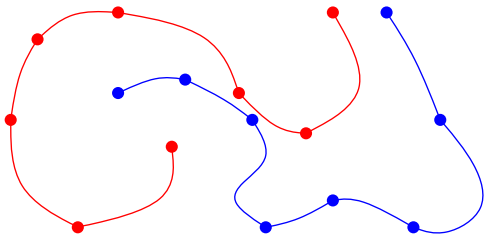
LineSets [Alper et al., 2011]

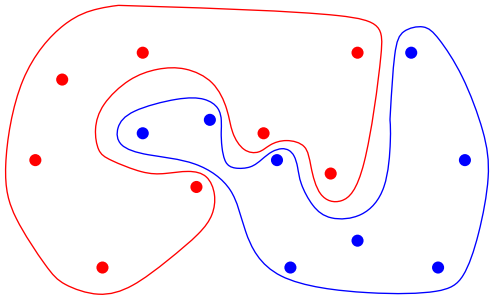




Motivation: Set Membership Visualization

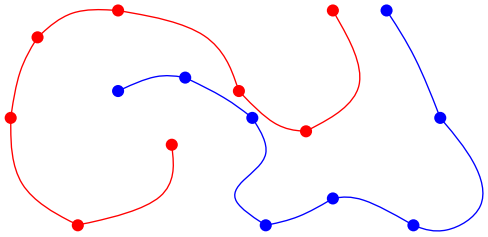
LineSets [Alper et al., 2011]

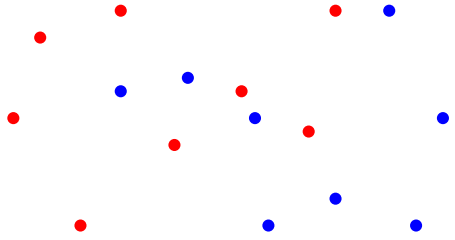


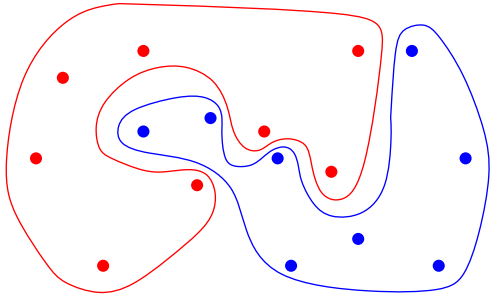


Motivation: Set Membership Visualization

LineSets [Alper et al., 2011]



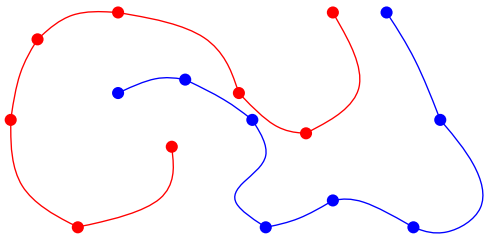


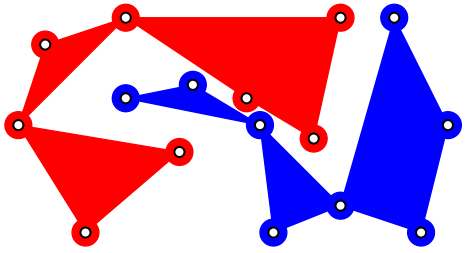


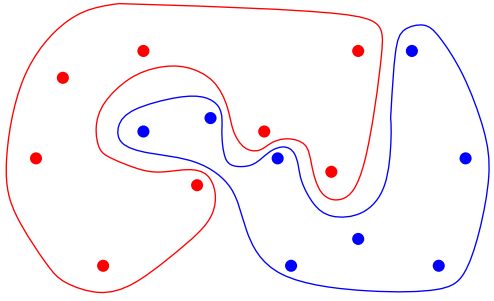
Eulerdiagrams

Motivation: Set Membership Visualization

LineSets [Alper et al., 2011]



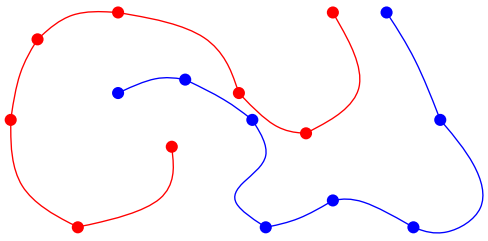




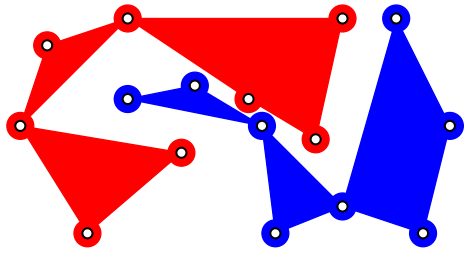
Eulerdiagrams

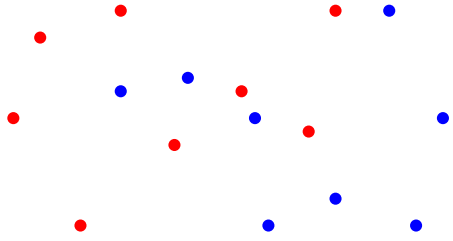
Motivation: Set Membership Visualization

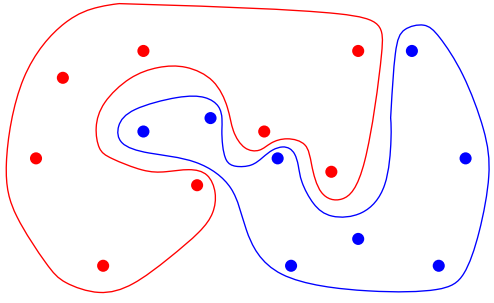
LineSets [Alper et al., 2011]



KelpFusion [Meulemans et al., 2013]



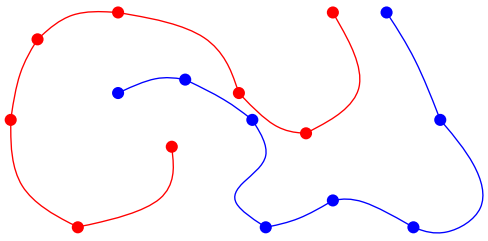




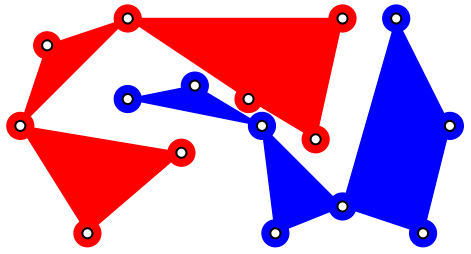
Eulerdiagrams

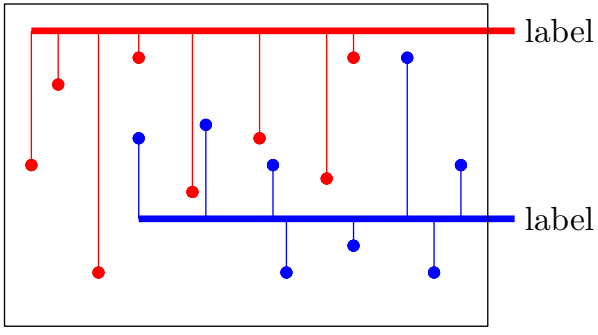
Motivation: Set Membership Visualization

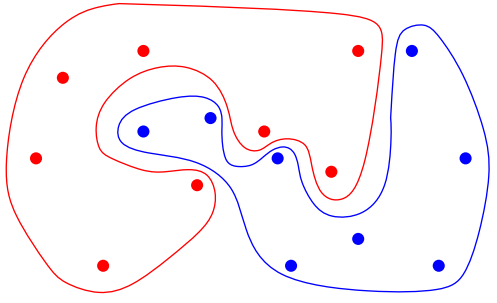
LineSets [Alper et al., 2011]



KelpFusion [Meulemans et al., 2013]



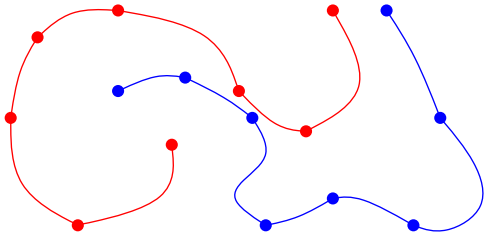




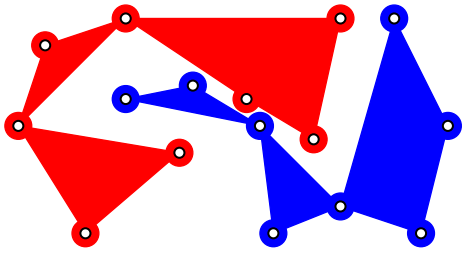
Eulerdiagrams

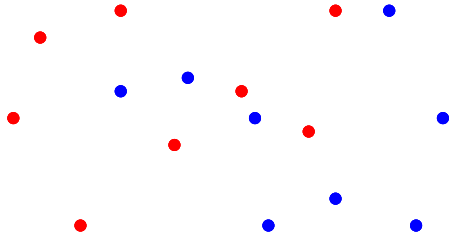
Motivation: Set Membership Visualization

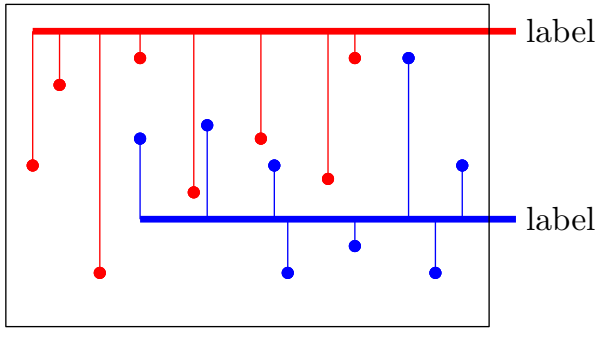
LineSets [Alper et al., 2011]



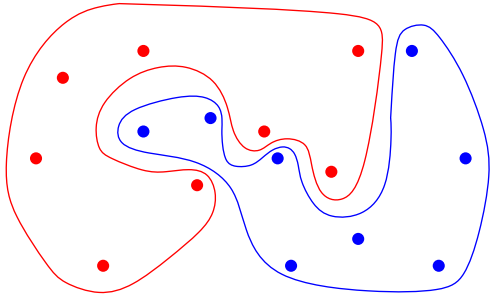
KelpFusion [Meulemans et al., 2013]







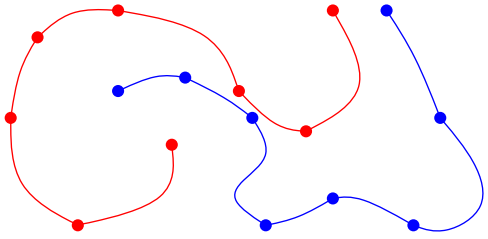
Backbone Boundary Labeling [Bekos et al., 2013]



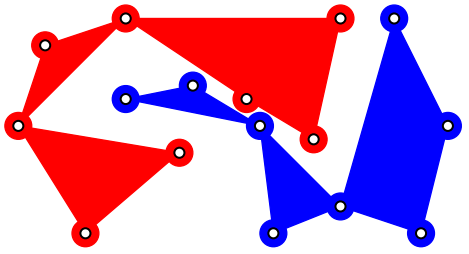
Eulerdiagrams

Motivation: Set Membership Visualization

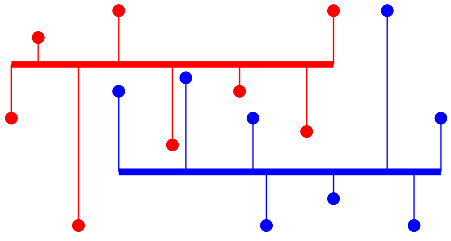
LineSets [Alper et al., 2011]

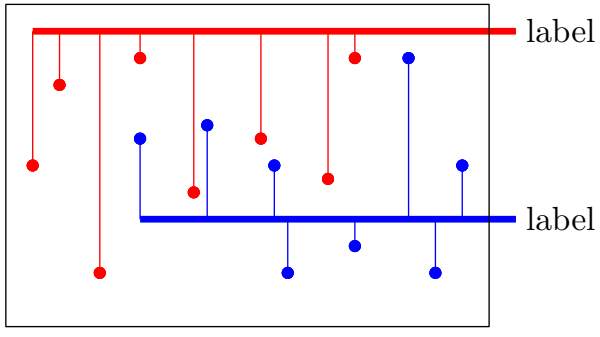


KelpFusion [Meulemans et al., 2013]

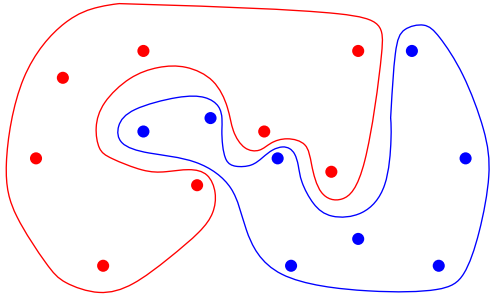


Bus [Ada et al., 2007]



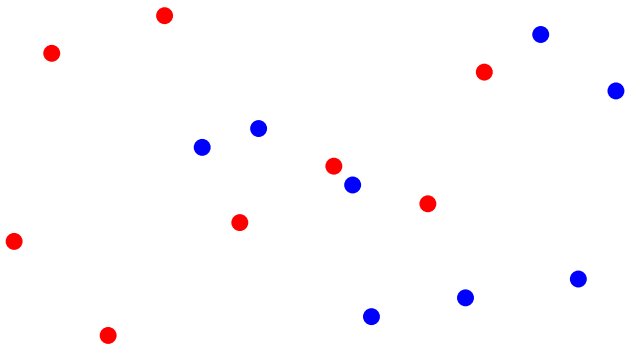


Backbone Boundary Labeling [Bekos et al., 2013]



Eulerdiagrams

Bus Embeddability Problem (BEP)

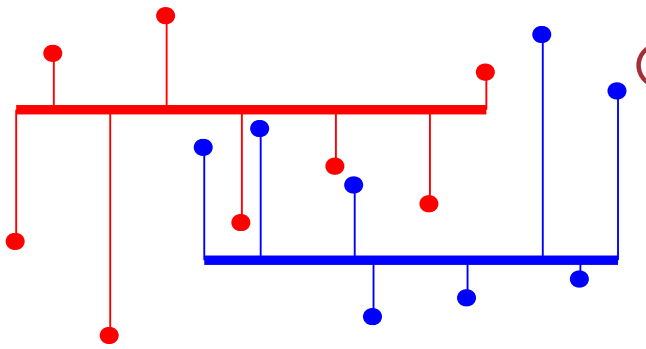


Input: Set of colored points

Output: Bus = colored horiz. segment

All points of the same color are orthogonally connected with their bus without crossings.

