Otoczka wypukła dla zbioru punktów w przestrzeni dwuwymiarowej

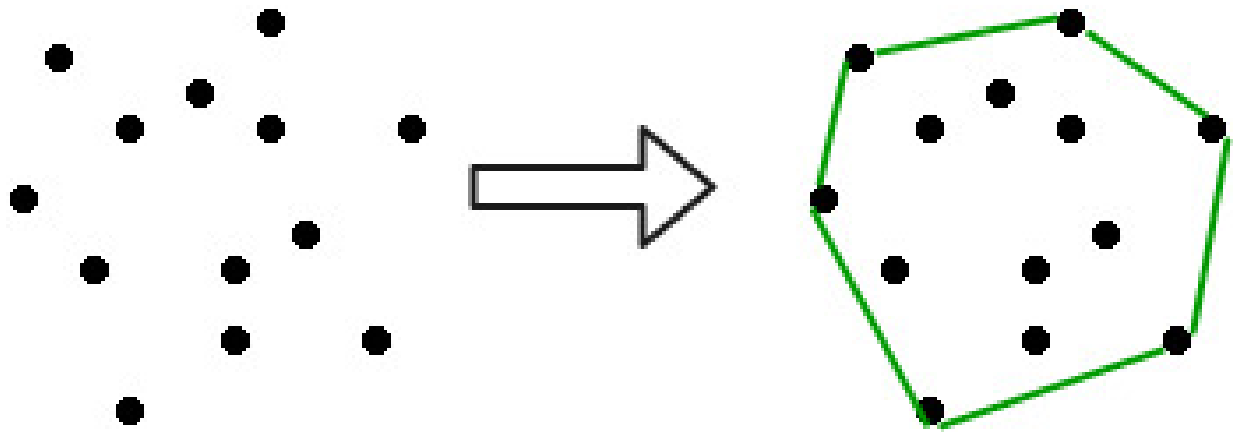
Projekt

Antoni Szczepański

Patryk Studziński

Otoczka wypukła – co to jest?

- Otoczka wypukła zbioru punktów Q to najmniejszy wypukły wielokąt taki, że każdy punkt ze zbioru Q leży albo na brzegu wielokąta albo w jego wnętrzu.



Zaimplementowane przez nas algorytmy do wyznaczania otoczki wypukłej:

- Graham
- Jarvis
- Przyrostowy
- Dolna i górna otoczka
- QuickHull
- Dziel i rządź
- Chan’a

Algorytm Grahama

Opis algorytmu:

1. W zbiorze danych wybieramy punkt p0, o najmniejszej współrzędnej y, w przypadku gdy jest ich więcej to wybieramy punkt o najmniejszej współrzędnej x.

2. Sortujemy pozostałe punkty ze względu na kąt, jaki tworzy wektor (p0,p) z dodatnim kierunkiem osi x. Jeśli kilka punktów tworzy ten sam kąt, usuwamy wszystkie z wyjątkiem najbardziej oddalonego od p0.

3. Tworzymy stos, na którym na początku umieszczamy 2 pierwsze punkty (p_0, i punkt z którym p_0 tworzy najmniejszy kąt)

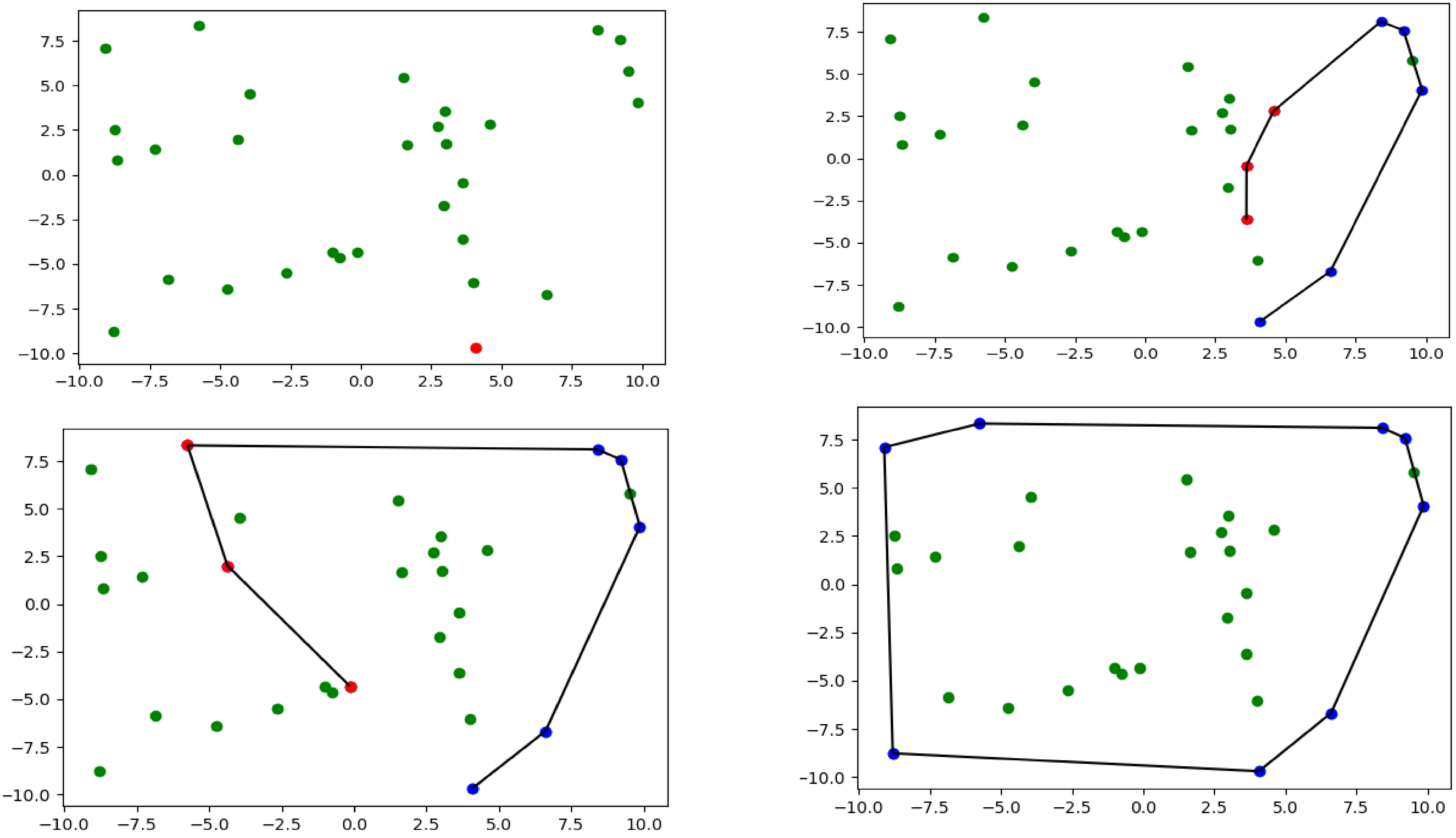
4. Przechodzimy po kolei po wszystkich punktach i dla każdego z nich:

- W przypadku, gdy skręcamy w lewo to wstawiamy aktualny wierzchołek do stosu. Przechodzimy do następnego punktu.
- W przypadku, gdy jest on współliniowy to zdejmujemy ostatni wierzchołek ze stosu i wstawiamy aktualny. Przechodzimy do następnego punktu.
- W przypadku, gdy skręcamy w prawo, dopóki będziemy skręcać w prawo usuwamy ostatni wierzchołek ze stosu.

