Geometria obliczeniowa
Wykład 2

1. Otoczka wypukła (c. d.) :

- algorytm Chana,
- algorytm przybliżony,
- algorytm dynamiczny,
- dla wielokąta prostego.

2. Punkty dominujące.
3. Triangulacja wielokąta przekątnymi.
4. Optymalna triangulacja wielokąta wypukłego.
5. Podział wielokąta prostego na wielokąty wypukłe.

Algorytm Chana.

Podobnie do algorytmu Kirkpatricka i Seidela jest to optymalny algorytm znajdujący otoczkę wypukłą dla danego zbioru punktów P wrażliwy na wynik.

Jest prostszy do implementacji.

Jeden krok algorytmu składa się z dwóch faz:

- w pierwszej tworzymy otoczki wypukłe dla (nieposortowanych) podzbiorów zbioru P wykorzystując dowolny algorytm znajdujący otoczkę wypukłą w czasie O(n log n),
- w drugiej stosujemy algorytm Jarvisa, badając styczne do otoczek wypukłych zbudowanych w pierwszej fazie.

Algorytm ten może też być użyty do konstrukcji otoczki wypukłej w przestrzeni trójwymiarowej w czasie O(n log h).

Algorytm Chana.

p – punkt o najmniejszej współrzędnej y-owej

kontynuuj:=true ;

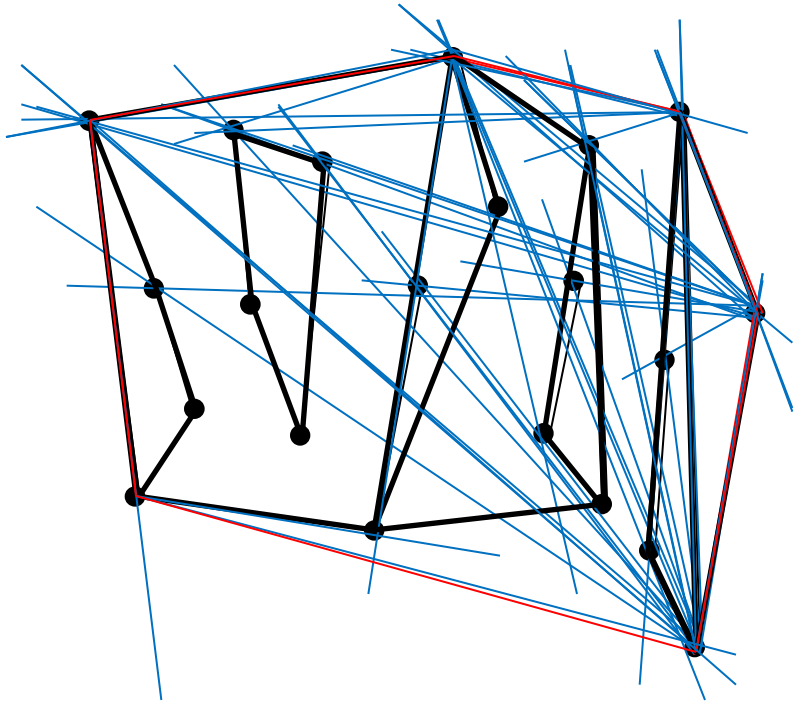
for t=0 **to** $\lceil \log \log n \rceil$ **do**

if kontynuuj **then**

podziel dany zbiór punktów P na podzbiory rozmiaru 2^{2^t} i znajdź otoczkę wypukłą dla każdego z nich;

zaczynając od p wykonaj co najwyżej 2^{2^t} kroków marszu Jarvisa względem prostych stycznych do danych otoczek;

if marsz Jarvisa skończył się w p **then** kontynuuj:=false;



Złożoność algorytmu.

Twierdzenie.

Algorytm znajduje otoczkę wypukłą w czasie O(n log h), gdzie h jest liczbą wierzchołków otoczki.

Dowód.

Istotne jest tylko $\lceil \log \log h \rceil$ kroków, w czasie których znajdujemy otoczkę.

W pierwszej fazie każdego kroku algorytmu dzielimy zbiór punktów na O(n/k) podzbiorów rozmiaru k (gdzie k jest odpowiednią potęgą 2).

Stąd koszt tej fazy wynosi O(n/k) x O(k log k) = O(n log k).

Znalezienie punktów styczności w drugiej fazie wymaga czasu O(log k)

(znajdując otoczkę zapamiętujemy jej łańcuchy w postaci umożliwiającej wyszukiwanie binarne (tablica, drzewo AVL)).

Stąd koszt drugiej fazy wynosi O(k) x O(n/k) x O(log k) = O(n log k).

Zatem złożoność algorytmu wynosi:

$$\sum_{t=0}^{\lceil \log \log h \rceil} O(n \log 2^{2^t}) = O(n) \sum_{t=0}^{\lceil \log \log h \rceil} 2^t = O(n 2^{1+\lceil \log \log h \rceil}) = O(n \log h)$$

Przybliżona otoczka wypukła.

Zalety rozwiązań przybliżonych: szybkość, prostota.

Wady: niedokładność wyniku.

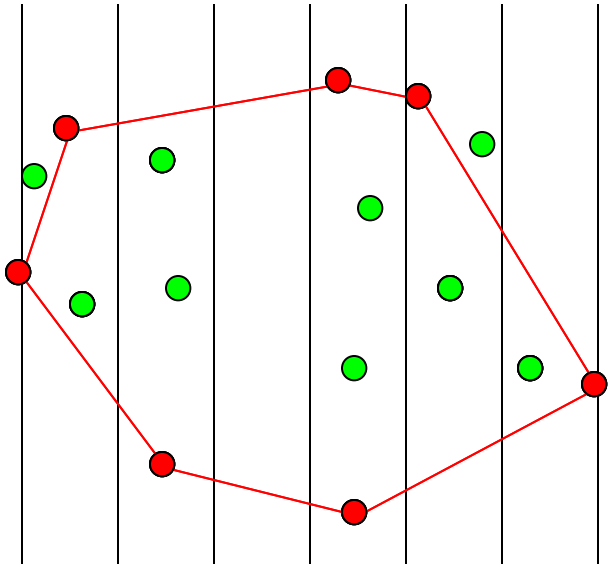
x_{min}:= skrajnie lewy punkt z S;

x_{max}:= skrajnie prawy punkt z S;

podziel przestrzeń między x_{min} a x_{max} na k przystających pasów;

w każdym pasie i obszarach zewnętrznych znajdź punkty skrajne względem współrzędnej y;

znajdź otoczkę wypukłą dla wybranych punktów;



Twierdzenie.

Dowolny punkt $p \in CH$, który nie należy do przybliżonej otoczki wypukłej, znajduje się w odległości co najwyżej $(x_{max}-x_{min})/k$ od tej otoczki.

Dowód.

