Algorytmy Geometryczne

Otoczka wypukła dla zbioru punktów w przestrzeni dwuwymiarowej

Smyda Tomasz
Wiśniewski Jakub

1 Problem

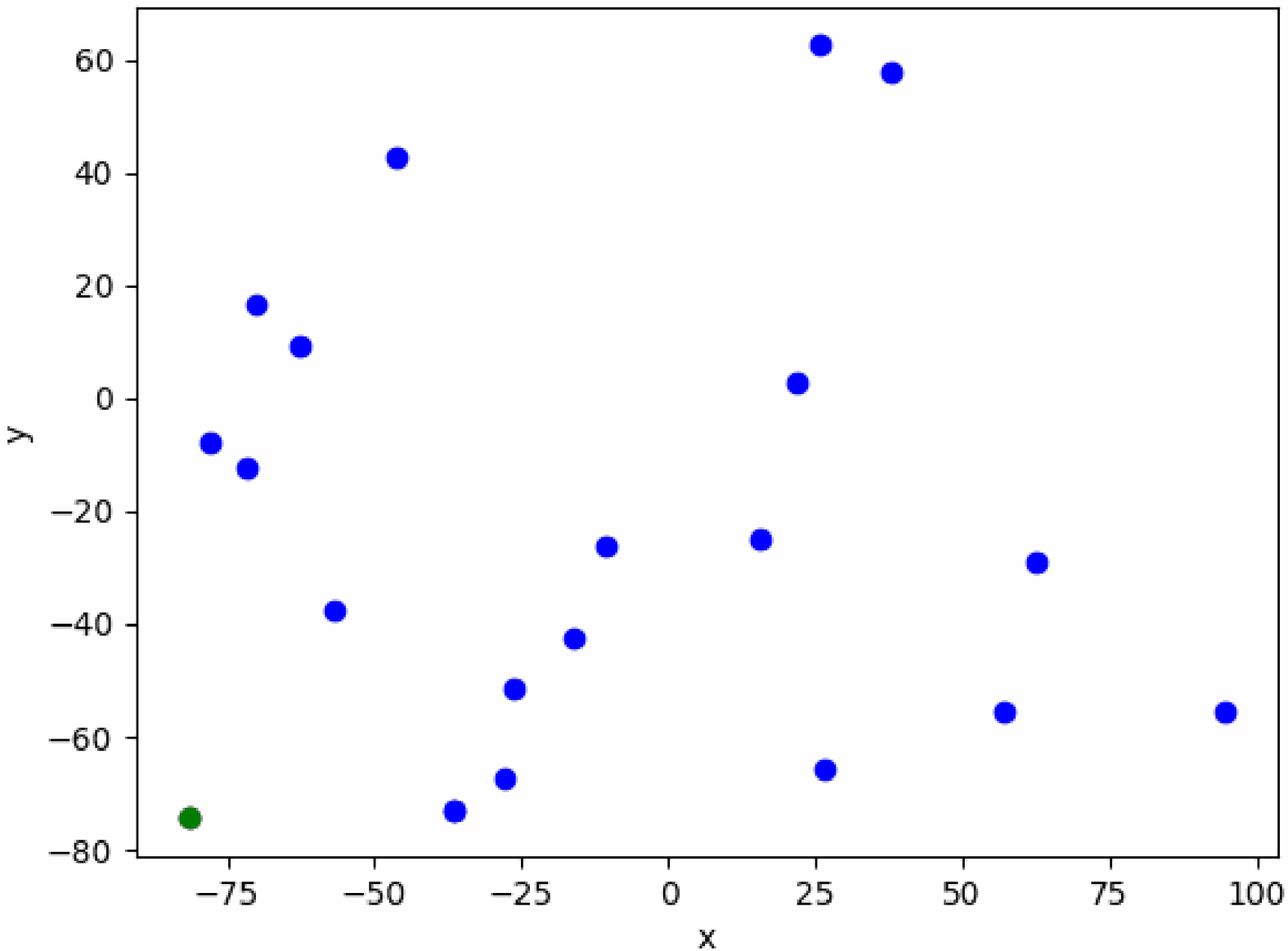
Otoczką wypukłą nazywamy najmniejszy zbiór wypukły zawierający w sensie inkluzji dany zbiór punktów. Inaczej - jest to najmniejszy wielokąt wypukły rozpięty na wierzchołkach danego zbioru punktów taki, że wszystkie punkty znajdują się albo wewnątrz niego, albo na jego krawędzi.