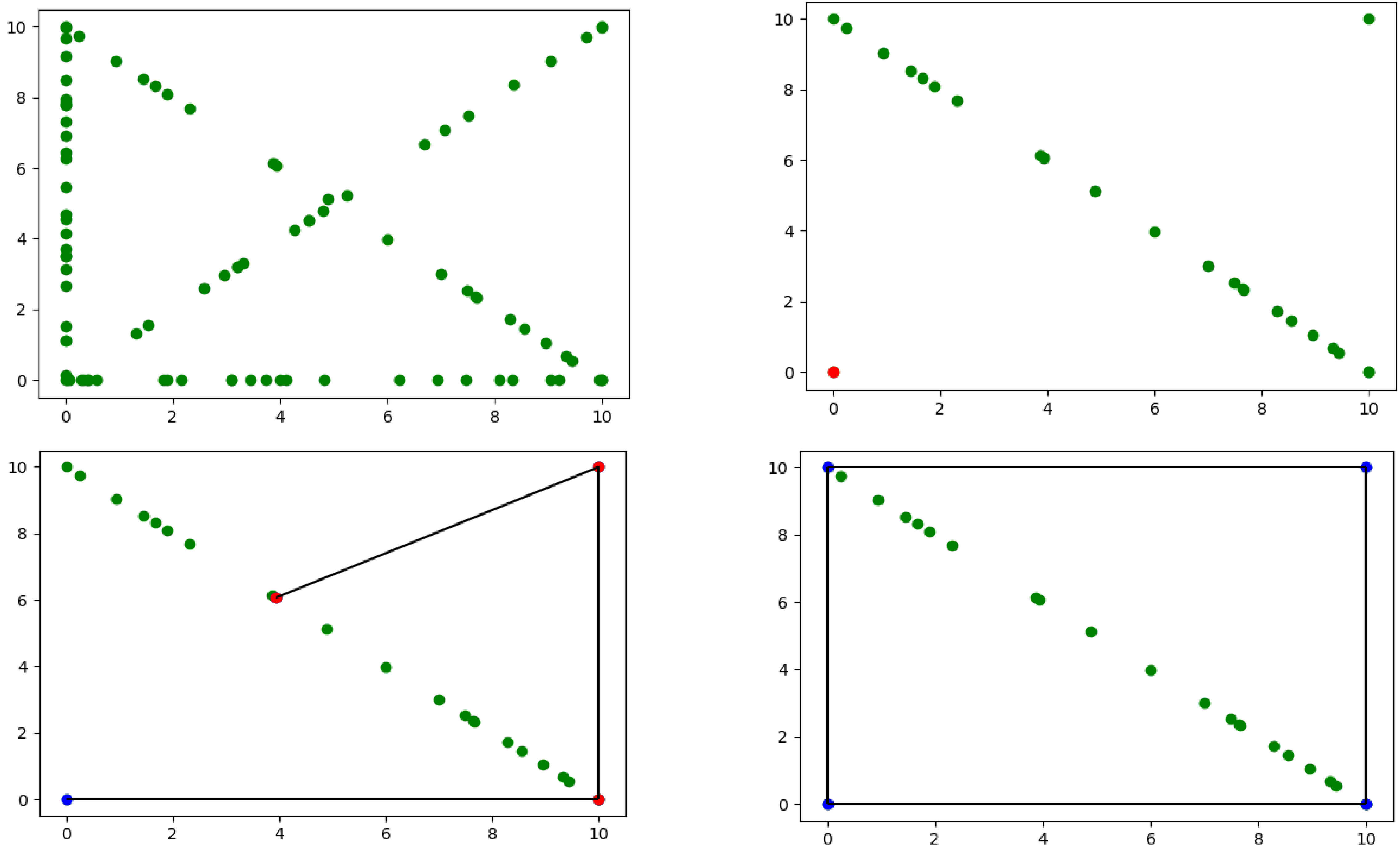
5. Punkty, które pozostaną na stosie, tworzą otoczkę wypukłą.

Do wyznaczeniu położenia punktu względem prostej (kierunku skrętu) używamy funkcji orient, która:

- zwraca 0, gdy punkty są współliniowe
- zwraca -1, gdy punkt leży po lewej stronie prostej
- zwraca 1, gdy punkt leży po prawej stronie prostej



7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−10.0 −7.5 −5.0 −2.5 0.0 2.5 5.0 7.5 10.0

