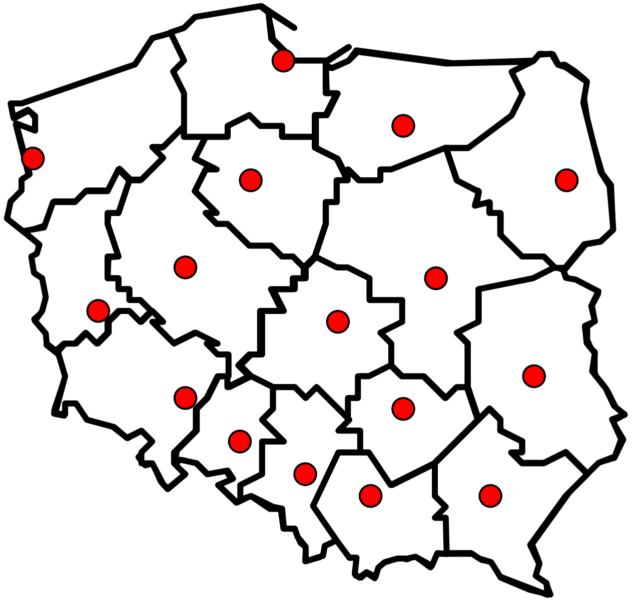
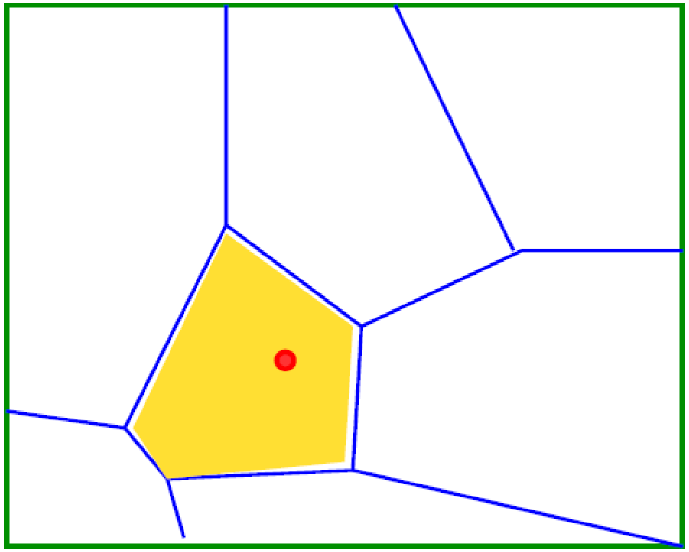
Lokalizacja punktu





Lokalizacja punktu na płaszczyźnie

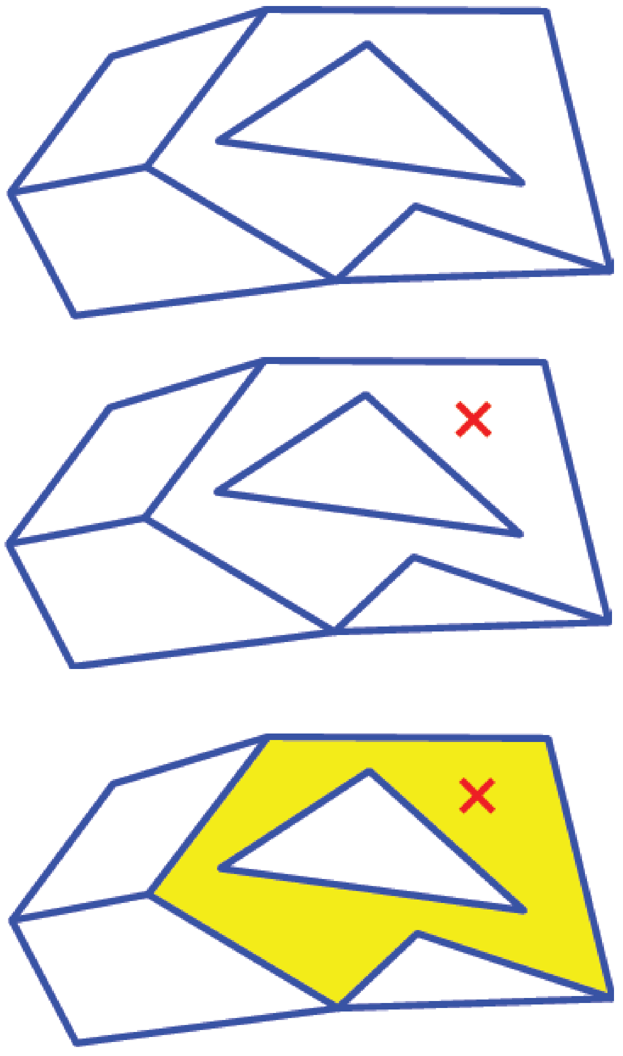
Ogólna definicja problemu:

Dane: *poligonowy podziału płaszczyzny (podział planarny) S*

Należy go przetworzyć (zapisując wyniki w odpowiedniej strukturze danych) tak, aby umożliwić efektywne osiągnięcie celu.

Cel: *odszukanie wielokąta (ściany) zawierającego zadany punkt.*

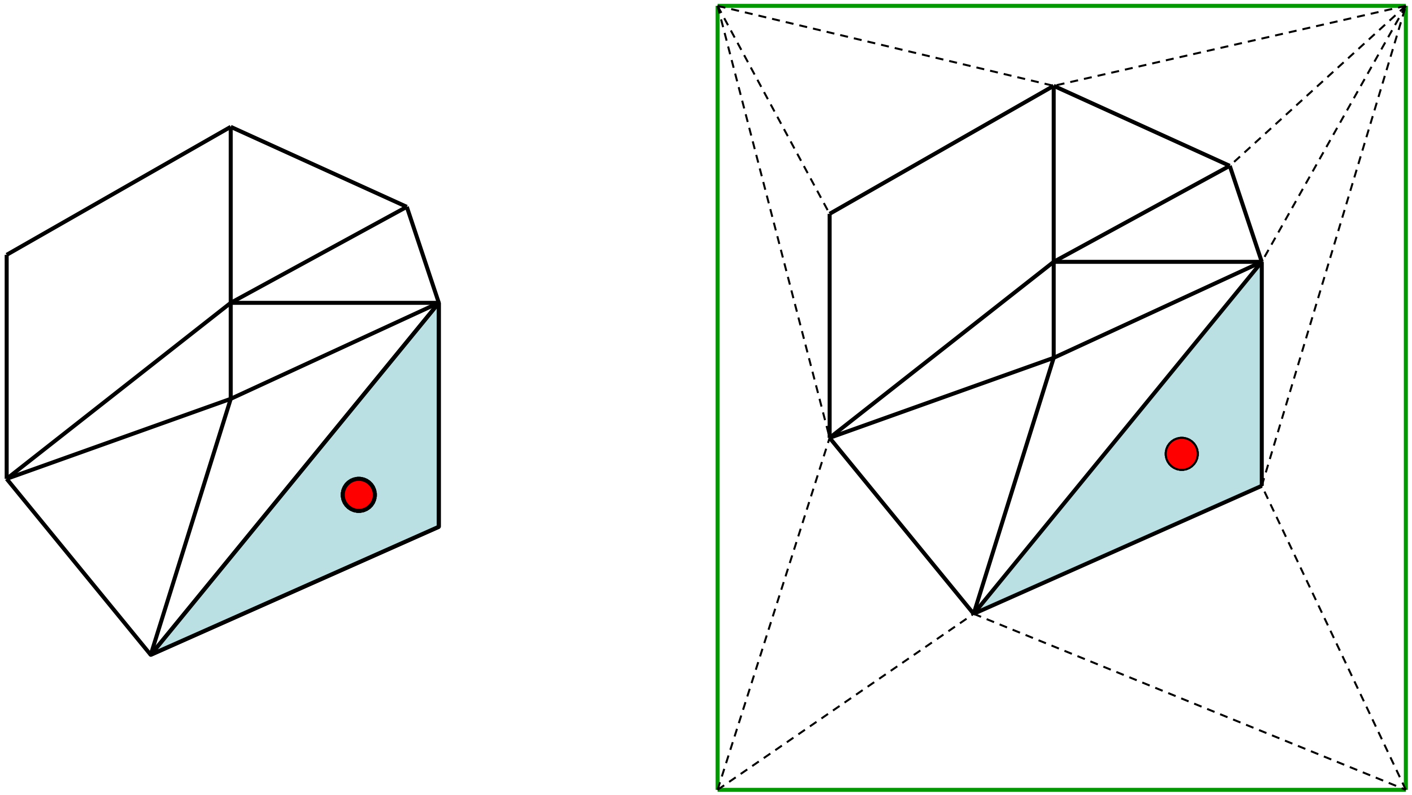
czas odpowiedzi na zapytanie o punkt będzie zależeć od rozmiaru struktury i wyboru metody jej konstrukcji



Lokalizacja punktu na płaszczyźnie

- Założenia
  - „rozsądna” reprezentacja podziału planarnego
    (np. w postaci grafu krawędzi i wierzchołków)
  - zadany punkt leży wewnątrz jednej ze ścian
    (łatwo rozszerzyć algorytm o specjalne przypadki)

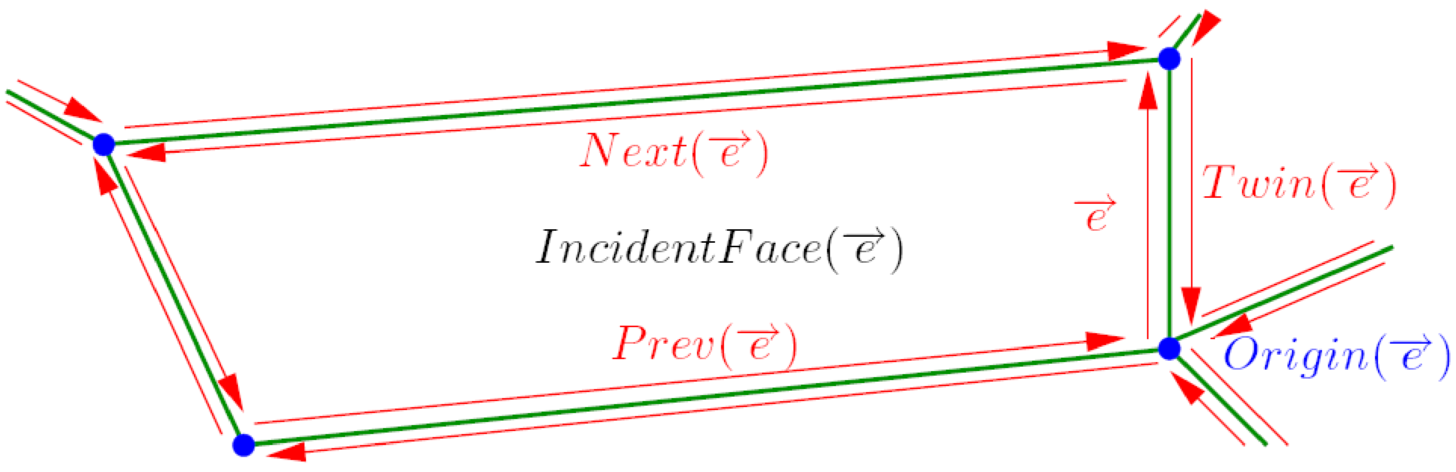
- Oczekiwania
  - złożoność pamięciowa: $O(n)$
  - złożoność czasowa lokalizacji: $O(\log n)$
  - złożoność czasowa konstrukcji: $O(n \log n)$



Ograniczony podział można umieścić
w odpowiednio dużym wielokącie
wypukłym o małej liczbie boków.

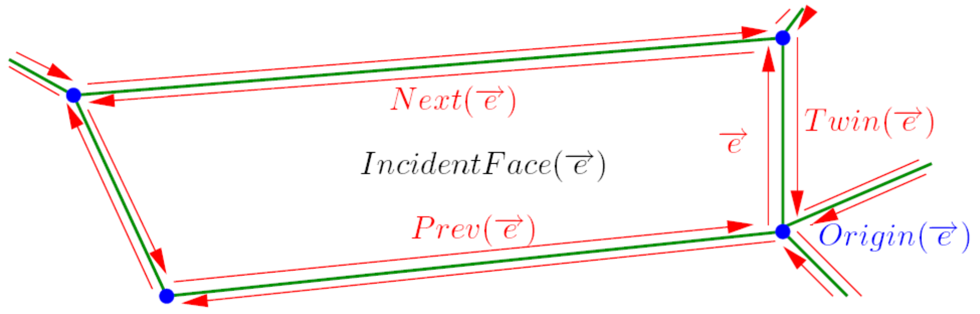
Podział planarny
***PSLG –(planar straight line graph)*,**

Struktura danych: **podwójnie łączona lista krawędzi**



Każda krawędź jest zastępowana poprzez dwie półkrawędzie (bliźnięta)

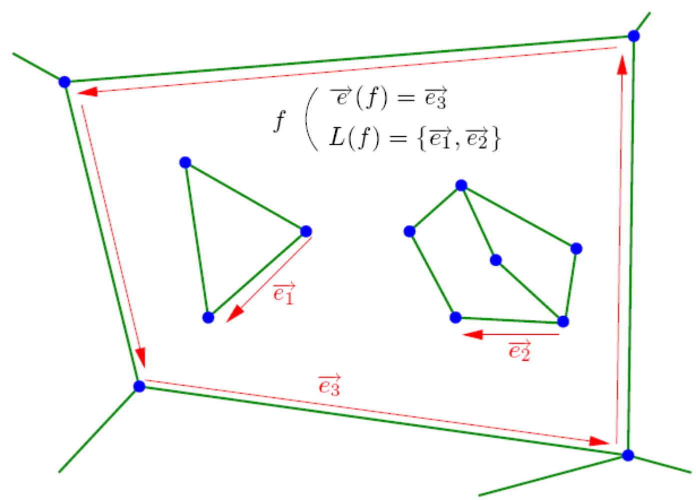
Podział planarny
podwójnie łączona lista krawędzi



- wierzchołek
  - ✓ *współrzędne*
  - ✓ *incydentna półkrawędź*

- półkrawędź
  - ✓ *3 krawędzie* *Twin(e), Next(e), Prev(e)*
  - ✓ *wierzchołek* *Origin(e)*
  - ✓ *ściana* *IncidentFace(e)*

- ściana
  - ✓ *półkrawędź z brzegu obszaru*
  - ✓ *półkrawędź z każdej ściany wewnątrz*



Lokalizacja punktu na płaszczyźnie

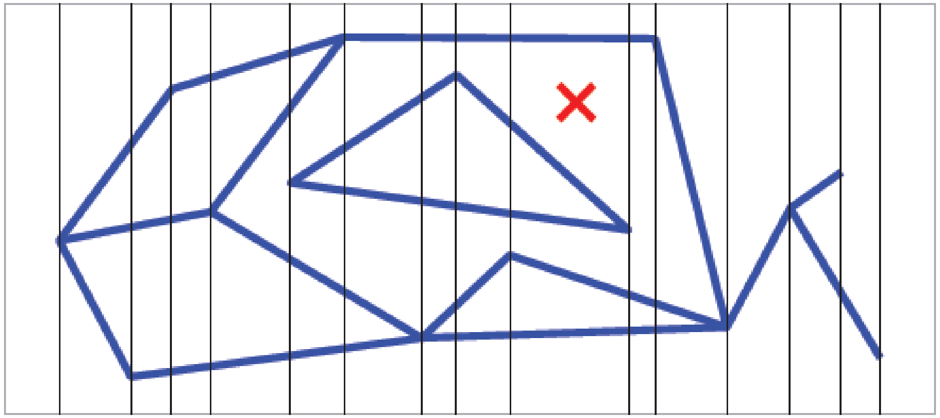
Metody rozwiązania problemu:

1. Metoda warstwowa
2. Metoda trapezowa
3. Metoda doskonalenia triangulacji
4. Metoda separatorów

Metoda warstwowa

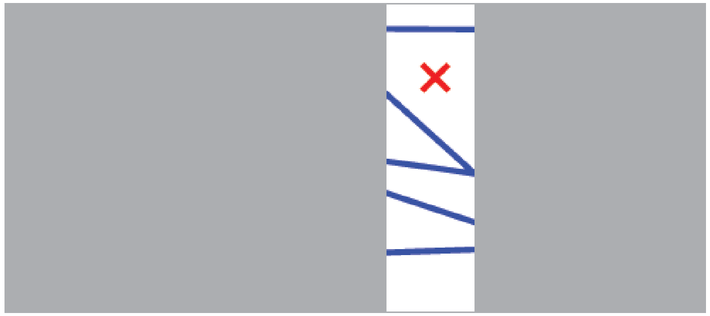
Konstrukcja struktury:

Podzielić przestrzeń na warstwy równoległymi prostymi przechodzącymi przez wierzchołki podziału



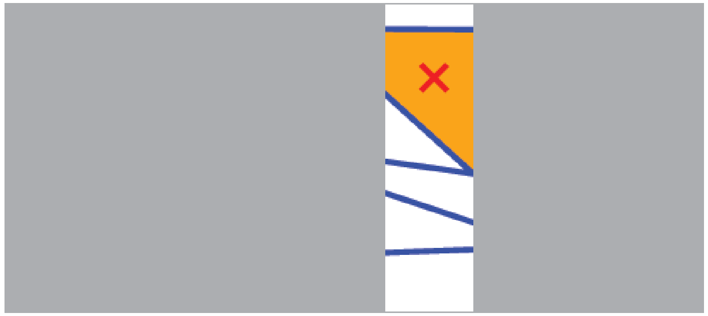
Metoda warstwowa

Lokalizacja punktu:

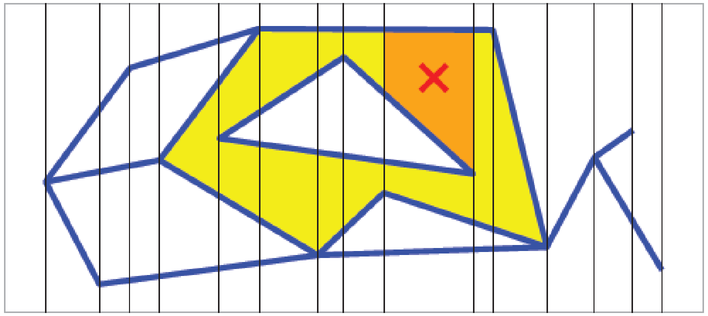


Stosując przeszukiwanie binarne znaleźć:

- warstwę zawierającą dany punkt,



- odpowiednią część warstwy zawierającą dany punkt.



