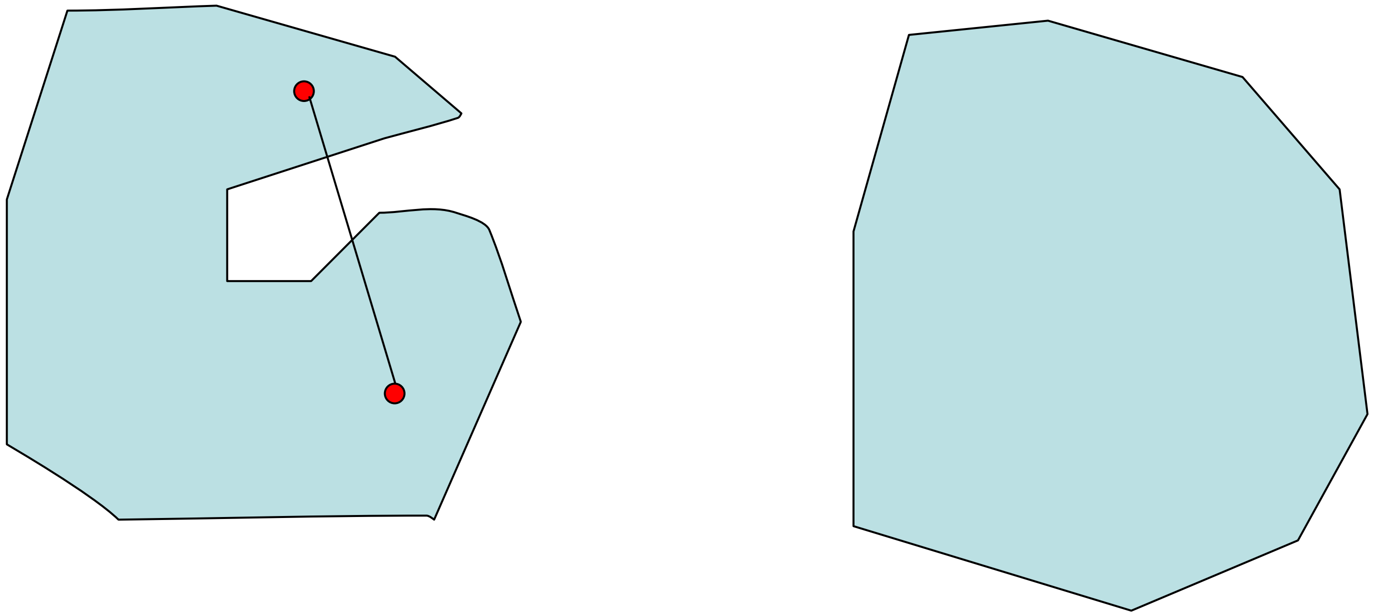
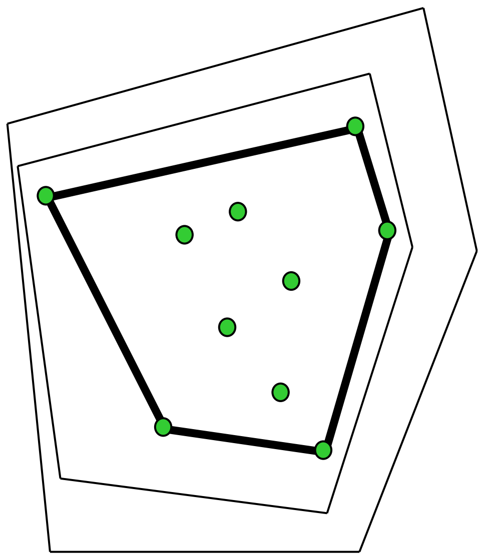
OTOCZKA WYPUKŁA

Wypukłość zbioru



Otoczka wypukła

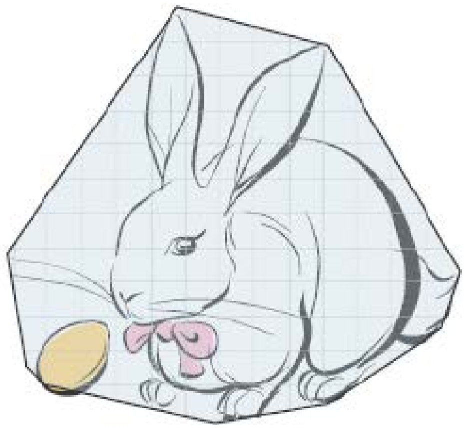
Otoczką wypukłą *CH*(*S*) dowolnego niepustego zbioru punktów ***S*** nazywamy najmniejszy zbiór wypukły zawierający *S*.



Otoczka wypukła punktów na płaszczyźnie jest najmniejszym wielokątem wypukłym zawierającym S

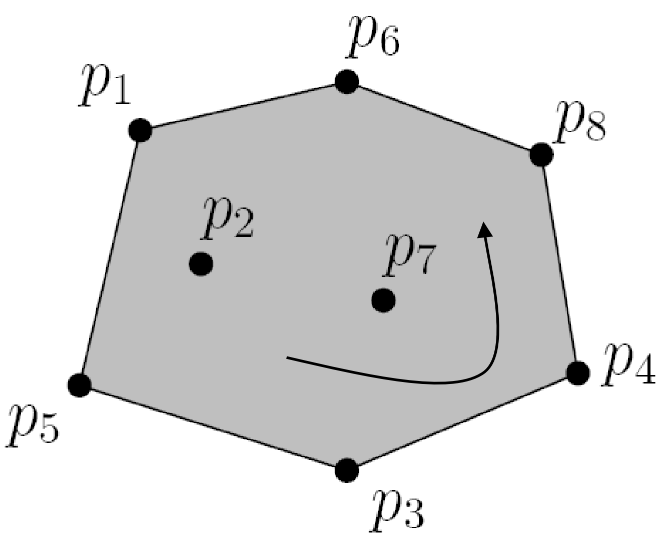
Zastosowanie

- Uproszczenie kształtu
- Planowanie ruchu
- Detekcja kolizji …

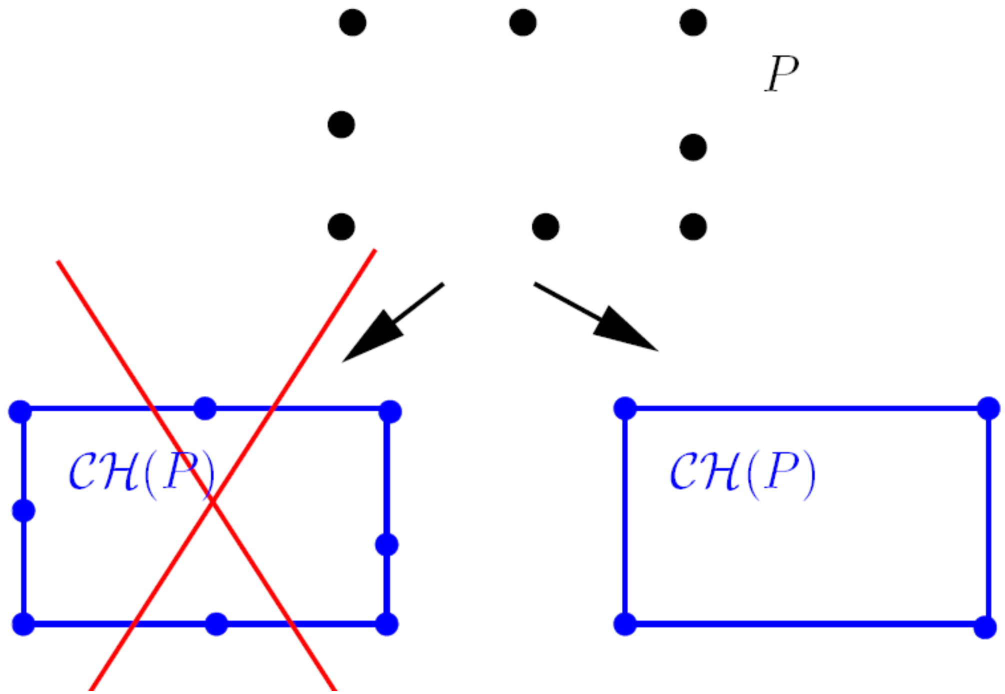


Co to znaczy obliczyć otoczkę wypukłą ?

Dla danego zbioru *S* punktów na płaszczyźnie
wyznaczyć listę punktów z *S*
będących wierzchołkami otoczki,
wymienione w porządku
przeciwnym (*zgodnym*) do ruchu wskazówek zegara



$$\mathcal{L} = (p_3, p_4, p_8, p_6, p_1, p_5)$$



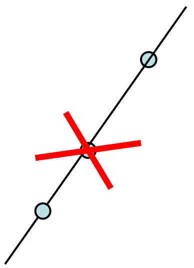
Do otoczki należą jedynie
punkty „ekstremalne”,
bez punktów współliniowych „wewnętrznych”

Jak znaleźć wierzchołki otoczki wypukłej?

Kryterium negatywne

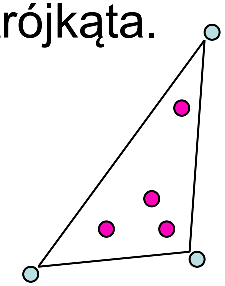
Punkt p **nie** jest wierzchołkiem otoczki wypukłej wtw gdy leży wewnątrz trójkąta o wierzchołkach ze zbioru S różnych od p lub gdy leży na odcinku łączącym dwa różne od p punkty zbioru S.

