Otoczka wypukła dla zbioru punktów w przestrzeni dwuwymiarowej

PROJEKT

Algorytmy geometryczne, 2022/2022

Grupa 4 – czwartek, 11:20-12:50, tydzień B

Adam Naumiec, Michał Kuszewski

Algorytmy wyznaczania otoczki wypukłej

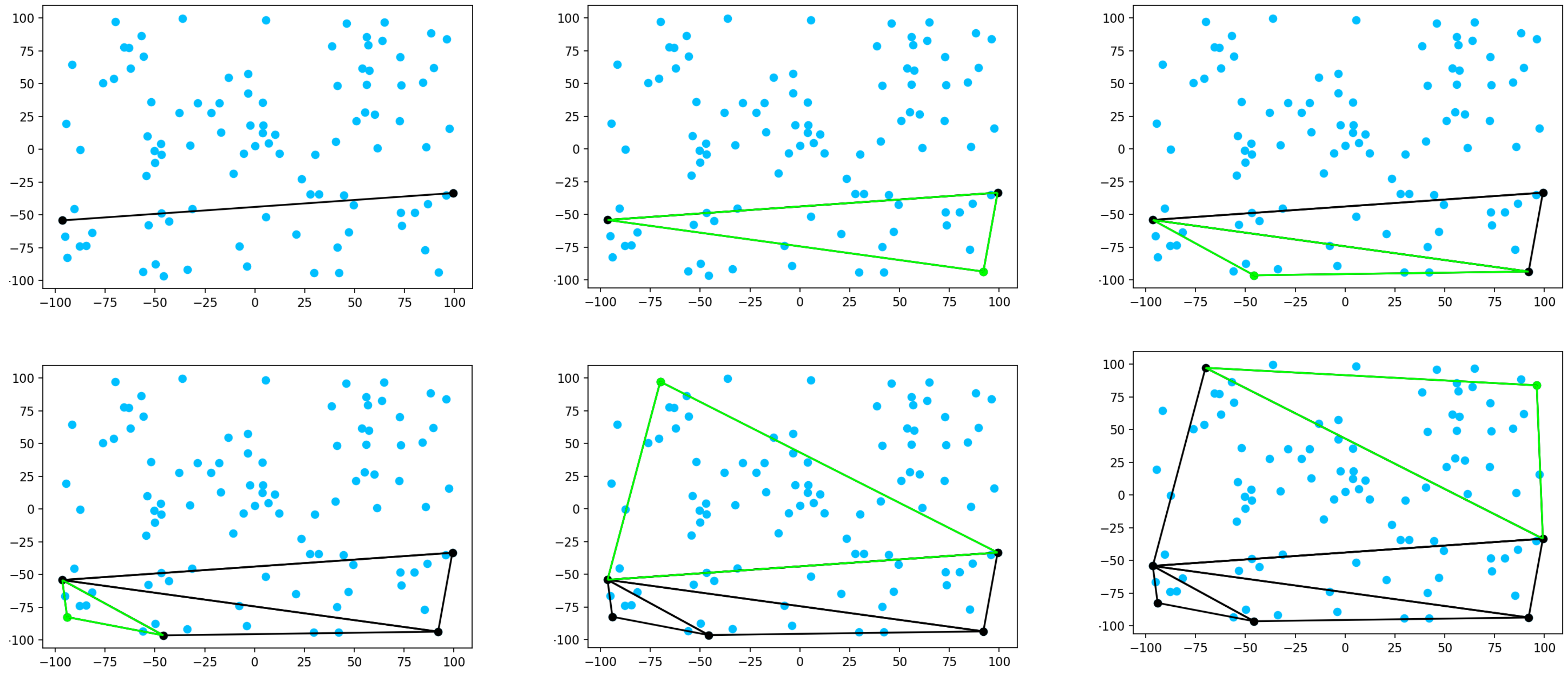
- Dziel i rządź
- Przyrostowy
- Chana
- Quickhull
- Górnej i dolnej otoczki
- Jarvisa
- Grahama

Algorytm Quickhull

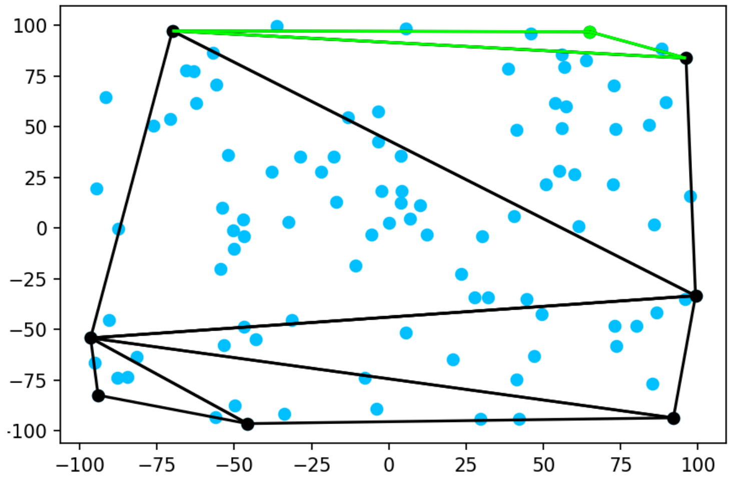
Przebieg działania algorytmu:

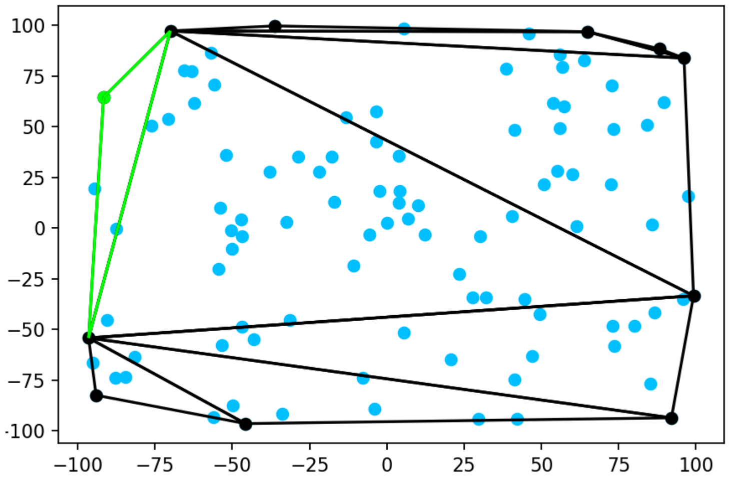
1. Znajdź najbardziej położone na lewo i prawo punkty (po współrzędnej x) dodaj te punkty do otoczki.
2. Te dwa punkty wyznaczają prostą, względem której podziel punkty na te znajdujące się powyżej i poniżej prostej.
3. Znajdź w obu podzbiorach punkt najdalej oddalony od prostej (stosując metrykę Euklidesową) – ten punkt należy do otoczki wypukłej. Żaden z punktów wewnątrz trójkąta zbudowanego z dwóch pierwszych punktów i punktu najdalej położone od prostej na pewno nie należy do otoczki.
4. Szukaj rekurencyjnie punktów odpowiednio powyżej i poniżej prostych skonstruowanych z punktu najdalej położonego od prostej i poprzednich punktów wyznaczających prostą w zbiorach górnym i dolny.
5. Za każdym razem punkt najdalej oddalony od prostej jest punktem, który należy do otoczki.
6. Punkty dzielimy obliczając wyznacznik. Kroki powtarzamy dopóki znajdujemy punkty najdalej położone od prostej. Jeżeli jest tylko jeden taki punkt to dodajemy go do otoczki i nie szukamy w tej części kolejnych punktów.