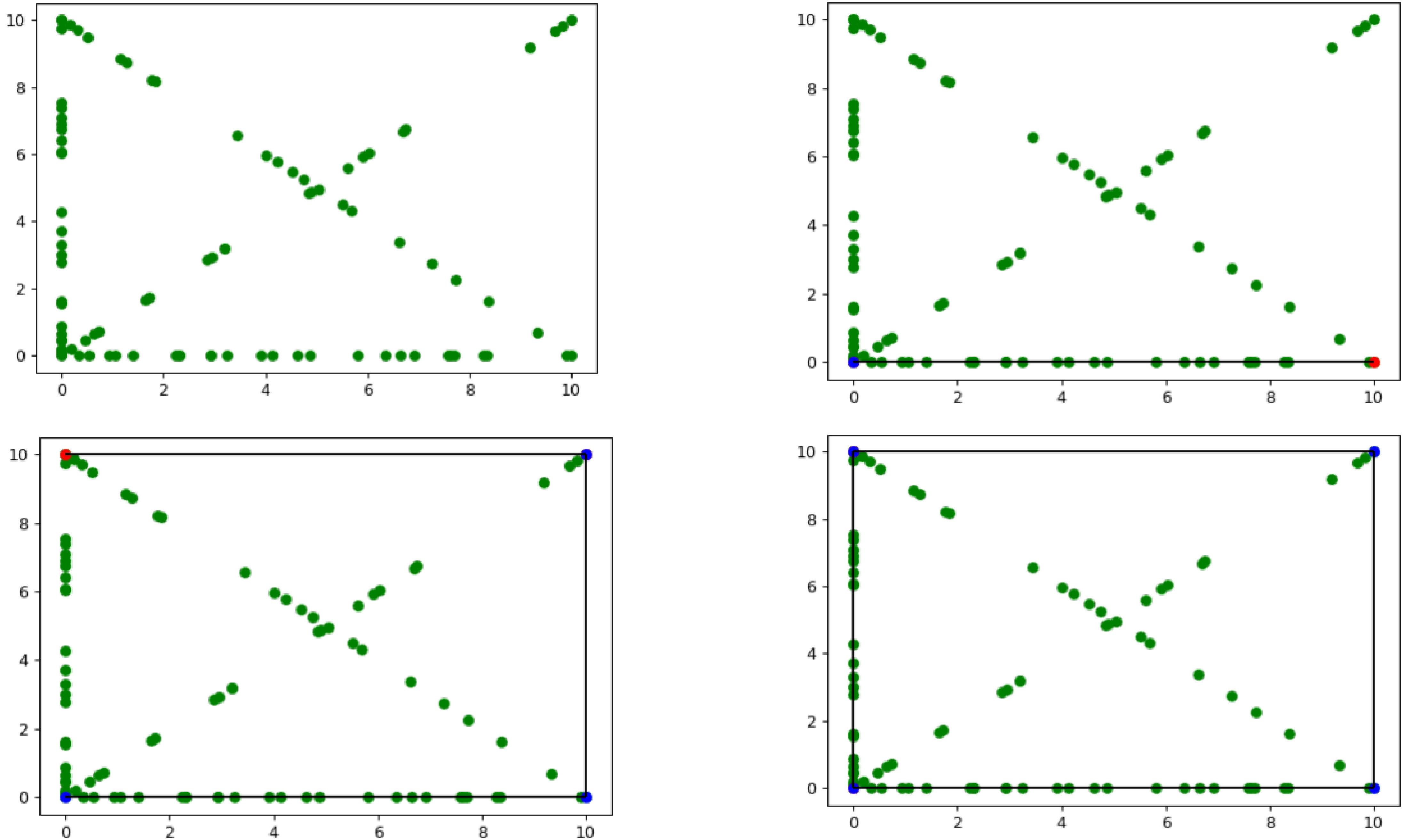


0
2
4
6
8
10

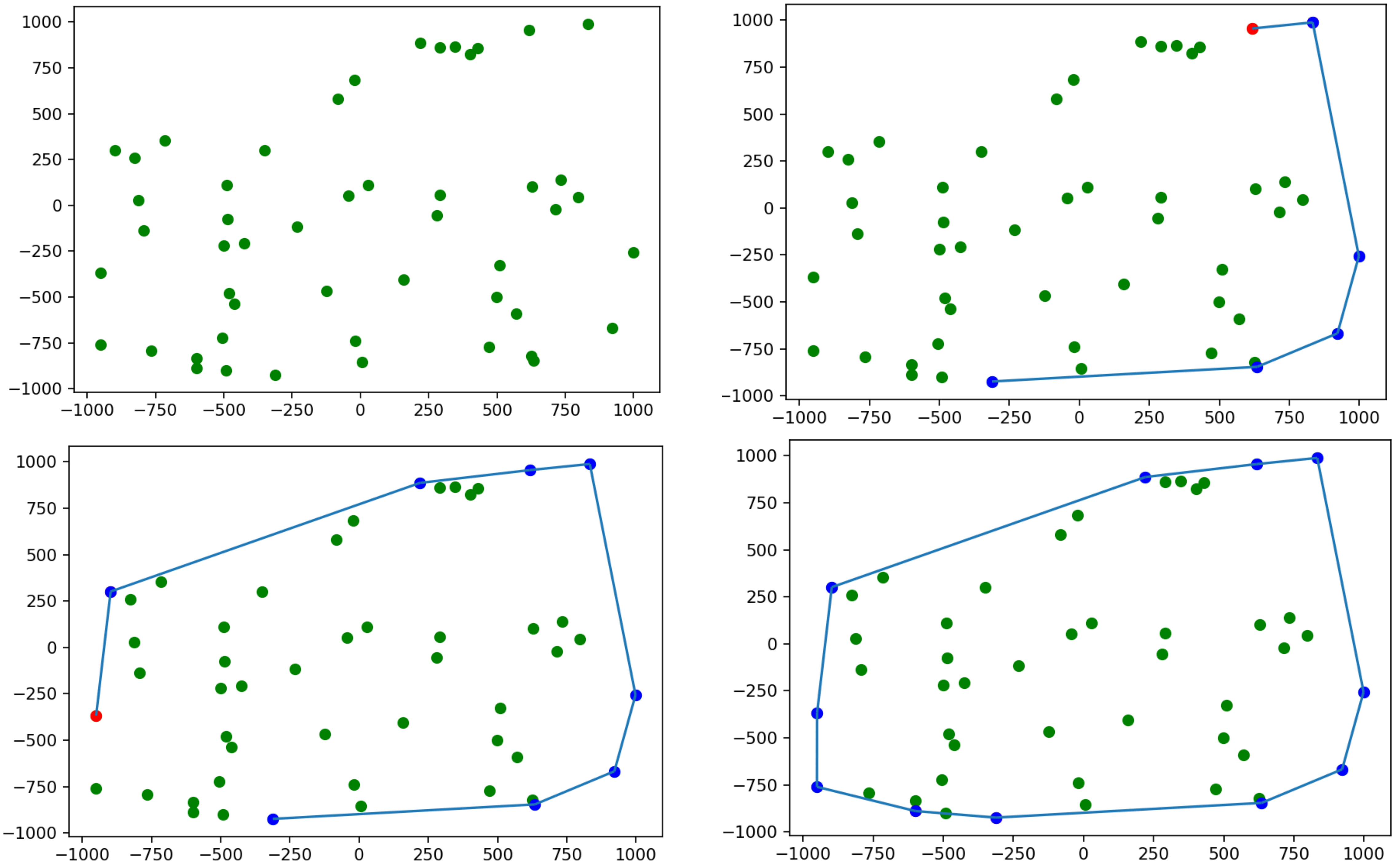
Algorytm Jarvisa

Opis algorytmu:

1. W zbiorze danych wybieramy punkt p0, o najmniejszej współrzędnej y, w przypadku gdy jest ich więcej to wybieramy punkt o najmniejszej współrzędnej x.
2. Następnie znajdujemy punkt, którego kąt jaki tworzy wektor (p0,p) z dodatnim kierunkiem osi x jest najmniejszy, w przypadku współliniowych punktów wybieramy najdalszy. Krawędź (p0,p) należy do otoczki.
3. Dla ostatnio znalezionej krawędzi szukamy punktu (nie sprawdzamy punktów, które już są w otoczce z wyjątkiem p0), dla którego kąt, jaki tworzy ta krawędź z punktem jest najmniejszy. Nowa krawędź należy do otoczki. Powtarzamy aż znalezionym punktem będzie punkt p0.
4. Znalezione krawędzie to krawędzie otoczki, a znalezione punkty to wierzchołki otoczki.



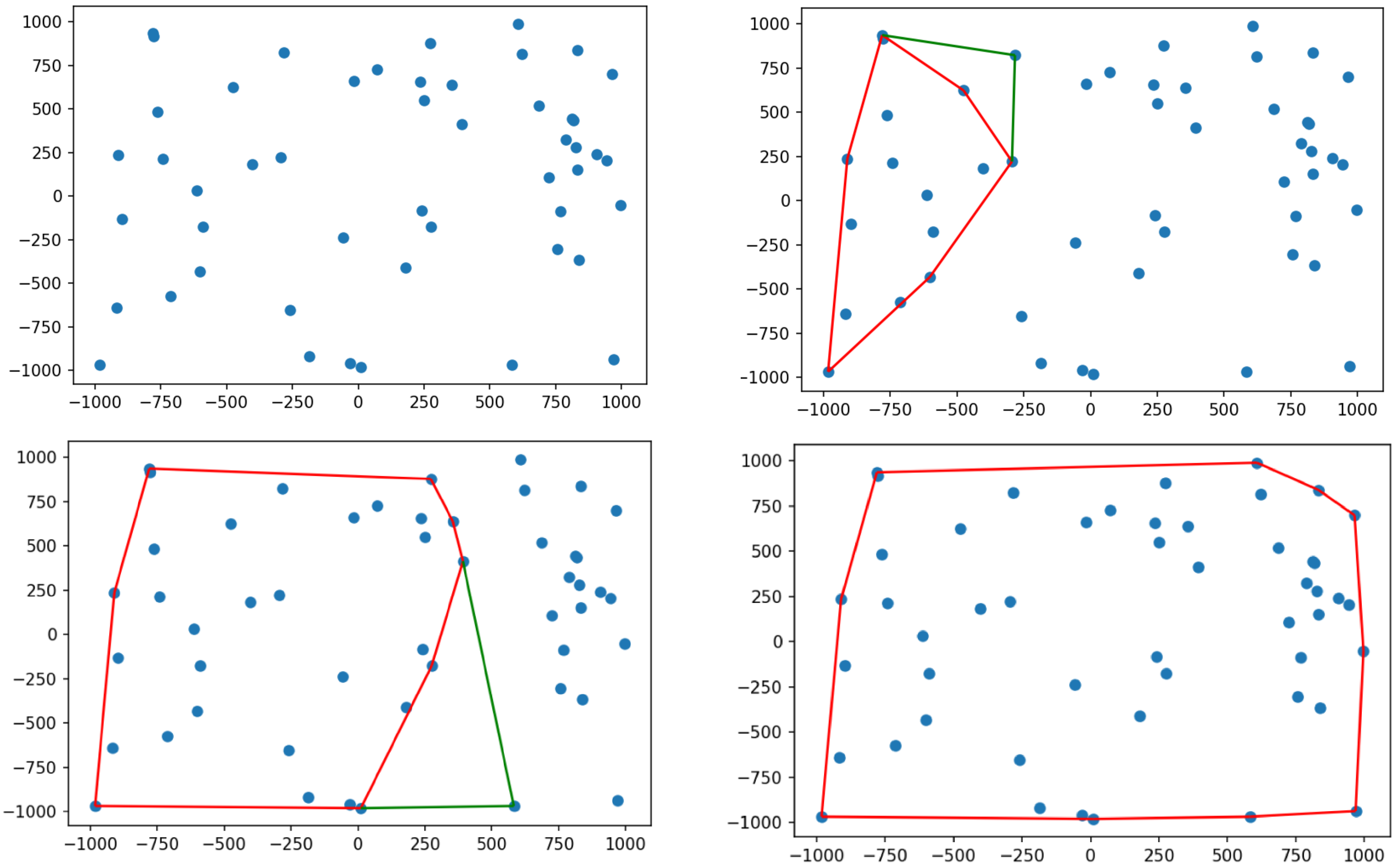
10
8
6
4
2
0
0 2 4 6 8 10
10
8
6
4
2
0
0 2 4 6 8 10
10
8
6
4
2
0
0 2 4 6 8 10
10
8
6
4
2
0
0 2 4 6 8 10