Określić obszar odpowiadający znalezionej części warstwy.

Metoda warstwowa

- złożoność czasowa wyszukiwania $O(\log n)$
- złożoność pamięciowa – pesymistycznie $O(n^2)$
- złożoność konstrukcji struktury $O(n^2)$

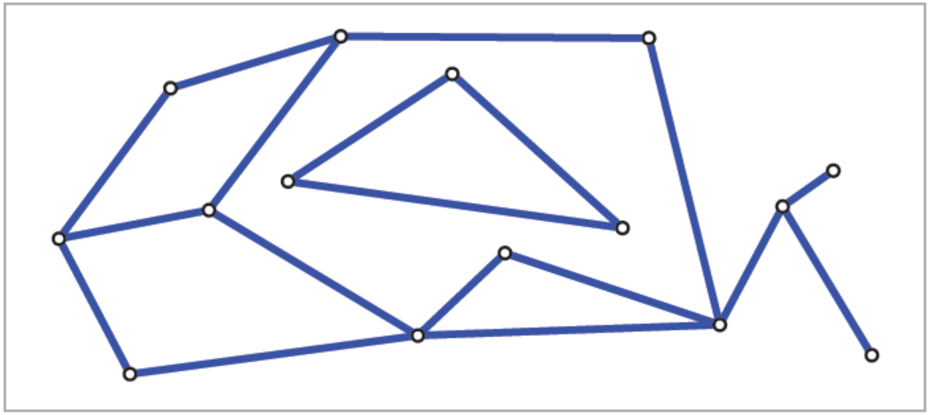
Zatem cel:

poszukać innego rozdrobnienia S, które:

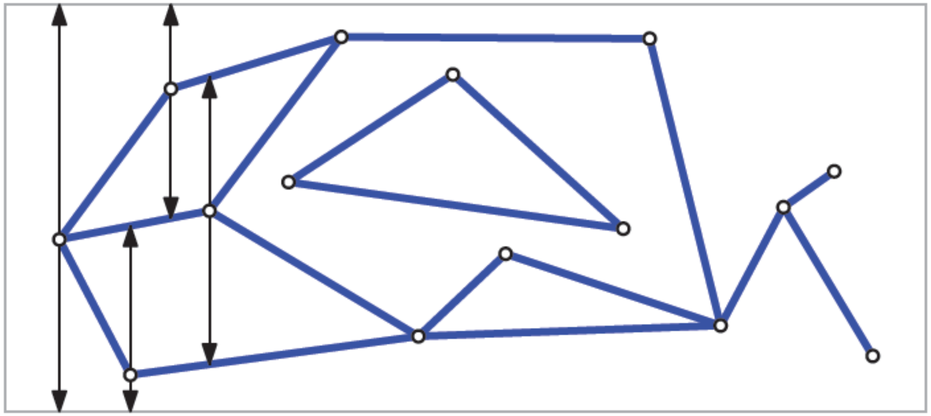
- łatwiej wykonuje zapytanie o położenie punktu
- ma złożoność pamięciową niewiele większą od początkowego podziału S

Mapa trapezowa

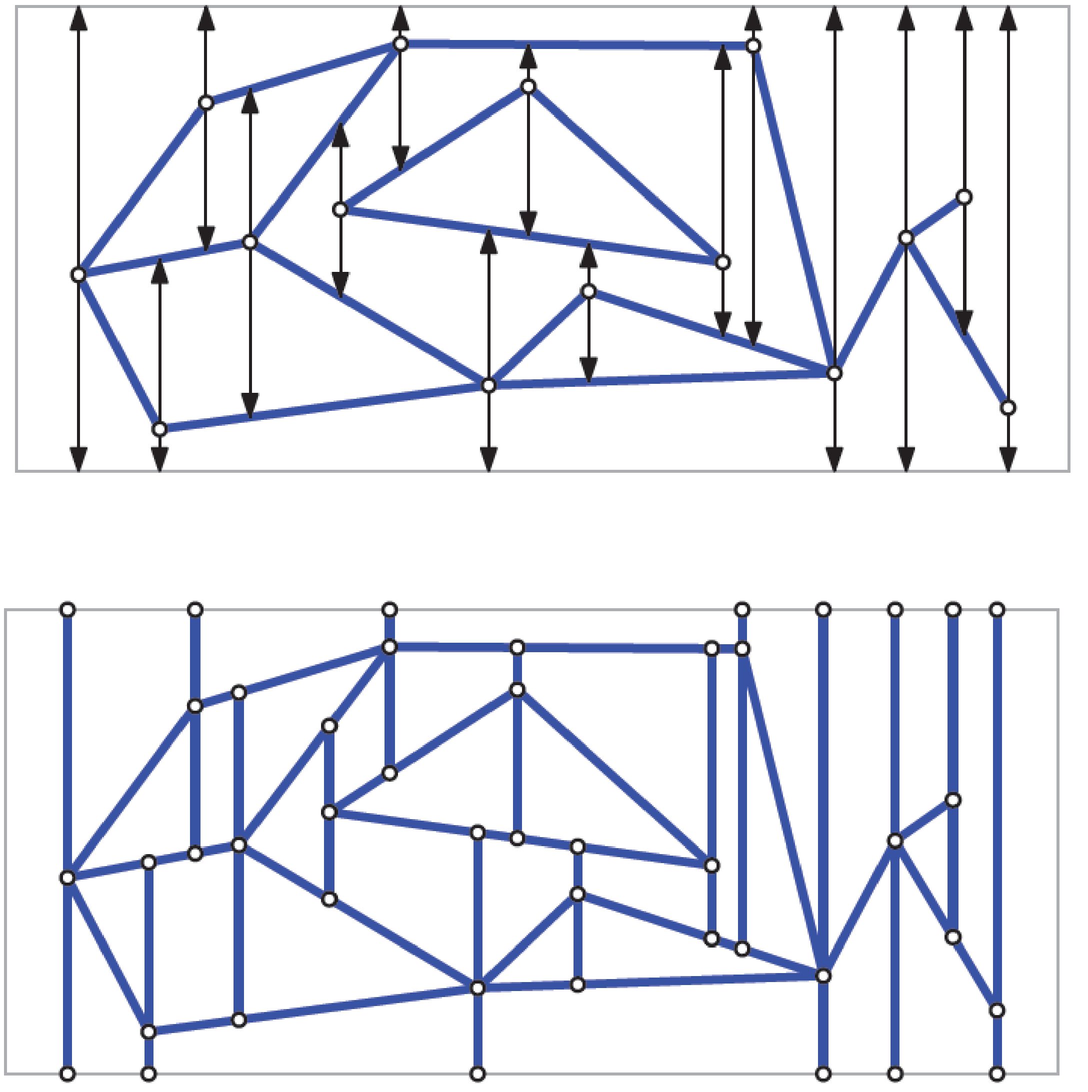
Mapa trapezowa T(*S*) jest podziałem *S* na wielokąty wypukłe (*trapezy lub trójkąty*) otrzymanym przez poprowadzenie dwóch rozszerzeń (odcinków) pionowych z każdego końca odcinka w *S*.



Rozszerzenia kończą się, gdy napotkają inny odcinek *S* lub brzeg prostokąta.



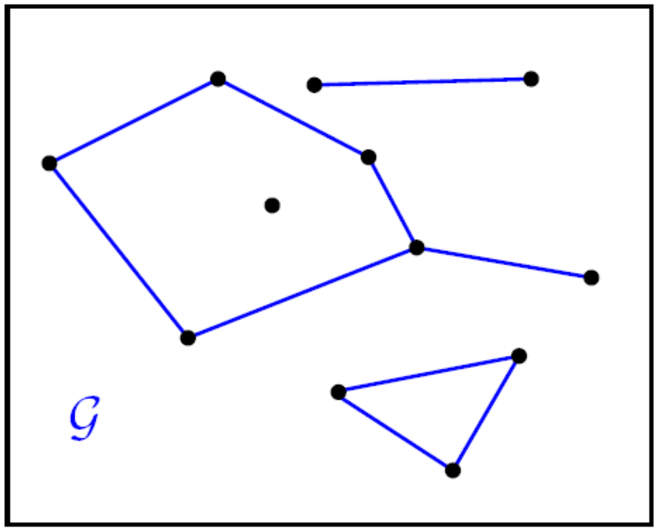
Mapa trapezowa



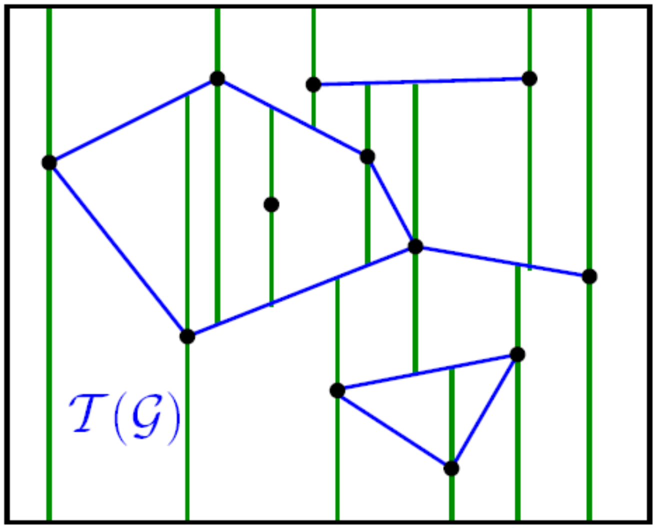
Elementy struktury – trapezy lub trójkąty

Konstrukcja mapy – algorytm zamiatania

Algorytm nie jest zbyt użyteczny dla problemu lokalizacji punktu



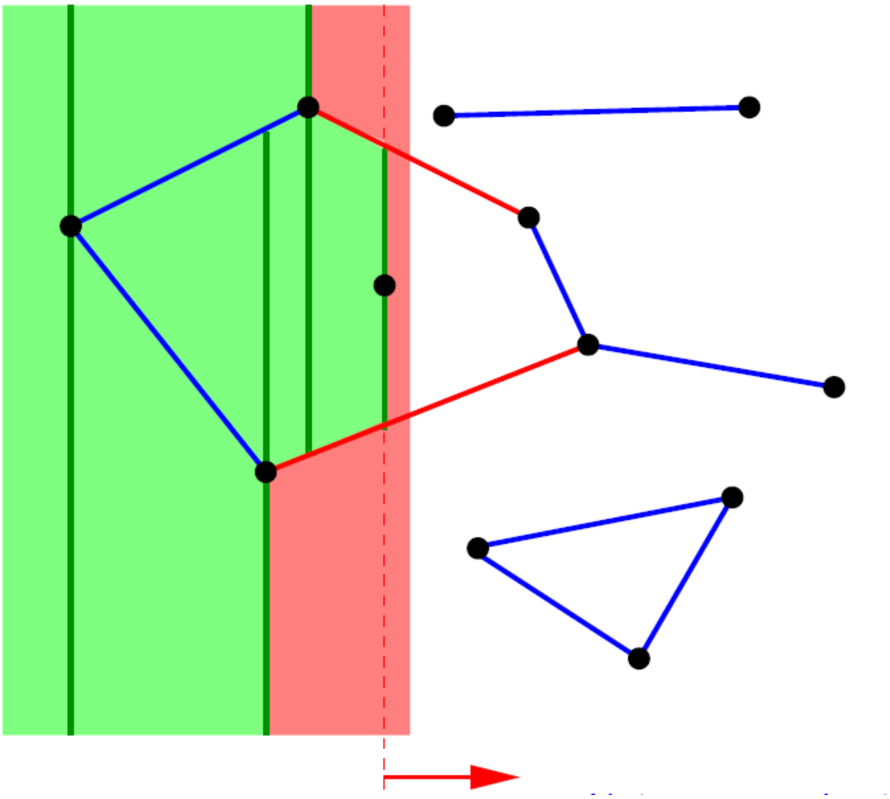
Struktura wyjściowa *G*



Mapa trapezowa

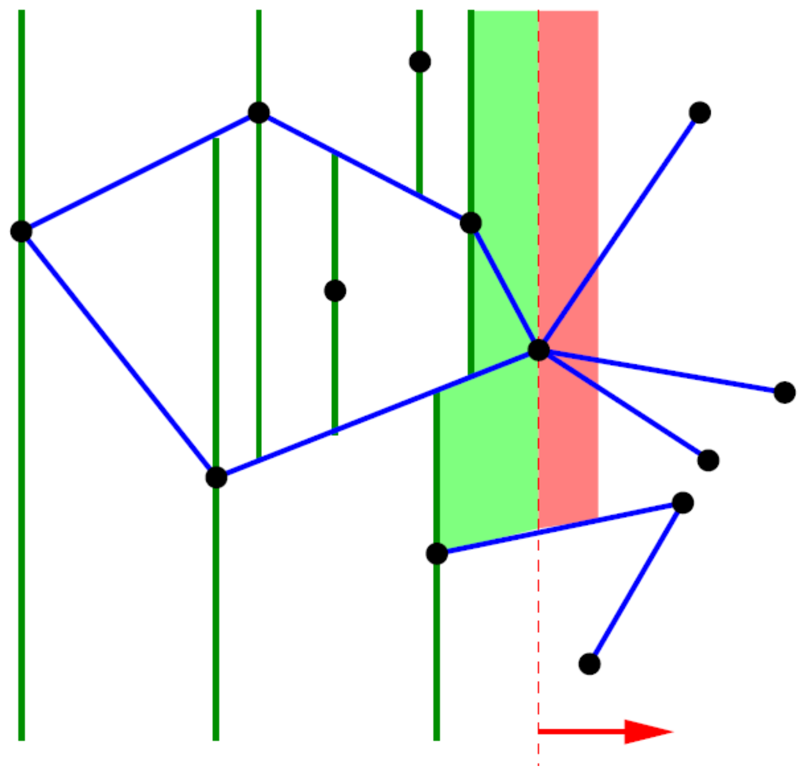
- Sortujemy wierzchołki *G* względem współrzędnej *x*
- Zdarzenie – miotła osiąga wierzchołek *G*