Bus Embeddability Problem (BEP)

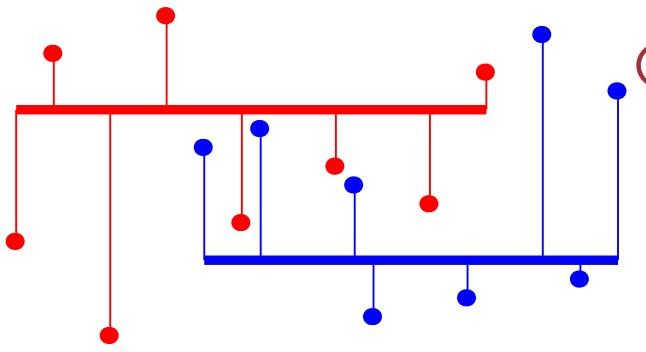


Input: Set of colored points

Output: Bus = colored horiz. segment

All points of the same color are orthogonally connected with their bus without crossings.

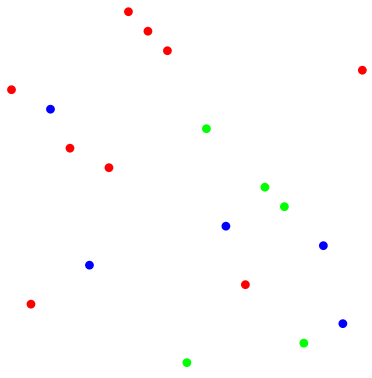
Bus Embeddability Problem (BEP)



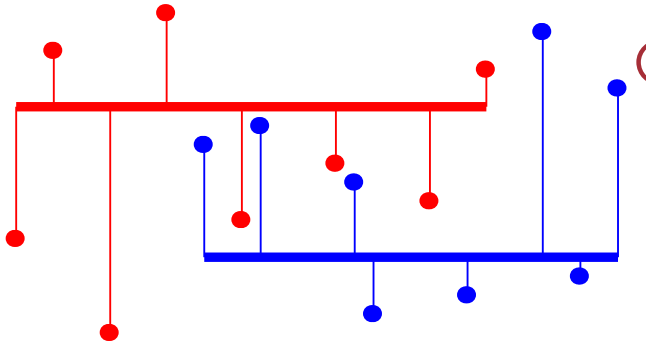
Input: Set of colored points

Output: Bus = colored horiz. segment

All points of the same color are orthogonally connected with their bus without crossings.



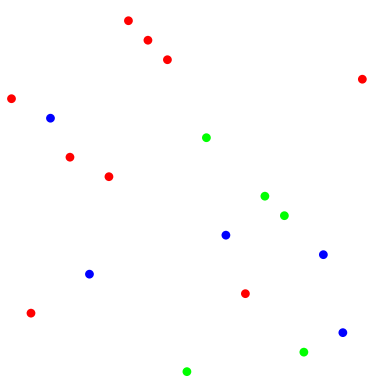
Bus Embeddability Problem (BEP)

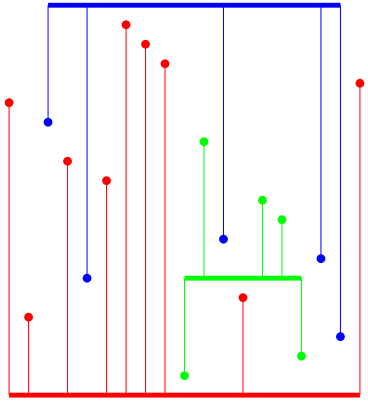


Input: Set of colored points

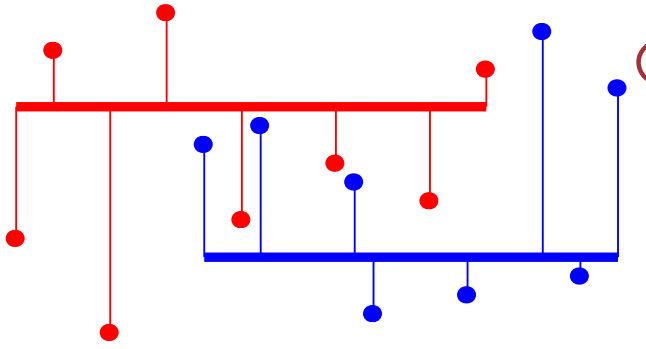
Output: Bus = colored horiz. segment

All points of the same color are orthogonally connected with their bus without crossings.





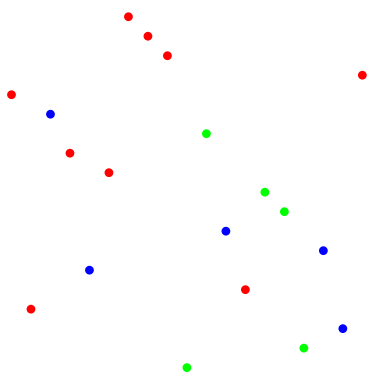
Bus Embeddability Problem (BEP)

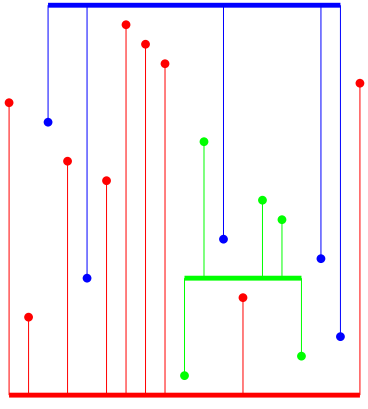


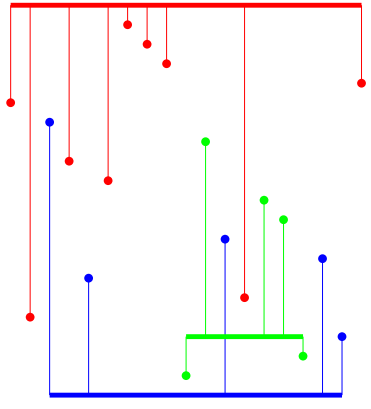
Input: Set of colored points

Output: Bus = colored horiz. segment

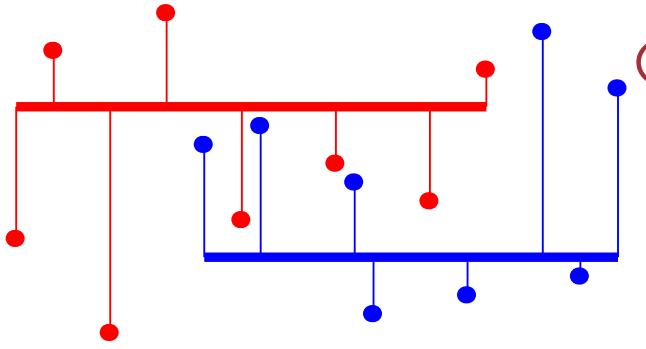
All points of the same color are orthogonally connected with their bus without crossings.







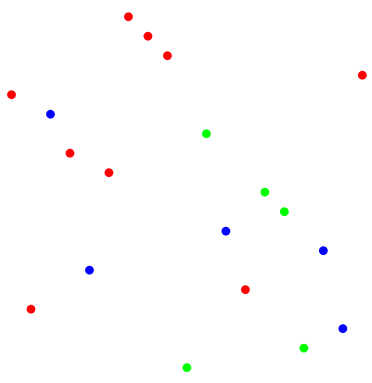
Bus Embeddability Problem (BEP)

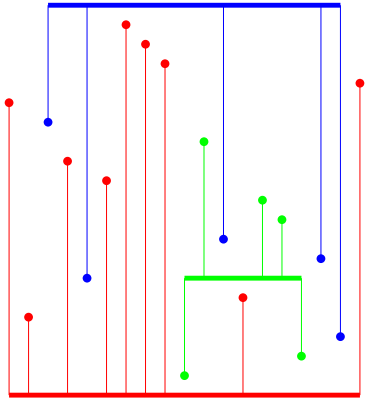


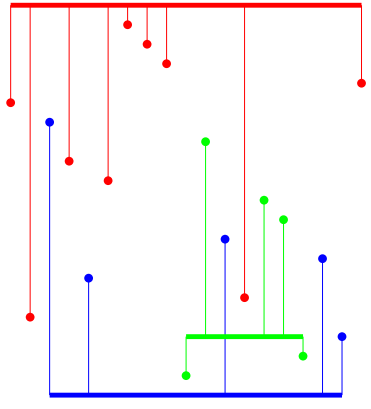
Input: Set of colored points

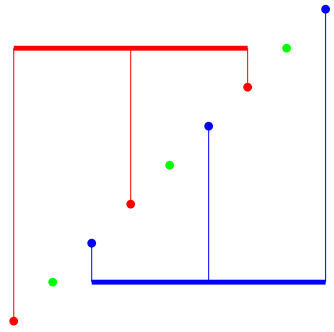
Output: Bus = colored horiz. segment

All points of the same color are orthogonally connected with their bus without crossings.

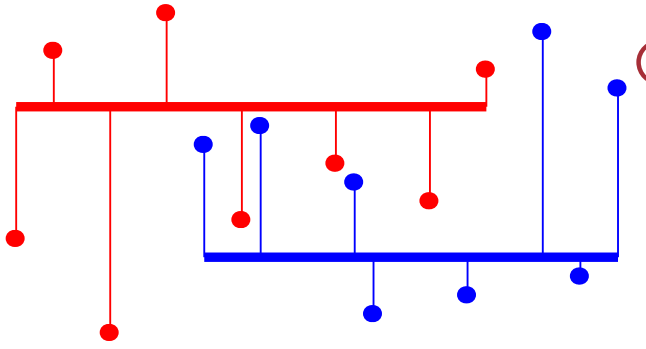








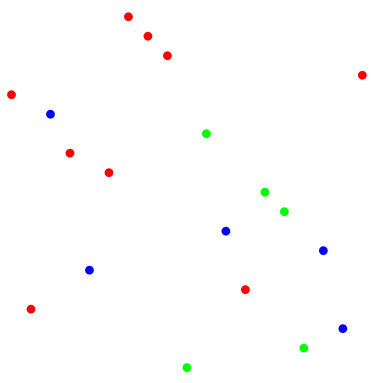
Bus Embeddability Problem (BEP)

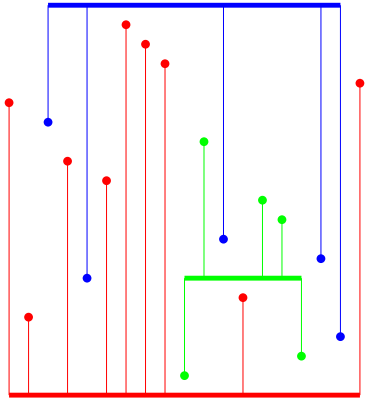


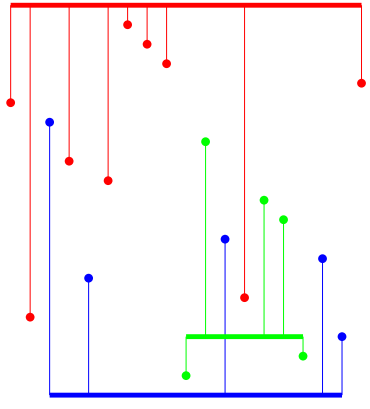
Input: Set of colored points

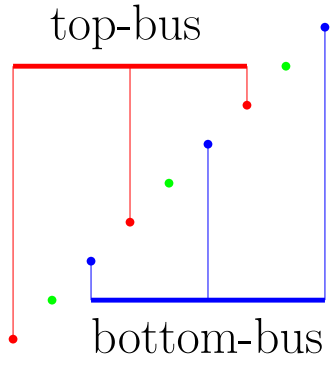
Output: Bus = colored horiz. segment

All points of the same color are orthogonally connected with their bus without crossings.