Punkt p znajduje się nie dalej od brzegu przybliżonej otoczki wypukłej niż wynosi szerokość pasa.

Lemat.

Algorytm ma złożoność czasową $O(n+k)$.

Dowód. Przekształcamy podział tak, aby pasy miały całkowitoliczbową szerokość. Rozdzielamy punkty kubełkowo i znajdujemy skrajne punkty w pasach. Budujemy na nich otoczkę stosując zamiatanie.

Algorytm ten działa również w przestrzeni trójwymiarowej w czasie $O(n+k^2 \log k)$.

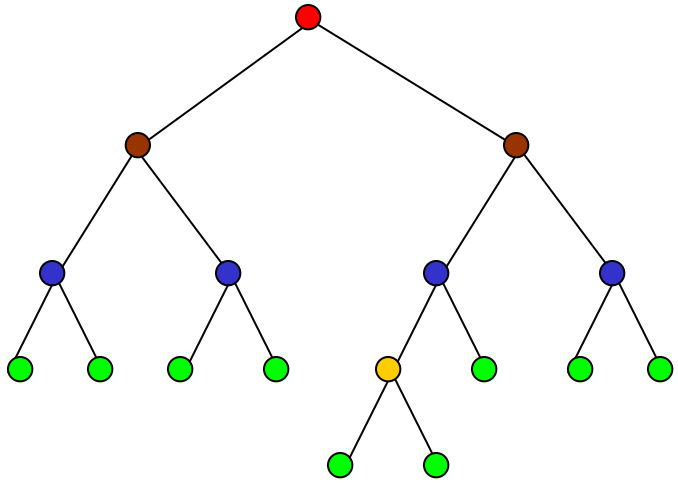
Dynamiczna otoczka wypukła.

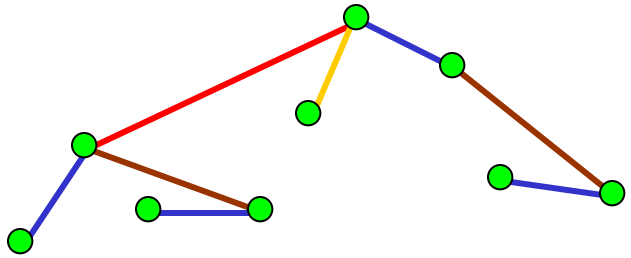
Dla danego ciągu wstawień i usunięć punktów ze zbioru S, chcemy stale utrzymywać aktualną otoczkę wypukłą.

Otoczkę zapamiętujemy w postaci dwóch łańcuchów: górnego i dolnego.

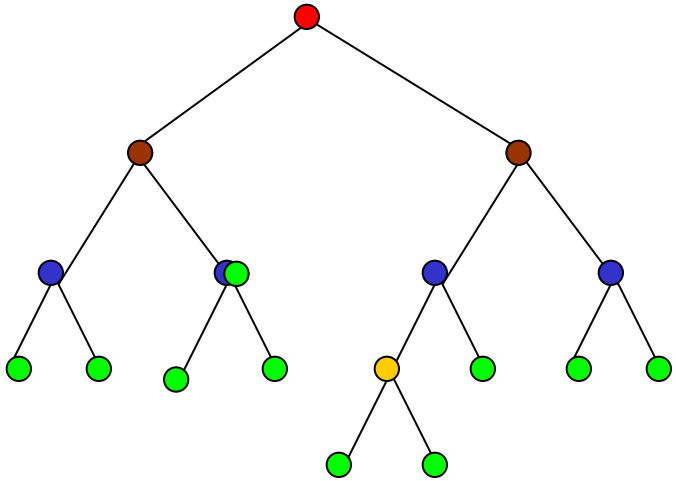
Łańcuchy przechowywane są w zrównoważonym drzewie poszukiwań binarnych.

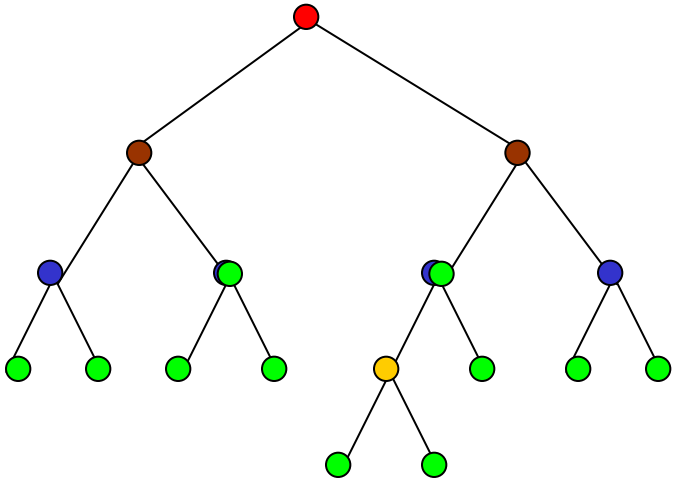
Każdy liść tej struktury odpowiada punktowi w S, zaś każdy węzeł odpowiada mostowi między łańcuchami określanymi przez oba poddrzewa. Zapamiętujemy dwukierunkowe wskaźniki między liśćmi i węzłami, aby móc szybko znajdować mosty i aktualizować strukturę.

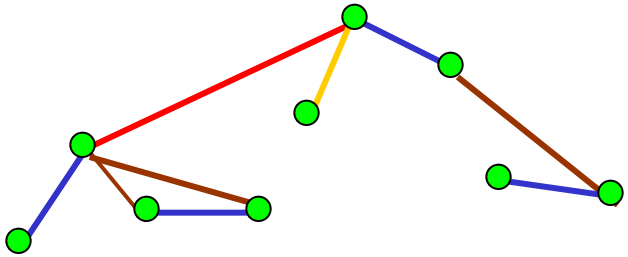


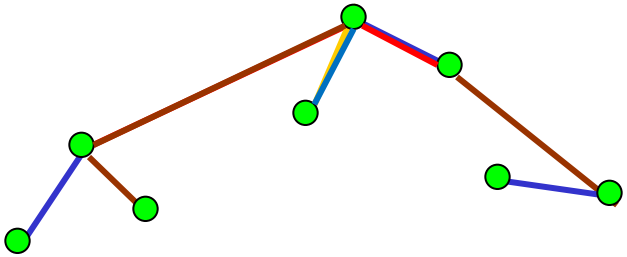


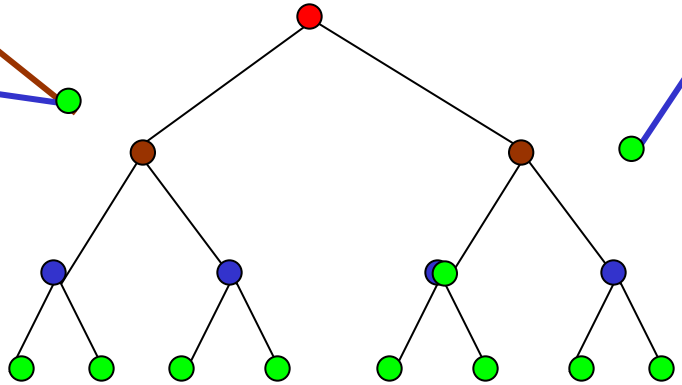
Przykład.