Konstrukcja mapy – algorytm zamiatania



znane trapezy　　aktywne trapezy　　nieprzetworzone

Konstrukcja mapy – algorytm zamiatania



Trzy trapezy zamykane

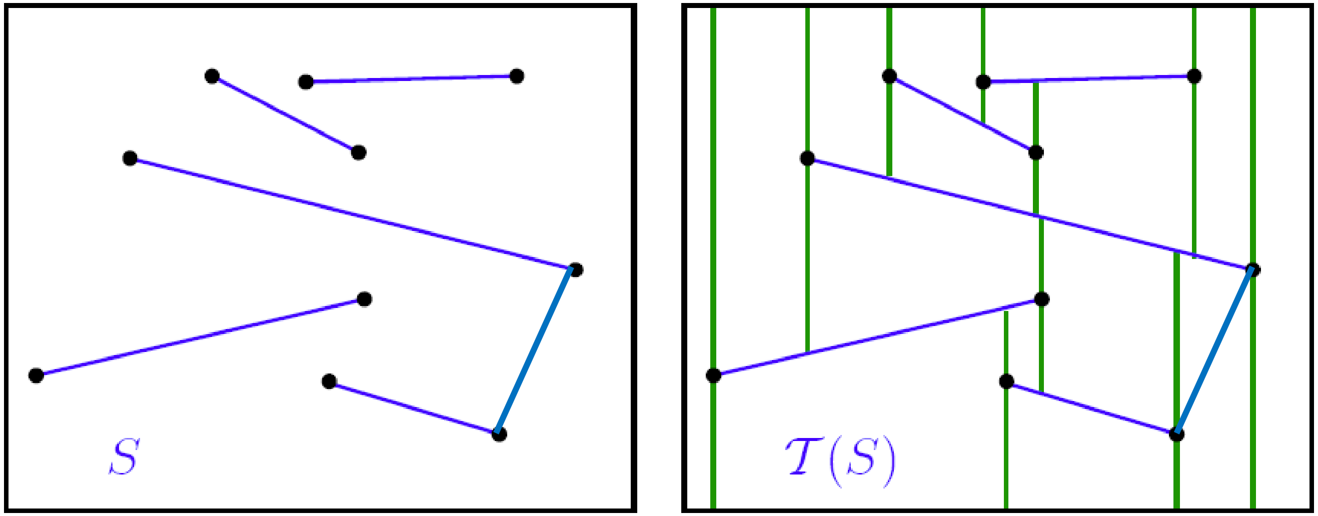
Cztery nowe aktywne trapezy tworzone

Dla zdarzenia – zamykamy pewne aktywne trapezy i tworzymy nowe

Złożoność czasowa konstrukcji $O(n \log n)$

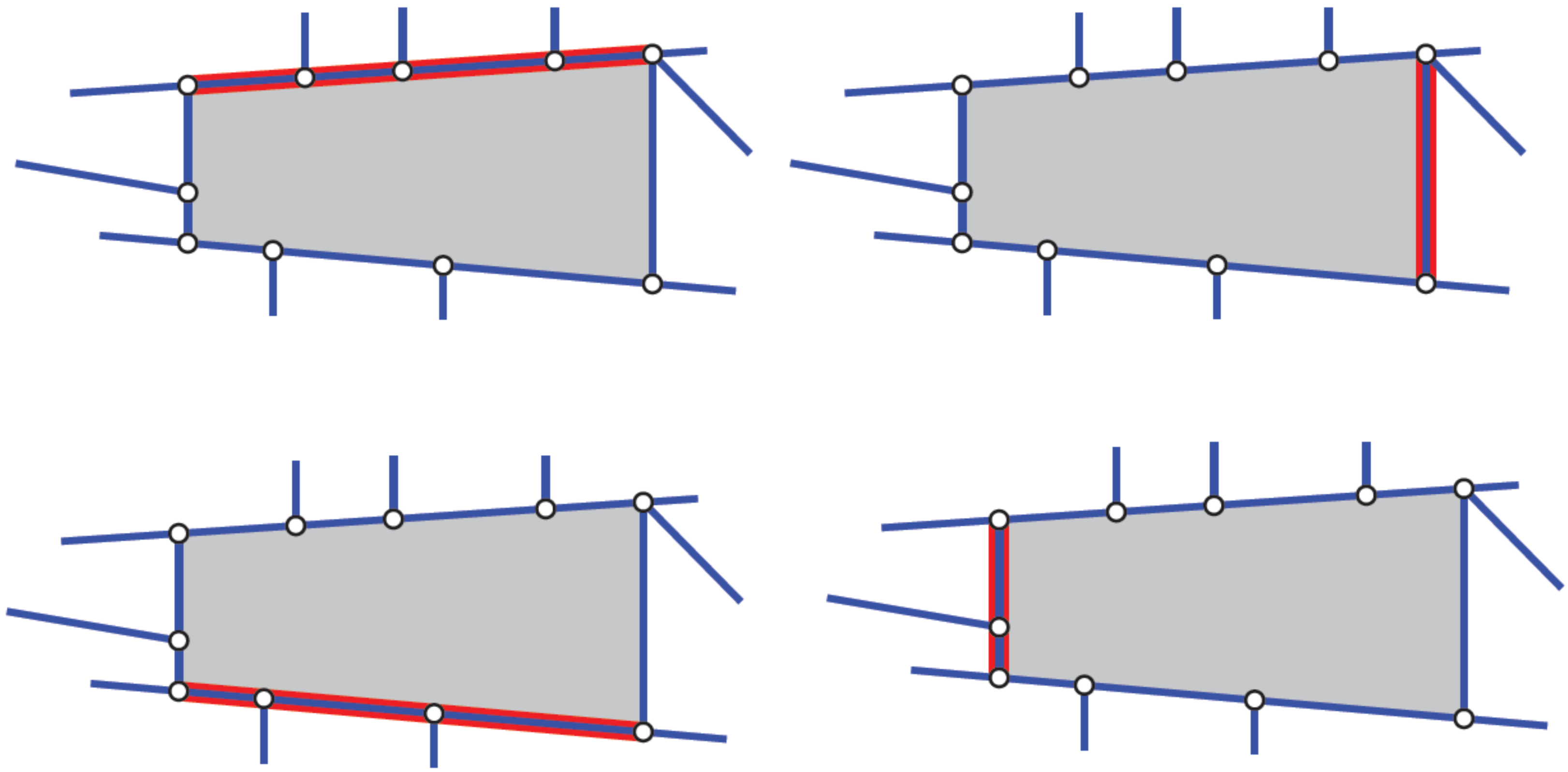
Mapa trapezowa

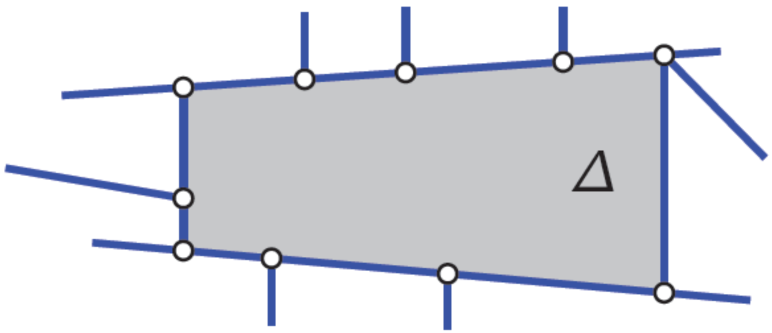
- Reprezentacja w postaci zbioru odcinków $S=\{s_1,s_2,\ldots,s_n\}$
  - odcinki nie przecinają się, poza ewentualnie wierzchołkami

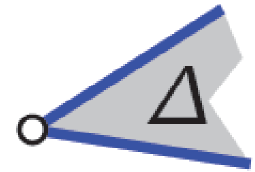


- Dla uproszczenia zakładamy ***położenie ogólne***:
  - żaden odcinek nie jest pionowy
  - wierzchołki żadnych dwóch odcinków nie mają takiej samej współrzędnej x (poza końcami połączonych odcinków)

Bokiem ściany Δ w T(S) jest odcinek o maksymalnej długości zawarty w brzegu ściany







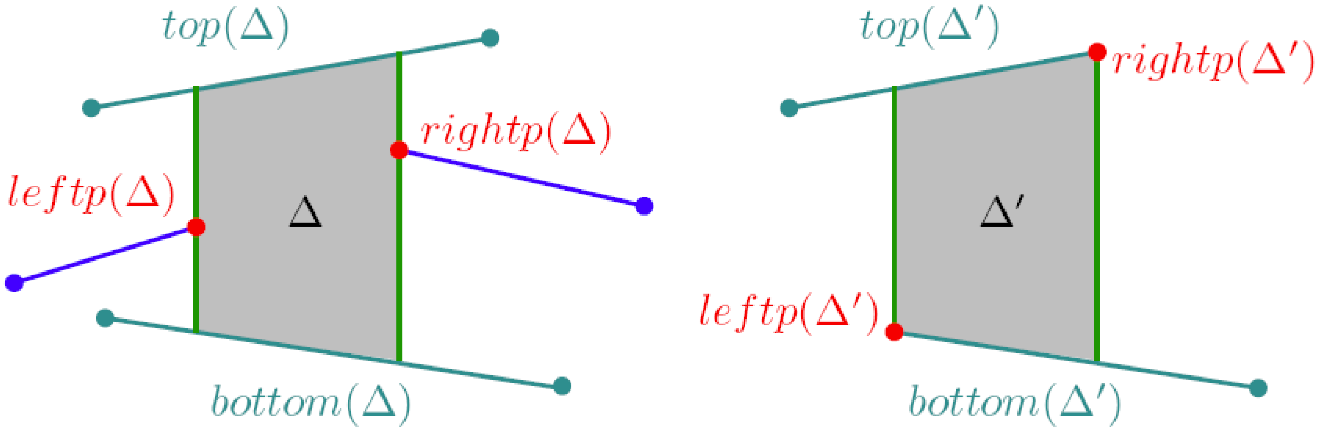
Każdy element mapy (*ściana, trapez*) posiada

- jeden lub dwa boki pionowe
- dokładnie dwa boki „*poziome*” (niepionowe)

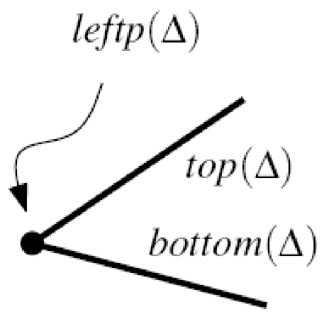
Mapa trapezowa T(S) zbioru S o n odcinkach
w położeniu ogólnym zawiera

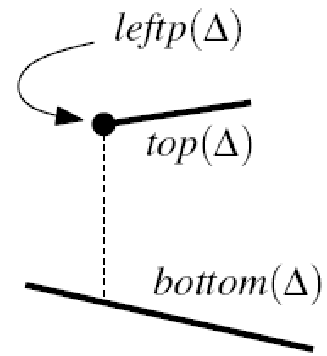
co najwyżej $6n+4$ wierzchołków
i co najwyżej $3n+1$ trapezów.

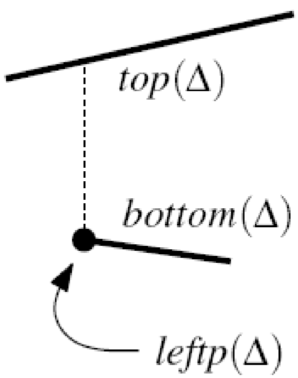
- Wszystkie boczne krawędzie trapezów są pionowe
- Każdy trapez jest *definiowany* przez cztery elementy podziału
  - odcinek dolny: bottom(Δ)
  - odcinek górny: top(Δ)
  - wierzchołek lewy: leftp(Δ)
  - wierzchołek prawy: rightp(Δ)

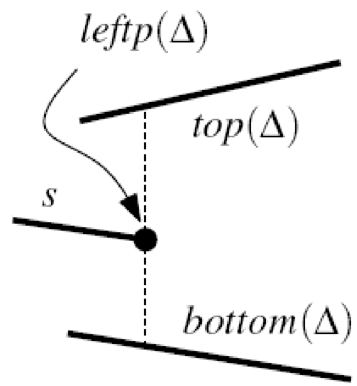


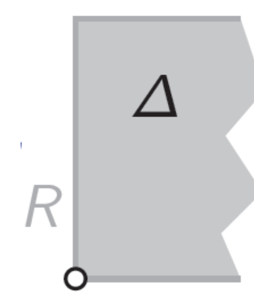
Przypadki dla lewego boku:



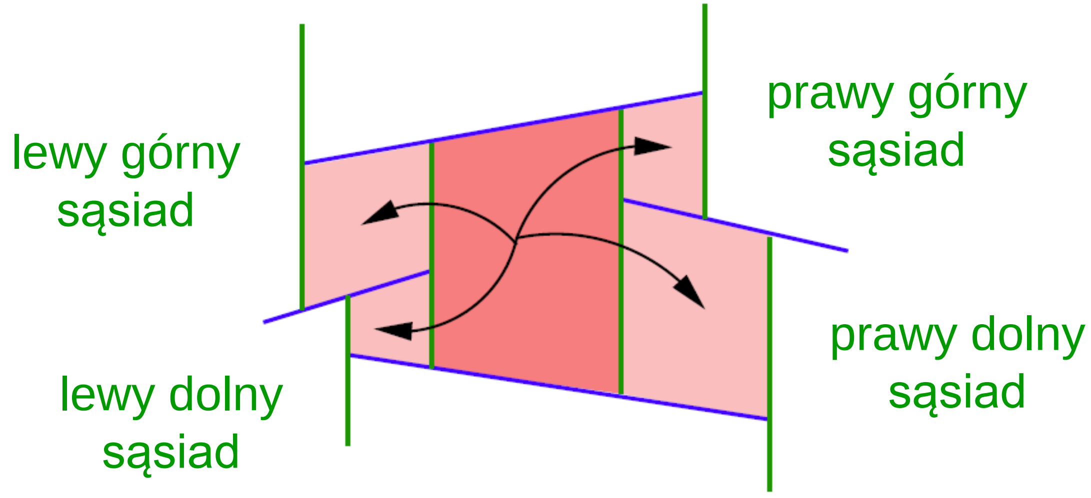




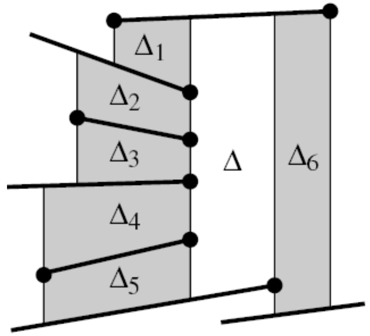




- Sąsiedztwo
  - dwa trapezy są sąsiadami wtedy i tylko wtedy, gdy mają wspólną krawędź pionową
  - każdy trapez ma co najwyżej czterech sąsiadów



Jeżeli zbiór nie jest w położeniu ogólnym, to liczba sąsiadów może być większa:



Struktura danych

– podwójnie łączona lista krawędzi

– albo prościej – tylko powiązania sąsiedzkie

- wierzchołki odcinków w S są reprezentowane przez ich współrzędne
- odcinki w S są reprezentowane przez dwa wierzchołki
- każdy trapez Δ mapy T(S) ma wskaźniki do
  - bottom(Δ), top(Δ)
  - leftp(Δ), rightp(Δ)
  - (co najwyżej) czterech sąsiadów

Randomizowany algorytm przyrostowy konstrukcji T(*S*)

Dane wejściowe:

zbiór odcinków $S=\{s_1,s_2,\ldots,s_n\}$ w położeniu ogólnym

Wynik:

mapa trapezowa T(*S*) i struktura przeszukiwań D dla T(*S*)

Wstęp

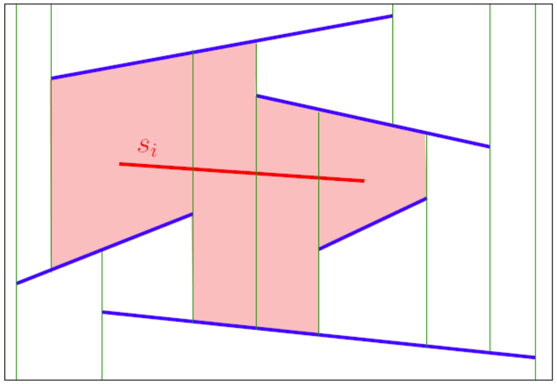
- wyznaczenie losowej permutacji $(s_1,s_2,\ldots,s_n)$ dla zbioru odcinków *S*
- inicjalizacja struktury danych dla prostokąta zewnętrznego

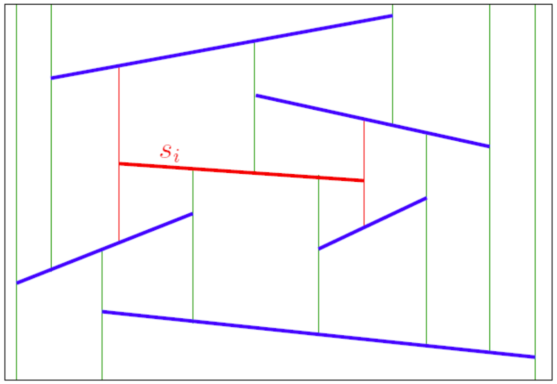
Randomizowany algorytm przyrostowy konstrukcji T(S)

Na etapie *i* – wstawienia odcinka s_i:

utworzone T_{i-1} i D_{i-1}

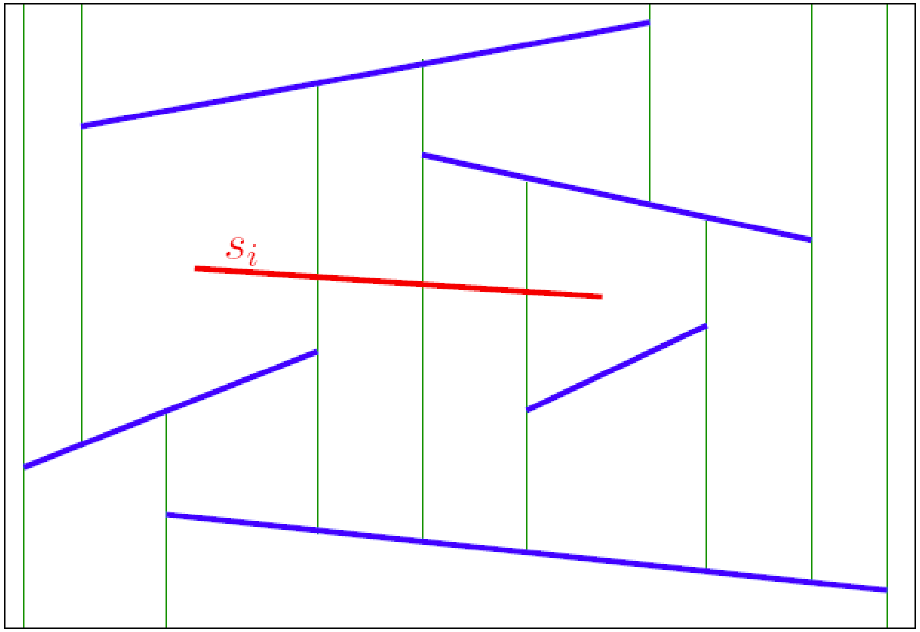
- Znajdź zbiór $\Delta_0, \Delta_1, \ldots, \Delta_k$ trapezów przeciętych przez s_i
- Usuń $\Delta_0, \Delta_1, \ldots, \Delta_k$ i zastąp je przez nowe trapezy
- Uaktualnij T_i oraz D_i .