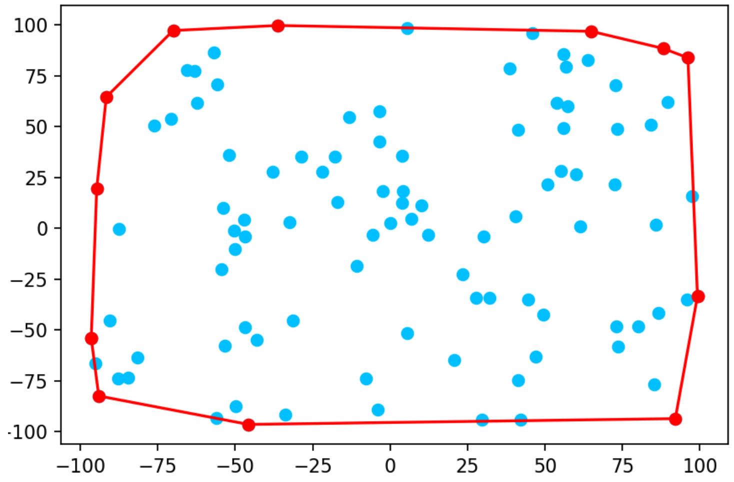
Przebieg algorytmu quickhull 1



Przebieg algorytmu quickhull 2





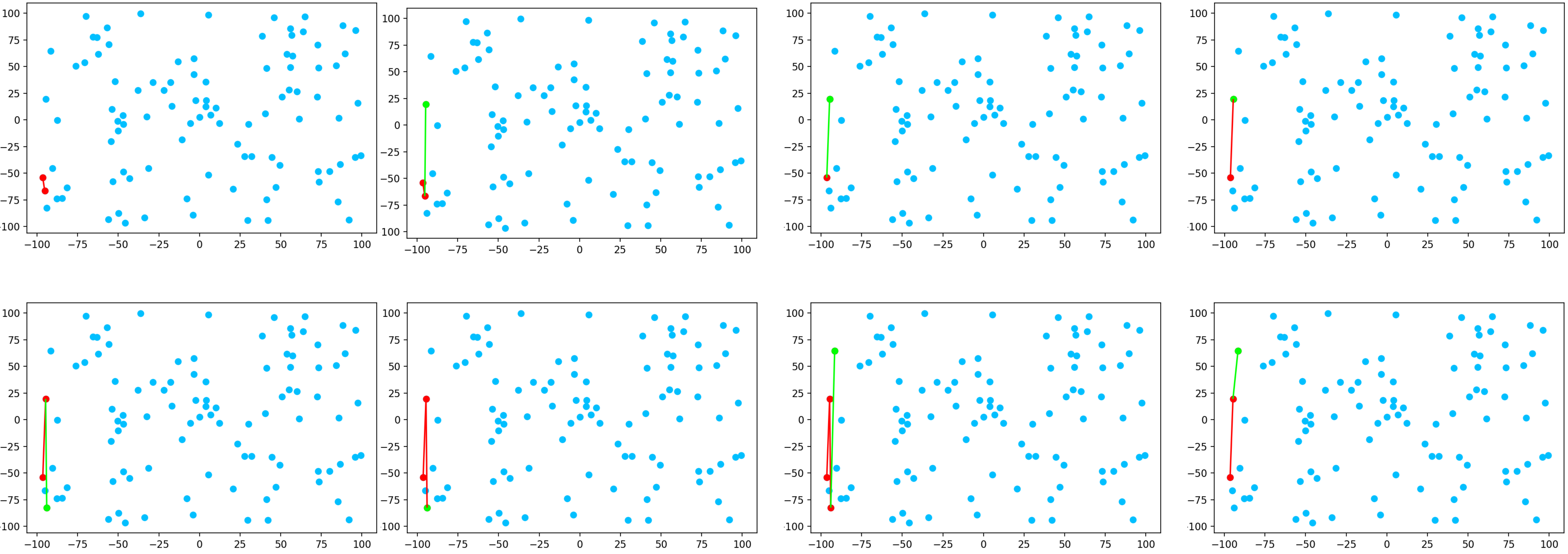


Algorytm górnej i dolnej otoczki

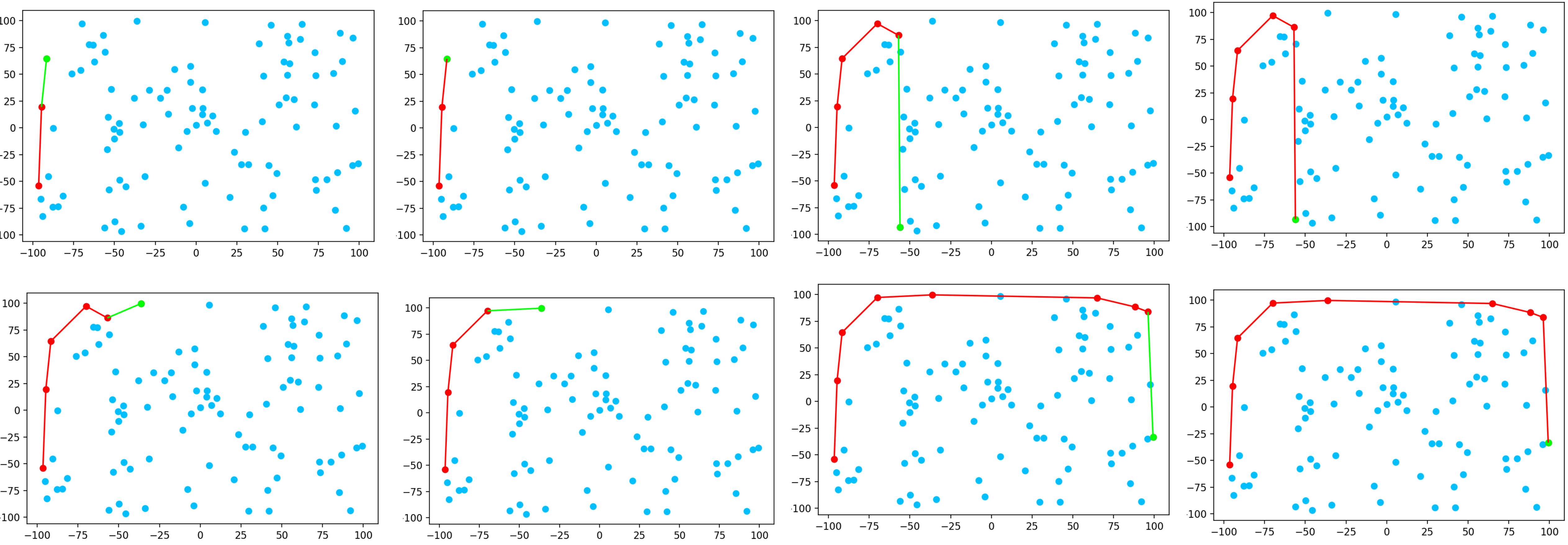
Przebieg działania algorytmu:

1. Posortuj punkty leksykograficznie względem współrzędnej x, a dla kilku punktów o tej samej współrzędnej x – względem współrzędnej y.
2. Dodaj do listy dwa pierwsze punkty z posortowanego zbioru. Przechodź po wszystkich punktach zgodnie z ich kolejnością i dodawaj każdy punkt na koniec listy.
3. Za każdym razem sprawdź, czy dodany punkt nie jest ujemnie zorientowany (obliczając wyznacznik) z dwoma ostatnimi punktami w liście. Jeśli tak usuń ostatnio dodany punkt i powtarzaj procedurę dopóki ostatnie punkty są ujemnie zorientowane i są w liście co najmniej 3 punkty
4. (punkt ujemnie zorientowany leży, przy szukaniu górnej otoczki – poniżej górnej otoczki, przy szukaniu dolnej otoczki – powyżej dolnej otoczki, oznacza to, że nie może on znaleźć się odpowiednio w górnej lub dolnej otoczce, zatem usuwamy wszystkie takie punkty z listy)
5. Gdy przejdziesz przez wszystkie punkty stwórz nową listę i wykonuj analogiczne kroki jak dla górnej otoczki.
6. Połącz obie listy – górną i dolną otoczkę – razem tworzą one całą właściwą otoczkę.

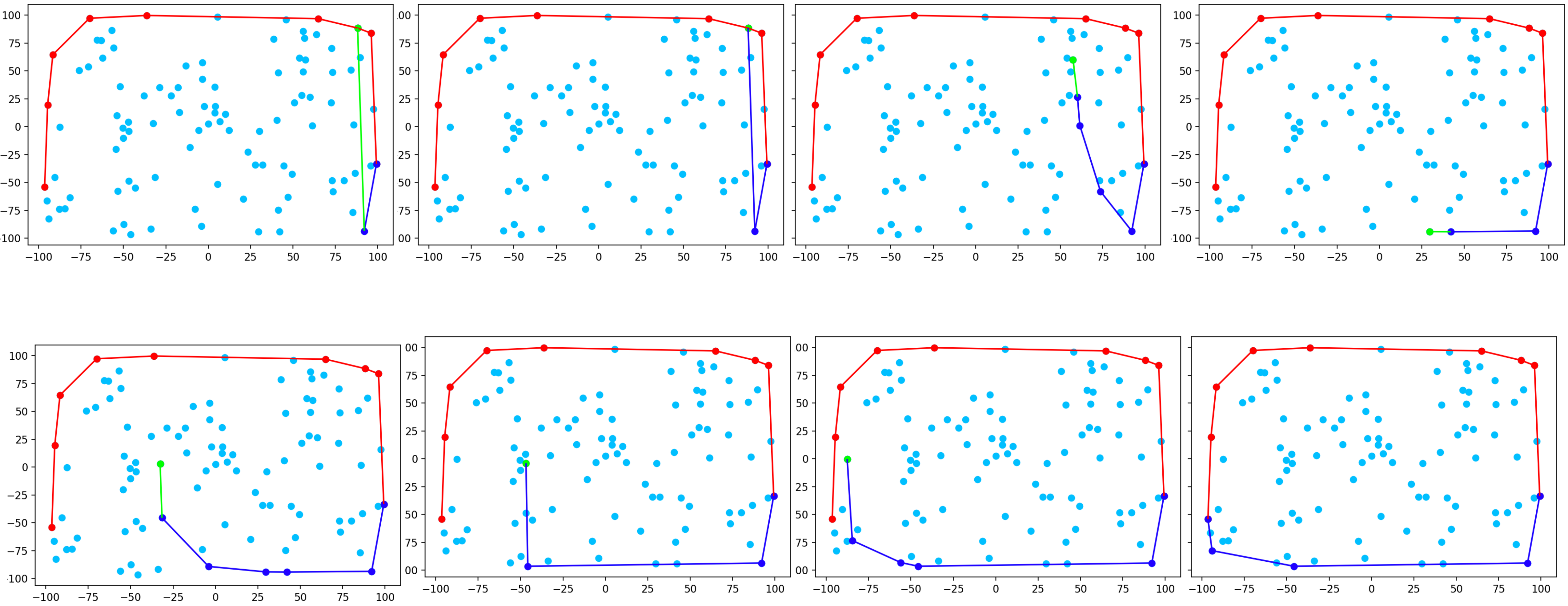
Przebieg algorytmu górnej i dolnej otoczki 1