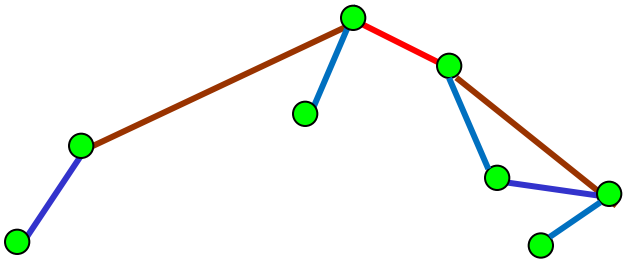












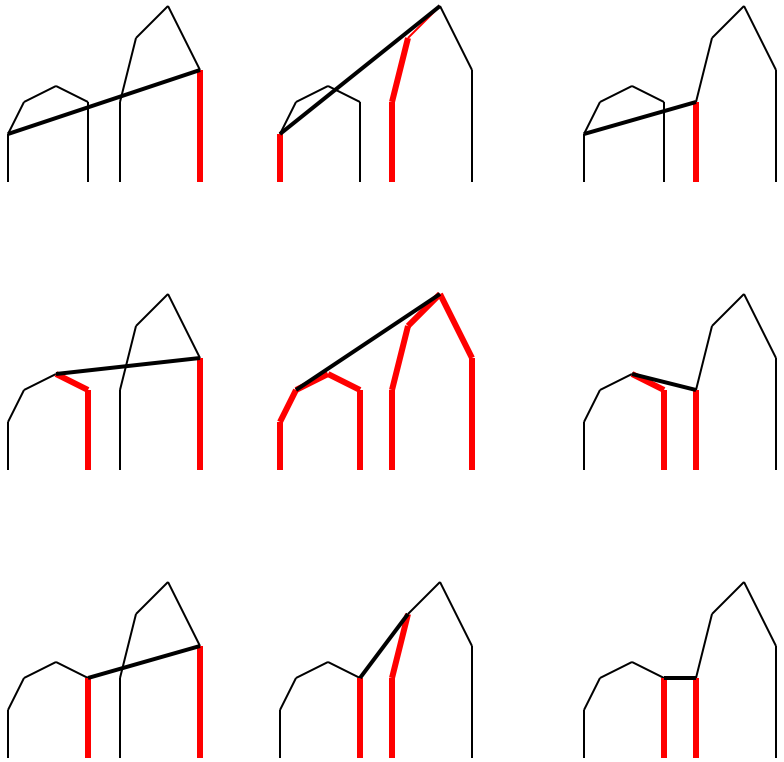
Lemat.

Mostkowanie dwóch rozłącznych łańcuchów wypukłych, zawierających łącznie n punktów, można wykonać w czasie O(log n).

Dowód.

Jest dziewięć przypadków (z dokładnością do symetrii). Dzięki wykorzystaniu struktury zrównoważonego drzewa poszukiwań binarnych, rozwiązanie każdego z nich wymaga logarytmicznego czasu.

Zaznaczone na czerwono fragmenty łańcuchów nie wpływają na wynik mostkowania.



Twierdzenie.

Koszt każdego wstawienia lub usunięcia punktu wynosi $O(\log^2 n)$.

Dowód.

Wstawiając lub usuwając punkt ze struktury znajdujemy ścieżkę od korzenia do odpowiedniego liścia. Następnie aktualizujemy drzewo na tej ścieżce dokonując rotacji w celu jego zrównoważenia oraz znajdując mosty między łańcuchami otoczek punktów odpowiadających liściom poddrzew o korzeniach w potomkach badanego węzła. W każdym węźle poświęcamy na to $O(1)+O(\log n)$ czasu. Zatem aktualizacja struktury wzdłuż całej ścieżki wymaga czasu $O(\log^2 n)$.

Algorytm ten zawdzięczamy Overmarsowi i van Leuwenowi.

Istnieją bardziej skomplikowane algorytmy, które umożliwiają aktualizację otoczki w zamortyzowanym czasie $O(\log n)$.

Otoczka wypukła wielokąta prostego.

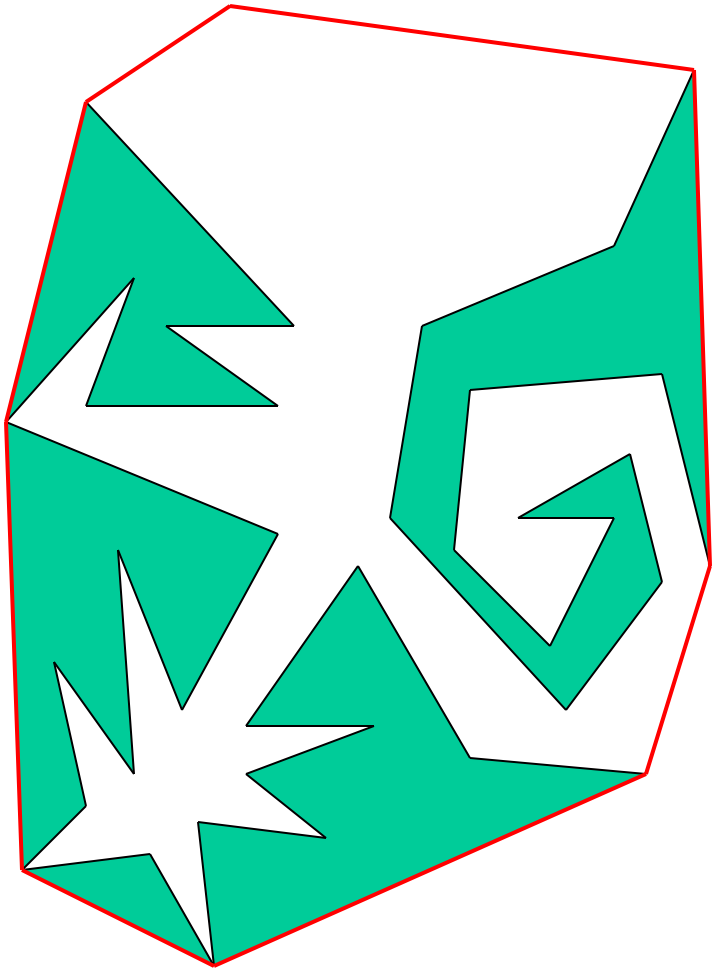
Definicja.

Wielokątem prostym nazywamy obszar ograniczony przez pojedynczy, domknięty, wielokątny łańcuch, który nie przecina się ze sobą.

Definicja.