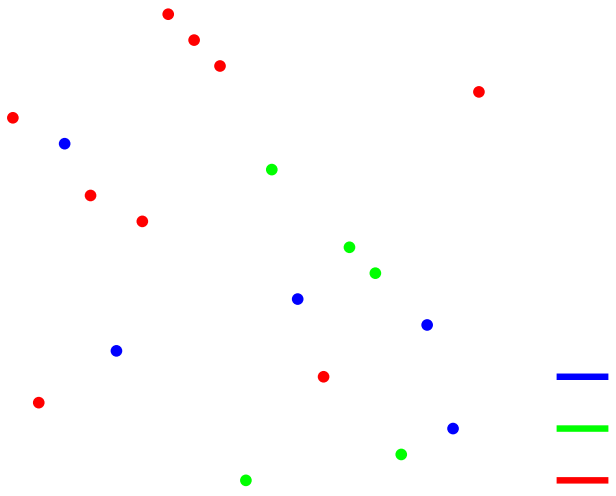
1. General Type of Buses: Given Relative Order

Input: n colored points

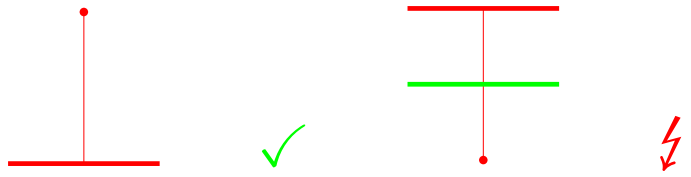
Output: y-coord. of buses

Constr.: given relative order

$\Rightarrow$ Test $O(n \log n)$ time.



- scan from bottom to top
- draw bus as early as possible



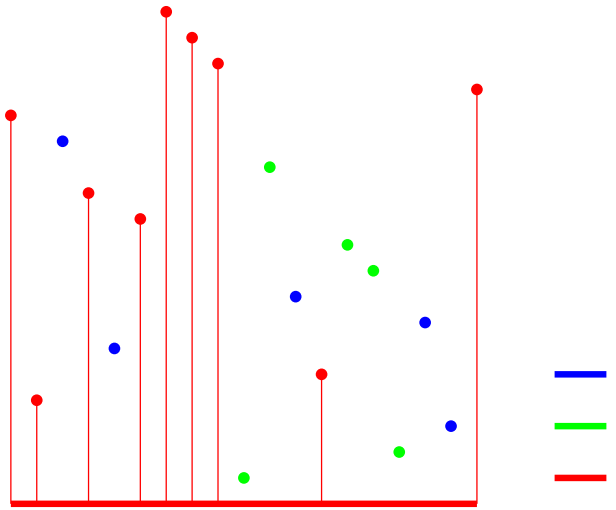
1. General Type of Buses: Given Relative Order

Input: n colored points

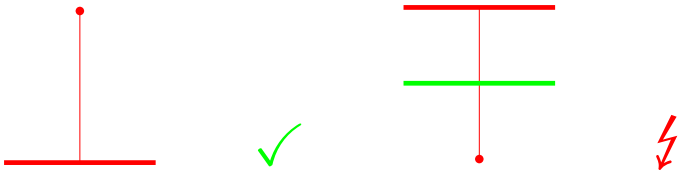
Output: y-coord. of buses

Constr.: given relative order

$\Rightarrow$ Test $O(n \log n)$ time.



- scan from bottom to top
- draw bus as early as possible



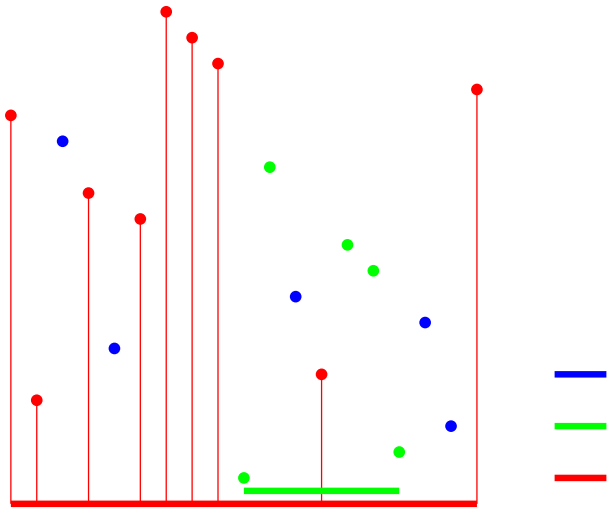
1. General Type of Buses: Given Relative Order

Input: n colored points

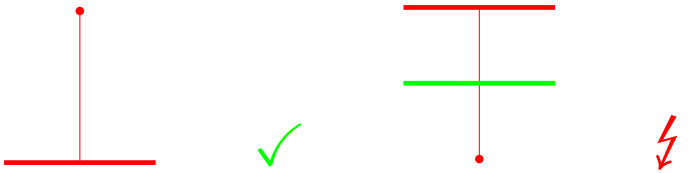
Output: y-coord. of buses

Constr.: given relative order

$\Rightarrow$ Test $O(n \log n)$ time.



- scan from bottom to top
- draw bus as early as possible



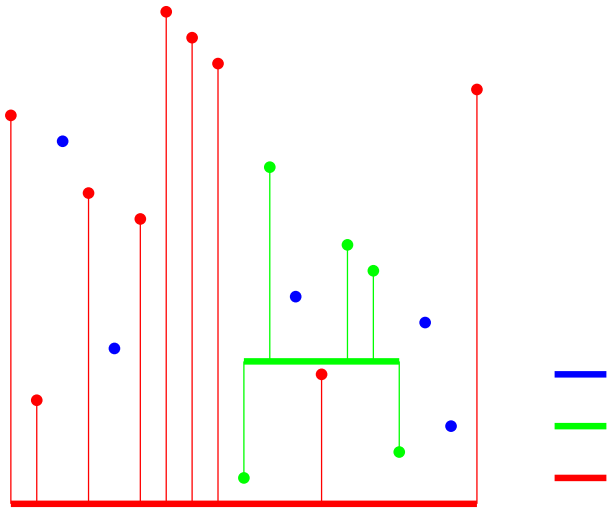
1. General Type of Buses: Given Relative Order

Input: n colored points

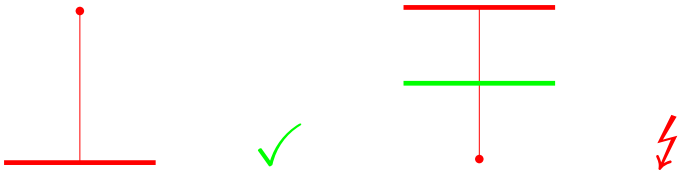
Output: y-coord. of buses

Constr.: given relative order

$\Rightarrow$ Test $O(n \log n)$ time.



- scan from bottom to top
- draw bus as early as possible



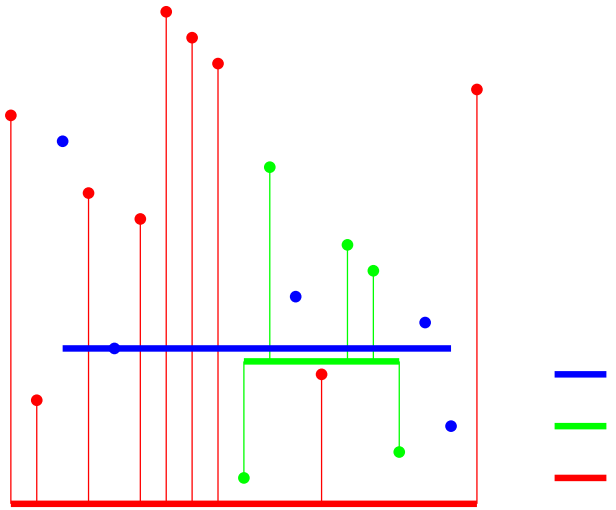
1. General Type of Buses: Given Relative Order

Input: n colored points

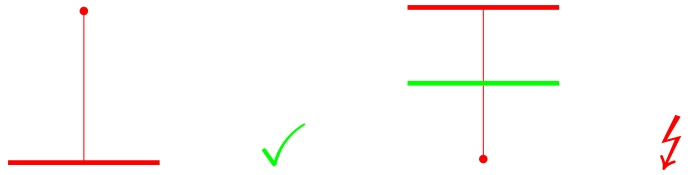
Output: y-coord. of buses

Constr.: given relative order

$\Rightarrow$ Test $O(n \log n)$ time.



- scan from bottom to top
- draw bus as early as possible



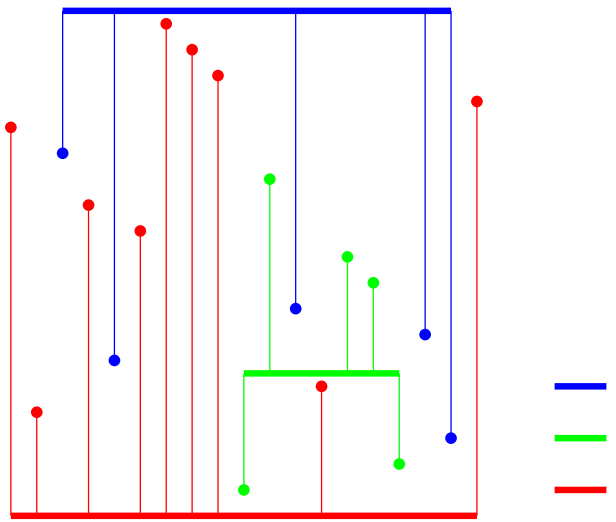
1. General Type of Buses: Given Relative Order

Input: n colored points

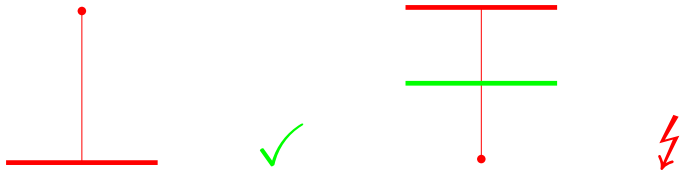
Output: y-coord. of buses

Constr.: given relative order

$\Rightarrow$ Test $O(n \log n)$ time.



- scan from bottom to top
- draw bus as early as possible



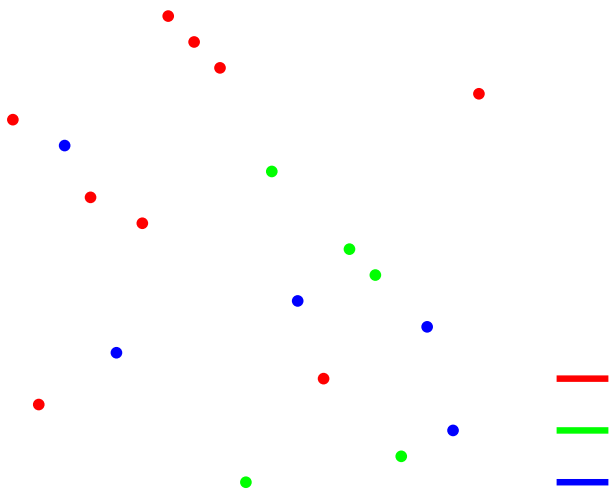
1. General Type of Buses: Given Relative Order

Input: n colored points

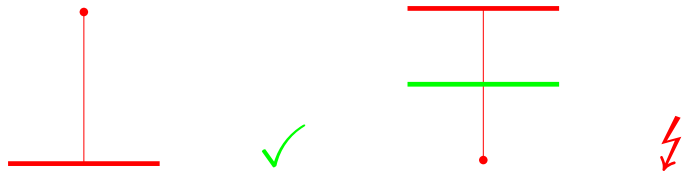
Output: y-coord. of buses

Constr.: given relative order

$\Rightarrow$ Test $O(n \log n)$ time.



- scan from bottom to top
- draw bus as early as possible



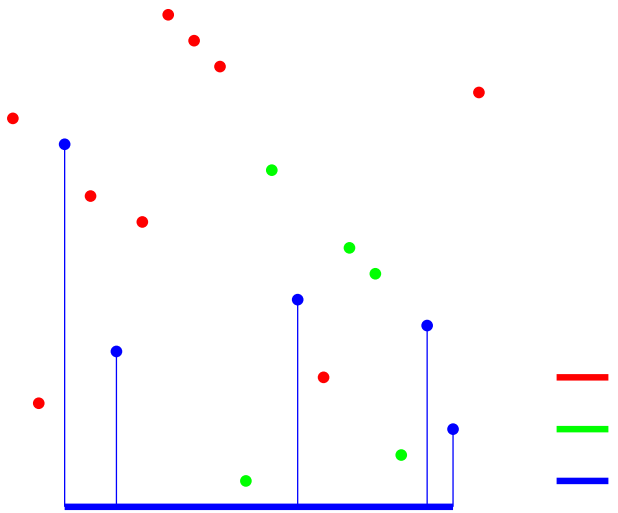
1. General Type of Buses: Given Relative Order

Input: n colored points

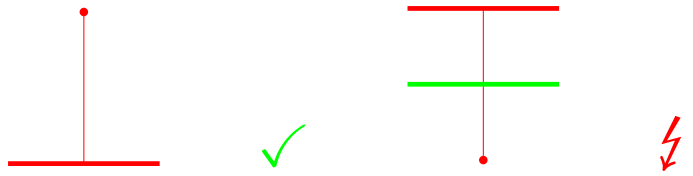
Output: y-coord. of buses

Constr.: given relative order

$\Rightarrow$ Test $O(n \log n)$ time.



