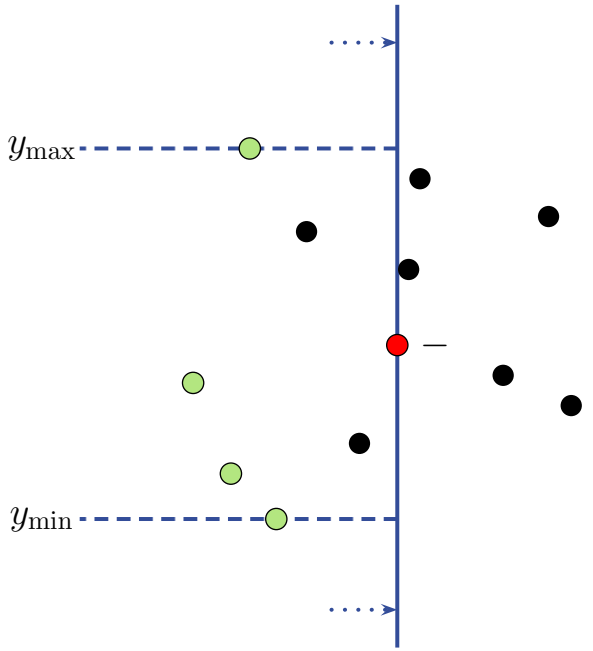
OPTYMALIZACJA KOMBINATORYCZNA

— TECHNIKA ZAMIATANIA —

OGÓLNA IDEA (na płaszczyźnie $\mathbb{R}^2$)

- Idea algorytmu zamiatania prostą polega na przesuwaniu w określonym kierunku prostej – miotły – po płaszczyźnie.
- Podczas takiego zamiatania utrzymywane są dodatkowe informacje – jest to tzw. status miotły.
- Status miotły zmienia się w punktach zdarzeń.



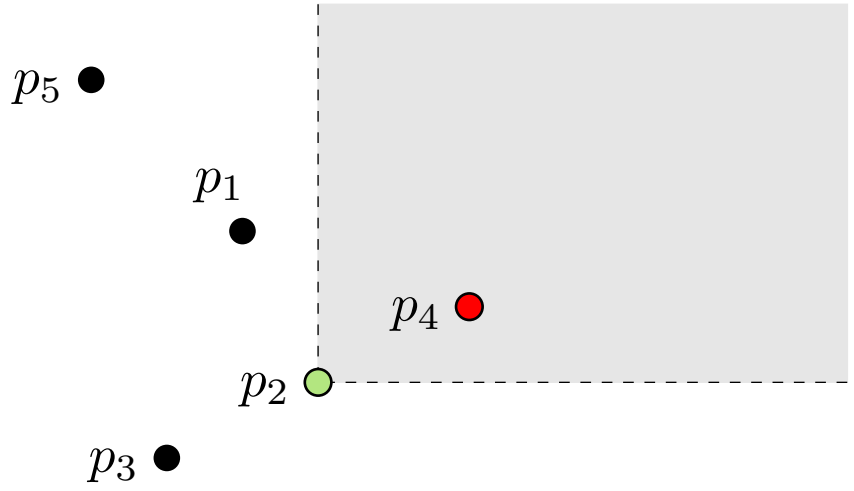
ZAGADNIENIA

1. Punkty maksymalne
2. Przecięcia odcinków poziomych/pionowych
3. Para najbliższych punktów
4.* Triangulacja wielokąta monotonicznego (**P.1**)
5. Otoczka wypukła
6.* Graf widzialności
7. Strzeżenie komórek

2.1 PUNKTY MAKSYMALNE

Dla dwóch punktów $p, q \in \mathbb{R}^2$ mówimy, że p dominuje nad punktem q, ozn. $q \prec p$, jeśli $\mathrm{x}(q) \leq \mathrm{x}(p)$ oraz $\mathrm{y}(q) \leq \mathrm{y}(p)$. Punkt $p \in S$ jest punktem maksymalnym (maksimum), jeśli nie istnieje żaden punkt $q \in S \setminus \{p\}$ taki, że $p \prec q$.

F.P. Preparata, M.I. Shamos
Geometria obliczeniowa — wprowadzenie
rozdział 4.1.3, Helion (2003)

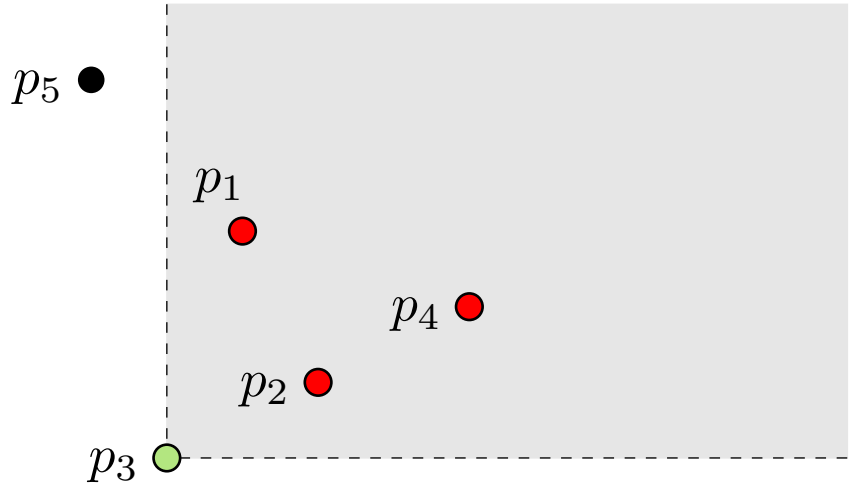


Przykład. Punkt p_2 jest zdominowany przez punkt p_4, a punkt p_3 jest zdominowany przez punkty p_1, p_2, p_4. Natomiast punkty p_1, p_4, p_5 są punktami maksymalnymi.

2.1 PUNKTY MAKSYMALNE

Dla dwóch punktów $p, q \in \mathbb{R}^2$ mówimy, że p dominuje nad punktem q, ozn. $q \prec p$, jeśli $\mathrm{x}(q) \leq \mathrm{x}(p)$ oraz $\mathrm{y}(q) \leq \mathrm{y}(p)$. Punkt $p \in S$ jest punktem maksymalnym (maksimum), jeśli nie istnieje żaden punkt $q \in S \setminus \{p\}$ taki, że $p \prec q$.

F.P. Preparata, M.I. Shamos
Geometria obliczeniowa — wprowadzenie
rozdział 4.1.3, Helion (2003)

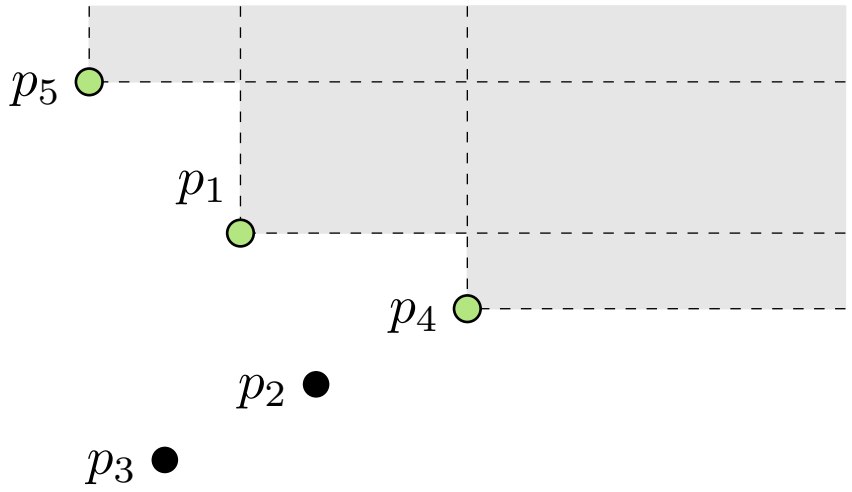


Przykład. Punkt p_2 jest zdominowany przez punkt p_4, a punkt p_3 jest zdominowany przez punkty p_1, p_2, p_4. Natomiast punkty p_1, p_4, p_5 są punktami maksymalnymi.

2.1 PUNKTY MAKSYMALNE

Dla dwóch punktów $p, q \in \mathbb{R}^2$ mówimy, że p dominuje nad punktem q, ozn. $q \prec p$, jeśli $\mathrm{x}(q) \leq \mathrm{x}(p)$ oraz $\mathrm{y}(q) \leq \mathrm{y}(p)$. Punkt $p \in S$ jest punktem maksymalnym (maksimum), jeśli nie istnieje żaden punkt $q \in S \setminus \{p\}$ taki, że $p \prec q$.

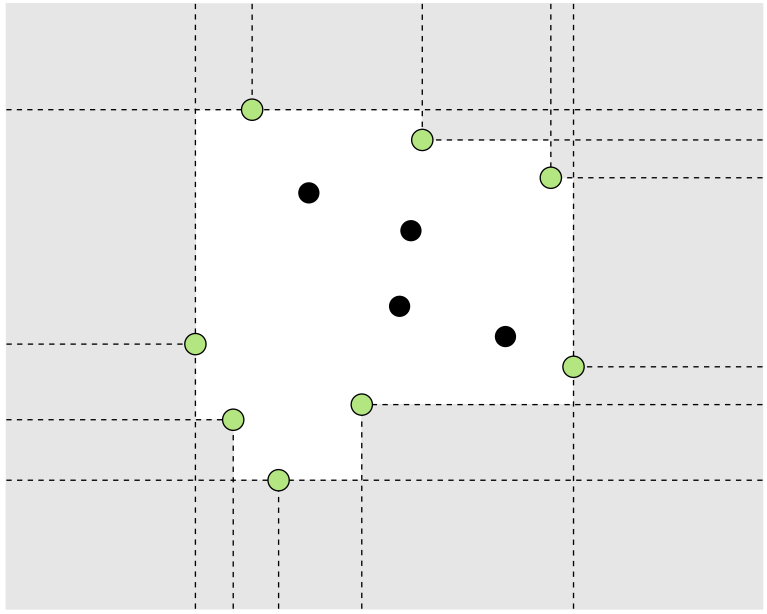
F.P. Preparata, M.I. Shamos
Geometria obliczeniowa — wprowadzenie
rozdział 4.1.3, Helion (2003)



Przykład. Punkt p_2 jest zdominowany przez punkt p_4, a punkt p_3 jest zdominowany przez punkty p_1, p_2, p_4. Natomiast punkty p_1, p_4, p_5 są punktami maksymalnymi.

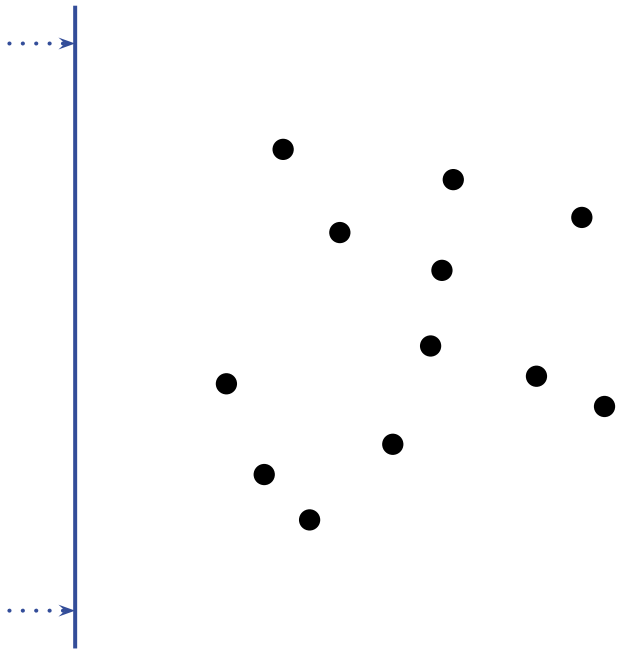
2.1 PUNKTY MAKSYMALNE

Dla dwóch punktów $p, q \in \mathbb{R}^2$ mówimy, że p dominuje nad punktem q, ozn. $q \prec p$, jeśli $\mathrm{x}(q) \leq \mathrm{x}(p)$ oraz $\mathrm{y}(q) \leq \mathrm{y}(p)$. Punkt $p \in S$ jest punktem maksymalnym (maksimum), jeśli nie istnieje żaden punkt $q \in S \setminus \{p\}$ taki, że $p \prec q$.



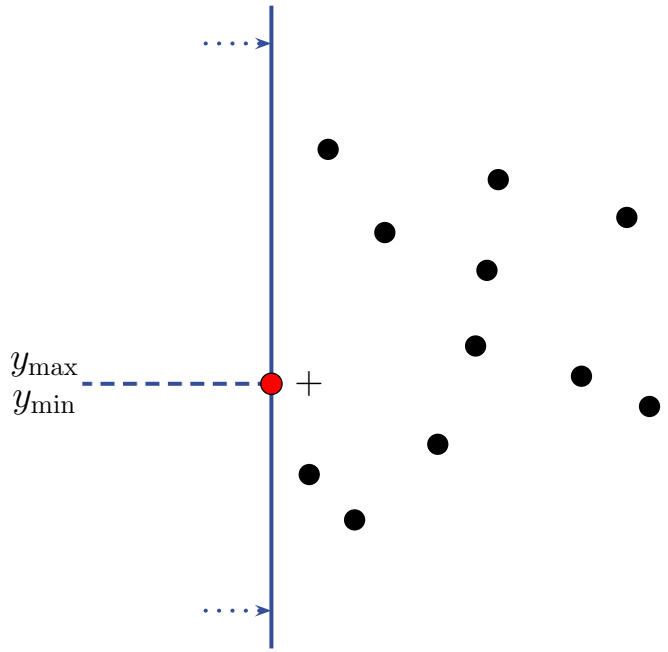
Dla zbioru $S \subset \mathbb{R}^2$, przypisując poszczególnym współrzędnym punktów znaki $+$ i $-$, można zdefiniować cztery problemy wyznaczenia maksimum.

Problem. Dla danego zbioru punktów $S \subset \mathbb{R}^2$ wyznacz wszystkie jego punkty maksymalne, po wszystkich czterech możliwych przypisaniach znaków.



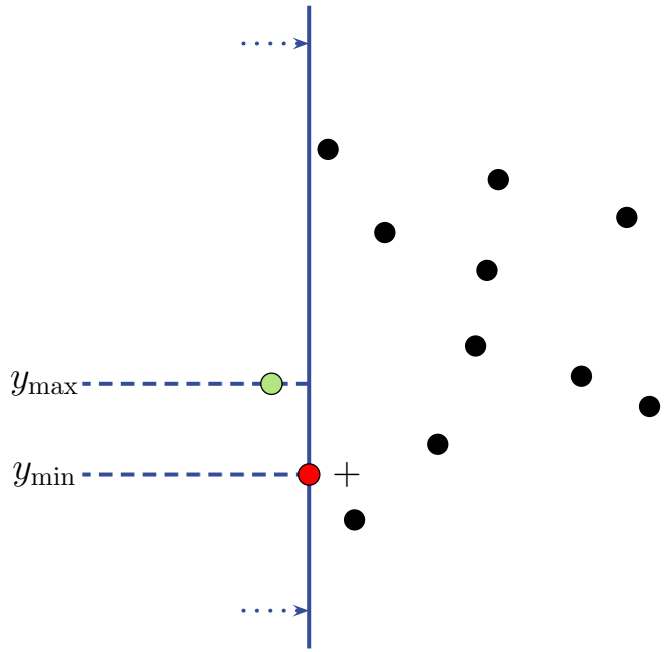
Idea algorytmu /Kung, Luccio, Preparata 1975/

- Przesuwamy (wirtualną) miotłę z lewa na prawo.
- Punktami zdarzeń są punkty ze zbioru wejściowego w kolejności rosnących współrzędnych x, przy czym z punktów o tej samej odciętej rozpatrujemy tylko te o najmniejszej i największej rzędnej.
- Statusem prostej jest najmniejsza i największa wartość współrzędnej y spośród wszystkich punktów na lewo od lub na miotle.
- Jeśli status prostej ulega zmianie, zgłaszane są odpowiednie punkty.
- W analogiczny sposób wszystkie punkty zostają przeglądnięte z prawa na lewo.



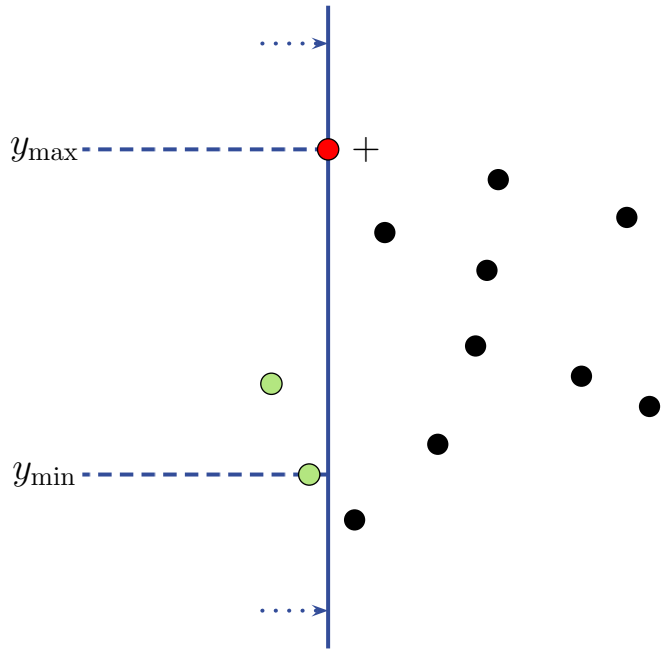
Idea algorytmu /Kung, Luccio, Preparata 1975/

- Przesuwamy (wirtualną) miotłę z lewa na prawo.
- Punktami zdarzeń są punkty ze zbioru wejściowego w kolejności rosnących współrzędnych x, przy czym z punktów o tej samej odciętej rozpatrujemy tylko te o najmniejszej i największej rzędnej.
- Statusem prostej jest najmniejsza i największa wartość współrzędnej y spośród wszystkich punktów na lewo od lub na miotle.
- Jeśli status prostej ulega zmianie, zgłaszane są odpowiednie punkty.
- W analogiczny sposób wszystkie punkty zostają przeglądnięte z prawa na lewo.



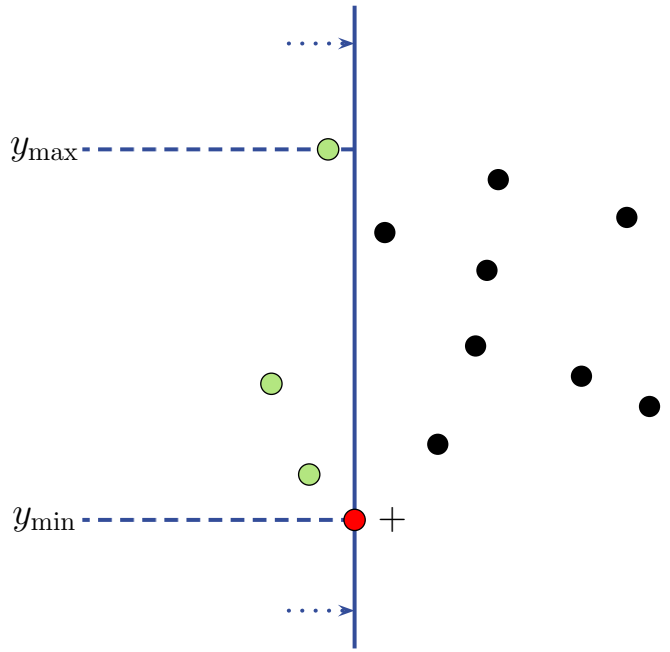
Idea algorytmu /Kung, Luccio, Preparata 1975/

- Przesuwamy (wirtualną) miotłę z lewa na prawo.
- Punktami zdarzeń są punkty ze zbioru wejściowego w kolejności rosnących współrzędnych x, przy czym z punktów o tej samej odciętej rozpatrujemy tylko te o najmniejszej i największej rzędnej.
- Statusem prostej jest najmniejsza i największa wartość współrzędnej y spośród wszystkich punktów na lewo od lub na miotle.
- Jeśli status prostej ulega zmianie, zgłaszane są odpowiednie punkty.
- W analogiczny sposób wszystkie punkty zostają przeglądnięte z prawa na lewo.



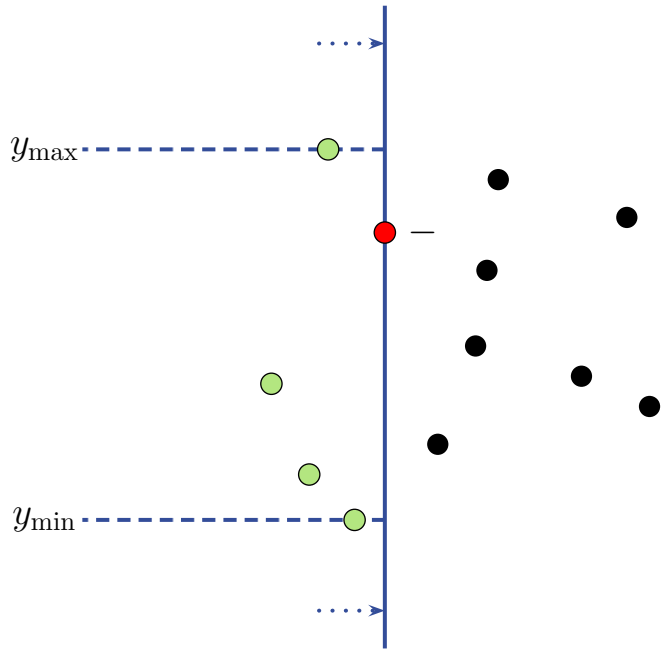
Idea algorytmu /Kung, Luccio, Preparata 1975/

- Przesuwamy (wirtualną) miotłę z lewa na prawo.
- Punktami zdarzeń są punkty ze zbioru wejściowego w kolejności rosnących współrzędnych x, przy czym z punktów o tej samej odciętej rozpatrujemy tylko te o najmniejszej i największej rzędnej.
- Statusem prostej jest najmniejsza i największa wartość współrzędnej y spośród wszystkich punktów na lewo od lub na miotle.
- Jeśli status prostej ulega zmianie, zgłaszane są odpowiednie punkty.
- W analogiczny sposób wszystkie punkty zostają przeglądnięte z prawa na lewo.



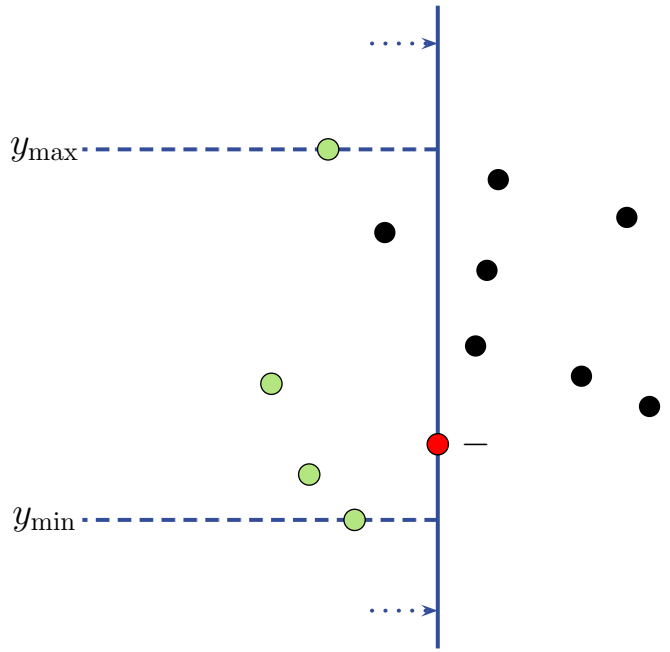
Idea algorytmu /Kung, Luccio, Preparata 1975/

- Przesuwamy (wirtualną) miotłę z lewa na prawo.
- Punktami zdarzeń są punkty ze zbioru wejściowego w kolejności rosnących współrzędnych x, przy czym z punktów o tej samej odciętej rozpatrujemy tylko te o najmniejszej i największej rzędnej.
- Statusem prostej jest najmniejsza i największa wartość współrzędnej y spośród wszystkich punktów na lewo od lub na miotle.
- Jeśli status prostej ulega zmianie, zgłaszane są odpowiednie punkty.
- W analogiczny sposób wszystkie punkty zostają przeglądnięte z prawa na lewo.



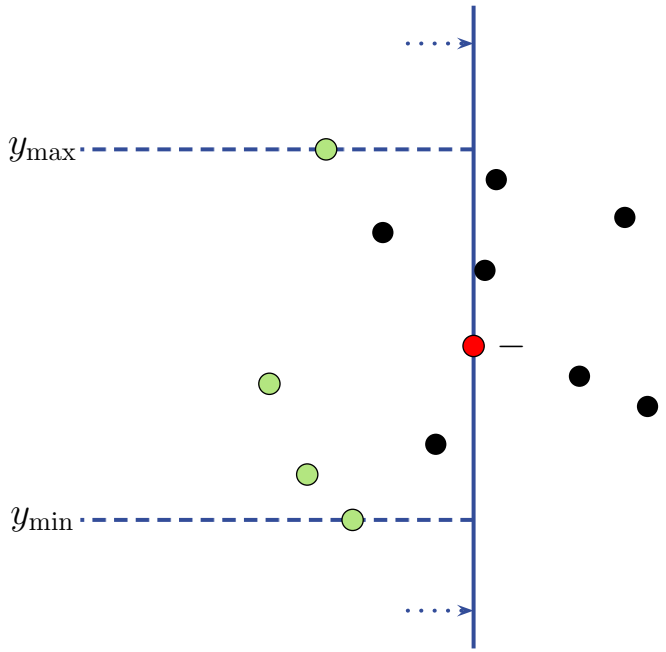
Idea algorytmu /Kung, Luccio, Preparata 1975/

- Przesuwamy (wirtualną) miotłę z lewa na prawo.
- Punktami zdarzeń są punkty ze zbioru wejściowego w kolejności rosnących współrzędnych x, przy czym z punktów o tej samej odciętej rozpatrujemy tylko te o najmniejszej i największej rzędnej.
- Statusem prostej jest najmniejsza i największa wartość współrzędnej y spośród wszystkich punktów na lewo od lub na miotle.
- Jeśli status prostej ulega zmianie, zgłaszane są odpowiednie punkty.
- W analogiczny sposób wszystkie punkty zostają przeglądnięte z prawa na lewo.



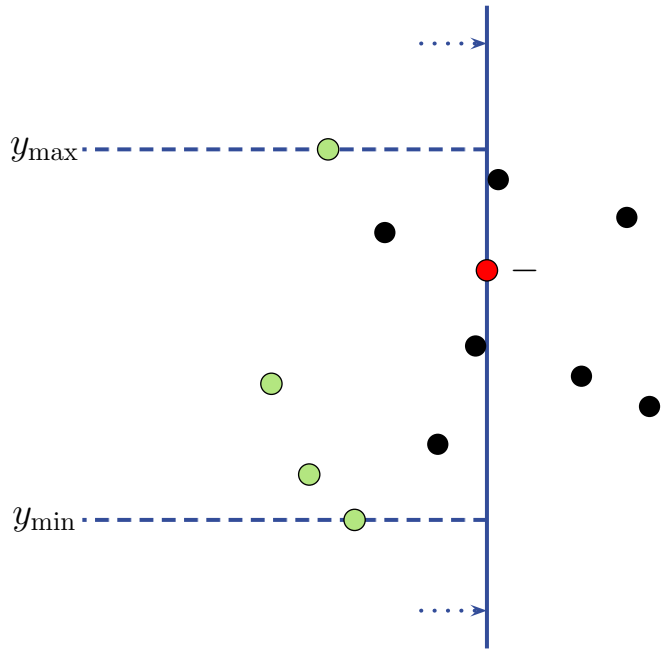
Idea algorytmu /Kung, Luccio, Preparata 1975/

- Przesuwamy (wirtualną) miotłę z lewa na prawo.
- Punktami zdarzeń są punkty ze zbioru wejściowego w kolejności rosnących współrzędnych x, przy czym z punktów o tej samej odciętej rozpatrujemy tylko te o najmniejszej i największej rzędnej.
- Statusem prostej jest najmniejsza i największa wartość współrzędnej y spośród wszystkich punktów na lewo od lub na miotle.
- Jeśli status prostej ulega zmianie, zgłaszane są odpowiednie punkty.
- W analogiczny sposób wszystkie punkty zostają przeglądnięte z prawa na lewo.



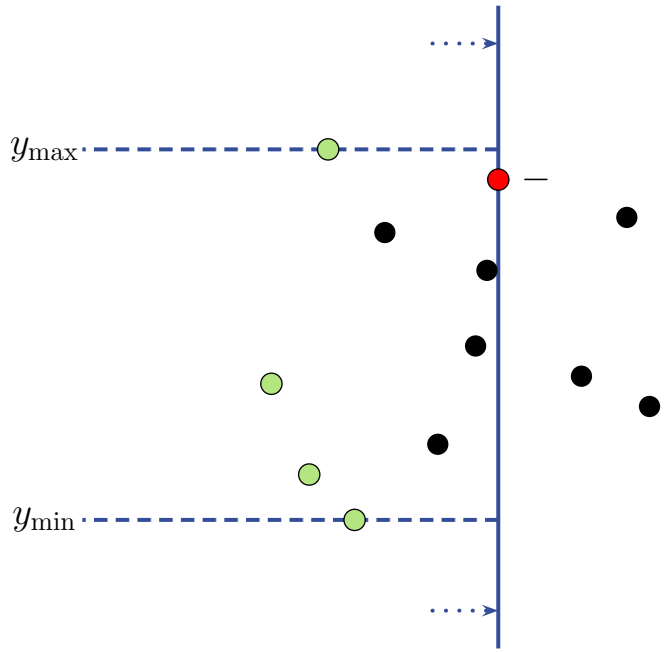
Idea algorytmu /Kung, Luccio, Preparata 1975/

- Przesuwamy (wirtualną) miotłę z lewa na prawo.
- Punktami zdarzeń są punkty ze zbioru wejściowego w kolejności rosnących współrzędnych x, przy czym z punktów o tej samej odciętej rozpatrujemy tylko te o najmniejszej i największej rzędnej.
- Statusem prostej jest najmniejsza i największa wartość współrzędnej y spośród wszystkich punktów na lewo od lub na miotle.
- Jeśli status prostej ulega zmianie, zgłaszane są odpowiednie punkty.
- W analogiczny sposób wszystkie punkty zostają przeglądnięte z prawa na lewo.



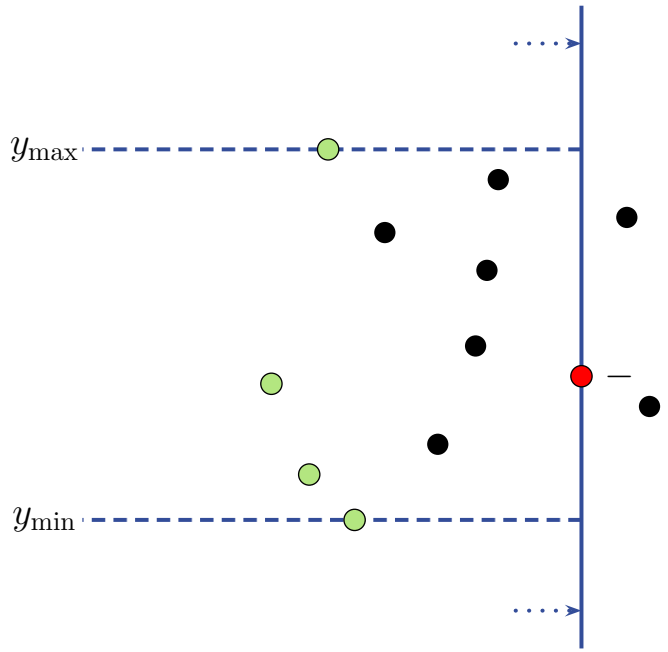
Idea algorytmu /Kung, Luccio, Preparata 1975/

- Przesuwamy (wirtualną) miotłę z lewa na prawo.
- Punktami zdarzeń są punkty ze zbioru wejściowego w kolejności rosnących współrzędnych x, przy czym z punktów o tej samej odciętej rozpatrujemy tylko te o najmniejszej i największej rzędnej.
- Statusem prostej jest najmniejsza i największa wartość współrzędnej y spośród wszystkich punktów na lewo od lub na miotle.
- Jeśli status prostej ulega zmianie, zgłaszane są odpowiednie punkty.
- W analogiczny sposób wszystkie punkty zostają przeglądnięte z prawa na lewo.



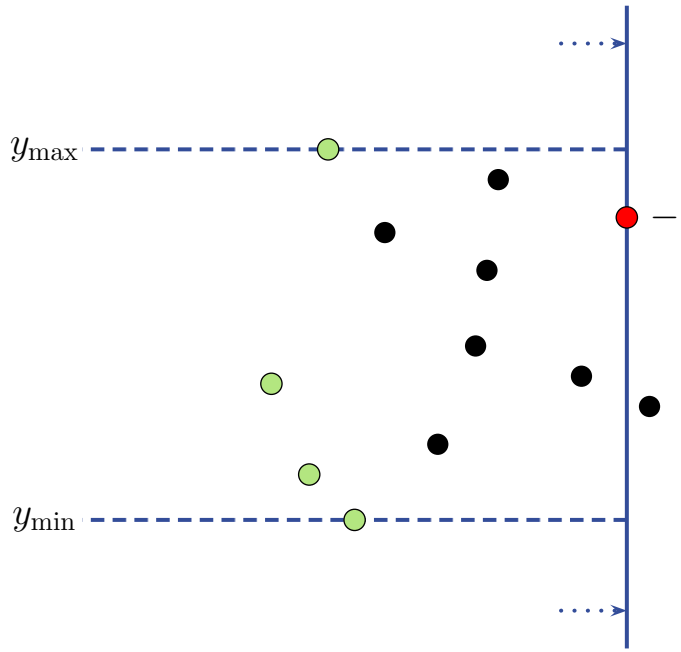
Idea algorytmu /Kung, Luccio, Preparata 1975/

- Przesuwamy (wirtualną) miotłę z lewa na prawo.
- Punktami zdarzeń są punkty ze zbioru wejściowego w kolejności rosnących współrzędnych x, przy czym z punktów o tej samej odciętej rozpatrujemy tylko te o najmniejszej i największej rzędnej.
- Statusem prostej jest najmniejsza i największa wartość współrzędnej y spośród wszystkich punktów na lewo od lub na miotle.
- Jeśli status prostej ulega zmianie, zgłaszane są odpowiednie punkty.
- W analogiczny sposób wszystkie punkty zostają przeglądnięte z prawa na lewo.



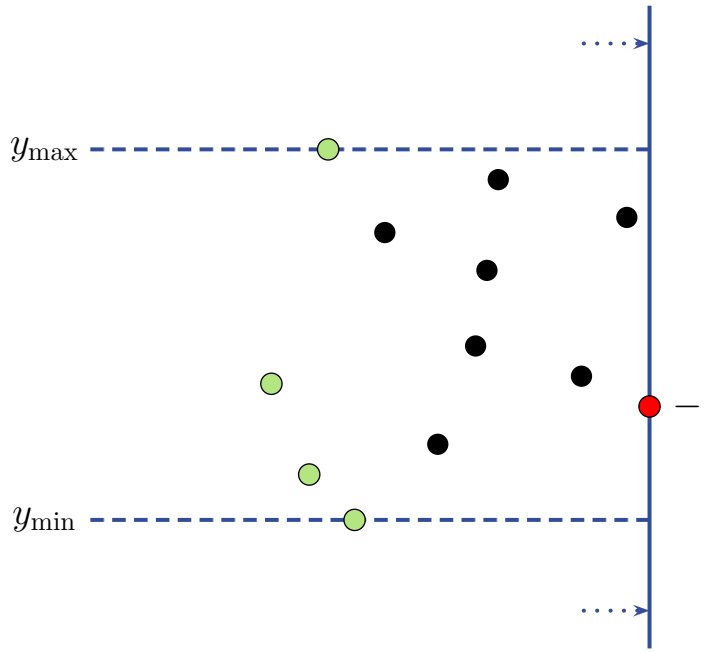
Idea algorytmu /Kung, Luccio, Preparata 1975/

- Przesuwamy (wirtualną) miotłę z lewa na prawo.
- Punktami zdarzeń są punkty ze zbioru wejściowego w kolejności rosnących współrzędnych x, przy czym z punktów o tej samej odciętej rozpatrujemy tylko te o najmniejszej i największej rzędnej.
- Statusem prostej jest najmniejsza i największa wartość współrzędnej y spośród wszystkich punktów na lewo od lub na miotle.
- Jeśli status prostej ulega zmianie, zgłaszane są odpowiednie punkty.
- W analogiczny sposób wszystkie punkty zostają przeglądnięte z prawa na lewo.