Metoda:



Dla każdej trójki punktów p_1, p_2, p_3 ze zbioru S :

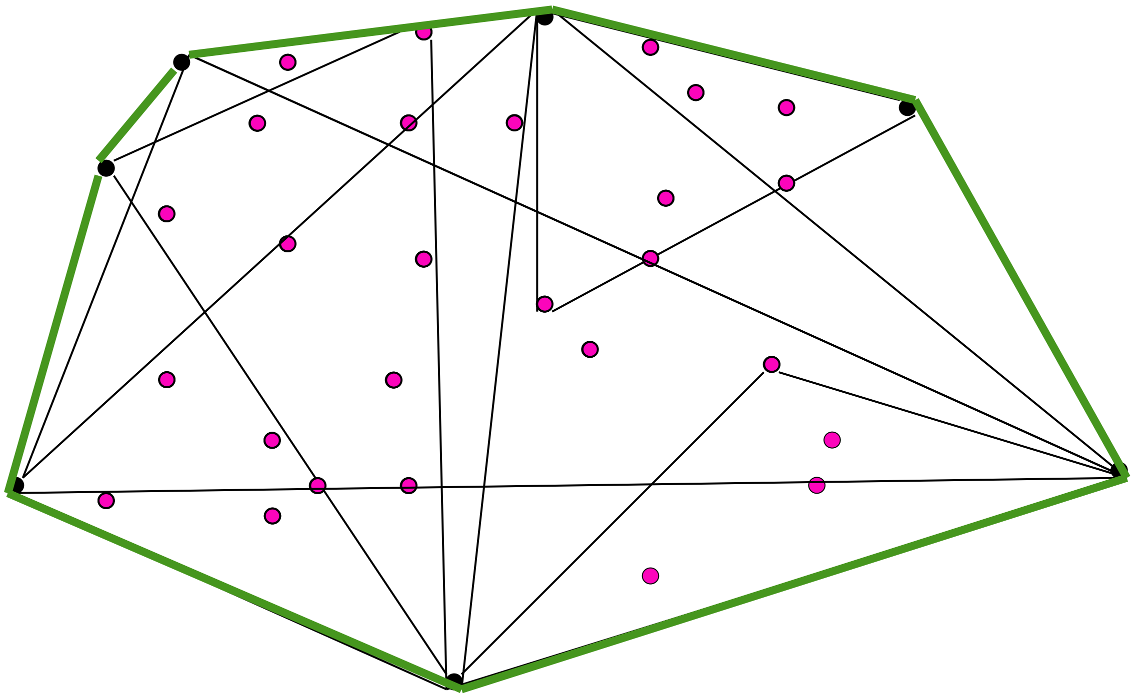
1. jeśli są one współliniowe i p jest punktem środkowym tej trójki, to ze zbioru S usuwamy punkt p,
2. jeśli nie są współliniowe, to oznaczmy przez d trójkąt, którego wierzchołkami są te punkty i usuwamy ze zbioru S wszystkie punkty leżące wewnątrz tego trójkąta.



Zbiór punktów, które pozostaną w *S*, jest zbiorem wierzchołków szukanej otoczki wypukłej.

Wymaga uporządkowania

Przykład Konstrukcja otoczki metodą trójkątów



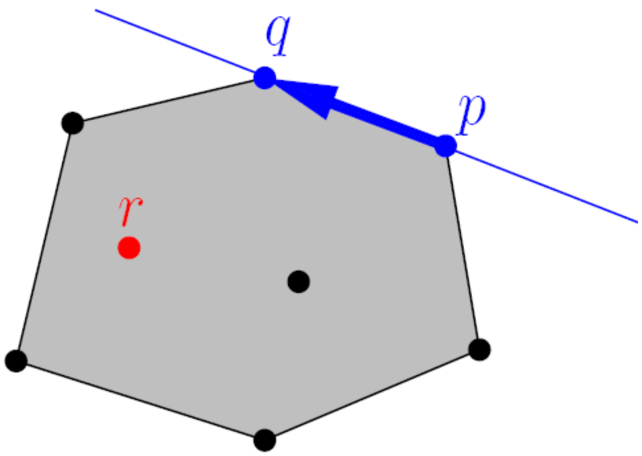
Niech $|S|=n$.

Jest $\binom{n}{3}$ różnych trójek punktów

liczba iteracji pętli zewnętrznej – $O(n^3)$.

Koszt pętli wewnętrznej – $O(n)$. Razem $O(n^4)$.

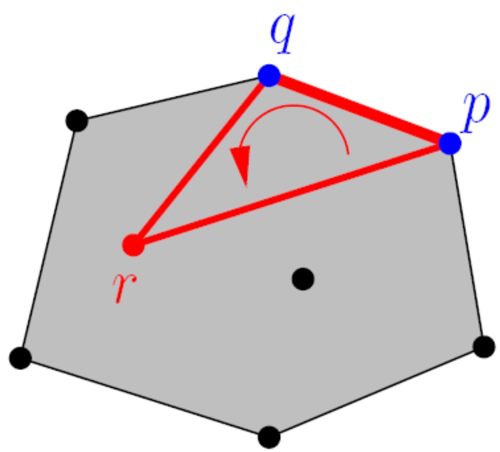
Skierowana krawędź (*p*,*q*) należy do otoczki, gdy



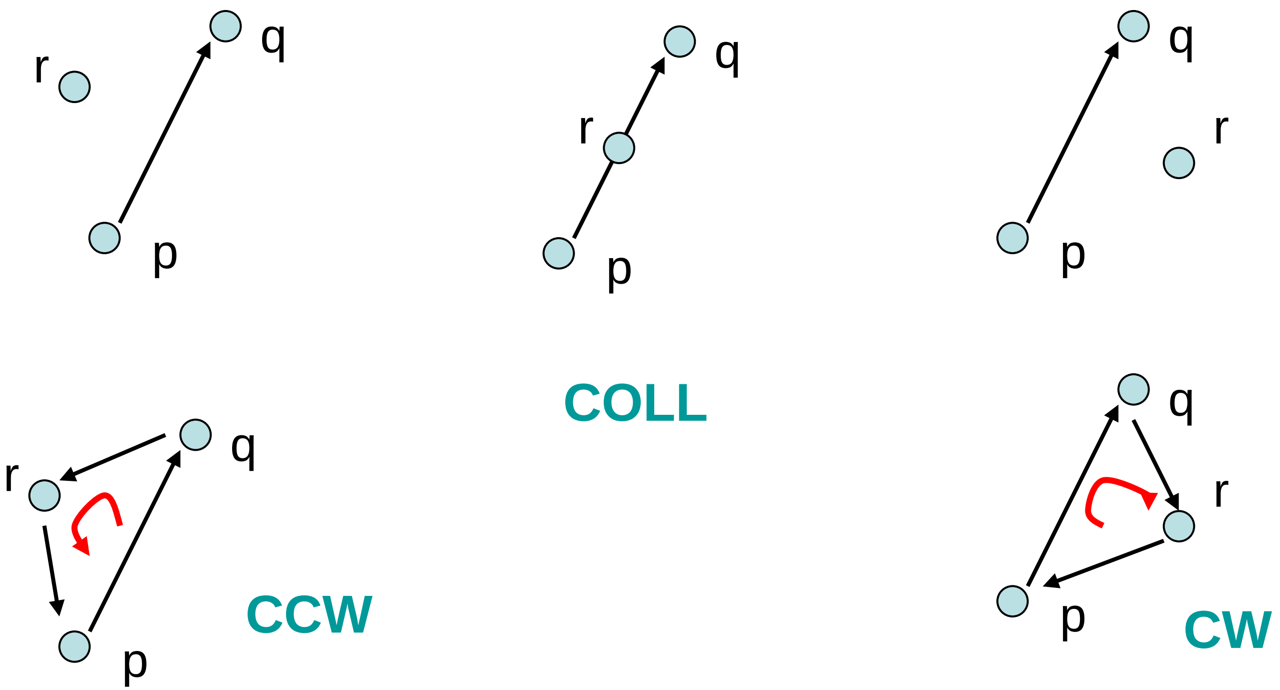
$$\forall r \in S \setminus (p, q)$$

r leży po stronie ***lewej*** prostej *pq*

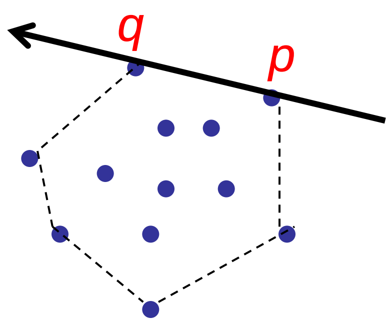
trójkąt (*p*, *q*, *r*) jest zorientowany przeciwnie do wskazówek zegara



$$
\text{orient}(p, q, r)
\begin{cases}
> 0 & \textbf{CCW} \\
= 0 & \textbf{COLL} \\
< 0 & \textbf{CW}
\end{cases}
$$

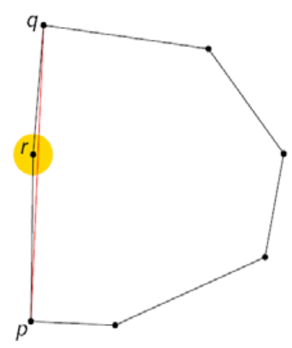


Krawędzie skrajne



E←∅
for każda para (p,q), $p \neq q$
    ważny←true
    for wszystkie punkty $r \in S$, $r \neq p$, $r \neq q$
        if r leży po prawej stronie (p,q) then ważny←false
    if ważny dodaj krawędź skierowaną (p,q) do E
Uporządkuj wierzchołki otoczki

Złożoność rzędu $O(n^3)$

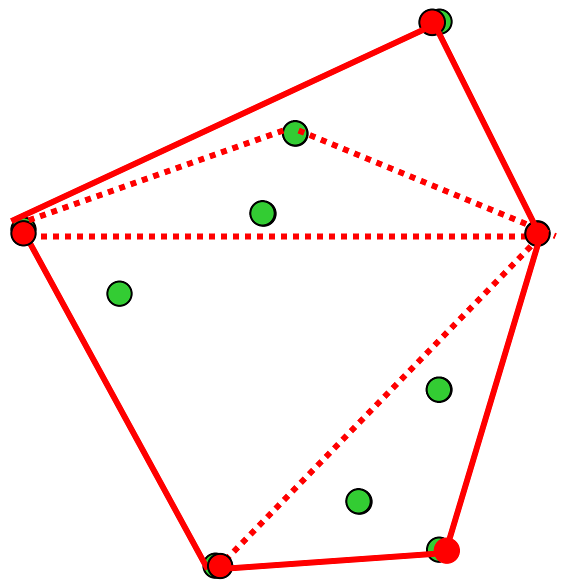


Istotna rola sposobu klasyfikacji punktów (lewa, prawa strona)