- scan from bottom to top
- draw bus as early as possible



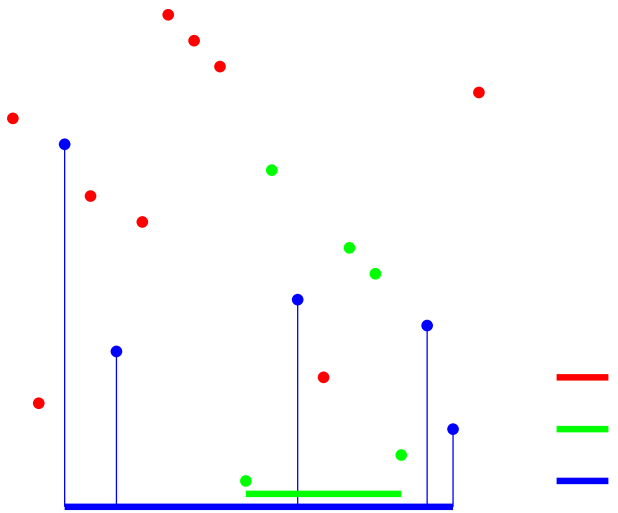
1. General Type of Buses: Given Relative Order

Input: n colored points

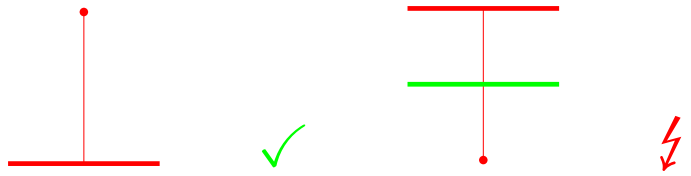
Output: y-coord. of buses

Constr.: given relative order

$\Rightarrow$ Test $O(n \log n)$ time.



- scan from bottom to top
- draw bus as early as possible



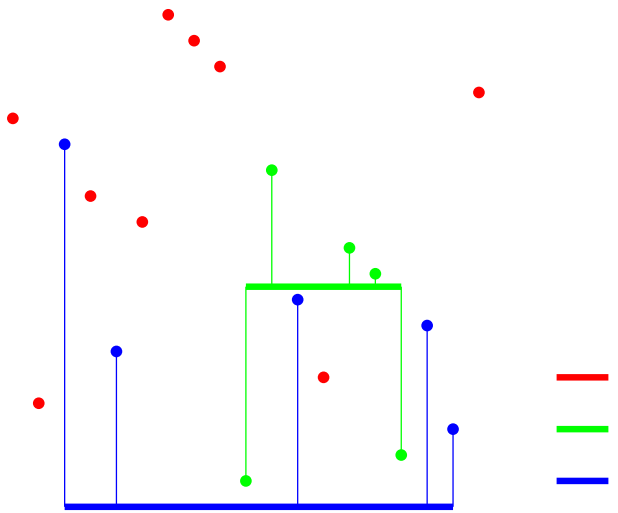
1. General Type of Buses: Given Relative Order

Input: n colored points

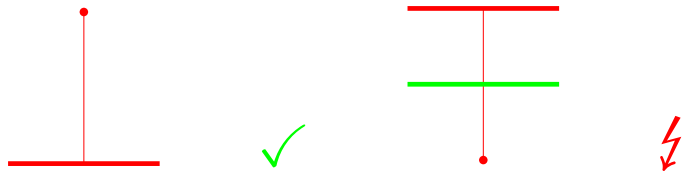
Output: y-coord. of buses

Constr.: given relative order

$\Rightarrow$ Test $O(n \log n)$ time.



- scan from bottom to top
- draw bus as early as possible



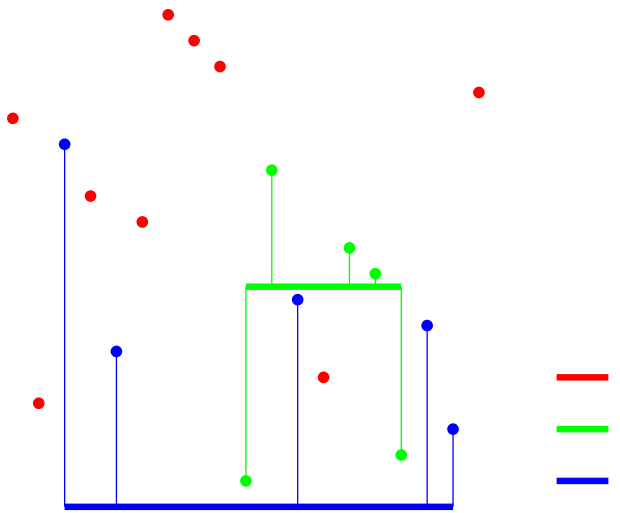
1. General Type of Buses: Given Relative Order

Input: n colored points

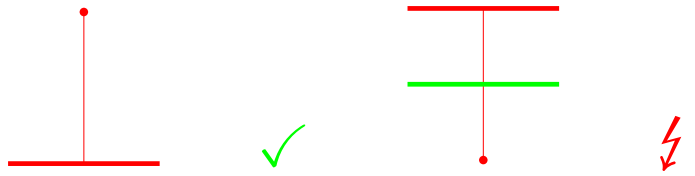
Output: y-coord. of buses

Constr.: given relative order

$\Rightarrow$ Test $O(n \log n)$ time.



- scan from bottom to top
- draw bus as early as possible



- use segment tree: support extract max. y-coordinate of processed points on range

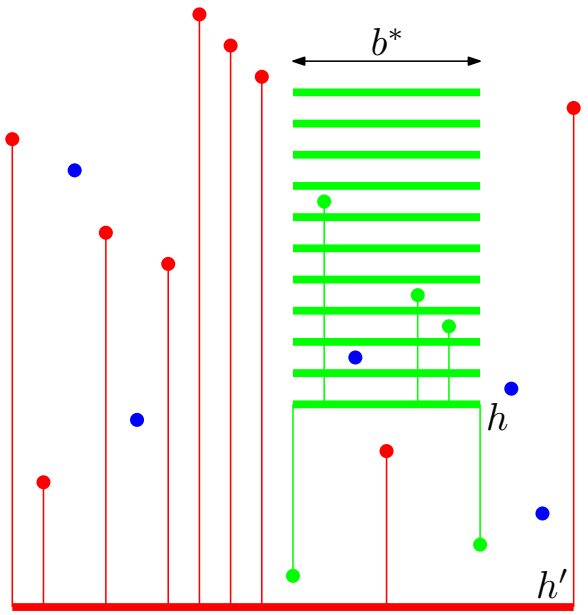
2. General Type of Buses: no Restriction

Input: n colored points

Output: y-coord. of buses

Constr.: —

$\Rightarrow$ Test $O(p(n)2^k)$ time.



- Dynamic Programming
- state pair (h,B), $B \subseteq \{b_1,\ldots,b_k\}$, uses all buses of B + topmost bus at y-coord. $0 \le h \le n+1$
- $F(h,B)$ =true $\Leftrightarrow$ $\exists$ solution using only buses of B
- $F(h,B)$:=true for all $|B| = 1$
- $F(h,B)$:=true, if $\exists\ b^* \in B$ with $F(h',B \setminus \{b^*\})$ =true for $h' < h$ and $h \ge$ topmost point "$\in b^*$"
- there are $\approx n2^k$ state pairs
- compute state pair in $O(p(n))$

2. General Type of Busese: ILP with Experiments

Input: n colored points

Output: y-coord. of buses

Constr.: —

$\Rightarrow$ ILP

$$\begin{aligned} \min\ & 1 \\ s.t.\ & \forall\ (b,p), c(p) \neq b, \text{“}p \in b\text{”}: \\ & y(p) < y(b) \wedge y(c(p)) < y(b) \\ \vee\ & y(p) > y(b) \wedge y(c(p)) > y(b) \end{aligned}$$

2. General Type of Busese: ILP with Experiments

Input: n colored points

Output: y-coord. of buses

Constr.: —

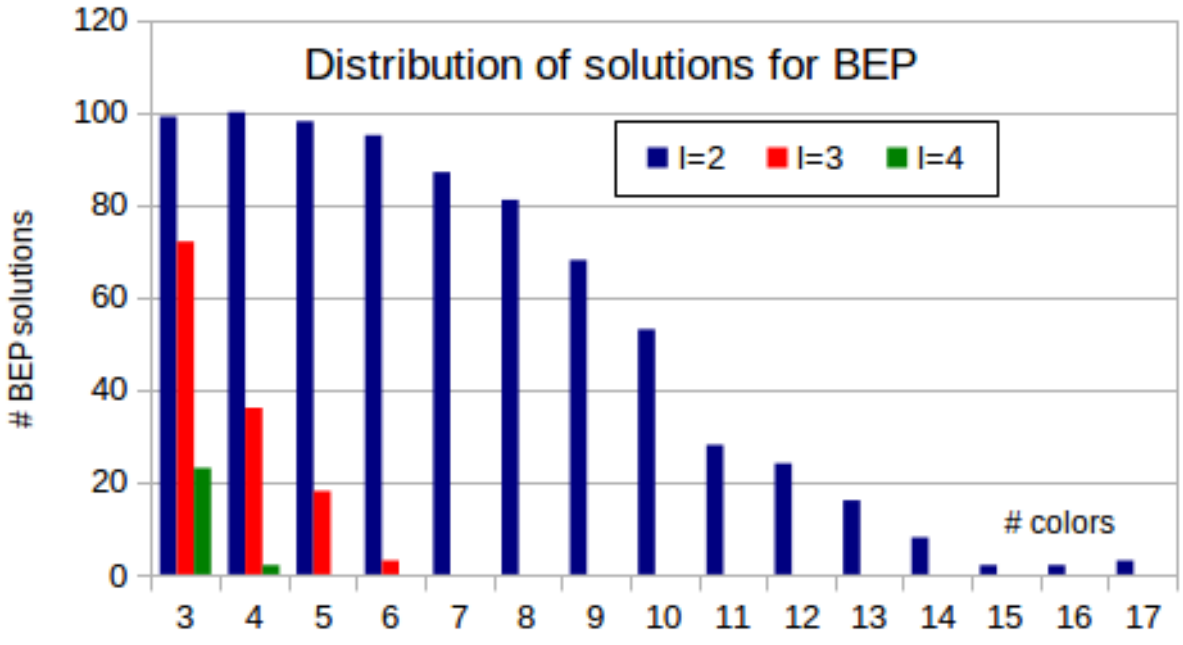
$\Rightarrow$ ILP

$$\min \ 1$$

$$s.t. \ \forall\,(b,p), c(p) \neq b, \text{“}p \in b\text{”}:$$

$$y(p) < y(b) \wedge y(c(p)) < y(b)$$

$$\vee \ y(p) > y(b) \wedge y(c(p)) > y(b)$$



3. Restricted Type of Buses: ⊓-BEP

Input: n colored points

Output: y-coord. of buses

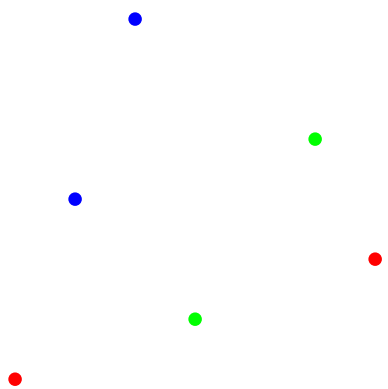
Constr.: using only ⊓-buses

$\Rightarrow$ Test $O(n \log n)$ time

- scan from bottom to top
- draw bus as early as possible
- store scanned points in array A
- $\exists$ pattern yxxy $\in A$

$\Rightarrow$ draw bus x & remove xx from A

- in the end $A = \varnothing \Rightarrow \exists$ solution
- pattern xyxy $\in A \Rightarrow \nexists$ solution



3. Restricted Type of Buses: ⊓-BEP

Input: n colored points

Output: y-coord. of buses

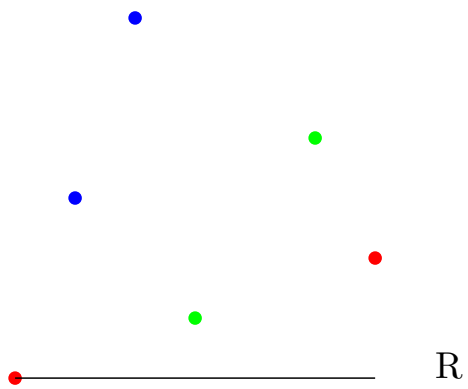
Constr.: using only ⊓-buses

$\Rightarrow$ Test $O(n \log n)$ time

- scan from bottom to top
- draw bus as early as possible
- store scanned points in array A
- $\exists$ pattern yxxy $\in A$

$\Rightarrow$ draw bus x & remove xx from A

- in the end $A = \varnothing \Rightarrow \exists$ solution
- pattern xyxy $\in A \Rightarrow \nexists$ solution