Przebieg algorytmu górnej i dolnej otoczki 2



Przebieg algorytmu górnej i dolnej otoczki 3



Algorytm dziel i rządź (zwyciężaj)

Przebieg działania algorytmu:

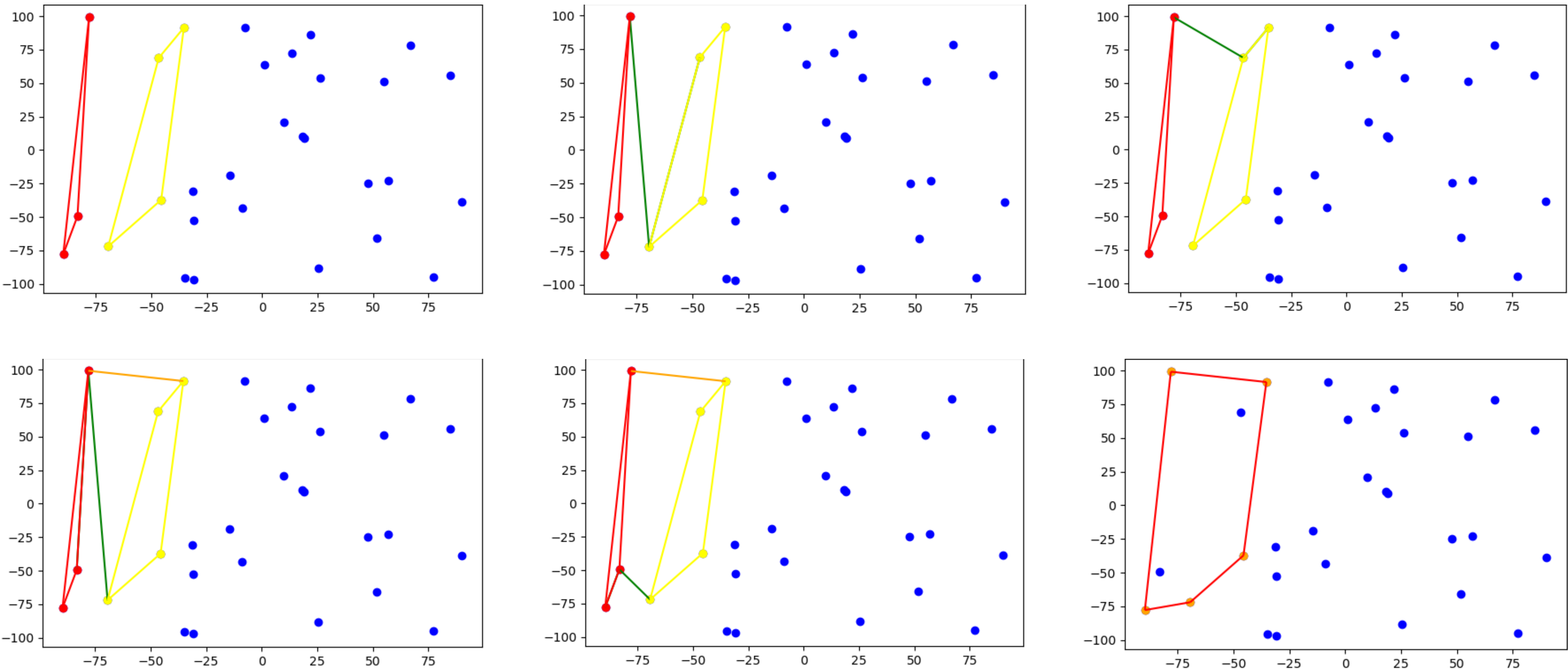
1. Posortuj punkty leksykograficznie.
2. Jeśli zbiór punktów liczy 5 lub mniej punktów znajdź otoczkę wypukłą korzystając z algorytmu krawędzi skrajnych.
3. W przeciwnym przypadku podziel zbiór punktów na dwa zbiory według mediany ich współrzędnych x.
4. Dla każdego ze zbiorów wyznacz rekurencyjnie otoczkę wypukłą.
5. Scal obliczone otoczki wypukłe w jedną otoczkę.

Scalanie otoczek w algorytmie dziel i rządź

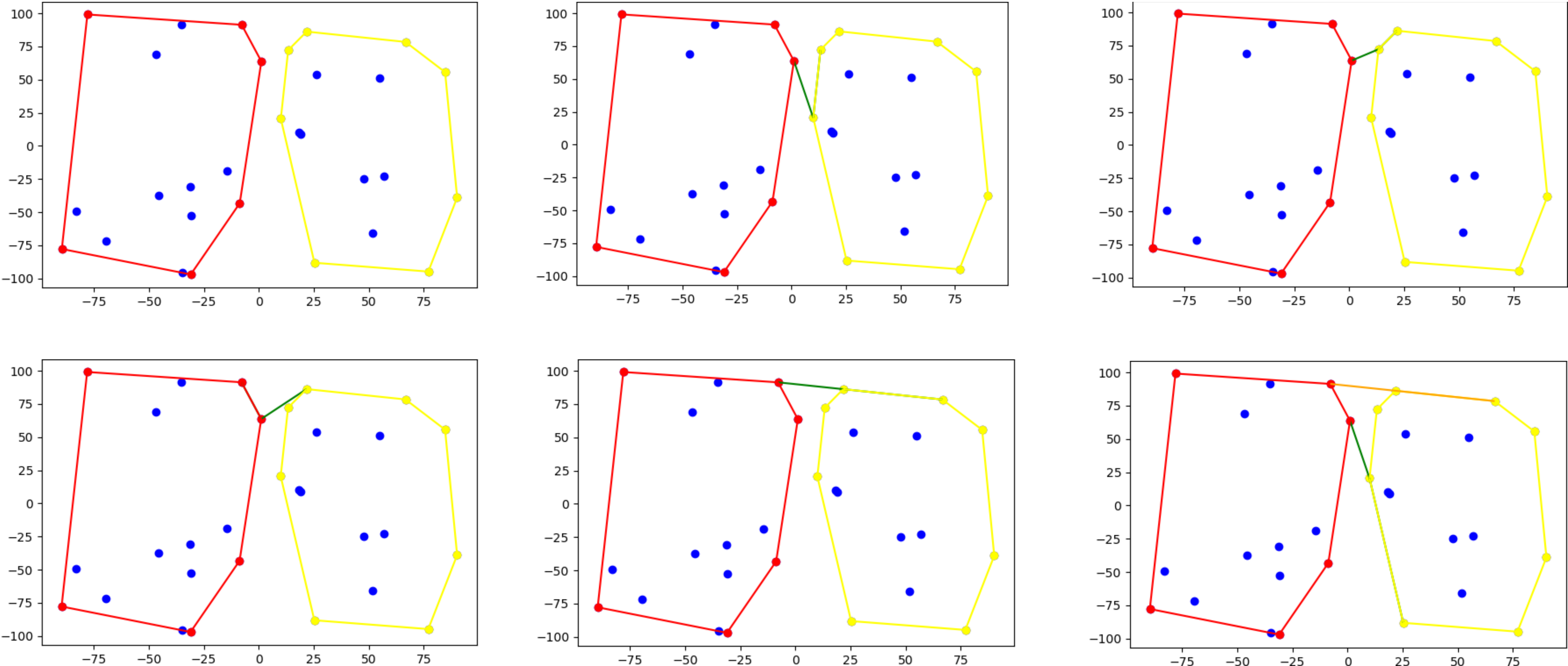
Scalanie otoczek przebiega następująco:

1. Wyznacz skrajnie prawy punkt lewej otoczki – zapisany jako a i skrajnie lewy punkt prawej otoczki – zapisany jako b.
2. Dopóki prosta łącząca punkty a i b nie jest górną styczną do dwóch otoczek:
    1. Dopóki prosta łącząca a i b nie jest górną styczną do prawej otoczki: Przypisz do b górnego sąsiada b.
    2. Dopóki prosta łącząca a i b nie jest górną styczną do lewej otoczki: Przypisz do a górnego sąsiada a.
3. Wyznaczone punkty a i b są górnymi punktami otoczek.
4. Przypisz do a i b skrajne punkty jak w punkcie 1.
5. Dopóki prosta łącząca punkty a i b nie jest dolną styczną do dwóch otoczek:
    1. Dopóki prosta łącząca a i b nie jest dolną styczną do prawej otoczki: Przypisz do b dolnego sąsiada b.
    2. Dopóki prosta łącząca a i b nie jest dolną styczną do lewej otoczki: Przypisz do a dolnego sąsiada a.
6. Wyznaczone punkty a i b są dolnymi punktami otoczek.
7. Utwórz otoczkę wynikową zapisując w niej punkty znajdujące się pomiędzy górnym a dolnym punktem lewej otoczki (wraz z tymi punktami) dodając punkty znajdujące się między dolnym a górnym punktem prawej otoczki (wraz z tymi punktami).