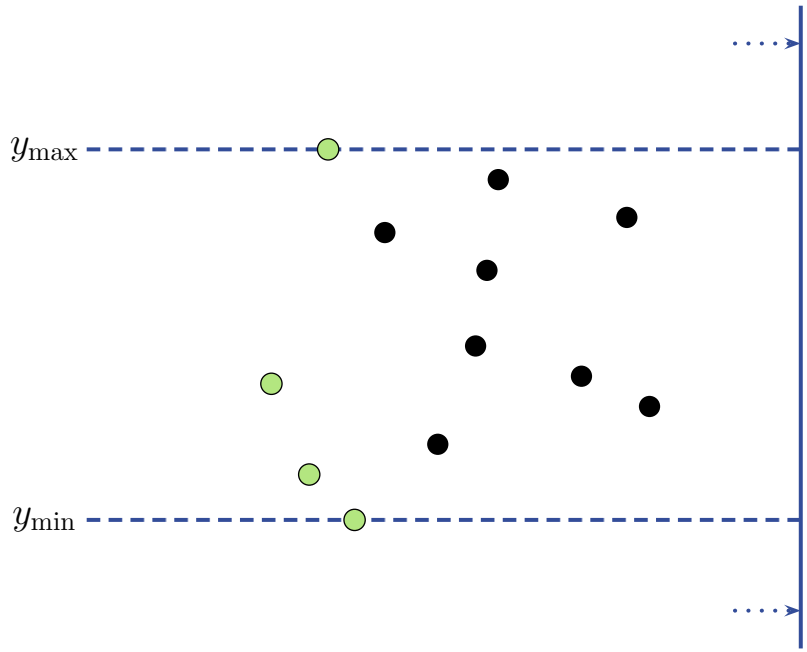
Idea algorytmu /Kung, Luccio, Preparata 1975/

- Przesuwamy (wirtualną) miotłę z lewa na prawo.
- Punktami zdarzeń są punkty ze zbioru wejściowego w kolejności rosnących współrzędnych x, przy czym z punktów o tej samej odciętej rozpatrujemy tylko te o najmniejszej i największej rzędnej.
- Statusem prostej jest najmniejsza i największa wartość współrzędnej y spośród wszystkich punktów na lewo od lub na miotle.
- Jeśli status prostej ulega zmianie, zgłaszane są odpowiednie punkty.
- W analogiczny sposób wszystkie punkty zostają przeglądnięte z prawa na lewo.



Idea algorytmu /Kung, Luccio, Preparata 1975/

- Przesuwamy (wirtualną) miotłę z lewa na prawo.
- Punktami zdarzeń są punkty ze zbioru wejściowego w kolejności rosnących współrzędnych x, przy czym z punktów o tej samej odciętej rozpatrujemy tylko te o najmniejszej i największej rzędnej.
- Statusem prostej jest najmniejsza i największa wartość współrzędnej y spośród wszystkich punktów na lewo od lub na miotle.
- Jeśli status prostej ulega zmianie, zgłaszane są odpowiednie punkty.
- W analogiczny sposób wszystkie punkty zostają przeglądnięte z prawa na lewo.



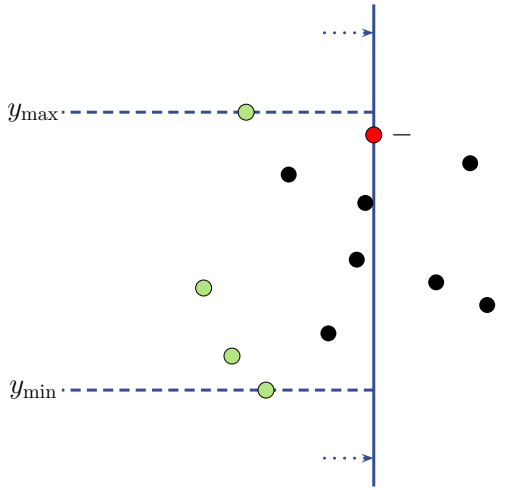
Idea algorytmu /Kung, Luccio, Preparata 1975/

- Przesuwamy (wirtualną) miotłę z lewa na prawo.
- Punktami zdarzeń są punkty ze zbioru wejściowego w kolejności rosnących współrzędnych x, przy czym z punktów o tej samej odciętej rozpatrujemy tylko te o najmniejszej i największej rzędnej.
- Statusem prostej jest najmniejsza i największa wartość współrzędnej y spośród wszystkich punktów na lewo od lub na miotle.
- Jeśli status prostej ulega zmianie, zgłaszane są odpowiednie punkty.
- W analogiczny sposób wszystkie punkty zostają przeglądnięte z prawa na lewo.

DetectMaxima(S) /wersja dla punktów o różnych współrzędnych x/

Wejście. Zbiór $S \subset \mathbb{R}^2$ – zbiór n punktów na płaszczyźnie.
Wyjście. Zbiór D elementów maksymalnych.

1. Posortuj S względem współrzędnej x, otrzymując punkty $p_1, p_2, \ldots, p_n$.
2. $D := \{p_1\}$; $y_{\max} := y_{\min} = \mathrm{y}(p_1)$.
3. **for** $i = 2$ **to** n **do**
4. **if** $\mathrm{y}(p_i) > y_{\max}$ **then** $D := D \cup \{p_i\}$; $y_{\max} := \mathrm{y}(p_i)$.
5. **if** $\mathrm{y}(p_i) < y_{\min}$ **then** $D := D \cup \{p_i\}$; $y_{\min} := \mathrm{y}(p_i)$.
6. $D := D \cup \{p_n\}$; $y_{\max} = y_{\min} = \mathrm{y}(p_n)$.
7. **for** $i = n - 1$ **to** 2 **do**
8. **if** $\mathrm{y}(p_i) > y_{\max}$ **then** $D := D \cup \{p_i\}$; $y_{\max} := \mathrm{y}(p_i)$.
9. **if** $\mathrm{y}(p_i) < y_{\min}$ **then** $D := D \cup \{p_i\}$; $y_{\min} := \mathrm{y}(p_i)$.
10. **return** D.



Poprawność podejścia.

- Niezmiennikiem algorytmu w wierszach 2-5 jest „Wszystkie maksima dla przypisań $(-,+)$ oraz $(-,-)$ na lewo od miotły M zostały zgłoszone”, natomiast niezmiennikiem w wierszach 6-9 jest „Wszystkie maksima dla przypisań $(+,+)$ oraz $(+,-)$ na prawo od miotły M zostały zgłoszone”.

Złożoność obliczeniowa algorytmu.

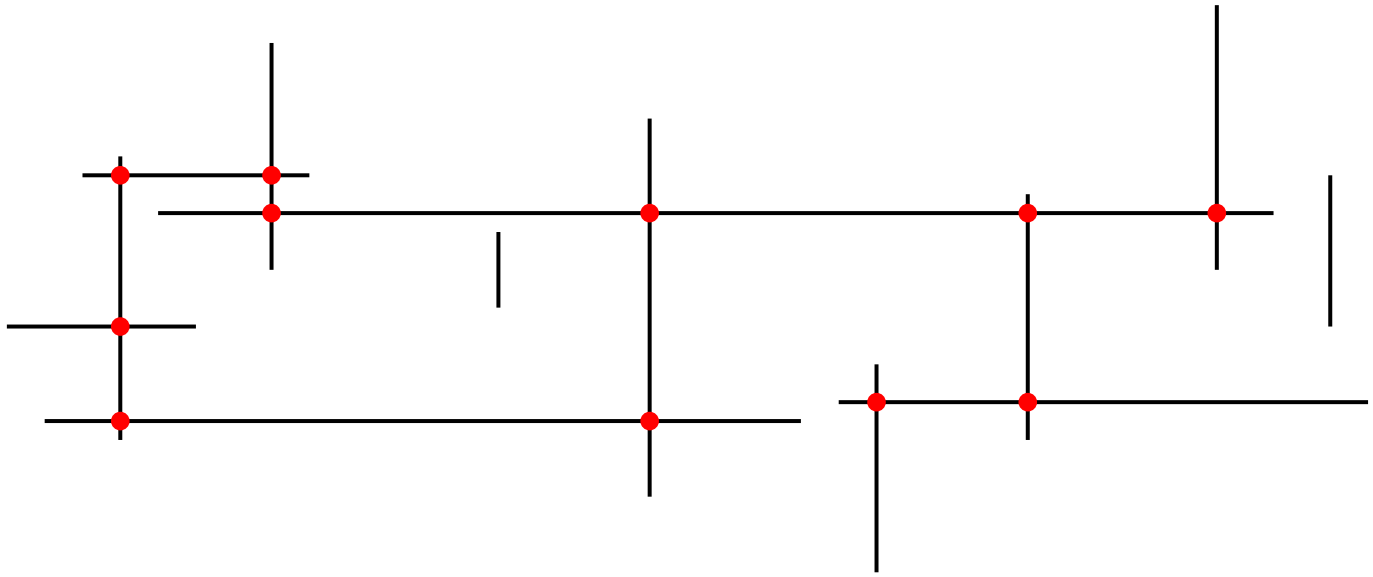
- Mając na uwadze wstępne posortowanie po odciętej: czas rzędu $O(n \log n)$.
- Zapamiętanie $y_{\min}$ oraz $y_{\max}$: pamięć rzędu $O(1)$.

2.2 PUNKTY PRZECIĘĆ

Dla danego zbioru poziomych i pionowych odcinków $S = H \cup V$ na płaszczyźnie wyznacz wszystkie punkty przecięć odcinków z S.

M. Smid

Computing intersections in a set of horizontal and vertical line segments
`http://people.scs.carleton.ca/~michiel/teaching.html` [23.03.2015]

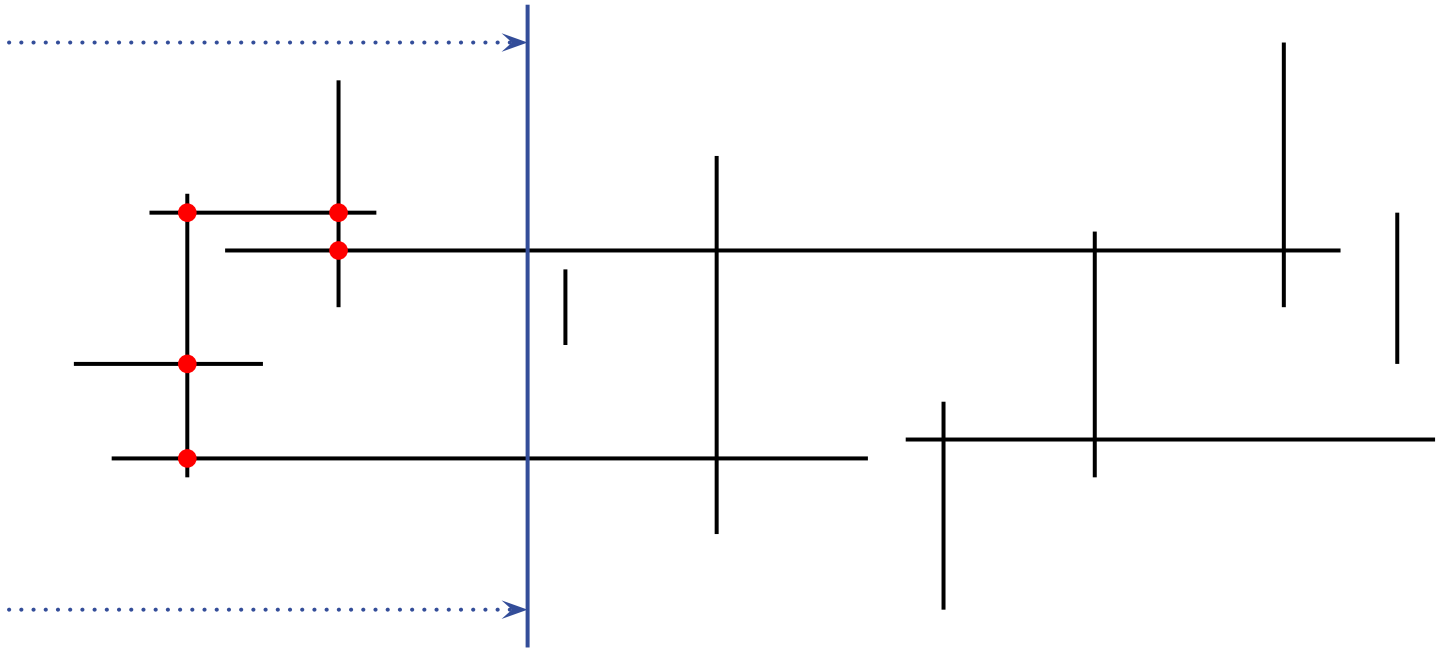


Założenie. Końce odcinków poziomych oraz odcinki pionowe mają różne odcięte.

2.2 PUNKTY PRZECIĘĆ

Problem. Dla danego zbioru poziomych i pionowych odcinków $S = H \cup V$ na płaszczyźnie wyznacz wszystkie punkty przecięć odcinków z S.

Niezmiennik. Wszystkie przecięcia na lewo od M zostały zgłoszone.



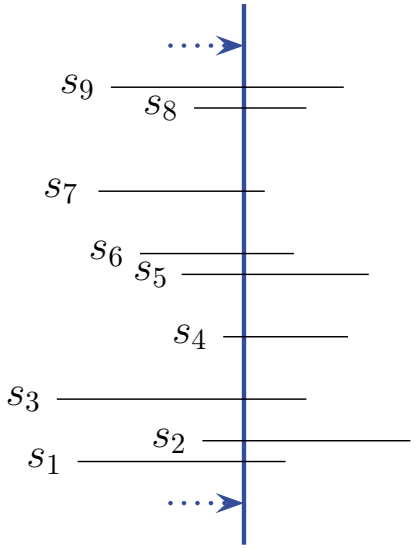
2.2 PUNKTY PRZECIĘĆ

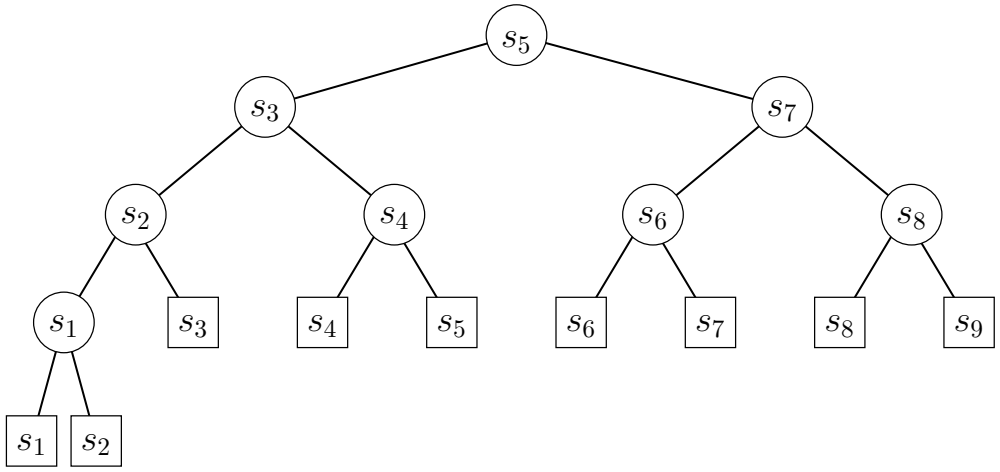
Problem. Dla danego zbioru poziomych i pionowych odcinków $S = H \cup V$ na płaszczyźnie wyznacz wszystkie punkty przecięć odcinków z S.

Niezmiennik. Wszystkie przecięcia na lewo od M zostały zgłoszone.

Status miotły. Poziome odcinki przecinane przez miotłę.

Status prostej przechowywany jest w zrównoważonym drzewie przeszukiwań; odcinki (w liściach) posortowane są względem współrzędnej y.





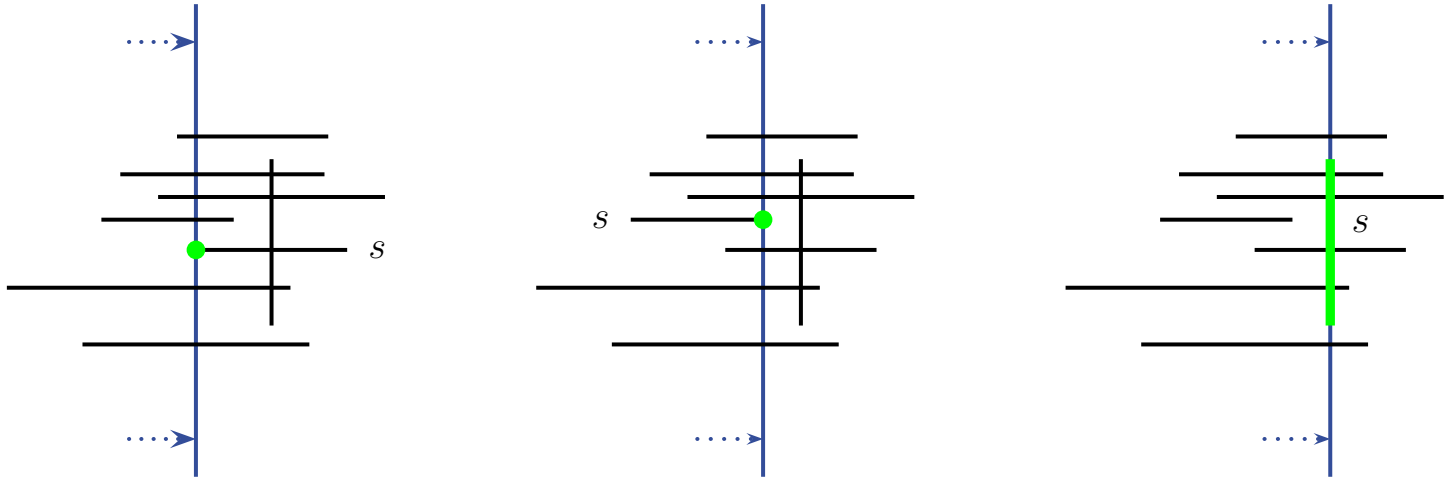
2.2 PUNKTY PRZECIĘĆ

Problem. Dla danego zbioru poziomych i pionowych odcinków $S = H \cup V$ na płaszczyźnie wyznacz wszystkie punkty przecięć odcinków z S.

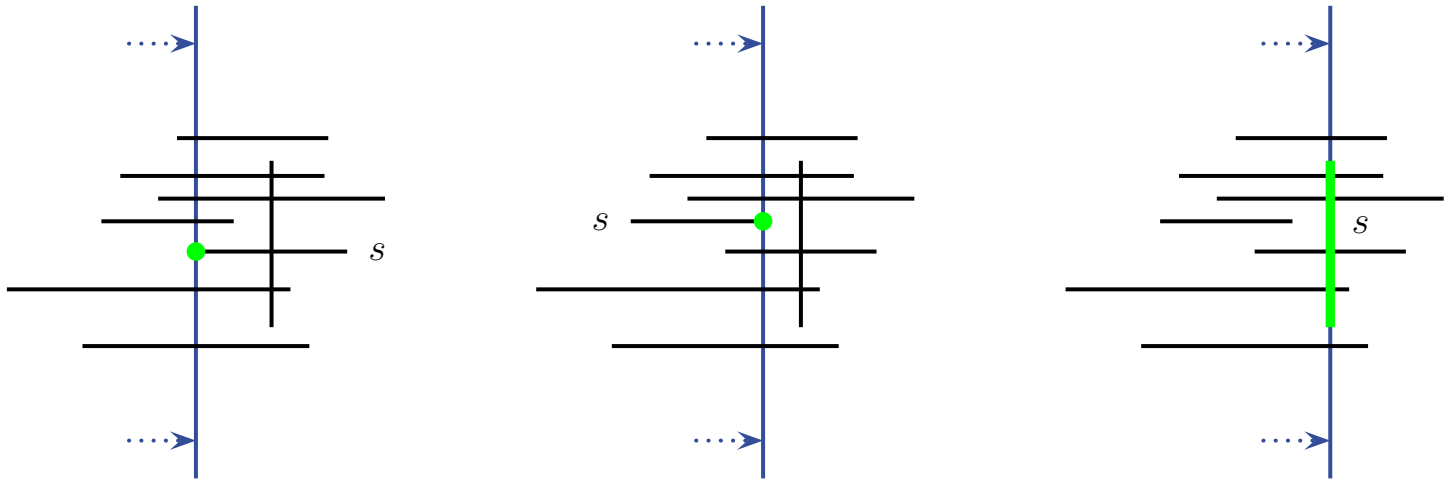
Niezmiennik. Wszystkie przecięcia na lewo od M zostały zgłoszone.

Status miotły. Poziome odcinki przecinane przez miotłę.

Status prostej przechowywany jest w zrównoważonym drzewie przeszukiwań; odcinki (w liściach) posortowane są względem współrzędnej y.



Punkty zdarzeń. Końce odcinków poziomych i odcinki pionowe.

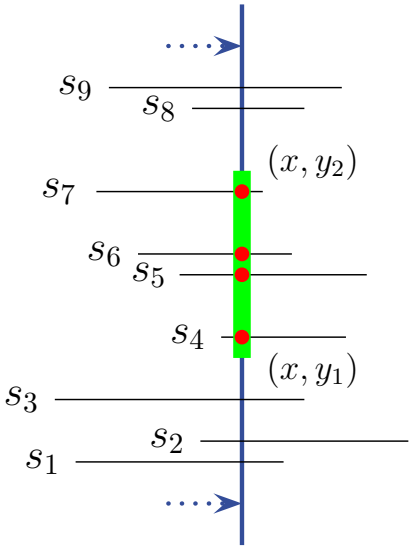


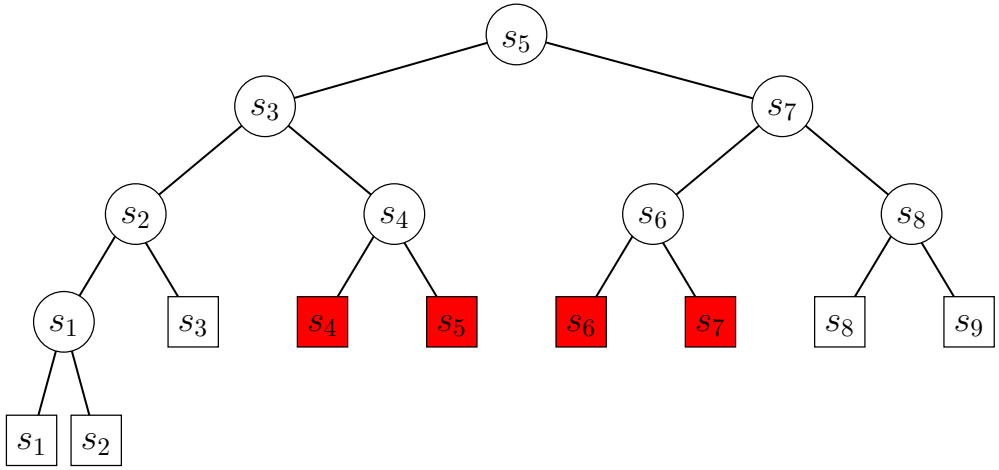
Zmiana statusu. Możliwe są trzy sytuacje:

1. Miotła napotyka lewy koniec poziomego odcinka s.
   Miotła zaczyna przecinać s, a zatem odcinek s musi zostać dodany do (zrównoważonego drzewa) statusu miotły.

2. Miotła napotyka prawy koniec poziomego odcinka s.
   Miotła przestaje przecinać s, a zatem odcinek s musi zostać usunięty ze (zrównoważonego drzewa) statusu miotły.

3. Miotła napotyka pionowy odcinek $s = [(x, y_1), (x, y_2)]$.
   Zauważmy, że poziome odcinki przecinające s muszą w tej chwili przecinać także miotłę M, a zatem, aby wyznaczyć wszystkie te przecięcia, wystarczy w zrównoważonym drzewie statusu prostej wyznaczyć te poziome odcinki, których współrzędne y należą do przedziału $[y_1, y_2]$.



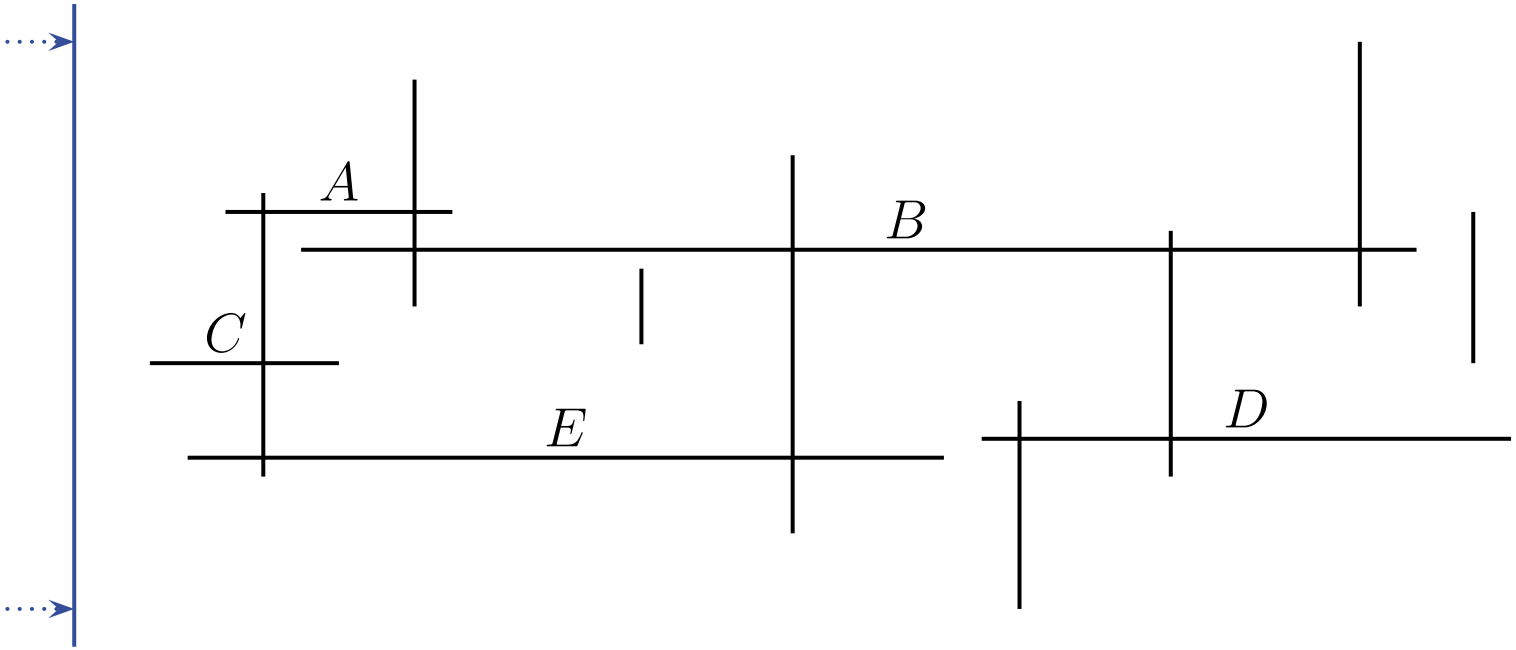


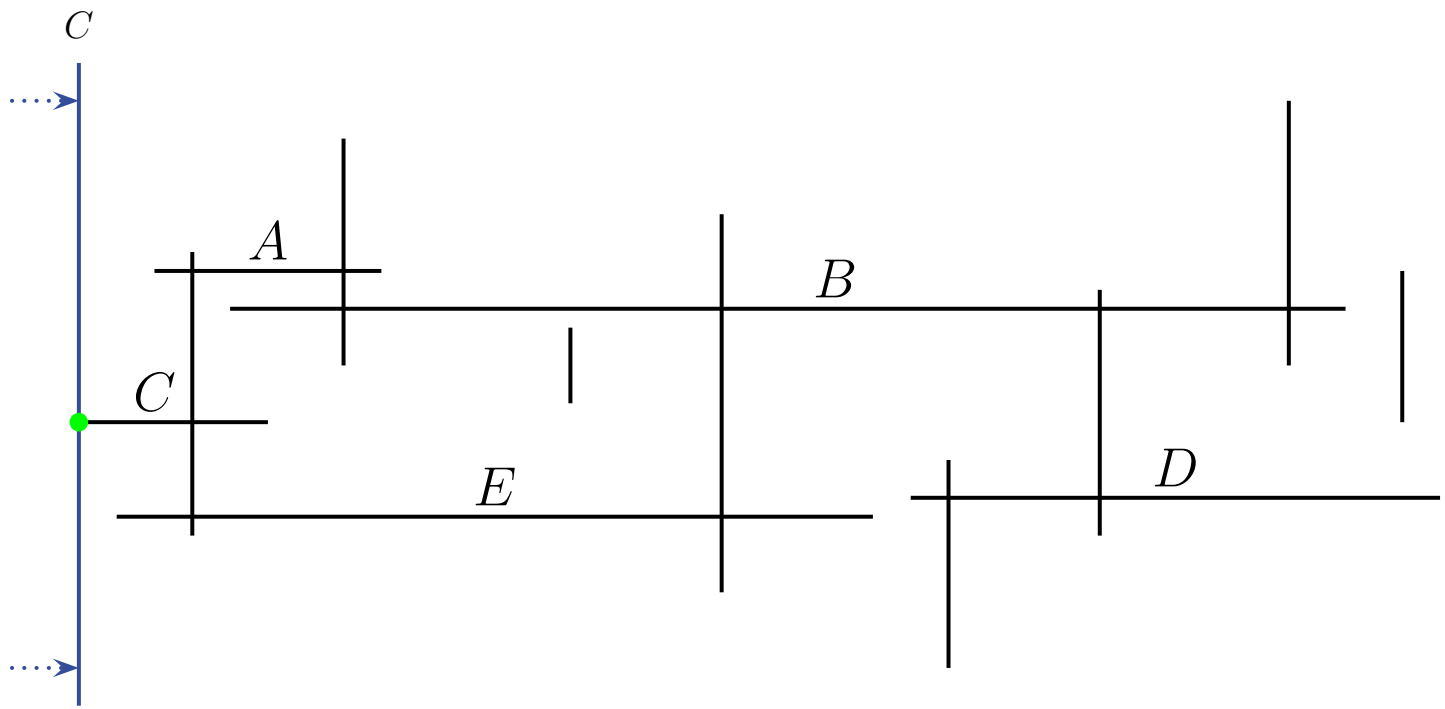
Zmiana statusu. Możliwe są trzy sytuacje:

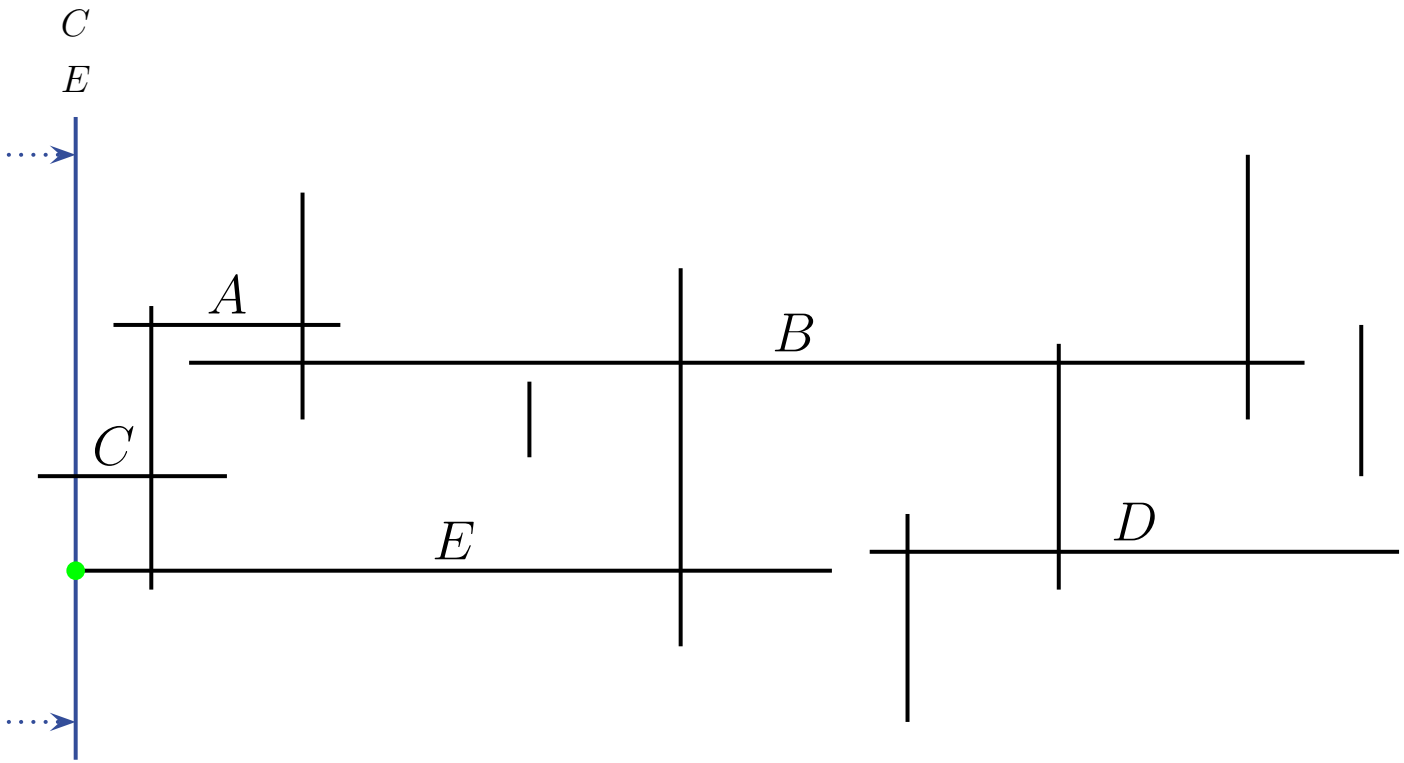
1. Miotła napotyka lewy koniec poziomego odcinka s.
   Miotła zaczyna przecinać s, a zatem odcinek s musi zostać dodany do (zrównoważonego drzewa) statusu miotły.

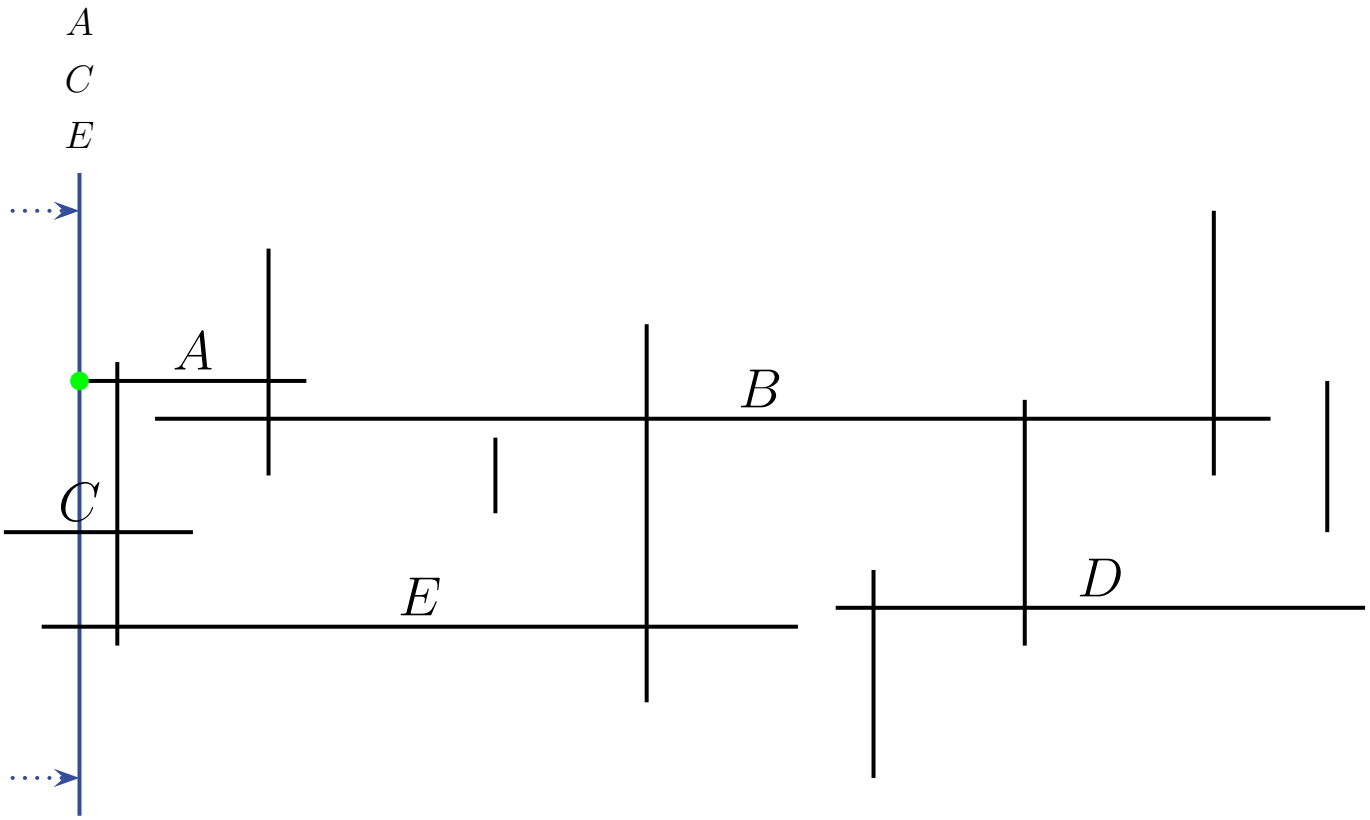
2. Miotła napotyka prawy koniec poziomego odcinka s.
   Miotła przestaje przecinać s, a zatem odcinek s musi zostać usunięty ze (zrównoważonego drzewa) statusu miotły.