3. Restricted Type of Buses: ⊓-BEP

Input: n colored points

Output: y-coord. of buses

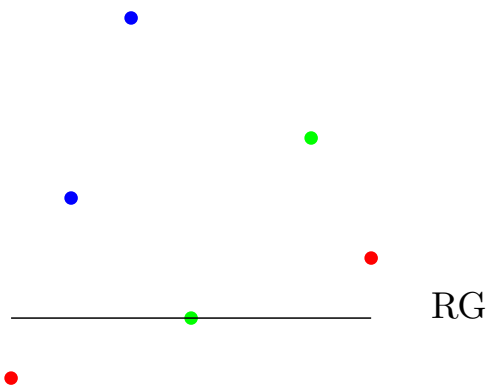
Constr.: using only ⊓-buses

$\Rightarrow$ Test $O(n \log n)$ time

- scan from bottom to top
- draw bus as early as possible
- store scanned points in array A
- $\exists$ pattern yxxy $\in A$

$\Rightarrow$ draw bus x & remove xx from A

- in the end $A = \varnothing \Rightarrow \exists$ solution
- pattern xyxy $\in A \Rightarrow \nexists$ solution



3. Restricted Type of Buses: ⊓-BEP

Input: n colored points

Output: y-coord. of buses

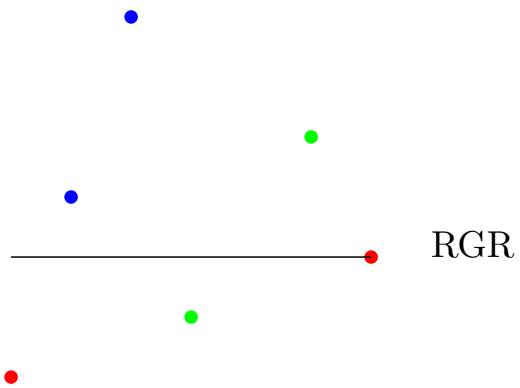
Constr.: using only ⊓-buses

$\Rightarrow$ Test $O(n \log n)$ time

- scan from bottom to top
- draw bus as early as possible
- store scanned points in array A
- $\exists$ pattern yxxy $\in A$

$\Rightarrow$ draw bus x & remove xx from A

- in the end $A = \varnothing \Rightarrow \exists$ solution
- pattern xyxy $\in A \Rightarrow \nexists$ solution



3. Restricted Type of Buses: ⊓-BEP

Input: n colored points

Output: y-coord. of buses

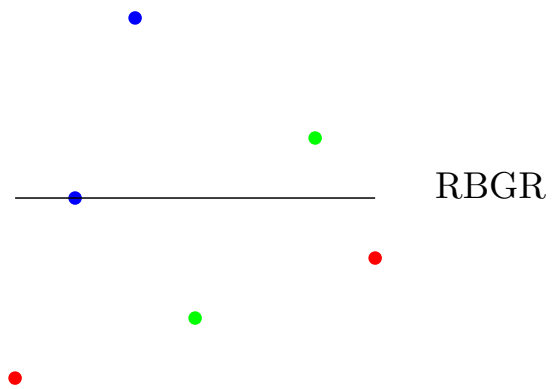
Constr.: using only ⊓-buses

$\Rightarrow$ Test $O(n \log n)$ time

- scan from bottom to top
- draw bus as early as possible
- store scanned points in array A
- $\exists$ pattern yxxy $\in A$

$\Rightarrow$ draw bus x & remove xx from A

- in the end $A = \emptyset \Rightarrow \exists$ solution
- pattern xyxy $\in A \Rightarrow \nexists$ solution



3. Restricted Type of Buses: ⊓-BEP

Input: n colored points

Output: y-coord. of buses

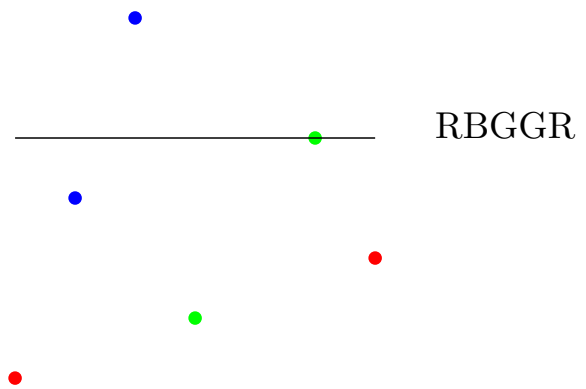
Constr.: using only ⊓-buses

$\Rightarrow$ Test $O(n \log n)$ time

- scan from bottom to top
- draw bus as early as possible
- store scanned points in array A
- $\exists$ pattern yxxy $\in A$

$\Rightarrow$ draw bus x & remove xx from A

- in the end $A = \emptyset \Rightarrow \exists$ solution
- pattern xyxy $\in A \Rightarrow \nexists$ solution



3. Restricted Type of Buses: ⊓-BEP

Input: n colored points

Output: y-coord. of buses

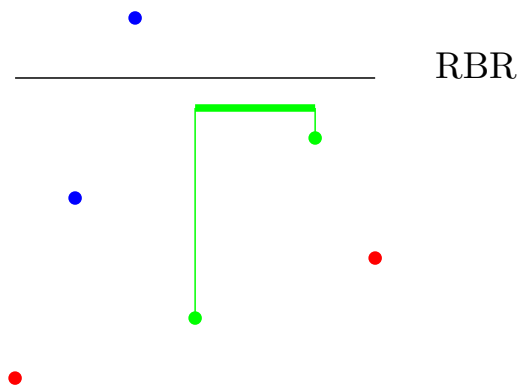
Constr.: using only ⊓-buses

$\Rightarrow$ Test $O(n \log n)$ time

- scan from bottom to top
- draw bus as early as possible
- store scanned points in array A
- $\exists$ pattern yxxy $\in A$

$\Rightarrow$ draw bus x & remove xx from A

- in the end $A = \varnothing \Rightarrow \exists$ solution
- pattern xyxy $\in A \Rightarrow \nexists$ solution



3. Restricted Type of Buses: ⊓-BEP

Input: n colored points

Output: y-coord. of buses

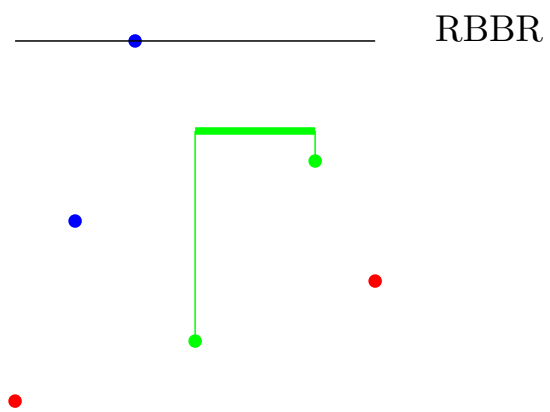
Constr.: using only ⊓-buses

$\Rightarrow$ Test $O(n \log n)$ time

- scan from bottom to top
- draw bus as early as possible
- store scanned points in array A
- $\exists$ pattern yxxy $\in A$

$\Rightarrow$ draw bus x & remove xx from A

- in the end $A = \varnothing \Rightarrow \exists$ solution
- pattern xyxy $\in A \Rightarrow \nexists$ solution



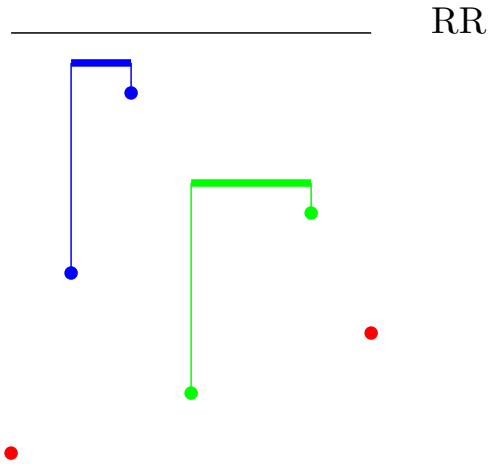
3. Restricted Type of Buses: ⊓-BEP

Input: n colored points

Output: y-coord. of buses

Constr.: using only ⊓-buses

$\Rightarrow$ Test $O(n \log n)$ time



- scan from bottom to top
- draw bus as early as possible
- store scanned points in array A
- $\exists$ pattern yxxy $\in A$

$\Rightarrow$ draw bus x & remove xx from A

- in the end $A = \varnothing \Rightarrow \exists$ solution
- pattern xyxy $\in A \Rightarrow \nexists$ solution

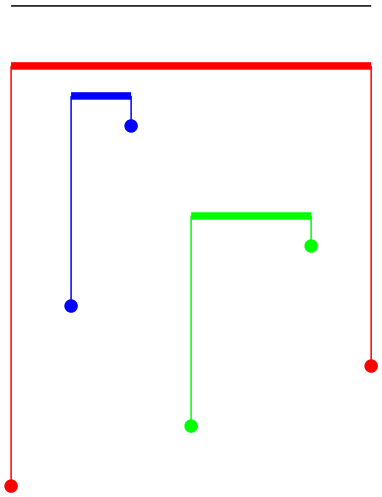
3. Restricted Type of Buses: ⊓-BEP

Input: n colored points

Output: y-coord. of buses

Constr.: using only ⊓-buses

$\Rightarrow$ Test $O(n \log n)$ time



- scan from bottom to top
- draw bus as early as possible
- store scanned points in array A
- $\exists$ pattern yxxy $\in A$

$\Rightarrow$ draw bus x & remove xx from A

- in the end $A = \varnothing \Rightarrow \exists$ solution
- pattern xyxy $\in A \Rightarrow \nexists$ solution

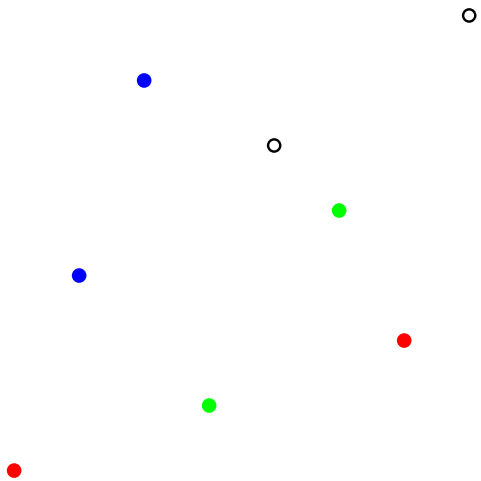
3. Restricted Type of Buses: ⊓-BEP

Input: n colored points

Output: y-coord. of buses

Constr.: using only ⊓-buses

$\Rightarrow$ Test $O(n \log n)$ time



- scan from bottom to top
- draw bus as early as possible
- store scanned points in array A
- $\exists$ pattern yxxy $\in A$

$\Rightarrow$ draw bus x & remove xx from A

- in the end $A = \varnothing \Rightarrow \exists$ solution
- pattern xyxy $\in A \Rightarrow \nexists$ solution

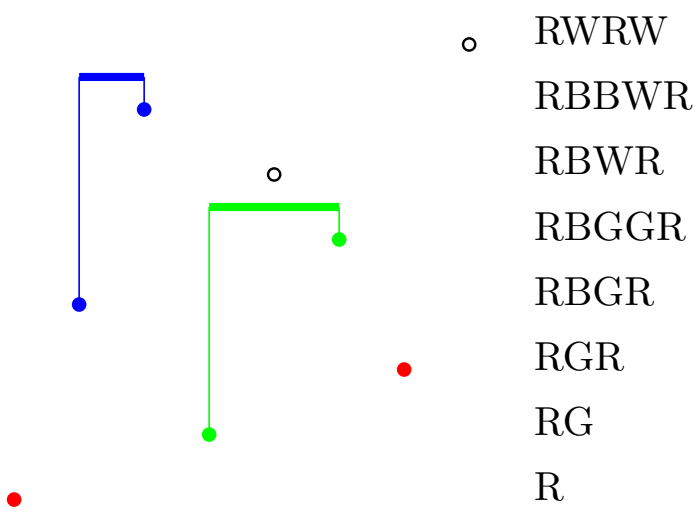
3. Restricted Type of Buses: ⊓-BEP

Input: n colored points

Output: y-coord. of buses

Constr.: using only ⊓-buses

$\Rightarrow$ Test $O(n \log n)$ time



- scan from bottom to top
- draw bus as early as possible
- store scanned points in array A
- $\exists$ pattern yxxy $\in A$

$\Rightarrow$ draw bus x & remove xx from A

- in the end $A = \varnothing \Rightarrow \exists$ solution
- pattern xyxy $\in A \Rightarrow \nexists$ solution

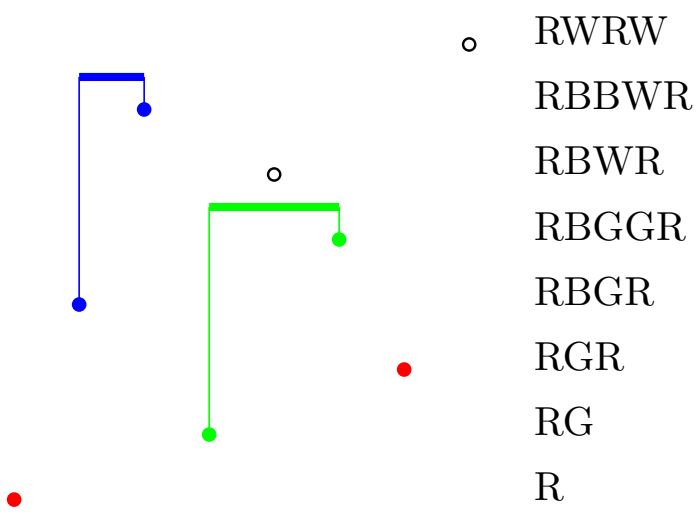
3. Restricted Type of Buses: ⊓-BEP

Input: n colored points

Output: y-coord. of buses

Constr.: using only ⊓-buses

$\Rightarrow$ Test $O(n \log n)$ time



- scan from bottom to top
- draw bus as early as possible
- store scanned points in array A
- $\exists$ pattern yxxy $\in A$