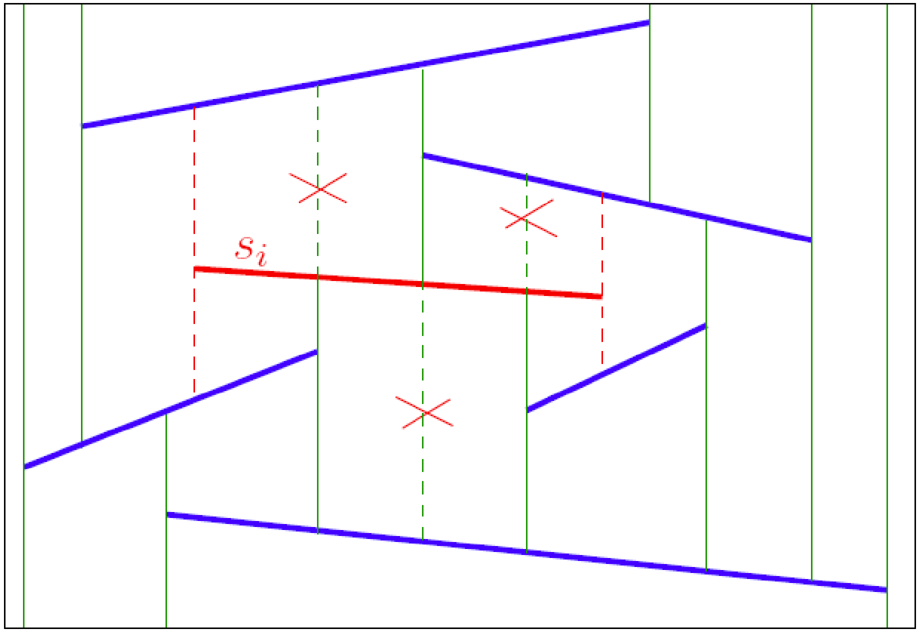
Konstrukcja T(*S*)

- Wstawienie s_i
  - s_i może przecinać kilka trapezów z $T(S_{i-1})$



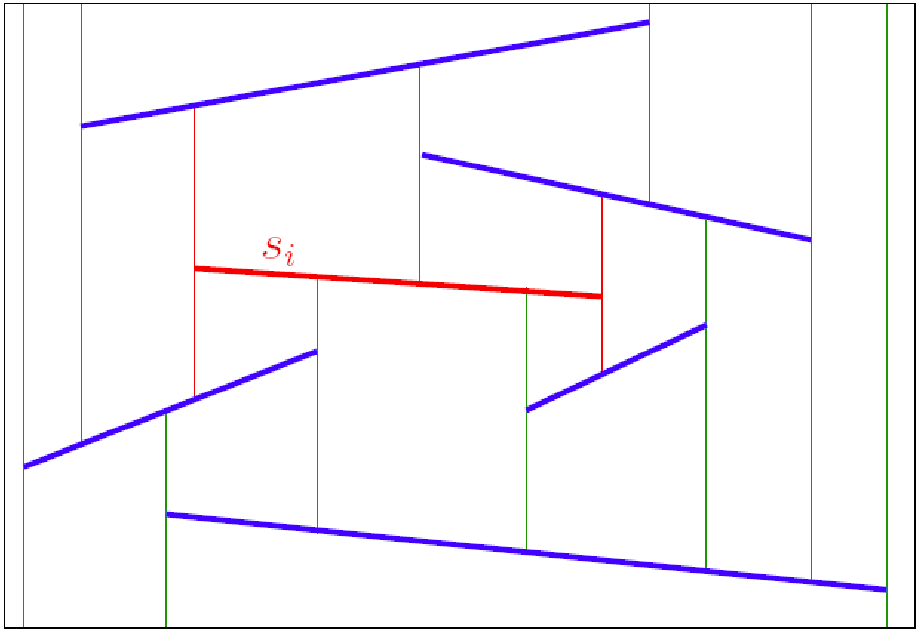
Konstrukcja T(*S*)

- Wstawienie s_i
  - – każdy trapez może zostać podzielony na maksymalnie cztery trapezy



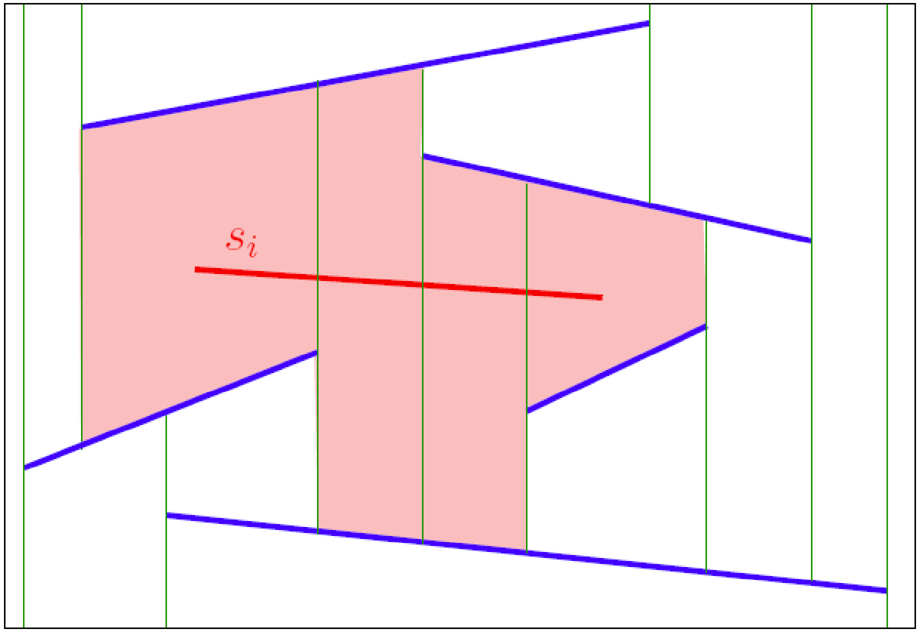
Konstrukcja T(*S*)

- Wstawienie s_i
  - niektóre trapezy mogą zostać połączone



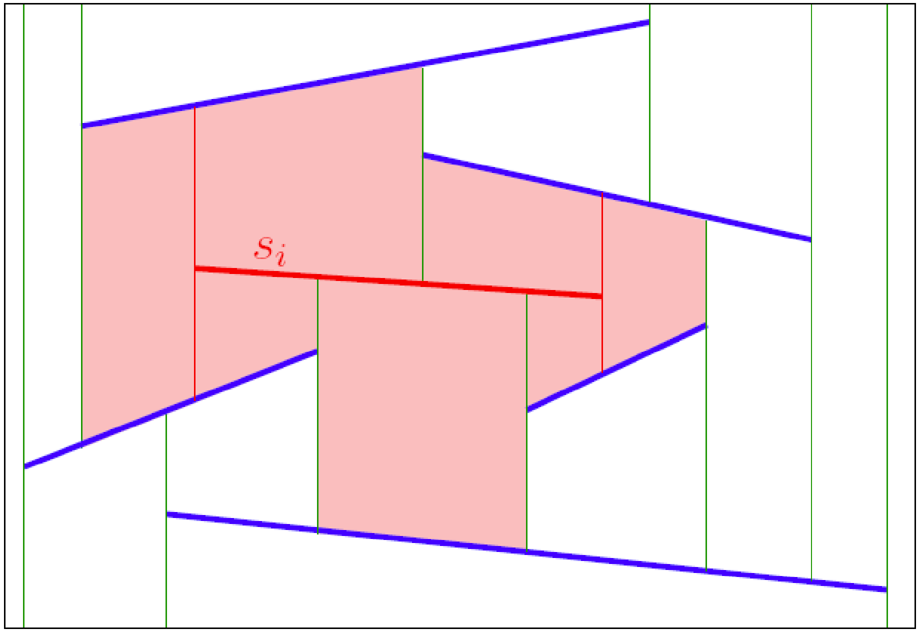
Konstrukcja T(*S*)

- Strefa dla s_i
  - strefę dla s_i w $T(S_{i-1})$ tworzą wszystkie trapezy przecinające s_i
    - jest to też suma wszystkich trapezów, które zostaną usunięte

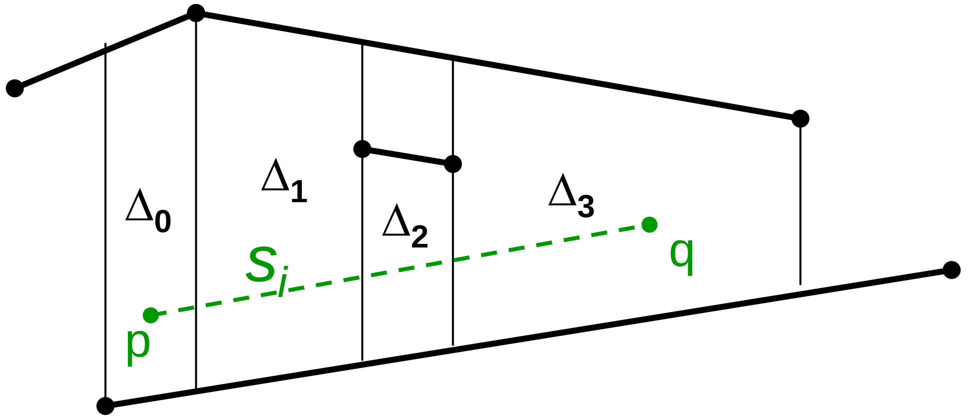


Konstrukcja T(*S*)

- Strefa dla s_i
  - strefę dla s_i w $T(S_i)$ tworzą wszystkie trapezy przecinające s_i
    - jest to też suma wszystkich trapezów, które zostaną stworzone

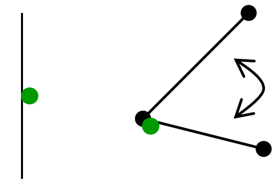


Konstrukcja T(*S*) – wyznaczenie strefy dla s_i



Przeszukaj dla p <u>strukturę D</u> aż do znalezienia Δ_0

jeśli p istniało już w strukturze – uwaga!



$j \leftarrow 0$

while q leży na prawo od *rightp*(Δ_j)

if *rightp*(Δ_j) leży powyżej s_i

then niech Δ_{j+1} będzie dolnym prawym sąsiadem Δ_j

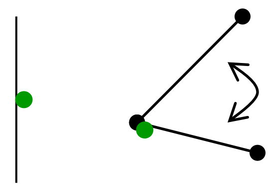
else niech Δ_{j+1} będzie górnym prawym sąsiadem Δ_j

$j \leftarrow j + 1$

return $\Delta_0, \Delta_1, \ldots, \Delta_k$

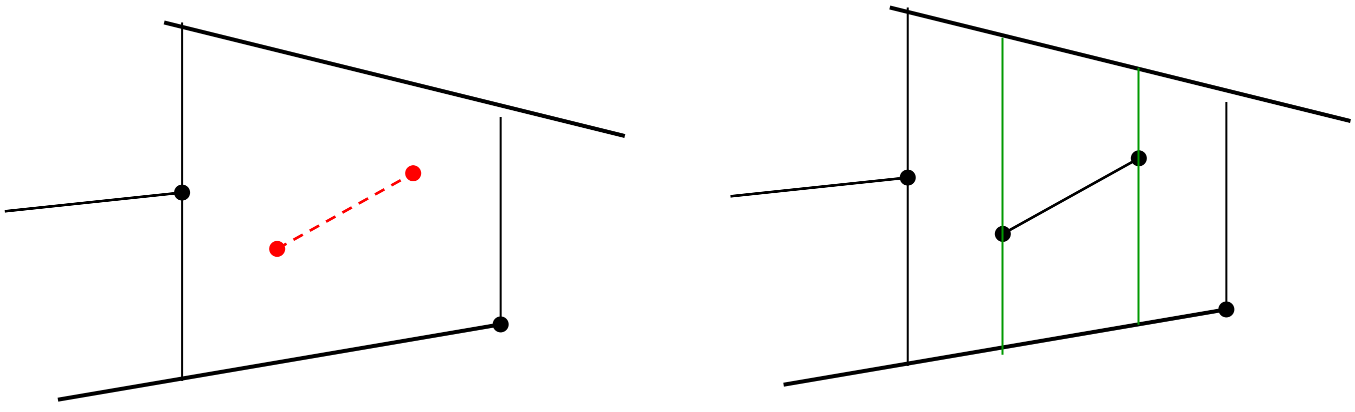
Konstrukcja T(S) – wyznaczenie strefy dla s_i

Jeśli p jest już w strukturze



- p leży na prostej pionowej
  – przyjmujemy, że leży po prawej stronie
- p jest wspólnym początkiem z innym odcinkiem s
  – porównujemy nachylenie:
  jeżeli nachylenie jest mniejsze od s,
  to p leży poniżej s

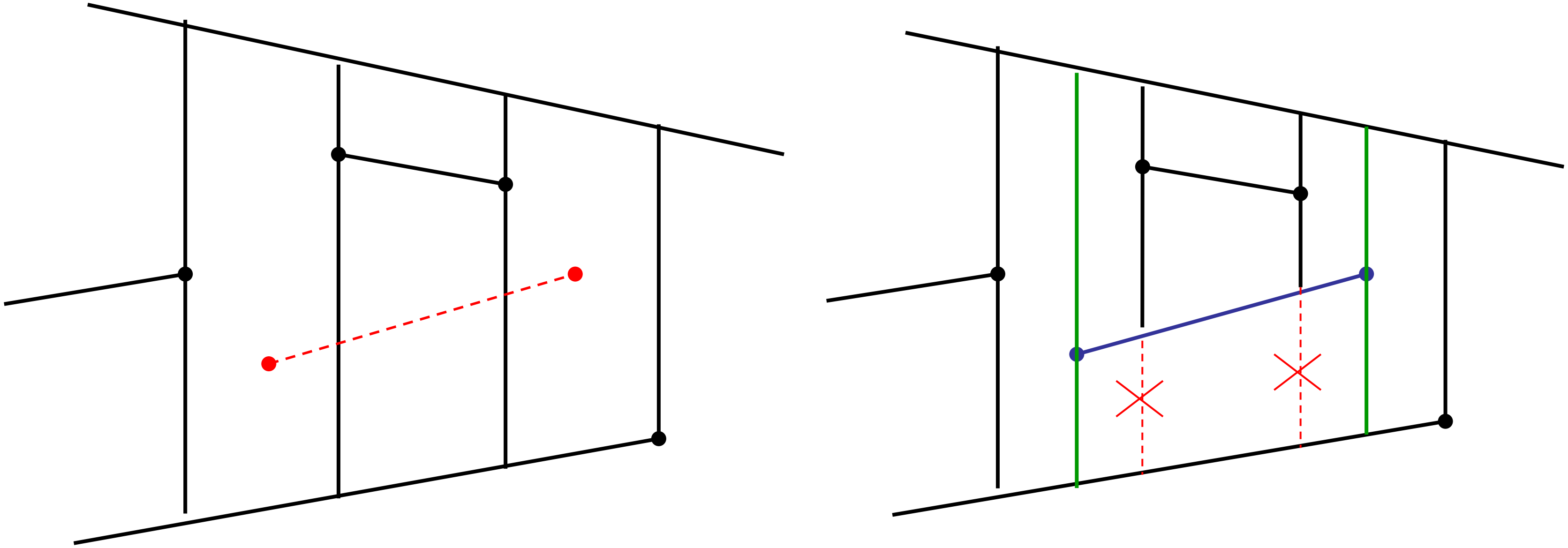
Konstrukcja T(*S*)



$T(S_{i-1})$ $T(S_i)$

- Usuwamy Δ z T
- Zastępujemy przez 4 trapezy
- Aktualizujemy informacje dla trapezów o sąsiadach, bottom(Δ), top(Δ) leftp(Δ), rightp(Δ)

Konstrukcja T(*S*)



Wstawiamy rozszerzenia pionowe przechodzące przez końce *s*
– każde dzieli trapezy Δ_0 i Δ_k na trzy nowe trapezy
Skracamy rozszerzenia pionowe, żeby stykały się z *s*
Aktualizujemy informacje dla trapezów

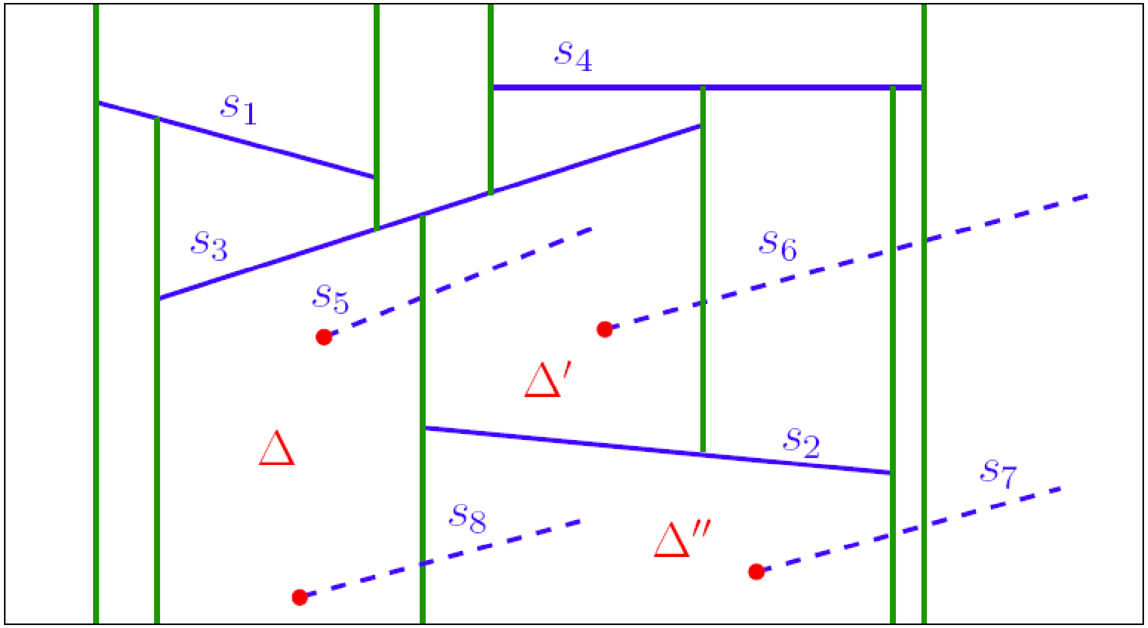
Oczekiwany czas konstrukcji
mapy trapezowej - $O(n \log n)$