Uwaga na przypadki zdegenerowane (np. współliniowość punktów)

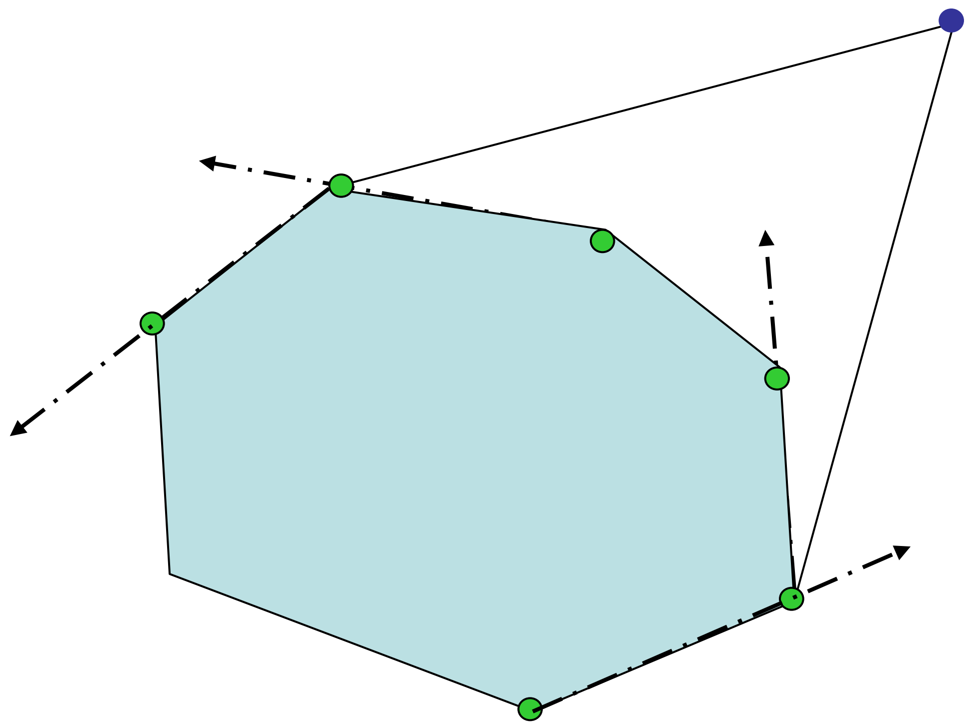
Algorytm przyrostowy

weź pierwsze trzy punkty ze zbioru S i stwórz z nich otoczkę CH(*S*);
for *i* = 4 ***to*** n **do**
    if *i*-ty punkt z *S* nie należy do wnętrza CH(*S*) **then**
        znajdź styczne do CH(*S*) przechodzące przez ten punkt;
        aktualizuj CH(*S*);

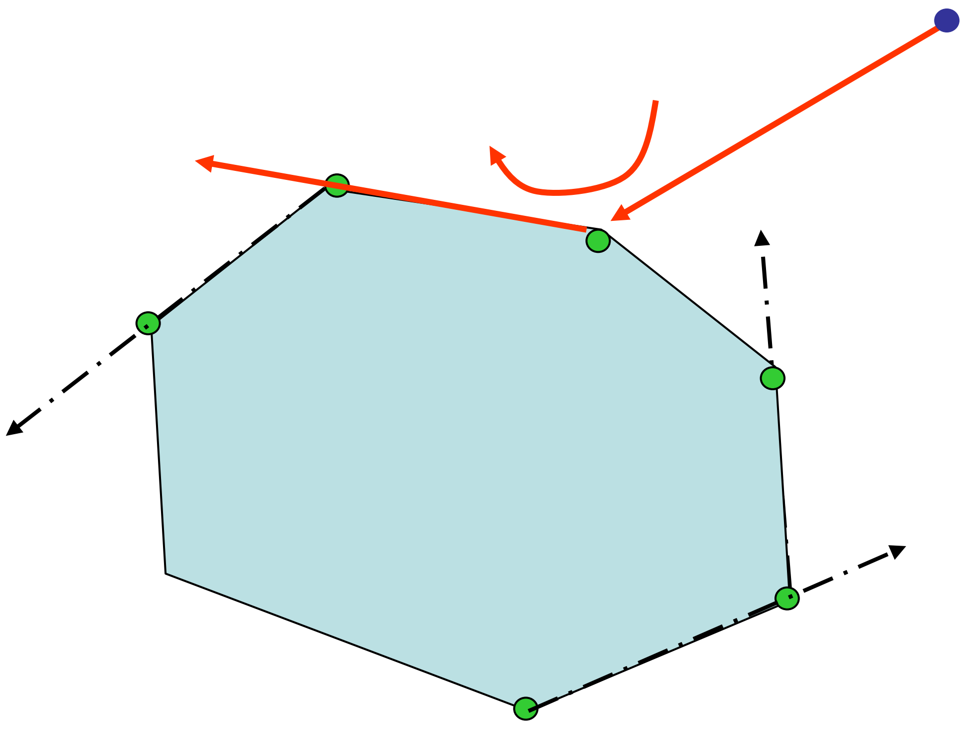


Złożoność rzędu O(*n* log *n*)

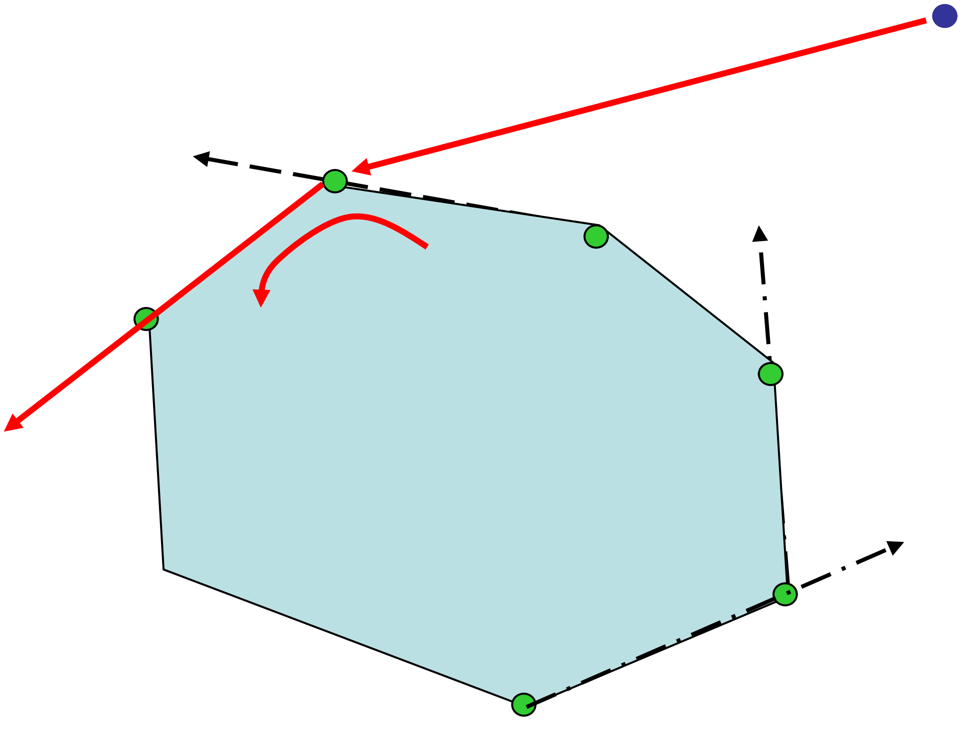
Dwie linie styczne do zbioru



Linia styczna do zbioru



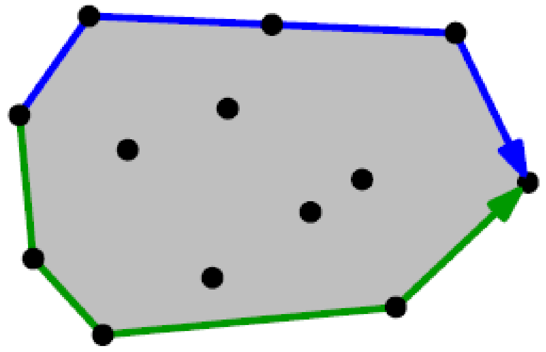
Linia styczna do zbioru



Górna i dolna otoczka

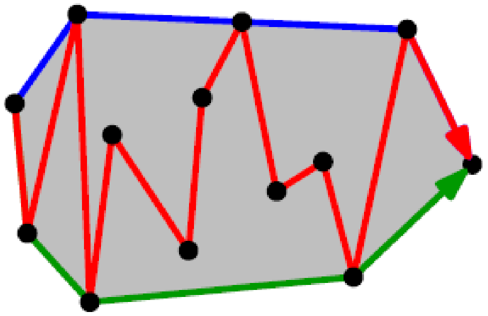
Rozdzielamy obliczenia na dwa:
obliczenie *górnej i dolnej* otoczki