$\Rightarrow$ draw bus x & remove xx from A

- in the end $A = \varnothing \Rightarrow \exists$ solution
- pattern xyxy $\in A \Rightarrow \nexists$ solution
- use balanced binary tree: support insert/remove in $O(\log n)$ time

4. Restricted Type of Buses: (⌜,⌞)-BEP

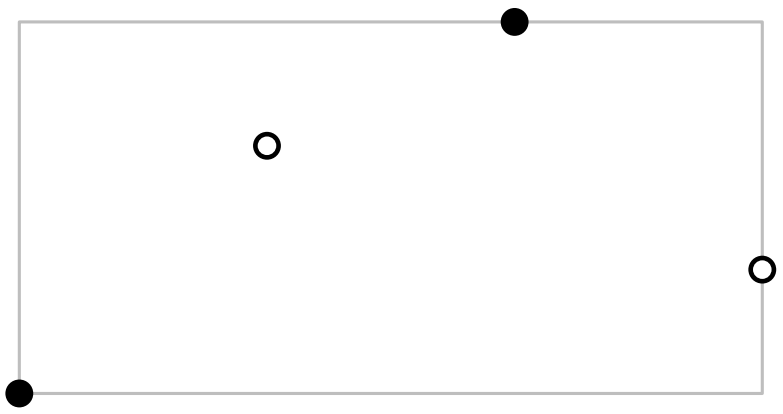
Input: n colored points

Output: y-coord. of buses

Constr.: using only ⌜,⌞-buses

$\Rightarrow$ Test $O(n^2)$ time

- via 2-SAT: variable $x_b \forall$ bus b
- $b = ⌜ \Leftrightarrow x_b =$ true
- $b = ⌞ \Leftrightarrow x_b =$ false
- build a clause for every pair of buses b, b':



$x_{\bullet} = true$

4. Restricted Type of Buses: (⌜,⌞)-BEP

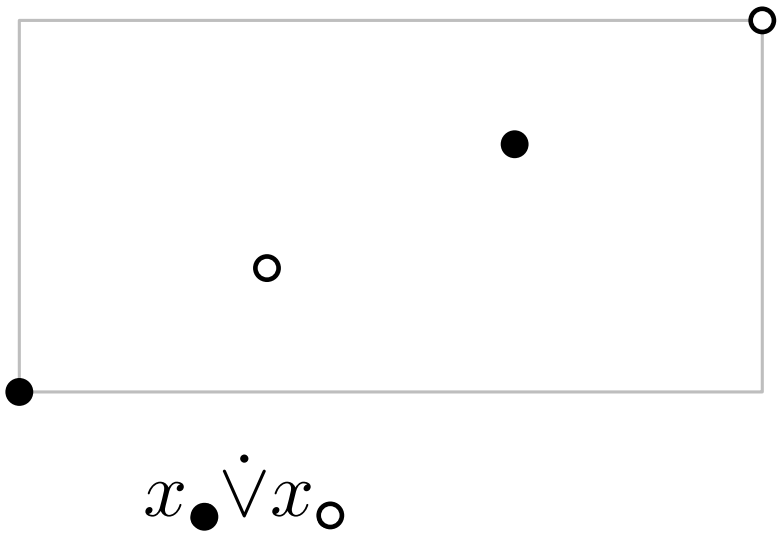
Input: n colored points

Output: y-coord. of buses

Constr.: using only ⌜,⌞-buses

$\Rightarrow$ Test $O(n^2)$ time

- via 2-SAT: variable $x_b \forall$ bus b
- $b = ⌜ \Leftrightarrow x_b =$ true
- $b = ⌞ \Leftrightarrow x_b =$ false
- build a clause for every pair of buses b, b':



4. Restricted Type of Buses: (⌜,⌞)-BEP

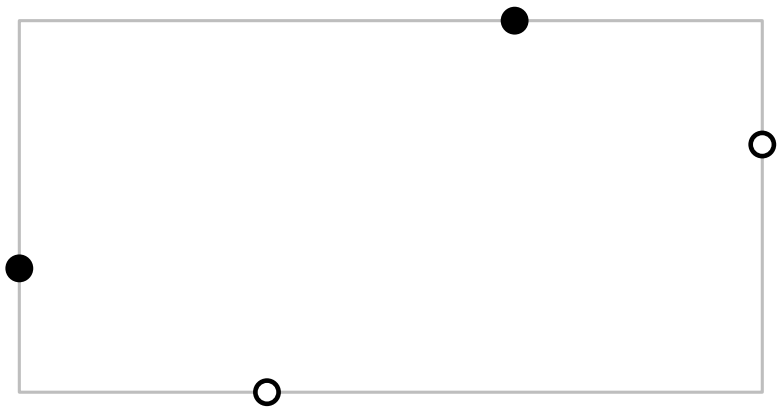
Input: n colored points

Output: y-coord. of buses

Constr.: using only ⌜,⌞-buses

$\Rightarrow$ Test $O(n^2)$ time

- via 2-SAT: variable $x_b \forall$ bus b
- $b = ⌜ \Leftrightarrow x_b$ = true
- $b = ⌞ \Leftrightarrow x_b$ = false
- build a clause for every pair of buses b, b':



$x_{\circ} \Rightarrow x_{\bullet}$

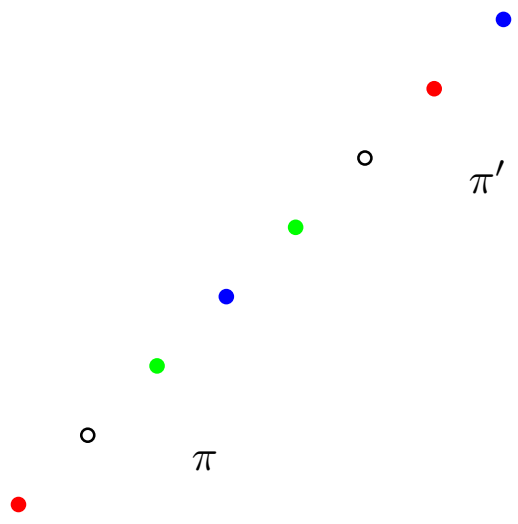
5. Restricted Type of Point Set: Diagonal BEP

Input: 2 points per color

Output: y-coord. of buses

Constr.: on diagonal (π, π')

$\Rightarrow$ Test $O(n^2)$ time

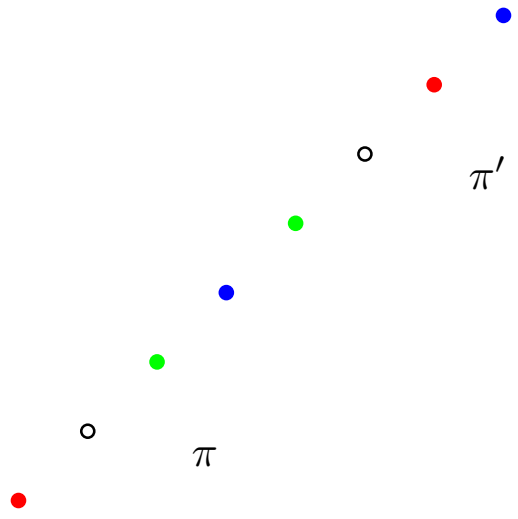


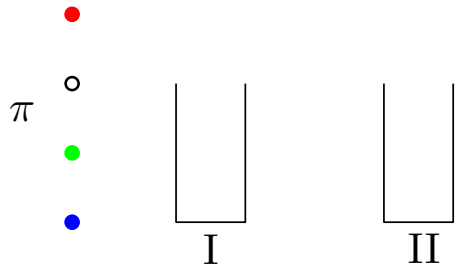
- via 2-stack pushall sorting: sort RWGB to GWRB

5. Restricted Type of Point Set: Diagonal BEP

Input: 2 points per color
Output: y-coord. of buses
Constr.: on diagonal (π, π')
$\Rightarrow$ Test $O(n^2)$ time

- via 2-stack pushall sorting: sort RWGB to GWRB





5. Restricted Type of Point Set: Diagonal BEP

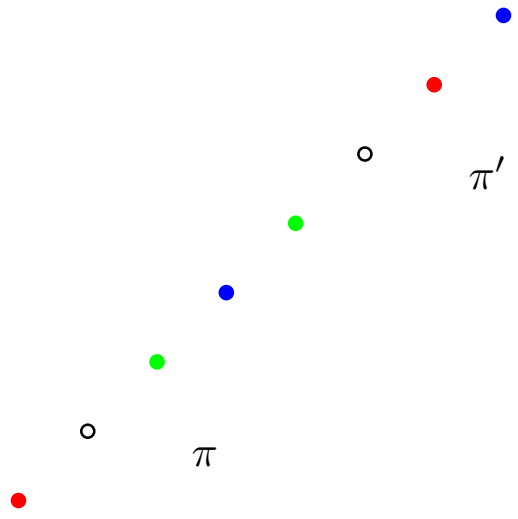
Input: 2 points per color

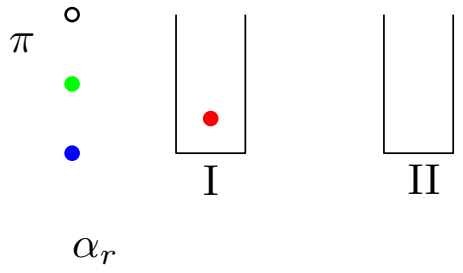
Output: y-coord. of buses

Constr.: on diagonal (π, π')

$\Rightarrow$ Test $O(n^2)$ time

- via 2-stack pushall sorting: sort RWGB to GWRB





5. Restricted Type of Point Set: Diagonal BEP

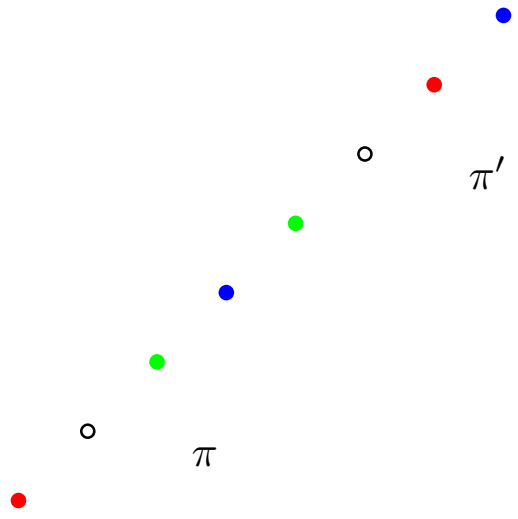
Input: 2 points per color

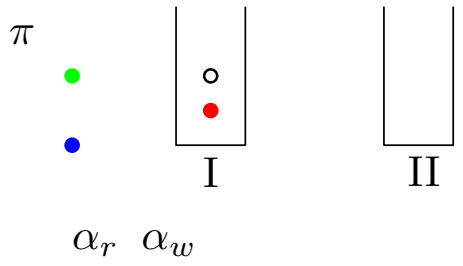
Output: y-coord. of buses

Constr.: on diagonal (π, π')

$\Rightarrow$ Test $O(n^2)$ time

- via 2-stack pushall sorting: sort RWGB to GWRB





5. Restricted Type of Point Set: Diagonal BEP

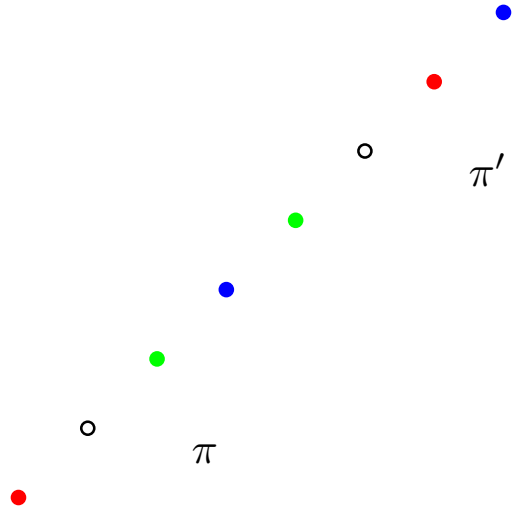
Input: 2 points per color

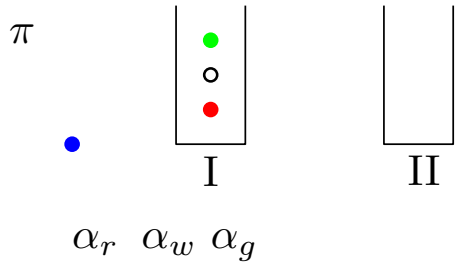
Output: y-coord. of buses

Constr.: on diagonal (π, π')