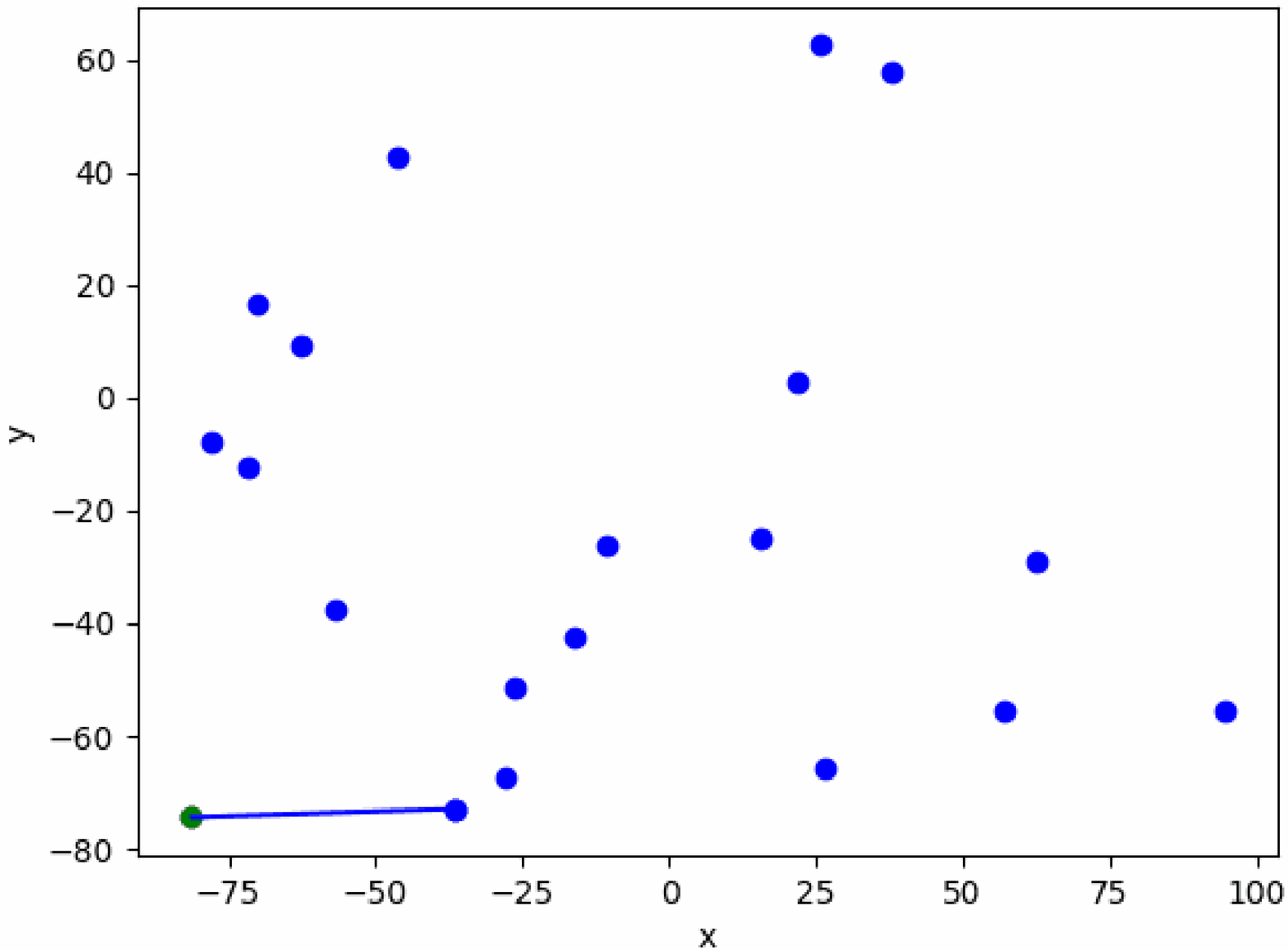
Wyznaczenie otoczki wypukłej dla danego zbioru punktów znajduje zastosowania w informatyce. Między innymi w:

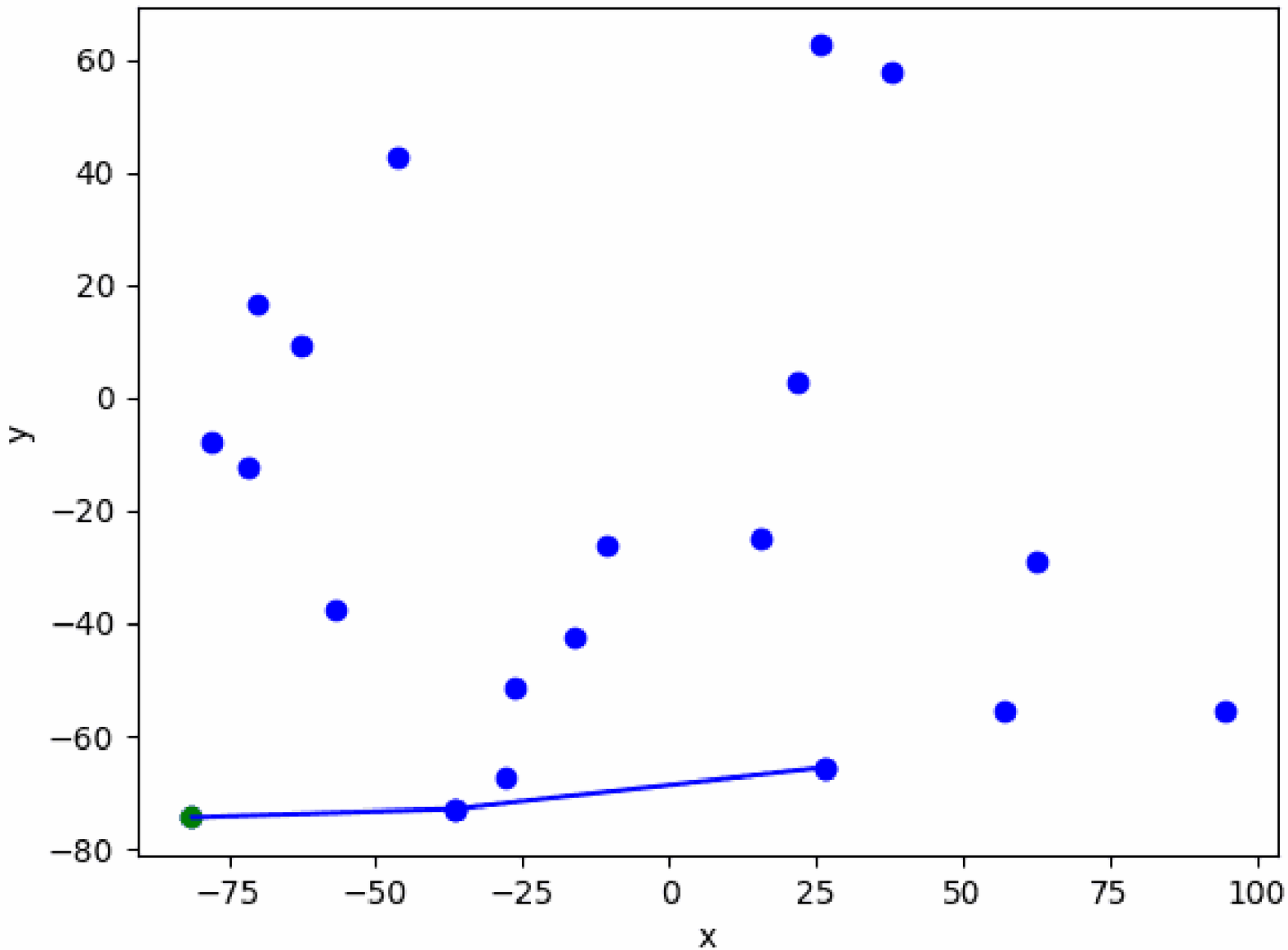
- Wyznaczenie najmeniejszego pojemnika
- Algorytm detekcji kolizji
- Analiza kształtu obiektu