górna otoczka



dolna otoczka

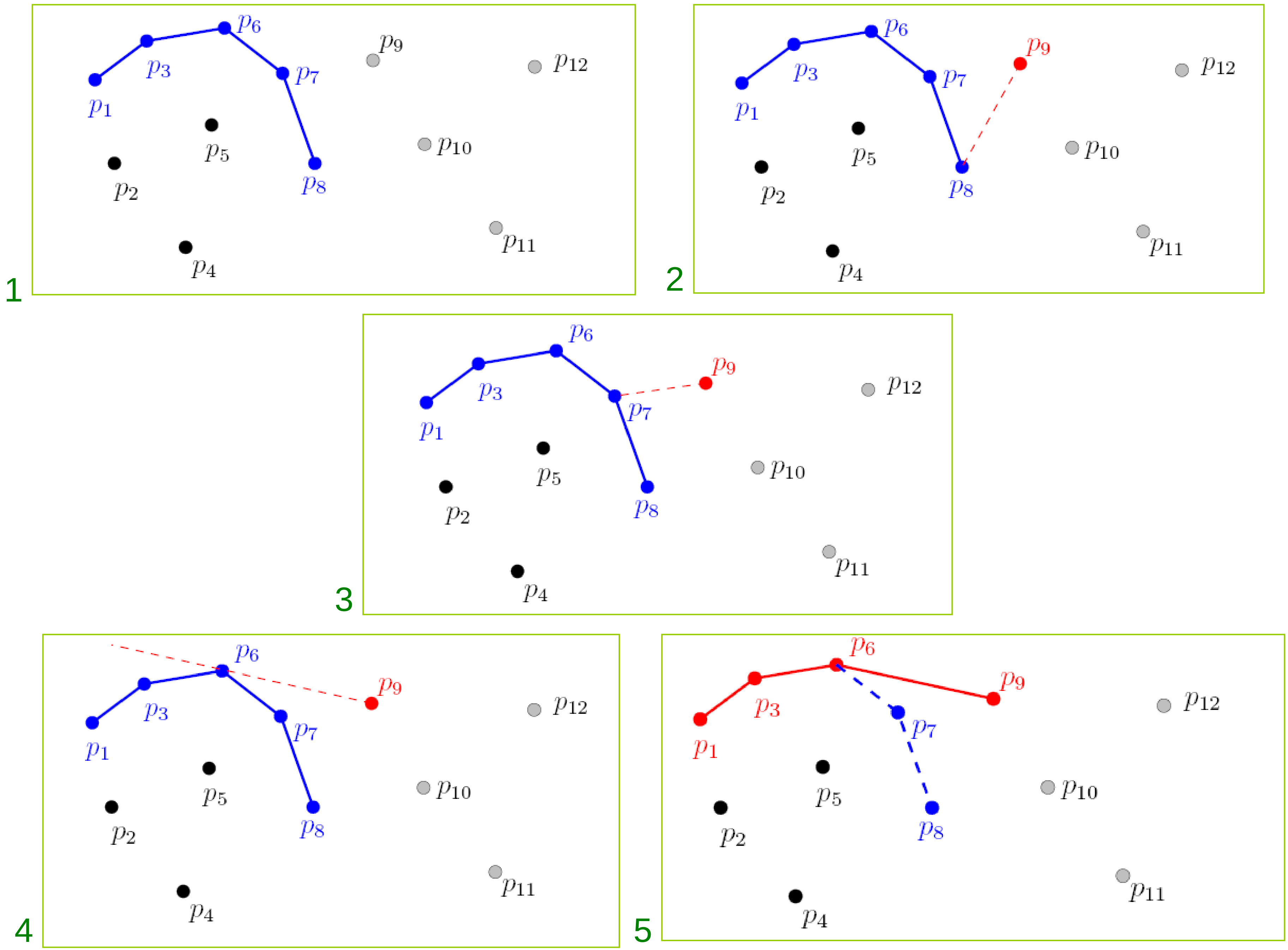
Górna i dolna otoczka



- Sortujemy punkty względem współrzędnej *x*

- Jeśli istnieją punkty o tej samej współrzędnej *x*, porządek leksykograficzny – tzn. sortujemy też względem współrzędnej *y*

- Gdy poruszamy się wzdłuż brzegu wielokąta zgodnie z ruchem wskazówek zegara, wykonujemy skręt – w wielokącie wypukłym musi być to skręt w prawo.



1
p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8 p_9 p_{10} p_{11} p_{12}
2
p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8 p_9 p_{10} p_{11} p_{12}
3
p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8 p_9 p_{10} p_{11} p_{12}
4
p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8 p_9 p_{10} p_{11} p_{12}
5
p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8 p_9 p_{10} p_{11} p_{12}

Wyznaczanie górnej otoczki

Uporządkuj punkty w porządku leksykograficznym tworząc ciąg $p_1, \ldots, p_n$

$L_{\text{górna}} \leftarrow \{p_1, p_2\}$

for $i = 3$ to n

dodaj p_i do $L_{\text{górna}}$

while $L_{\text{górna}}$ zawiera więcej niż dwa punkty

and trzy ostatnie punkty nie tworzą skrętu w prawo

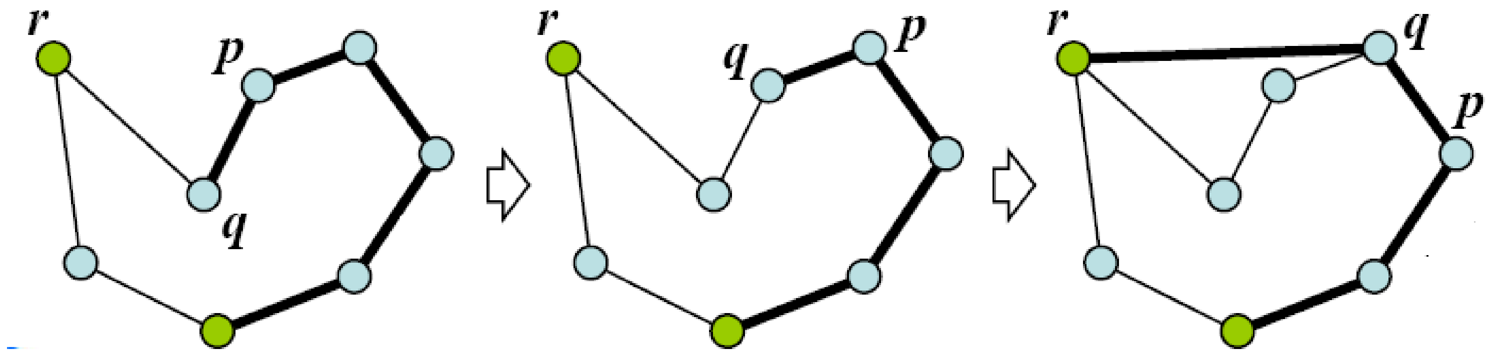
usuń środkowy z ostatnich trzech punktów $L_{\text{górna}}$

Analogicznie wyznaczanie dolnej otoczki + lista wynikowa

Złożoność rzędu O(*n* log *n*)

Algorytm Grahama

Systematyczne usuwanie wierzchołków *wklęsłych*



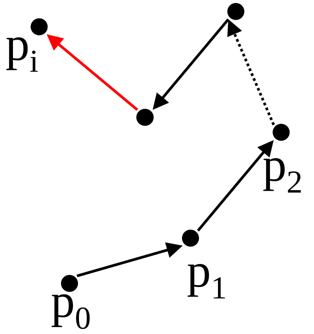
Algorytm Grahama

1. W zbiorze S wybieramy punkt p_0 o najmniejszej współrzędnej y. Jeżeli jest kilka takich punktów, to wybieramy ten z nich, który ma najmniejszą współrzędną x.

2. Sortujemy pozostałe punkty ze względu na kąt, jaki tworzy wektor (p_0,p) z dodatnim kierunkiem osi x. Jeśli kilka punktów tworzy ten sam kąt, usuwamy wszystkie z wyjątkiem najbardziej oddalonego od p_0.
   Niech uzyskanym ciągiem będzie $p_1, p_2,...p_m$.

3. Do początkowo pustego stosu s wkładamy punkty p_0, p_1, p_2.
   t – indeks stosu; i $\leftarrow 3$

4. **while** i < *m* **do**
   if p_i leży na lewo od prostej (p_{t-1}, p_t))
   then push(p_i), $i \leftarrow i+1$
   else pop(s)