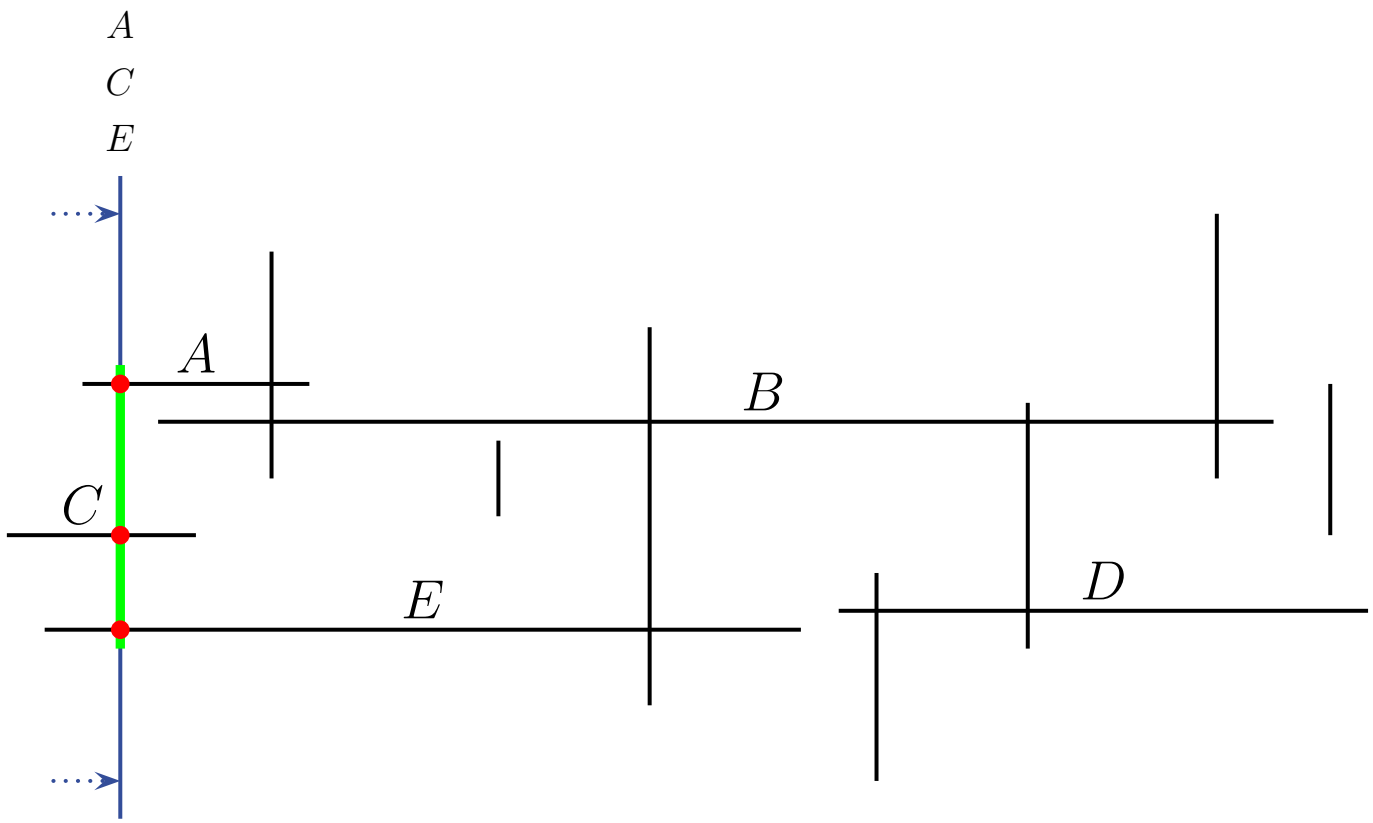
3. Miotła napotyka pionowy odcinek $s = [(x, y_1), (x, y_2)]$.
   Zauważmy, że poziome odcinki przecinające s muszą w tej chwili przecinać także miotłę M, a zatem, aby wyznaczyć wszystkie te przecięcia, wystarczy w zrównoważonym drzewie statusu prostej wyznaczyć te poziome odcinki, których współrzędne y należą do przedziału $[y_1, y_2]$.

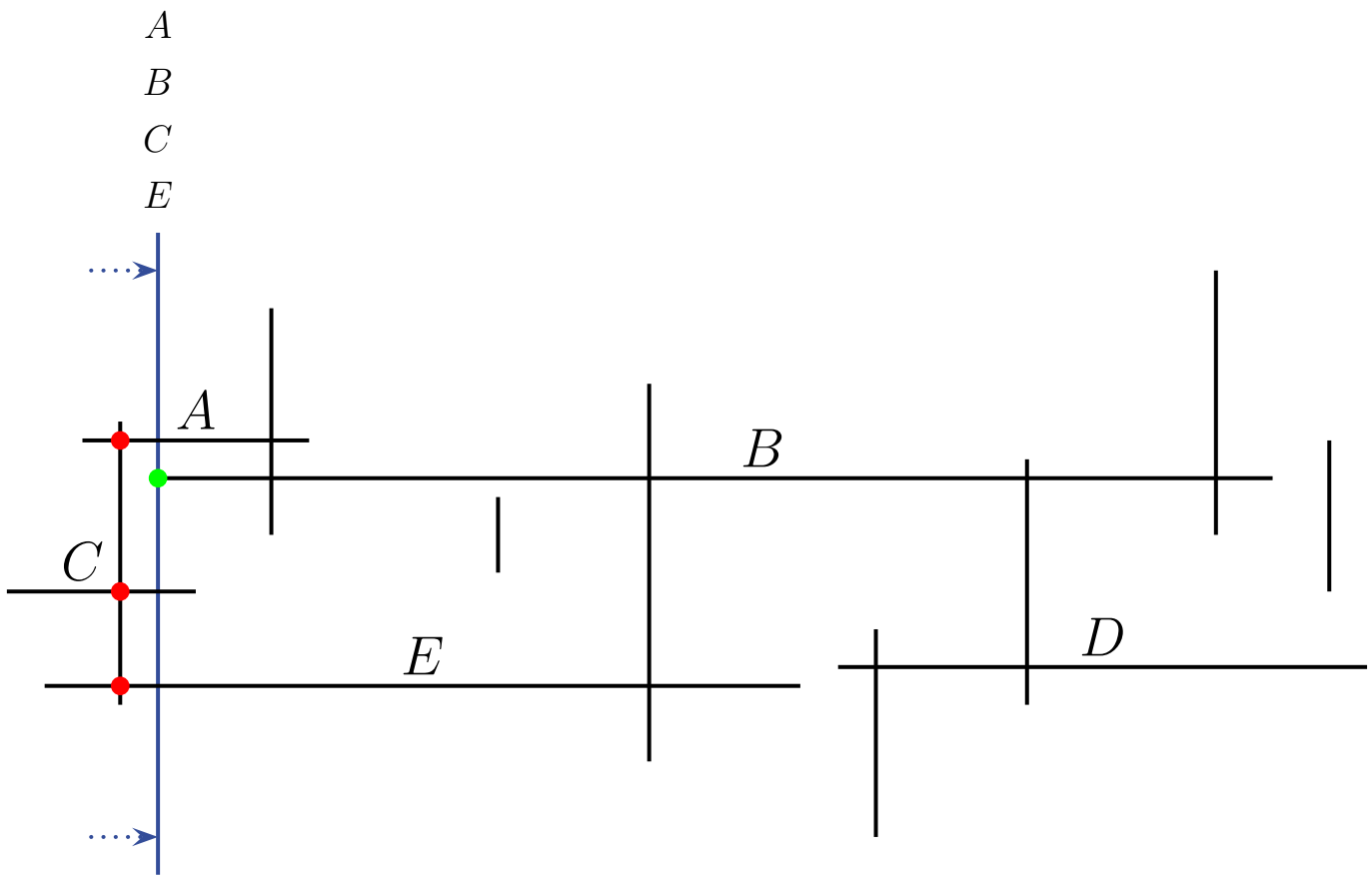

A
B
C
D
E

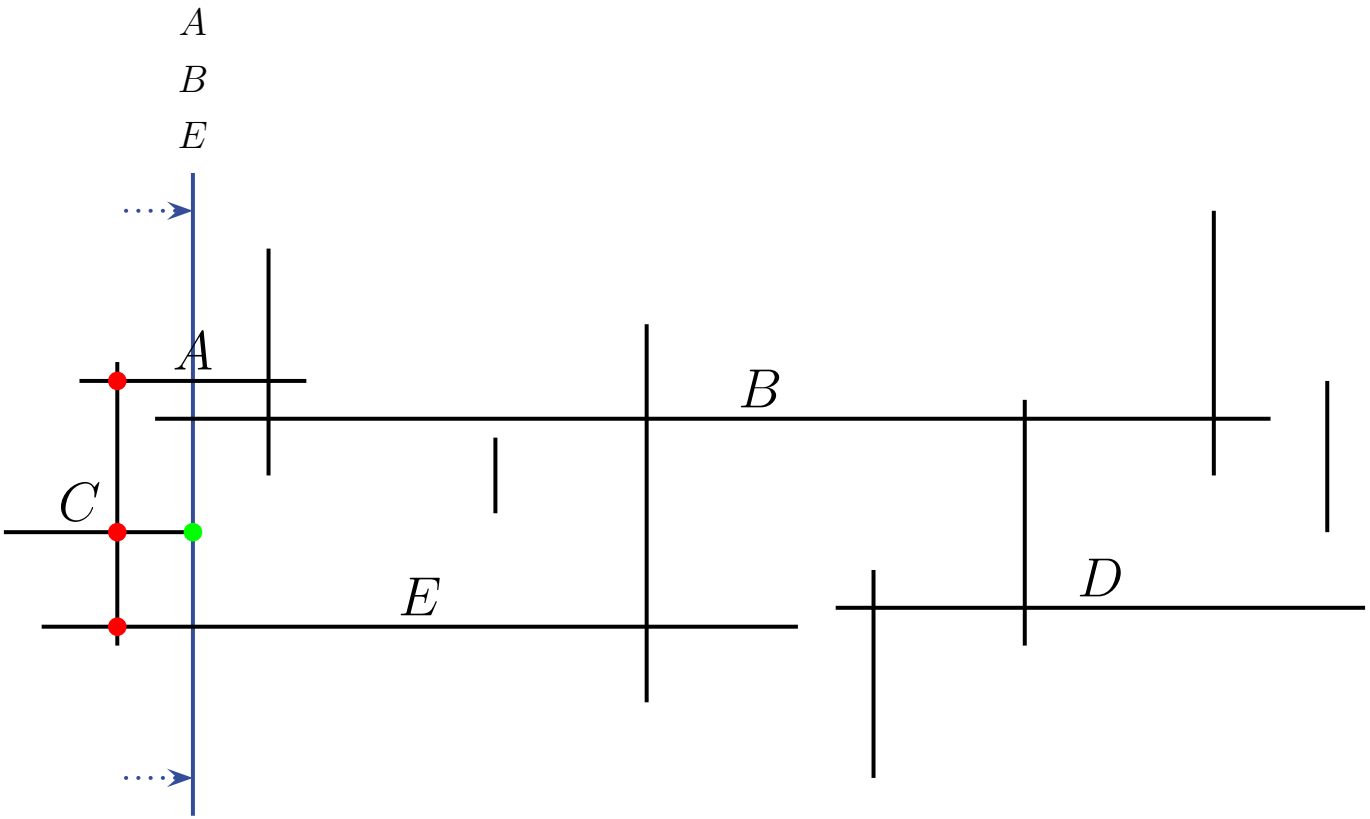

C
A
B
C
E
D

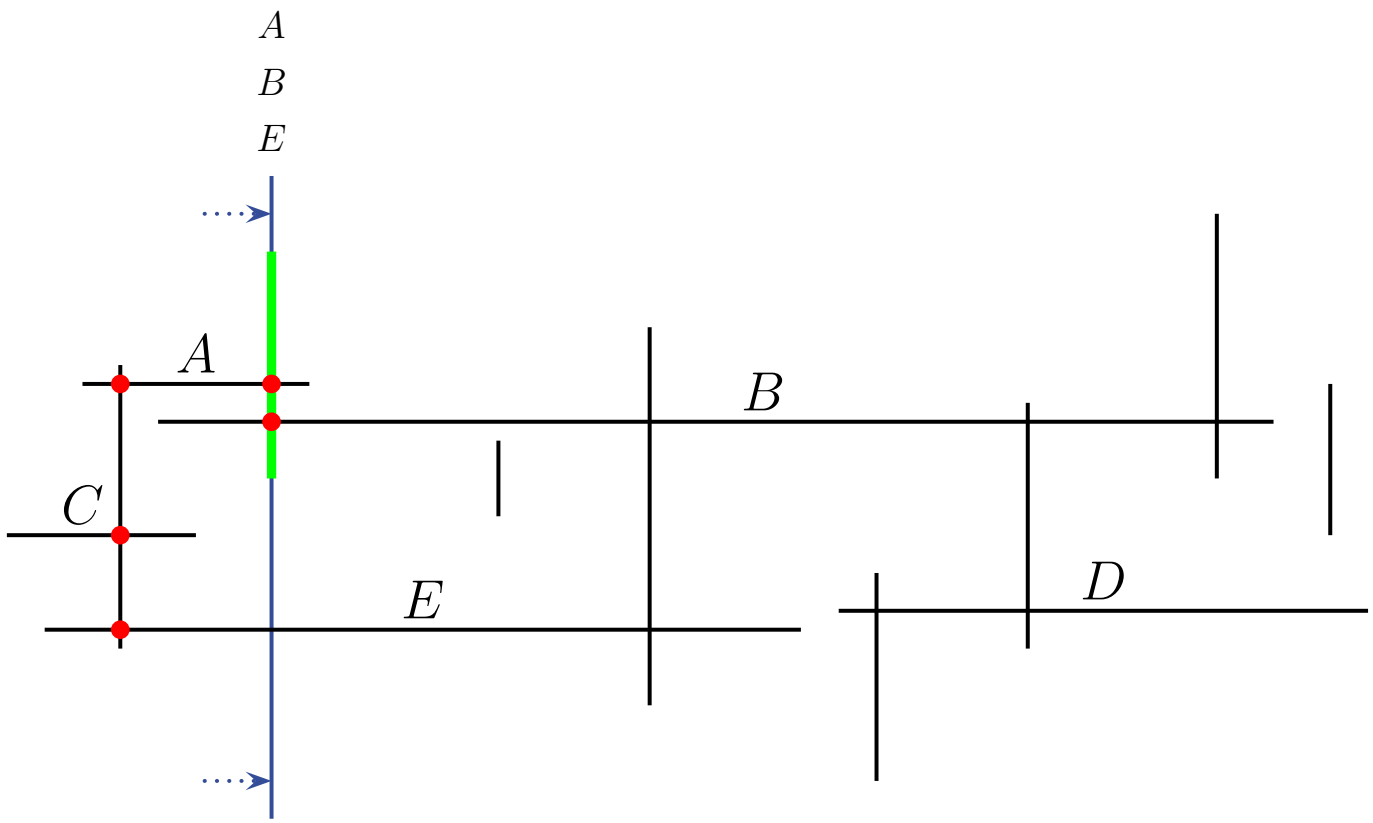

C
E
A
B
C
E
D

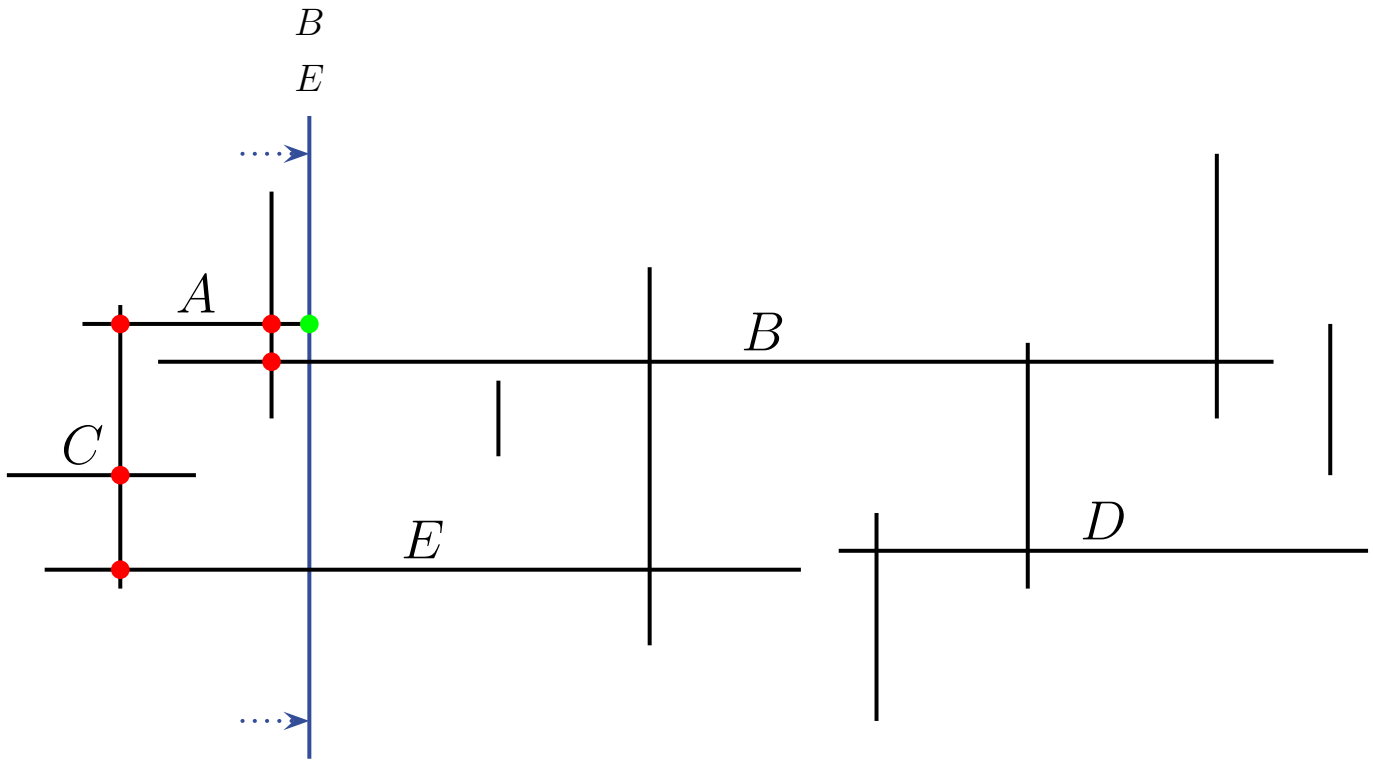

A
C
E
A
B
C
E
D

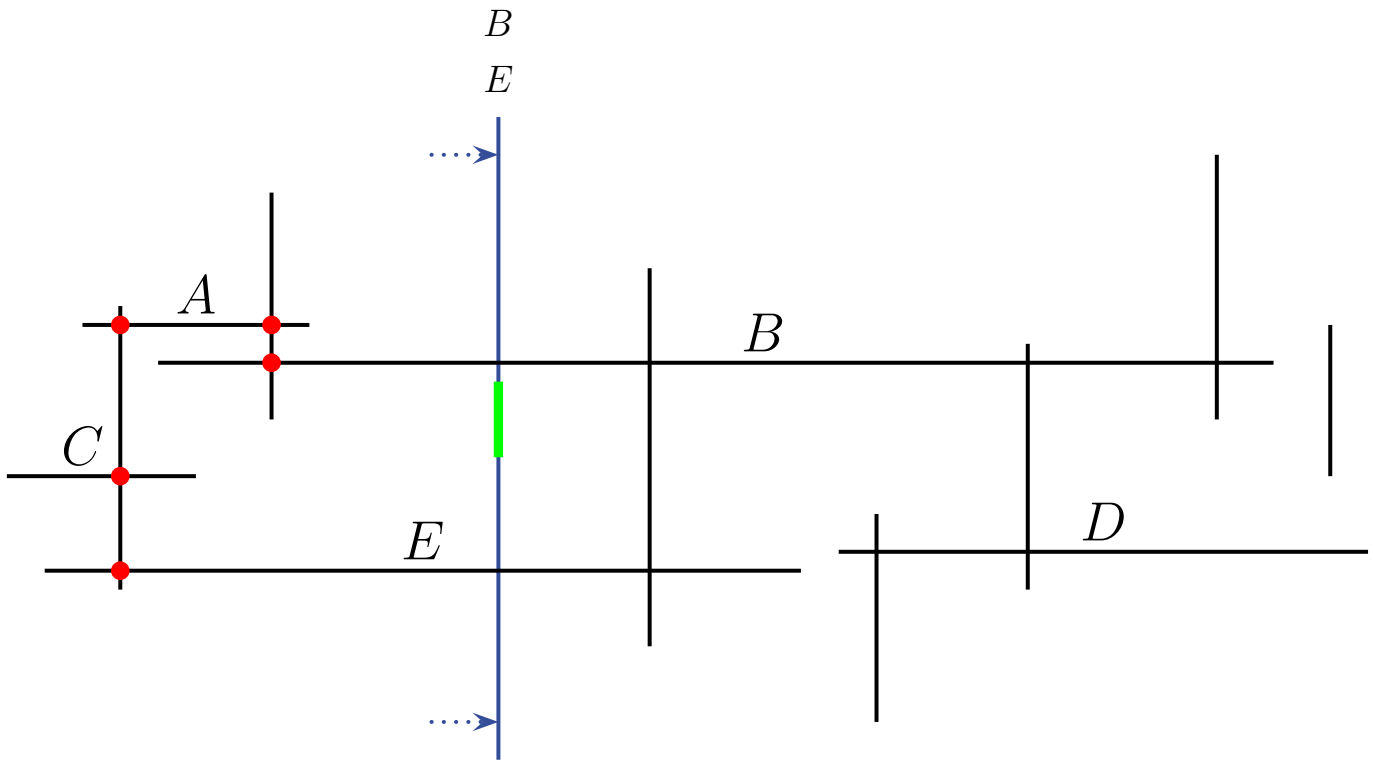

A
C
E
A
B
C
E
D

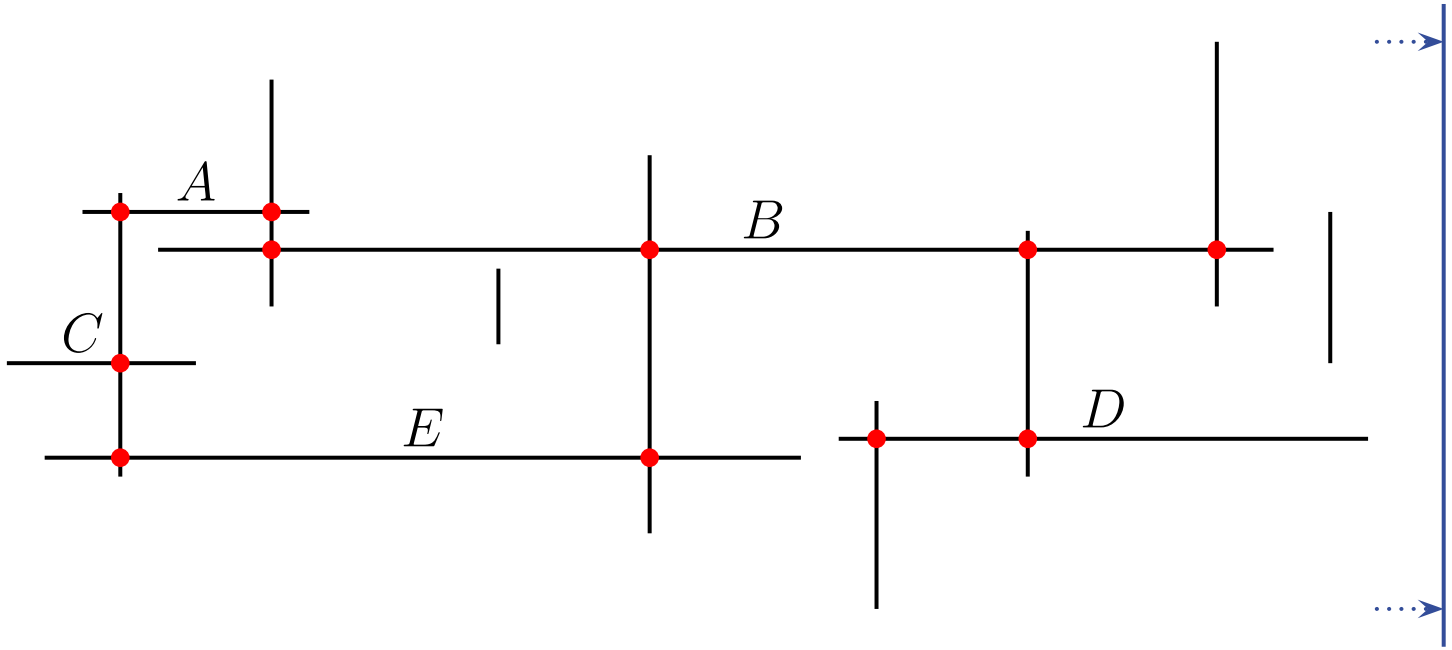

A
B
C
E
A
B
C
E
D


A
B
E
A
B
C
E
D


A
B
E
A
B
C
E
D

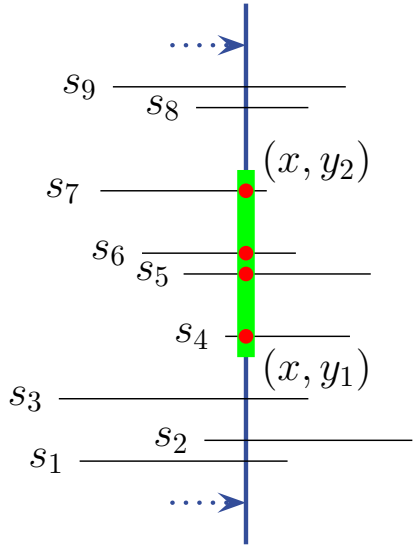

B
E
A
B
C
E
D


B
E
A
B
C
E
D

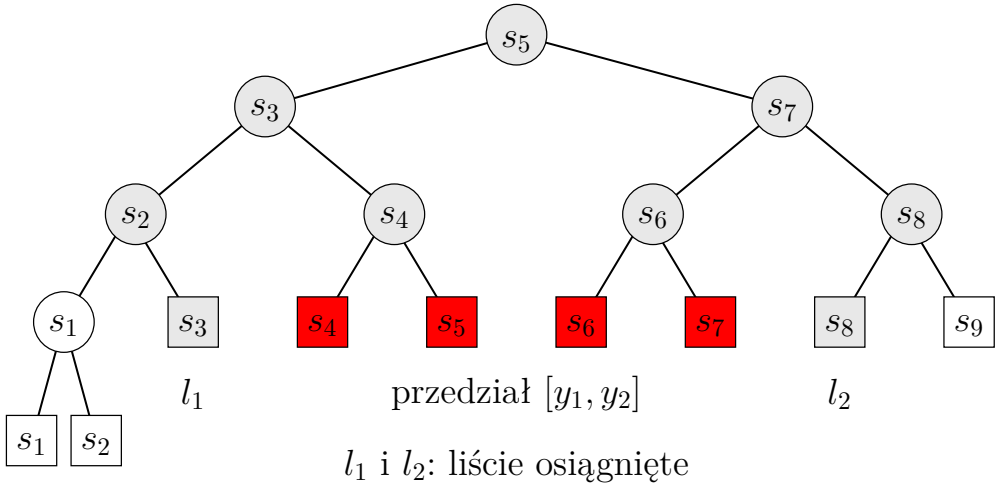

A
B
C
D
E

Algorytm SweepCrossings

0. Posortuj zbiór $S^\bullet := \overline{H} \cup V$ względem odciętej, gdzie $\overline{H}$ jest zbiorem końców poziomych odcinków z H. /Wstępne przetwarzanie./
1. Niech $\mathcal{T}$ będzie pustym zrównoważonym drzewem binarnym.
/Drzewo $\mathcal{T}$ przechowuje status miotły./
2. **while** $S^\bullet \neq \emptyset$ **do**
   2.1 Usuń pierwszy element $v \in S^\bullet$.
   2.2 **if** (v jest lewym końcem odcinka s) **then** dodaj s do $\mathcal{T}$.
   2.3 **else if** (v jest prawym końcem odcinka s) **then** usuń s z $\mathcal{T}$.
   2.4 **else** 1DRangeQuery($\mathcal{T}, [y_1, y_2]$), gdzie $v = [(x, y_1), (x, y_2)]$.



1DRangeQuery($\mathcal{T}, [y_1, y_2]$)



l_1 i l_2: liście osiągnięte
w poszukiwaniu y_1 i odpowiednio y_2

Algorytm SweepCrossings

0. Posortuj zbiór $S^\bullet := \overline{H} \cup V$ względem odciętej, gdzie $\overline{H}$ jest zbiorem końców poziomych odcinków z H. /Wstępne przetwarzanie./
1. Niech $\mathcal{T}$ będzie pustym zrównoważonym drzewem binarnym. /Drzewo $\mathcal{T}$ przechowuje status miotły./
2. **while** $S^\bullet \neq \emptyset$ **do**
   2.1 Usuń pierwszy element $v \in S^\bullet$.
   2.2 **if** (v jest lewym końcem odcinka s) **then** dodaj s do $\mathcal{T}$.
   2.3 **else if** (v jest prawym końcem odcinka s) **then** usuń s z $\mathcal{T}$.
   2.4 **else** 1DRangeQuery($\mathcal{T}, [y_1, y_2]$), gdzie $v = [(x, y_1), (x, y_2)]$.

Analiza poprawności podejścia

- Zachowywanie niezmiennika wynika z konstrukcji algorytmu, tj. wiersza 2.4.

Analiza złożoności obliczeniowej algorytmu

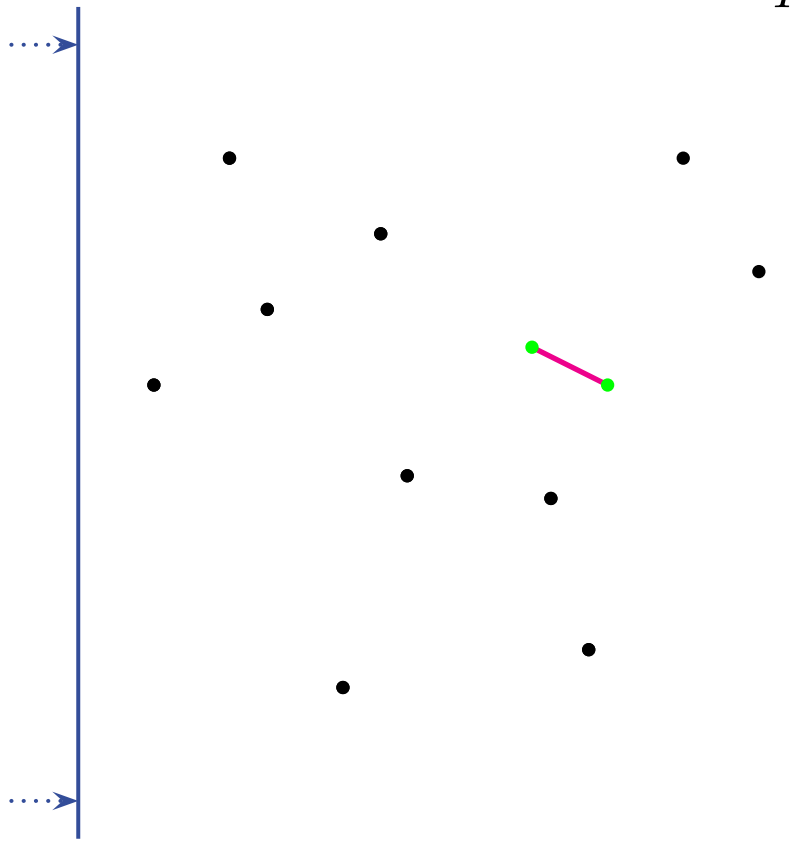
- Przetwarzanie wstępne wymaga czasu rzędu $O(n \log n)$.
- Czas działania algorytmu: $O(n \log n) + A(n)$, gdzie $A(n)$ jest czasem zużytym przez wszystkie wywołania 1DRangeQuery.
- Status miotły (zrównoważone drzewo binarne): pamięć rzędu $O(n)$.

Twierdzenie 2.1. (Smid 2003) Algorytm SweepCrossings wyznacza wszystkie przecięcia w czasie rzędu $O(n \log n + k)$, gdzie n jest liczbą odcinków, a k jest liczbą przecięć; zużycie pamięci jest rzędu $O(n)$.

6.3 Para najbliższych punktów

Dla danego zbioru punktów $S \subset \mathcal{E}^2$ wyznaczyć parę najbliższych punktów.

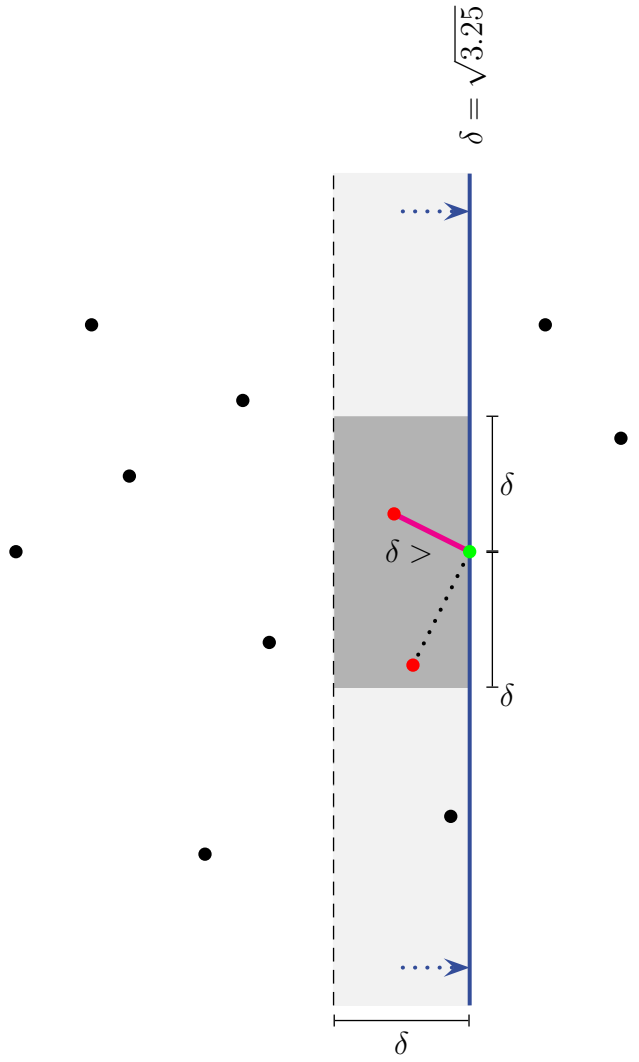
K. Hinrichs, J. Nievergelt, P. Schorn
Plane-sweep solves the closest pair problem elegantly
IPL 26(5), 255-261 (1988)



Założenie. Odcięte punktów wejściowych są różne.

Idea algorytmu

- Miotła M przesuwa się z lewo na prawo pamiętając najmniejszą znalezioną do tej pory odległość δ.
- Status miotły stanowi zbiór punktów na lewo od miotły w odległości co najwyżej δ i zmienia się on przy napotkaniu kolejnych punktów.
- Napotykając nowy punkt $p = (x, y)$, sprawdzana jest odległość p do punktów ze statusu miotły leżących „nie za daleko” od p.