$\Rightarrow$ Test $O(n^2)$ time

- via 2-stack pushall sorting: sort RWGB to GWRB





5. Restricted Type of Point Set: Diagonal BEP

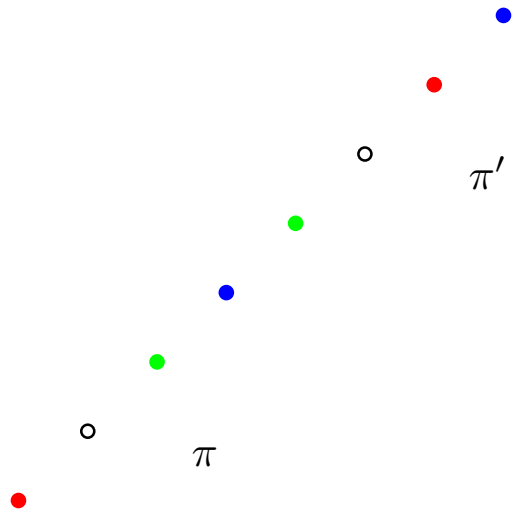
Input: 2 points per color

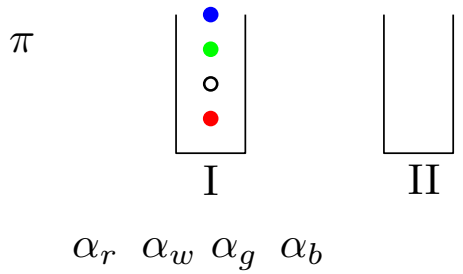
Output: y-coord. of buses

Constr.: on diagonal (π, π')

$\Rightarrow$ Test $O(n^2)$ time

- via 2-stack pushall sorting: sort RWGB to GWRB





5. Restricted Type of Point Set: Diagonal BEP

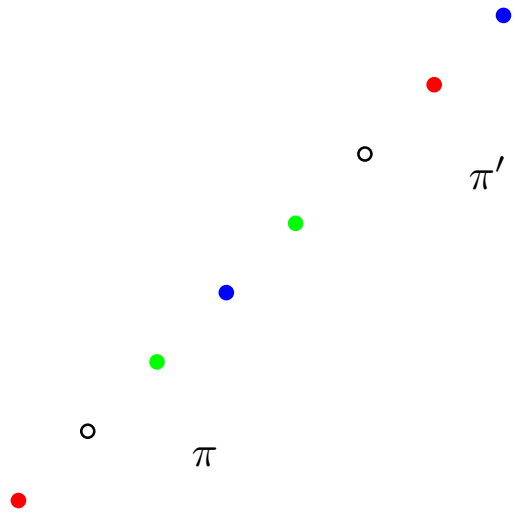
Input: 2 points per color

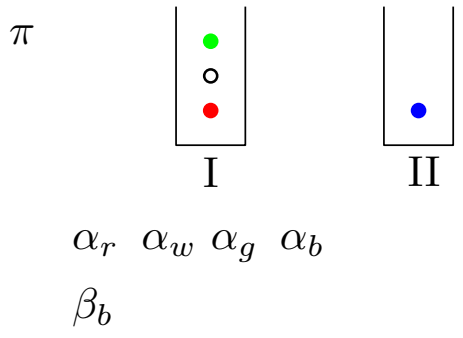
Output: y-coord. of buses

Constr.: on diagonal (π, π')

$\Rightarrow$ Test $O(n^2)$ time

- via 2-stack pushall sorting: sort RWGB to GWRB





5. Restricted Type of Point Set: Diagonal BEP

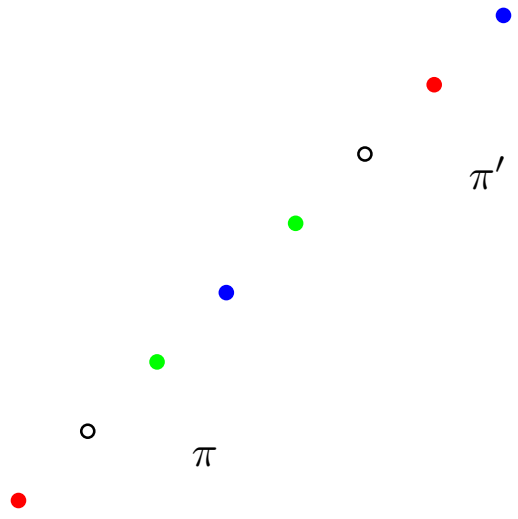
Input: 2 points per color

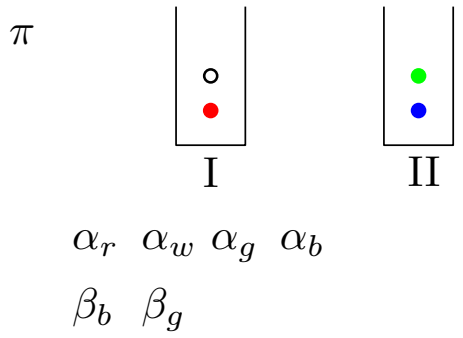
Output: y-coord. of buses

Constr.: on diagonal (π, π')

$\Rightarrow$ Test $O(n^2)$ time

- via 2-stack pushall sorting: sort RWGB to GWRB





5. Restricted Type of Point Set: Diagonal BEP

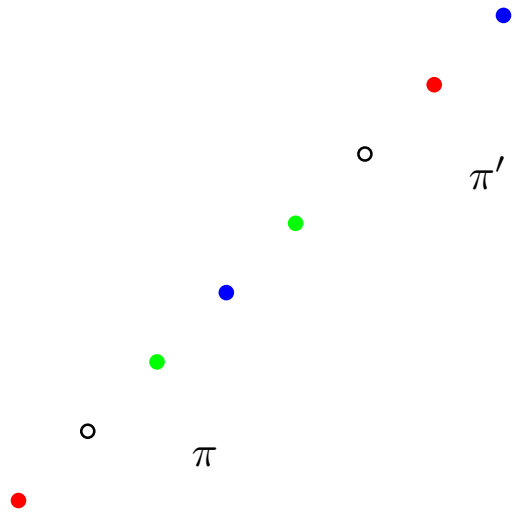
Input: 2 points per color

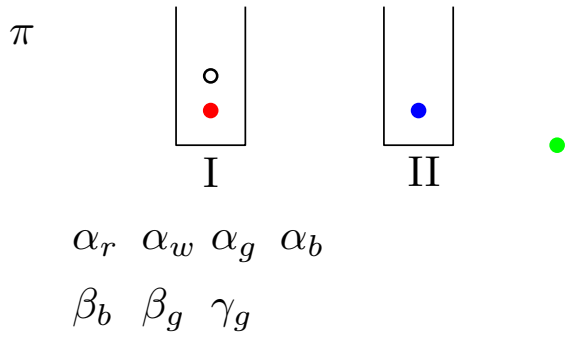
Output: y-coord. of buses

Constr.: on diagonal (π, π')

$\Rightarrow$ Test $O(n^2)$ time

- via 2-stack pushall sorting: sort RWGB to GWRB





5. Restricted Type of Point Set: Diagonal BEP

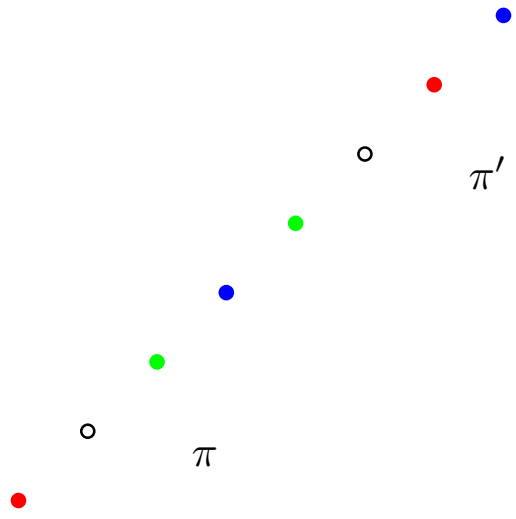
Input: 2 points per color

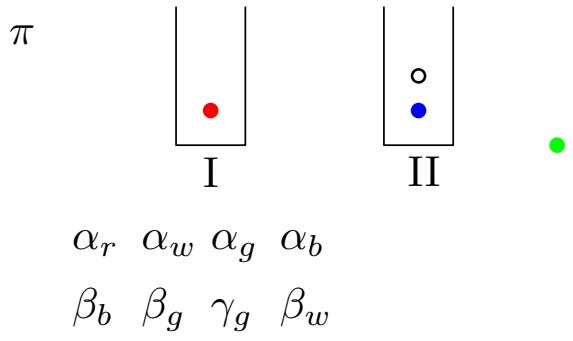
Output: y-coord. of buses

Constr.: on diagonal (π, π')

$\Rightarrow$ Test $O(n^2)$ time

- via 2-stack pushall sorting: sort RWGB to GWRB





5. Restricted Type of Point Set: Diagonal BEP

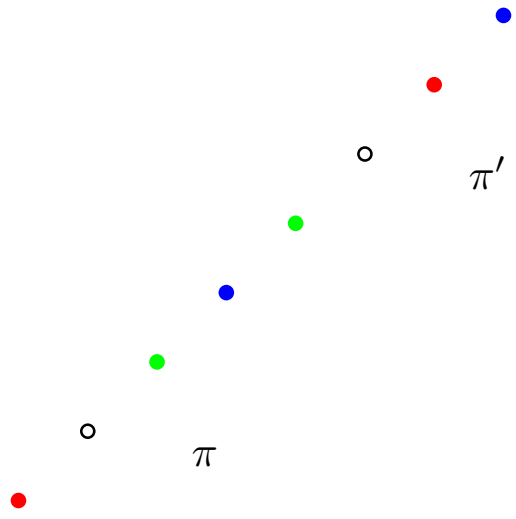
Input: 2 points per color

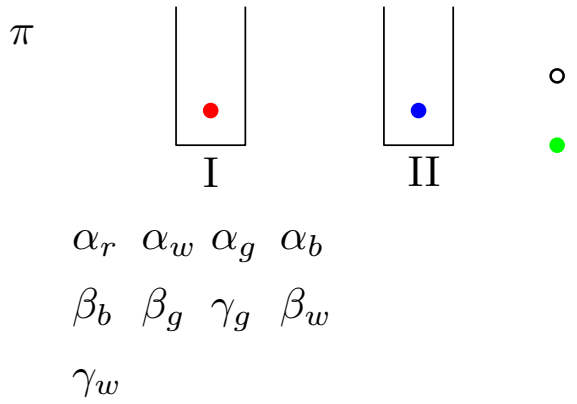
Output: y-coord. of buses

Constr.: on diagonal (π, π')

$\Rightarrow$ Test $O(n^2)$ time

- via 2-stack pushall sorting: sort RWGB to GWRB





5. Restricted Type of Point Set: Diagonal BEP

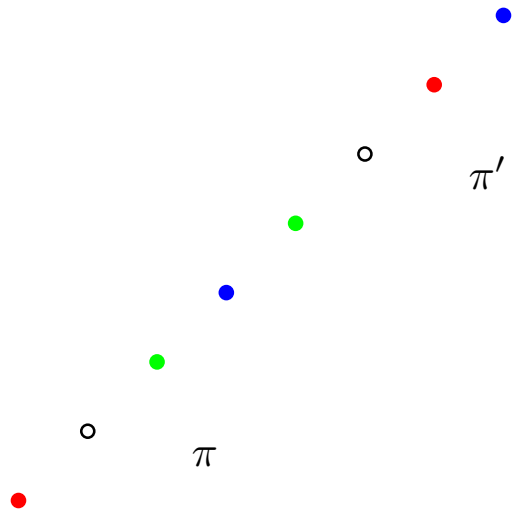
Input: 2 points per color

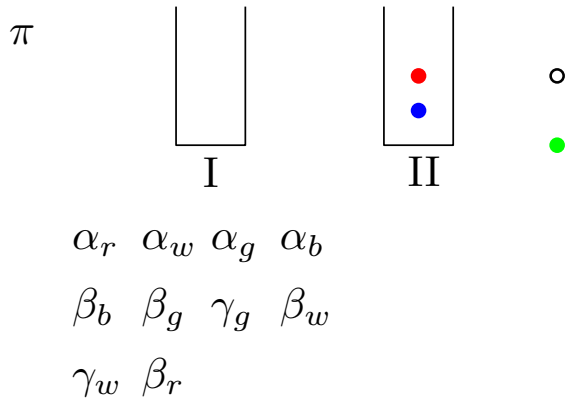
Output: y-coord. of buses

Constr.: on diagonal (π, π')