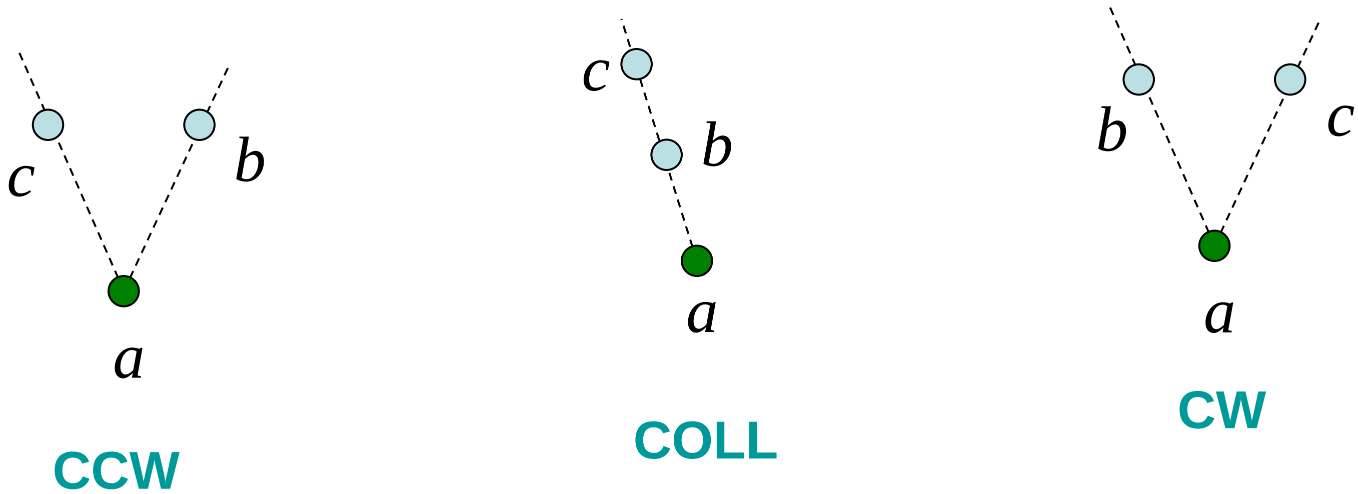
Algorytm Grahama

Sortowanie z wykorzystaniem funkcji ***orient***

$$b < c \Leftrightarrow orient(a,b,c) > 0$$

$$b = c \Leftrightarrow orient(a,b,c) = 0$$

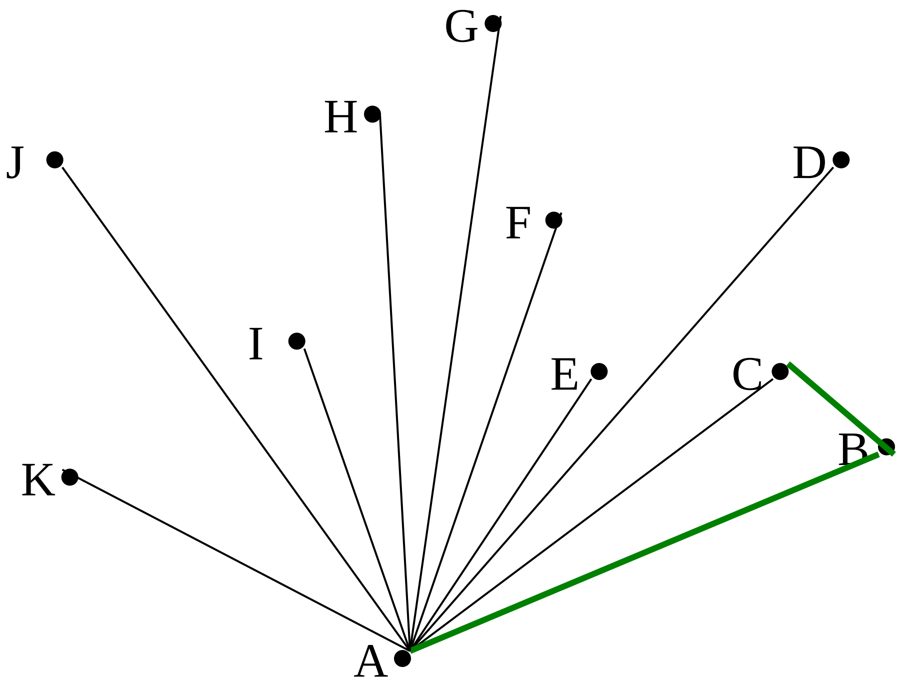
$$b > c \Leftrightarrow orient(a,b,c) < 0$$



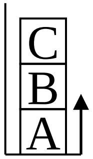
Przykład

Dany zbiór punktów S:

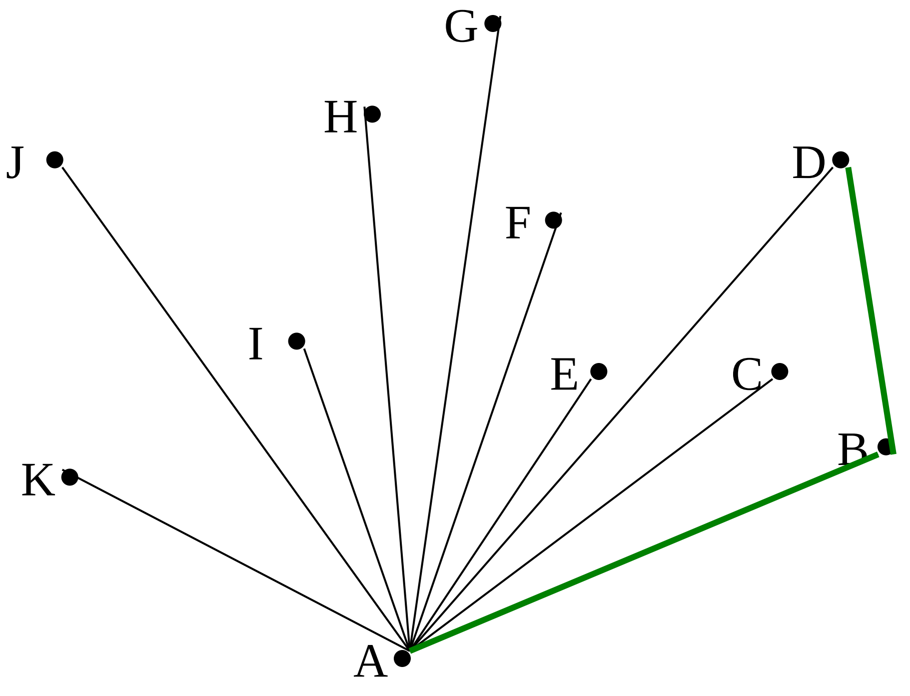
Po posortowaniu:
ABCDEFGHIJK



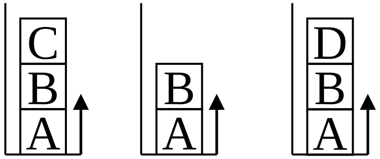
stos



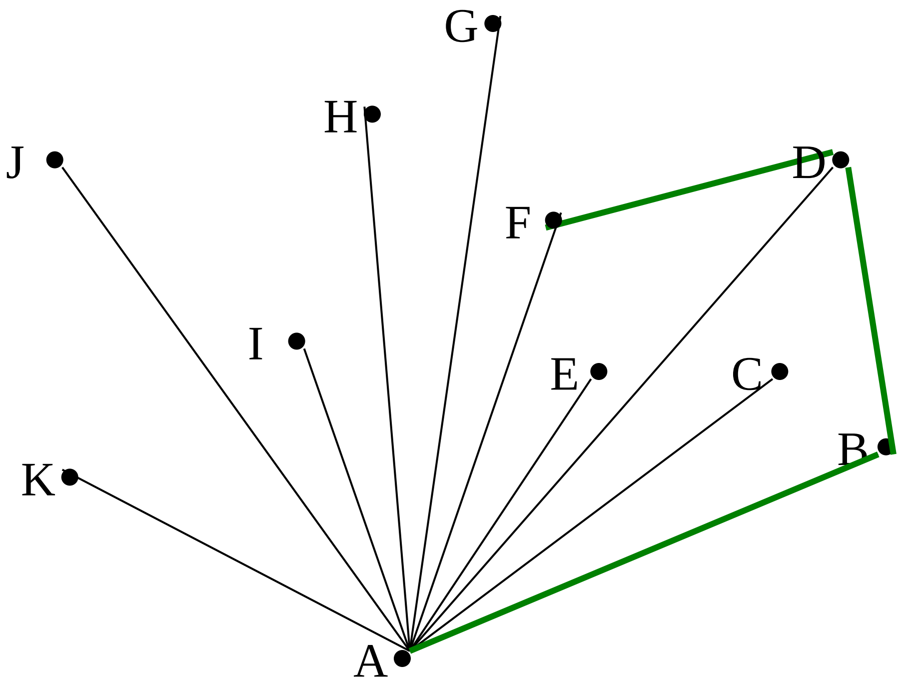
Przykład



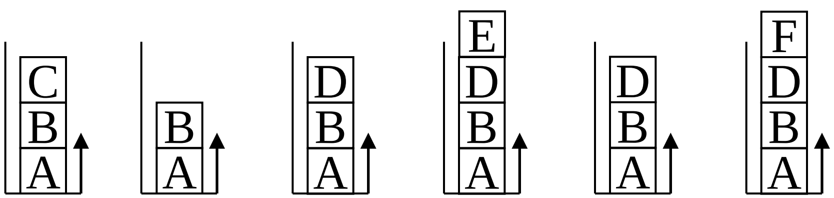
stos



Przykład

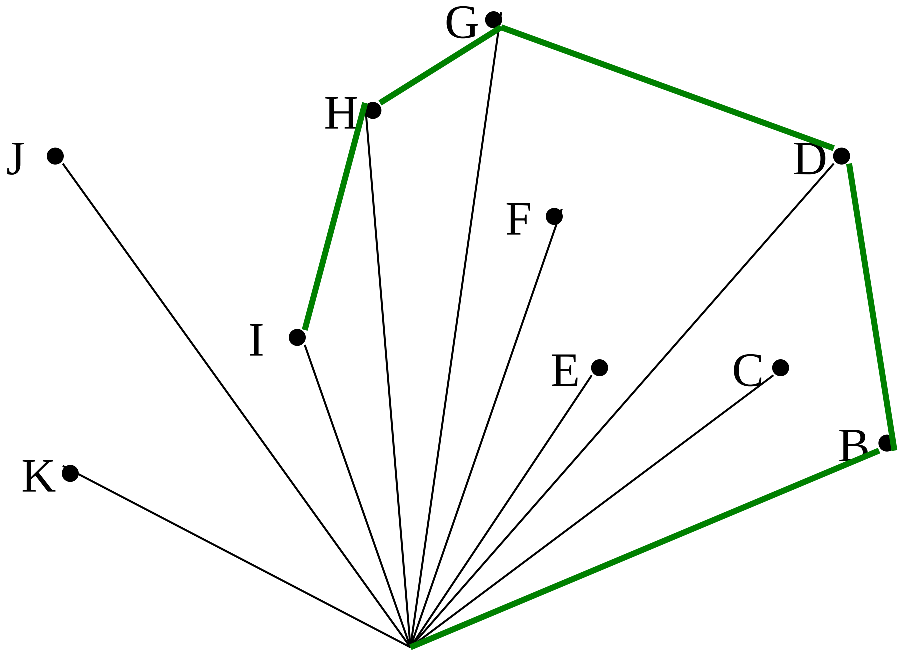


stos

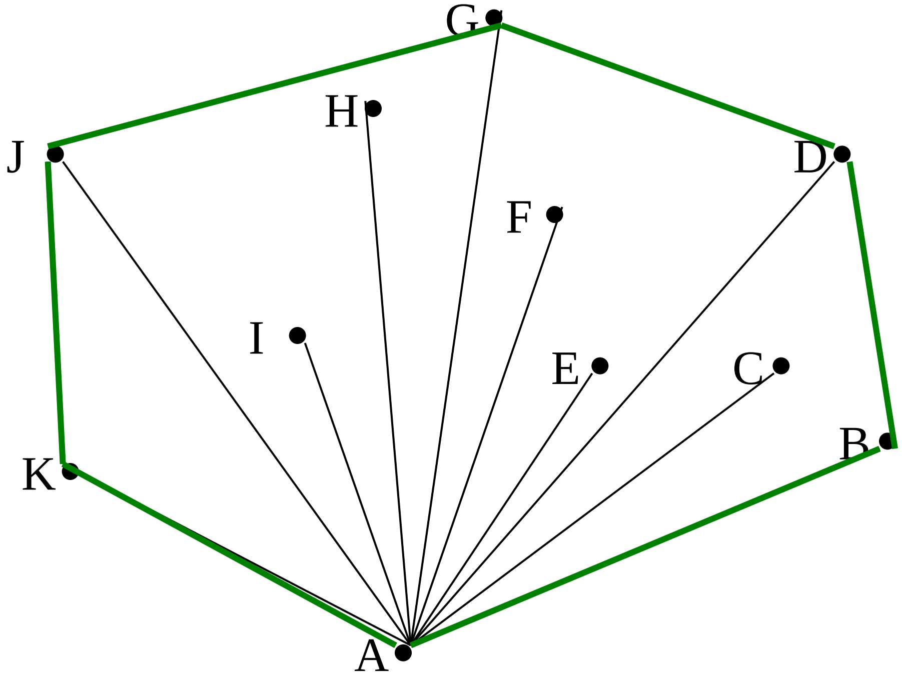


itd

Przykład



Przykład



Każdy punkt, który został usunięty ze stosu,

nie należy do otoczki wypukłej.