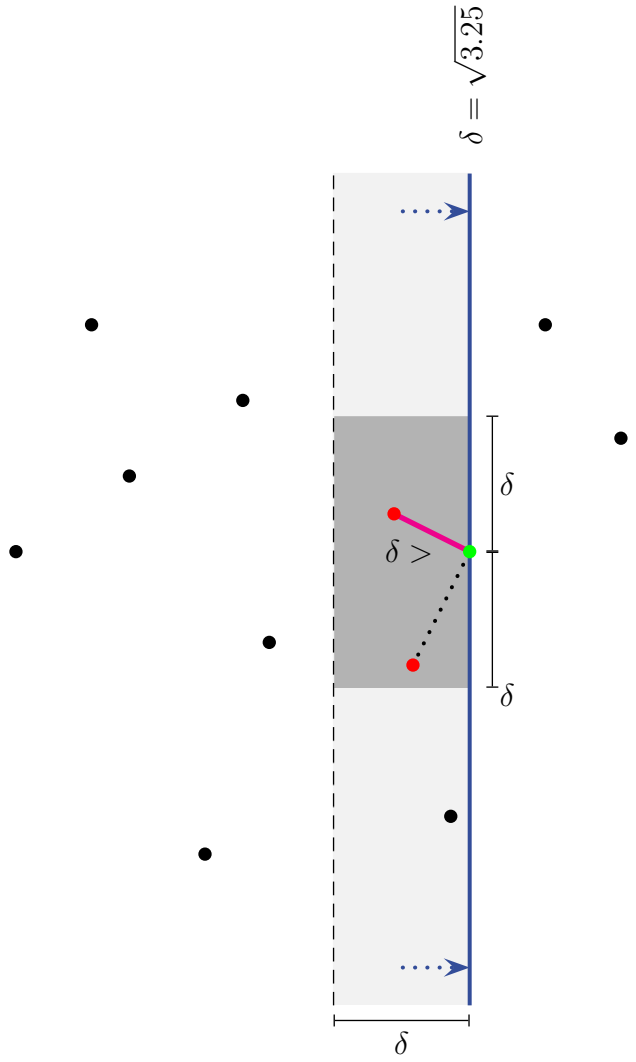
Niezmiennik.
Wartość δ jest najmniejszą odległością spośród punktów na lewo od miotły M.

> Jeśli M przesunie się po całym obszarze, niezmiennik zagwarantuje, że wyznaczona zostanie para najbliższych punktów.

Status miotły.
Punkty na lewo od miotły i w jej δ-otoczeniu.

> Na miotłę można patrzeć jak na pionowy pasek o szerokości δ ograniczony przez dwie pionowe proste. Status miotły przechowywany jest w zrównoważonym drzewie przeszukiwań; punkty posortowane są względem współrzędnej y.

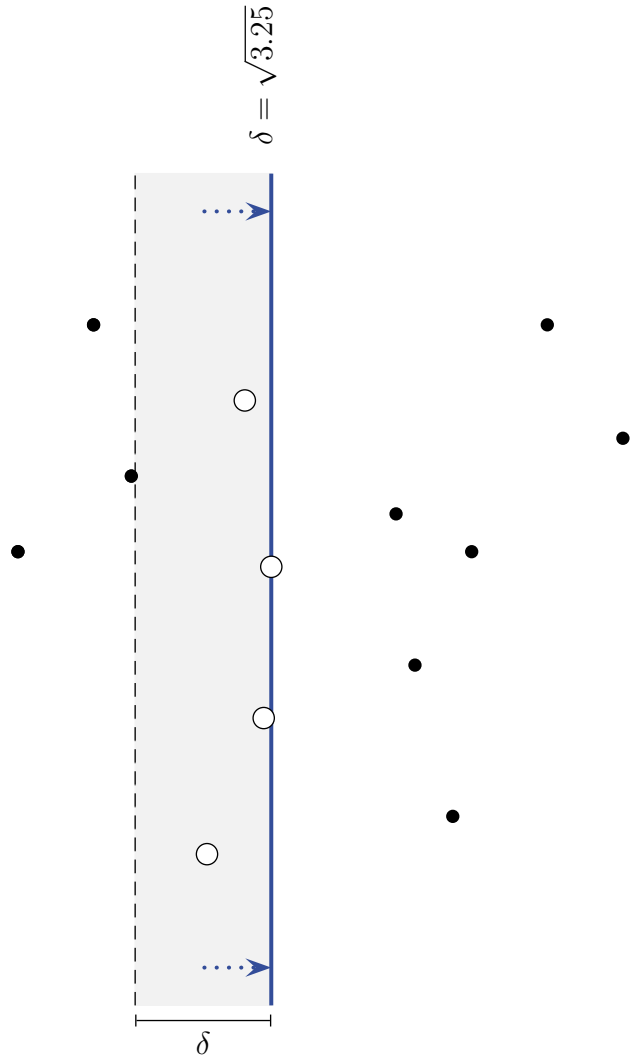
Punkty zdarzeń.
Punkty z S posortowane względem odciętej.



Zmiana statusu.

Miotła napotyka nowy punkt $p = (x, y)$.

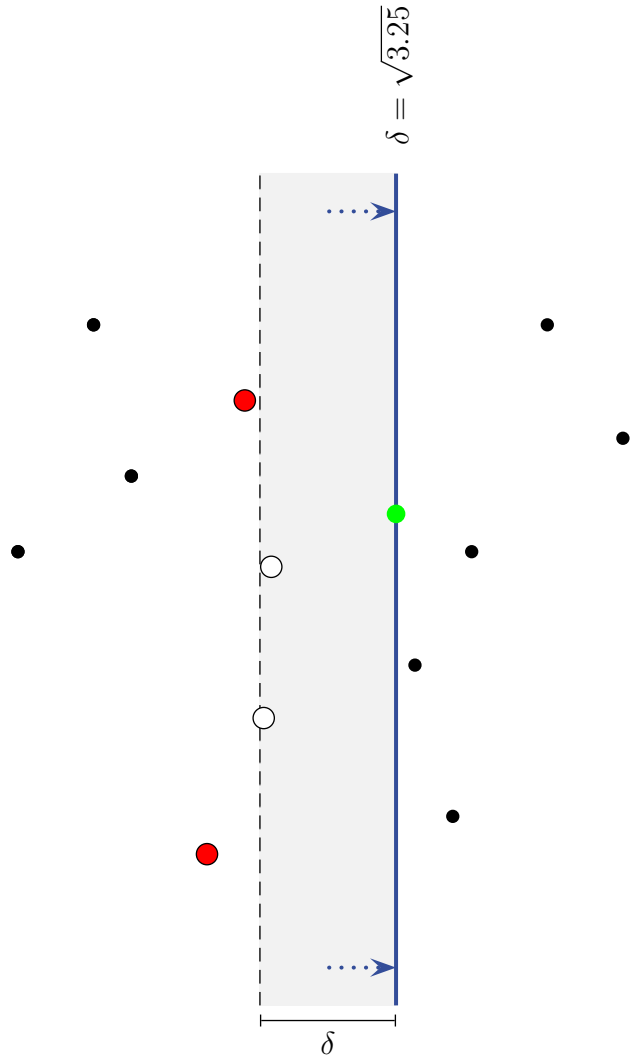
1. Usuwamy ze statusu miotły wszystkie te punkty, które nie leżą w jej δ-otoczeniu.
2. Sprawdzana jest odległość p do punktów ze statusu miotły, których rzędna mieści się w przedziale $[y - \delta, y + \delta]$.
3. Jeśli któraś odległości wyznaczonych w (2) jest mniejsza od dotychczasowej δ, to δ ulega zmianie na najmniejszą z nich. Zapamiętujemy również parę punktów realizującą tę najmniejszą odległość.
4. Wstawiamy p do statusu miotły.



Zmiana statusu.

Miotła napotyka nowy punkt $p = (x, y)$.

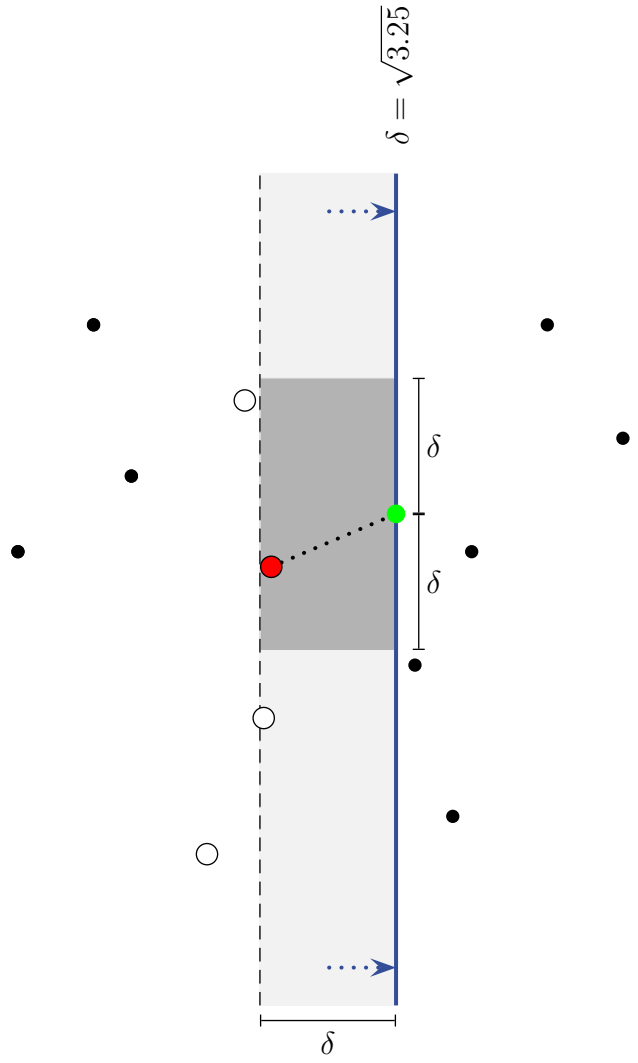
1. Usuwamy ze statusu miotły wszystkie te punkty, które nie leżą w jej δ-otoczeniu.
2. Sprawdzana jest odległość p do punktów ze statusu miotły, których rzędna mieści się w przedziale $[y - \delta, y + \delta]$.
3. Jeśli któraś odległości wyznaczonych w (2) jest mniejsza od dotychczasowej δ, to δ ulega zmianie na najmniejszą z nich. Zapamiętujemy również parę punktów realizującą tę najmniejszą odległość.
4. Wstawiamy p do statusu miotły.

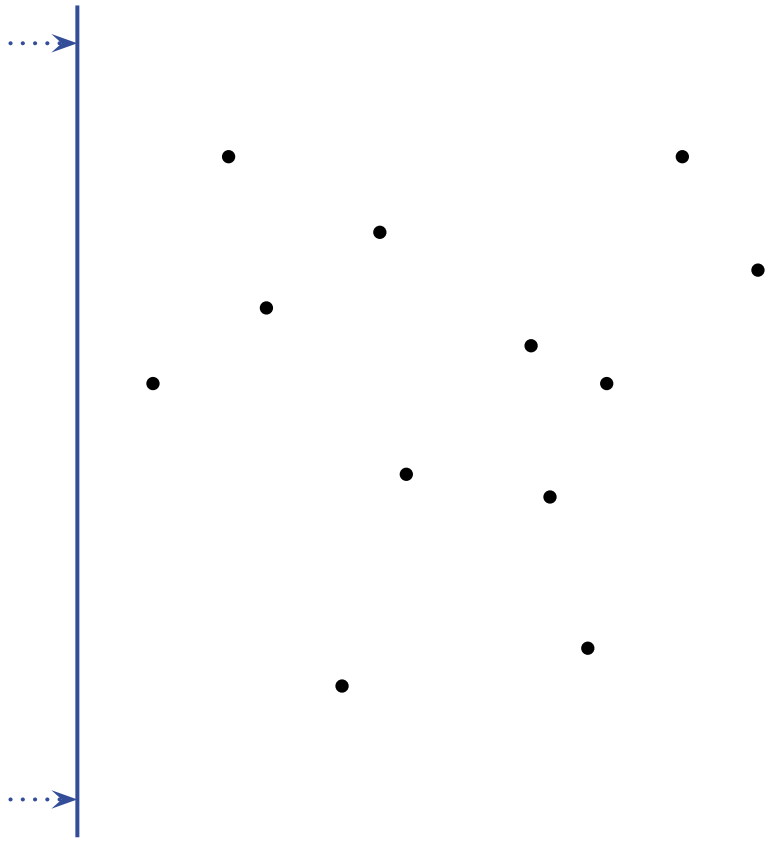


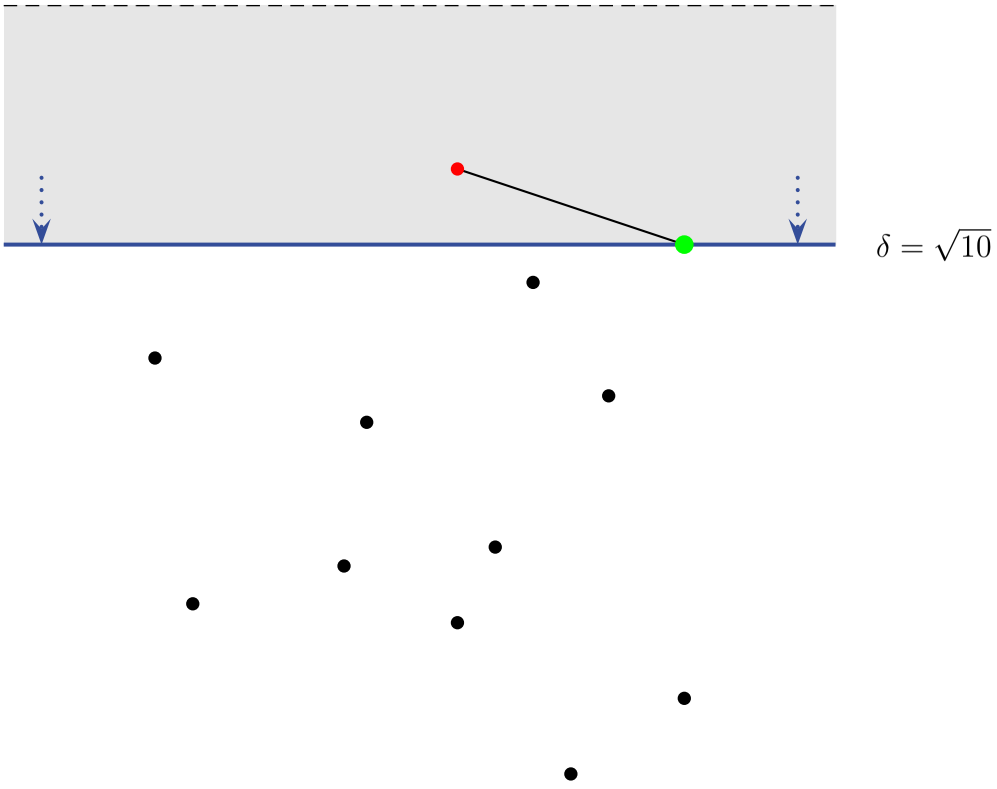
Zmiana statusu.

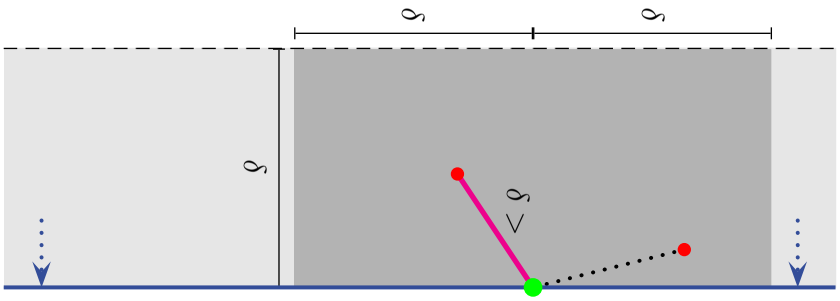
Miotła napotyka nowy punkt $p = (x, y)$.

1. Usuwamy ze statusu miotły wszystkie te punkty, które nie leżą w jej δ-otoczeniu.
2. Sprawdzana jest odległość p do punktów ze statusu miotły, których rzędna mieści się w przedziale $[y - \delta, y + \delta]$.
3. Jeśli któraś odległości wyznaczonych w (2) jest mniejsza od dotychczasowej δ, to δ ulega zmianie na najmniejszą z nich. Zapamiętujemy również parę punktów realizującą tę najmniejszą odległość.
4. Wstawiamy p do statusu miotły.