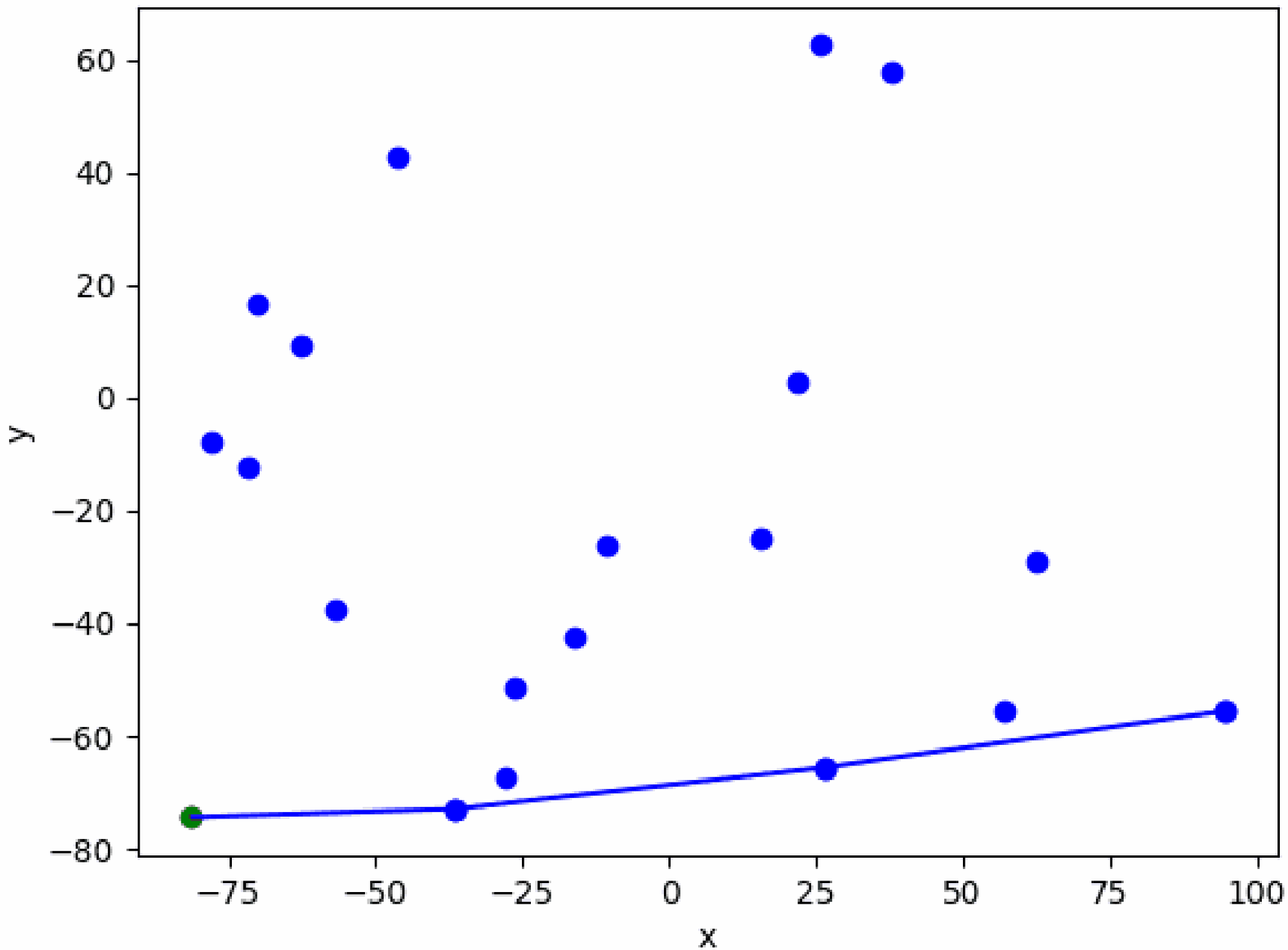
2 Algorytm Grahama

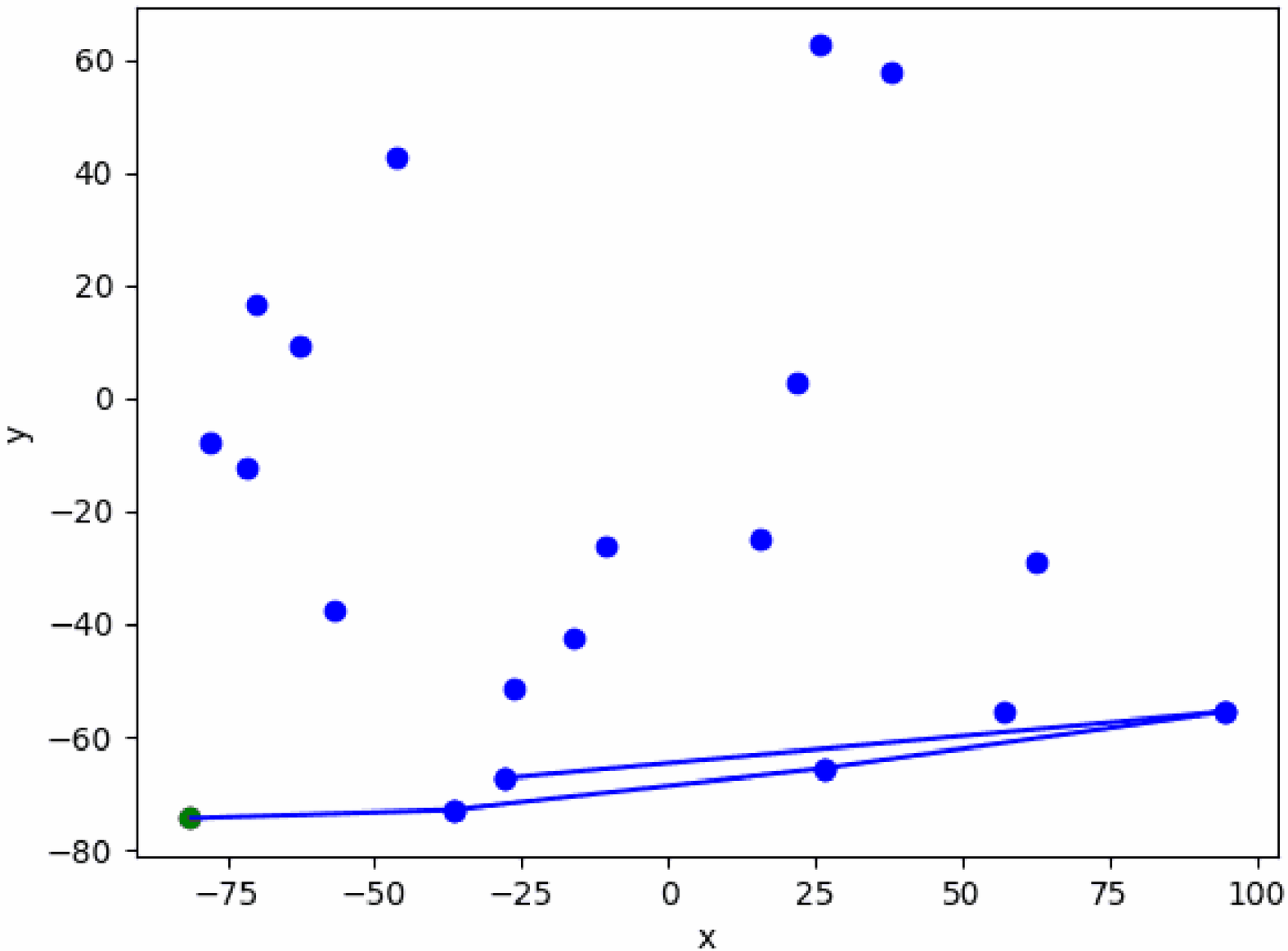
- W zbiorze S wybieramy punkt p_0 o najmniejszej współrzędnej y. Jeżeli jest kilka takich punktów, to wybieramy ten z nich, który ma najmniejszą współrzędną x.
- Sortujemy pozostałe punkty ze względu na kąt, jaki tworzy wektor $[p_0, p]$ z dodatnim kierunkiem osi OX. Jeśli kilka punktów tworzy ten sam kąt, usuwamy wszystkie z wyjątkiem najbardziej oddalonego od p_0. Niech uzyskanym ciągiem będzie $p_1, p_2, \ldots, p_m$
- Do poczatkowo pustego stosu s wkładamy punkty p_0, p_1, p_2
- Iterujemy po reszcie posortowanych punktów. Jeżeli kolejny punkt znajduje się po prawej stronie odcinka utworzonego z dwóch ostatnich punktów ze stosu lub jest z nim współliniowy, to usuwamy ostatni punkt ze stosu. W przeciwnym przypadku wstawiamy punkt na szczyt stosu i przechodzimy do następnego punktu.
- Punkty, które zostały na stosie po ukończeniu algorytmu tworzą otoczkę wypukłą dla danego zbioru punktów
- Algorytm ma złożoność $O(n \log n)$