1000
750
500
250
0
−250
−500
−750
−1000
−1000 −750 −500 −250 0 250 500 750 1000

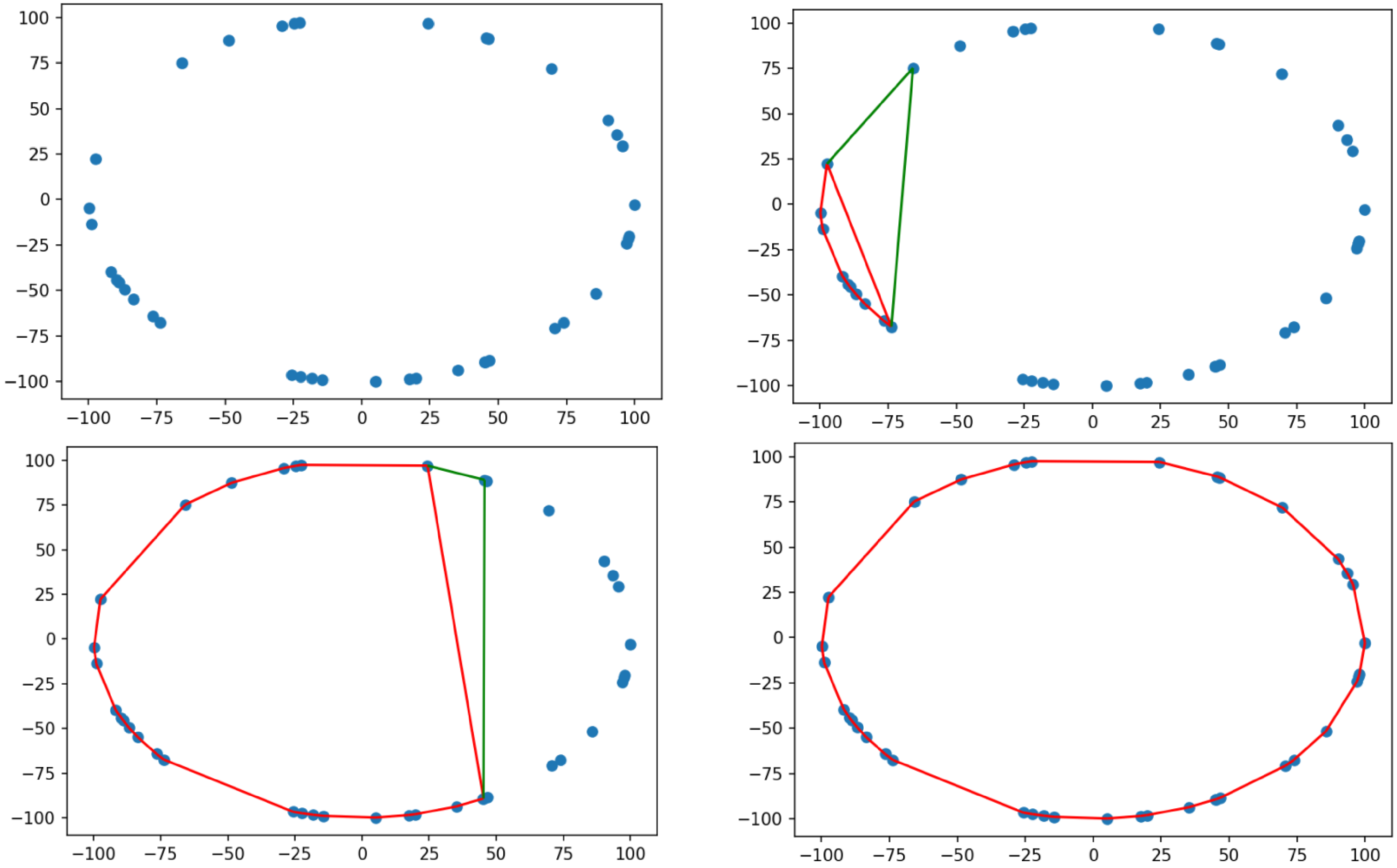
Algorytm Przyrostowy

Opis algorytmu:

1. Sortujemy punkty rosnąco względem współrzędnej x (każdy następny punkt jest na zewnątrz aktualnej otoczki).
2. Usuwamy punkty współliniowe o tej samej współrzędnej x (oprócz najwyższego i najniższego).
3. Dodajemy pierwsze 3 punkty do otoczki.
4. Przechodząc po kolejnych punktach znajdujemy styczne do otoczki i aktualizujemy otoczkę (dodajemy aktualnie rozpatrywany punkt i usuwamy punkty które leżą wewnątrz).



1000
750
500
250
0
−250
−500
−750
−1000
−1000 −750 −500 −250 0 250 500 750 1000



100
75
50
25
0
−25
−50
−75
−100
−100
−75
−50
−25
0
25
50
75
100

Górna i dolna otoczka

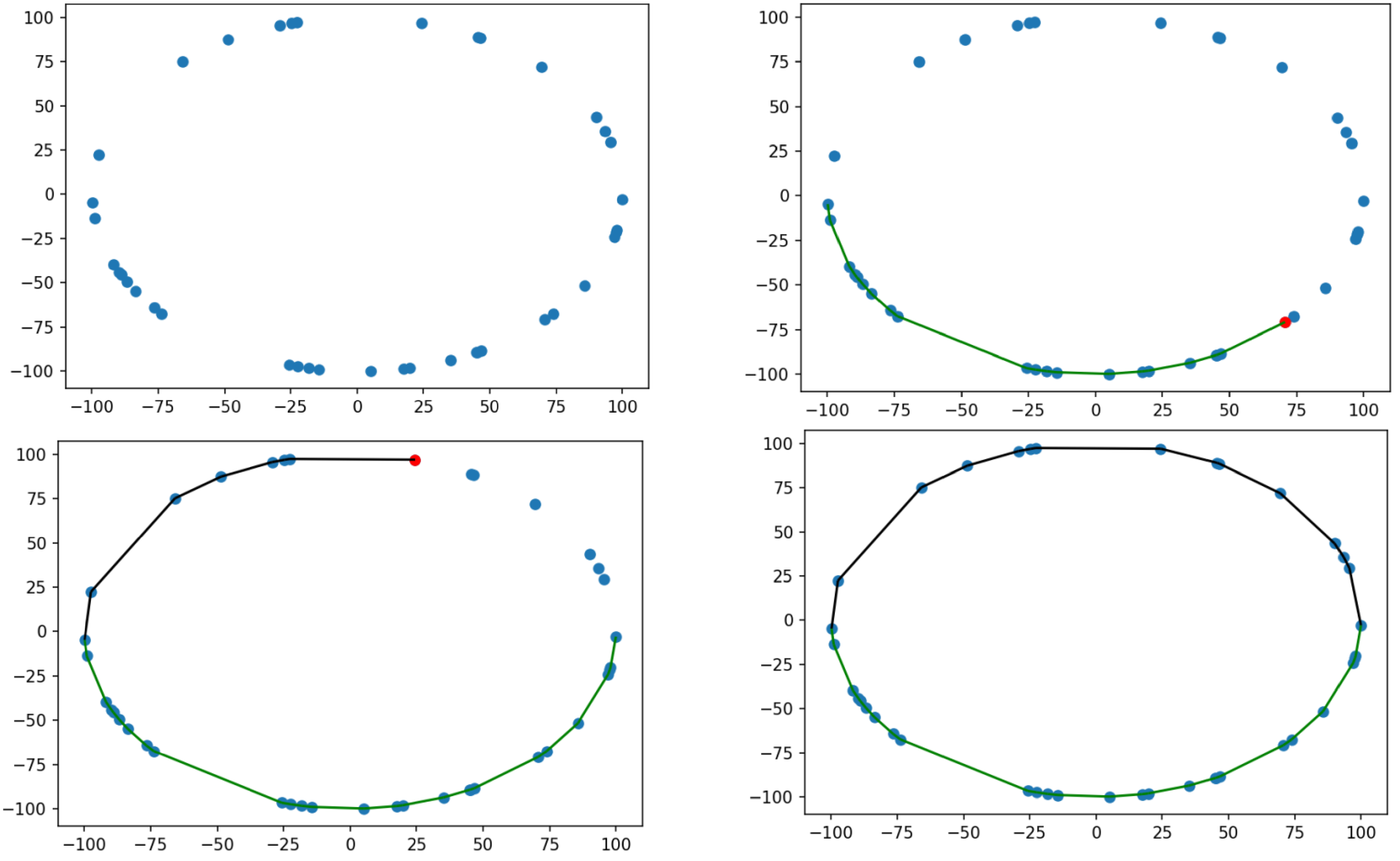
Opis algorytmu:

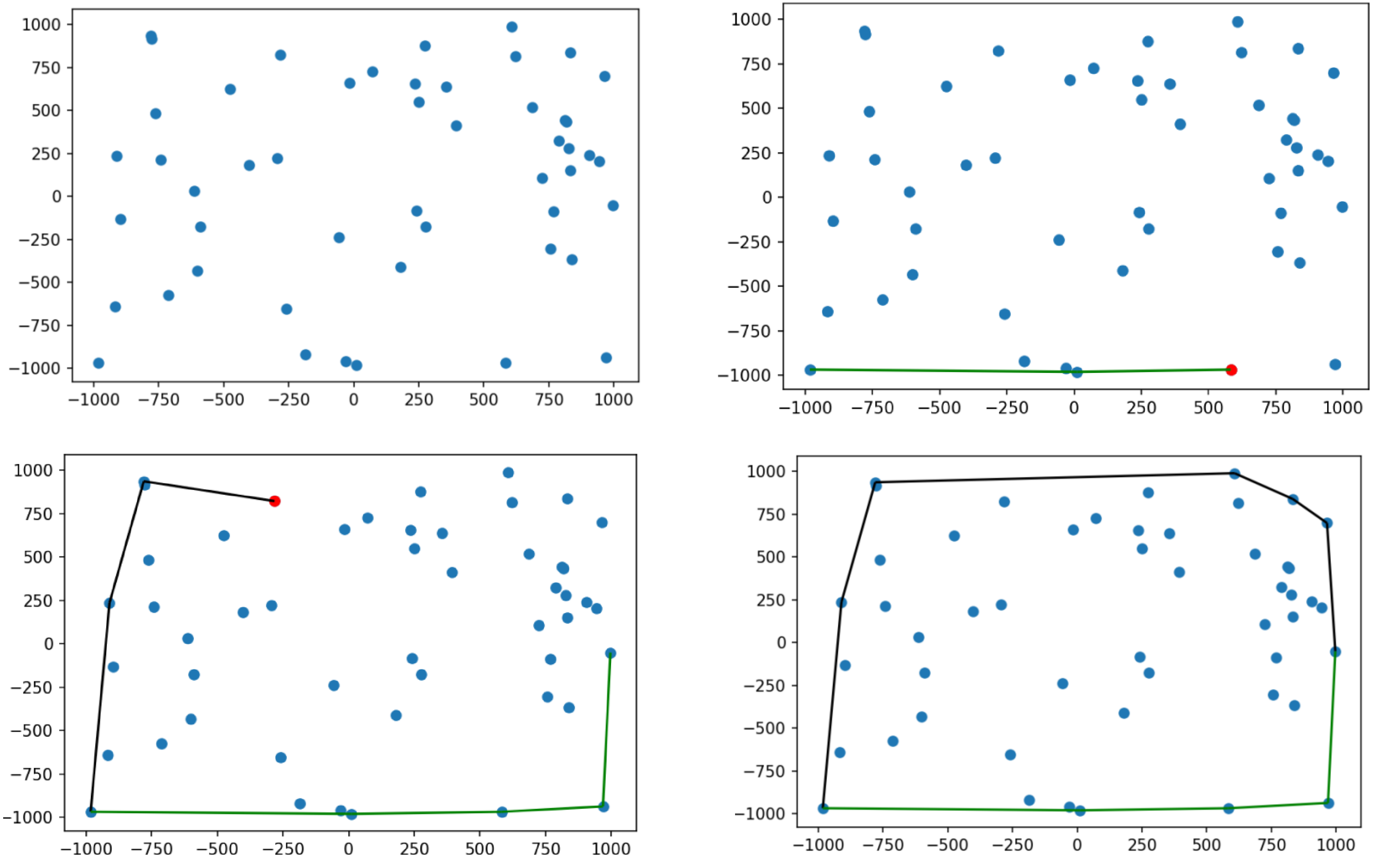
Będziemy wyznaczać otoczkę górną a następnie otoczkę dolną

1. Sortujemy punkty względem współrzędnej x, a dla takich samych x sortujemy jeszcze po współrzędnej y
2. Wkładamy do stosu pierwsze 2 punkty
3. Przechodząc po kolejnych punktach:

- Jeżeli 2 ostatnie na stosie i aktualny punkt tworzą skręt w prawo wkładamy go na stos
- W przeciwnym przypadku dopóki wielkość stosu jest większa niż 1 oraz 2 ostatnie punkty na stosie i aktualny nie tworzą skrętu w prawo zdejmujemy ostatni punkt ze stosu. Wstawiamy punkt do stosu.

4. Analogicznie wykonujemy punkt 2 i 3 dla wyznaczenia otoczki dolnej, z tą różnicą, że warunki co do skrętu są odwrotne.

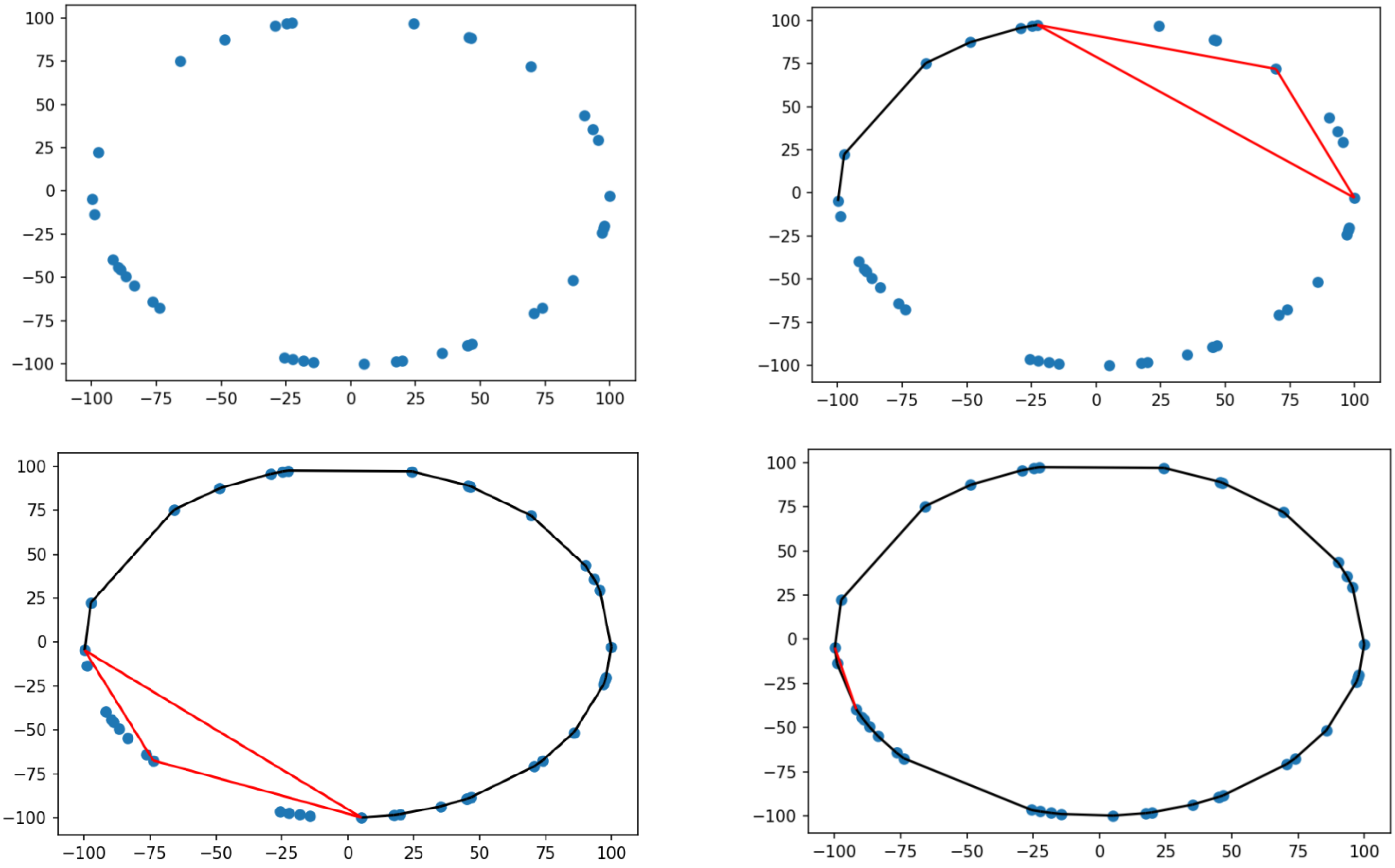

100 75 50 25 0 −25 −50 −75 −100
−100 −75 −50 −25 0 25 50 75 100
100 75 50 25 0 −25 −50 −75 −100
−100 −75 −50 −25 0 25 50 75 100
100 75 50 25 0 −25 −50 −75 −100
−100 −75 −50 −25 0 25 50 75 100
100 75 50 25 0 −25 −50 −75 −100
−100 −75 −50 −25 0 25 50 75 100