Zbiór punktów na stosie tworzy wielokąt wypukły.

Koszt algorytmu:

Operacje dominujące to porównywanie współrzędnych
lub badanie położenia punktu względem prostej.

$$\underbrace{O(n)}_{\text{szukanie minimum}} + \underbrace{O(n \log n)}_{\text{sortowanie}} + \underbrace{O(1)}_{\text{inicjalizacja stosu}} + \underbrace{O(n-3)}_{\text{krok4}} = O(n \log n)$$

Algorytm Jarvisa (owijanie prezentu)

znajdź punkt i_0 z S o najmniejszej współrzędnej y; $i \leftarrow i_0$

repeat

for $j \neq i$ **do**

znajdź punkt, dla którego kąt liczony przeciwnie do wskazówek zegara w odniesieniu do ostatniej krawędzi otoczki jest najmniejszy

niech k będzie indeksem punktu z najmniejszym kątem

zwróć (p_i, p_k) jako krawędź otoczki

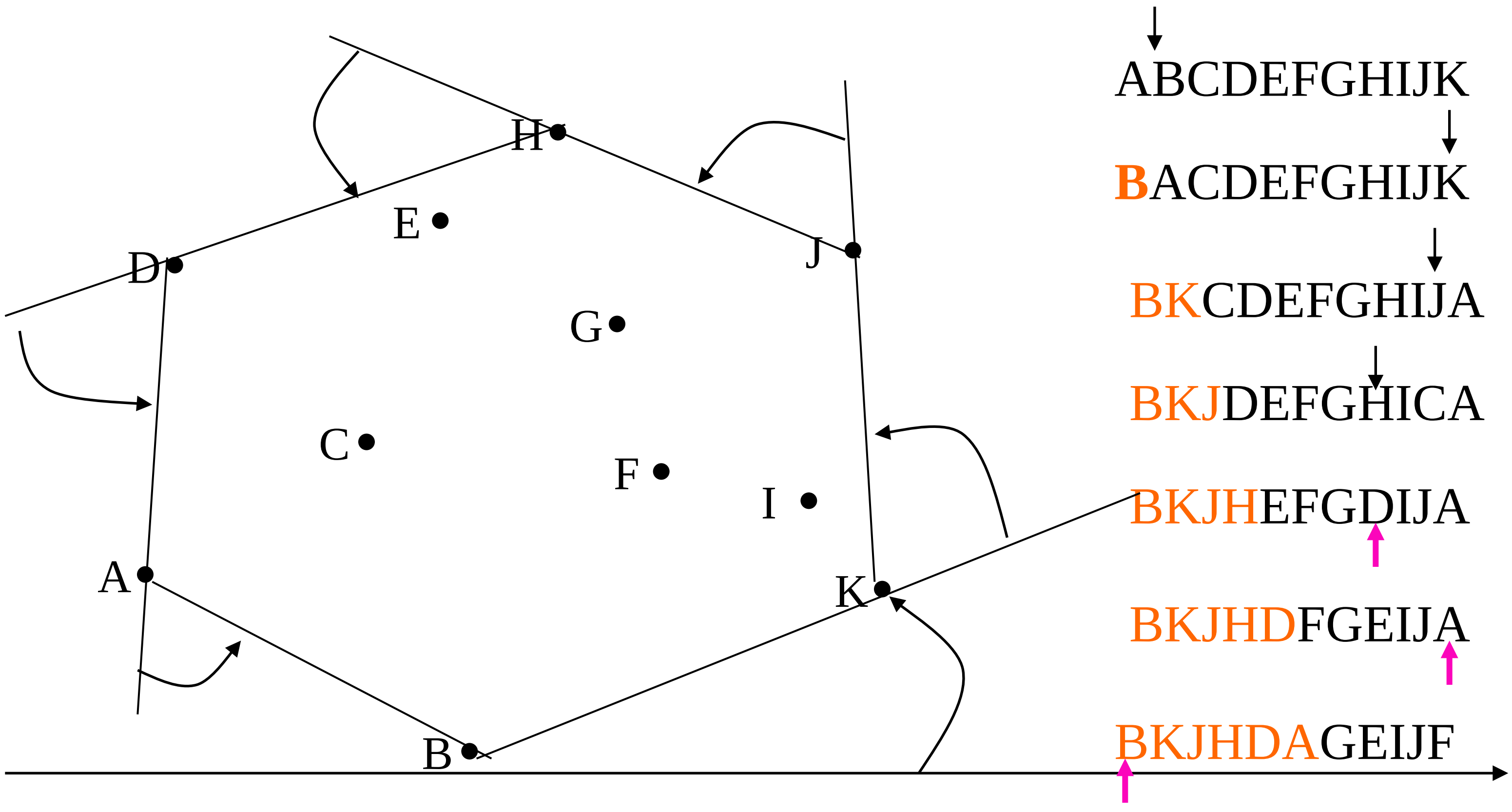
$i \leftarrow k$

until $i = i_0$

Złożoność rzędu $O(n^2)$

Gdy liczba wierzchołków otoczki jest ograniczona przez stałą k, jego złożoność jest rzędu $O(kn)$.

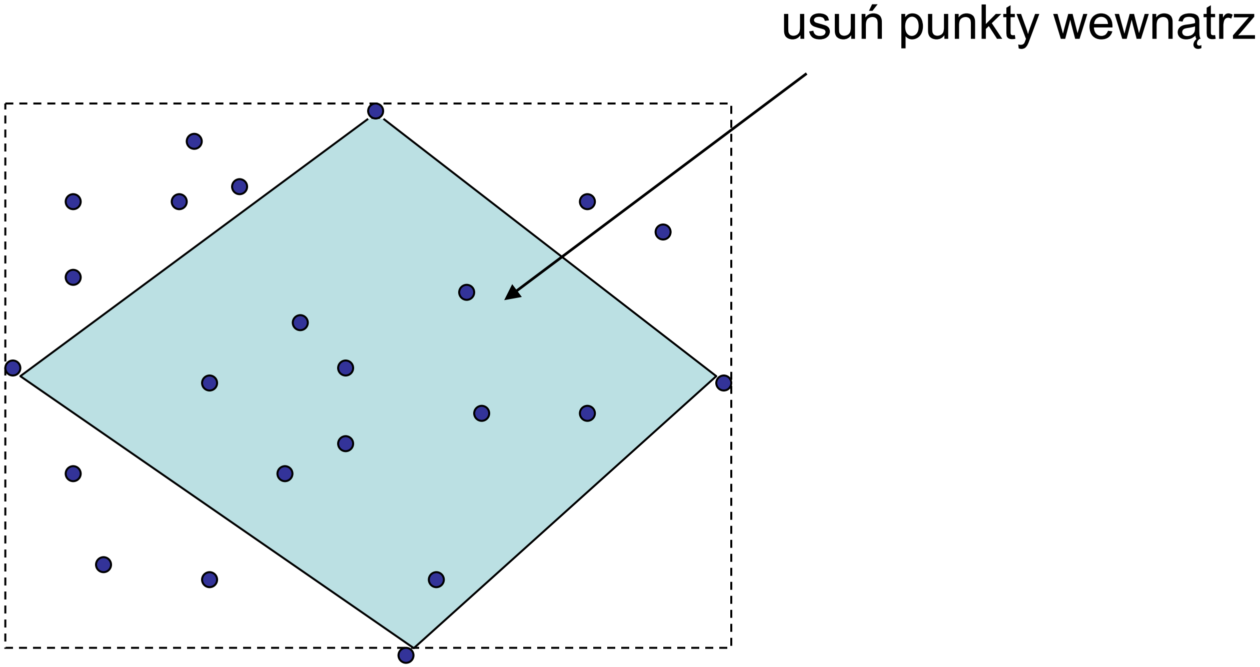
Przykład



H
E
D
J
G
C
F
I
A
K
B
ABCDEFGHIJK
BACDEFGHIJK
BKCDEFGHIJA
BKJDEFGHICA
BKJHEFGDIJA
BKJHDFGEIJA
BKJHDAGEIJF

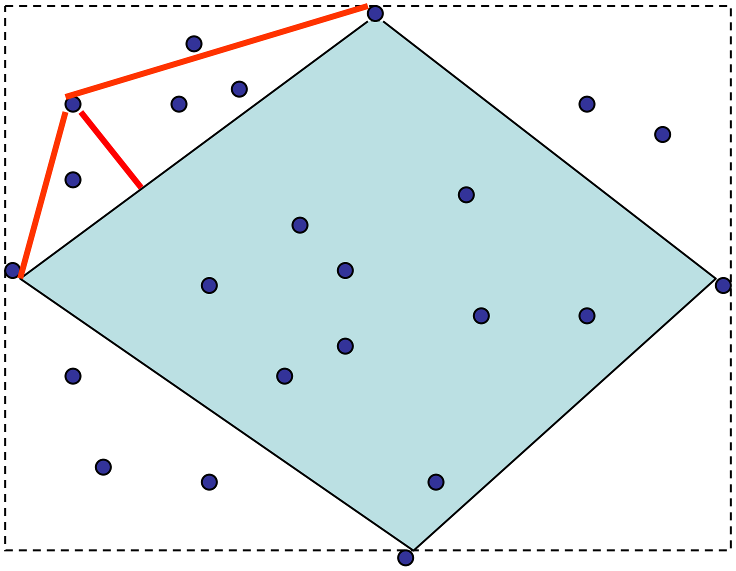
Algorytm *QuickHull*

- Znajdź 4 punkty ekstremalne
- Dodaj 4 krawędzie do ***TempCH***
- Usuń punkty wewnętrzne