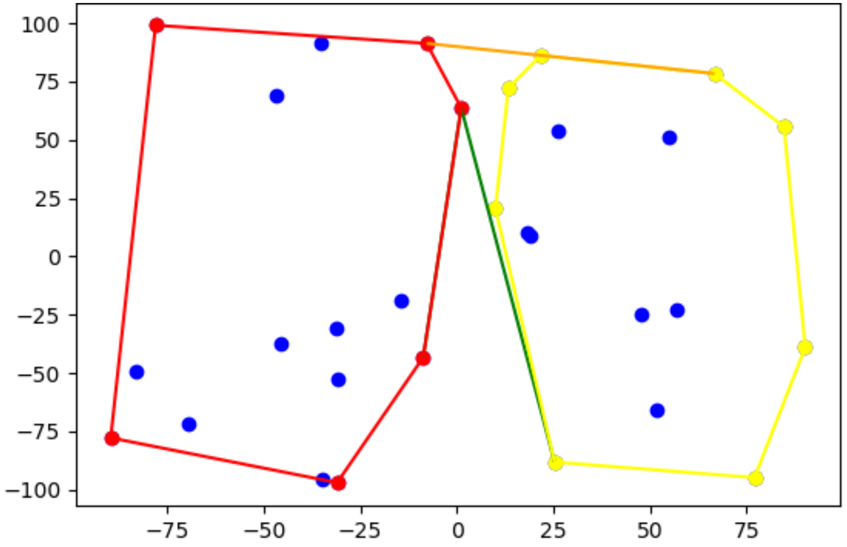
Przebieg algorytmu dziel i rządź

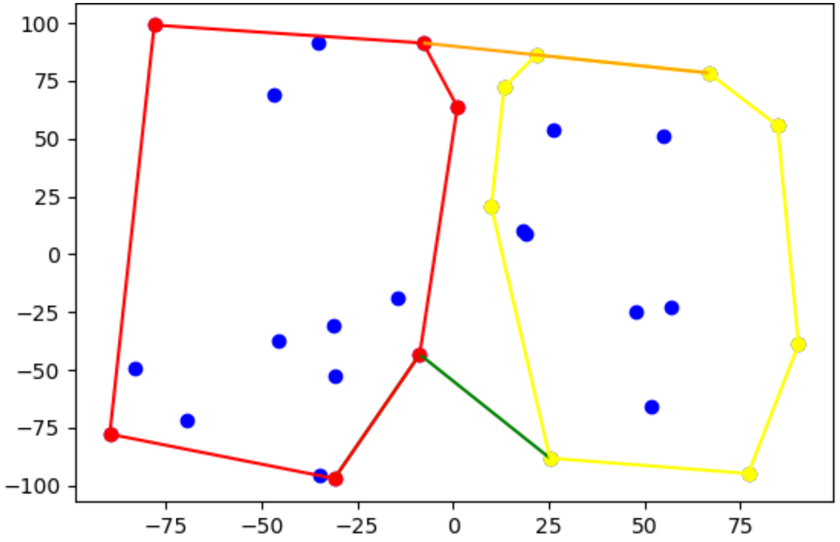


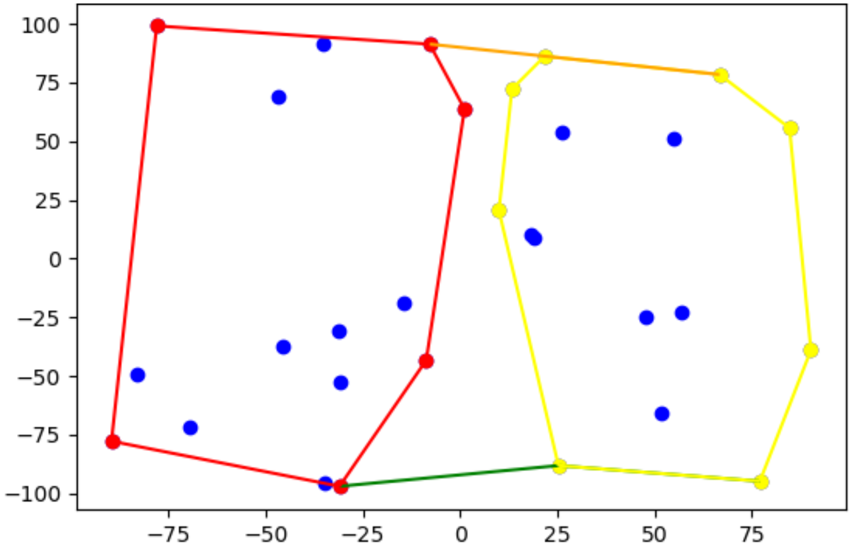
Przebieg algorytmu dziel i rządź

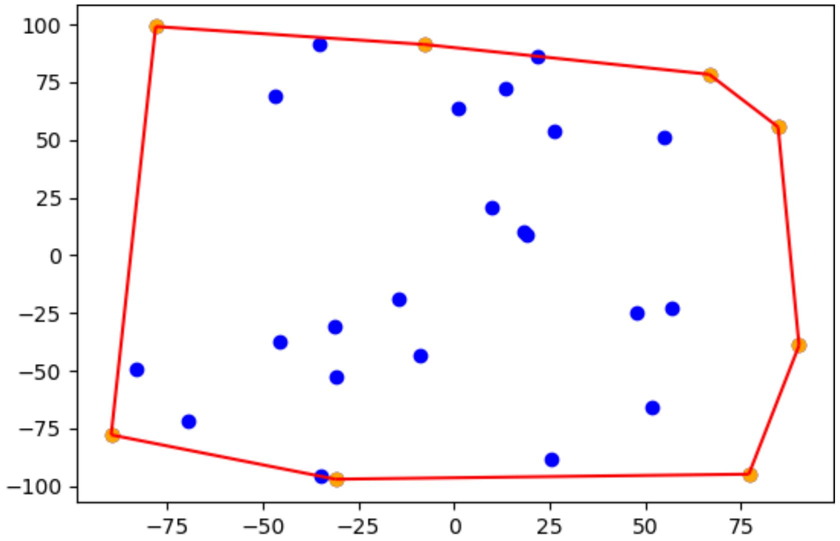


Przebieg algorytmu dziel i rządź









Algorytm przyrostowy

Przebieg działania algorytmu:

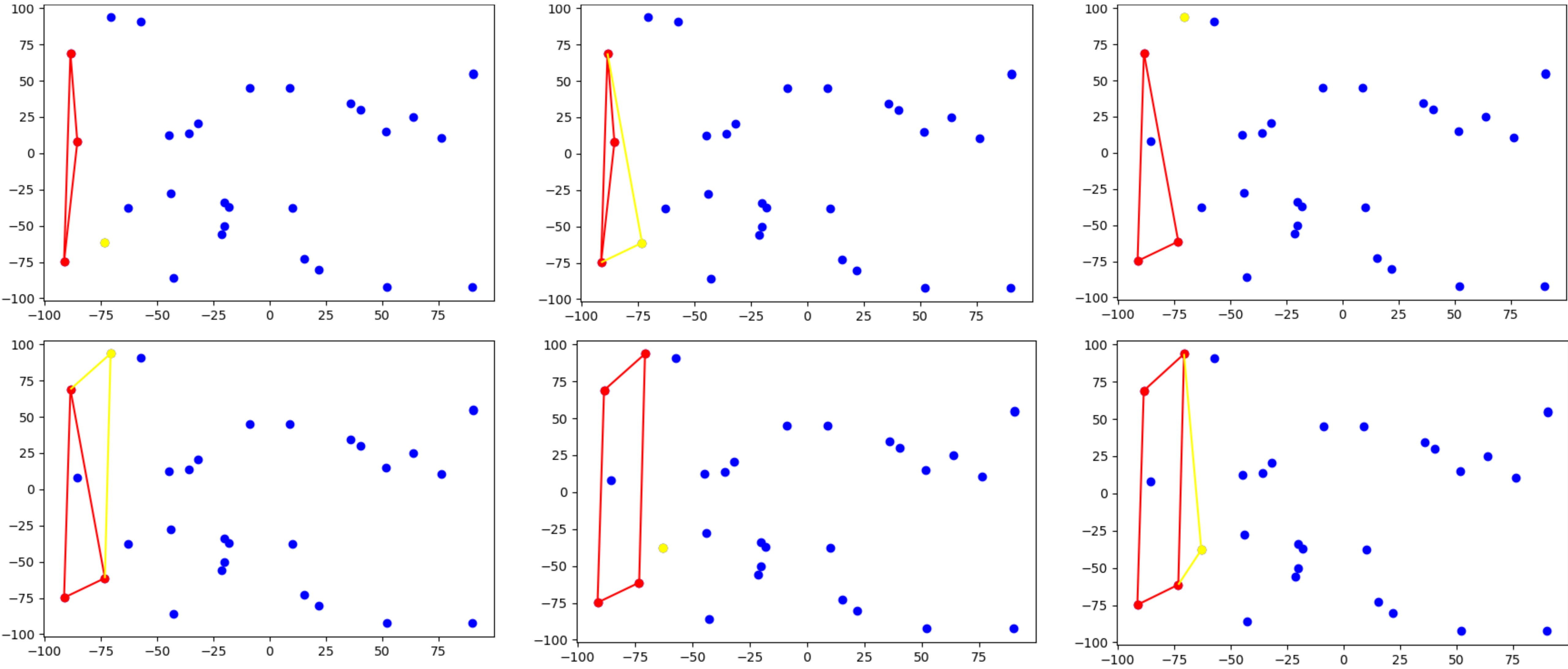
1. Posortuj punkty leksykograficznie.
2. Z pierwszych 3 punktów utwórz otoczkę wypukłą.
3. Przejdź po wszystkich punktach w tablicy, zaczynając od czwartego punktu.
4. W każdej iteracji dla rozpatrywanego punktu wyznacz styczne z tego punktu do istniejącej otoczki. Styczne punkty w otoczce oznaczone są jako a i b.
5. Usuń wszystkie punkty znajdujące się pomiędzy a i b (nie wliczając a i b).
6. Wstaw do otoczki obecnie rozpatrywany punkt między a i b.