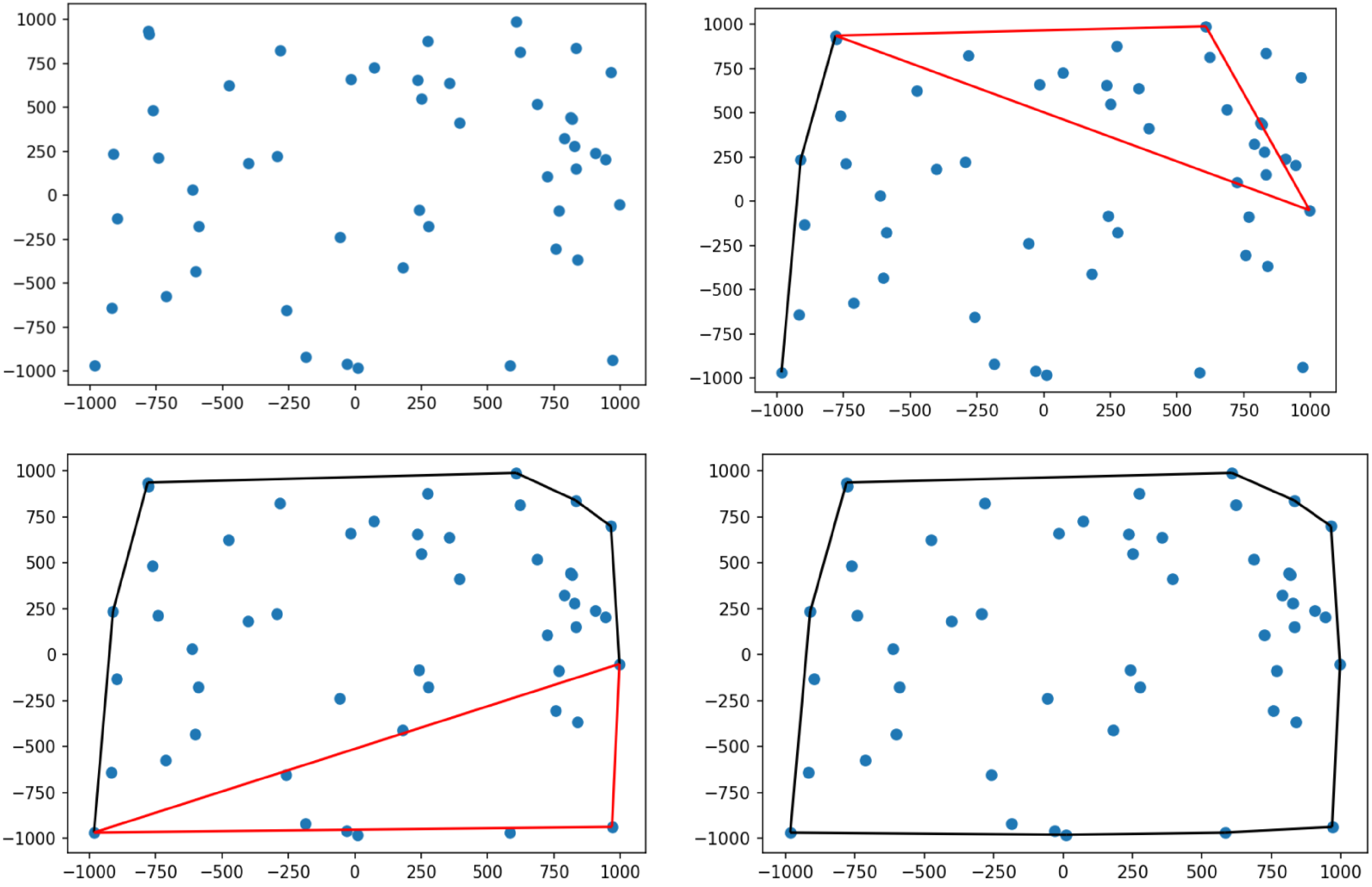

1000
750
500
250
0
−250
−500
−750
−1000
−1000 −750 −500 −250 0 250 500 750 1000

Algorytm QuickHull

Opis algorytmu:

1. Na starcie znajdujemy punkty o najmniejszej i największej współrzędnej x, znalezione punkty tworzą prostą
2. Dzielimy punkty na 2 zbiory, na prawo i na lewo od prostej
3. Dla każdego ze zbiorów znajdujemy punkt p0 najbardziej oddalony od prostej. Tworzymy trójkąt o wierzchołkach w końcach prostej i p0. Następnie dla 2 nowych krawędzi rekurencyjnie wykonujemy tą procedurę aż poza znalezionym trójkątem nie będzie żadnych punktów. Oznacza to, że wierzchołki znalezionych trójkątów należą do otoczki.


100
75
50
25
0
−25
−50
−75
−100
−100 −75 −50 −25 0 25 50 75 100


1000 750 500 250 0 −250 −500 −750 −1000
−1000 −750 −500 −250 0 250 500 750 1000
1000 750 500 250 0 −250 −500 −750 −1000
−1000 −750 −500 −250 0 250 500 750 1000
1000 750 500 250 0 −250 −500 −750 −1000
−1000 −750 −500 −250 0 250 500 750 1000
1000 750 500 250 0 −250 −500 −750 −1000
−1000 −750 −500 −250 0 250 500 750 1000

Algorytm dziel i rządź

Opis algorytmu:

1. Dopóki rozmiar dowolnego zbioru przekracza daną stałą k, dzielimy ten zbiór na 2 zbiory względem mediany x-owych współrzędnych punktów.
2. Wyznaczamy otoczkę każdego zbioru (w naszym algorytmie, wyznaczamy otoczkę algorytmem Grahama), a następnie łączymy otoczki, w kolejności odwrotnej do dzielenia zbiorów dopóki nie znajdziemy końcowej otoczki.