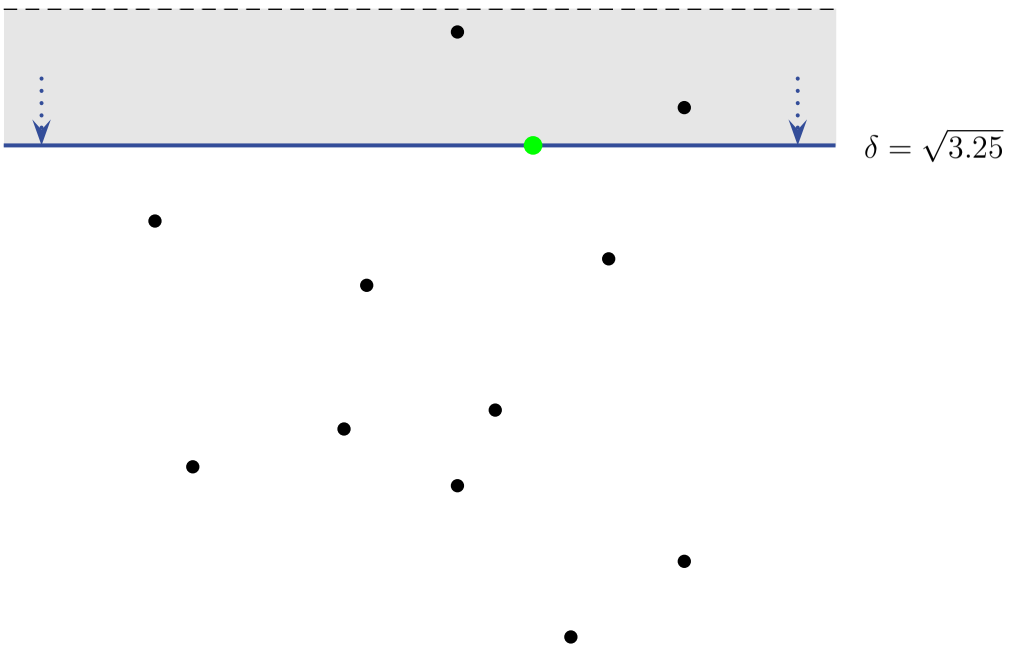


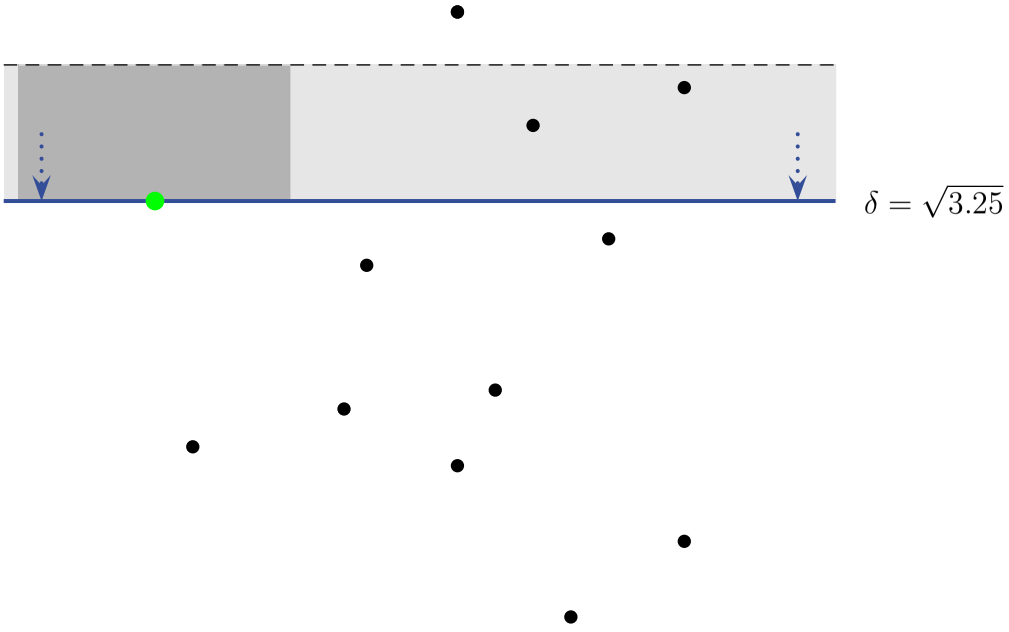


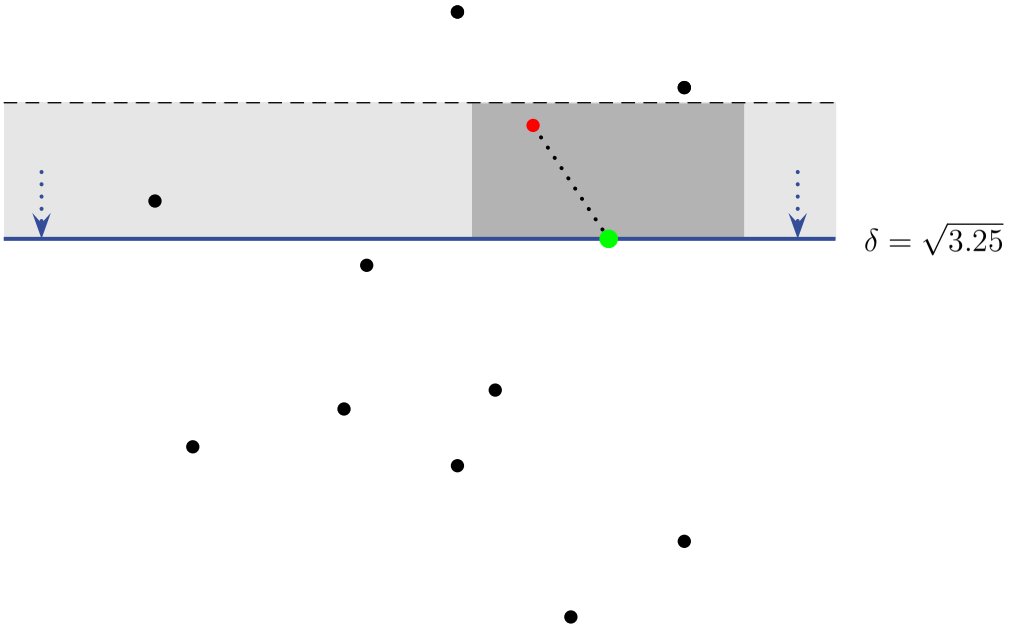

$\delta = \sqrt{10}$

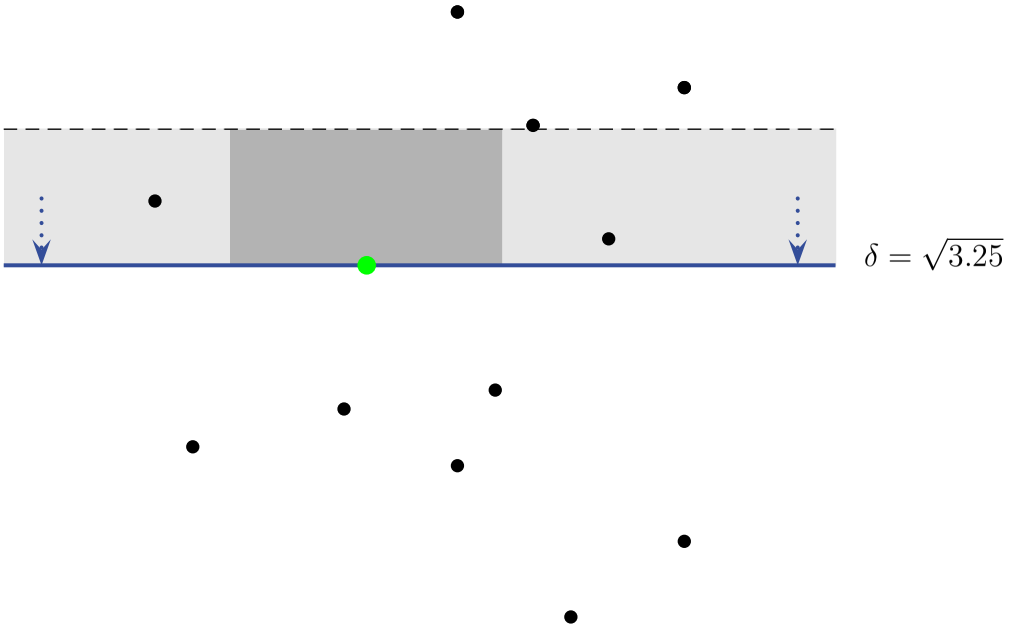


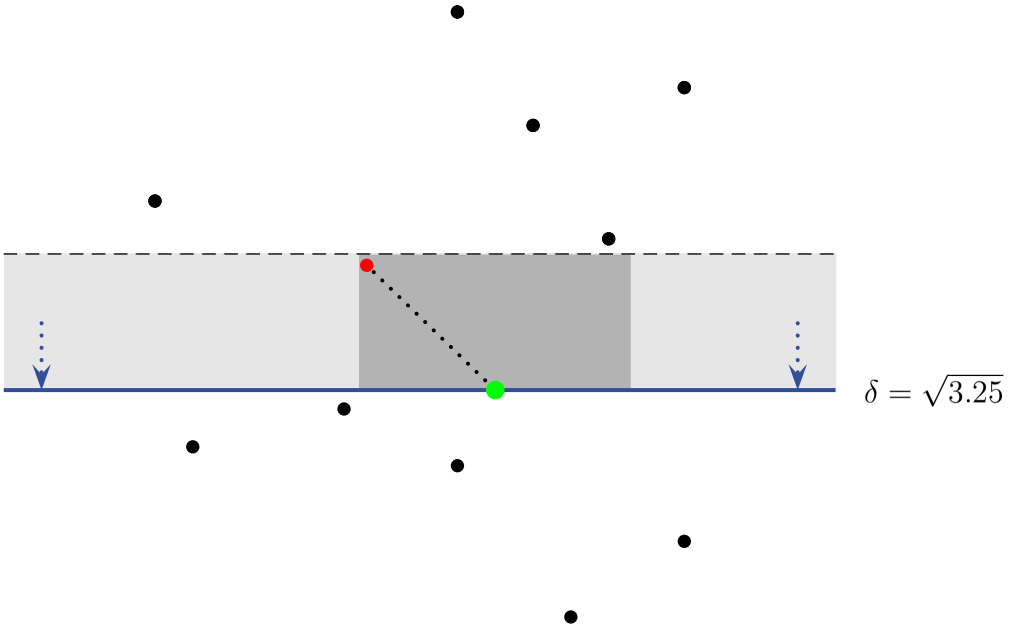
$\delta = \sqrt{10}$

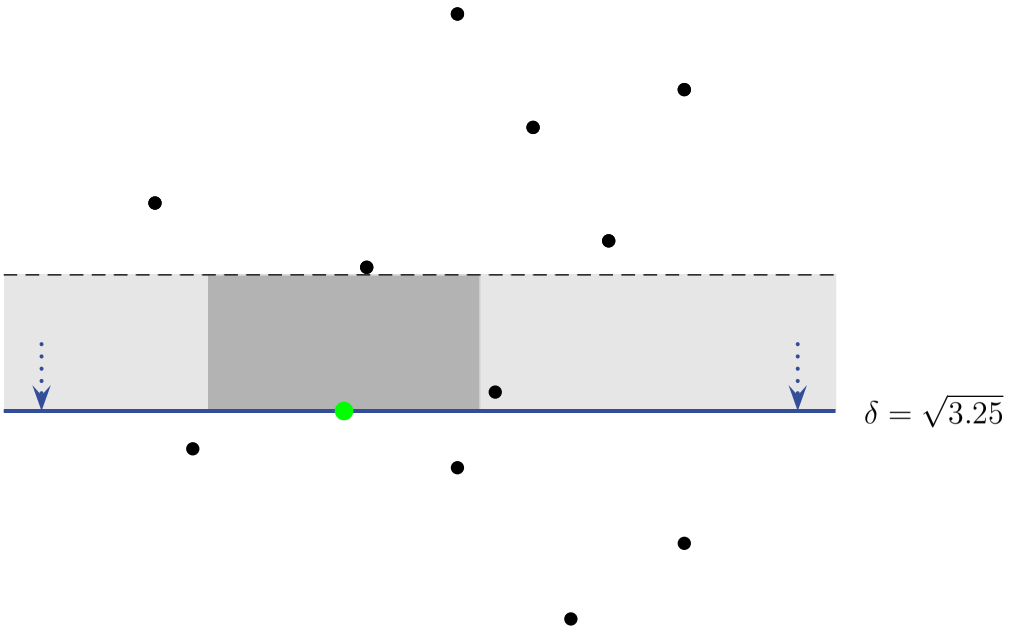

$\delta = \sqrt{3.25}$

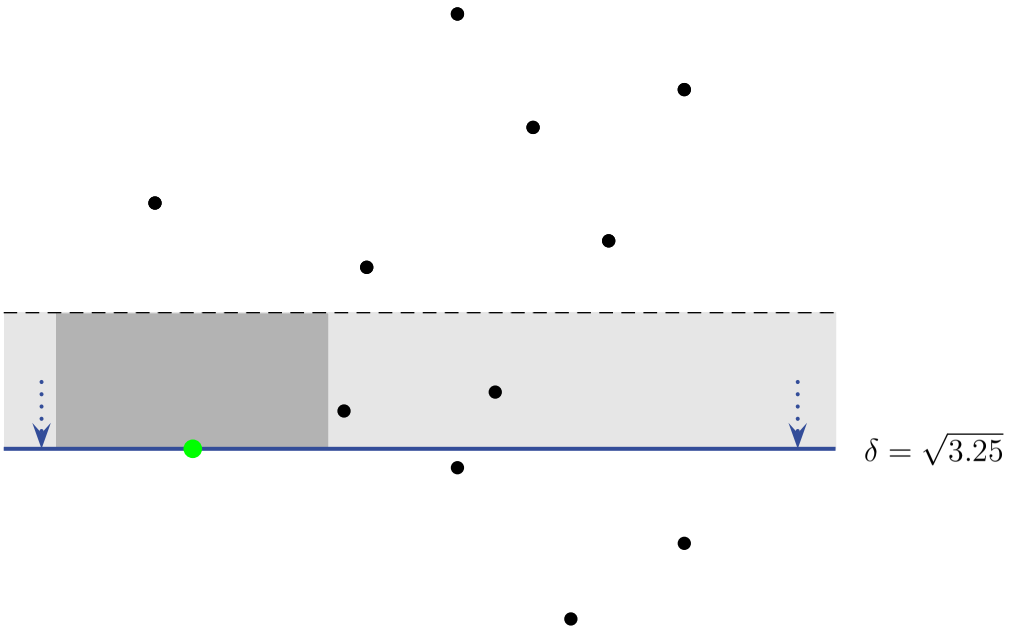

$\delta = \sqrt{3.25}$

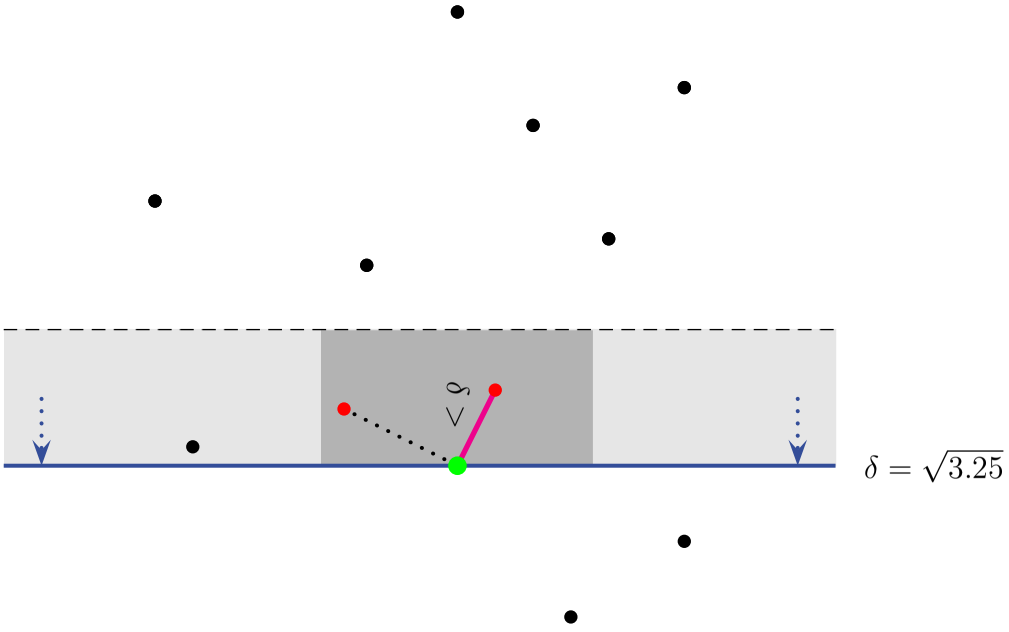

$\delta = \sqrt{3.25}$

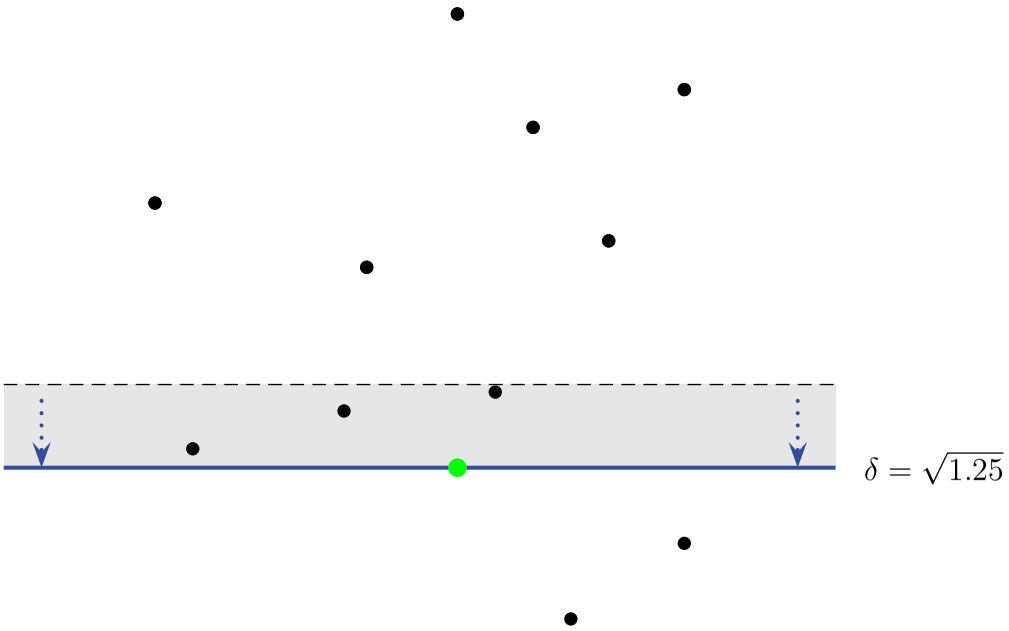

$\delta = \sqrt{3.25}$

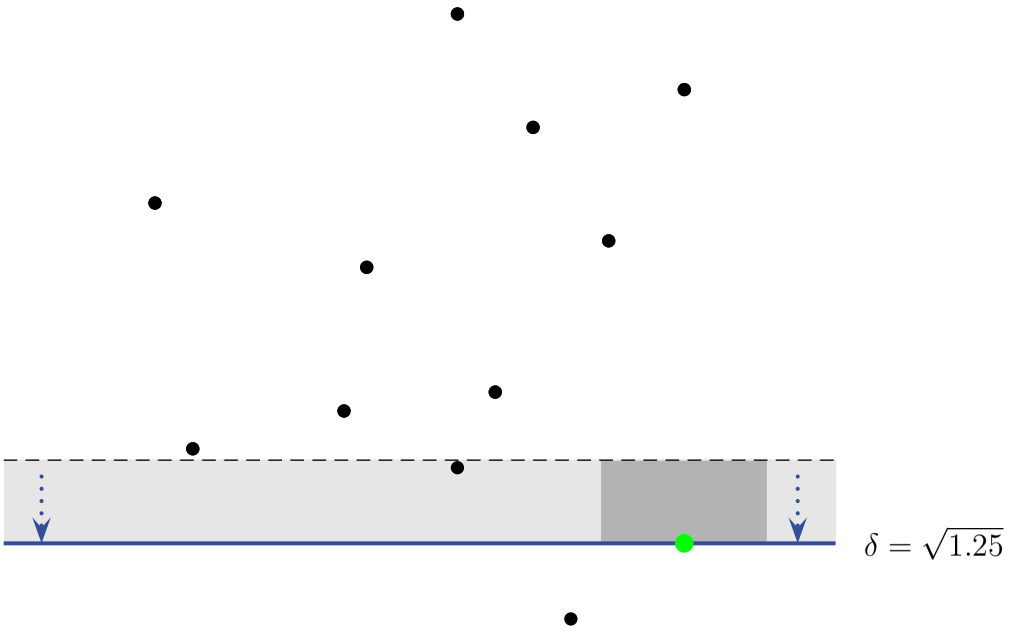

$\delta = \sqrt{3.25}$

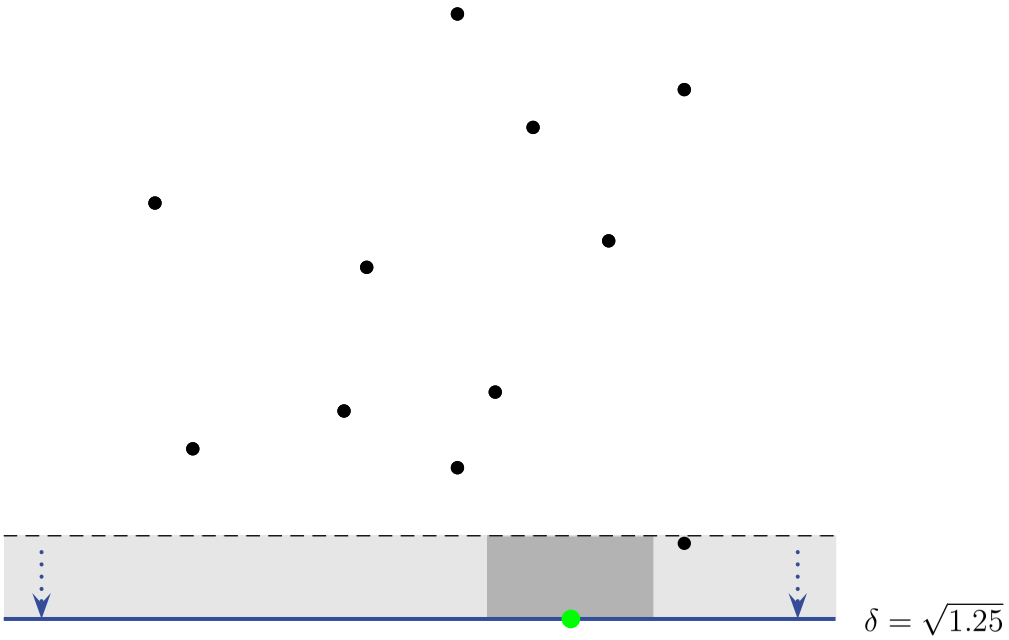

$\delta = \sqrt{3.25}$

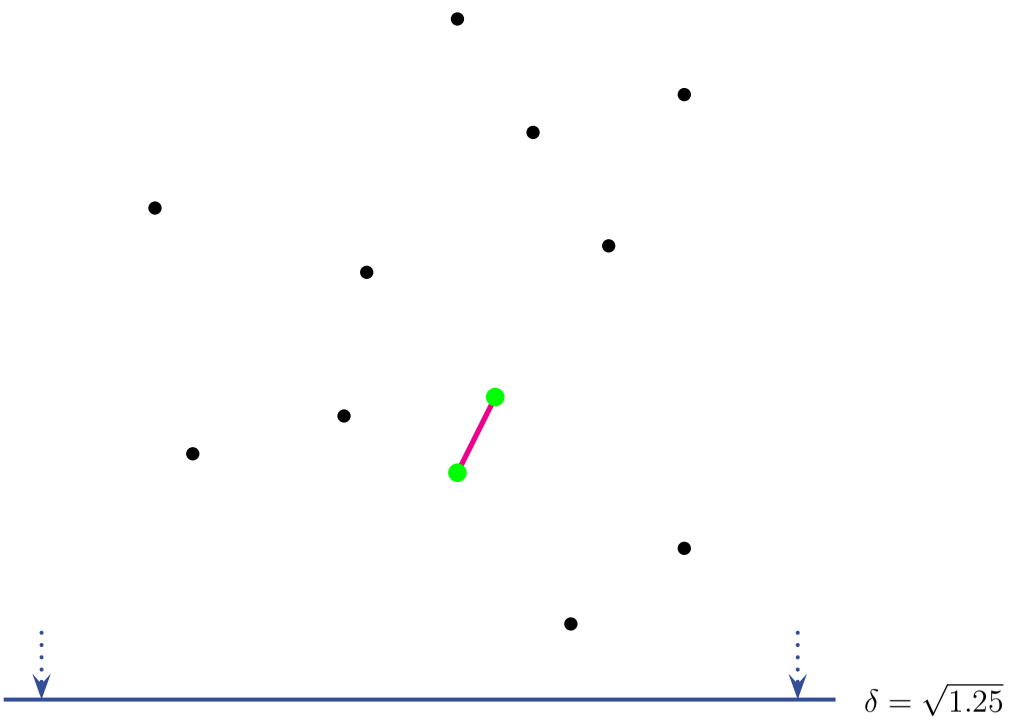

$\delta = \sqrt{3.25}$

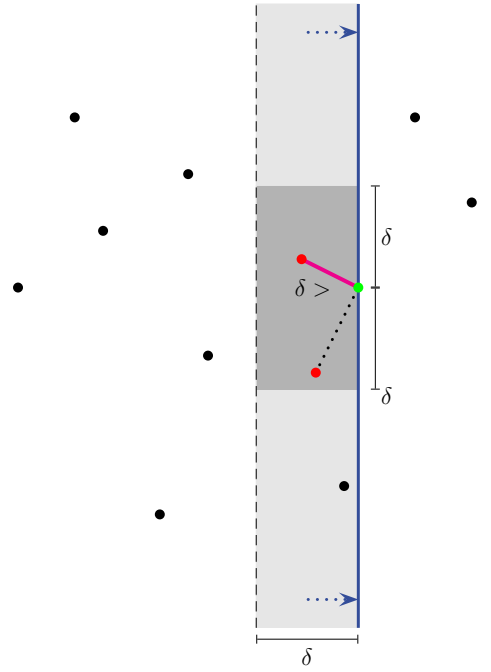

δ
$\delta = \sqrt{3.25}$


$\delta = \sqrt{1.25}$


$\delta = \sqrt{1.25}$


$\delta = \sqrt{1.25}$


$\delta = \sqrt{1.25}$



Analiza poprawności podejścia

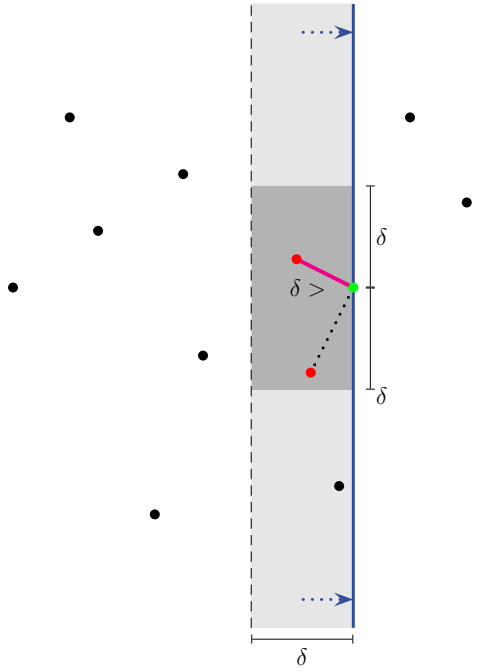
- Prawdziwość niezmiennika w przypadku bazowym – tj. gdy rozpatrujemy dwa pierwsze punkty p_1 p_2 i ustalamy $\delta = d_{\mathcal{E}}(p_1, p_2)$ – wynika z definicji.
- Rozpatrzyliśmy już k punktów, zachowując prawdziwość niezmiennika, tzn. δ przechowuje najmniejszą z odległości dla $\{p_1, p_2, \ldots, p_k\} = S_k$.
- Nowe zdarzenie/punkt $p_{k+1} = (x, y)$.
  - ↳ Minimalna odległość w zbiorze $\{p_1, p_2, \ldots, p_k, p_{k+1}\} = S_{k+1}$:

    $\min(\delta, d_{\mathcal{E}}(p_{k+1}, S_k))$, gdzie $d_{\mathcal{E}}(p_{k+1}, S_k) = \min_{p \in S_k} d_{\mathcal{E}}(p_{k+1}, p)$.

  - ↳ Jeśli $p \in S_k$ oraz $p \notin [x - \delta, x] \times [y - \delta, y + \delta]$, to $d_{\mathcal{E}}(p_{k+1}, p) > \delta$.
  - ↳ A zatem należy sprawdzić jedynie $d_{\mathcal{E}}(p_{k+1}, p)$ dla $p \in [x-\delta, x] \times [y-\delta, y+\delta]$.

Algorytm SweepClosest
Wejście. Zbiór punktów S na płaszczyźnie.
Wyjście. Para najbliższych punktów z S.

0. Posortuj zbiór S względem współrzędnej x, otrzymując zbiór $S = \{p_1, \ldots, p_n\}$.
1. $\delta = d_{\mathcal{E}}(p_1, p_2)$; zapamiętaj parę (p_1, p_2).
2. Wstaw p_1 i p_2 do pustego zrównoważonego drzewa przeszukiwań $\mathcal{T}$ o porządku względem współrzędnej y.
3. lewy:=1; prawy:=3.
4. **while** (prawy $\leq n$) **do**
   4.1 **while** $\big(\mathrm{x}(p_{\text{lewy}}) < \mathrm{x}(p_{\text{prawy}}) - \delta\big)$ **do**
   usuń p_{lewy} z drzewa $\mathcal{T}$;
   lewy:=lewy+1;
   4.2 Niech S_δ :=1DRangeQuery$\big(\mathcal{T}, [\mathrm{y}(p_{\text{prawy}}) - \delta, \mathrm{y}(p_{\text{prawy}}) + \delta]\big)$.
   4.3 $\delta := \min(\delta, d_{\mathcal{E}}(p_{\text{prawy}}, S_\delta))$; zapamiętaj parę realizującą minimum.
   4.4 Wstaw p_{prawy} do drzewa $\mathcal{T}$; prawy:=prawy+1.
5. Zwróć parę realizującą δ.

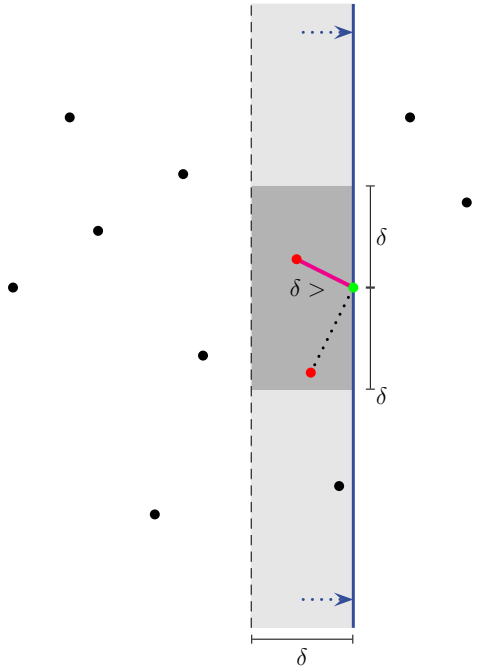


Analiza złożoności obliczeniowej

- Złożoność czasowa jest rzędu $O(n \log n)$.
  ↳ Czas 1DRangeQuery jest rzędu $O(\log n) + |S_\delta| = O(\log n)$, bo $|S_\delta| \leq 6$.
- Złożoność pamięciowa jest rzędu $O(n)$.
  ↳ Status prostej (zrównoważone drzewo binarne): pamięć rzędu $O(n)$.

Algorytm SweepClosest
Wejście. Zbiór punktów S na płaszczyźnie.
Wyjście. Para najbliższych punktów z S.

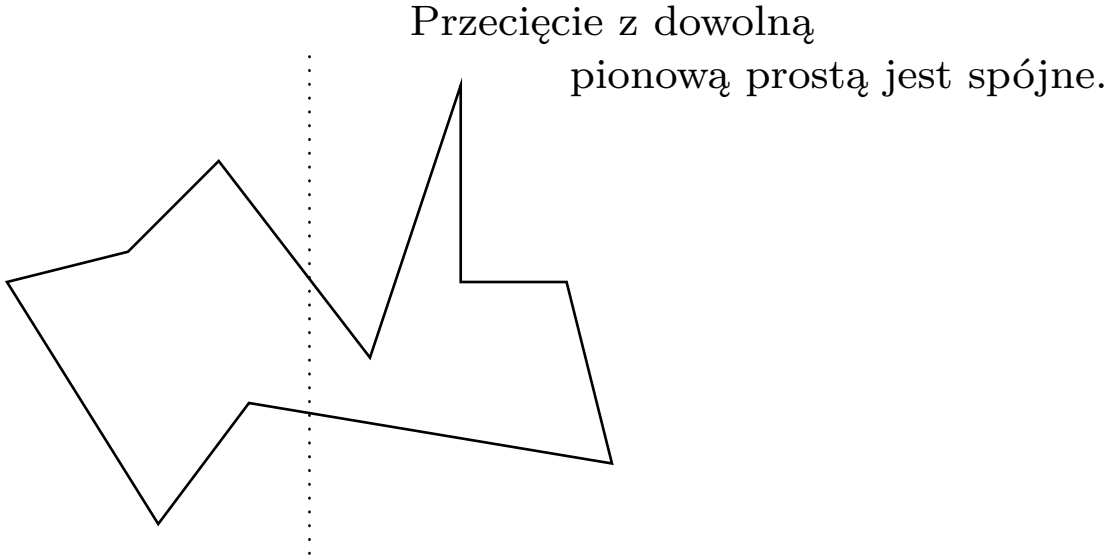
0. Posortuj zbiór S względem współrzędnej x,
   otrzymując zbiór $S = \{p_1, \ldots, p_n\}$.
1. $\delta = d_{\mathcal{E}}(p_1, p_2)$; zapamiętaj parę (p_1, p_2).
2. Wstaw p_1 i p_2 do pustego zrównoważonego drzewa przeszukiwań $\mathcal{T}$
   o porządku względem współrzędnej y.
3. lewy:=1; prawy:=3.
4. **while** (prawy $\leq n$) **do**
   4.1 **while** $\big(\mathrm{x}(p_{\text{lewy}}) < \mathrm{x}(p_{\text{prawy}}) - \delta\big)$ **do**
   usuń p_{lewy} z drzewa $\mathcal{T}$;
   lewy:=lewy+1;
   4.2 Niech S_δ :=1DRangeQuery$\big(\mathcal{T}, [\mathrm{y}(p_{\text{prawy}}) - \delta, \mathrm{y}(p_{\text{prawy}}) + \delta]\big)$.
   4.3 $\delta := \min(\delta, d_{\mathcal{E}}(p_{\text{prawy}}, S_\delta))$; zapamiętaj parę realizującą minimum.
   4.4 Wstaw p_{prawy} do drzewa $\mathcal{T}$; prawy:=prawy+1.
5. Zwróć parę realizującą δ.



Twierdzenie 2.2. (Hinrichs, Nievergelt, Schorn 1988)
Algorytm SweepClosest wyznacza najbliższą parę w zbiorze n punktów na płaszczyźnie w czasie $O(n \log n)$, używając pamięci rzędu $O(n)$.

2.4* TRIANGULACJA WIELOKĄTA MONOTONICZNEGO

Wielokąt prosty nazywamy monotonicznym względem prostej l, jeśli dla dowolnej prostej l' prostopadłej do l przecięcie wielokąta z l' jest spójne (jest odcinkiem, punktem, lub zbiorem pustym). Wielokąt, który jest monotoniczny względem osi x, nazywamy x-monotonicznym.

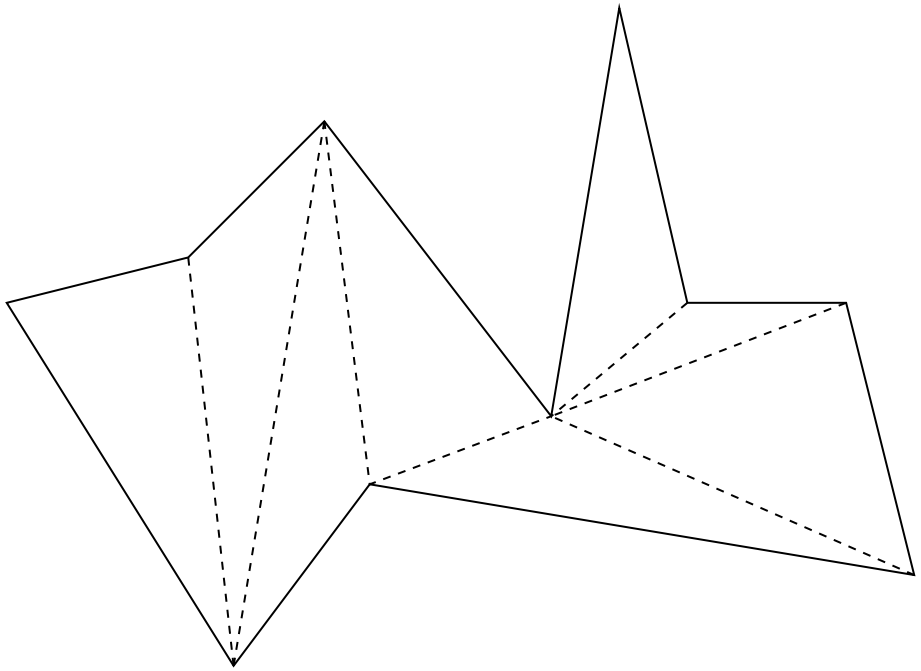


W wielokącie x-monotonicznym, jeśli idziemy wzdłuż łańcucha wierzchołków — dolnego lub górnego — ze skrajnego lewego wierzchołka do skrajnego prawego wierzchołka, to zawsze poruszamy się w prawo lub pionowo, nigdy w lewo.

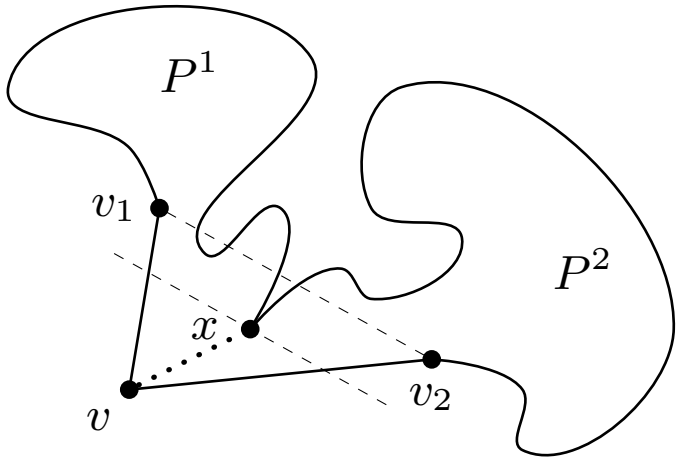
Założenie. Wielokąty *ściśle* monotoniczne: odcięte wierzchołków są różne.

Problem. Podzielić ściśle x-monotoniczny wielokąt na trójkąty przez dodanie wewnętrznych nieprzecinających się przekątnych łączących wierzchołki wielokąta.

M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf
Geometria obliczeniowa, rozdział 3, WNT (2007)

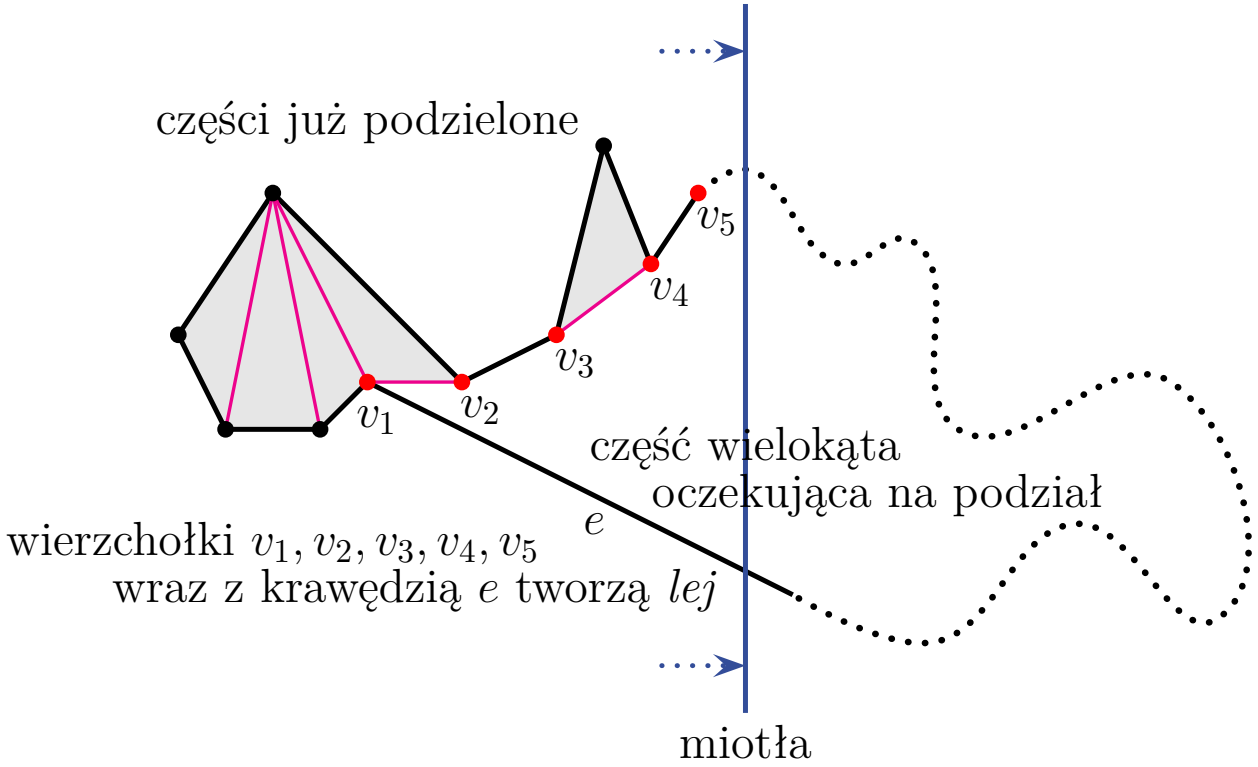


Problem. Podzielić ściśle x-monotoniczny wielokąt na trójkąty przez dodanie wewnętrznych nieprzecinających się przekątnych łączących wierzchołki wielokąta.



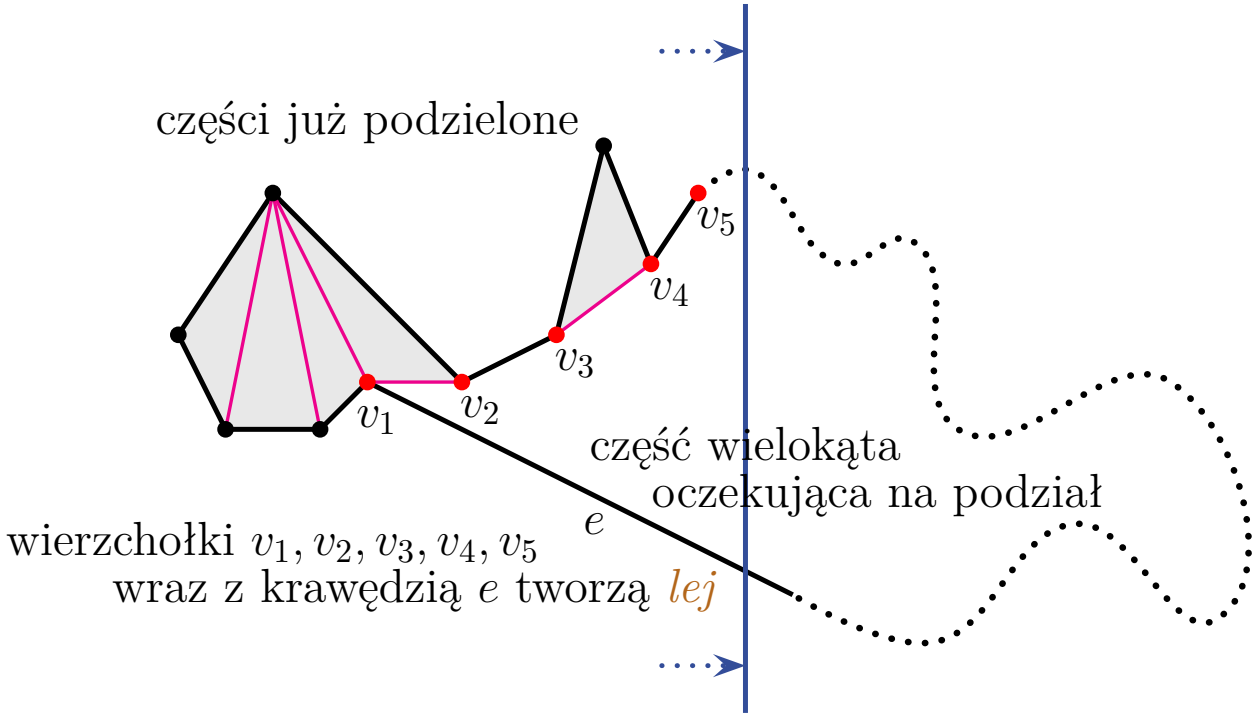
Twierdzenie 2.4. (m.in. Meisters 1975) Dowolny wielokąt prosty o n wierzchołkach można podzielić na $n-2$ trójkąty przez dodanie $n-3$ wewnętrznych nieprzecinających się przekątnych o wierzchołkach będących wierzchołkami wielokąta.

↳ Dowolny wielokąt prosty ($n \geq 4$) ma przynajmniej dwa tzw. *ucha*.
Dowód indukcyjny po liczbie wierzchołków (patrz ilustracja).



Idea algorytmu zamiatania

- Miotła przesuwa się z lewo na prawo.
- Status miotły stanowi zbiór wierzchołków na stosie tworzących tzw. *lej*.
- Napotykając nowy wierzchołek v, w zależności od położenia v względem leja, wierzchołek ten jest albo odkładany na stos, powiększając lej, albo cześć wierzchołków jest zdejmowana ze stosu i zgłaszane są odpowiednie przekątne.

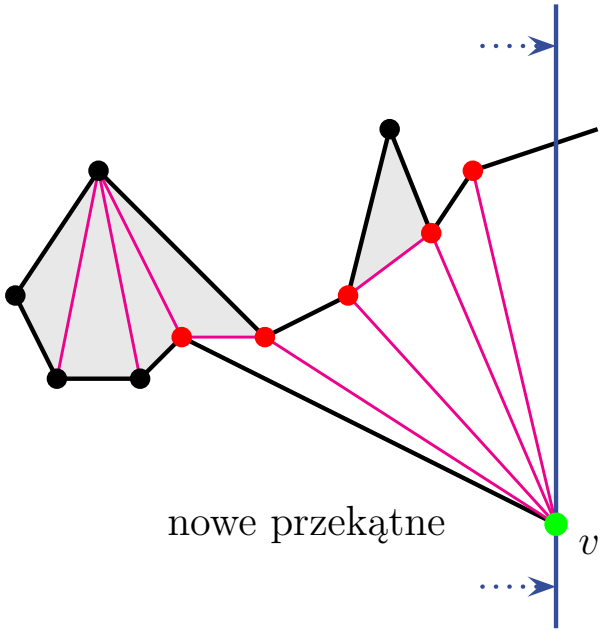


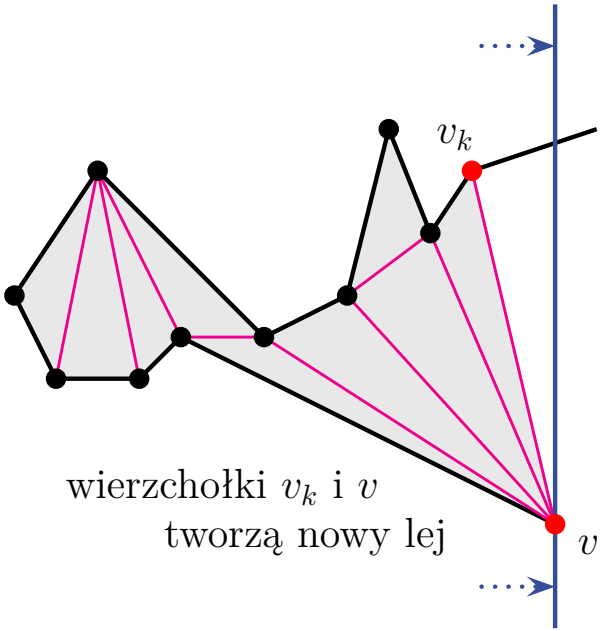
Status miotły. Wierzchołki na stosie.

Niezmiennik. Wierzchołki $v_1, \ldots, v_k$ na stosie tworzą tzw. lej, a wielokąt na lewo od łańcucha $v_1, \ldots, v_k$ jest już podzielony na trójkąty..

- Wierzchołek v_1, który jest na dnie stosu tworzy kąt wypukły z incydentną krawędzią i kolejnym wierzchołkiem v_2 ze stosu.
- Pozostałe wierzchołki $v_2, \ldots, v_k$ ze stosu należą do tego samego łańcucha (dolnego lub górnego) i są wklęsłe.

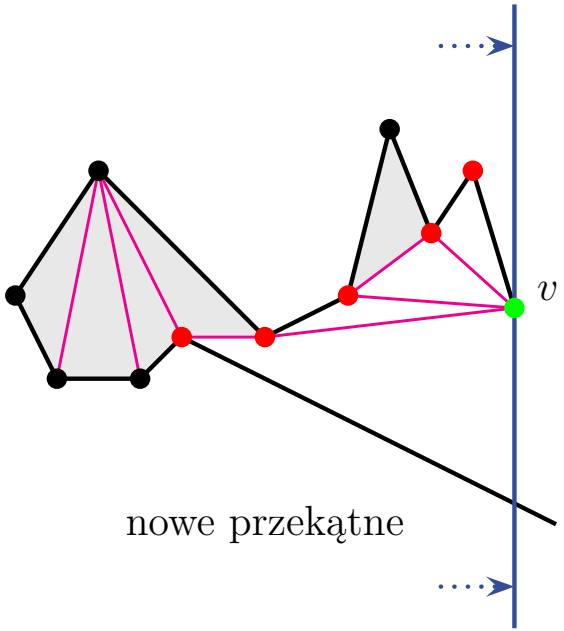
Punkty zdarzeń. Wierzchołki P posortowane względem współrzędnej x.

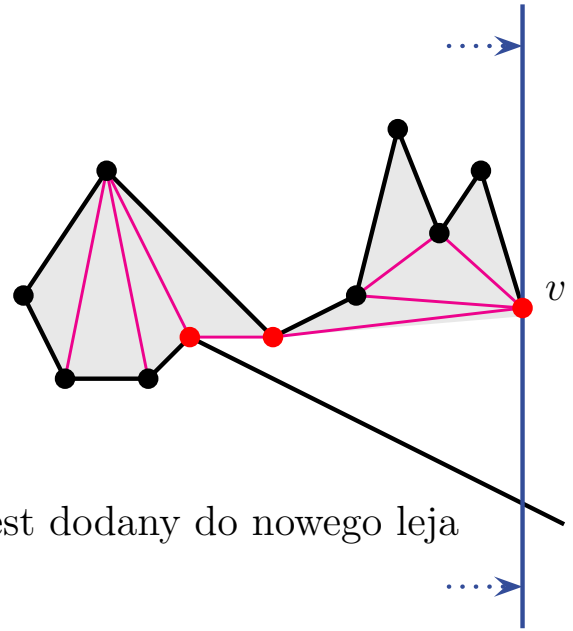




Zmiana statusu. Miotła napotyka nowy wierzchołek v.

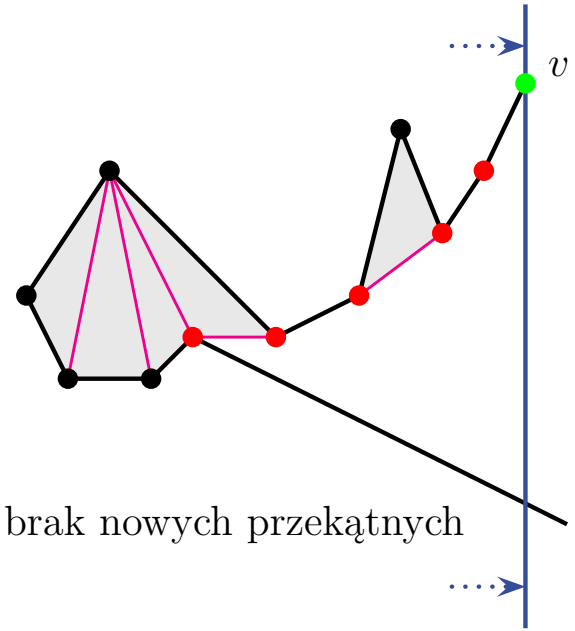
1. Jeśli wierzchołek v leży na przeciwległym łańcuchu do wierzchołków $v_2, \ldots, v_k$ ze stosu, tzn. jest prawym końcem krawędzi ograniczającej lej, wówczas możemy dodać przekątne z v do wszystkich wierzchołków znajdujących się na stosie, za wyjątkiem tego z dna stosu (czyli dodajemy przekątne postaci $\{v, v_i\}$, $i = 2, \ldots, k$). Wierzchołki $v_1, \ldots, v_k$ zostają zdjęte ze stosu i włożone są jedynie v_k i v. Niezmiennik – mając na uwadze x-monotoniczność wielokąta oraz położenie v względem $v_2, \ldots, v_k$ – zostaje zachowany.



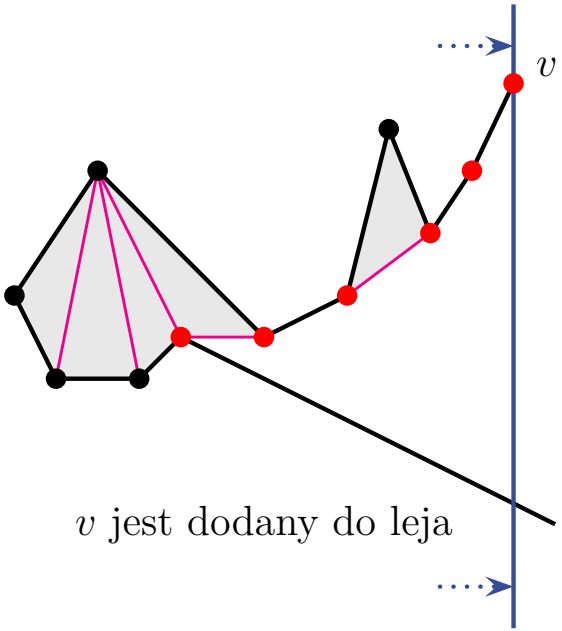


Zmiana statusu. Miotła napotyka nowy wierzchołek v.

2. Jeśli wierzchołek v leży na tym samym łańcuchu, co wierzchołki $v_2, \ldots, v_k$, wówczas możemy nie być w stanie poprowadzić przekątnych z v do wszystkich wierzchołków ze stosu. Ale te wierzchołki, z którymi możemy połączyć v, są kolejne i znajdują się na szczycie stosu. Zatem zdejmujemy je i dodajemy przekątne, aż nie będzie to możliwe. Wierzchołek v wkładany jest na stos. Zachowanie niezmiennika wynika z niemożliwości dodania dalszych przekątnych.



brak nowych przekątnych



v jest dodany do leja

Zmiana statusu. Miotła napotyka nowy wierzchołek v.