Wyznaczanie stycznych w algorytmie przyrostowym

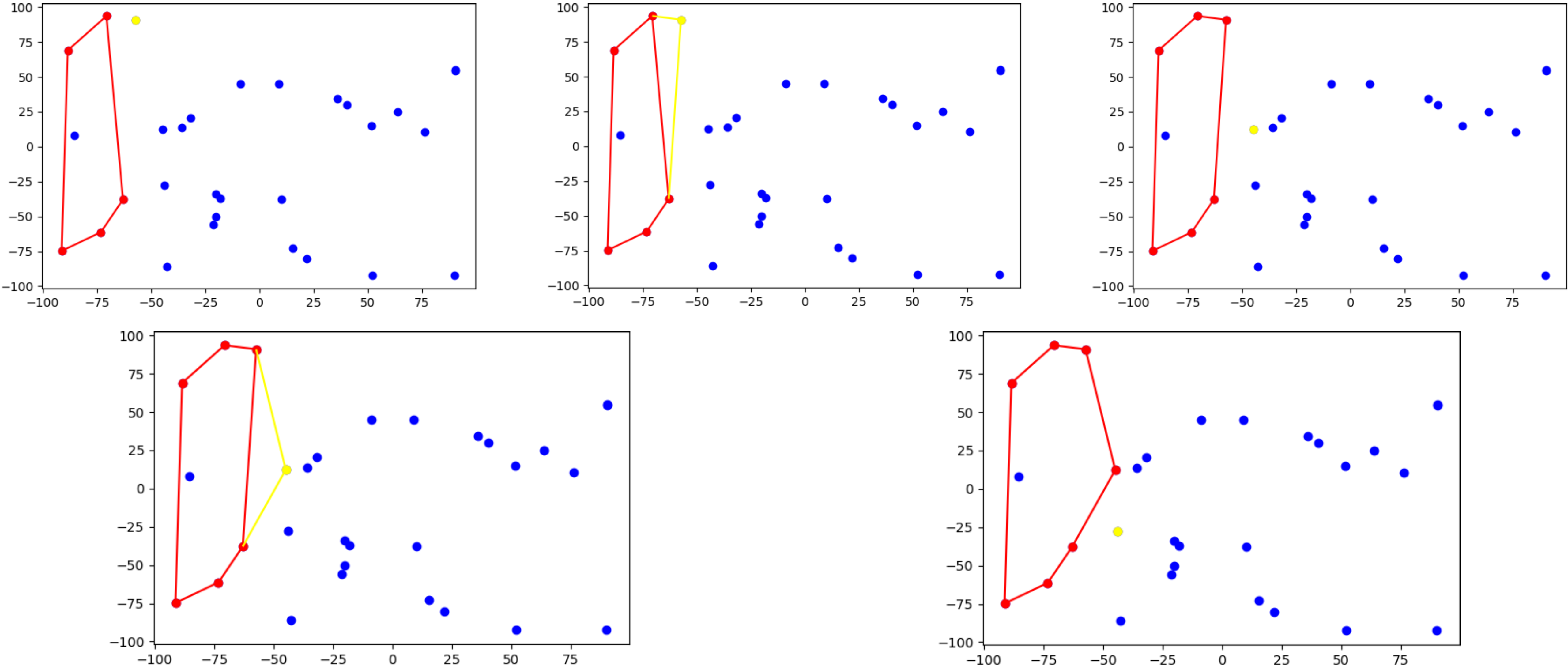
Wyznaczanie lewej stycznej z punktu do otoczki polega na znalezieniu ostatniego punktu, dla którego punkty: obecnie dodawany do otoczki punkt, rozpatrywany punkt otoczki, poprzednik rozpatrywanego punktu otoczki tworzą skręt w prawo.

Wyznaczanie prawej stycznej z punktu do otoczki polega na znalezieniu pierwszego punktu, dla którego punkty: obecnie dodawany do otoczki punkt, rozpatrywany punkt otoczki, następnik rozpatrywanego punktu otoczki tworzą skręt w lewo.