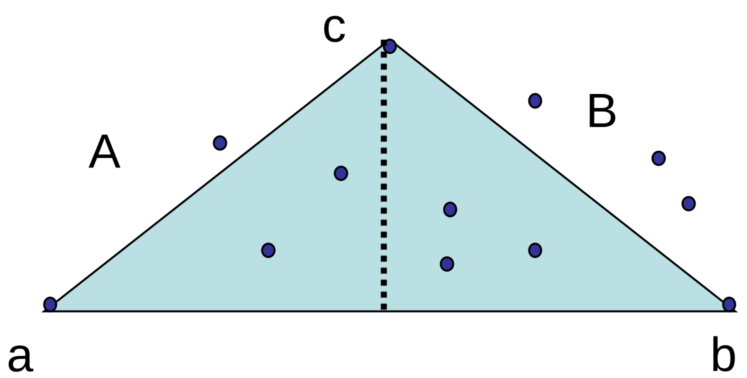


- Proceduj z punktami na zewnątrz *TempCH*

Algorytm *QuickHull*



Algorytm *QuickHull*



- Znajdź punkt maksymalnie odległy i ⊥ do *ab*
- Usuń punkty wewnątrz trójkąta *abc*
- Podziel punkty na dwa podzbiory:
  - na zewnątrz *ac* (na lewo do *ac*)
  - na zewnątrz *cb* (na lewo do *cb*)
- Zastąp krawędź *ab* w *TempCH* przez *ac* i *cb*
- Proceduj z punktami na zewnątrz *ac* i *cb* rekurencyjnie

Algorytm *QuickHull*

Function ***QuickHull***(a,b,S)
if $S=\{a,b\}$ **then return** $\{a,b\}$
    else
        $c \leftarrow$ indeks punktu maksymalnie odległego i $\perp$ do ab
        $A \leftarrow$ punkty z prawej strony ac
        $B \leftarrow$ punkty z prawej strony cb
        return ***QuickHull***(a,c,A) + ***QuickHull***(c,b,B)

Złożoność oczekiwana rzędu O(*n log n*)

Złożoność w najgorszym przypadku rzędu O(n^2)

Algorytm *dziel i rządź*

F:={S};

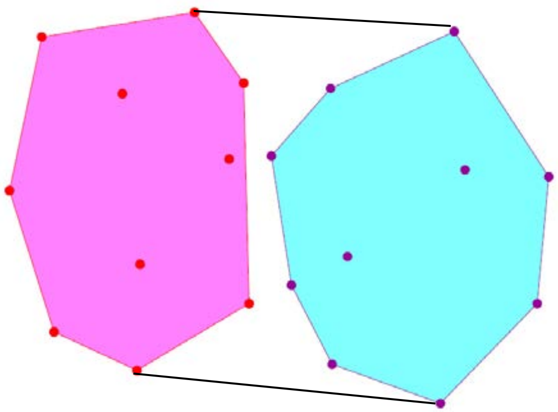
while rozmiar dowolnego zbioru z F przekracza daną stałą *k*

dziel zbiory względem mediany *x*-owych współrzędnych punktów;

znajdź otoczki wypukłe zbiorów z F

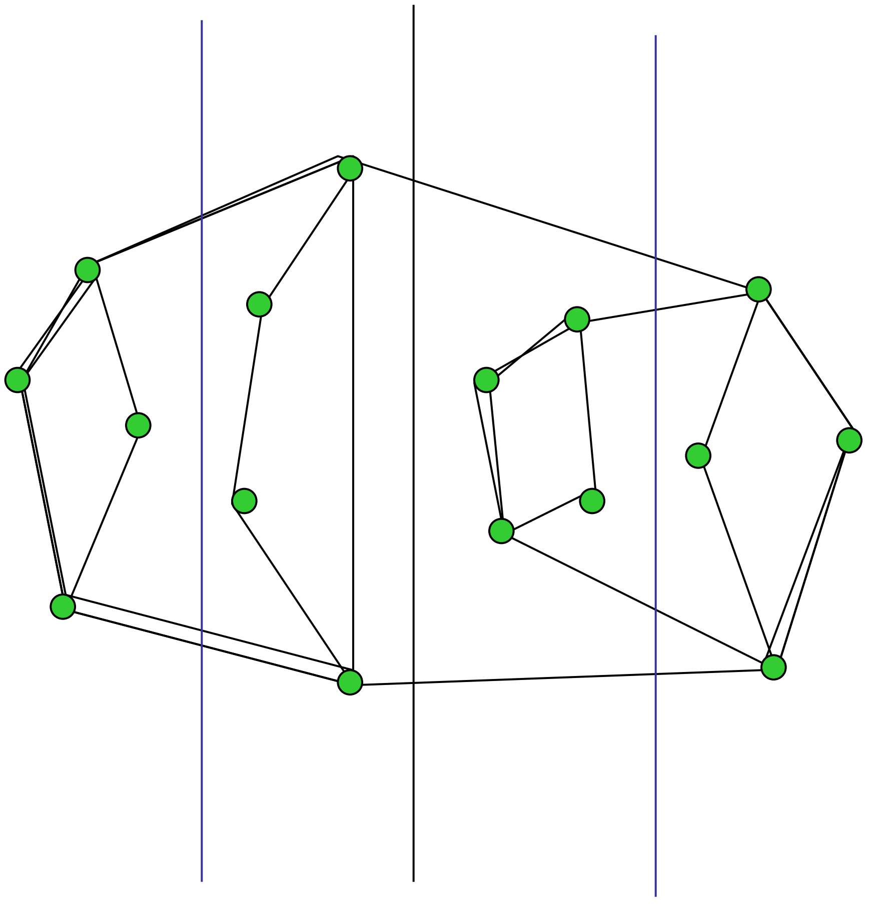
while otoczka zbioru S nie została znaleziona

sklejaj otoczki sąsiadujących zbiorów w kolejności odwrotnej do podziału rodziny F;

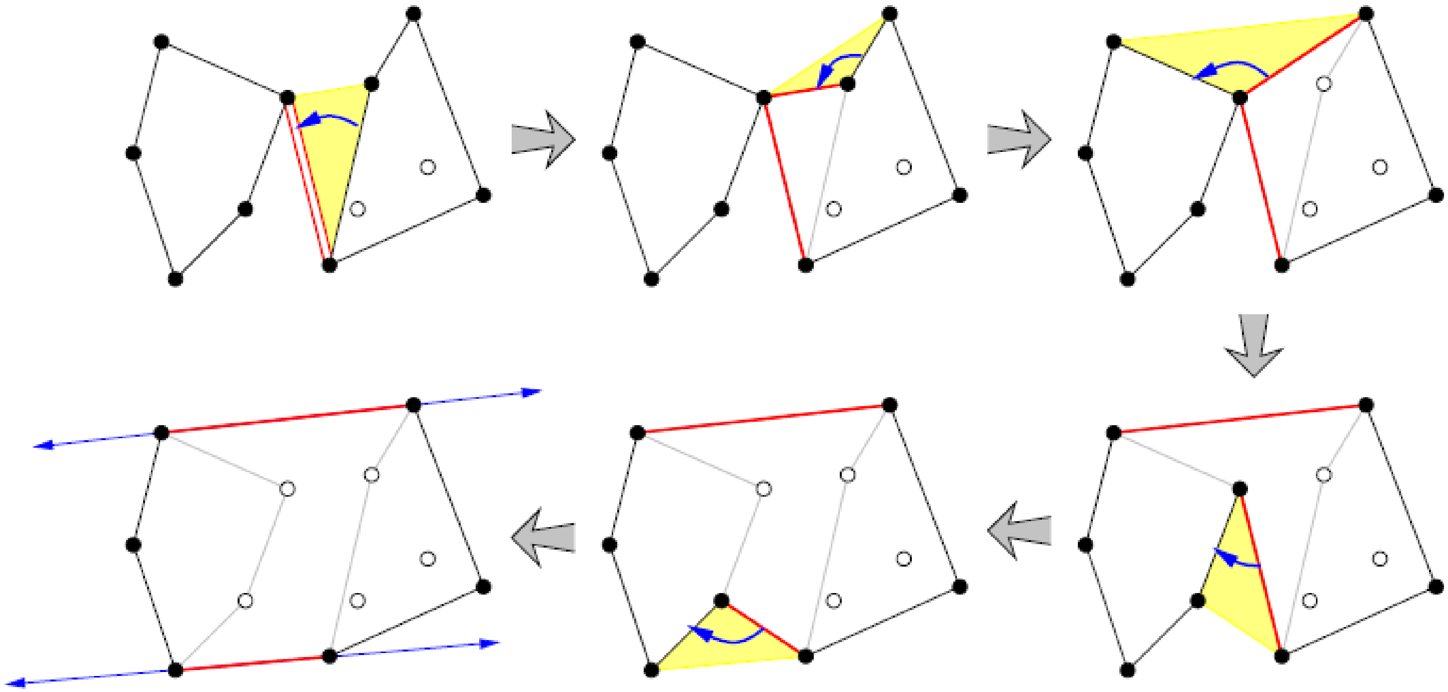


Złożoność rzędu O(*n* log *n*)

Algorytm *dziel i rządź*



Łączenie sąsiednich otoczek



Łączenie sąsiednich otoczek

Niech l(a,b) oznacza prostą wyznaczoną przez punkty a i b.