Konstrukcja T(*S*) – wersja 2

Wstęp

- wyznaczenie losowej permutacji $(s_1,s_2,\ldots,s_n)$ dla zbioru odcinków *S*
- inicjalizacja struktury danych dla prostokąta zewnętrznego
- inicjalizacja list konfliktów dla lewych wierzchołków odcinków w *S*
  - początkowo, jest tylko jedna lista konfliktów (dla wnętrza prostokąta zewnętrznego) obejmująca wszystkie odcinki
  - każdy odcinek zawiera wskaźnik do swojego konfliktowego trapezu (czyli na początku do prostokąta zewnętrznego)

Listy konfliktów – przykład



- $\mathcal{L}(\Delta) = \{s_5, s_8\}$
- $\mathcal{L}(\Delta') = \{s_6\}$
- $\mathcal{L}(\Delta'') = \{s_7\}$

Konstrukcja T(*S*) – wersja 2

- W *i*-tym kroku dysponujemy
  - reprezentacją $T(S_i)$, $S_i=\{s_1,s_2,\ldots,s_i\}$
  - dla każdego trapezu Δ z $T(S_i)$
    - lista konfliktów $\mathcal{L}(\Delta)$ ze wskaźnikami do wszystkich odcinków w $S \setminus S_i$ których lewe wierzchołki znajdują się w Δ
  - dla każdego odcinka s z $S \setminus S_i$
    - wskaźnik do trapezu Δ z $T(S_i)$, który zawiera lewy wierzchołek
- … i wstawiamy s_{i+1} z uaktualnieniem struktury danych

Uaktualnienie mapy trapezowej dla wprowadzanego odcinka s_i

- wiadomo, który trapez w $T(S_{i-1})$ zawiera lewy wierzchołek s_i - dzięki liście konfliktów
- wszystko jest przeprowadzane w strefie dla s_i
- idziemy (miotłą) od lewej do prawej, uaktualniając mapę trapezów
- miotła przecina na raz nie więcej niż dwa trapezy
- k_i – liczba trapezów w $T(S_i)$ definiowanych przez $s_{i,}$, to jednocześnie maksymalna liczba zdarzeń w procedurze zamiatania
- aktualizacja jest przeprowadzana w czasie $O(k_i)$

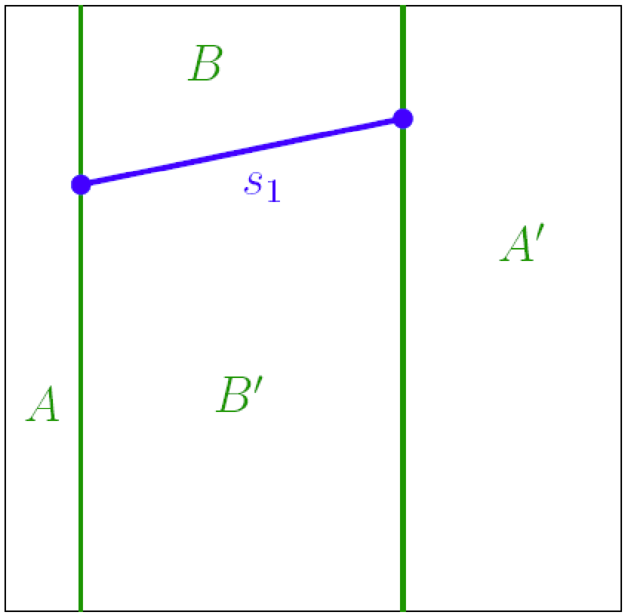
Uaktualnienie list konfliktów

- musimy także uaktualnić listy konfliktów
- lewe wierzchołki jeszcze niewstawionych odcinków są przenoszone z usuniętych trapezów do nowo stworzonych
- każdy usuwany trapez jest zawarty w sumie czterech nowych trapezów
- aktualizacja może zostać przeprowadzona w czasie $O(X_i)$, gdzie X_i jest liczbą lewych wierzchołków niewstawionych odcinków w strefie dla s_i

Procedura konstrukcji mapy trapezowej tą metodą ma złożoność $O(n \log n)$

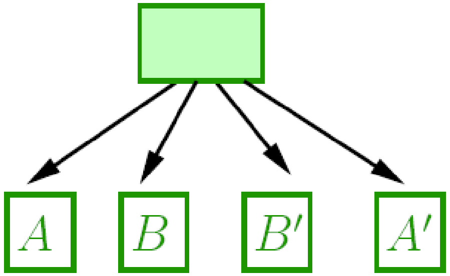
Struktura przeszukiwań D – wersja 1: graf historii

Graf historii dla konstrukcji mapy trapezowej



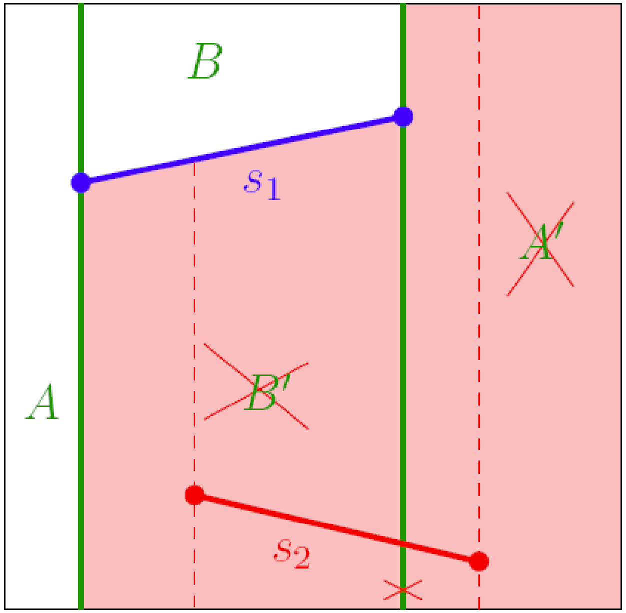
mapa trapezowa

korzeń odpowiada prostokątowi zewnętrznemu



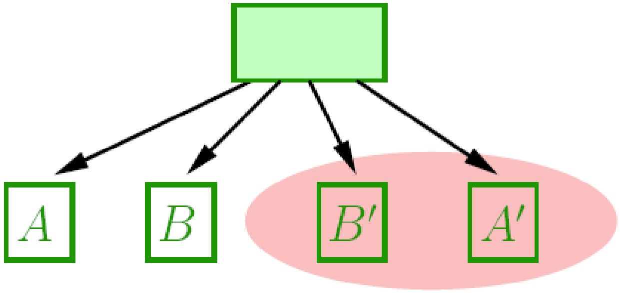
graf historii

Uaktualnienie grafu historii (1)



mapa trapezowa

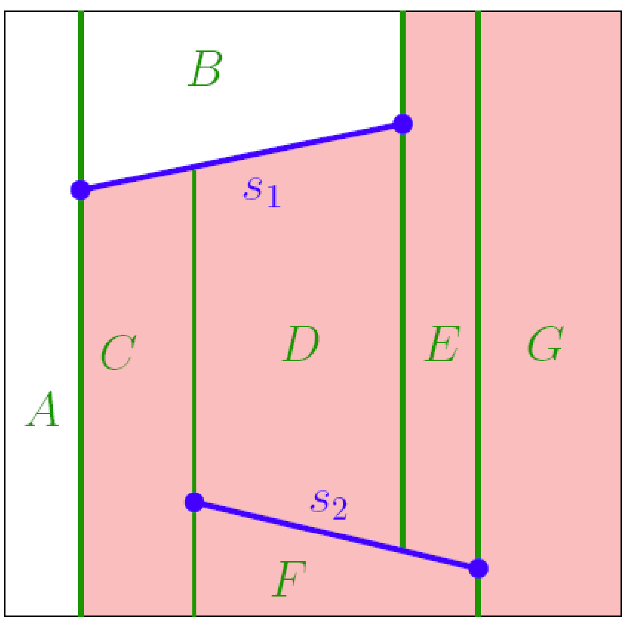
A' i B' zostają usunięte z mapy trapezów, ale pozostają w grafie historii

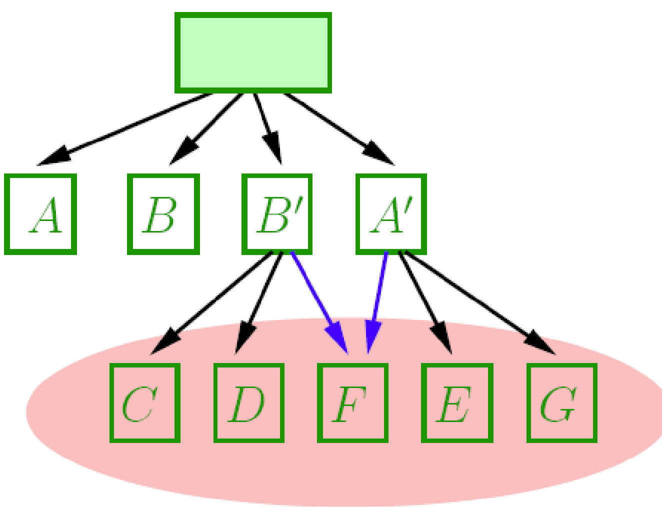


graf historii

Uaktualnienie grafu historii (2)

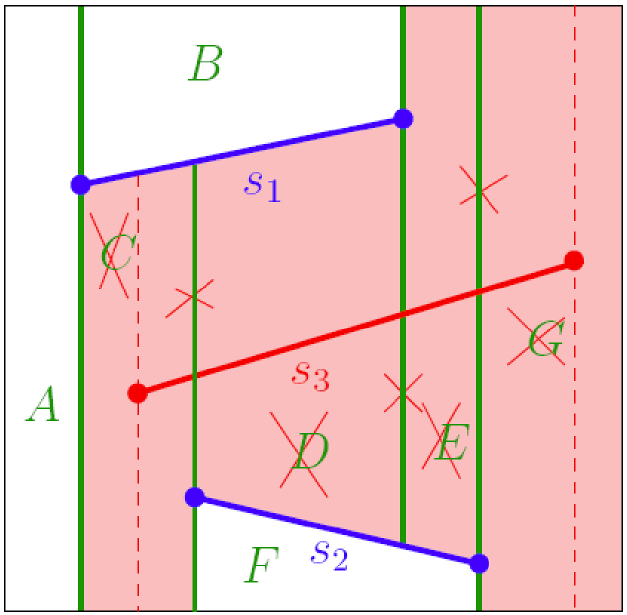
połączenie każdego usuniętego trapezu z wszystkimi nowymi trapezami, które mają z nim część wspólną (więcej niż tylko wspólną krawędź)

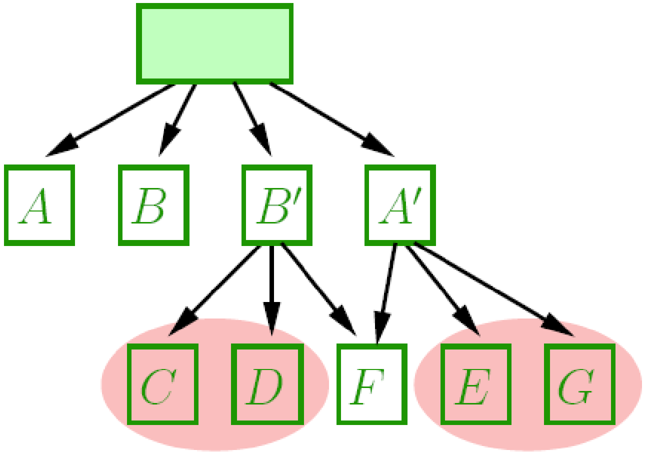




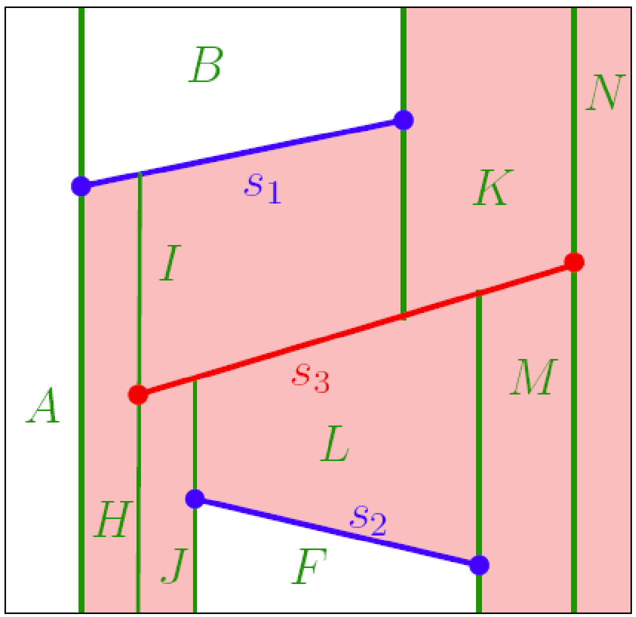
graf historii nie jest drzewem!
Acykliczny graf skierowany

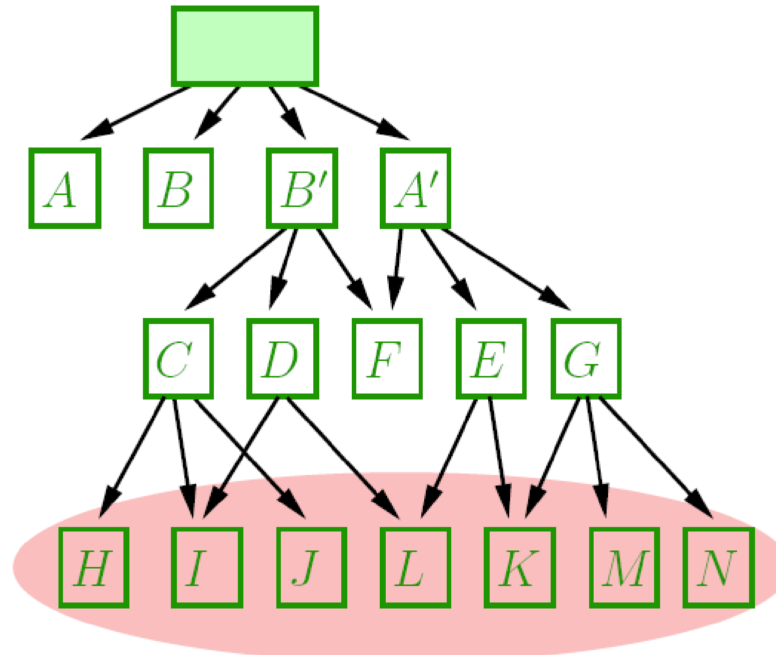
Uaktualnienie grafu historii (3)





Uaktualnienie grafu historii (4)





Uaktualnianie grafu historii – koszt *O*(*n*)

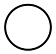
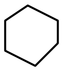
Udzielanie odpowiedzi z wykorzystaniem grafu historii

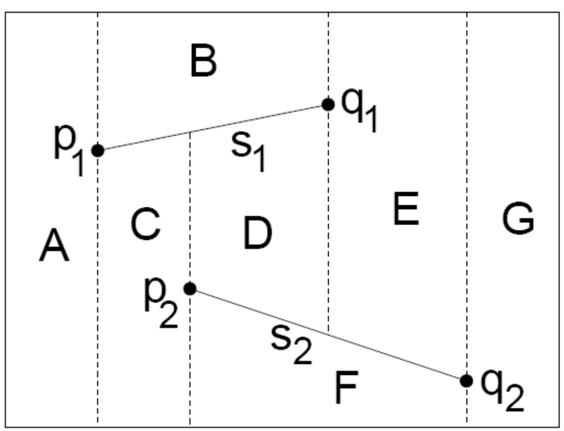
- zadany punkt *q* jest definiowany przez swoje współrzędne
- jeśli bieżący element grafu jest liściem, koniec
- w przeciwnym wypadku jeden z trapezów – potomków w grafie zawiera punkt *q*
  - maksymalnie 4 trapezy potomne
- przechodzimy w dół do potomka i powtarzamy procedurę

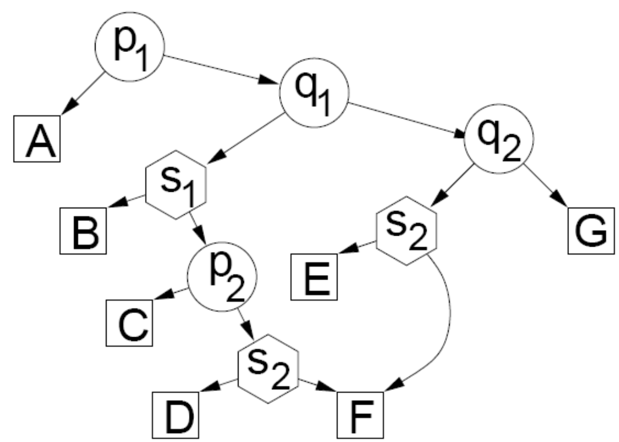
Złożoność $O(\log n)$

Struktura przeszukiwań D
- wersja 2: graf wyszukiwania

Alternatywna wersja grafu wyszukiwania dla mapy trapezowej

- ściany wyłącznie w liściach
- węzły wewnętrzne dwóch kategorii
  - x-węzły związane z wierzchołkiem (współrzędna wierzchołka) 
  - y-węzły związane z odcinkiem (wskaźnik do odcinka) 