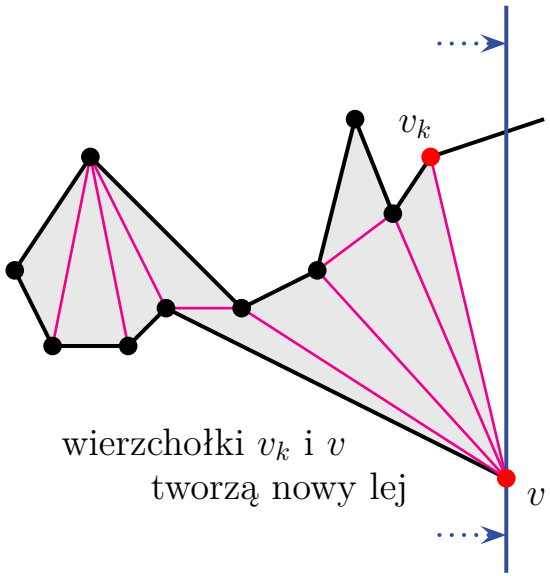
2. Jeśli wierzchołek v leży na tym samym łańcuchu, co wierzchołki $v_2, \ldots, v_k$, wówczas możemy nie być w stanie poprowadzić przekątnych z v do wszystkich wierzchołków ze stosu. Ale te wierzchołki, z którymi możemy połączyć v, są kolejne i znajdują się na szczycie stosu. Zatem zdejmujemy je i dodajemy przekątne, aż nie będzie to możliwe. Wierzchołek v wkładany jest na stos. Zachowanie niezmiennika wynika z niemożliwości dodania dalszych przekątnych.

Algorytm SweepTriangulation

Wejście. Zbiór wierzchołków wielokąta $P = \{v_1, \ldots, v_n\}$ na płaszczyźnie, $n \geq 3$. P jest posortowany względem odciętej.

Wyjście. Podział P na trójkąty.

1. Inicjuj pusty stos S i włóż na niego v_1 i v_2.
2. **for** $i := 3$ **to** $n - 1$ **do**
3. **if** (v_i oraz wierzchołek na szczycie stosu są na różnych łańcuchach)
4. **then** Zdejmij wszystkie wierzchołki ze stosu $S = (v'_1, \ldots, v'_k)$.
   Zgłoś wszystkie przekątne $\{v_i, v'_j\}$, $j = 2, \ldots, k$.
   Włóż v'_k i v_i na stos.
5. **else** Zdejmij wierzchołek v'_k ze stosu $S = (v'_1, \ldots, v'_k)$.
   Zdejmuj pozostałe wierzchołki tak długo, aż odpowiednie przekątne $\{v_i, v'_j\}$ są wewnątrz wielokąta; zgłaszaj te przekątne.
   Włóż ostatnio zdjęty wierzchołek na stos.
   Włóż v_i na stos.
6. Dodaj przekątne z v_n do wszystkich wierzchołków na stosie, oprócz pierwszego i ostatniego.

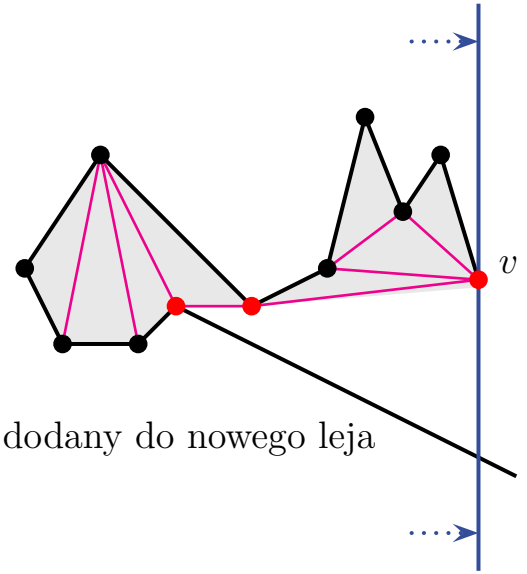


Algorytm SweepTriangulation

Wejście. Zbiór wierzchołków wielokąta $P = \{v_1, \ldots, v_n\}$ na płaszczyźnie, $n \geq 3$. P jest posortowany względem odciętej.

Wyjście. Podział P na trójkąty.

1. Inicjuj pusty stos S i włóż na niego v_1 i v_2.
2. **for** $i := 3$ **to** $n - 1$ **do**
3. **if** (v_i oraz wierzchołek na szczycie stosu są na różnych łańcuchach)
4. **then** Zdejmij wszystkie wierzchołki ze stosu $S = (v'_1, \ldots, v'_k)$.
   Zgłoś wszystkie przekątne $\{v_i, v'_j\}$, $j = 2, \ldots, k$.
   Włóż v'_k i v_i na stos.
5. **else** Zdejmij wierzchołek v'_k ze stosu $S = (v'_1, \ldots, v'_k)$.
   Zdejmuj pozostałe wierzchołki tak długo, aż odpowiednie przekątne $\{v_i, v'_j\}$ są wewnątrz wielokąta; zgłaszaj te przekątne.
   Włóż ostatnio zdjęty wierzchołek na stos.
   Włóż v_i na stos.
6. Dodaj przekątne z v_n do wszystkich wierzchołków na stosie, oprócz pierwszego i ostatniego.



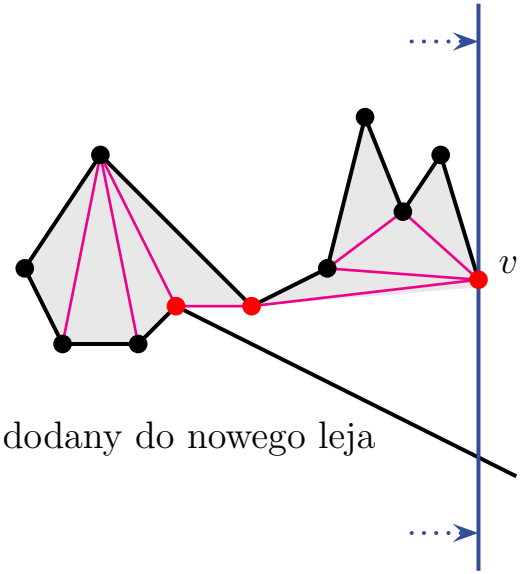
v jest dodany do nowego leja

Algorytm SweepTriangulation

Wejście. Zbiór wierzchołków wielokąta $P = \{v_1, \ldots, v_n\}$ na płaszczyźnie, $n \geq 3$. P jest posortowany względem odciętej.

Wyjście. Podział P na trójkąty.

1. Inicjuj pusty stos S i włóż na niego v_1 i v_2.
2. **for** $i := 3$ **to** $n - 1$ **do**
3. **if** (v_i oraz wierzchołek na szczycie stosu są na różnych łańcuchach)
4. **then** Zdejmij wszystkie wierzchołki ze stosu $S = (v'_1, \ldots, v'_k)$.
   Zgłoś wszystkie przekątne $\{v_i, v'_j\}$, $j = 2, \ldots, k$.
   Włóż v'_k i v_i na stos.
5. **else** Zdejmij wierzchołek v'_k ze stosu $S = (v'_1, \ldots, v'_k)$.
   Zdejmuj pozostałe wierzchołki tak długo, aż odpowiednie przekątne $\{v_i, v'_j\}$ są wewnątrz wielokąta; zgłaszaj te przekątne.
   Włóż ostatnio zdjęty wierzchołek na stos.
   Włóż v_i na stos.
6. Dodaj przekątne z v_n do wszystkich wierzchołków na stosie, oprócz pierwszego i ostatniego.



v jest dodany do nowego leja

Analiza złożoności obliczeniowej

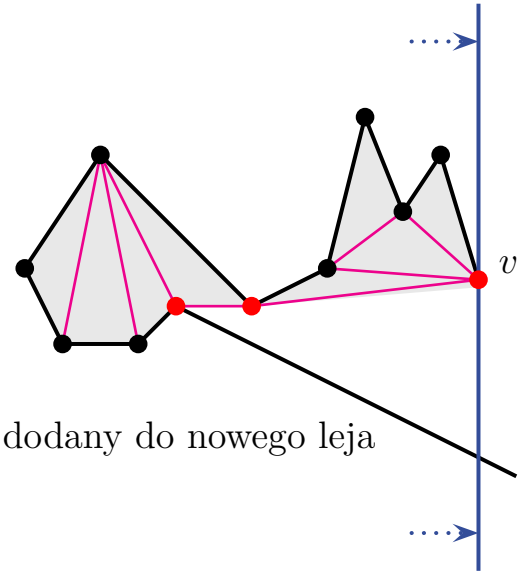
- Złożoność czasowa rzędu $O(n)$.
  - ↳ Pętla **for** wykonywana jest $n - 3$ razy, a jedno wywołanie może wymagać czasu $O(n)$. Ale przy każdym obiegu pętli wstawiane są dwa wierzchołki. Zatem całkowita liczba wstawień, włączając dwa wstawienia w wierszu 1, wynosi $2n - 4$. Ponieważ liczba usunięć nie przewyższa liczby wstawień, całkowity czas wszystkich wykonań pętli **for** wynosi $O(n)$.
- Złożoność pamięciowa: rozmiar stosu rzędu $O(n)$.

Algorytm SweepTriangulation

Wejście. Zbiór wierzchołków wielokąta $P = \{v_1, \ldots, v_n\}$ na płaszczyźnie, $n \geq 3$. P jest posortowany względem odciętej.

Wyjście. Podział P na trójkąty.

1. Inicjuj pusty stos S i włóż na niego v_1 i v_2.
2. **for** $i := 3$ **to** $n - 1$ **do**
3. **if** (v_i oraz wierzchołek na szczycie stosu są na różnych łańcuchach)
4. **then** Zdejmij wszystkie wierzchołki ze stosu $S = (v'_1, \ldots, v'_k)$.
   Zgłoś wszystkie przekątne $\{v_i, v'_j\}$, $j = 2, \ldots, k$.
   Włóż v'_k i v_i na stos.
5. **else** Zdejmij wierzchołek v'_k ze stosu $S = (v'_1, \ldots, v'_k)$.
   Zdejmuj pozostałe wierzchołki tak długo, aż odpowiednie przekątne $\{v_i, v'_j\}$ są wewnątrz wielokąta; zgłaszaj te przekątne.
   Włóż ostatnio zdjęty wierzchołek na stos.
   Włóż v_i na stos.
6. Dodaj przekątne z v_n do wszystkich wierzchołków na stosie, oprócz pierwszego i ostatniego.

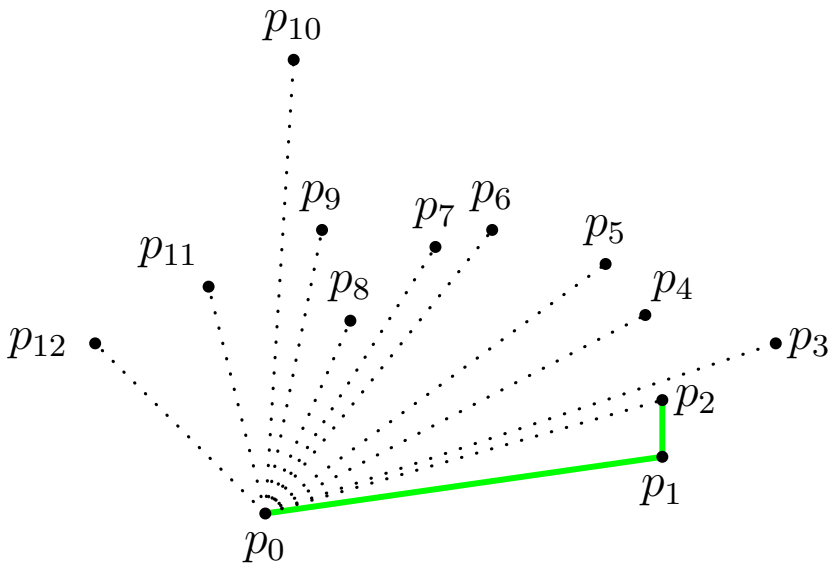


v jest dodany do nowego leja

Twierdzenie 2.5. (Garey, Johnson, Preparata, Tarjan 1978)
Wielokąt ściśle x-monotoniczny o n wierzchołkach można podzielić na trójkąty przez dodanie nieprzecinających się wewnętrznych przekątnych łączących wierzchołki wielokąta w czasie i pamięci rzędu $O(n)$.

2.5 PROBLEM OTOCZKI WYPUKŁEJ – skan Grahama

Otoczka wypukła zbioru punktów $S \subset \mathbb{R}^2$, ozn. $\mathrm{CH}(S)$, to najmniejszy wielokąt wypukły P taki, że każdy punkt ze zbioru S należy do wielokąta P.

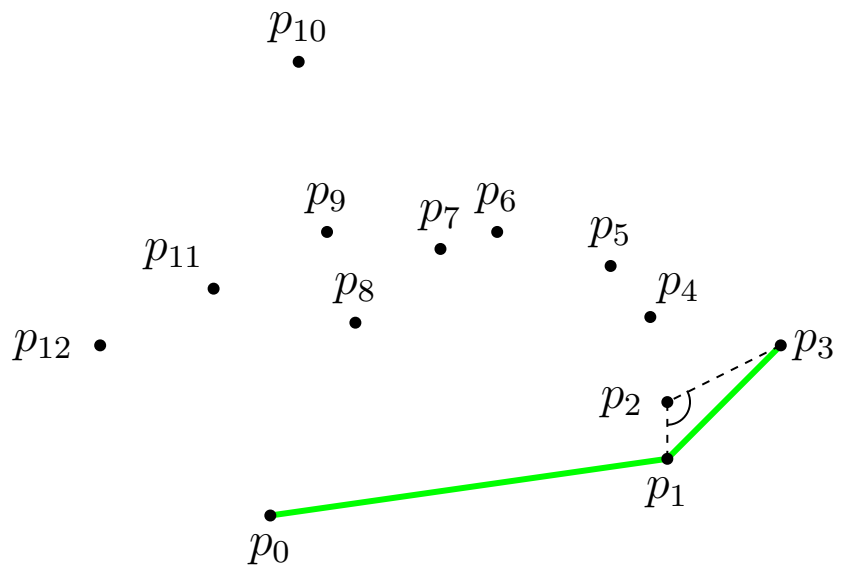


Idea skanu Grahama (1972)

- Obracamy półprostą wokół najniższego punktu.
- Punktami zdarzeń są kolejno napotykane wierzchołki.
- Status miotły przechowywany jest na stosie, który zawiera wierzchołki tworzące otoczkę wypukłą dla wierzchołków do tej pory przetworzonych.
- Jeśli przetwarzany wierzchołek nie tworzy skrętu w prawo ze szczytem stosu oraz z elementem bezpośrednio pod nim, stos musi być uaktualniony.

2.5 PROBLEM OTOCZKI WYPUKŁEJ – skan Grahama

Otoczka wypukła zbioru punktów $S \subset \mathbb{R}^2$, ozn. $\mathrm{CH}(S)$, to najmniejszy wielokąt wypukły P taki, że każdy punkt ze zbioru S należy do wielokąta P.

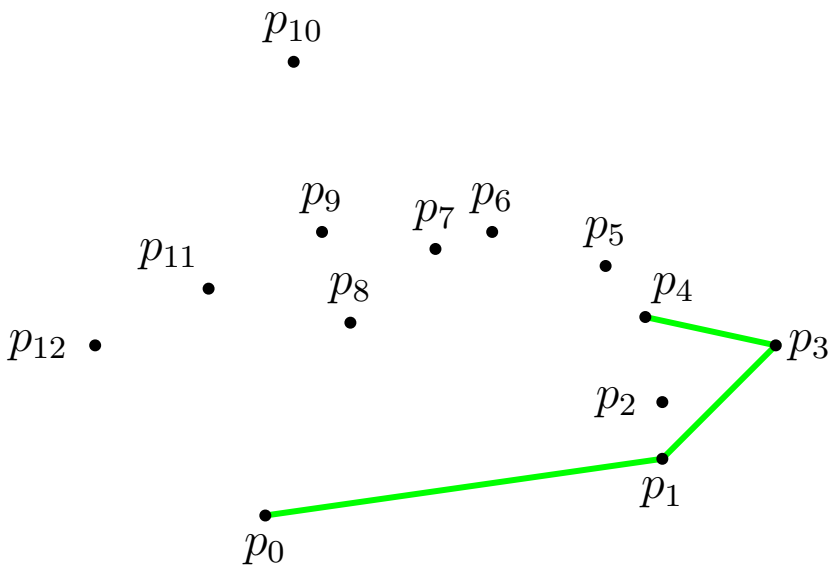


Idea skanu Grahama (1972)

- Obracamy półprostą wokół najniższego punktu.
- Punktami zdarzeń są kolejno napotykane wierzchołki.
- Status miotły przechowywany jest na stosie, który zawiera wierzchołki tworzące otoczkę wypukłą dla wierzchołków do tej pory przetworzonych.
- Jeśli przetwarzany wierzchołek nie tworzy skrętu w prawo ze szczytem stosu oraz z elementem bezpośrednio pod nim, stos musi być uaktualniony.

2.5 PROBLEM OTOCZKI WYPUKŁEJ – skan Grahama

Otoczka wypukła zbioru punktów $S \subset \mathbb{R}^2$, ozn. $\mathrm{CH}(S)$, to najmniejszy wielokąt wypukły P taki, że każdy punkt ze zbioru S należy do wielokąta P.

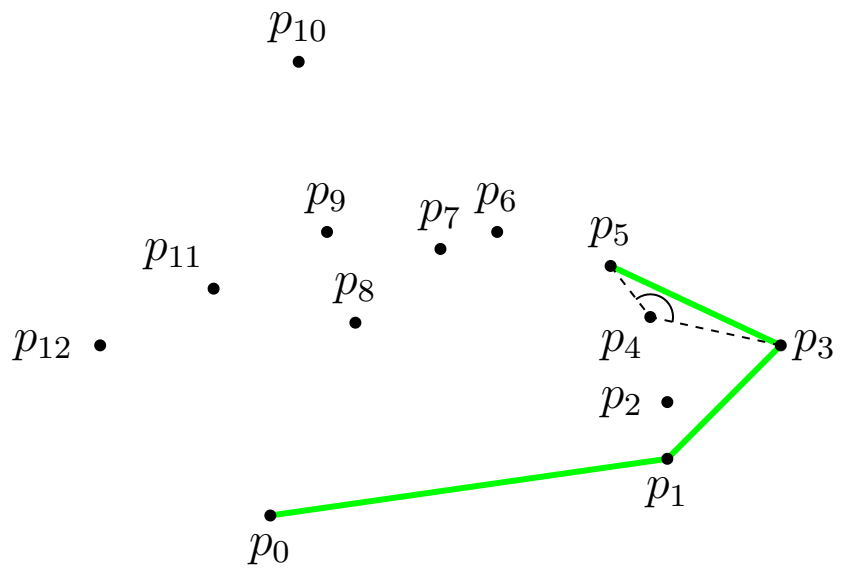


Idea skanu Grahama (1972)

- Obracamy półprostą wokół najniższego punktu.
- Punktami zdarzeń są kolejno napotykane wierzchołki.
- Status miotły przechowywany jest na stosie, który zawiera wierzchołki tworzące otoczkę wypukłą dla wierzchołków do tej pory przetworzonych.
- Jeśli przetwarzany wierzchołek nie tworzy skrętu w prawo ze szczytem stosu oraz z elementem bezpośrednio pod nim, stos musi być uaktualniony.

2.5 PROBLEM OTOCZKI WYPUKŁEJ – skan Grahama

Otoczka wypukła zbioru punktów $S \subset \mathbb{R}^2$, ozn. $\mathrm{CH}(S)$, to najmniejszy wielokąt wypukły P taki, że każdy punkt ze zbioru S należy do wielokąta P.

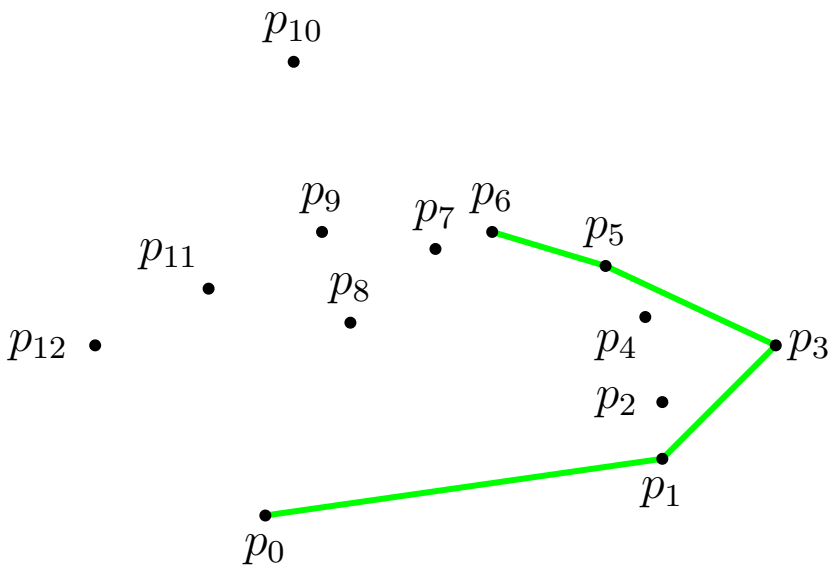


Idea skanu Grahama (1972)

- Obracamy półprostą wokół najniższego punktu.
- Punktami zdarzeń są kolejno napotykane wierzchołki.
- Status miotły przechowywany jest na stosie, który zawiera wierzchołki tworzące otoczkę wypukłą dla wierzchołków do tej pory przetworzonych.
- Jeśli przetwarzany wierzchołek nie tworzy skrętu w prawo ze szczytem stosu oraz z elementem bezpośrednio pod nim, stos musi być uaktualniony.

2.5 PROBLEM OTOCZKI WYPUKŁEJ – skan Grahama

Otoczka wypukła zbioru punktów $S \subset \mathbb{R}^2$, ozn. $\mathrm{CH}(S)$, to najmniejszy wielokąt wypukły P taki, że każdy punkt ze zbioru S należy do wielokąta P.

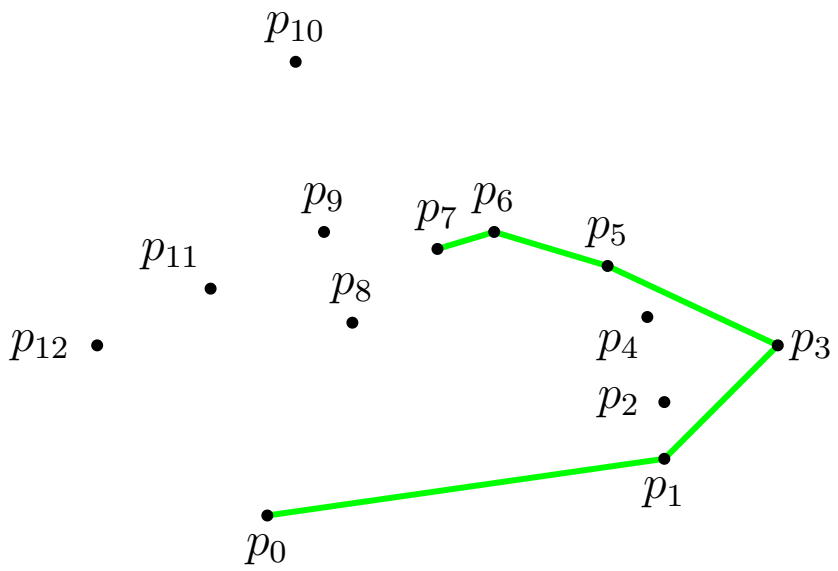


Idea skanu Grahama (1972)

- Obracamy półprostą wokół najniższego punktu.
- Punktami zdarzeń są kolejno napotykane wierzchołki.
- Status miotły przechowywany jest na stosie, który zawiera wierzchołki tworzące otoczkę wypukłą dla wierzchołków do tej pory przetworzonych.
- Jeśli przetwarzany wierzchołek nie tworzy skrętu w prawo ze szczytem stosu oraz z elementem bezpośrednio pod nim, stos musi być uaktualniony.

2.5 PROBLEM OTOCZKI WYPUKŁEJ – skan Grahama

Otoczka wypukła zbioru punktów $S \subset \mathbb{R}^2$, ozn. $\mathrm{CH}(S)$, to najmniejszy wielokąt wypukły P taki, że każdy punkt ze zbioru S należy do wielokąta P.

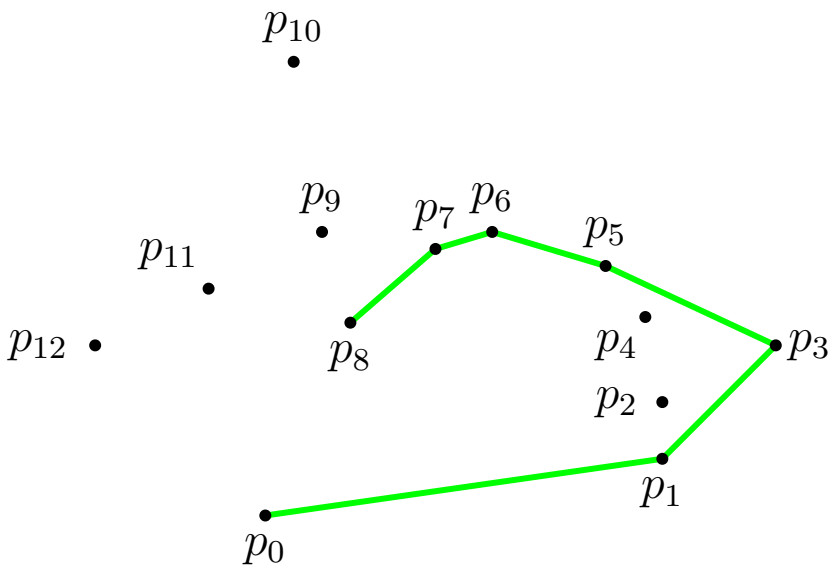


Idea skanu Grahama (1972)

- Obracamy półprostą wokół najniższego punktu.
- Punktami zdarzeń są kolejno napotykane wierzchołki.
- Status miotły przechowywany jest na stosie, który zawiera wierzchołki tworzące otoczkę wypukłą dla wierzchołków do tej pory przetworzonych.
- Jeśli przetwarzany wierzchołek nie tworzy skrętu w prawo ze szczytem stosu oraz z elementem bezpośrednio pod nim, stos musi być uaktualniony.

2.5 PROBLEM OTOCZKI WYPUKŁEJ – skan Grahama

Otoczka wypukła zbioru punktów $S \subset \mathbb{R}^2$, ozn. $\mathrm{CH}(S)$, to najmniejszy wielokąt wypukły P taki, że każdy punkt ze zbioru S należy do wielokąta P.

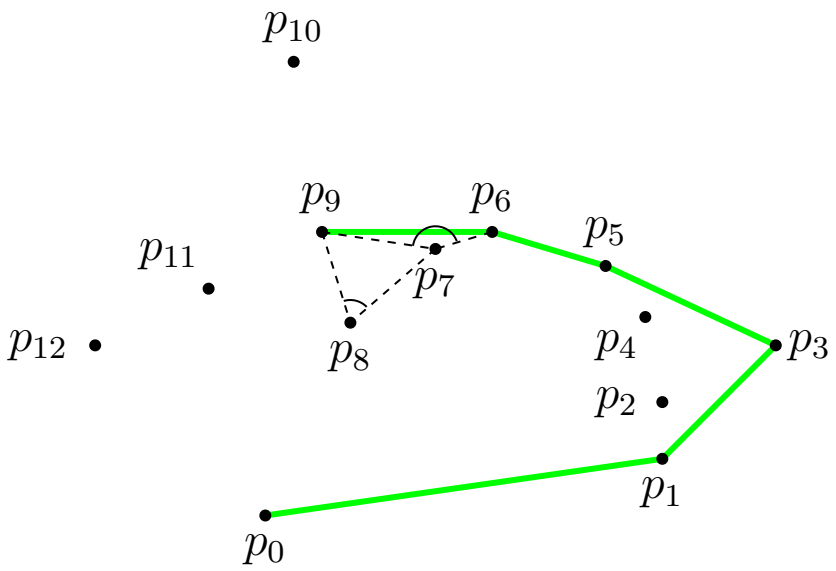


Idea skanu Grahama (1972)

- Obracamy półprostą wokół najniższego punktu.
- Punktami zdarzeń są kolejno napotykane wierzchołki.
- Status miotły przechowywany jest na stosie, który zawiera wierzchołki tworzące otoczkę wypukłą dla wierzchołków do tej pory przetworzonych.
- Jeśli przetwarzany wierzchołek nie tworzy skrętu w prawo ze szczytem stosu oraz z elementem bezpośrednio pod nim, stos musi być uaktualniony.

2.5 PROBLEM OTOCZKI WYPUKŁEJ – skan Grahama

Otoczka wypukła zbioru punktów $S \subset \mathbb{R}^2$, ozn. $\mathrm{CH}(S)$, to najmniejszy wielokąt wypukły P taki, że każdy punkt ze zbioru S należy do wielokąta P.

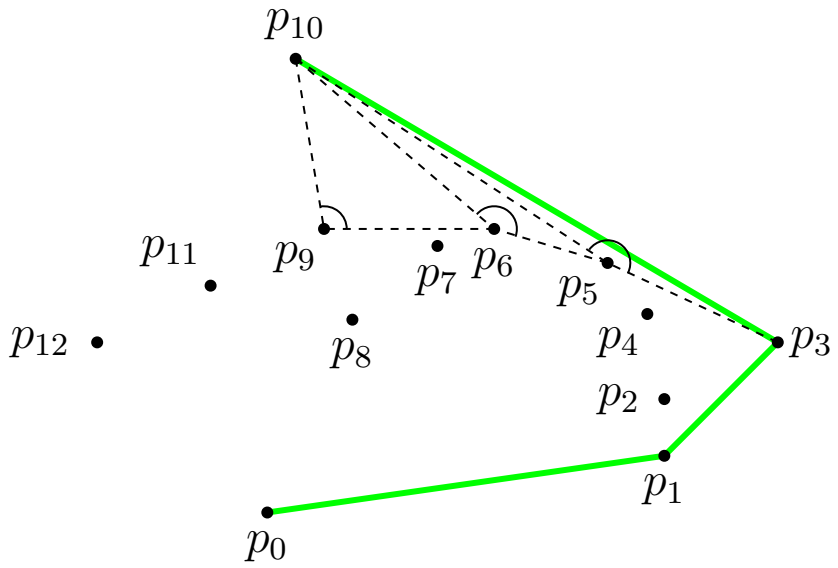


Idea skanu Grahama (1972)

- Obracamy półprostą wokół najniższego punktu.
- Punktami zdarzeń są kolejno napotykane wierzchołki.
- Status miotły przechowywany jest na stosie, który zawiera wierzchołki tworzące otoczkę wypukłą dla wierzchołków do tej pory przetworzonych.
- Jeśli przetwarzany wierzchołek nie tworzy skrętu w prawo ze szczytem stosu oraz z elementem bezpośrednio pod nim, stos musi być uaktualniony.

2.5 PROBLEM OTOCZKI WYPUKŁEJ – skan Grahama

Otoczka wypukła zbioru punktów $S \subset \mathbb{R}^2$, ozn. $\mathrm{CH}(S)$, to najmniejszy wielokąt wypukły P taki, że każdy punkt ze zbioru S należy do wielokąta P.

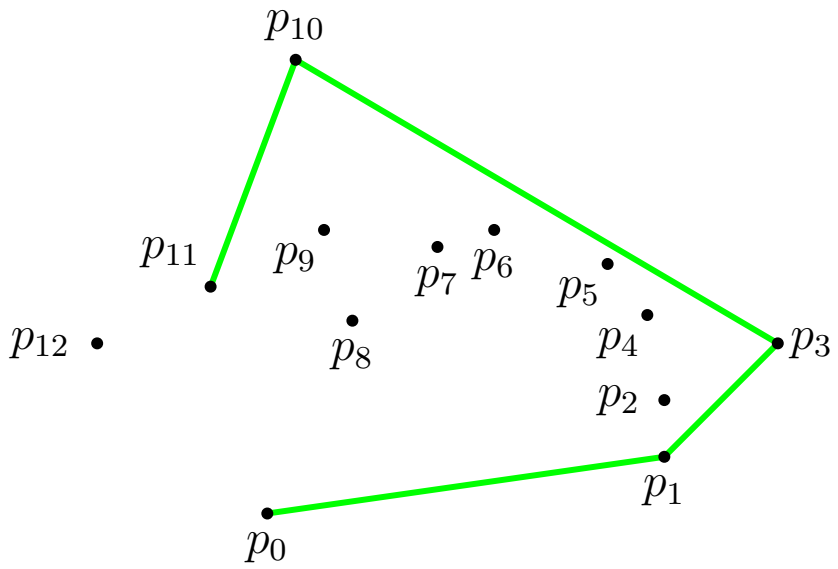


Idea skanu Grahama (1972)

- Obracamy półprostą wokół najniższego punktu.
- Punktami zdarzeń są kolejno napotykane wierzchołki.
- Status miotły przechowywany jest na stosie, który zawiera wierzchołki tworzące otoczkę wypukłą dla wierzchołków do tej pory przetworzonych.
- Jeśli przetwarzany wierzchołek nie tworzy skrętu w prawo ze szczytem stosu oraz z elementem bezpośrednio pod nim, stos musi być uaktualniony.

2.5 PROBLEM OTOCZKI WYPUKŁEJ – skan Grahama

Otoczka wypukła zbioru punktów $S \subset \mathbb{R}^2$, ozn. $\mathrm{CH}(S)$, to najmniejszy wielokąt wypukły P taki, że każdy punkt ze zbioru S należy do wielokąta P.

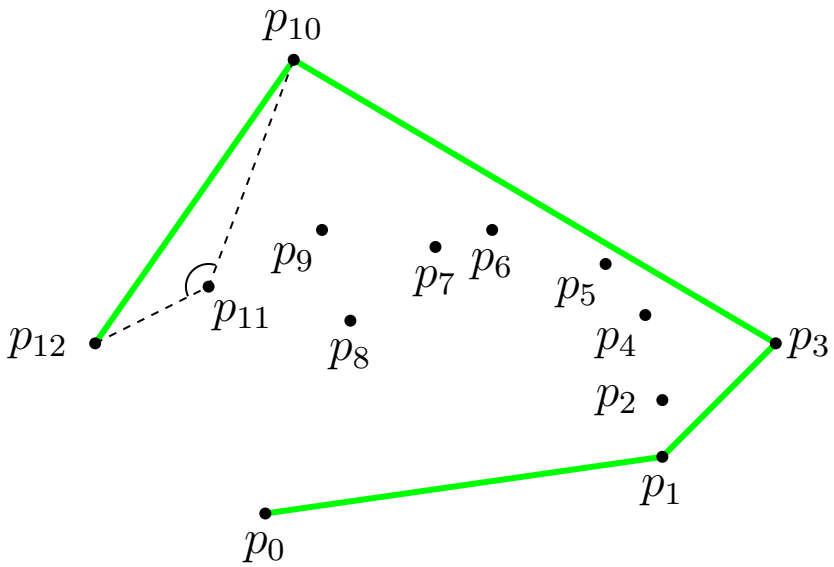


Idea skanu Grahama (1972)

- Obracamy półprostą wokół najniższego punktu.
- Punktami zdarzeń są kolejno napotykane wierzchołki.
- Status miotły przechowywany jest na stosie, który zawiera wierzchołki tworzące otoczkę wypukłą dla wierzchołków do tej pory przetworzonych.
- Jeśli przetwarzany wierzchołek nie tworzy skrętu w prawo ze szczytem stosu oraz z elementem bezpośrednio pod nim, stos musi być uaktualniony.

2.5 PROBLEM OTOCZKI WYPUKŁEJ – skan Grahama

Otoczka wypukła zbioru punktów $S \subset \mathbb{R}^2$, ozn. $\mathrm{CH}(S)$, to najmniejszy wielokąt wypukły P taki, że każdy punkt ze zbioru S należy do wielokąta P.

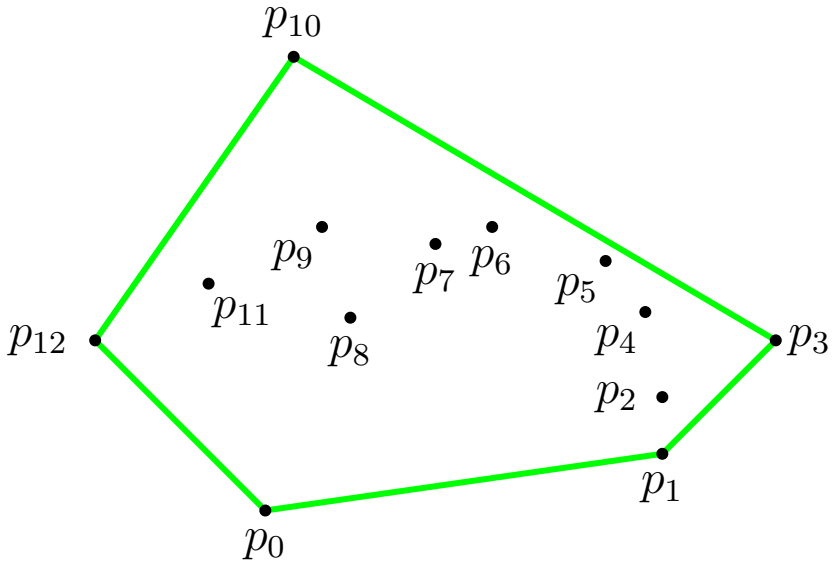


Idea skanu Grahama (1972)

- Obracamy półprostą wokół najniższego punktu.
- Punktami zdarzeń są kolejno napotykane wierzchołki.
- Status miotły przechowywany jest na stosie, który zawiera wierzchołki tworzące otoczkę wypukłą dla wierzchołków do tej pory przetworzonych.
- Jeśli przetwarzany wierzchołek nie tworzy skrętu w prawo ze szczytem stosu oraz z elementem bezpośrednio pod nim, stos musi być uaktualniony.