a - skrajnie prawy wierzchołek lewej otoczki A

b - skrajnie lewy wierzchołek prawej otoczki B

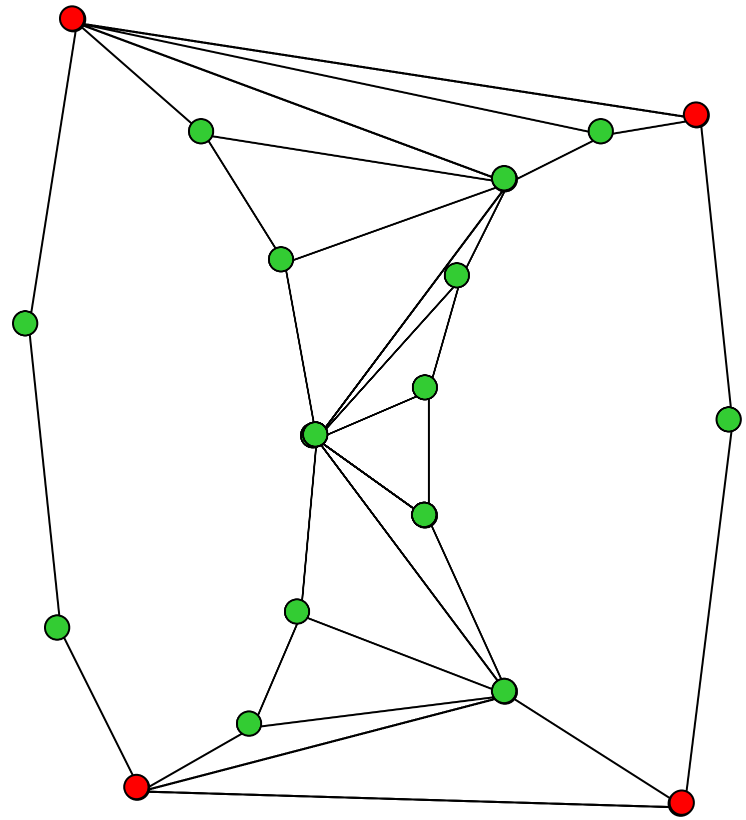
while l(a,b) nie jest styczna do A i B

while l(a,b) nie jest dolną (górną) styczną do B

b := dolny (górny) sąsiad b;

while l(a,b) nie jest dolną (górną) styczną do A

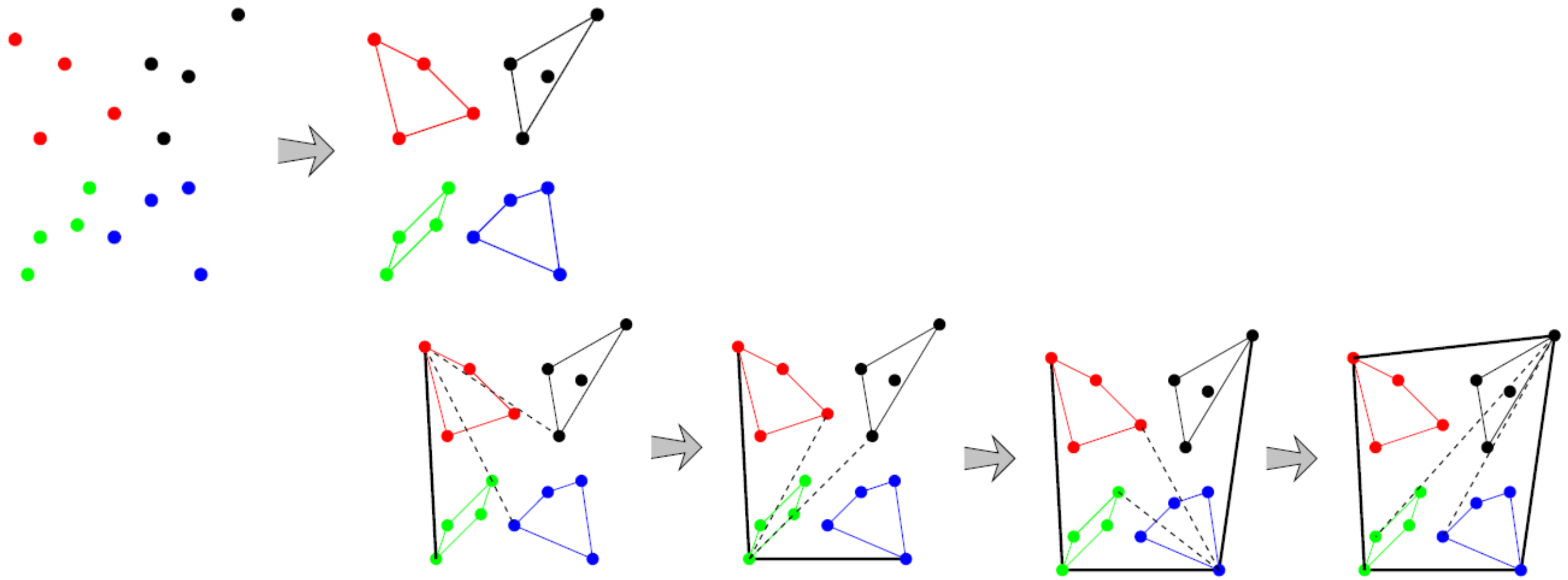
a := dolny (górny) sąsiad a;



Algorytm *Chan'a*

Łączy algorytmy Grahama i Jarvisa

- Podzielić *n* punktów na *r* grup o rozmiarze *m*
- Wyznaczyć otoczkę dla każdej grupy (Graham)
- Uruchomić algorytm Jarvisa dla grup



Algorytm *Chan’a*

Algorytm wymaga znajomości rozmiaru otoczki *h*

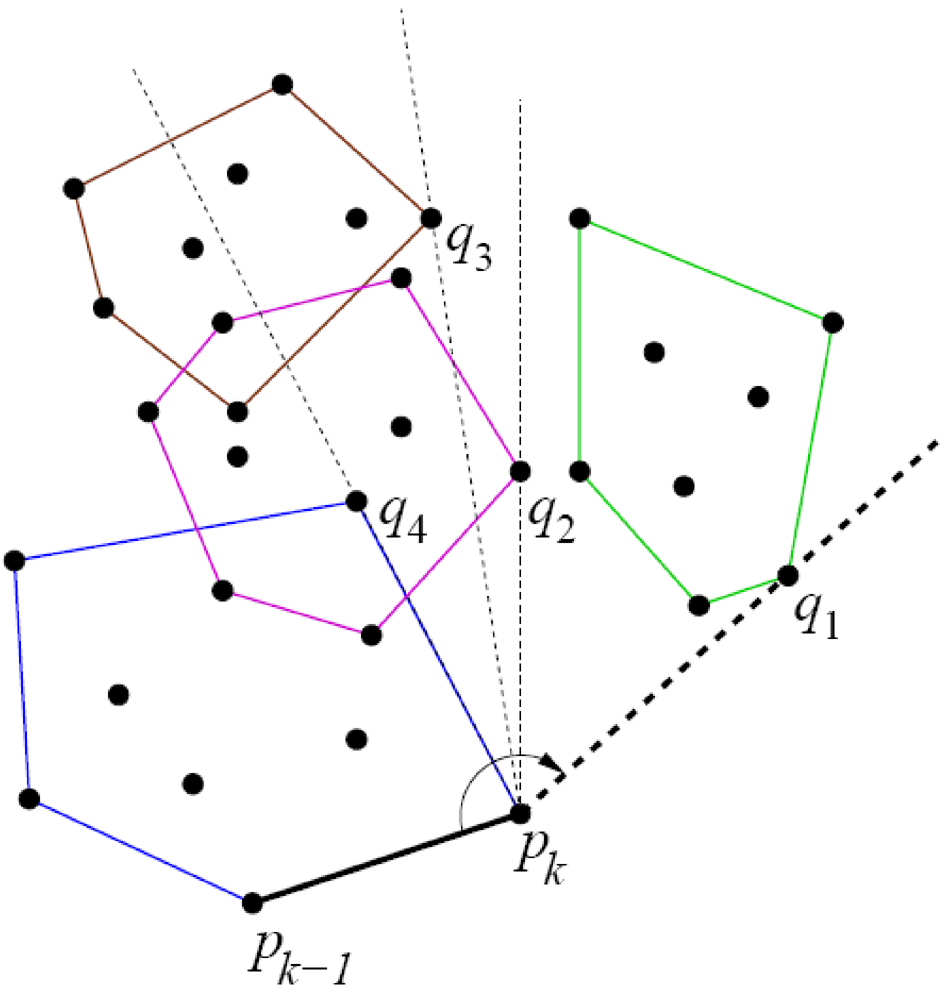
Załóżmy na razie, że *m*=*h*

$$r = \lceil n / m \rceil$$

PartialHull (*S*,*m*)

- Podzielić S na r grup o rozmiarze m
- Wyznaczyć algorytmem Grahama CH(S_i) dla $i = 1, \ldots, r$
- Znaleźć najniższy punkt p_1 zbioru S
- **for** $k = 1$ to m **do**
  - **for** i = 1 to r **do**
    znajdź q_i □ S który maksymalizuje kąt □ $p_{k-1}\, p_k\, q_i$
  - $p_{k+1} \leftarrow$ znalezione q (nowy punkt CH)
  - if $p_{k+1} = p_1$ **zwróć** $(p_1, \ldots, p_k)$
- **zwróć** „*m* za małe”

Algorytm *Chan'a*



Algorytm *Chan'a*

Złożoność:

- Algorytm Grahama: $O(rm \log m) = O(n \log m)$
- Znalezienie stycznej z punktu do otoczki: $O(\log n)$
- Koszt algorytmu Jarvisa dla r otoczek: $O((hn/m) \log m)$

Razem: $O((n+(hn/m)) \log m)$

Jeśli $m = h$, to $O(n \log h)$

Problem: nie znamy h

Algorytm *Chan'a*

Hull(s)

for t = 1, 2, … **do**

1. niech $m = \min(2^{2^\wedge t}, n)$
2. L = PartialHull(S, m)
3. jeśli $L \neq$ „m za małe”, to **zwróć** L

Porównanie złożoności

Graham	$O(n \log n)$		Graham, 1972
Jarvis	$O(n\ \mathrm{k})$	max $\mathrm{O}(n^2)$	Jarvis, 1973
Quickhull	$O(n \log n)$	max $\mathrm{O}(n^2)$	Eddy, 1977, Bykat, 1978
Dziel i rządź	$O(n \log n)$		Preparata & Hong, 1977
Dolna i górna	$O(n \log n)$		Andrew, 1979
Przyrostowy	$O(n \log n)$		Kallay, 1984
Chan	$O(n \log k)$		Chan, 1996