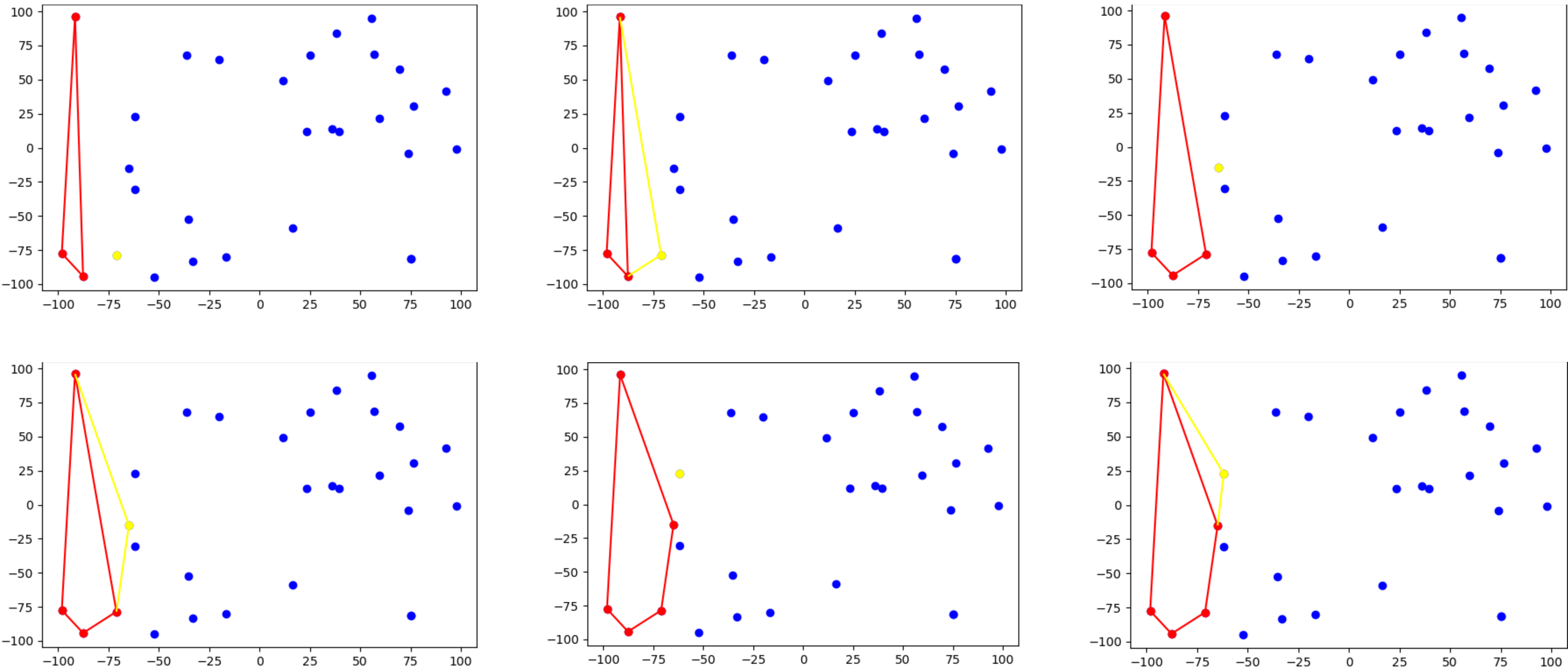
Przebieg algorytmu przyrostowego



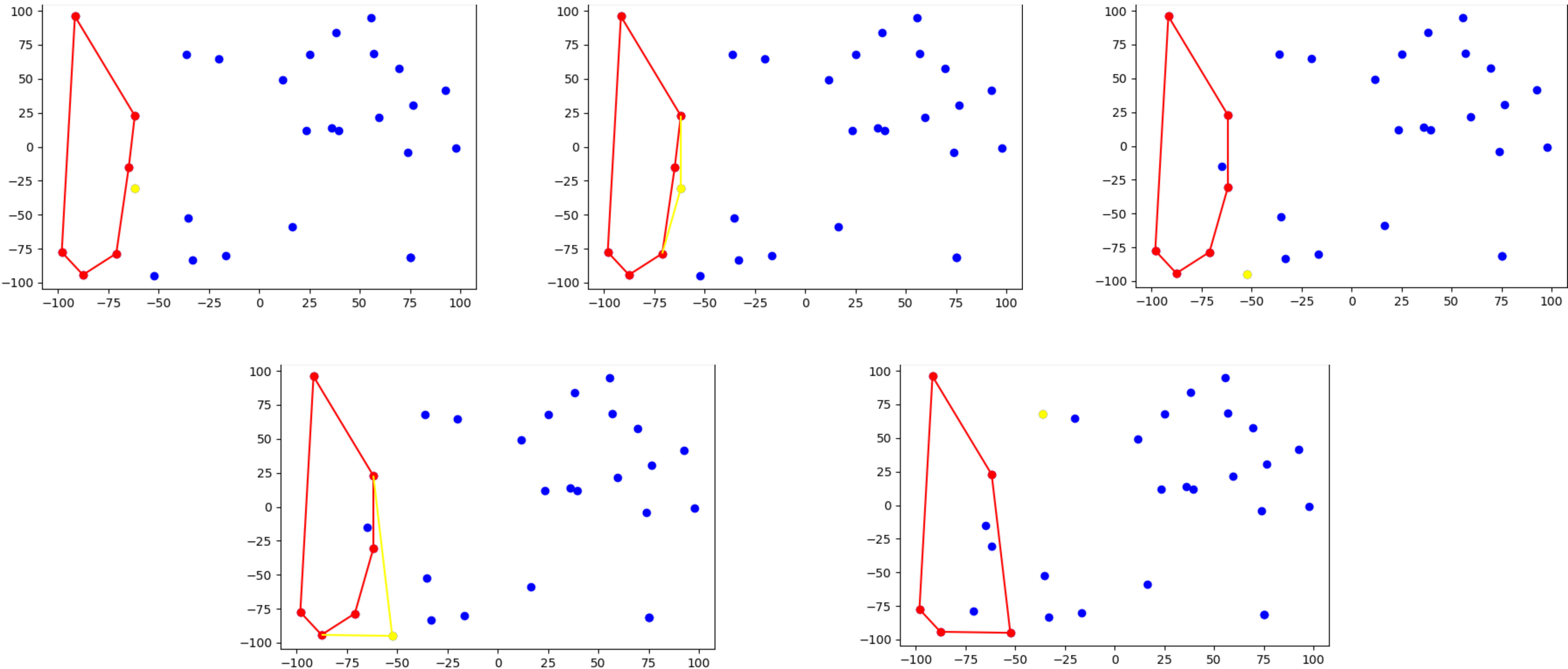
Przebieg algorytmu przyrostowego



Przebieg algorytmu przyrostowego



Przebieg algorytmu przyrostowego



Algorytm Chana

Przebieg działania algorytmu:

1. Mając aproksymację m liczby punktów w wynikowej otoczce, podziel zbiór punktów na $r = \left\lceil \frac{n}{m} \right\rceil$ zbiorów.
2. Dla każdego ze zbiorów wyznacz otoczkę wypukłą, korzystając z algorytmu Grahama.
3. Znajdź punkt z najmniejszą wartością współrzędnej y (punkt ten zawiera się w otoczce wynikowej).
4. m razy znajdź kolejny punkt do tworzonej otoczki, wyznaczając styczne z ostatniego punktu tworzonej otoczki do każdej z pomniejszych otoczek. Wybierz punkt, który tworzy największy kąt z punktami tworzonej otoczki. Jeśli znaleziony punkt jest pierwszym punktem tworzonej otoczki to zakończ działanie.
5. Jeśli po m iteracjach nie utworzono otoczki, to przyjmuj nową aproksymację m.

Algorytm Chana- szczegóły

Optymalne aproksymacje wartości m dane są wzorem $m = 2^{2^t}, dla\ t = 1, 2, 3 \dots$. Algorytm kończy działanie dla iteracji w której $m \geq h$.

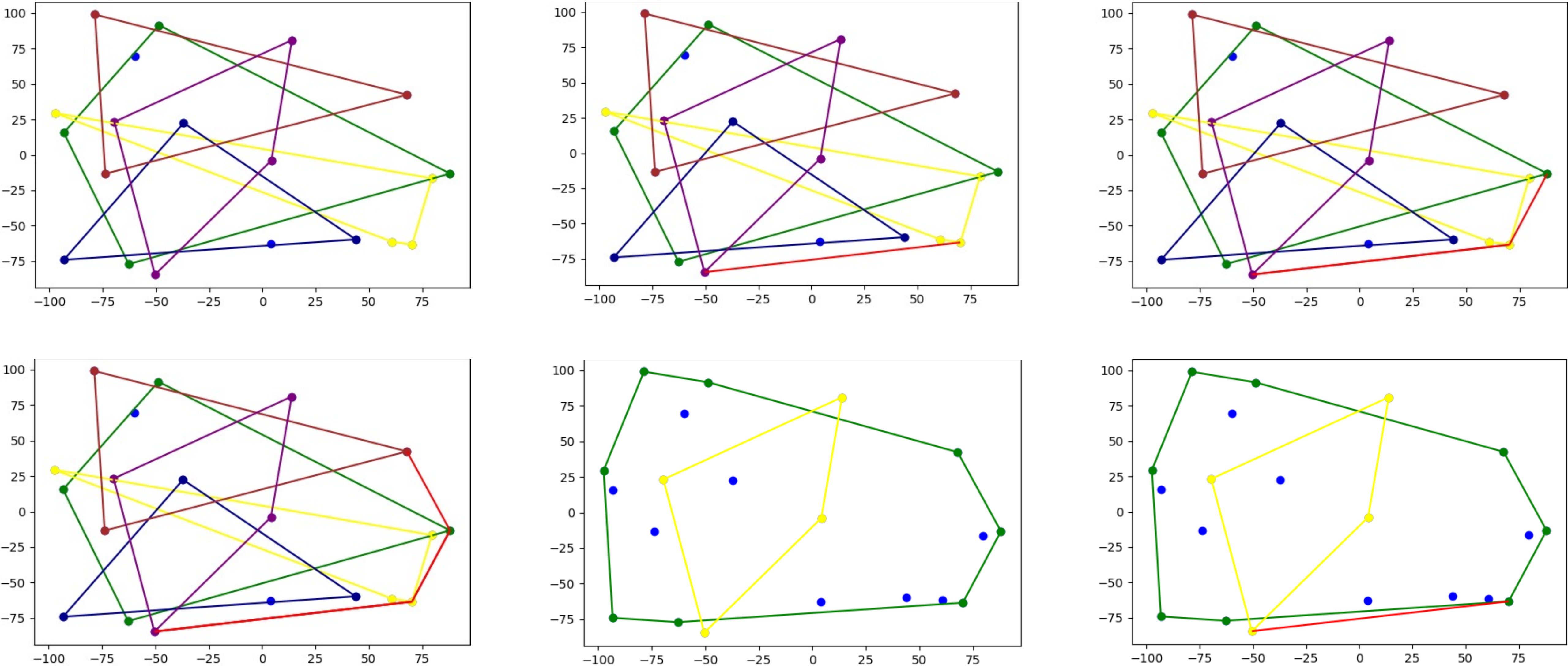
Dowód:

Niech $m = 2^{2^s}$ to aproksymacja, dla której algorytm znalazł otoczkę wypukłą. Z tego wynika, że ostatnia aproksymacja $m = 2^{2^{s-1}} < h$ i jednocześnie $2^{s-1} < \log h$.