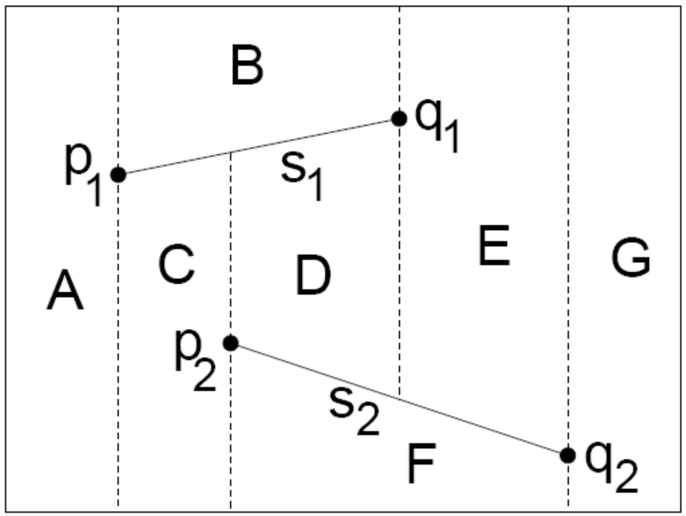


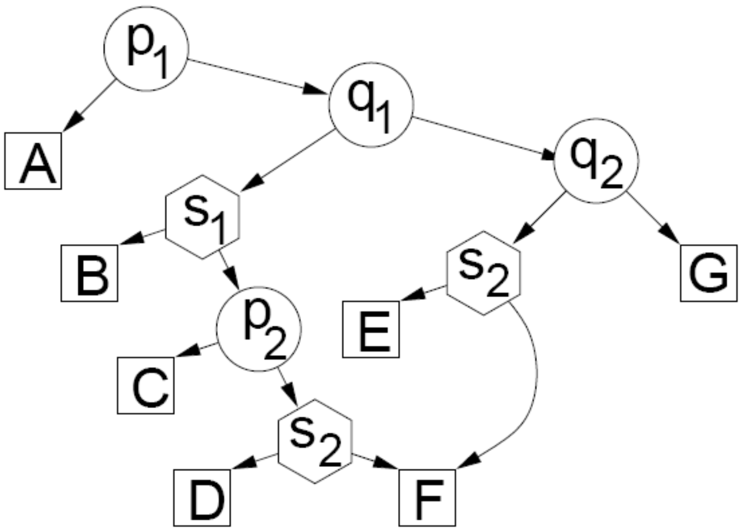
Trapez z T(S) ma wskaźnik do odpowiadającego mu liścia w D,
a liść w D ma wskaźnik do odpowiadającemu mu trapezu w T(S)

Graf wyszukiwania

W każdym węźle na ścieżce punkt q jest sprawdzany, do którego dziecka węzła go skierować

- w x-węźle test: czy q leży na lewo, czy prawo pionowej prostej przechodzącej przez p?
- w y-węźle test: czy q leży powyżej, czy poniżej odcinka s?



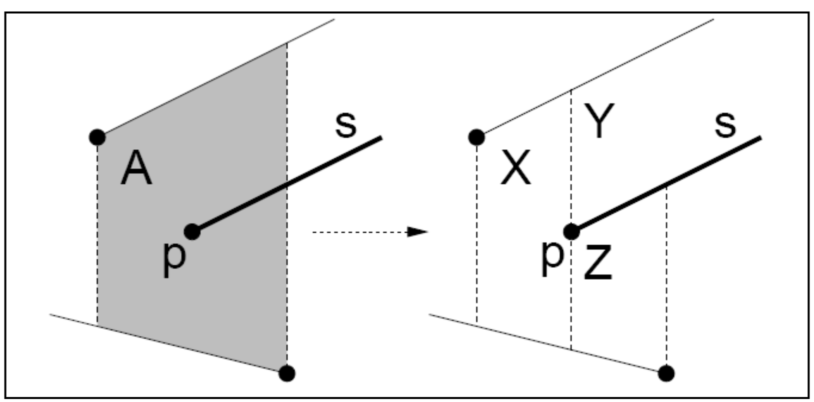


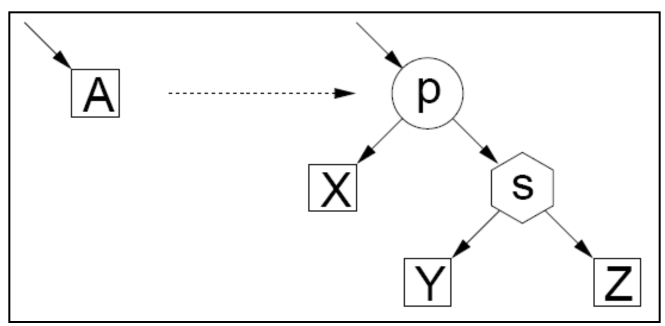
Graf wyszukiwania

Przyrostowa konstrukcja grafu

– usuwany trapez jest zastępowany fragmentem struktury wyszukiwania kierującym do jednego z nowo stworzonych trapezów

  - przypadek 1 – pojedynczy trapez A (zawierający jeden wierzchołek odcinka) jest zastępowany przez trzy trapezy X, Y i Z

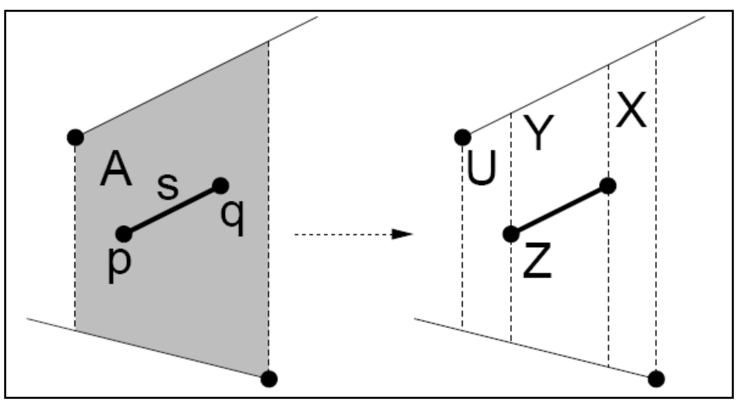


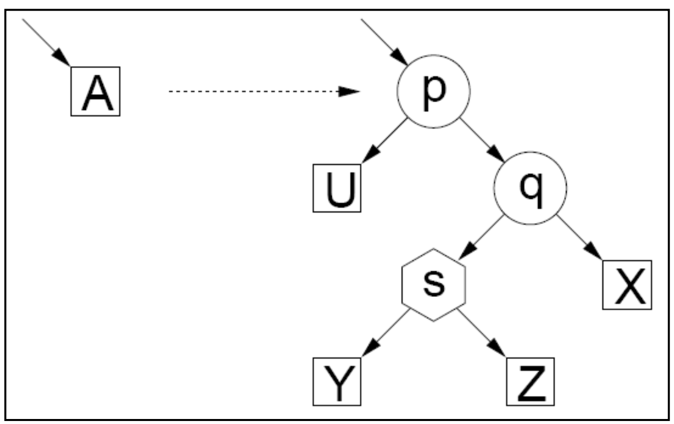


Graf wyszukiwania

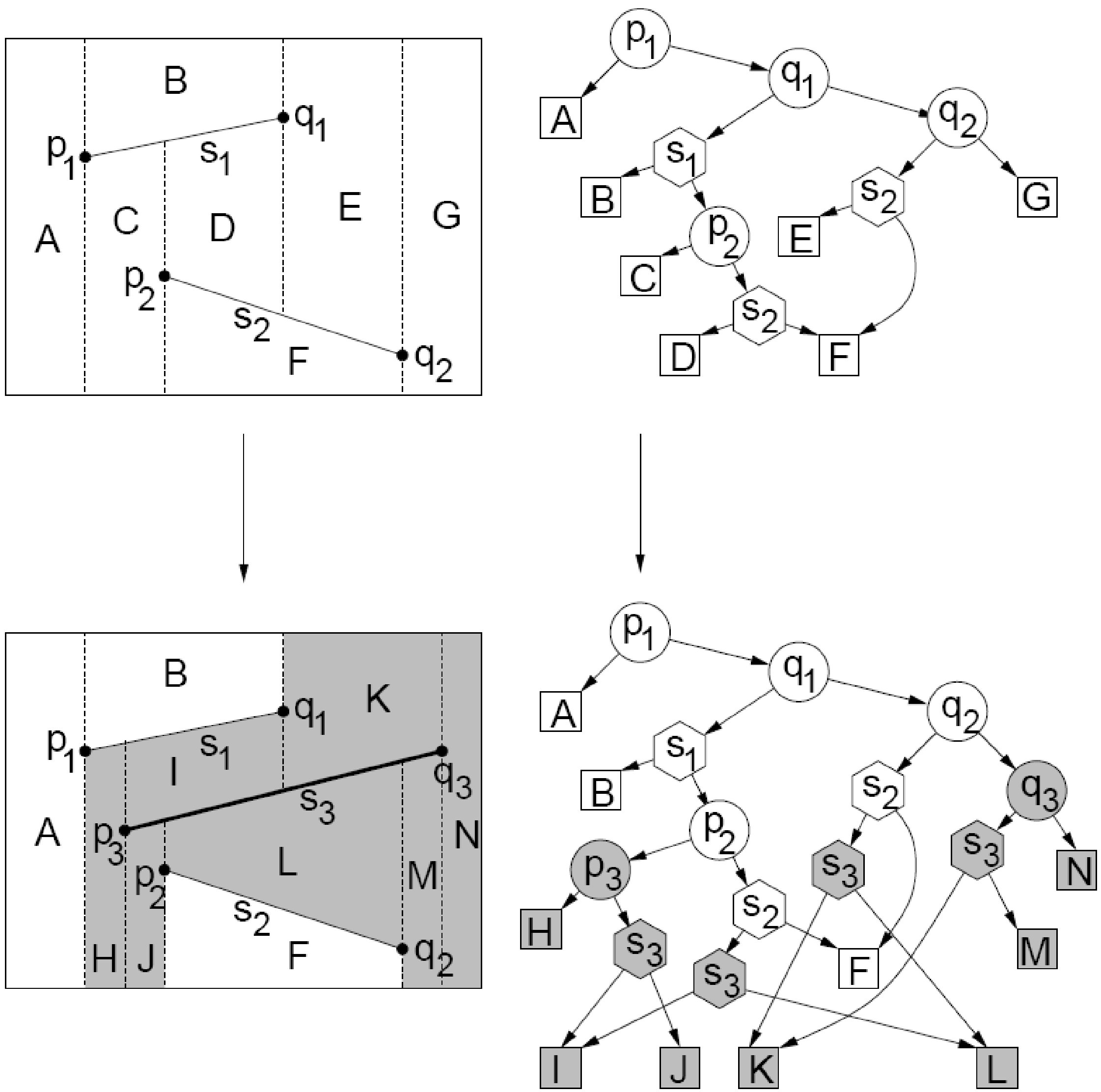
Przyrostowa konstrukcja grafu

- przypadek 2 – pojedynczy trapez A (przecięty całkowicie) jest zastępowany przez dwa trapezy Y i Z
- przypadek 3 – pojedynczy trapez zawiera cały odcinek i zostaje podzielony na cztery trapezy U, X, Y i Z