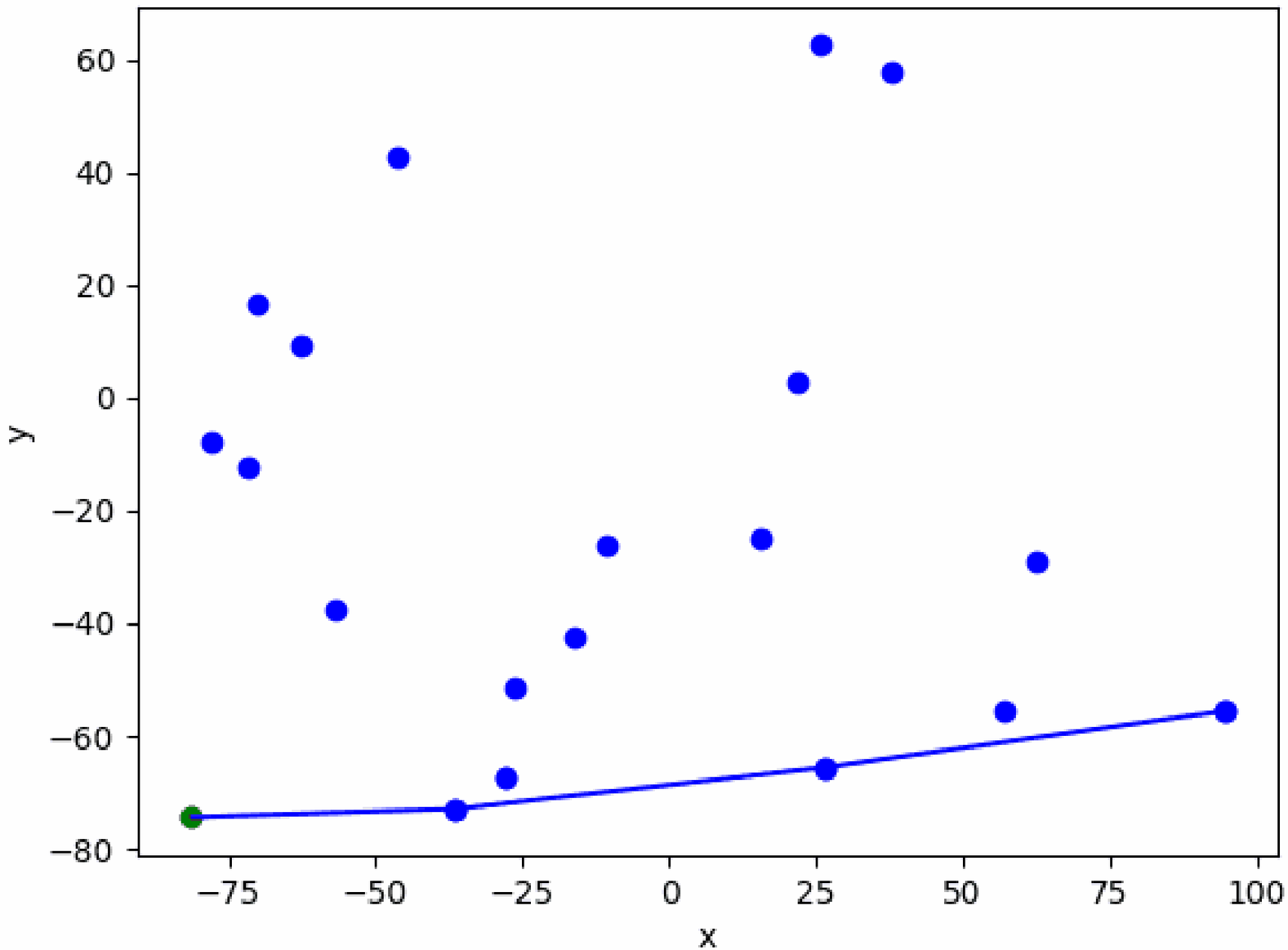

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

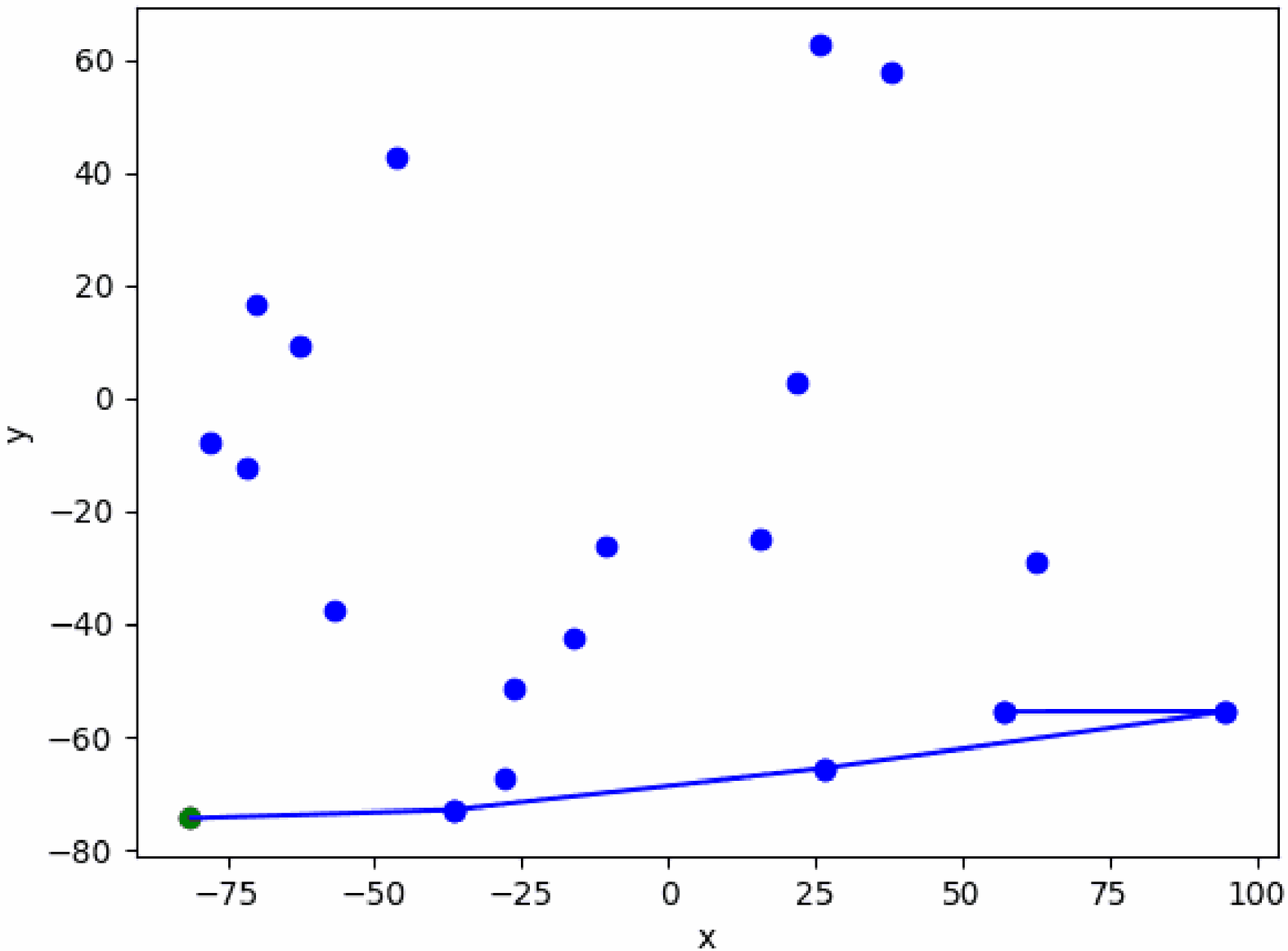