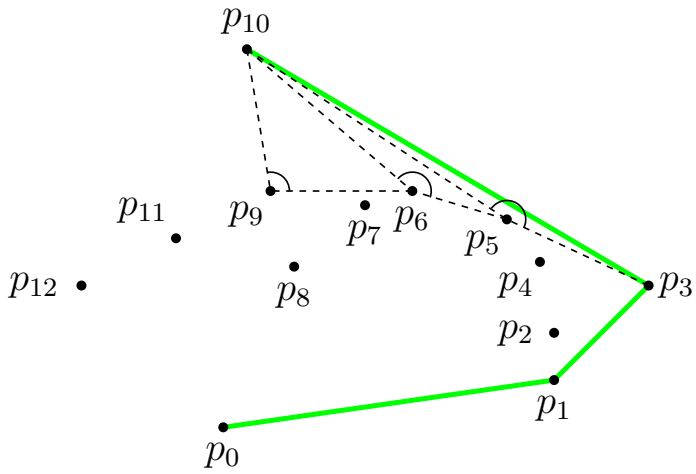
2.5 PROBLEM OTOCZKI WYPUKŁEJ – skan Grahama

Otoczka wypukła zbioru punktów $S \subset \mathbb{R}^2$, ozn. $\mathrm{CH}(S)$, to najmniejszy wielokąt wypukły P taki, że każdy punkt ze zbioru S należy do wielokąta P.



Idea skanu Grahama (1972)

- Obracamy półprostą wokół najniższego punktu.
- Punktami zdarzeń są kolejno napotykane wierzchołki.
- Status miotły przechowywany jest na stosie, który zawiera wierzchołki tworzące otoczkę wypukłą dla wierzchołków do tej pory przetworzonych.
- Jeśli przetwarzany wierzchołek nie tworzy skrętu w prawo ze szczytem stosu oraz z elementem bezpośrednio pod nim, stos musi być uaktualniony.

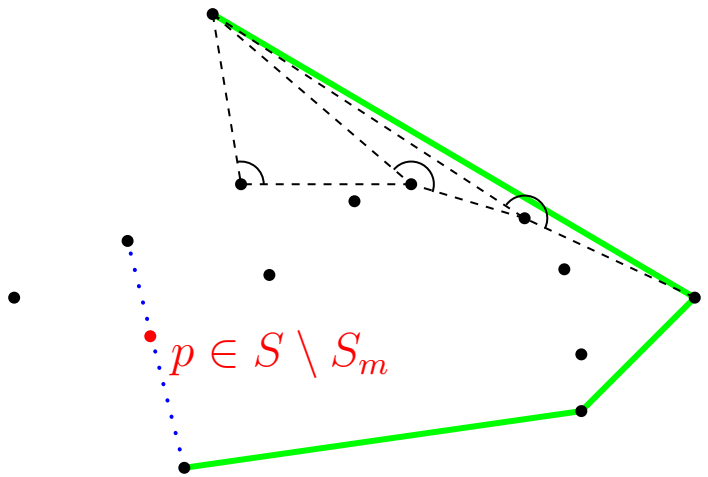


GrahamScan(S)
Wejście. Zbiór n punktów $S \subset \mathbb{R}^2$.
Wyjście. Otoczka wypukła $\mathrm{CH}(S)$ zbioru S.

1. Niech p_0 będzie punktem w S o najmniejszej współrzędnej y; w przypadku remisu − pierwszym takim punktem z lewej.
2. Niech $p_1, p_2, \ldots, p_m$ będą pozostałymi punktami z $S \setminus \{p_0\}$, posortowanymi ze względu na współrzędną kątową względem p_0 w kierunku przeciwnym do ruchu wskazówek zegara; jeśli więcej niż jeden punkt ma taką samą współrzędna kątową, usuń wszystkie takie punkty za wyjątkiem punktu położonego najdalej od p_0.
3. PUSH$(p_0, \textsc{Stos})$; PUSH$(p_1, \textsc{Stos})$; PUSH$(p_2, \textsc{Stos})$;
4. **for** $i = 3$ **to** m **do**
5. **while** kąt utworzony przez punkty p_i ,TOP$(\textsc{Stos})$ oraz NEXT-TO-TOP$(\textsc{Stos})$ nie stanowi skrętu w prawo
6. **do** POP$(\textsc{Stos})$;
7. PUSH$(p_i, \textsc{Stos})$;
8. **return** Stos.

Analiza poprawności algorytmu

- Dla $i = 2, 3, \ldots, m$, niech $S_i := \{p_1, p_2, \ldots, p_i\}$.
- Otoczki wypukłe $\mathrm{CH}(S_m)$ oraz $\mathrm{CH}(\mathrm{S})$ są tymi samymi otoczkami.

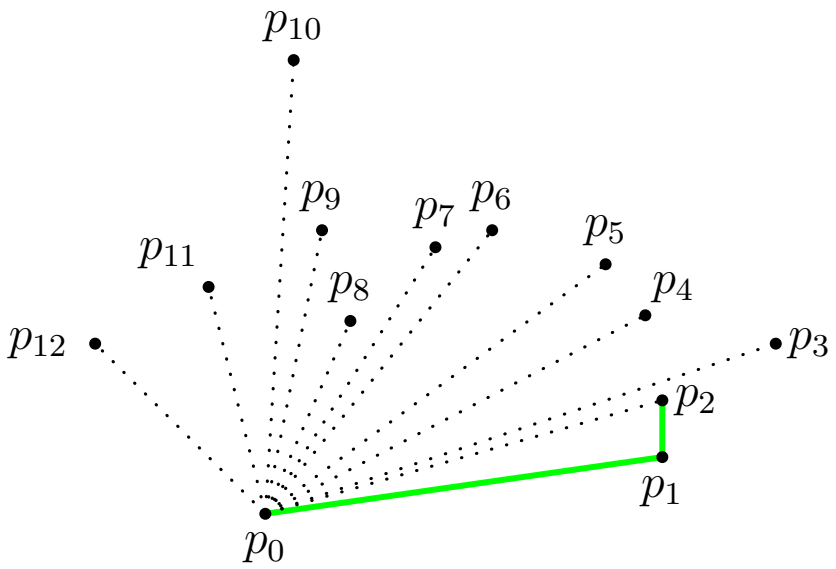


Punkt p nie należy do otoczki $\mathrm{CH}(S)$.

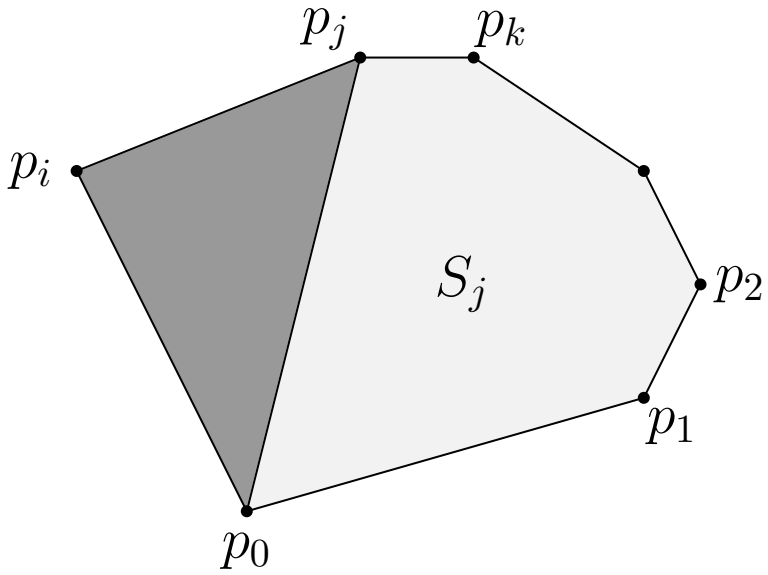
Analiza poprawności algorytmu

- Dla $i = 2, 3, \ldots, m$, niech $S_i := \{p_1, p_2, \ldots, p_i\}$.
- Otoczki wypukłe $\mathrm{CH}(S_m)$ oraz $\mathrm{CH}(\mathrm{S})$ są tymi samymi otoczkami.

Niezmiennik. Na początku każdej iteracji pętli **for** w wierszach 4-7 STOS zawiera, patrząc od dołu, wszystkie punkty otoczki $\mathrm{CH}(S_{i-1})$ w kolejności odwrotnej do ruchu wskazówek zegara i tylko te punkty.

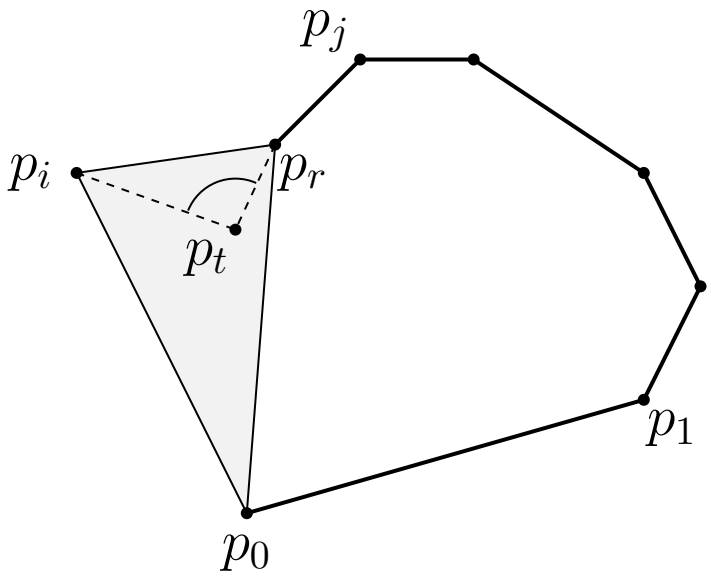


- Niezmiennik jest spełniony przy pierwszym wykonaniu wiersza 4.



- Niech p_j będzie punktem na wierzchołku STOSU po zakończeniu pętli **while** w wierszach 5-6, ale przed włożeniem p_i na STOS w wierszu 7, i niech p_k będzie punktem poniżej p_j na STOSIE.

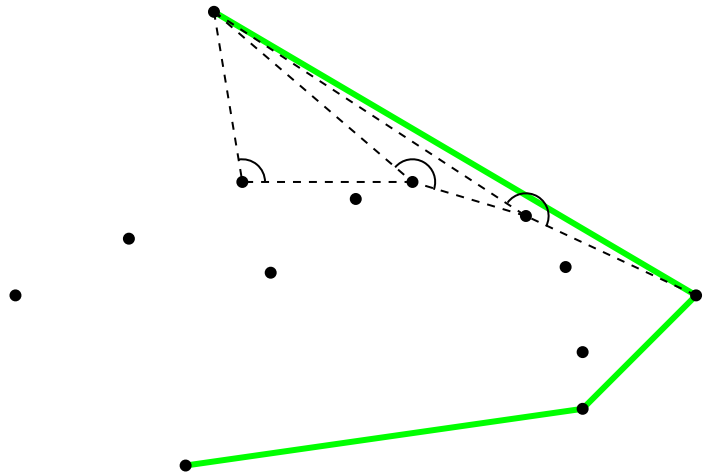
- Kąt $\angle p_i p_j p_k$ stanowi skręt w prawo, a tym samym po włożeniu p_i na STOS, stos ten zawiera dokładnie wierzchołki $\mathrm{CH}(S_j \cup \{p_i\})$.



- Rozważmy dowolny punkt p_t, który został zdjęty ze stosu podczas i-tej iteracji pętli **for**, i niech p_r będzie punktem tuż poniżej p_t na STOSIE w chwili, kiedy p_t był zdejmowany (p_r to być może p_j).

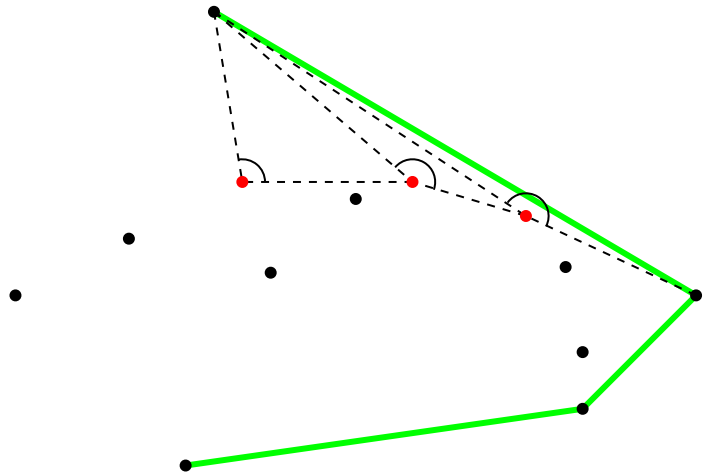
- Kąt $\angle p_i p_t p_r$ nie jest skrętem w prawo, punkt p_t leży wewnątrz trójkąta $p_0 p_r p_i$ albo na jego brzegu. Jako że $p_0, p_r, p_i \in S_i$, zatem $\mathrm{CH}(S_i) = \mathrm{CH}(S_i \setminus \{p_t\})$. Jako że p_t był dowolnym punktem zdjętym ze stosu w wierszu 6, otrzymujemy, że $\mathrm{CH}(S_i) = \mathrm{CH}(S_i \setminus \{P_i\})$, gdzie P_i jest zbiorem punktów zdjętych ze STOSU podczas i-tej iteracji pętli `for`. A skoro $S_i \setminus P_i = S_j \cup \{p_i\}$, otrzymujemy

$$\mathrm{CH}(S_i) = \mathrm{CH}(S_j \cup \{p_i\}).$$



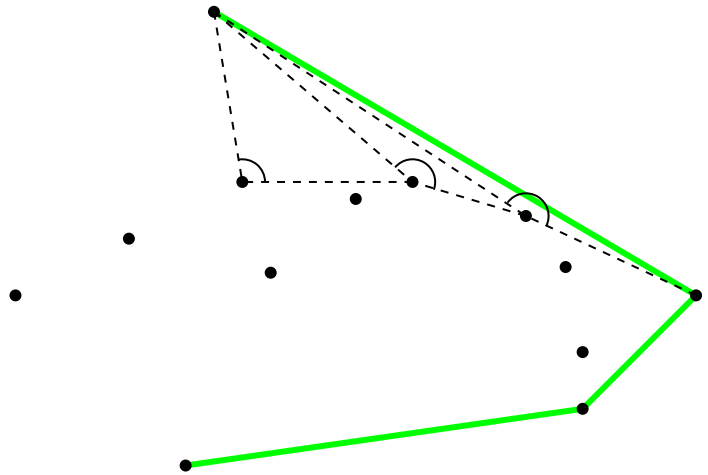
Analiza złożoności czasowej algorytmu

- Wyznaczenie p_0: czas rzędu $\Theta(n)$.
- Sortowanie punktów oraz usunięcie współliniowych punktów: $O(n \log n)$.
- Wstawienie p_0, p_1, p_2: czas $O(1)$.
- Pętla **for** w wierszach 4-7 wykonywana jest $m \leq n - 3$ razy.
  Przy pominięciu czasu dla zagnieżdżonej pętli **while**: czas $O(n)$.



Analiza złożoności czasowej algorytmu

- Wyznaczenie p_0: czas rzędu $\Theta(n)$.
- Sortowanie punktów oraz usunięcie współliniowych punktów: $O(n \log n)$.
- Wstawienie p_0, p_1, p_2: czas $O(1)$.
- Pętla **for** w wierszach 4-7 wykonywana jest $m \leq n - 3$ razy.
  Przy pominięciu czasu dla zagnieżdżonej pętli **while**: czas $O(n)$.
- Całkowity czas wykonywania pętli **while**: $O(n)$.
  - ↳ Każdy punkt wkładany jest na stos dokładnie raz, a na każdą operację PUSH przypada co najwyżej jedna operacja POP. A zatem wykonanych zostaje co najwyżej $m - 2$ operacji POP.
  - ↳ W każdym przebiegu pętli **while** wykonuje się jedna operacja POP, a zatem tych przebiegów może być co najwyżej $m - 2 \leq n - 3$.



Analiza złożoności czasowej algorytmu

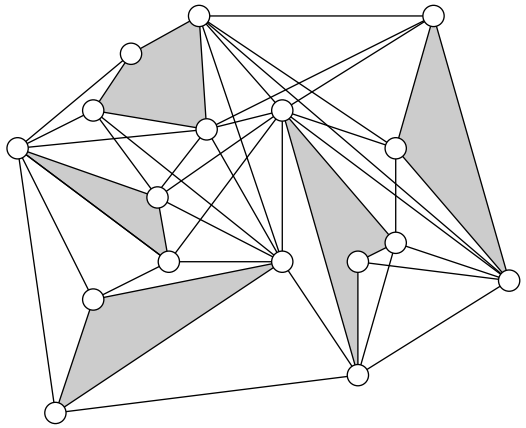
- Wyznaczenie p_0: czas rzędu $\Theta(n)$.
- Sortowanie punktów oraz usunięcie współliniowych punktów: $O(n \log n)$.
- Wstawienie p_0, p_1, p_2: czas $O(1)$.
- Pętla **for** w wierszach 4-7 wykonywana jest $m \leq n - 3$ razy.
  Przy pominięciu czasu dla zagnieżdżonej pętli **while**: czas $O(n)$.
- Całkowity czas wykonywania pętli **while**: $O(n)$.
  - ↳ Każdy punkt wkładany jest na stos dokładnie raz, a na każdą operację PUSH przypada co najwyżej jedna operacja POP. A zatem wykonanych zostaje co najwyżej $m - 2$ operacji POP.
  - ↳ W każdym przebiegu pętli **while** wykonuje się jedna operacja POP, a zatem tych przebiegów może być co najwyżej $m - 2 \leq n - 3$.

Twierdzenie 2.6. Skan Grahama wyznacza otoczkę wypukłą n punktów na płaszczyźnie w czasie rzędu $O(n \log n)$; skan ten używa pamięci liniowej.

2.6* GRAF WIDZIALNOŚCI NA PŁASZCZYŹNIE

Dla danego zbioru S rozłącznych wielokątów na płaszczyźnie graf widzialności $G_{\text{widz}}(S)$ jest grafem, którego wierzchołkami są wierzchołki wielokątów z S, a dwa wierzchołki v_1 i v_2 połączone są krawędzią wtedy i tylko wtedy, gdy widzą się nawzajem, tzn. odcinek $\overline{v_1 v_2}$ nie przecina wnętrza żadnego z wielokątów z S.

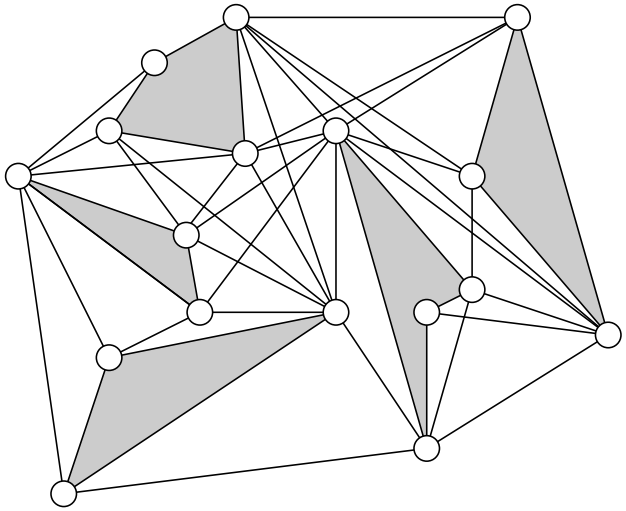
M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf
Geometria obliczeniowa, rozdział 15, WNT (2007)

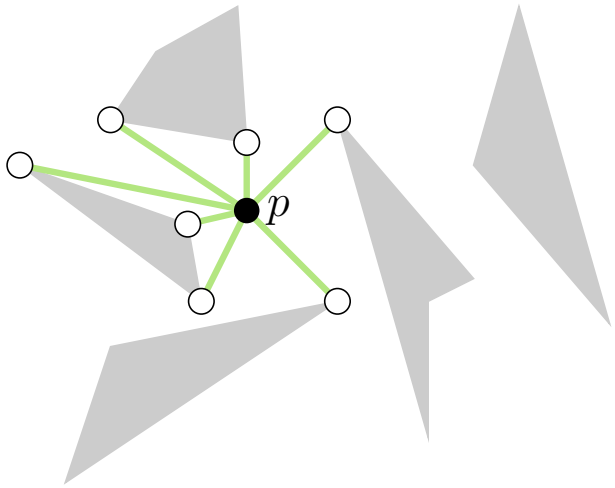


Problem. Dla danego zbioru S rozłącznych wielokątów na płaszczyźnie $\mathbb{R}^2$ wyznaczyć graf widzialności $G_{\text{widz}}(S)$.

Algorytm naiwny 1

▶ Dla każdej pary wierzchołków sprawdzić, czy łączący je odcinek przecina jakikolwiek wielokąt z S. /Złożoność czasowa: $O(n^3)$./





Algorytm naiwny 2

▶ Dla każdego wierzchołka v wyznacz zbiór wierzchołków widzialnych z v.

Problem. (Widzialność z punktu p)

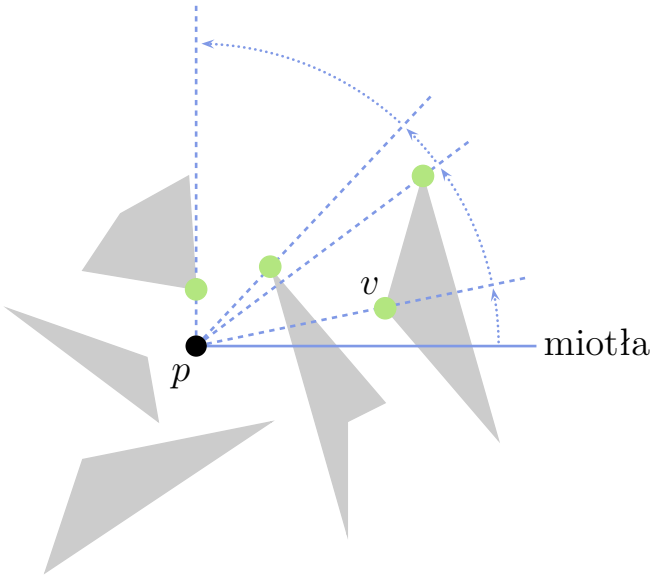
Wejście: *Zbiór S rozłącznych wielokątów na $\mathbb{R}^2$ oraz punkt $p \in \mathbb{R}^2$.*

Wyjście: *Zbiór wierzchołków wielokątów widzialnych z p.*

Obliczanie wierzchołków widzialnych z dowolnego punktu

- Jeśli chcemy sprawdzić, czy jeden wyszczególniony wierzchołek v jest widzialny z p, wówczas w najgorszym przypadku potrzebujemy czasu liniowego.

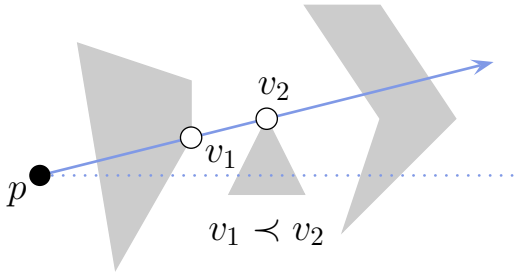
Idea algorytmu zamiatania biegunowego

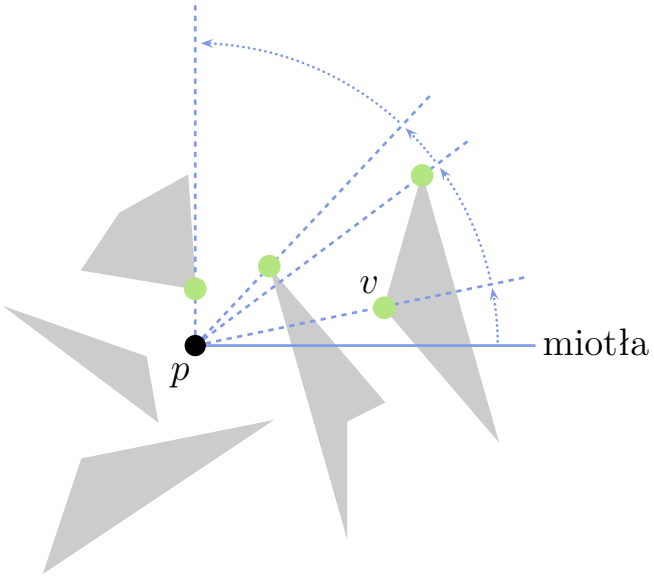


- Miotła o początku w punkcie p obraca się przeciwnie do ruchu wsk. zegara.
- Status miotły stanowi zbiór przecinanych odcinków i zmienia się on przy rozpatrywaniu kolejnych wierzchołków.
- Napotykając nowy wierzchołek v, sprawdzane jest, czy odcinek $\overline{pv}$ przecinany jest przez krawędź przechowywaną w statusie.
- Ze statusu usuwane są krawędzie incydentne do v, które leżą po stronie zgodnej z ruchem wskazówek zegara, i dodawane te, które leżą po stronie przeciwnej do ruchu wskazówek zegara.

Porządek $\prec$ wierzchołków.

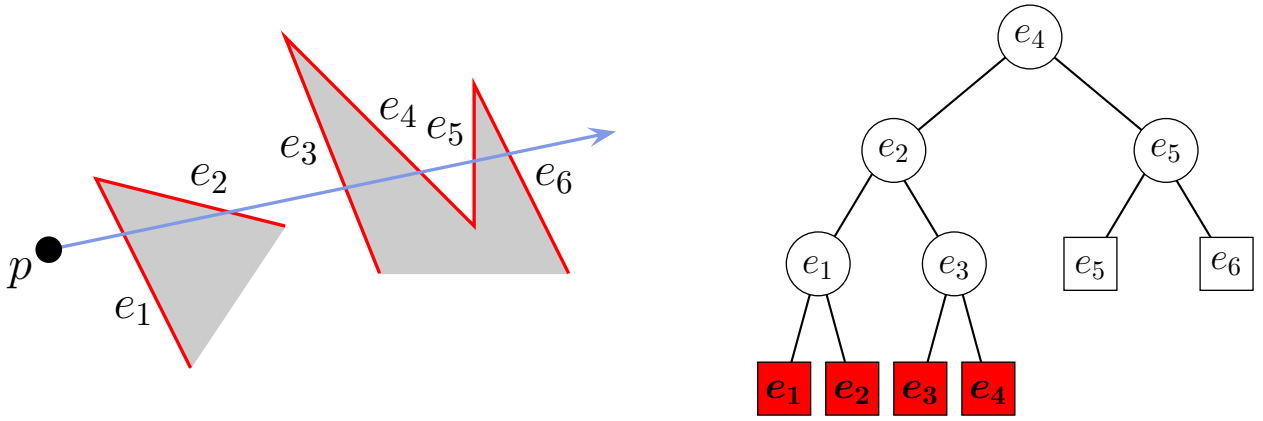
Wierzchołki posortowane są w *porządku biegunowym* względem punktu p; jeśli wierzchołki v_1 i v_2 tworzą ten sam kąt, wówczas $v_1 \prec v_2$ wtedy i tylko wtedy, gdy v_1 leży bliżej punktu p.





Punkty zdarzeń.

Wierzchołki wielokątów ze zbioru S
posortowane w porządku biegunowym względem p.



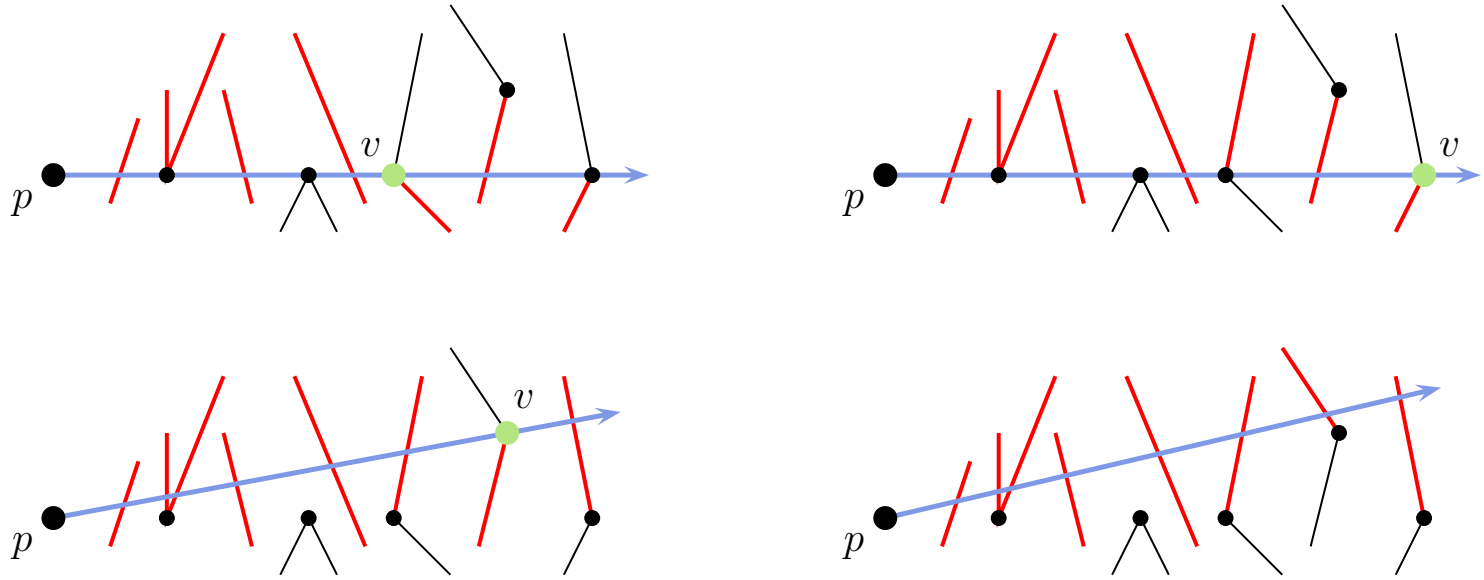
Status miotły. Krawędzie przecinane przez miotłę.

Podczas badania wierzchołków w porządku biegunowym względem p utrzymujemy krawędzie wielokątów przecinane przez miotłę M w zrównoważonym drzewie wyszukiwań binarnych $\mathcal{T}$.

↳ Liście przechowują krawędzie w kolejności przecinania ich przez półprostą M patrząc od jej początku p.

↳ Węzły wewnętrzne wskazują na krawędzie będące prawymi skrajnymi w lewym poddrzewie.

↳ Krawędzie, których oba końce leżą na M, nie muszą być pamiętane.

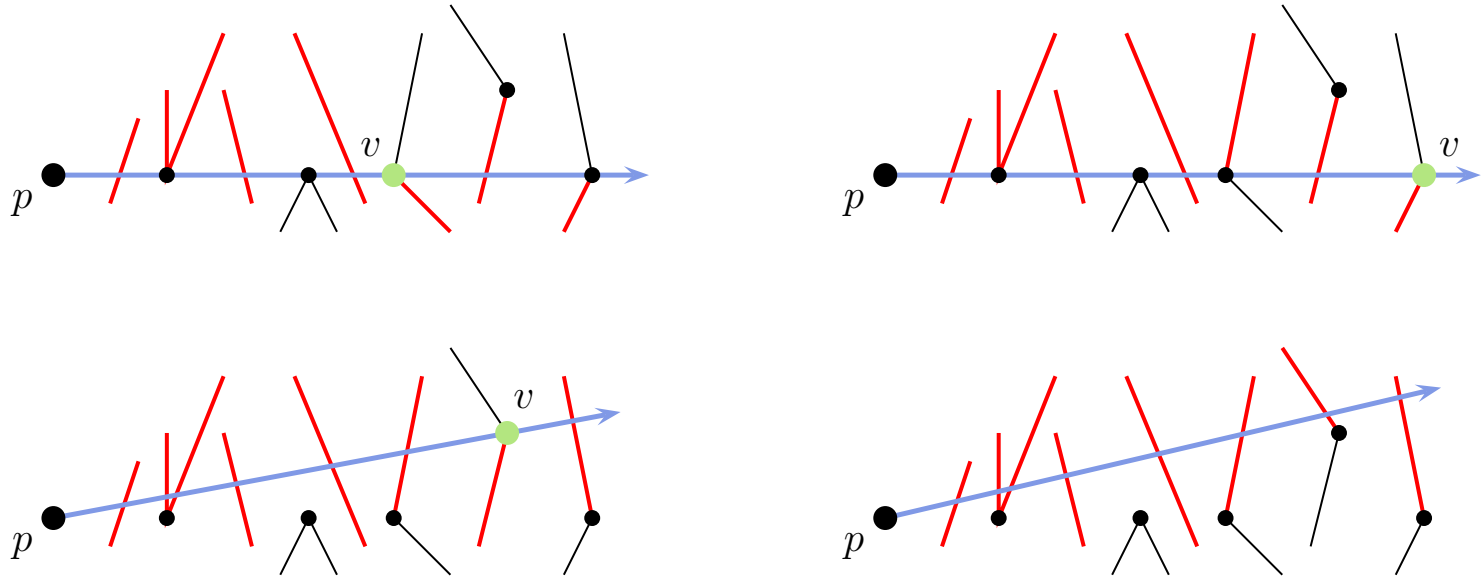


Zmiana statusu. Miotła napotyka nowy wierzchołek v.

↳ Sprawdzenie widzialności między p i v.

↳ Usunięcie ze statusu krawędzi wielokąta incydentnych z v, które leżą po stronie półprostej z p do v zgodnej z ruchem wskazówek zegara.

↳ Wstawienie do statusu krawędzi wielokąta incydentnych z v, które leżą po stronie półprostej z p do v przeciwnej do ruchu wskazówek zegara.



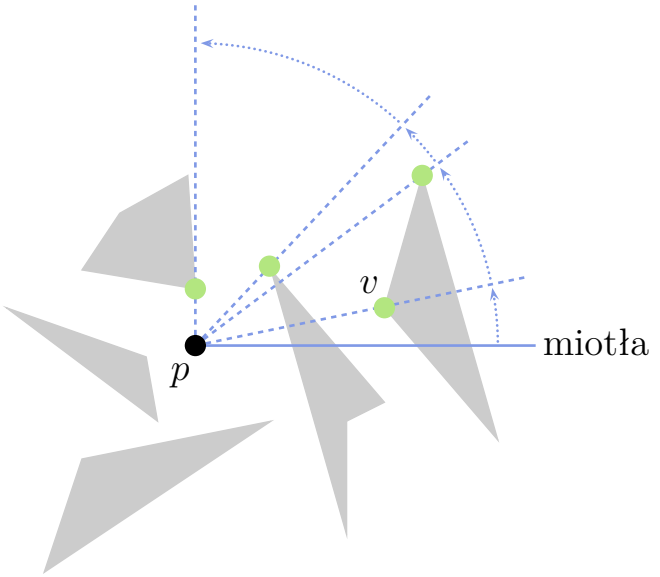
Niezmiennik.

Napotykając nowy wierzchołek v, status na odcinku $\overline{pv}$ zawiera krawędzie *właściwie* przecinane przez miotłę oraz te, które leżą po stronie przeciwnej do ruchu wskazówek zegara, o jednym końcu na odcinku $\overline{pv}$; natomiast na odcinku $\overline{p(+\infty)}$ status zawiera krawędzie *właściwie* przecinane przez miotłę oraz te, które leżą po stronie zgodnej do ruchu wskazówek zegara, o jednym końcu na odcinku $\overline{p(+\infty)}$.