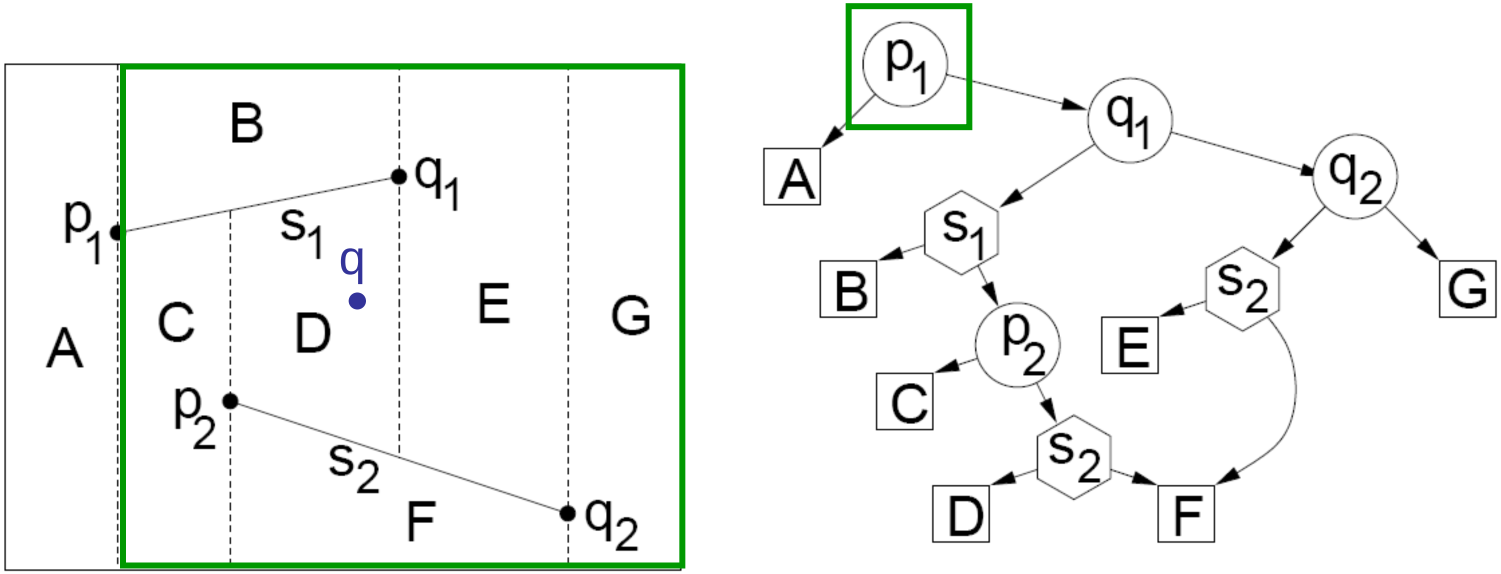




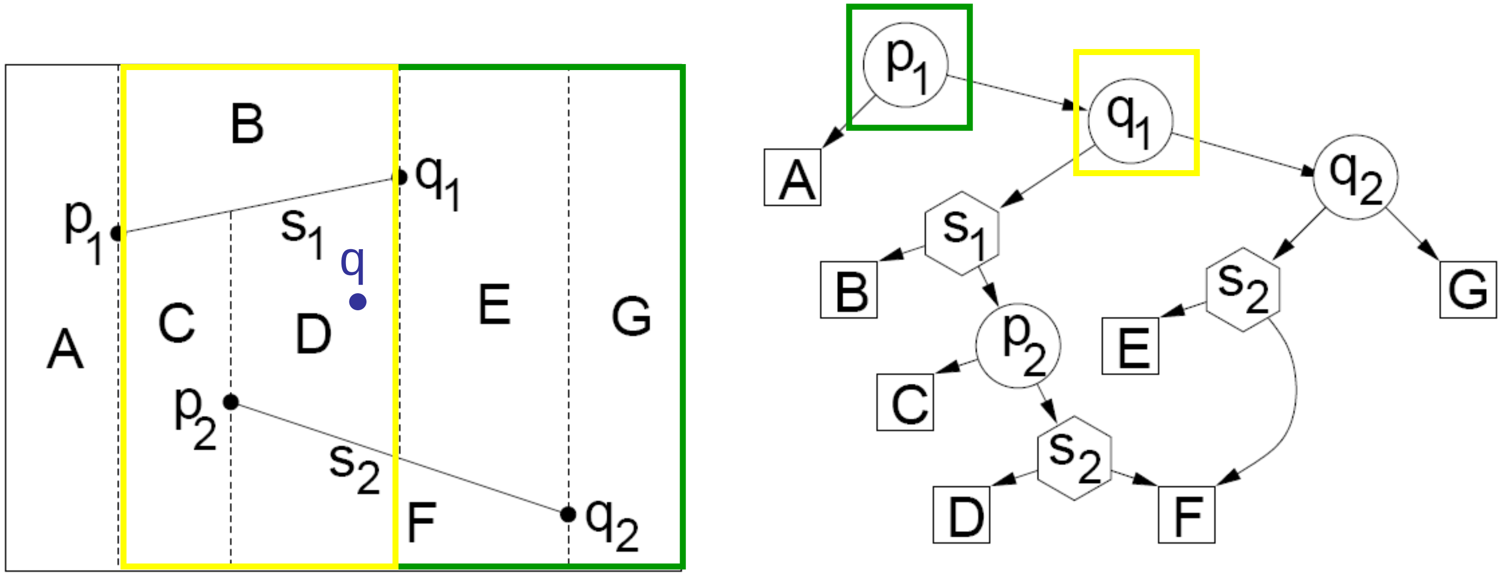
Graf wyszukiwania
Przykład konstrukcji



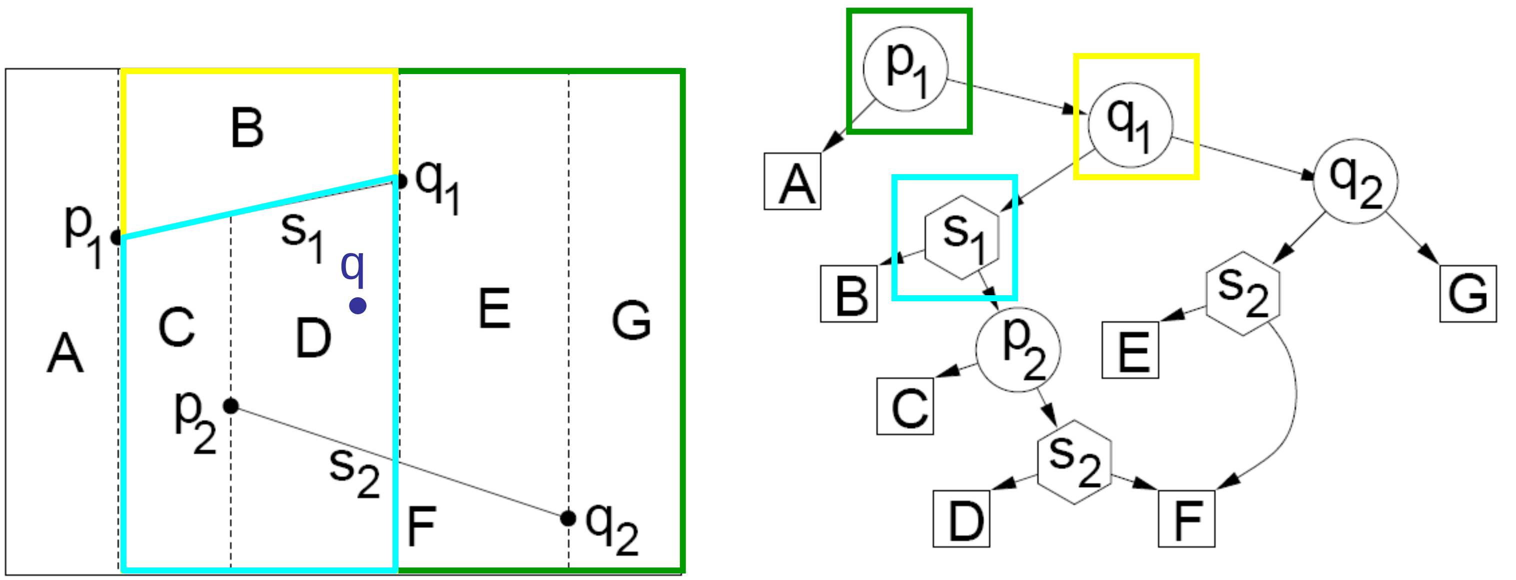
Udzielanie odpowiedzi



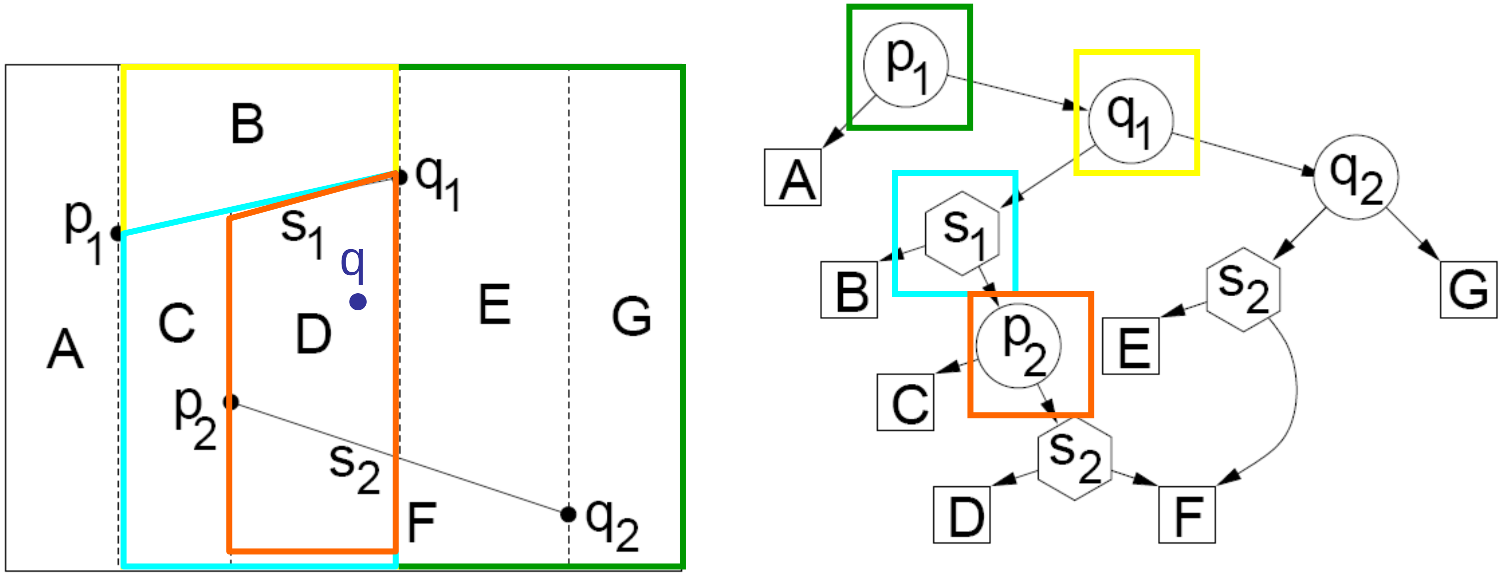
Udzielanie odpowiedzi



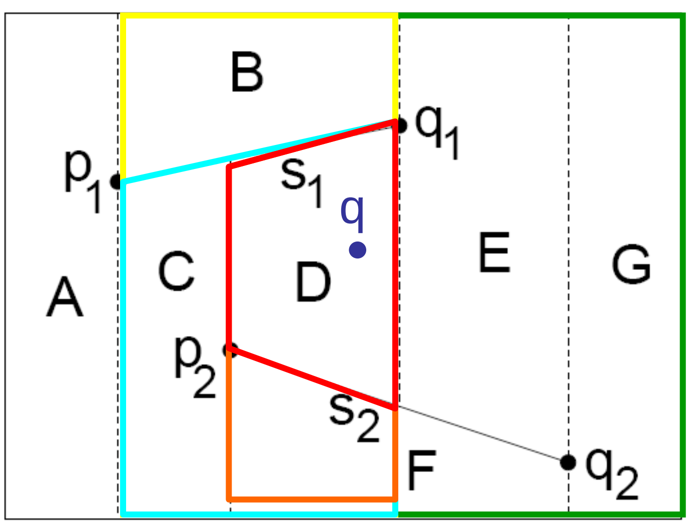
Udzielanie odpowiedzi

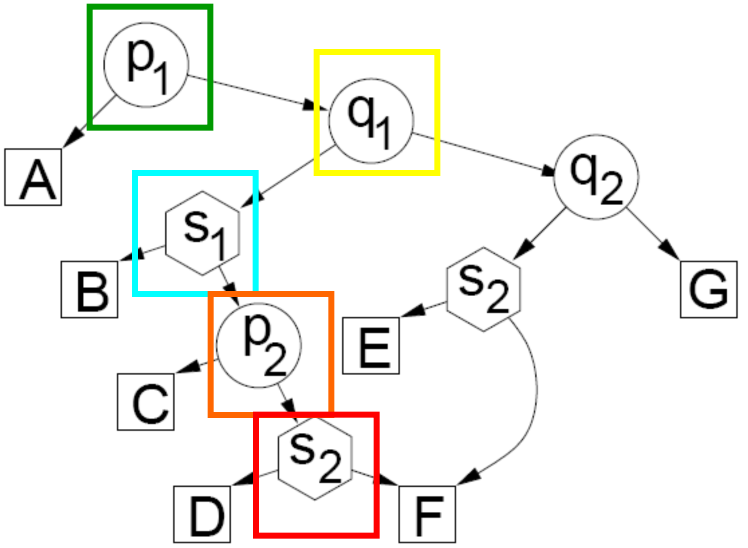


Udzielanie odpowiedzi

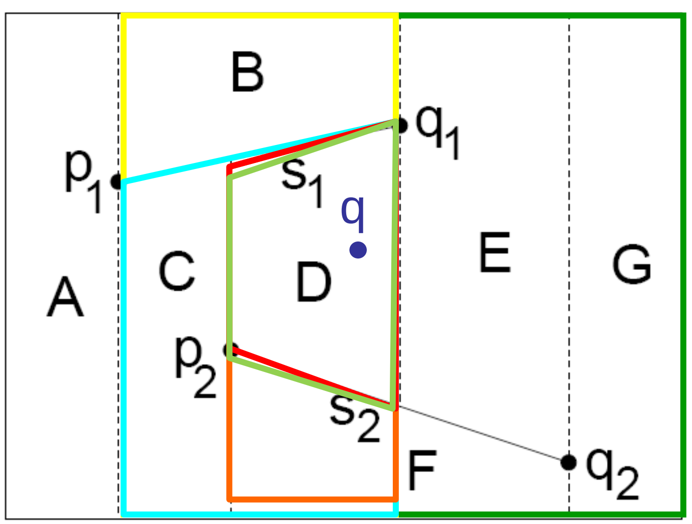


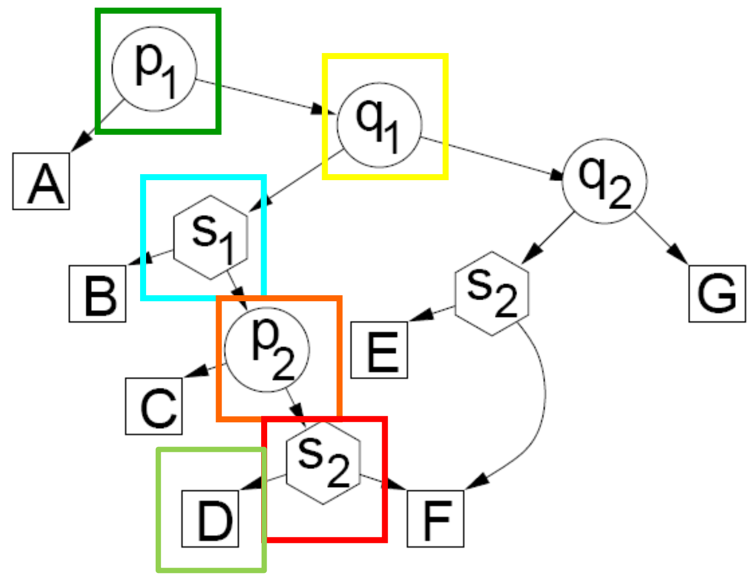
Udzielanie odpowiedzi





Udzielanie odpowiedzi





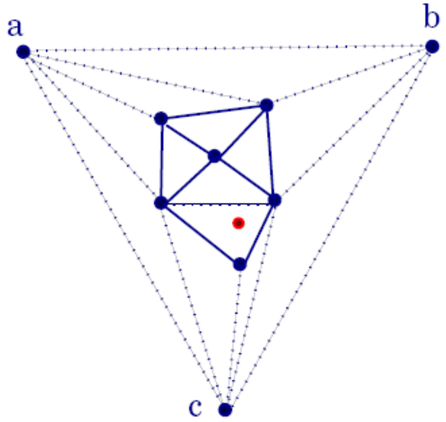
Oczekiwany czas konstrukcji grafu przeszukiwań - $O(n \log n)$
(razem z konstrukcją mapy trapezowej)

Oczekiwany rozmiar grafu przeszukiwań $O(n)$

Dla dowolnego punktu q oczekiwany czas zapytania $O(\log n)$

Metoda doskonalenia triangulacji
Algorytm Kirkpatrick'a

Założenie: dany podział S jest triangulacją



Tworzymy sekwencję T_0 , T_1 , … , T_k

coraz *<u>zgrubniejszych</u>* triangulacji, gdzie:

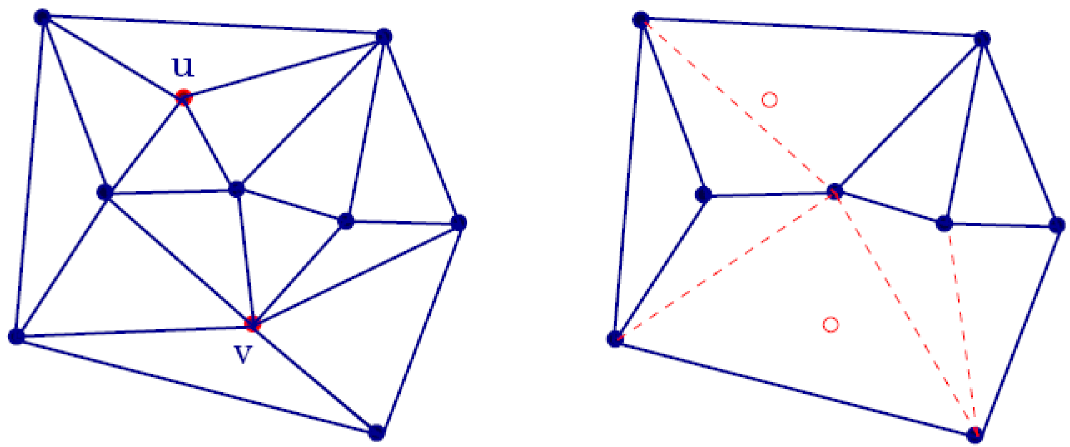
T_0 jest początkową triangulacją,

T_k jest trójkątem *abc* otaczającym obszar

Każdy trójkąt w T_{i+1} zachodzi na pewne trójkąty T_i

Metoda doskonalenia triangulacji

W kolejnych krokach usuwamy pewne wierzchołki i retriangulujemy wnękę



Jeśli usuwany wierzchołek ma stopień *d*, to wnęka może być wypełniona *d* – 2 trójkątami.

Jeżeli wierzchołki rozpatrywane są niezależne, to czyni retriangulację łatwiejszą.
Wnęki mogą być rozpatrywane osobno.

Metoda doskonalenia triangulacji

procedure TRIANGLE(S)

T := zbiór trójkątów tworzących S;

V := {odpowiedniki trójkątów}; E := $\emptyset$;

while |T| > 1 ***do***

wybierz niezależny zbiór wewnętrznych wierzchołków;

usuń z T trójkąty z tymi wierzchołkami;

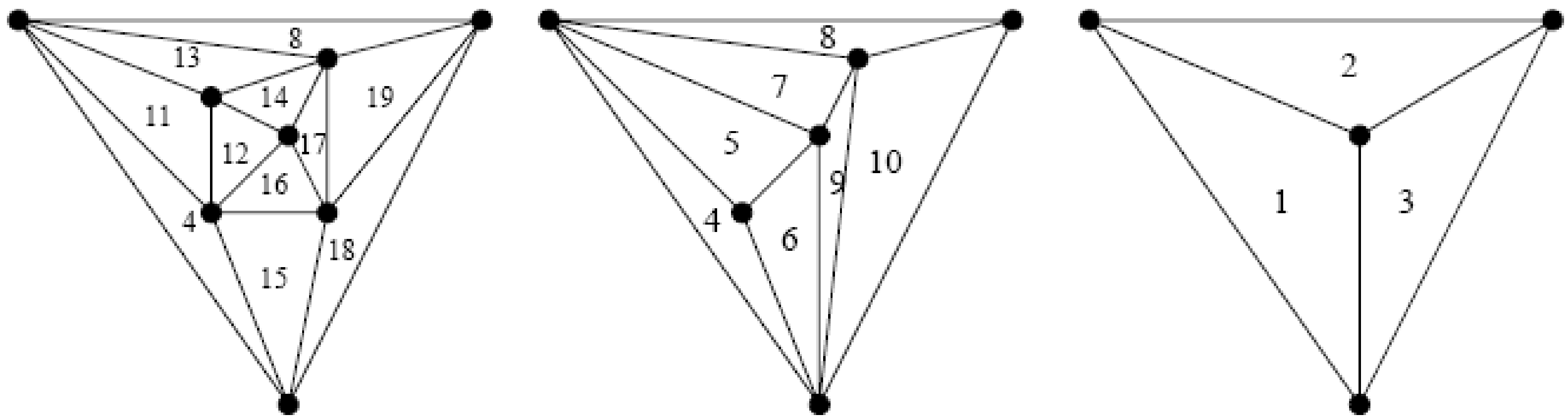
usuń z S wybrane wierzchołki wraz z incydentnymi krawędziami;

ponownie strianguluj S i dodaj nowe trójkąty do T;

V := V $\cup$ {nowe trójkąty};

E := E $\cup$ {(N,U): N-nowy, U-usunięty, N $\cap$ S $\neq \emptyset$};