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

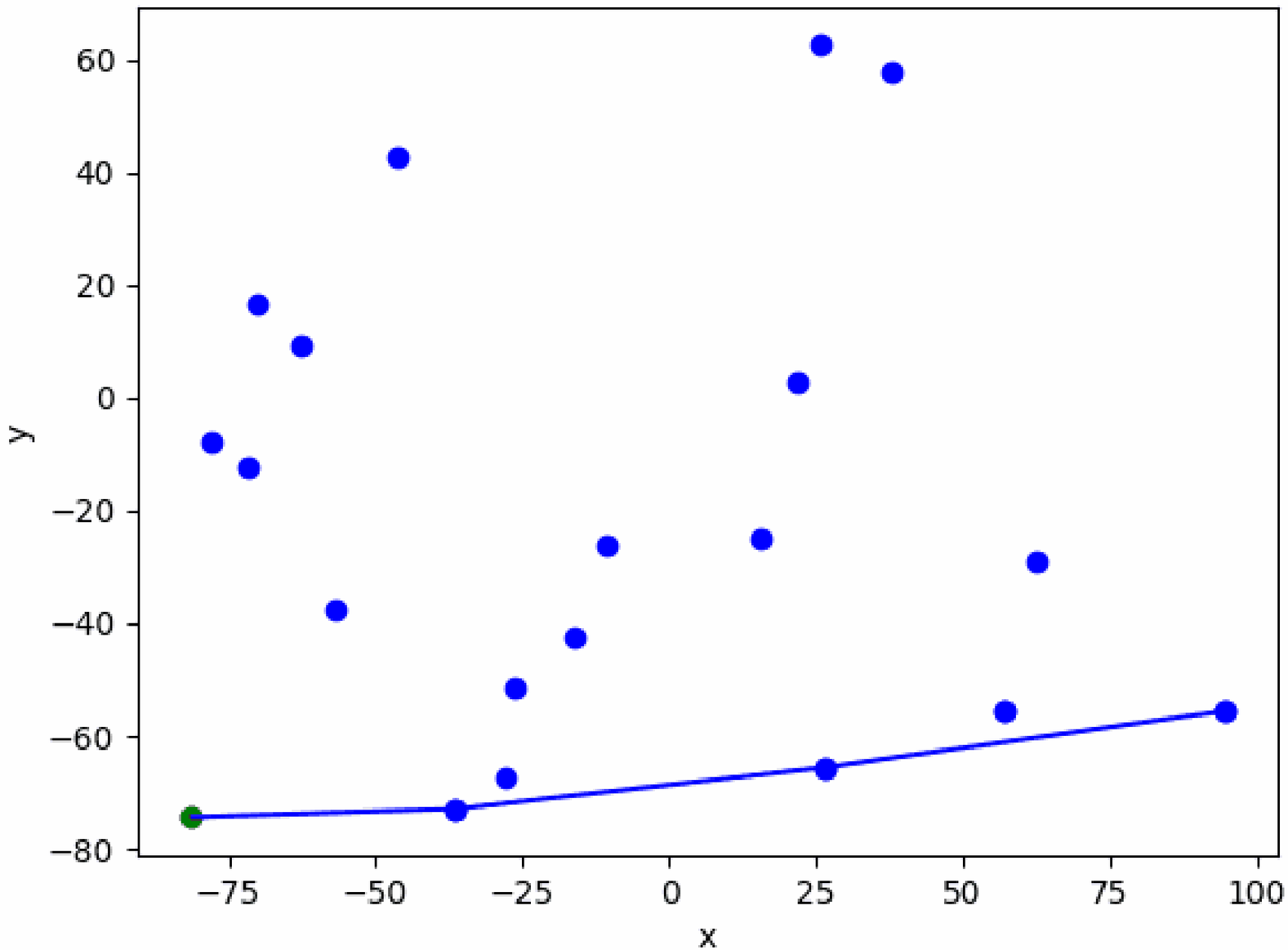

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