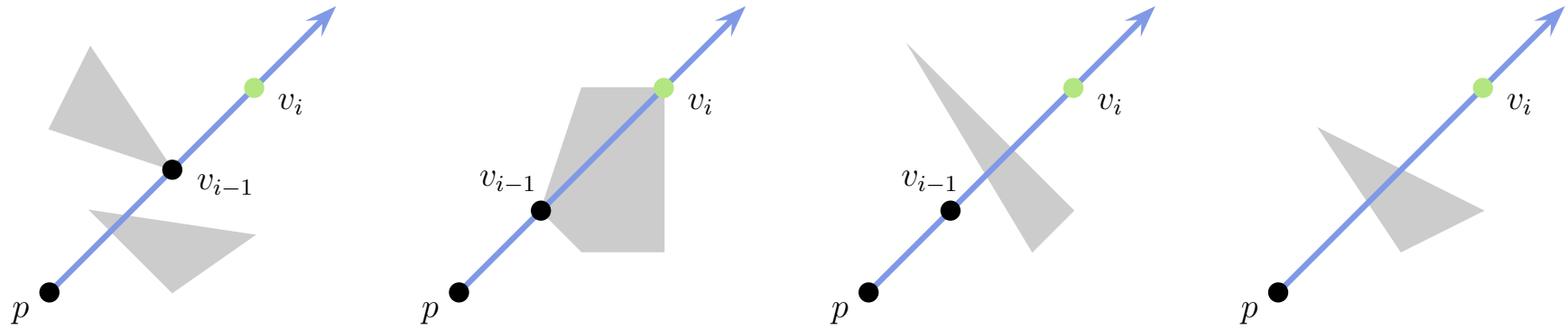
↳ Niezmiennik ten gwarantuje poprawne wyznaczenie widzialności pomiędzy v a kolejnym rozpatrywanym wierzchołkiem.

Idea algorytmu

- Niech $v_1, \ldots, v_n$ będzie listą wierzchołków (wszystkich wielokątów) posortowanych biegunowo, tzn. w kierunku przeciwnym do ruchu wskazówek zegara i zgodnie z kątami, jakie półproste od p do każdego z wierzchołków tworzą z dodatnią osią x; porządek wierzchołków o tym samym kącie determinowany jest przez ich odległość do p.

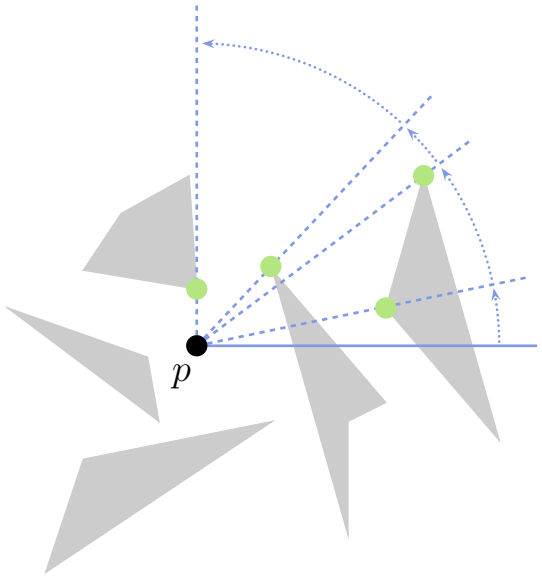


- Wyznaczamy krawędzie wielokątów, które są właściwie przecinane przez poziomą półprostą l o początku w p i zapamiętujemy je w zrównoważonym drzewie wyszukiwań $\mathcal{T}$ w porządku, w jakim są przecinane przez l.
- Dla każdego z wierzchołków $v_1, v_2, \ldots, v_n$ sprawdzamy, czy p widzi v_i.
  - ↳ Usuwamy z $\mathcal{T}$ krawędzie wielokąta incydentne z v_i, które leżą po stronie półprostej z p do v_i zgodnej z ruchem wskazówek zegara.
  - ↳ Wstawiamy do $\mathcal{T}$ krawędzie wielokąta incydentne z v_i, które leżą po stronie półprostej z p do v_i przeciwnej do ruchu wskazówek zegara.



Sprawdzanie widzialności pomiędzy p i v_i

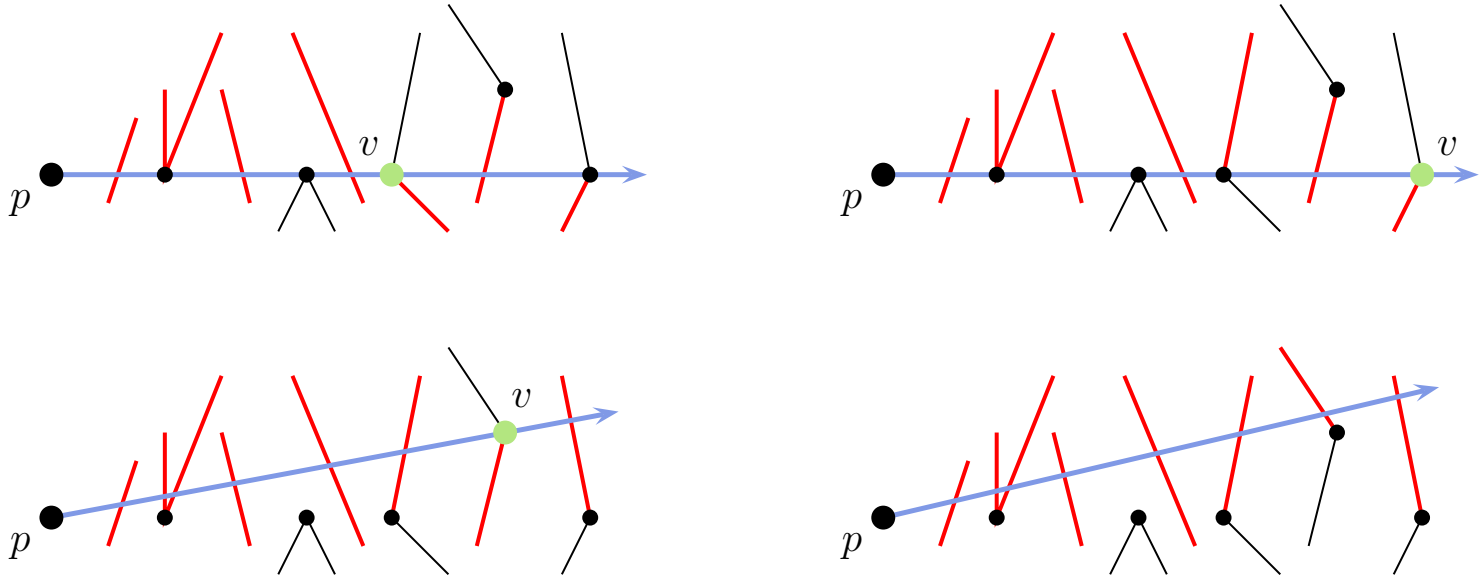
- Jeśli na odcinku $\overline{pv_i}$ leży jakiś inny wierzchołek – spośród wszystkich tych wierzchołków rozważmy wierzchołek w najbliższy v_i. Wówczas, z porządku sortowania, zachodzi $w = v_{i-1}$. A zatem:
  - ↳ Jeśli wierzchołek v_{i-1} nie jest widzialny z p, to v_i też nie.
  - ↳ W przeciwnym wypadku, jeśli v_{i-1} jest widzialny, to zablokowanie widzialności z p do v_i możliwe jest tylko w dwóch przypadkach:
    - ∗ odcinek $\overline{v_{i-1}v_i}$ leży wewnątrz wielokąta;
    - ∗ odcinek $\overline{v_{i-1}v_i}$ przecina (właściwie) krawędź z $\mathcal{T}$.
- Jeśli odcinek $\overline{pv_i}$ przecinany jest we *właściwy* sposób przez jakąś krawędź przechowywaną w statusie $\mathcal{T}$, wówczas p i v_i nie widzą się.
- W przeciwnym wypadku v_i jest widzialny z p.



Analiza złożoności obliczeniowej

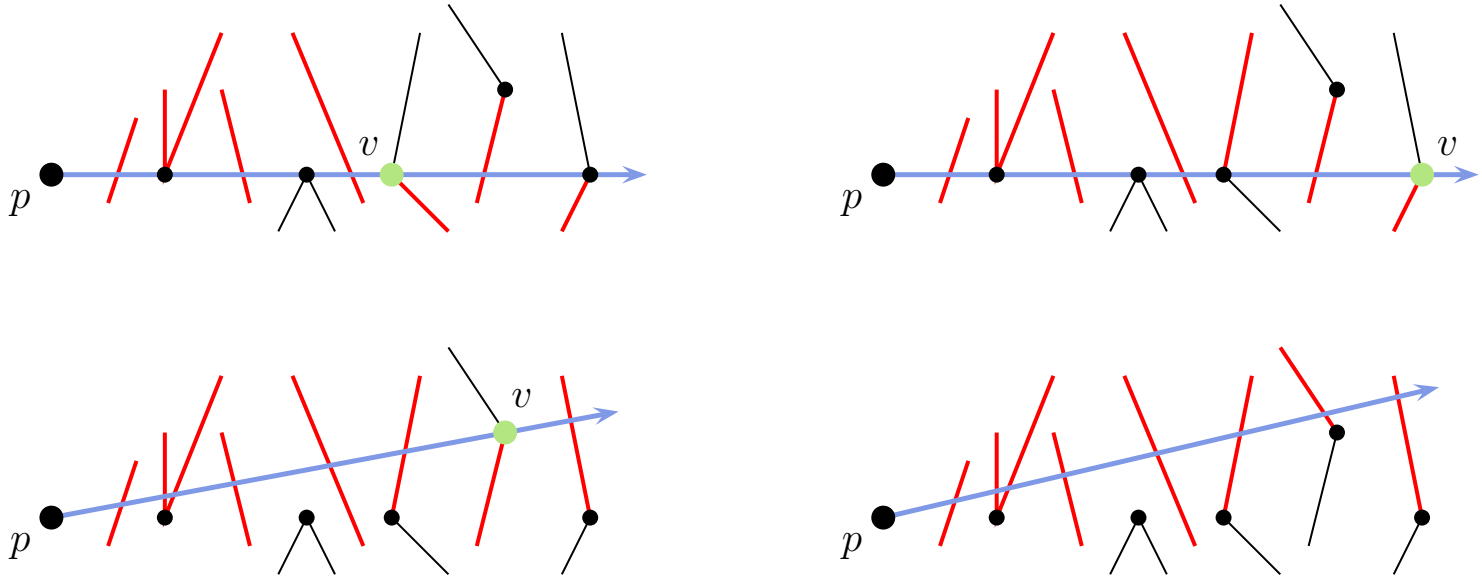
- Złożoność czasowa rzędu $O(n \log n)$.
  - ↳ Sortowanie w porządku biegunowym: $O(n \log n)$.
  - ↳ Sprawdzenie widzialności pomiędzy p i v_i wymaga $O(1)$ operacji na zrównoważonym drzewie wyszukiwań $\mathcal{T}$, które zajmują czas $O(\log n)$ oraz $O(1)$ geometrycznych testów przeprowadzanych w czasie $O(1)$.
- Złożoność pamięciowa: rozmiar drzewa $\mathcal{T}$ rzędu $O(n)$.

Analiza poprawności podejścia



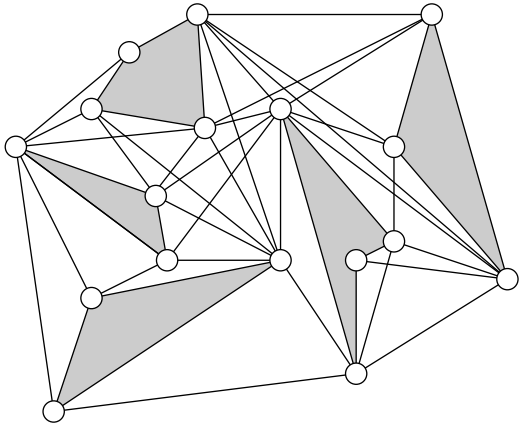
- Poprawność rozstrzygnięcia, czy p widzi v_i, wynika z obserwacji poczynionych przy omawianiu procedury $\textsc{Visible}(p, v_i)$.
- Poprawność drzewa $\mathcal{T}$, tzn. zachowywanie niezmiennika: indukcja.

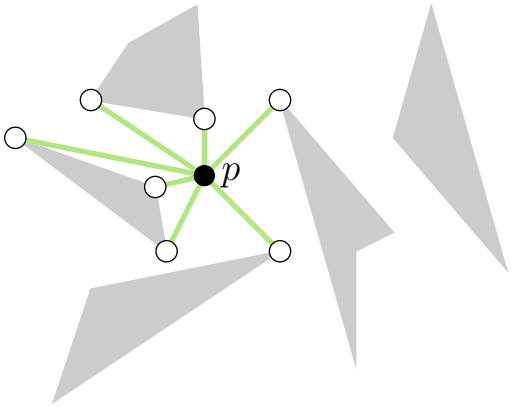
Analiza poprawności podejścia



- Poprawność rozstrzygnięcia, czy p widzi v_i, wynika z obserwacji poczynionych przy omawianiu procedury $\textsc{Visible}(p, v_i)$.
- Poprawność drzewa $\mathcal{T}$, tzn. zachowywanie niezmiennika: indukcja.

Twierdzenie 2.7. (Lee 1978)
Dla zbioru rozłącznych wielokątów na płaszczyźnie oraz punktu p problem wyznaczenia wierzchołków widzialnych z p można rozwiązać w czasie rzędu $O(n \log n)$ i pamięci rzędu $O(n)$, gdzie n jest liczbą wierzchołków wszystkich wielokątów.





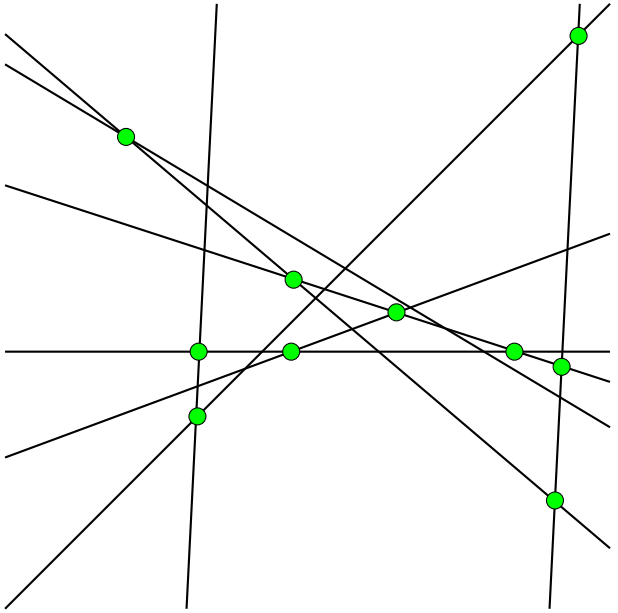
Algorytm VisibilityGraph(S)
Wejście. Zbiór S rozłącznych wielokątów na płaszczyźnie.
Wyjście. Graf widzialności $G_{\text{widz}}(S)$.

1. Inicjuj graf $G = (V, E)$, gdzie V jest zbiorem wszystkich wierzchołków wielokątów z S i $E = \emptyset$.
2. **for each** $v \in V$ **do**
   2.1 $W :=$ VisibleVertices(v, S).
   2.2 **for each** $w \in W$ **do** $E := E \cup \{v, w\}$.
3. **return** G.

Twierdzenie 2.8. (Lee 1978)
Graf widzialności zbioru rozłącznych wielokątów na płaszczyźnie można wyznaczyć w czasie $O(n^2 \log n)$, gdzie n jest liczbą wierzchołków wszystkich wielokątów.

2.7 STRZEŻENIE KOMÓREK

Dla danej prostej kompozycji $\mathcal{A}$ prostych na płaszczyźnie (sieć ulic), jaki jest najmniejszy rozmiar zbioru strażników umieszczonych w punktach przecięć prostych (skrzyżownia), który pozwala na strzeżenie każdej komórki w $\mathcal{A}$? (Bose et al. '13)



Inne problemy (Bose et al. '13)**:**

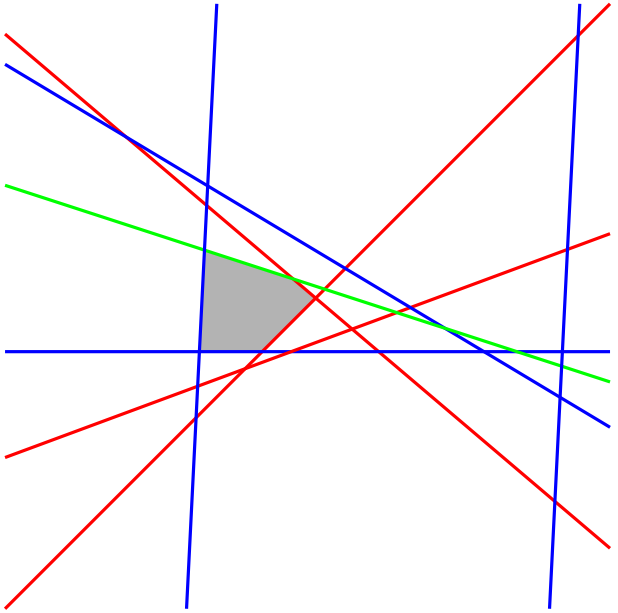
#1. Jaka jest najmniejsza liczba kolorów pozwalająca na pokolorowanie prostych w taki sposób, aby żadna z komórek nie była monochromatyczna?

#2. Jaka jest największa liczba prostych taka, aby żadna z komórek nie była otoczona tylko przez te proste?

#3. Jaka jest najmniejsza liczba wybranych komórek taka, że każda prosta jest incydentna do przynajmniej jednej wybranej komórki?

2.7 STRZEŻENIE KOMÓREK

Dla danej prostej kompozycji $\mathcal{A}$ prostych na płaszczyźnie (sieć ulic), jaki jest najmniejszy rozmiar zbioru strażników umieszczonych w punktach przecięć prostych (skrzyżownia), który pozwala na strzeżenie każdej komórki w $\mathcal{A}$? (Bose et al. '13)



Inne problemy (Bose et al. '13)**:**

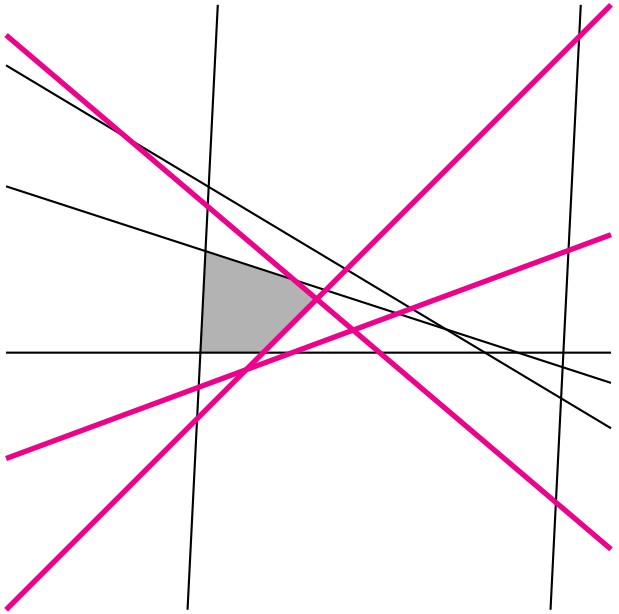
#1. Jaka jest najmniejsza liczba kolorów pozwalająca na pokolorowanie prostych w taki sposób, aby żadna z komórek nie była monochromatyczna?

#2. Jaka jest największa liczba prostych taka, aby żadna z komórek nie była otoczona tylko przez te proste?

#3. Jaka jest najmniejsza liczna wybranych komórek taka, że każda prosta jest incydentna do przynajmniej jednej wybranej komórki?

2.7 STRZEŻENIE KOMÓREK

Dla danej prostej kompozycji $\mathcal{A}$ prostych na płaszczyźnie (sieć ulic), jaki jest najmniejszy rozmiar zbioru strażników umieszczonych w punktach przecięć prostych (skrzyżownia), który pozwala na strzeżenie każdej komórki w $\mathcal{A}$? (Bose et al. '13)



Inne problemy (Bose et al. '13)**:**

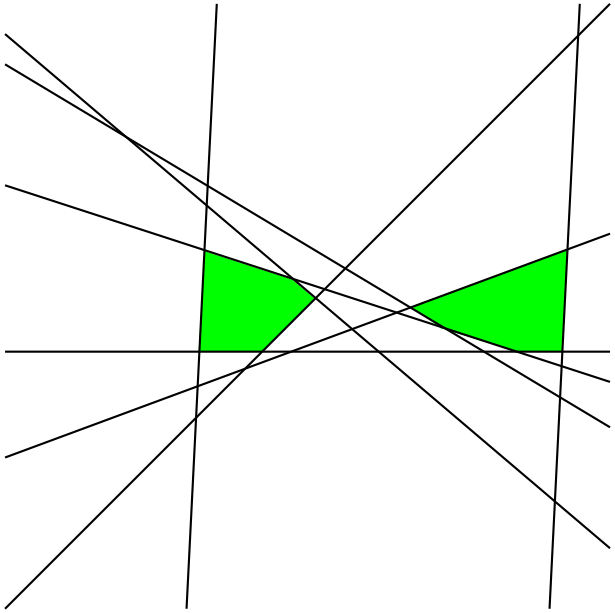
#1. Jaka jest najmniejsza liczba kolorów pozwalająca na pokolorowanie prostych w taki sposób, aby żadna z komórek nie była monochromatyczna?

#2. Jaka jest największa liczba prostych taka, aby żadna z komórek nie była otoczona tylko przez te proste?

#3. Jaka jest najmniejsza liczna wybranych komórek taka, że każda prosta jest incydentna do przynajmniej jednej wybranej komórki?

2.7 STRZEŻENIE KOMÓREK

Dla danej prostej kompozycji $\mathcal{A}$ prostych na płaszczyźnie (sieć ulic), jaki jest najmniejszy rozmiar zbioru strażników umieszczonych w punktach przecięć prostych (skrzyżownia), który pozwala na strzeżenie każdej komórki w $\mathcal{A}$? (Bose et al. '13)



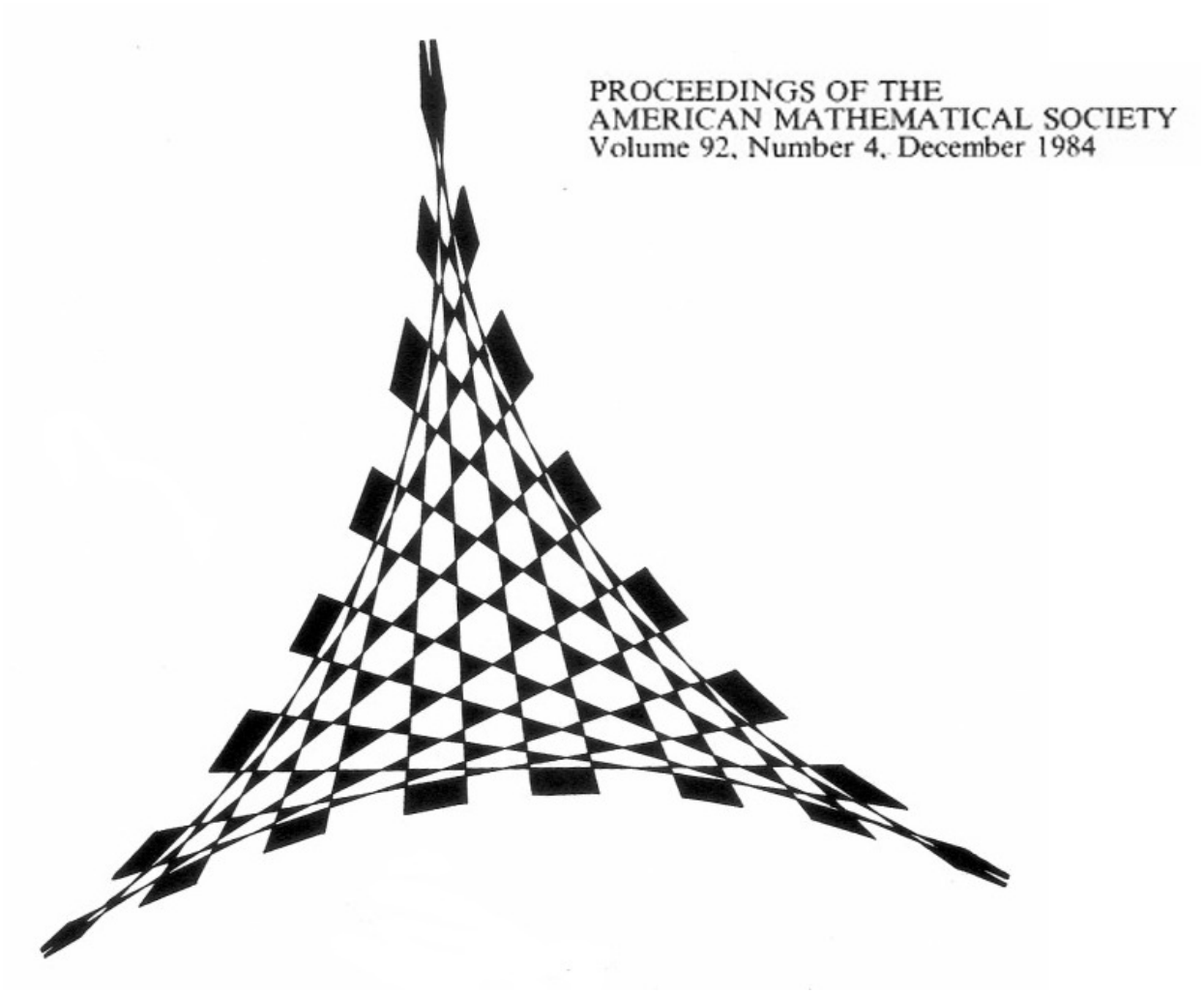
Inne problemy (Bose et al. '13)**:**

#1. Jaka jest najmniejsza liczba kolorów pozwalająca na pokolorowanie prostych w taki sposób, aby żadna z komórek nie była monochromatyczna?

#2. Jaka jest największa liczba prostych taka, aby żadna z komórek nie była otoczona tylko przez te proste?

#3. Jaka jest najmniejsza liczna wybranych komórek taka, że każda prosta jest incydentna do przynajmniej jednej wybranej komórki?

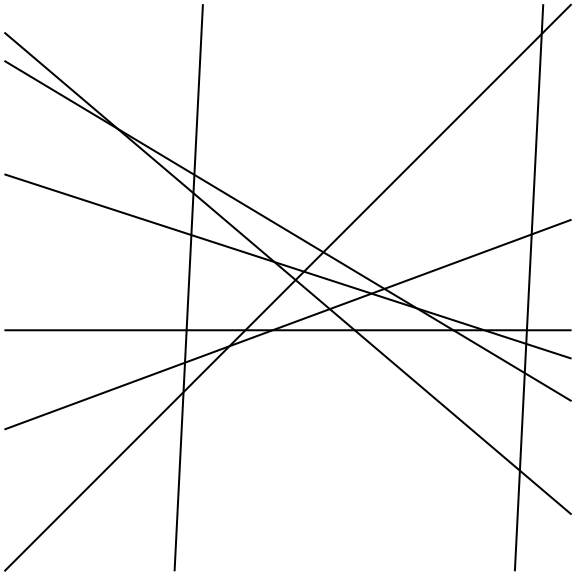
Oszacowanie dolne: $n^2/6$ (Bose et al. '13, ref. Füredi oraz Palásti '84)

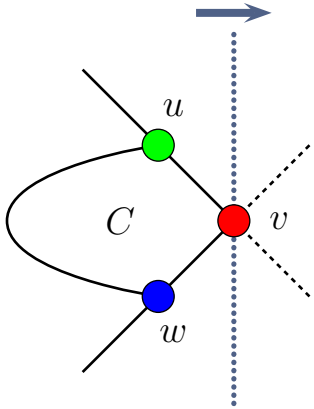


- Kompozycja zawiera przynajmniej $n(n-3)/3$ trójkątów.
- Każdy z punktów przecięcia prostych jest incydentny do co najwyżej dwóch takich trójkątów.

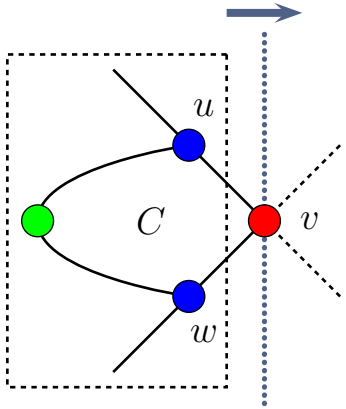
Oszacowanie górne: $n^2/6 + n$ (Bose et al. '13)

- Legalne 3-kolorowanie wierzchołków (punktów przecięć) w taki sposób, że każda ograniczona komórka jest „otoczona" przez trzy kolory.
- Sposób kolorowania: **technika zamiatania**.



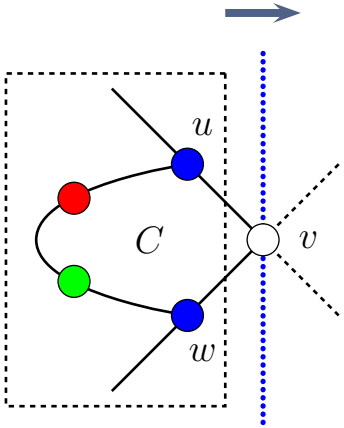


v otrzymuje trzeci kolor



v otrzymuje trzeci kolor

dwa kolory już użyte

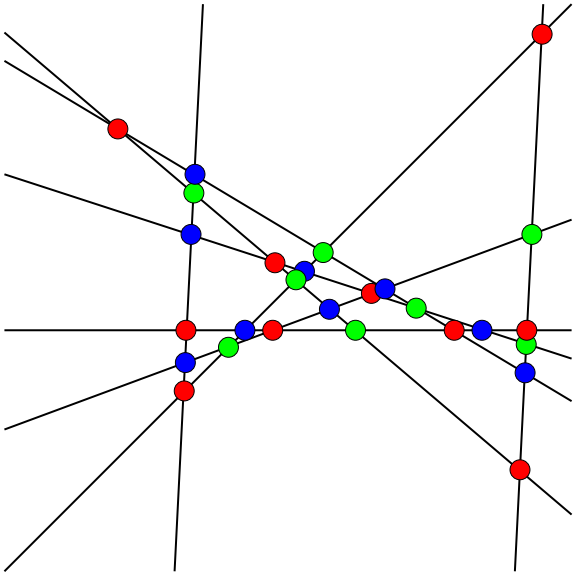


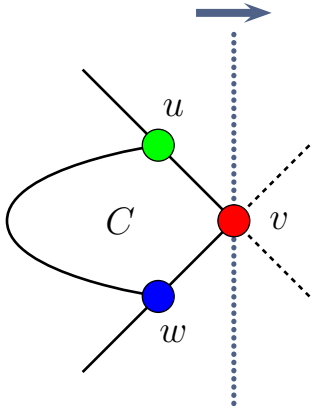
v otrzymuje dowolny kolor (jeden z dwóch)

trzy kolory już użyte

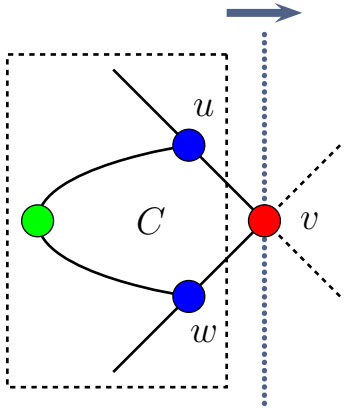
Oszacowanie górne: $n^2/6 + n$ (Bose et al. '13)

- Legalne 3-kolorowanie wierzchołków (punktów przecięć) w taki sposób, że każda ograniczona komórka jest „otoczona" przez trzy kolory.
- Sposób kolorowania: **technika zamiatania**.



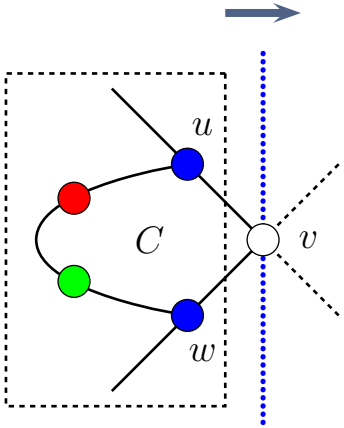


v otrzymuje trzeci kolor



v otrzymuje trzeci kolor

dwa kolory już użyte

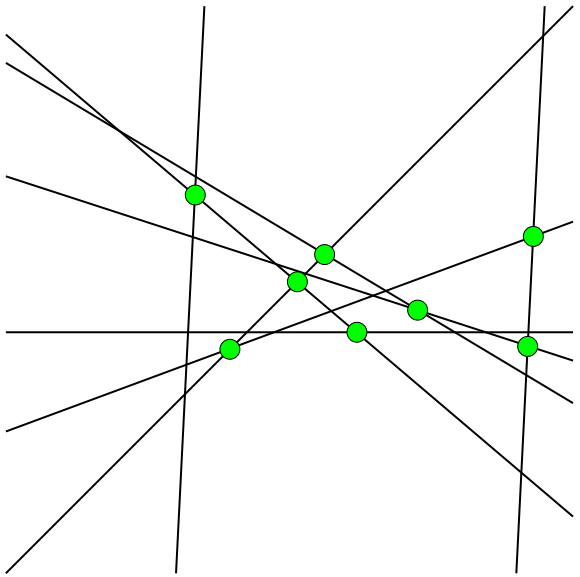


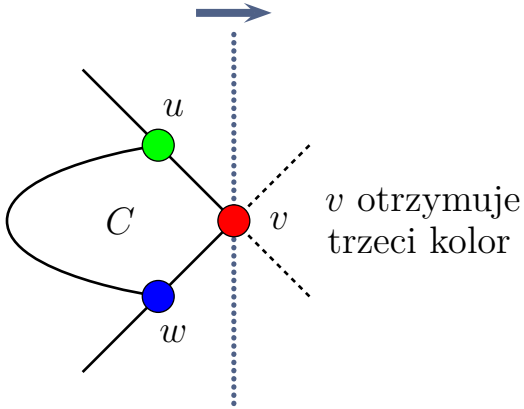
v otrzymuje dowolny kolor (jeden z dwóch)

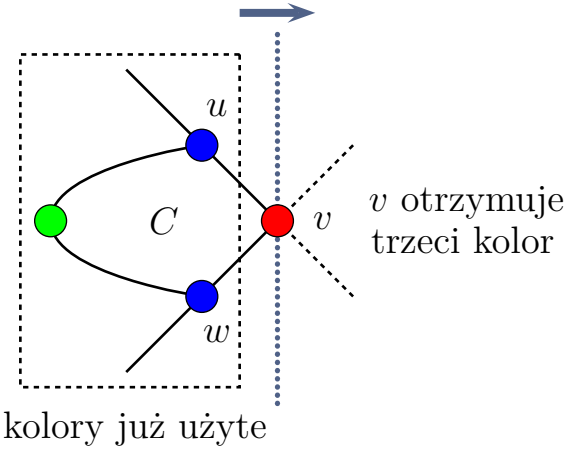
trzy kolory już użyte

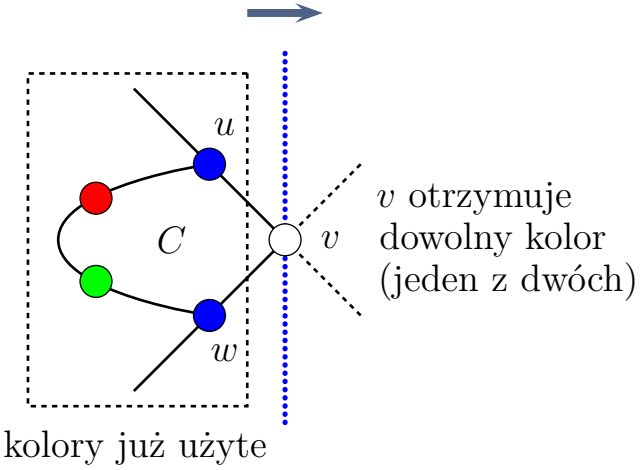
Oszacowanie górne: $n^2/6 + n$ (Bose et al. '13)

- Legalne 3-kolorowanie wierzchołków (punktów przecięć) w taki sposób, że każda ograniczona komórka jest „otoczona" przez trzy kolory.
- Sposób kolorowania: **technika zamiatania**.
- Wierzchołki w najrzadziej występującym kolorze (jest ich co najwyżej $n^2/6$) strzegą wszystkie ograniczone komórki.
- Nieograniczone komórki (jest ich $2n$) można strzec dodatkowymi n strażnikami (lub mniej).









Oszacowanie górne: $n^2/6 + n$ (Bose et al. '13)

- Legalne 3-kolorowanie wierzchołków (punktów przecięć) w taki sposób, że każda ograniczona komórka jest „otoczona" przez trzy kolory.
- Sposób kolorowania: **technika zamiatania**.
- Wierzchołki w najrzadziej występującym kolorze (jest ich co najwyżej $n^2/6$) strzegą wszystkie ograniczone komórki.
- Nieograniczone komórki (jest ich $2n$) można strzec dodatkowymi n strażnikami (lub mniej).