Zagłębieniem wielokąta prostego nazywamy obszar znajdujący się na zewnątrz wielokąta, ale wewnątrz jego otoczki wypukłej.

Wierzchołki uwypuklenia będziemy przechowywać na stosie Q.



Obliczamy uwypuklenie wielokąta osobno dla jego górnego i dolnego łańcucha.

Analizujemy zależności między

-ostatnim punktem danego łańcucha wielokąta r,

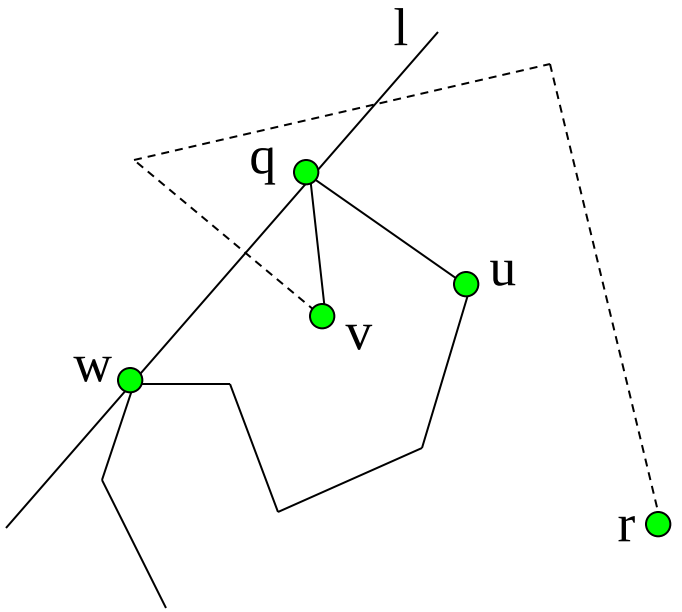
- ostatnim wierzchołkiem aktualnie stworzonego uwypuklenia q,

- jego poprzednikiem u i badanym następnikiem v na brzegu wielokąta oraz

- poprzednikiem w na uwypukleniu.

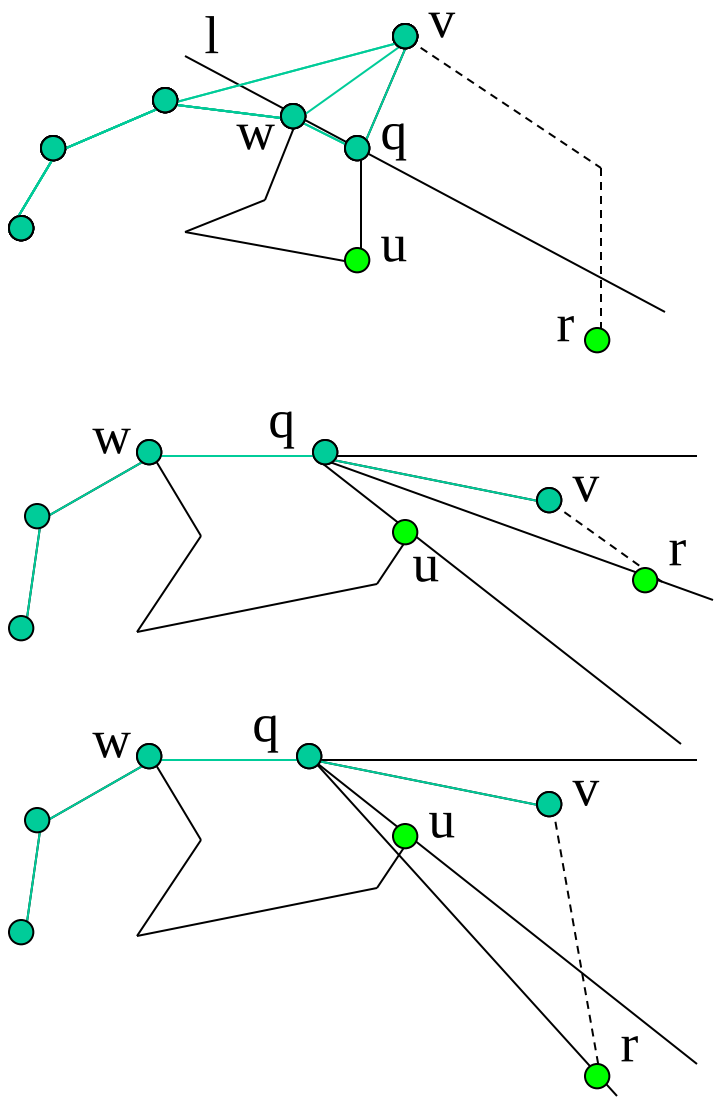
Mamy 4 przypadki.

1. Punkt v znajduje się w zagłębieniu ograniczonym przez prostą l przechodzącą przez w i q – aktualne uwypuklenie nie zmienia się do momentu aż brzeg wielokąta przetnie prostą l.

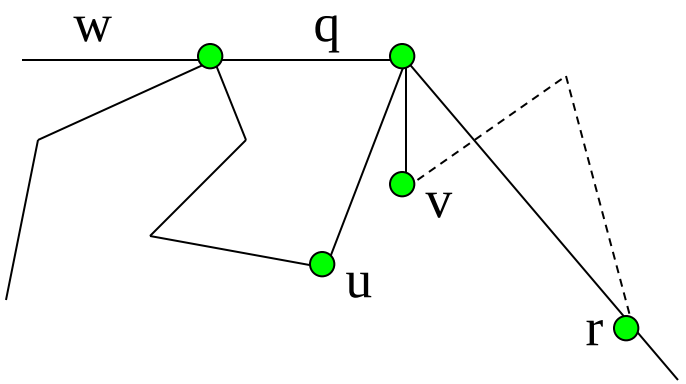


2. Punkt v znajduje się po przeciwnej stronie prostej l przechodzącej przez w i q niż punkt r – punkt v staje się ostatnim wierzchołkiem aktualnego uwypuklenia. Sprawdzamy, czy wierzchołki z Q i v tworzą łamaną wypukłą. Jeśli nie, to usuwamy ze stosu wierzchołki wpadające do wnętrza uwypuklenia i wstawiamy v.

3. Punkt v znajduje się w kącie wyznaczonym przez półprostą o początku w w i przechodzącą przez q oraz bliższą z półprostych o początku w q i przechodzących przez r lub u - punkt v staje się ostatnim wierzchołkiem aktualnego uwypuklenia. Wstawiamy go na stos Q.



4. Punkt v znajduje się w kącie o wierzchołku w punkcie q i ramionach wyznaczonych przez punkty w i r, ale nie należy do zagłębienia, którego wierzchołkiem jest u – aktualne uwypuklenie nie zmienia się do momentu aż brzeg wielokąta przetnie prostą przechodzącą przez q i r.