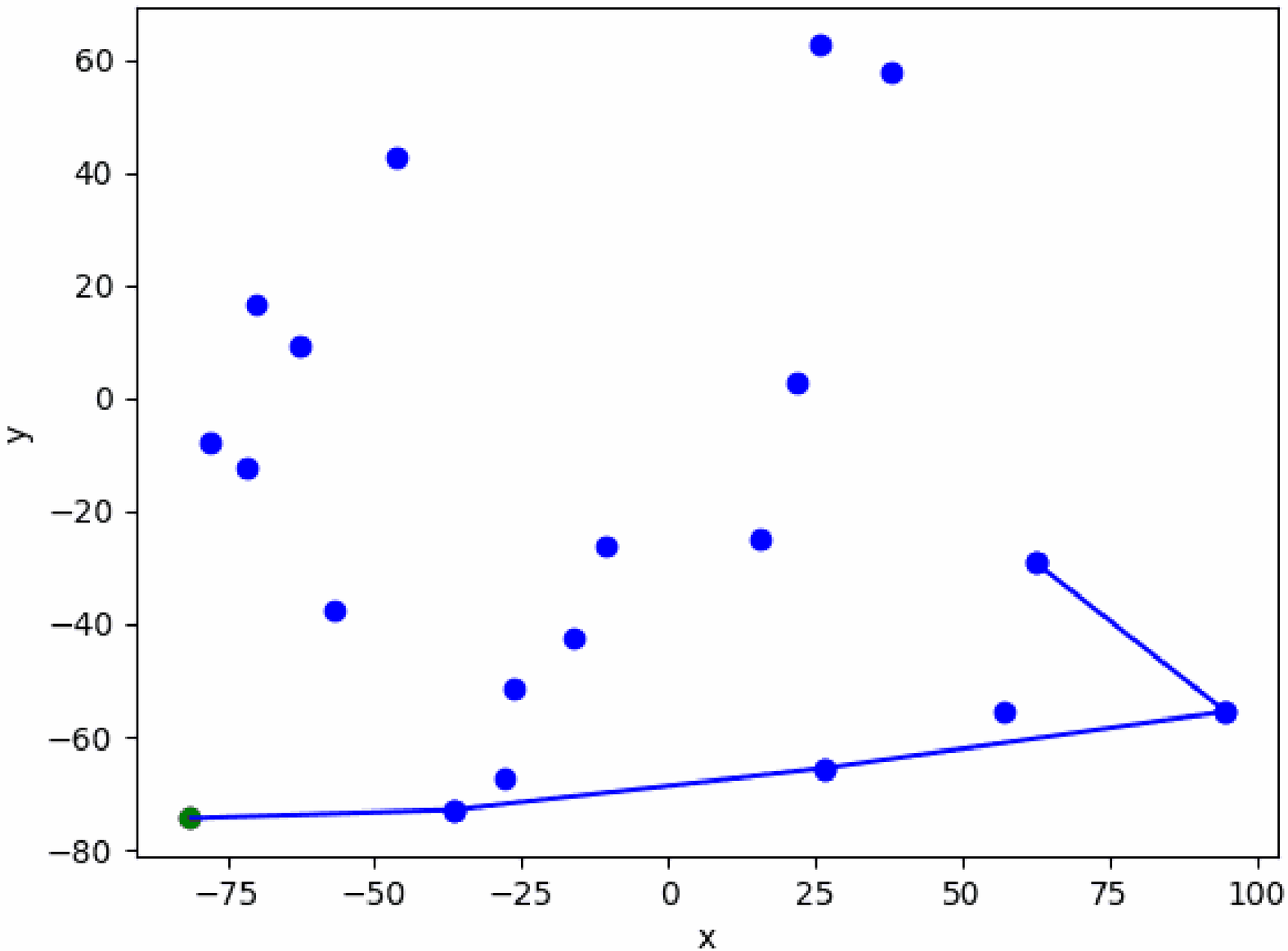

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

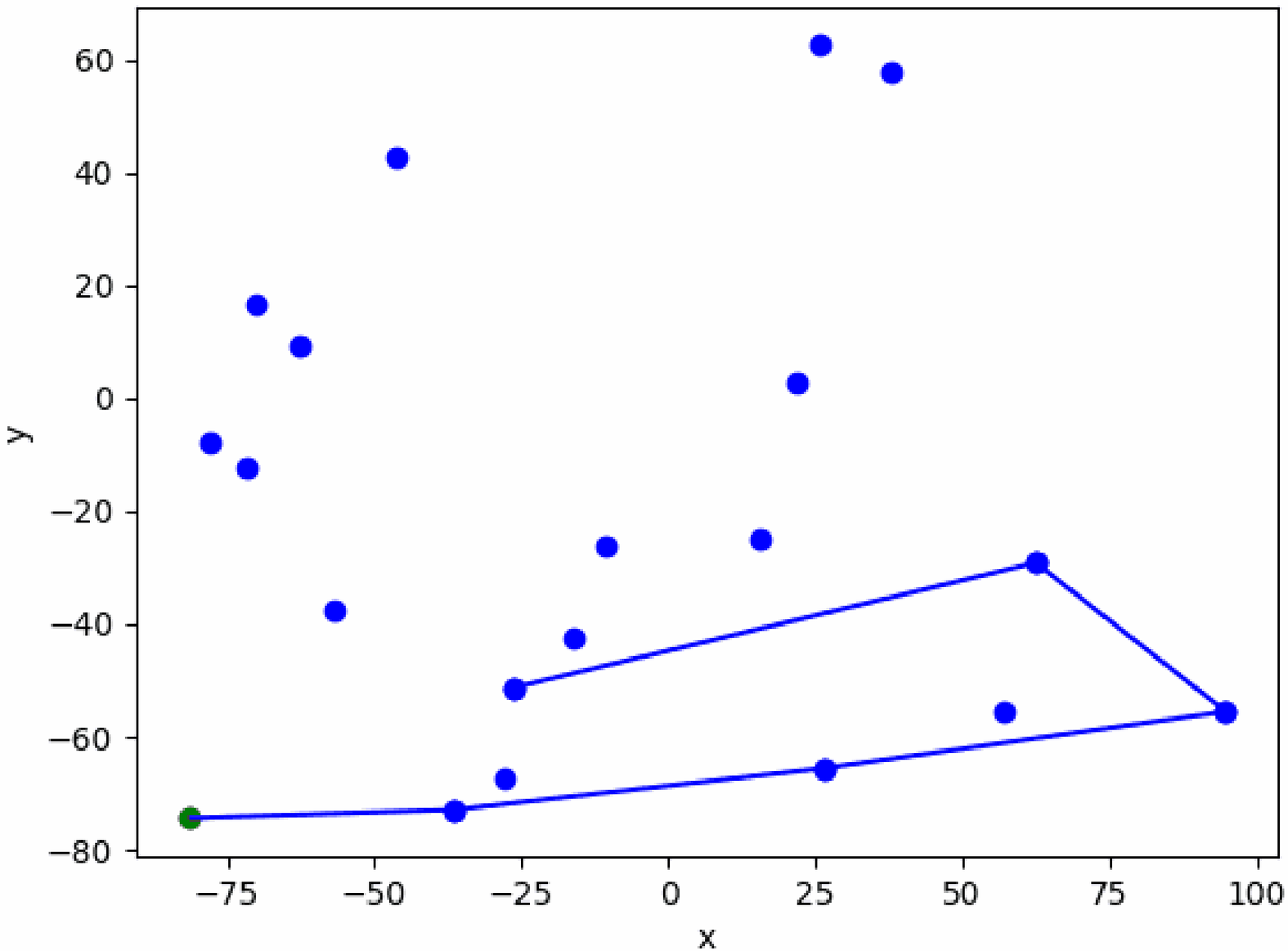