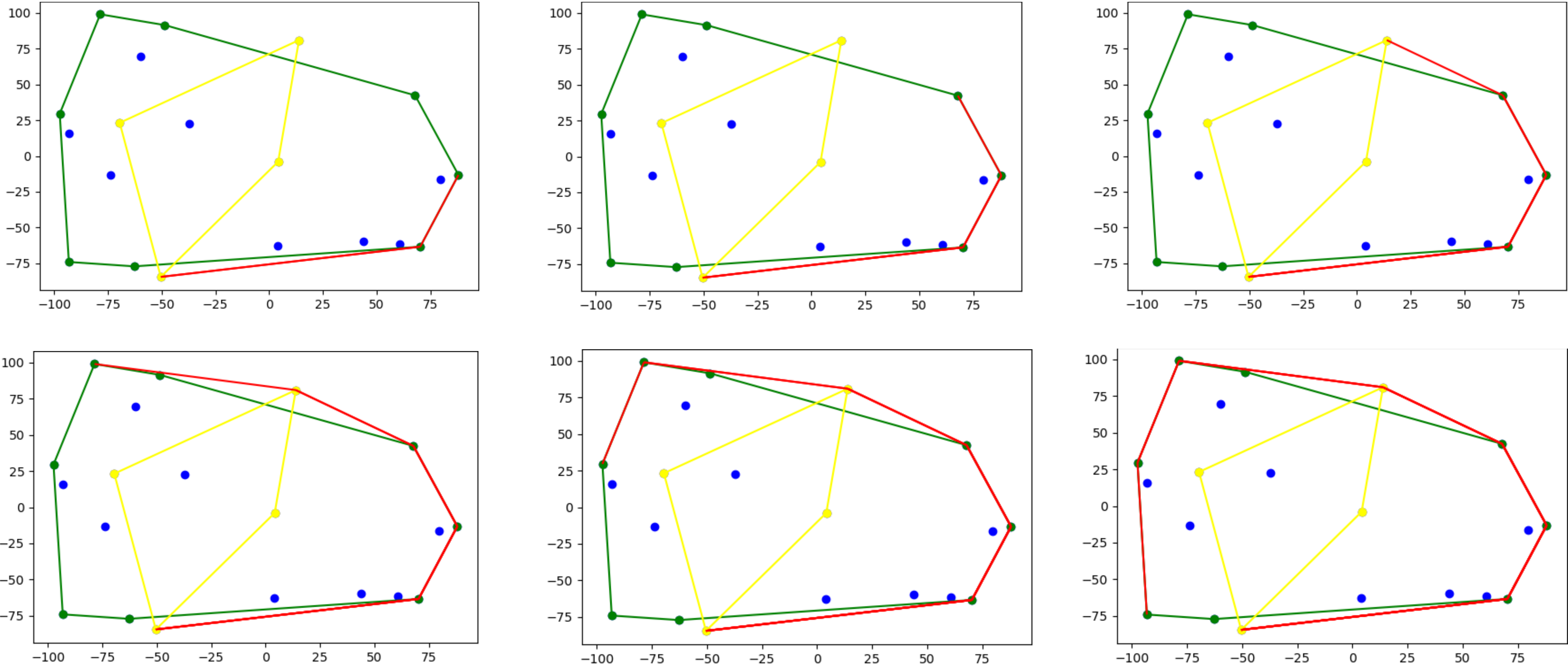
$$\sum_{i=1}^{s} n \log 2^{2^i} = \sum_{i=1}^{s} n 2^i = n(2^{s+1} - 2) < 4n \log h = O(n \log h)$$

Wyznaczanie stycznych z punktu do wszystkich otoczek działa na tej samej zasadzie, co wyznaczanie prawej stycznej w algorytmie przyrostowym.

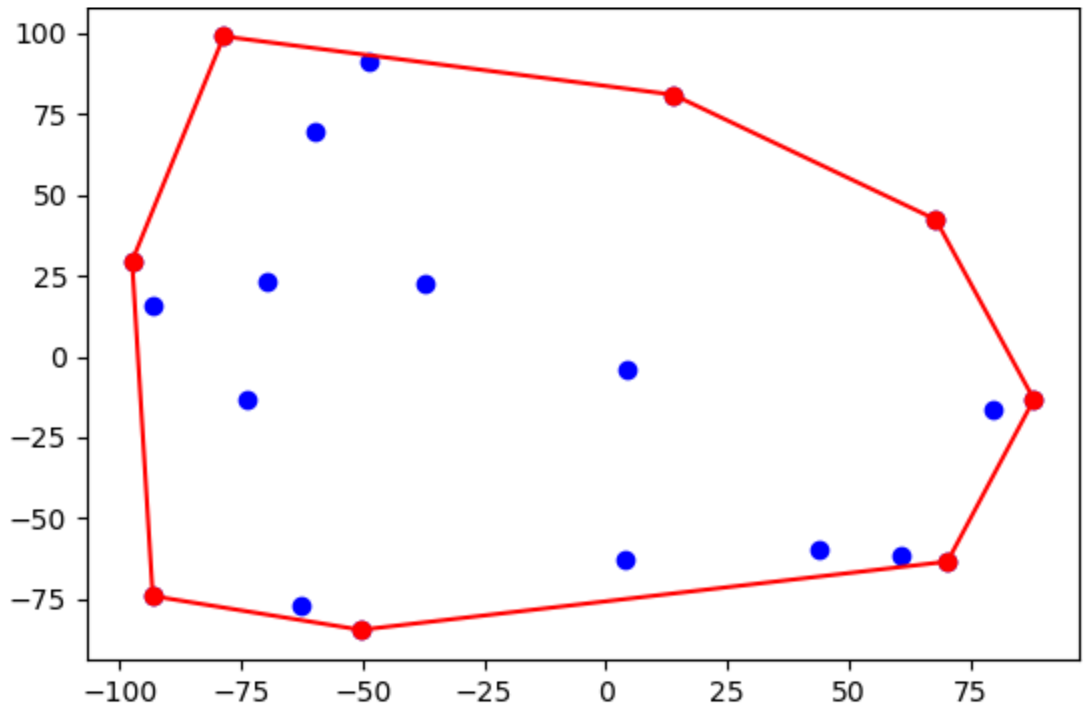
Przebieg algorytmu Chana



Przebieg algorytmu Chana



Przebieg algorytmu Chana



Porównanie algorytmów

- Algorytm dziel i rządź jest całkowicie różny od algorytmów Grahama i Jarvisa. Algorytmy te polegają w głównej mierze na znajdowaniu kolejnych punktów ostatecznej otoczki wypukłej. Algorytm dziel i rządź polega natomiast w dużym stopniu na rekurencyjnym scalaniu dwóch mniejszych otoczek w jedną większą.
- Algorytm przyrostowy przejawia pewne podobieństwo do algorytmu Jarvisa. W algorytmie Jarvisa, mając tworzoną otoczkę wypukłą, szukany jest kolejny punkt do dodania do otoczki. W algorytmie przyrostowym, mając kolejny punkt dodawany do otoczki i samą otoczkę, szukane są punkty otoczki, które należy połączyć z nowo dodawanym punktem ze zbioru.

Porównanie algorytmów

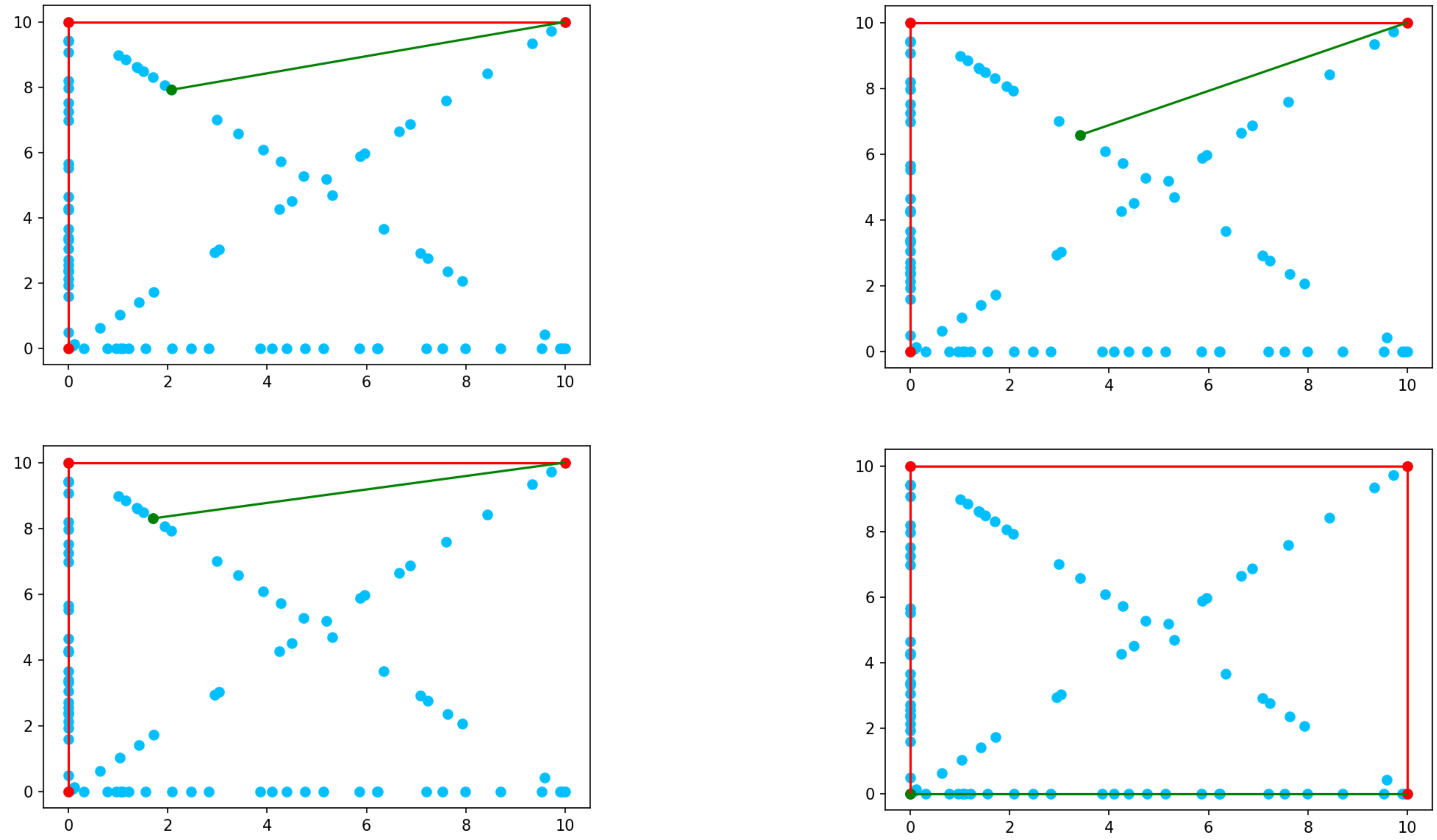
- Algorytm Chana jest w dużym stopniu zależy zarówno od algorytmu Grahama jak i algorytmu Jarvisa. Algorytm ten dzieli zbiór punktów na pomniejsze zbiory, dla których otoczki obliczane są bezpośrednio algorytmem Grahama. Podobieństwo do algorytmu Jarvisa przejawia się podobnie jak w przypadku algorytmu przyrostowego – szukana jest styczna do obecnego punktu maksymalizująca kąt w otoczce.