Niech ciąg (q_i) dla i = 1, 2, ... oznacza wierzchołki tworzonego uwypuklenia, a ciąg (p_i) dla i = 1, 2, ..., n określa wierzchołki górnego (dolnego) łańcucha badanego wielokąta pamiętane w kolejce P. FRONT(P) oznacza pierwszy element P, a POP(P), POP(Q) – usunięcie początku P lub Q. Niech $q_0=(p_{1x},-\infty)$ $(q_0=(p_{1x},+\infty))$.

Algorytm (Lee).

$P:=\{p_1, p_2, \ldots, p_n\}$; $Q \leftarrow q_0$; $u:=q_0$;

$Q \leftarrow p_1$; POP(P); i:=1; j:=1;

while $P \neq \varnothing$ do

v:=FRONT(P);

if $|\angle q_{i-1}q_iv| \leq \pi$ **then**

if $|\angle uq_iv| \leq \pi$ **then**

if $|\angle p_nq_iv| \leq \pi$ **then** $Q \leftarrow v$; $u:=p_{j-1}$;

else

while $|\angle p_nq_i\text{FRONT(P)}| \geq \pi$ **do**

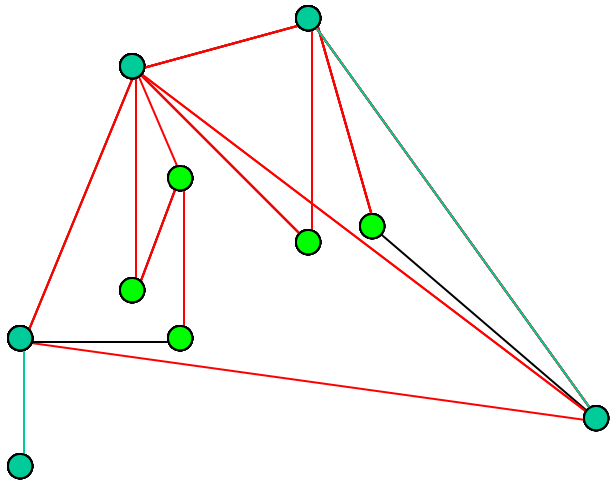
POP(P); j:=j+1

else while $|\angle q_iq_{i-1}\text{FRONT(P)}| \geq \pi$ **do**

POP(P); j:=j+1

else while $|\angle q_{i-1}q_iv| > \pi$ **do** POP(Q);

$Q \leftarrow v$; $u:=p_{j-1}$;



Złożoność algorytmu.

Twierdzenie.

Otoczkę wypukłą wielokąta prostego zawierającego n wierzchołków można zbudować w optymalnym czasie $\Theta(n)$ i pamięci $\Theta(n)$.

Dowód.

Czas potrzebny na analizę jednego wierzchołka wielokąta jest stały w przypadku, gdy nie powoduje to zmian otoczki. W przeciwnym przypadku stały jest czas zamortyzowany (sumarycznie wykonujemy co najwyżej liniową liczbę korekt otoczki, podobnie jak w algorytmie Grahama).

Punkty dominujące.

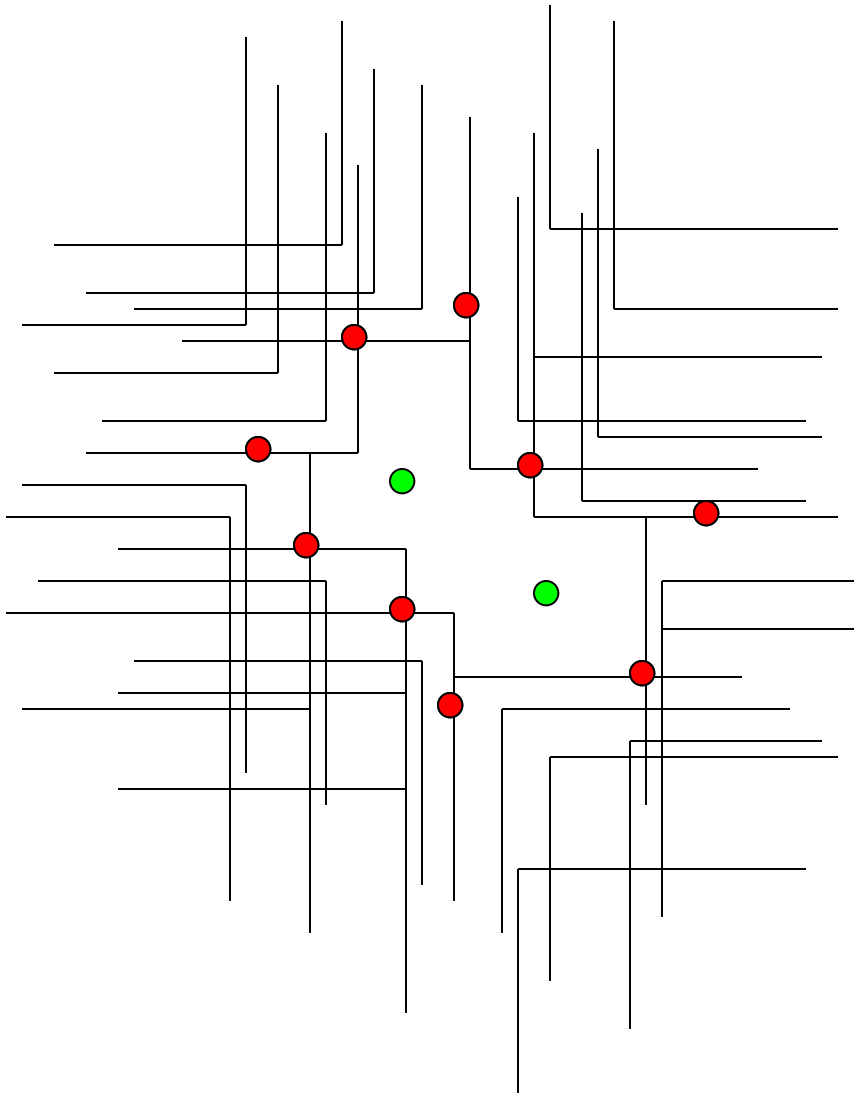
Definicja.

Punkt p_1 dominuje nad punktem p_2 względem współrzędnej x, gdy $x(p_1) > x(p_2)$.

W pierwszej ćwiartce układu współrzędnych R^2 punkt p jest dominujący w zbiorze S, gdy żaden punkt z S nie dominuje nad nim względem obu współrzędnych.

Inaczej: punkt p jest dominujący w zbiorze S, gdy można dosunąć do p nieograniczony prostokąt o bokach równoległych do osi współrzędnych, który nie zawiera w swoim wnętrzu punktów z S.

Podobnie możemy zdefiniować punkty dominujące w innych ćwiartkach i dla wyższych wymiarów.