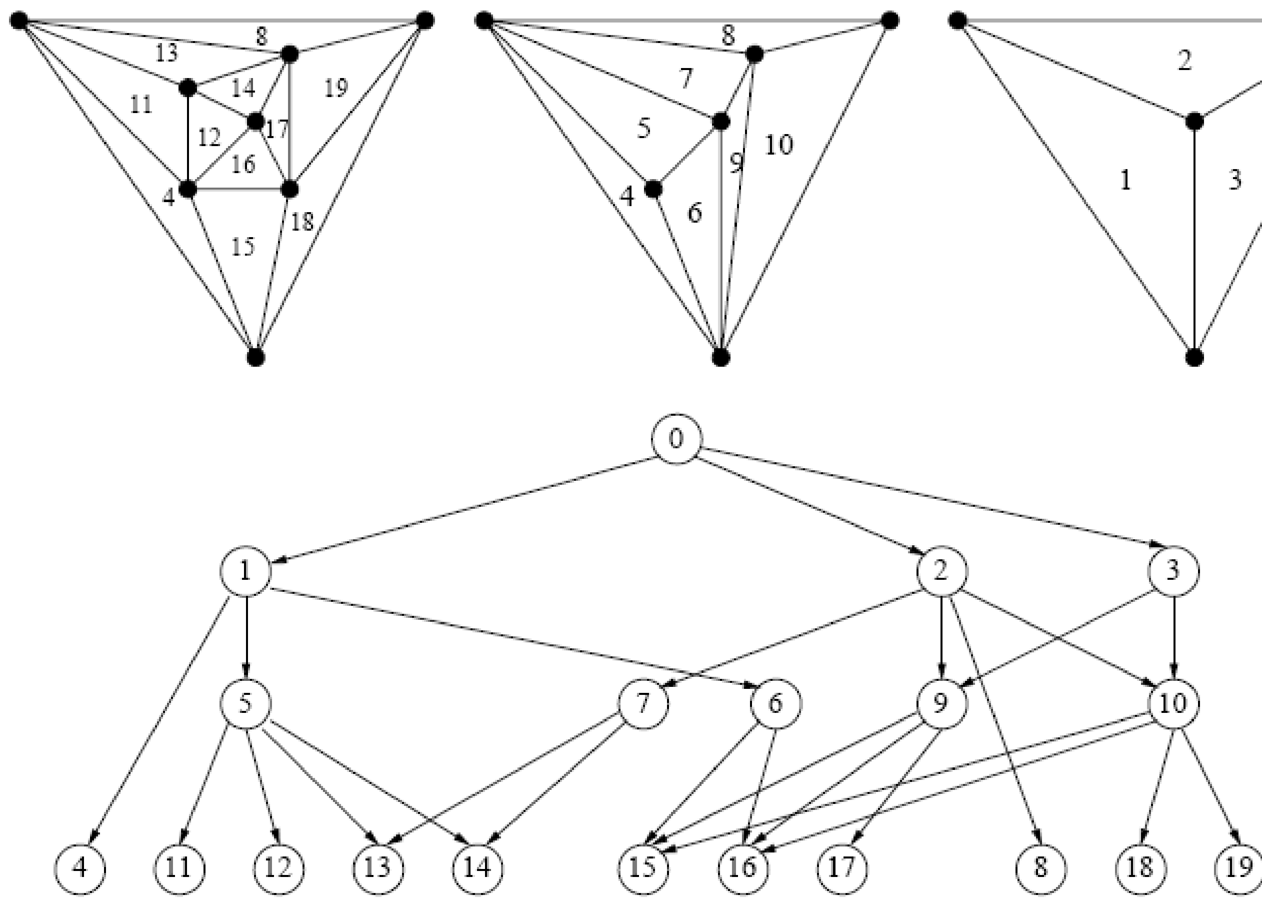
return (V, E);



Usuwane i tworzone trójkąty odpowiadają wierzchołkom grafu skierowanego

Każdy poziom odpowiada nowym trójkątom

Krawędzie są skierowane w dół i łączą trójkąty, których przecięcie nie jest puste

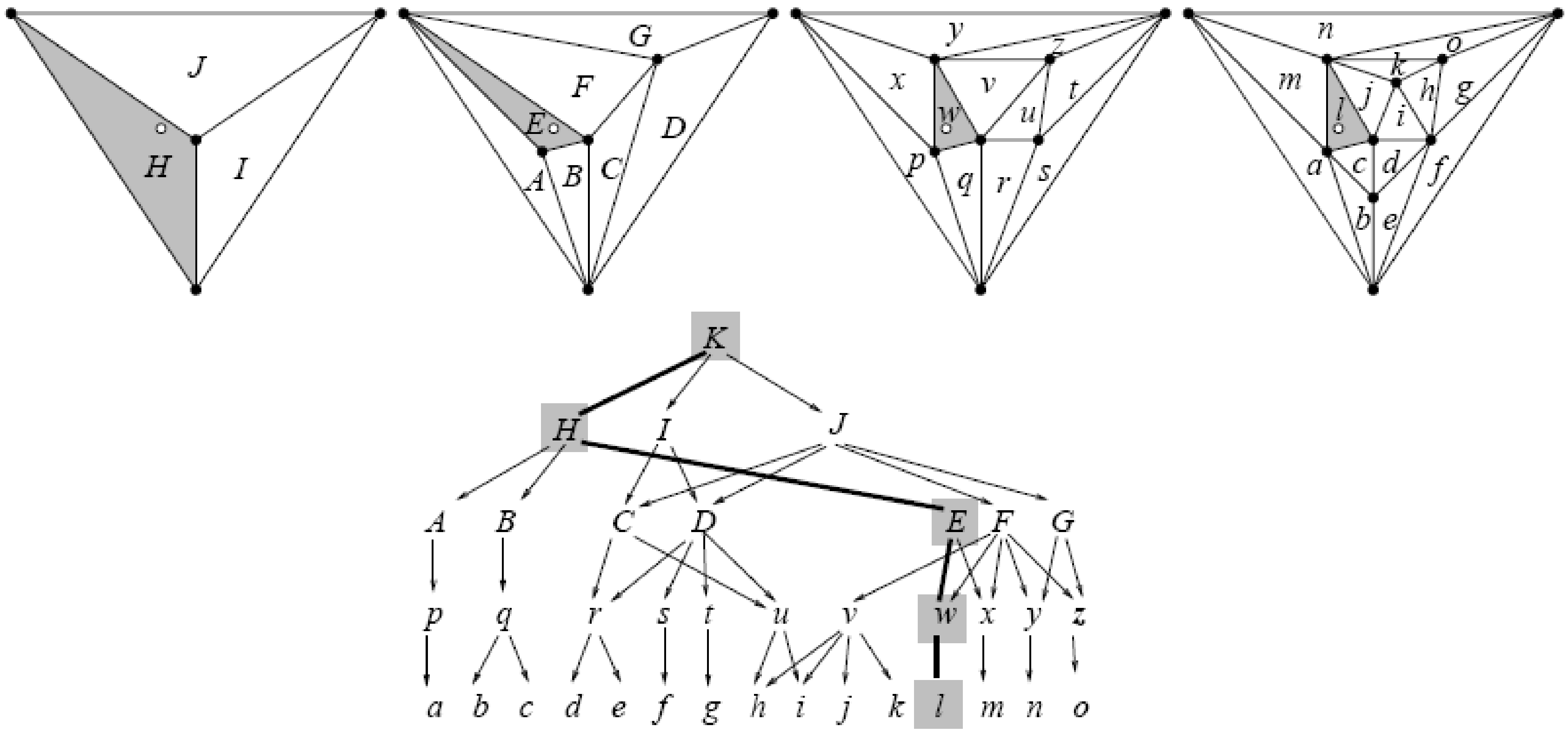


Liczba poziomów w grafie jest logarytmiczna

Jego rozmiar jest liniowy ze względu na liczbę wierzchołków

Każdy węzeł ma stopień ograniczony przez stałą

Przeszukiwanie



Wyszukujemy właściwy trójkąt idąc od korzenia (pojedynczego trójkąta) sprawdzając, do których trójkątów należy dany punkt

Na każdym poziomie wykonujemy stałą liczbę porównań

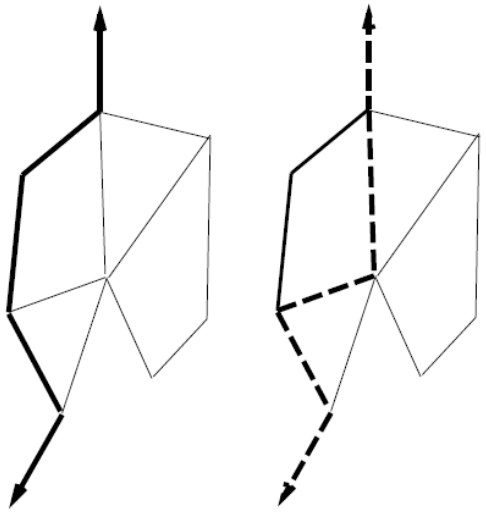
Dla obszaru striangulowanego o n trójkątach można
- zlokalizować punkt w czasie $O(\log n)$
- z pomocą struktury o rozmiarze $O(n)$ stworzonej w czasie $O(n)$

Czas preprocessingu – triangulacji: $O(n \log n)$

Metoda separatorów

Wejście – obszar z podziałem poligonowym
dany jako graf acykliczny z jednym źródłem i jednym ujściem

Definicja
Separatorem nazywamy łamaną monotoniczną względem danego kierunku, rozdzielającą podział i tworzoną przez jego krawędzie.

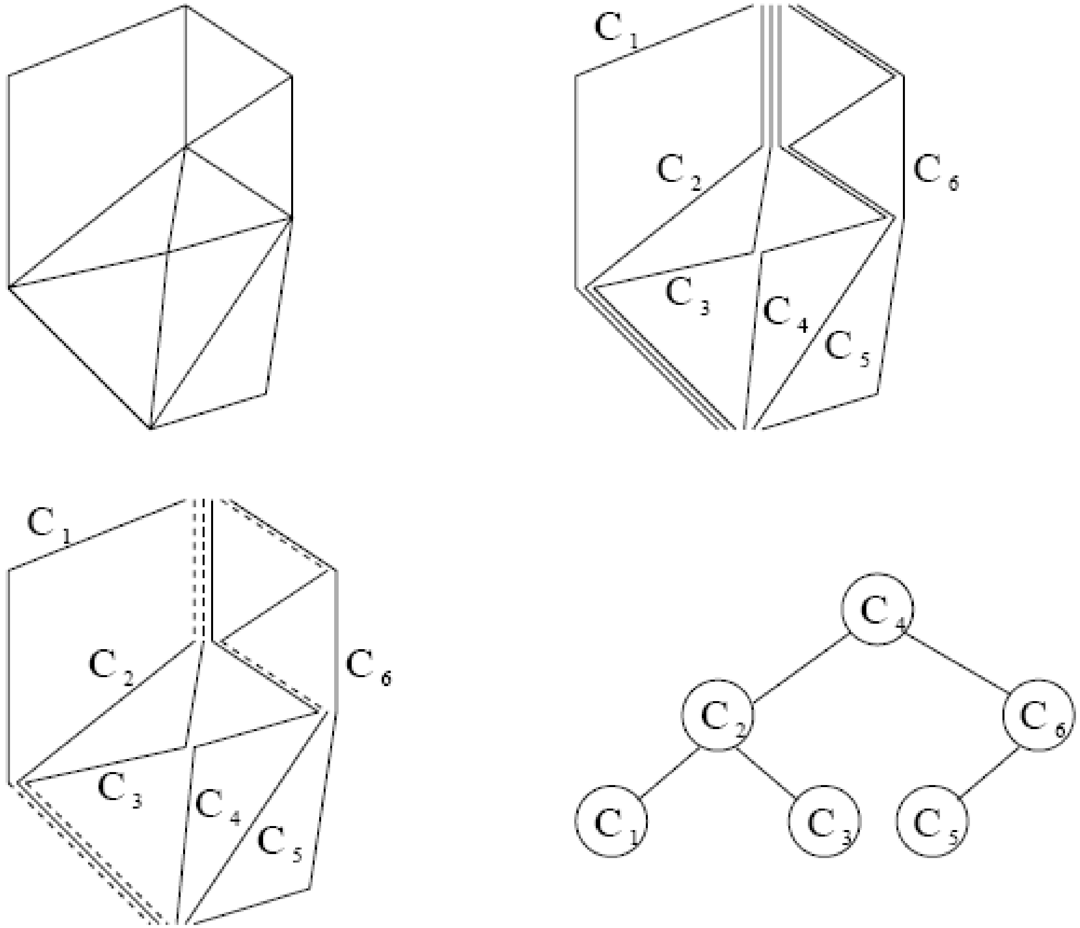


Metoda separatorów

Cel: stworzyć ciąg separatorów (C_n) taki, że :

- każde dwa sąsiednie separatory wyznaczają dokładnie jeden obszar należący do podziału S,
- każdy obszar podziału jest wyznaczany jednoznacznie przez separatory,
- wszystkie wierzchołki separatorów o niższych numerach leżą po tej samej stronie separatora o numerze wyższym.

Metoda separatorów



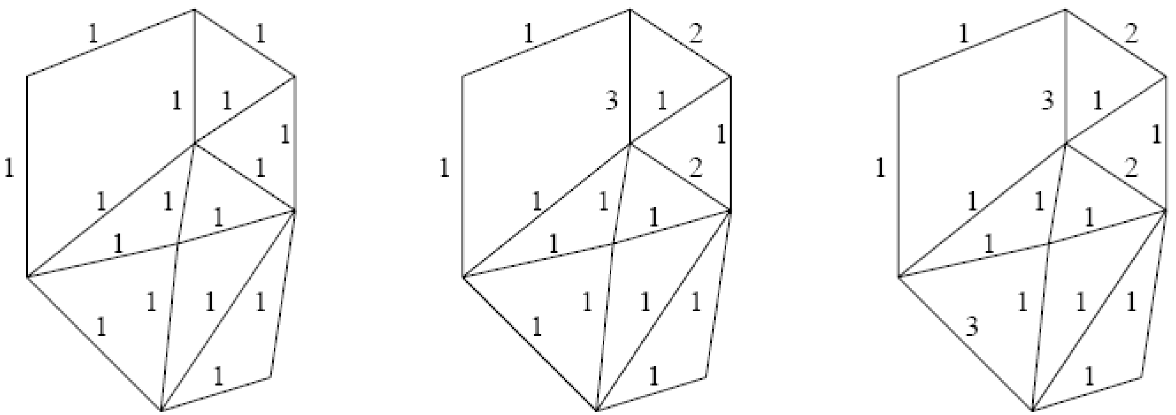
Metoda separatorów

Definicja.
Niech zbiór wierzchołków podziału tworzy ciąg uporządkowany względem współrzędnej *y*-owej (w przypadku równych wartości - względem współrzędnej *x*-owej).
Wierzchołek v_k jest ***regularny***, gdy istnieją krawędzie (v_i,v_k) i (v_k,v_j) dla $i < k < j$. Podział jest regularny, gdy wszystkie wierzchołki ciągu poza pierwszym i ostatnim są regularne.

Każdy podział można zregularyzować stosując algorytm podobny do algorytmu podziału wielokąta na wielokąty monotoniczne (łączymy krawędzią wierzchołek nieregularny v z pomocnikiem sąsiadującej z v krawędzi) lub triangulując otoczkę wypukłą podziału.

Metoda separatorów

- Inicjujemy graf
- Wyznaczamy wagi krawędzi
  - liczba separatorów, do których będzie należeć dana krawędź



- Wyznaczamy separatory

Metoda separatorów

Zakładamy, że krawędzie podziału są skierowane od wierzchołka o mniejszym indeksie do wierzchołka o większym indeksie.

IN(v) – zbiór krawędzi dochodzących do wierzchołka v
OUT(v) – zbiór krawędzi odchodzących od wierzchołka v

W(e) – **waga** krawędzi e (liczba separatorów, do których należy e)

$W_{IN}(v) := \Sigma_{e \in IN(v)} W(e)$ oraz $W_{OUT}(v) := \Sigma_{e \in OUT(v)} W(e)$.

każda krawędź należy do co najmniej jednego łańcucha

Wagi można tak dobrać, aby:
- waga każdej krawędzi była dodatnia,
- $W_{IN}(v) = W_{OUT}(v)$ dla każdego v.

separatory nie przecinają się

Metoda separatorów

Wyznaczanie wag

Przypisz każdej krawędzi wagę według następującego algorytmu:

for każda krawędź e: $W(e) := 1$

for każdy wierzchołek v w kolejności indeksów rosnących:

$W_{IN}(v) := \Sigma_{e \in IN(v)} W(e)$

d: = pierwsza z lewej krawędź odchodząca z v

if $W_{IN}(v) > W_{OUT}(v)$

then $W(d) := W_{IN}(v) - W_{OUT}(v) + W(d)$;

for każdy wierzchołek v w kolejności indeksów malejących:

$W_{OUT}(v) := \Sigma_{e \in OUT(v)} W(e)$

d := pierwsza z lewej krawędź dochodząca do v

if $W_{OUT}(v) > W_{IN}(v)$

then $W(d) := W_{OUT}(v) - W_{IN}(v) + W(d)$

Metoda separatorów

Po wyznaczeniu wag krawędzi
możemy wyznaczyć separatory w odpowiedniej
kolejności tak, by były od razu posortowane

W każdym wierzchołku od źródła do ujścia
wybieramy pierwszą z lewej krawędź,
która ma niezerową wagę.
Zmniejszamy wagę tej krawędzi o jeden
i przechodzimy rekurencyjnie dalej

Jedno przejście od źródła do ujścia wyznacza jeden separator

Metoda separatorów

struktura danych:

Drzewo binarne:

- liście – podobszar S
- węzły: ciąg krawędzi separatora

Ale: każda krawędź separatora jest przechowywana tylko raz.

W danym węźle przechowujemy tylko te krawędzie separatora, które jeszcze nie zostały zapamiętane w węźle przodku

Struktura o liniowym rozmiarze względem liczby krawędzi obszarów

Strukturę można utworzyć w czasie $O(n \log n)$

Metoda separatorów

Wyszukujemy binarnie separator, który znajduje się powyżej lokalizowanego punktu i taki, że poprzedzający go separator znajduje się poniżej tego punktu (wyszukujemy odpowiednią krawędź separatora i sprawdzamy, czy jest powyżej, czy poniżej danego punktu).

Kontynuujemy poszukiwanie w zawężonym obszarze.

Wyznaczamy w ten sposób obszar zawierający punkt.

Czas lokalizacji punktu w n-wierzchołkowym podziale wynosi $O(\log^2 n)$