Procedura łączenia otoczek

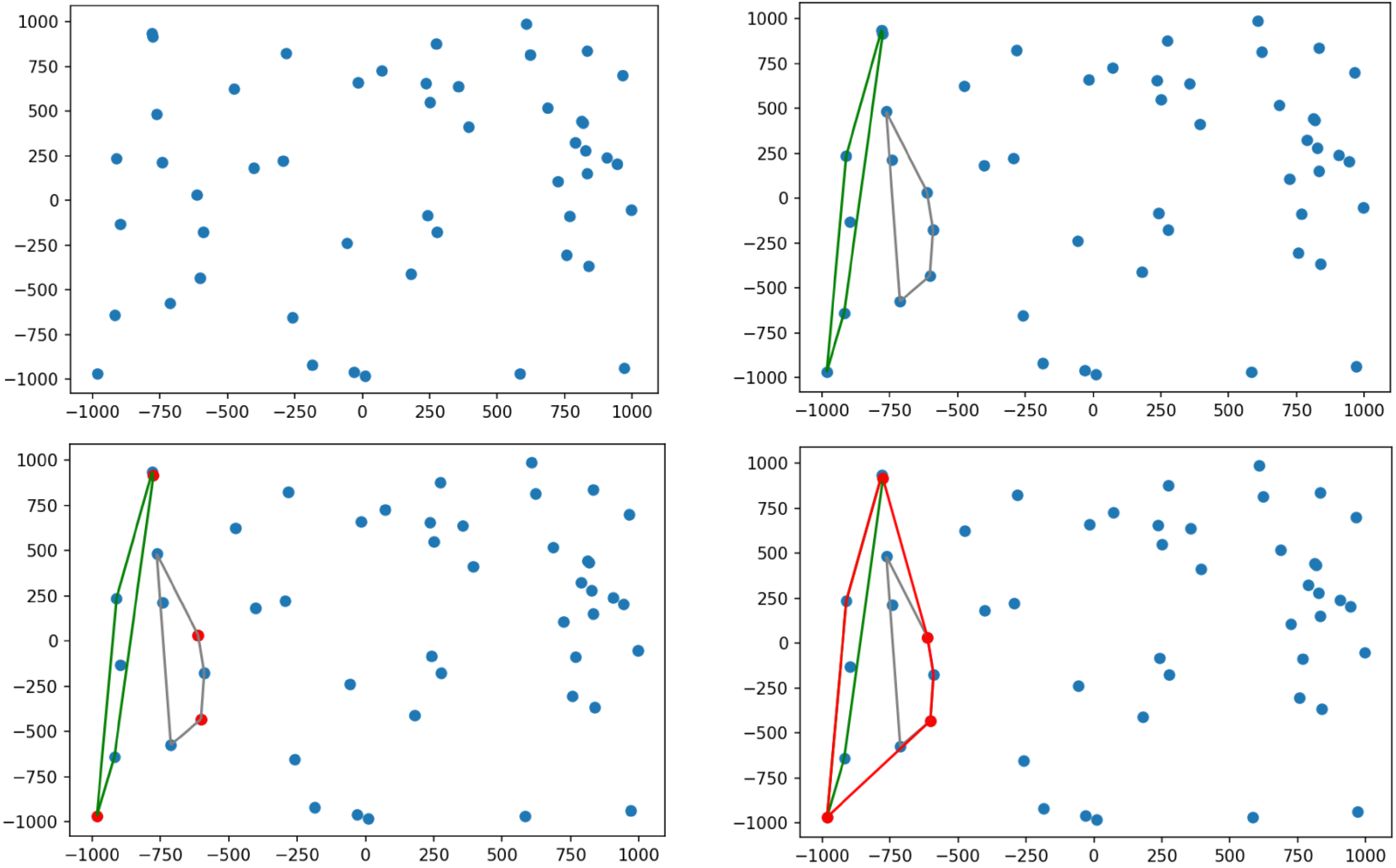
1. Dla lewej otoczki znajdujemy skrajnie prawy wierzchołek (a), a dla prawej otoczki, skrajnie lewy (b).
2. Znajdujemy górną styczną do obu otoczek w następujący sposób:

- Dopóki prosta(a,b) nie jest styczna do obu otoczek:

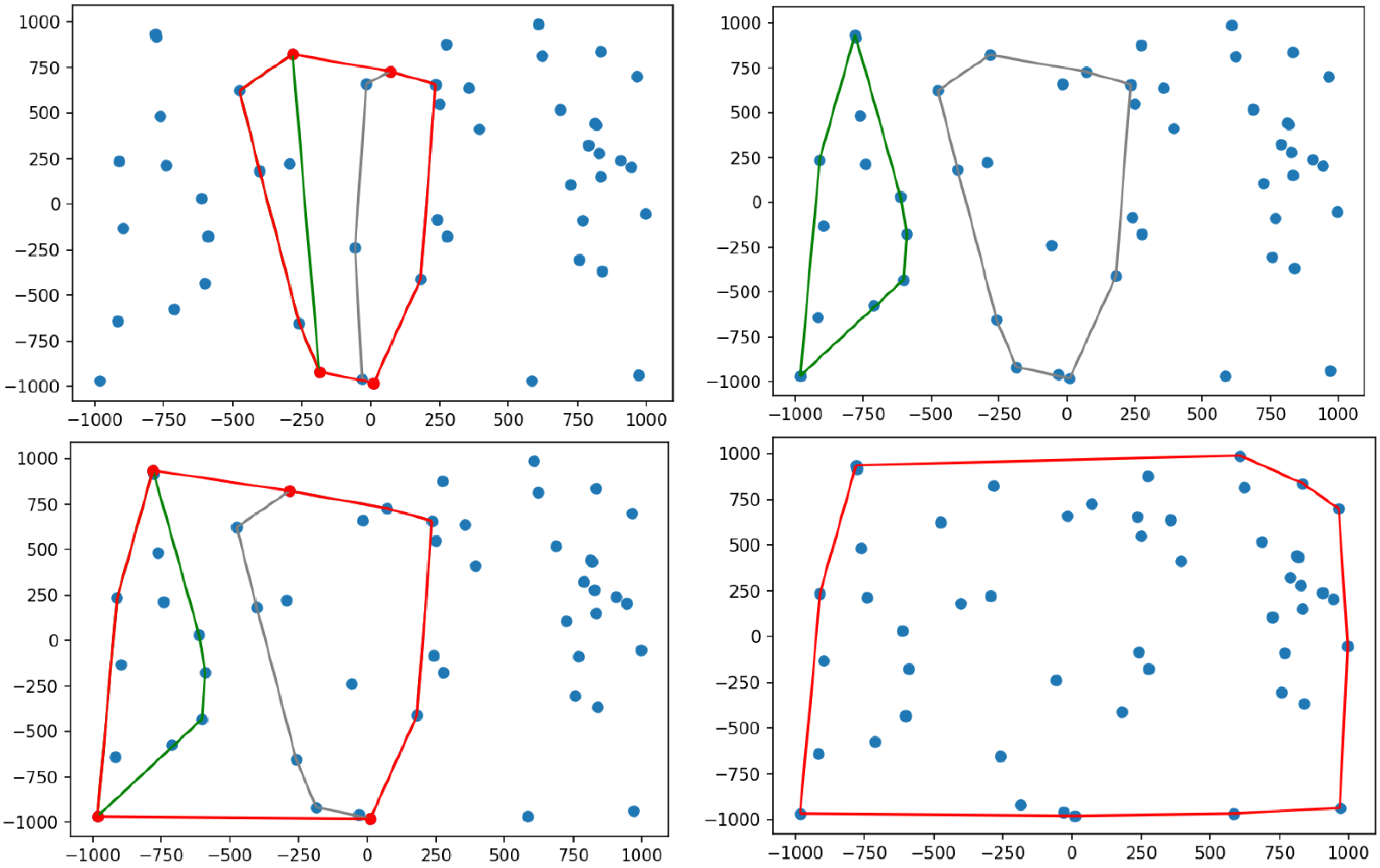
  - dopóki prosta (a,b) nie jest styczna do prawej otoczki, zmieniamy punkt b na jego wyższego sąsiada

  - dopóki prosta (a,b) nie jest styczna do lewej otoczki, zmieniamy punkt a na jego wyższego sąsiada