Algorytm zamiatania.

posortuj zbiór S względem x;

$D = \{p_1\}$; max = min = p_{y1};

for i = 2 ***to*** n ***do***

    if $p_{yi} \geq$ max ***then*** $D = D \cup \{p_i\}$;

        max = p_{yi};

    if $p_{yi} \leq$ min ***then*** $D = D \cup \{p_i\}$;

        min = p_{yi};

$D = D \cup \{p_n\}$; max = min = p_{yn};

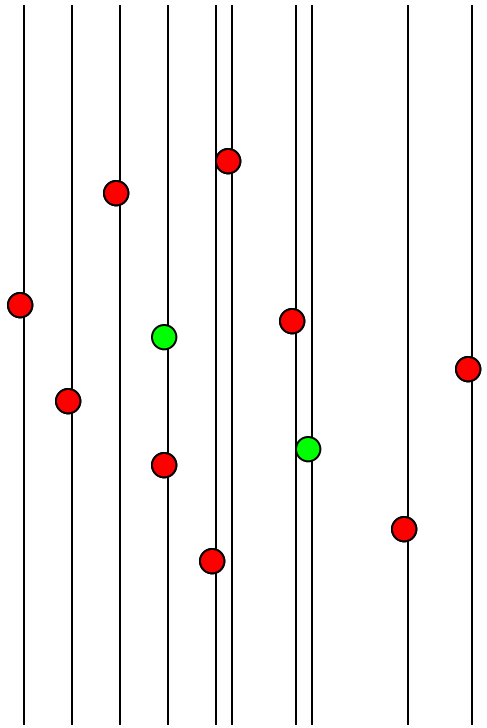
for i = n-1 ***to*** 1 ***do***

    if $p_{yi} \geq$ max ***then*** $D = D \cup \{p_i\}$;

        max = p_{yi};

    if $p_{yi} \leq$ min ***then*** $D = D \cup \{p_i\}$;

        min = p_{yi};

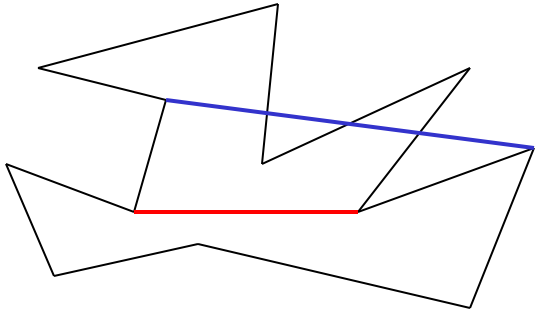


Złożoność algorytmu wynosi O(n log n).

Triangulacja wielokąta przekątnymi.

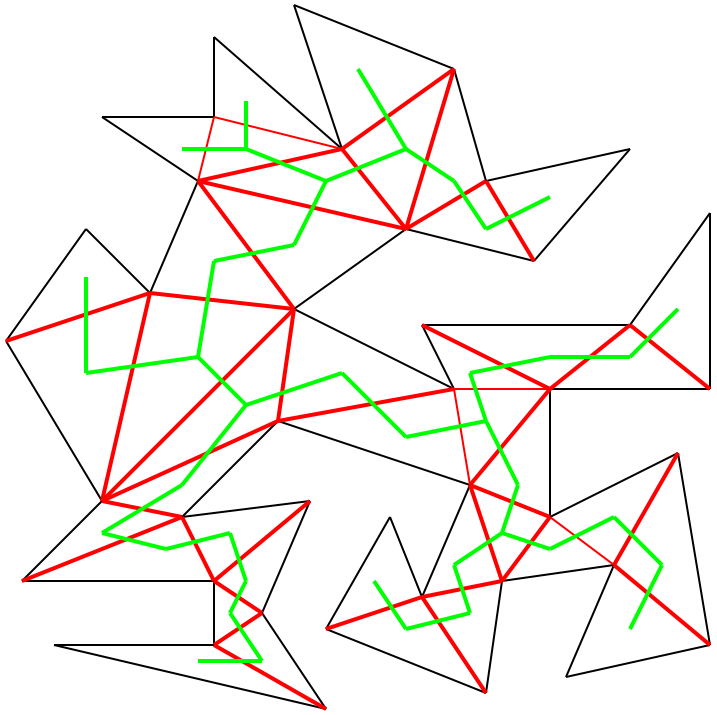
Definicja.

Przekątną wielokąta F nazywamy otwarty odcinek łączący dwa niesąsiednie wierzchołki wielokąta F i leżący wewnątrz F.



Definicja.

Podział wielokąta na trójkąty przez maksymalny zbiór nieprzecinających się przekątnych nazywamy ***triangulacją*** wielokąta.

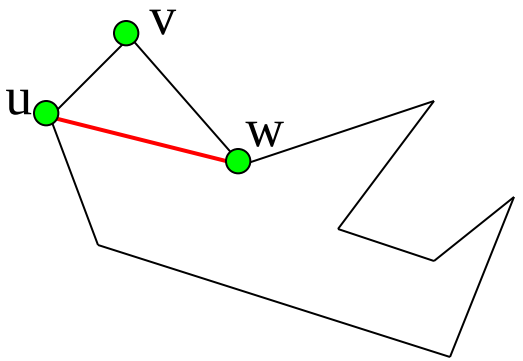


Fakt.

Każda triangulacja jest dualna do drzewa, którego węzły odpowiadają trójkątom a krawędzie - przekątnym.

Lemat.

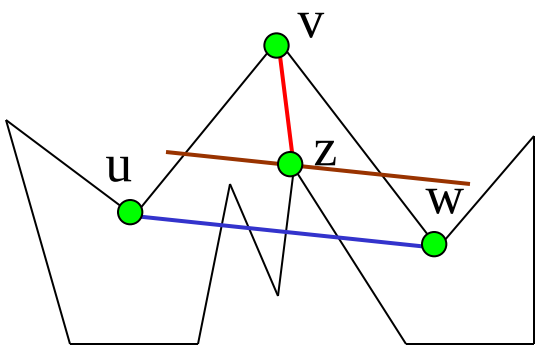
W każdym wielokącie istnieje wierzchołek, w którym kąt ma rozwartość mniejszą niż π.



Twierdzenie.

W każdym wielokącie o $n > 3$ wierzchołkach istnieje przekątna.

Dowód. Konstrukcyjny. Znajdujemy wierzchołek v, przy którym kąt ma rozwartość mniejszą niż π. Badamy trójkąt utworzony przez ten wierzchołek i wierzchołki sąsiednie u i w. Jeśli do tego trójkąta nie należy żaden inny wierzchołek, to odcinek $\overline{uw}$ jest przekątną. W przeciwnym przypadku w trójkącie uvw znajdujemy wierzchołek wielokąta, którego odległość od $\overline{uw}$ jest największa. Wtedy odcinek $\overline{vz}$ jest przekątną wielokąta.



Wniosek.

Powyższa metoda pozwala na znalezienie przekątnej w czasie liniowym względem liczby wierzchołków.