Algorytm Jarvisa (owijanie prezentu)

Przebieg działania algorytmu:

1. Znajdź punkt z S o najmniejszej współrzędnej y, jest to początkowy punkt i obecnie rozpatrywany.
2. Znajdź punkt, dla którego kąt liczony przeciwnie do wskazówek zegara w odniesieniu do ostatniej krawędzi otoczki jest najmniejszy.
3. Do obliczeń użyj wyznacznika.
4. Zwróć krawędź obecnego punktu i znalezionego punktu jako krawędź otoczki. Ustaw znaleziony punkt jako obecny punkt.
5. Powtarzaj szukanie dopóki nie wrócisz do punktu początkowego (znaleziony punkt nie będzie punktem początkowym). (wykład)

Przebieg algorytmu Jarvisa

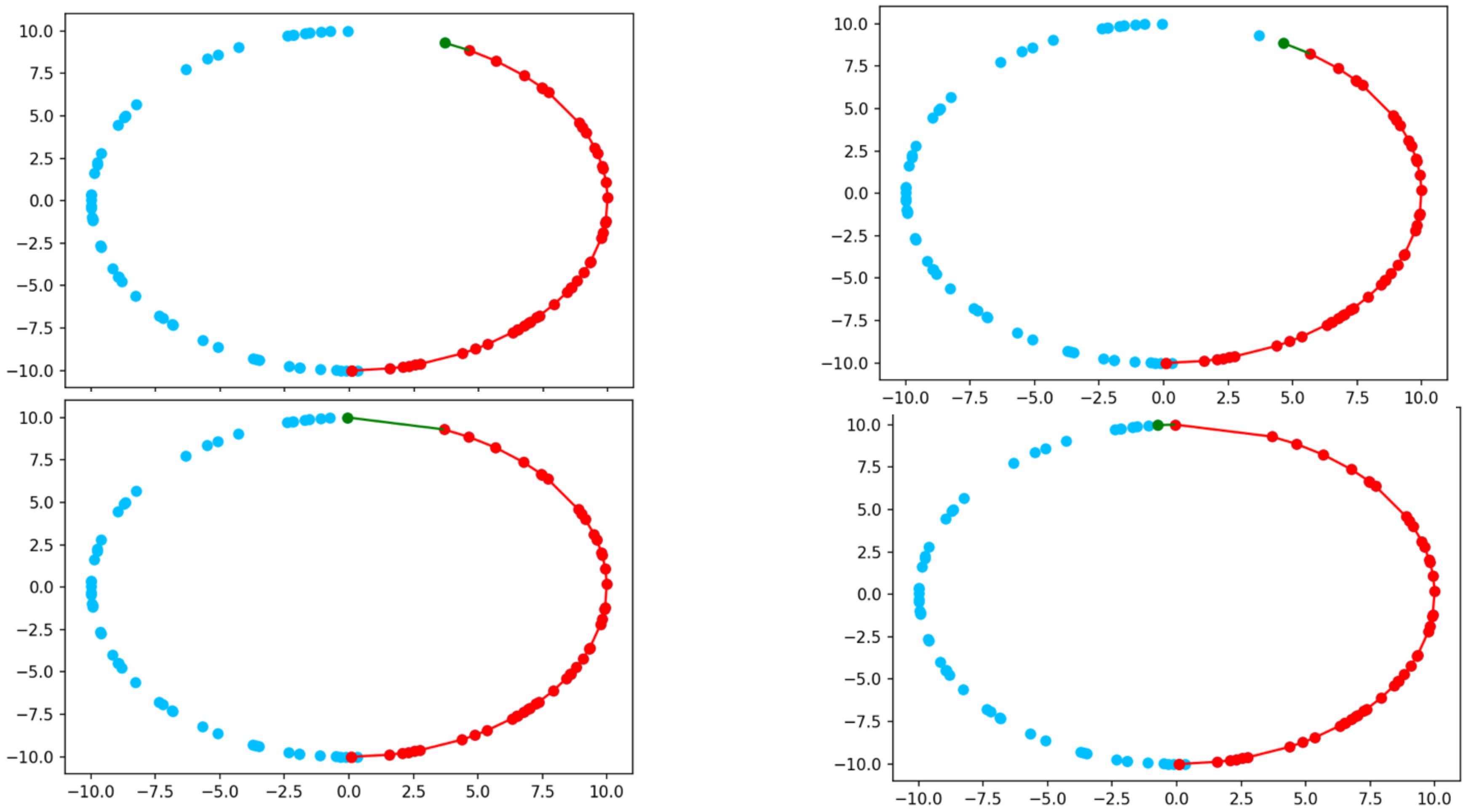


Algorytm Grahama

Przebieg działania algorytmu:

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. 3. Do początkowo pustego stosu S wkładamy punkty p_0, p_1, p_2. t to indeks stosu, za i podstaw 3,
4. Dopóki i jest mniejsze od m powtarzaj:
5. Jeżeli p_i leży na lewo od prostej wyznaczonej przez punkty p_t-1 i p_t to dodaj na stos punkt p_i oraz zwiększ i o 1, w przeciwnym wypadku usuń punkt na stosie. (wykład)

Przebieg algorytmu Grahama



Zbiory punktów użyte do testowania algorytmów

Zbiory punktów:

- Zbiór A – 100 losowo wybranych punktów o współrzędnych z zakresu [-100;100]
- Powiększony zbiór A - 100000 losowo wybranych punktów o współrzędnych z zakresu [-100;100]
- Zbiór B – 100 losowo wybranych punktów leżących na okręgu o promieniu 10 i środku w punkcie (0;0)
- Powiększony zbiór B - 100000 losowo wybranych punktów leżących na okręgu o promieniu 1000 i środku w punkcie (0;0)

Zbiory punktów użyte do testowania algorytmów

- Zbiór C – 100 losowo wybranych punktów na bokach kwadratu o wierzchołkach: (-10;10), (-10;-10), (10; -10), (10;10).
- Powiększony zbiór C – 100000 losowo wybranych punktów na bokach kwadratu o wierzchołkach: (-1000; 1000), (-1000; -1000), (1000; -1000), (1000; 1000).
- Zbiór D – po 25 punktów leżących na osiach x i y oraz po 20 punktów leżących na przekątnych kwadratu o wierzchołkach: (0;0), (10;0), (10;10), (0;10) (wraz z tymi wierzchołkami).
- Powiększony zbiór D – po 100000 punktów leżących na osiach x i y oraz po 1000 punktów leżących na przekątnych kwadratu o wierzchołkach: (0;0), (1000;0), (1000;1000), (0;1000) (wraz z tymi wierzchołkami).

Porównanie wydajności algorytmów

Czasy wykonania:

Algorytm / zbiór	Zbiór A	Powiększony zbiór A	Zbiór B	Powiększony zbiór B	Zbiór C	Powiększony zbiór C	Zbiór D	Powiększony zbiór D
Dziel i rządź	0 s	1.18 s	0 s	1.573 s	0 s	1.134 s	0 s	2.461 s
Przyrostowy	0.005 s	11.49 s	0.01 s	57.293 s	0 s	13.499 s	0.005 s	75.801 s
Quickhull	0.005 s	0.43 s	0 s	3.925 s	0 s	0.762 s	0 s	1.081 s
Górna i dolna otoczka	0 s	0.44 s	0 s	0.348 s	0 s	0.401 s	0 s	0.769 s
Chana	0 s	3. 803 s	0 s	10.768 s	0 s	1.15 s	0 s	2.282 s
Jarvis	0 s	3.28 s	0.016 s	???	0 s	0.686 s	0 s	1.449 s
Graham	0 s	1.333 s	0 s	1.285 s	0 s	2.249 s	0 s	8.279 s

Dziękujemy za uwagę!

Źródła:

1. Wykłady oraz laboratoria z przedmiotu „Algorytmy geometryczne” na 3. semestrze studiów na kierunku Informatyka w AGH w Krakowie, dr inż. Barbara Głut
2. „Computational Geometry: Algorithms and Applications”, Mark de Berg
3. “Wprowadzenie do algorytmów”, Thomas H. Cormen
4. Wikipedia:

a. https://en.wikipedia.org/wiki/Convex_hull
b. https://en.wikipedia.org/wiki/Convex_hull_algorithms c.
https://en.wikipedia.org/wiki/Gift_wrapping_algorithm d.
https://en.wikipedia.org/wiki/Graham_scan
e. https://en.wikipedia.org/wiki/Quickhull
f. https://en.wikipedia.org/wiki/Chan%27s_algorithm