3. Analogicznie znajdujemy dolną styczną do obu otoczek



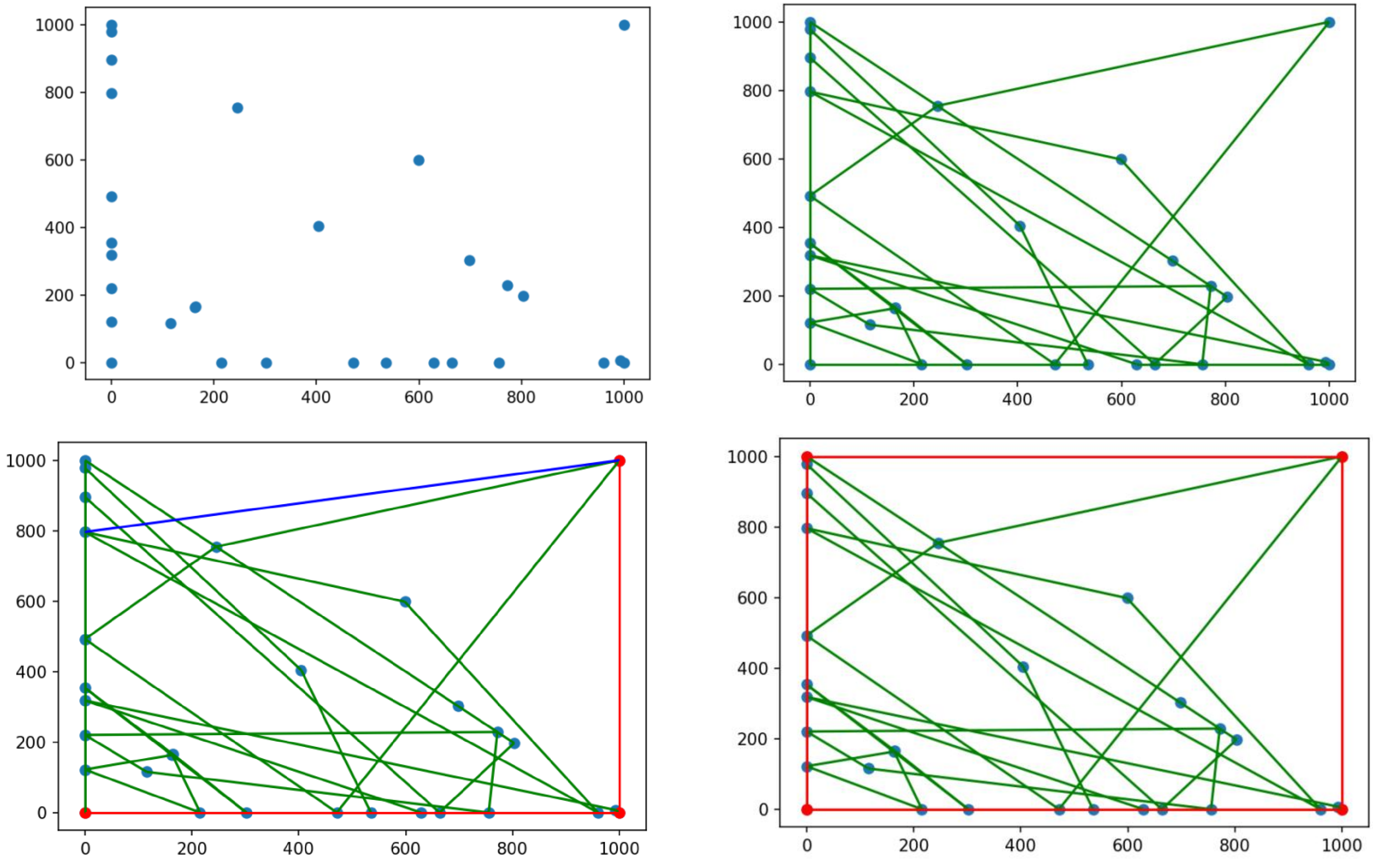
1000
750
500
250
0
−250
−500
−750
−1000
−1000 −750 −500 −250 0 250 500 750 1000

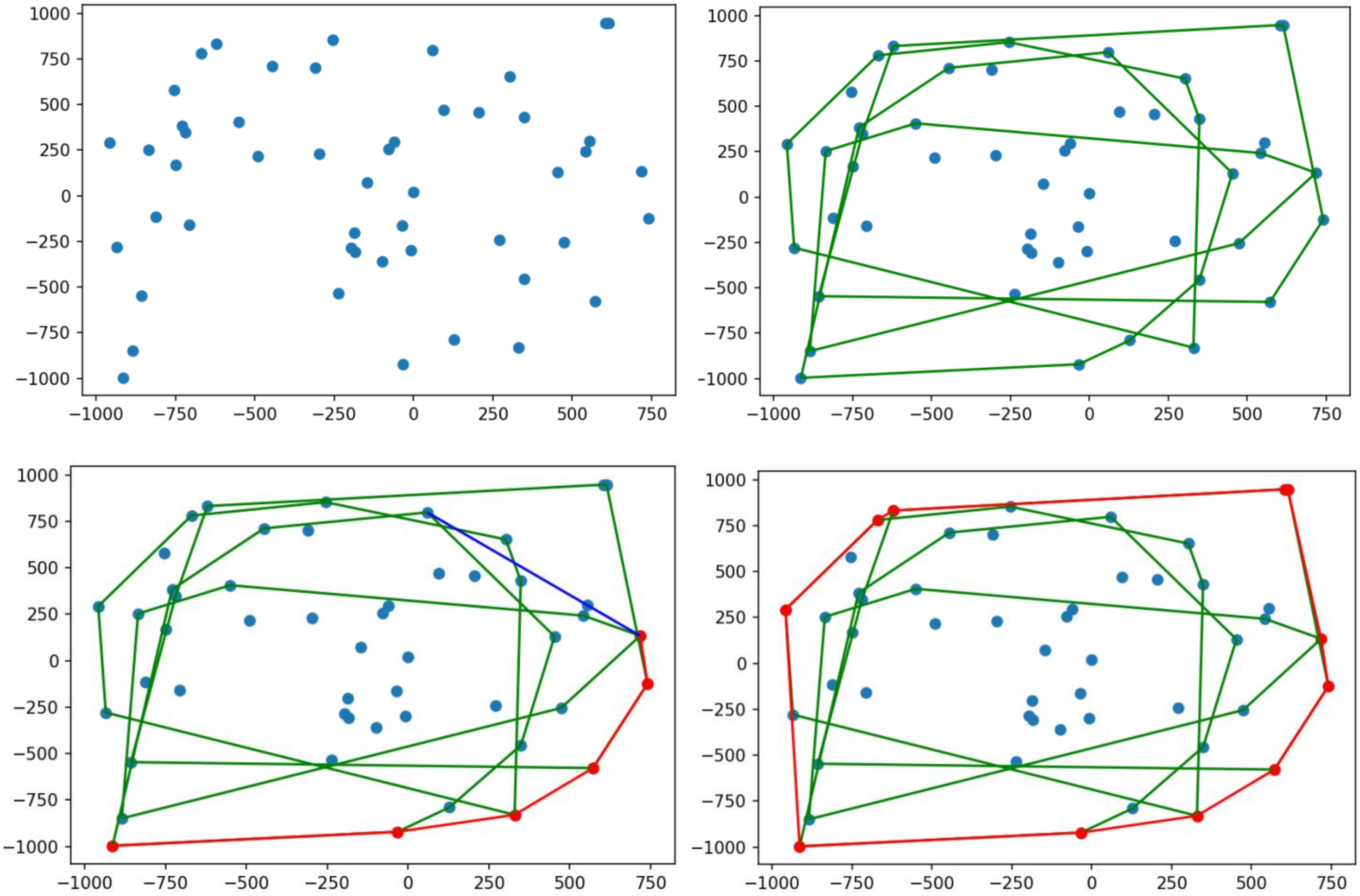

1000
750
500
250
0
−250
−500
−750
−1000
−1000
−750
−500
−250
0
250
500
750
1000

Algorytm Chan'a

Opis algorytmu:

1. Zakładamy że otoczkę tworzy m punktów (jeśli nie znamy m, to wyznaczamy je jako min(2^{2^t}, n) gdzie t = 1,2,3…)
2. Dzielimy zbiór danych na r zbiorów o rozmiarze m
3. Dla każdej grupy punktów wyznaczamy otoczkę algorytmem Grahama
4. Znajdujemy najniższy punkt p0 całego zbioru
5. m razy powtarzamy: dla każdego zbioru szukamy punktu który maksymalizuje kąt z 2 ostatnimi punktami otoczki, a następnie spośród znalezionych punktów również wybieramy ten który maksymalizuje ten kąt
6. Jeżeli w ciągu m wykonań pętli nie dotarliśmy do punktu p0,zwiększamy m, w przeciwnym przypadku zwracamy wynik


0
200
400
600
800
1000


1000
750
500
250
0
−250
−500
−750
−1000
−1000 −750 −500 −250 0 250 500 750
1000
750
500
250
0
−250
−500
−750
−1000
−1000 −750 −500 −250 0 250 500 750
1000
750
500
250
0
−250
−500
−750
−1000
−1000 −750 −500 −250 0 250 500 750
1000
750
500
250
0
−250
−500
−750
−1000
−1000 −750 −500 −250 0 250 500 750

Dane bibliograficzne

- https://www.coursera.org/lecture/computational-geometry/2-7-incremental-algorithms-ZBrkm
- https://www.geeksforgeeks.org/quickhull-algorithm-convex-hull/
- https://iq.opengenus.org/chans-algorithm-convex-hull/
- Prezentacja z wykładu