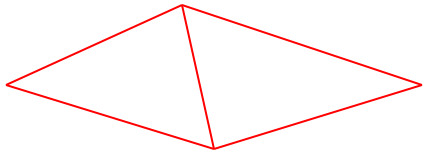
Wniosek.

Przekątna dzieli wielokąt na dwie części. Powtarzając tę procedurę otrzymujemy algorytm triangulacji wielokąta o n wierzchołkach w czasie $O(n^2)$.

Fakt.

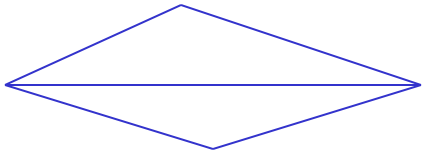
Triangulacja wielokąta może być, ale zwykle nie jest, jednoznaczna.

Optymalna ważona triangulacja wielokątów wypukłych.

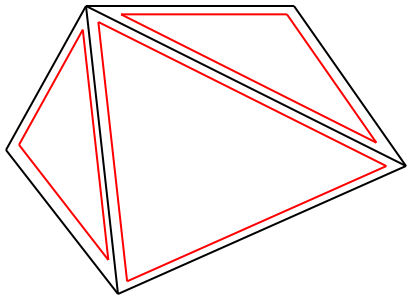


Definicja.

Dla danego wielokąta wypukłego P i funkcji wagowej w( ) zdefiniowanej na trójkątach o krawędziach ze zbioru boków i przekątnych wielokąta P, znajdź triangulację o minimalnej sumie wag trójkątów wchodzących w jej skład.



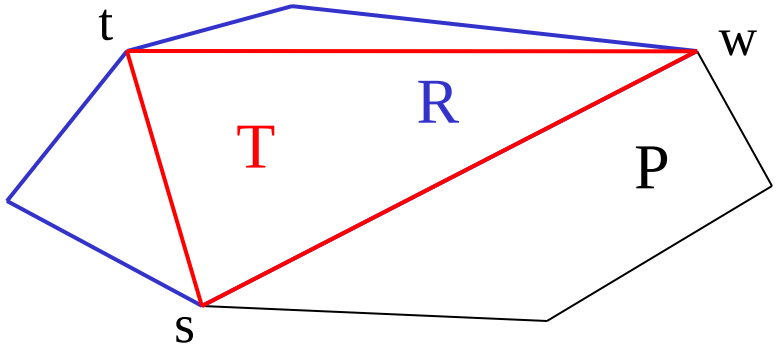
Jedną z naturalnych funkcji wagowych jest suma długości boków trójkąta. Wtedy wagą triangulacji jest suma długości boków wielokąta P oraz podwojona suma długości jego przekątnych. Zatem minimalną wagę ma triangulacja o minimalnej sumie długości przekątnych.



Prezentowany algorytm jest algorytmem dynamicznym tzn. w obliczeniach wykorzystywane są wyniki otrzymane wcześniej.

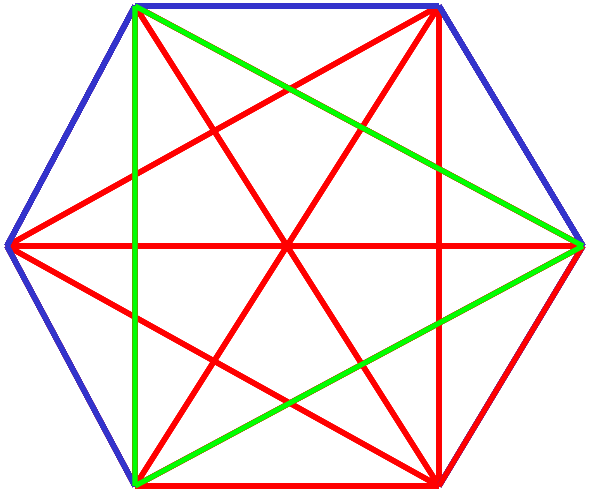
Aby rozwiązać ten problem obliczamy kolejno wartości optymalnych triangulacji dla wielokątów wyznaczanych przez pary, trójki, czwórki.... kolejnych wierzchołków wielokąta P. Niech v_1, ..., v_n będą wierzchołkami wielokąta P, a t[i,j] oznacza aktualnie minimalną wagę triangulacji wielokąta wyznaczanego przez wierzchołki v_i, ... , v_j.

Aby obliczyć optymalną triangulację dla wielokąta R o wierzchołkach v_s, v_{s+1}, ... , v_w, poszukujemy minimalnej wagi triangulacji wyznaczonej przez trójkąt T ($\Delta v_s v_w v_t$, gdzie $t \in \{s+1, ..., w-1\}$) oraz minimalne triangulacje wielokątów powstałych z wielokąta R po wycięciu trójkąta T (wielokąty mogą być zdegenerowane). Zauważmy, że optymalne triangulacje takich wielokątów są obliczane we wcześniejszej fazie działania algorytmu.



Algorytm dynamiczny.

```
for i := 1 to n-1 do t[i,i+1] := 0;
for m := 3 to n do
    for i := 1 to n-m+1 do
        j := i+m-1;  t[i,j] := ∞;
        for k := i+1 to j-1 do
            q := t[i,k]+t[k,j]+w(Δv_i v_k v_j);
            if q < t[i,j] then t[i,j] := q;
```



Lemat.

Algorytm ma złożoność

$$\sum_{m=3}^{n} \sum_{i=1}^{n-m+1} \sum_{k=i+1}^{i+m-2} O(1) = \Theta(n^3) \quad .$$

Podział wielokąta prostego na wielokąty wypukłe.

Lemat.

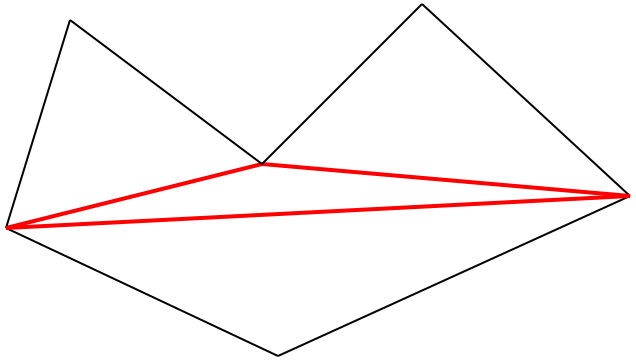
Niech Φ oznacza minimalną liczbę wielokątów wypukłych, na które można podzielić dany wielokąt odcinkami (niekoniecznie przekątnymi). Niech r będzie liczbą kątów wewnętrznych wielokąta o rozwartości większej niż π. Wtedy: $\lceil r/2 \rceil + 1 \leq \Phi \leq r + 1$.

Definicja.

Dla danego podziału wielokąta przekątnymi, istotną przekątną nazywamy taką, której usunięcie powoduje powstanie wielokąta niewypukłego (mającego kąt wewnętrzny o rozwartości większej niż π).