$\Rightarrow$ Test $O(n^2)$ time

- via 2-stack pushall sorting: sort RWGB to GWRB





5. Restricted Type of Point Set: Diagonal BEP

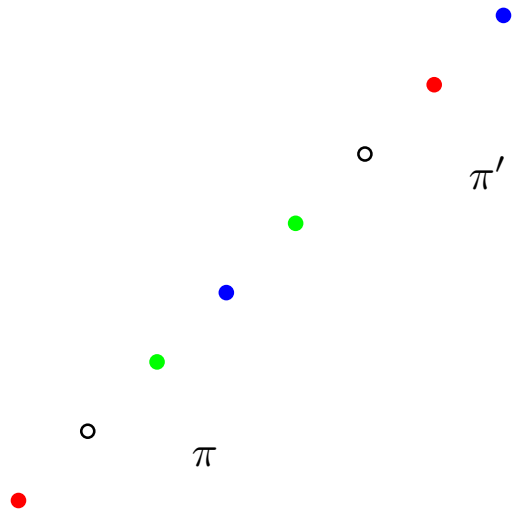
Input: 2 points per color

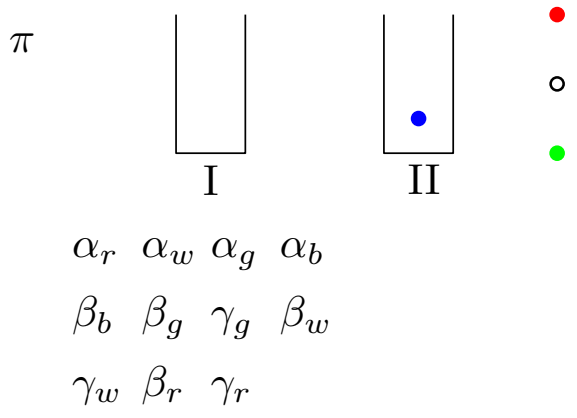
Output: y-coord. of buses

Constr.: on diagonal (π, π')

$\Rightarrow$ Test $O(n^2)$ time

- via 2-stack pushall sorting: sort RWGB to GWRB





5. Restricted Type of Point Set: Diagonal BEP

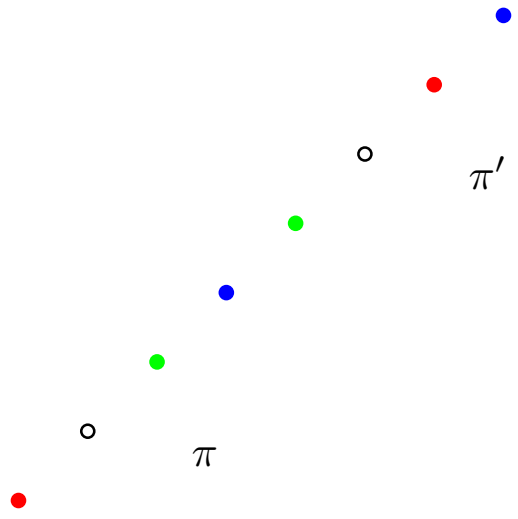
Input: 2 points per color

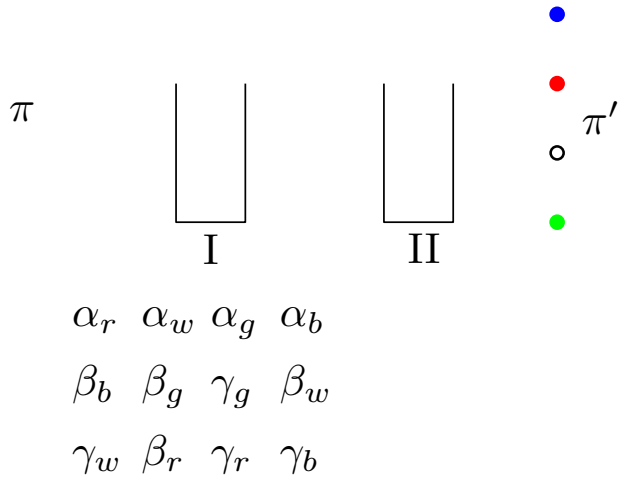
Output: y-coord. of buses

Constr.: on diagonal (π, π')

$\Rightarrow$ Test $O(n^2)$ time

- via 2-stack pushall sorting: sort RWGB to GWRB





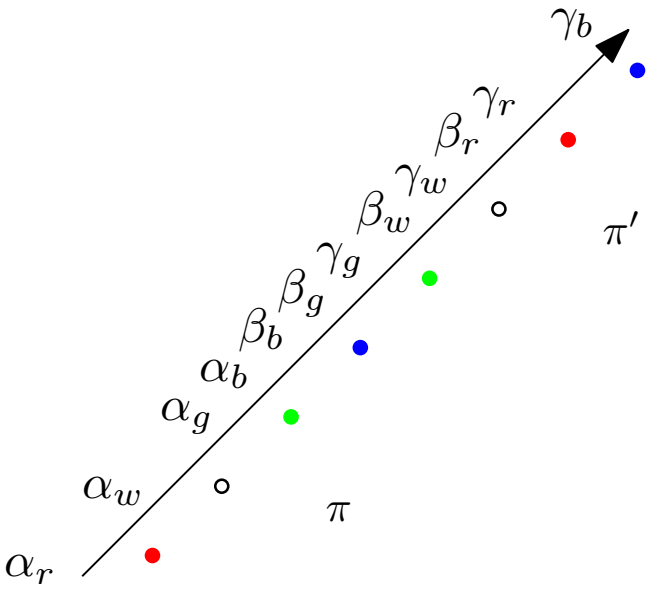
5. Restricted Type of Point Set: Diagonal BEP

Input: 2 points per color

Output: y-coord. of buses

Constr.: on diagonal (π, π')

$\Rightarrow$ Test $O(n^2)$ time



- via 2-stack pushall sorting: sort RWGB to GWRB

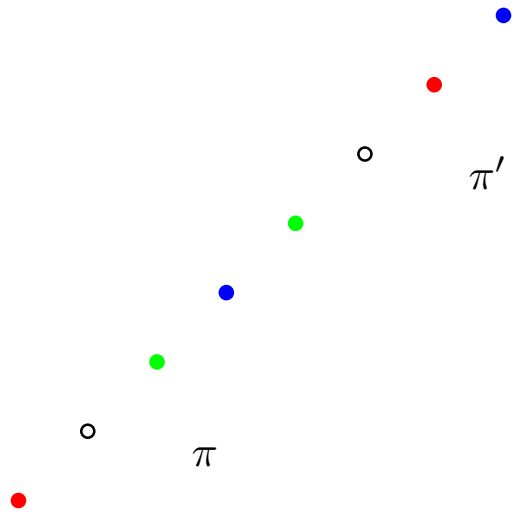
5. Restricted Type of Point Set: Diagonal BEP

Input: 2 points per color

Output: y-coord. of buses

Constr.: on diagonal (π, π')

$\Rightarrow$ Test $O(n^2)$ time



- via 2-stack pushall sorting: sort RWGB to GWRB

$\alpha_r \ \alpha_w \ \alpha_g \ \alpha_b \ \beta_b \ \beta_g \ \gamma_g \ \beta_w \ \gamma_w \ \beta_r \ \gamma_r \ \gamma_b$

α = pop from input and push on stack I
$\hat{=}$ connection up

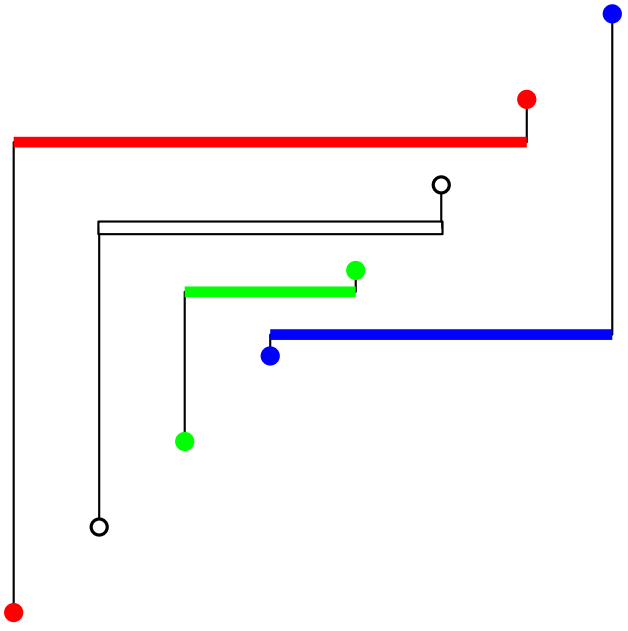
β = pop from stack I and push on stack II
$\hat{=}$ cross diagonal from left to right

γ = pop from stack II and push on putput
$\hat{=}$ connection up

5. Restricted Type of Point Set: Diagonal BEP

Input: 2 points per color
Output: y-coord. of buses
Constr.: on diagonal (π, π')
$\Rightarrow$ Test $O(n^2)$ time

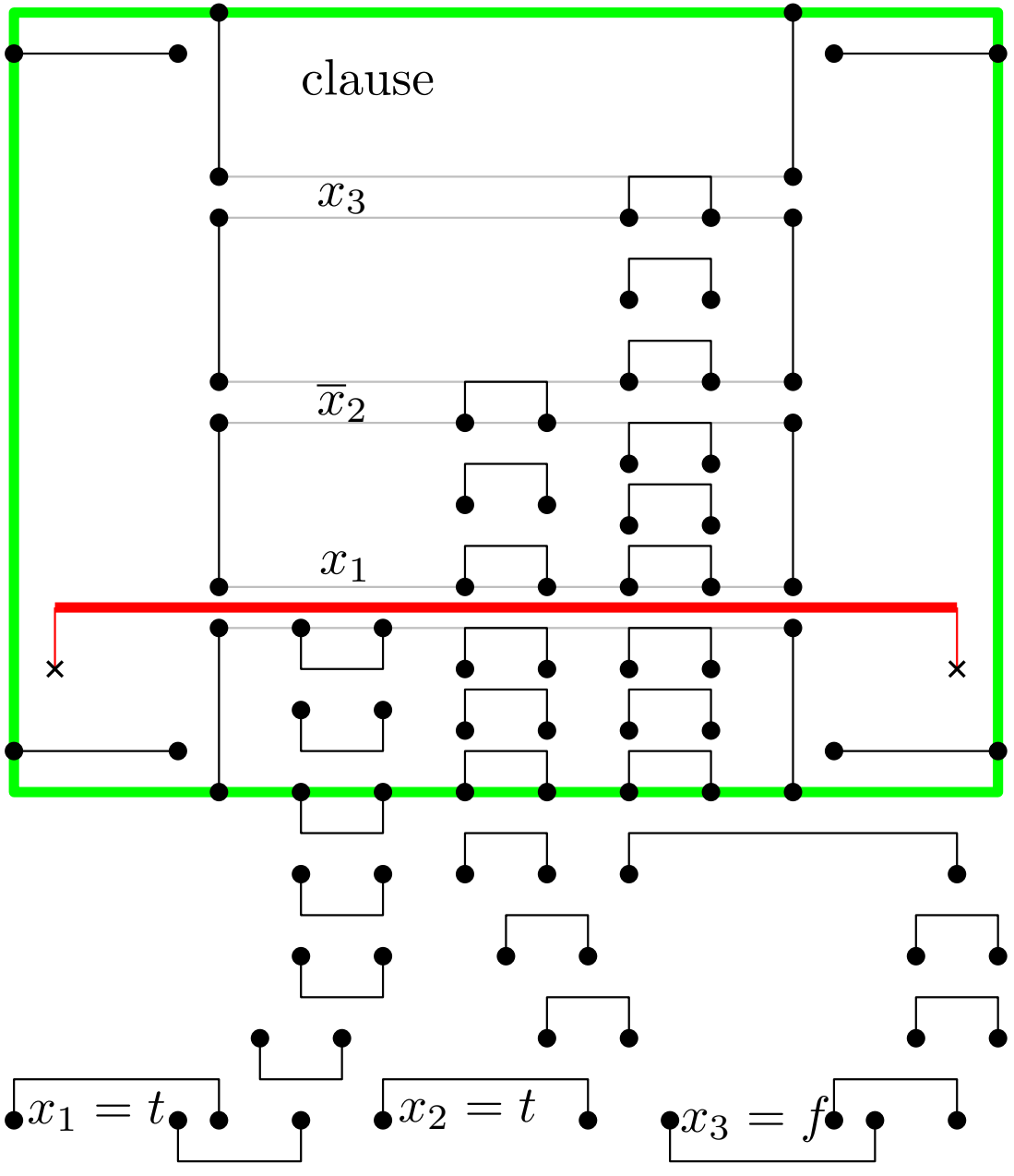
- via 2-stack pushall sorting: sort RWGB to GWRB



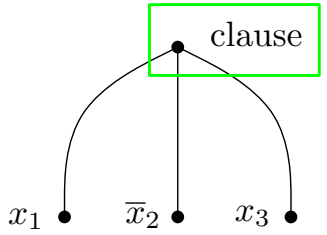
$\alpha_r \ \alpha_w \ \alpha_g \ \alpha_b \ \beta_b \ \beta_g \ \gamma_g \ \beta_w \ \gamma_w \ \beta_r \ \gamma_r \ \gamma_b$

α = pop from input and push on stack I
$\quad \hat{=}$ connection up
β = pop from stack I and push on stack II
$\quad \hat{=}$ cross diagonal from left to right
γ = pop from stack II and push on putput
$\quad \hat{=}$ connection up

6. Hardness of BEP$^{\varepsilon}$ ($\varepsilon \hat{=} \exists$ min. dist. buses $\leftrightarrow$ points)



- reduction from planar 3-SAT



→ ($\sqcap$,$\sqcup$)-BEP$^{\varepsilon}$

→ points may share a coordinate

→ 2 points per color

Conclusions and Future Work

- we used buses for visualization of set membership
- some variants of BEP are efficiently solvable
  (e.g. with 2-SAT or 2-stack pushall sorting)
- BEP seems to be hard in general

- we want to study non-planar settings:
  - minimize number of crossings
    minimize number of buses (stay planar)