Fakt.

Istotne przekątne nie są wyznaczone jednoznacznie.



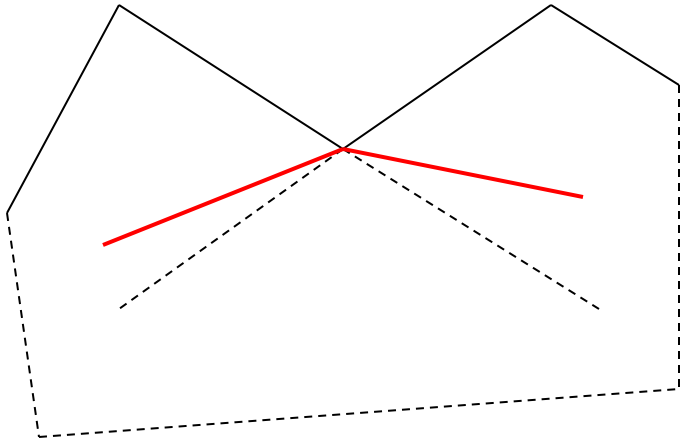
Lemat.

Wierzchołek kąta wewnętrznego o rozwartości większej niż π jest końcem co najwyżej dwóch przekątnych istotnych.

Dowód.

Proste zawierające ramiona kąta wyznaczają dwie półpłaszczyzny.

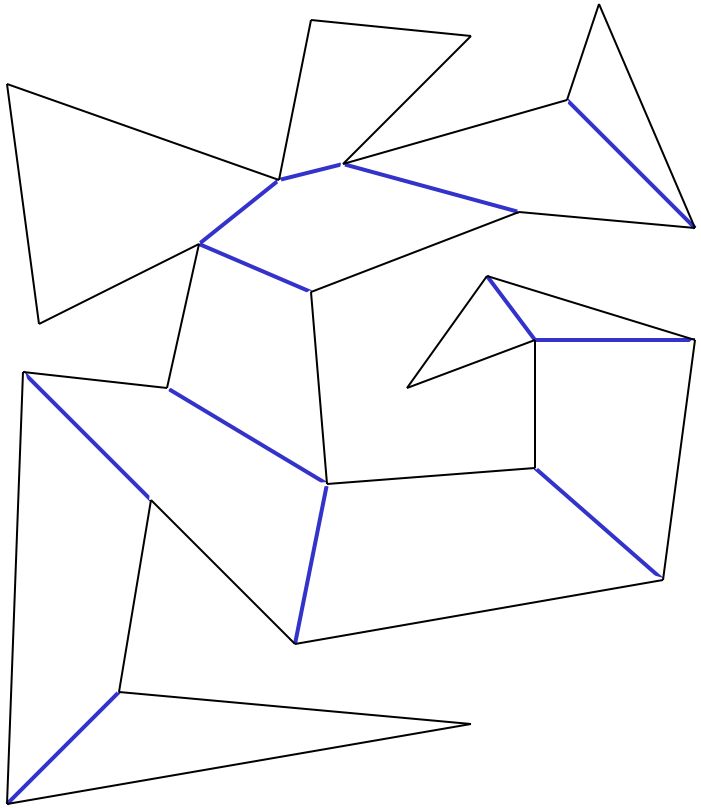
Do każdej z nich może należeć co najwyżej jedna przekątna istotna.



Algorytm Hertela i Mehlhorna.

znajdź dowolną triangulację wielokąta;

usuń wszystkie przekątne, które nie są istotne;



Nieistotne przekątne można wyeliminować w czasie O(n) (w stałym czasie sprawdzamy, czy po usunięciu danej przekątnej, przy którymś z wierzchołków powstanie kąt większy od π) .

Twierdzenie.

Liczba wielokątów wypukłych otrzymanych z pomocą algorytmu Hertela-Mehlhorna jest co najwyżej czterokrotnie większa niż minimalna liczba takich wielokątów.

Dowód.

Algorytm Hertela-Mehlhorna tworzy co najwyżej $2r+1$ wielokątów.

Na podstawie poprzednich lematów mamy: $2r+1 \leq 2r+4 \leq 4(\lceil r/2 \rceil+1) \leq 4\Phi$.

Optymalny podział wielokąta prostego na wielokąty wypukłe można znaleźć w czasie $O(n+r^3)$ (Chazelle-Dobkin), gdzie r jest liczbą kątów wewnętrznych o rozwartości większej niż π).

Dziękuję za uwagę.

Ćwiczenia 2.

1. Udowodnij, że algorytm znajdywania przybliżonej otoczki wypukłej metodą podziału płaszczyzny na pasy działa w przestrzeni trójwymiarowej w czasie $O(n+k^2 \log k)$.
2. Podaj przykłady n-kątów, których triangulacja jest lub nie jest jednoznaczna.
3. Niech $S=\{p_1,p_2, \ldots , p_n\}$ będzie zbiorem punktów w R^3 takim, że $x(p_1) < x(p_2) < \ldots < x(p_n)$. Wykaż, że mimo znajomości porządku S w kierunku x znalezienie otoczki wypukłej S i tak wymaga czasu $\Omega(n \log n)$.
4. Dla danych n prostych na płaszczyźnie stwórz algorytm znajdujący otoczkę wypukłą punktów wyznaczonych przez przecięcia tych prostych działający w czasie subkwadratowym (np. $O(n \log n)$).
5. Dla danego zbioru S zawierającego n punktów na płaszczyźnie stwórz w czasie $O(n \log n)$ strukturę danych rozmiaru $O(n)$, która umożliwi sprawdzenie w czasie $O(\log n)$, czy dany punkt q jest dominujący w zbiorze $S \cup \{q\}$.

6. Otoczką dominacji zbioru S nazywamy zbiór punktów dominujących w S. Głębokość dominacji punktu p w S, to liczba otoczek dominacji, które trzeba odrzucić przed odrzuceniem p. Stwórz algorytm wyznaczający głębokość dominacji każdego punktu z S w czasie O(n log n).
7. Udowodnij lub zaprzecz: każde drzewo binarne jest grafem dualnym triangulacji jakiegoś wielokąta.
8. Jak wiele triangulacji ma n-kąt wypukły ?
9. Dany jest nieuporządkowany zbiór przekątnych tworzących triangulację wielokąta (wierzchołki są etykietowane kolejnymi liczbami naturalnymi). Stwórz algorytm budujący dualne drzewo triangulacji w czasie O(n).
10. Niech Φ oznacza minimalną liczbę wielokątów wypukłych, na które można podzielić dany wielokąt (niekoniecznie przekątnymi). Niech r będzie liczbą kątów wewnętrznych wielokąta o rozwartości większej niż π. Wtedy: $\lceil r/2 \rceil + 1 \leq \Phi \leq r + 1$.