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

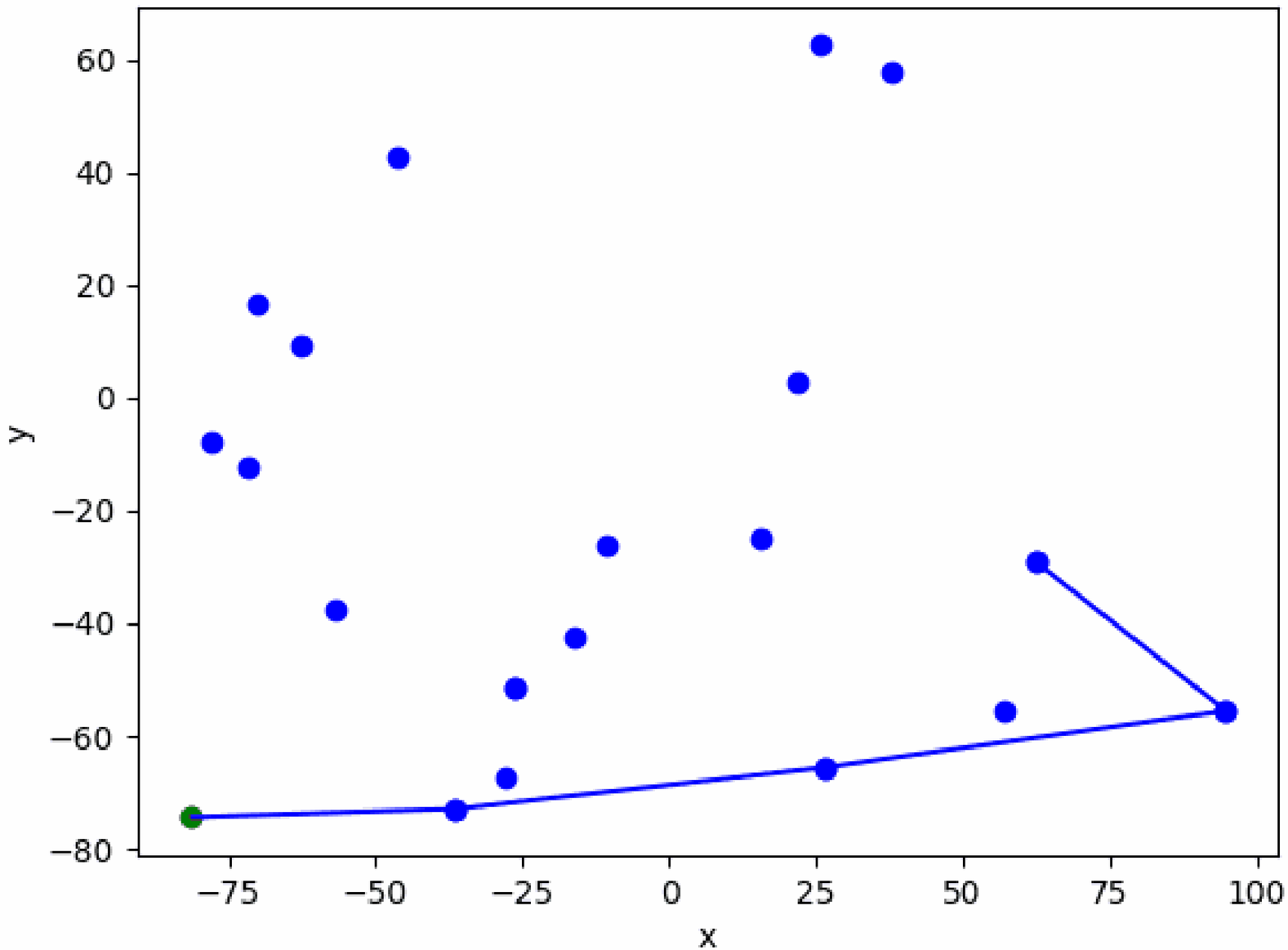

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

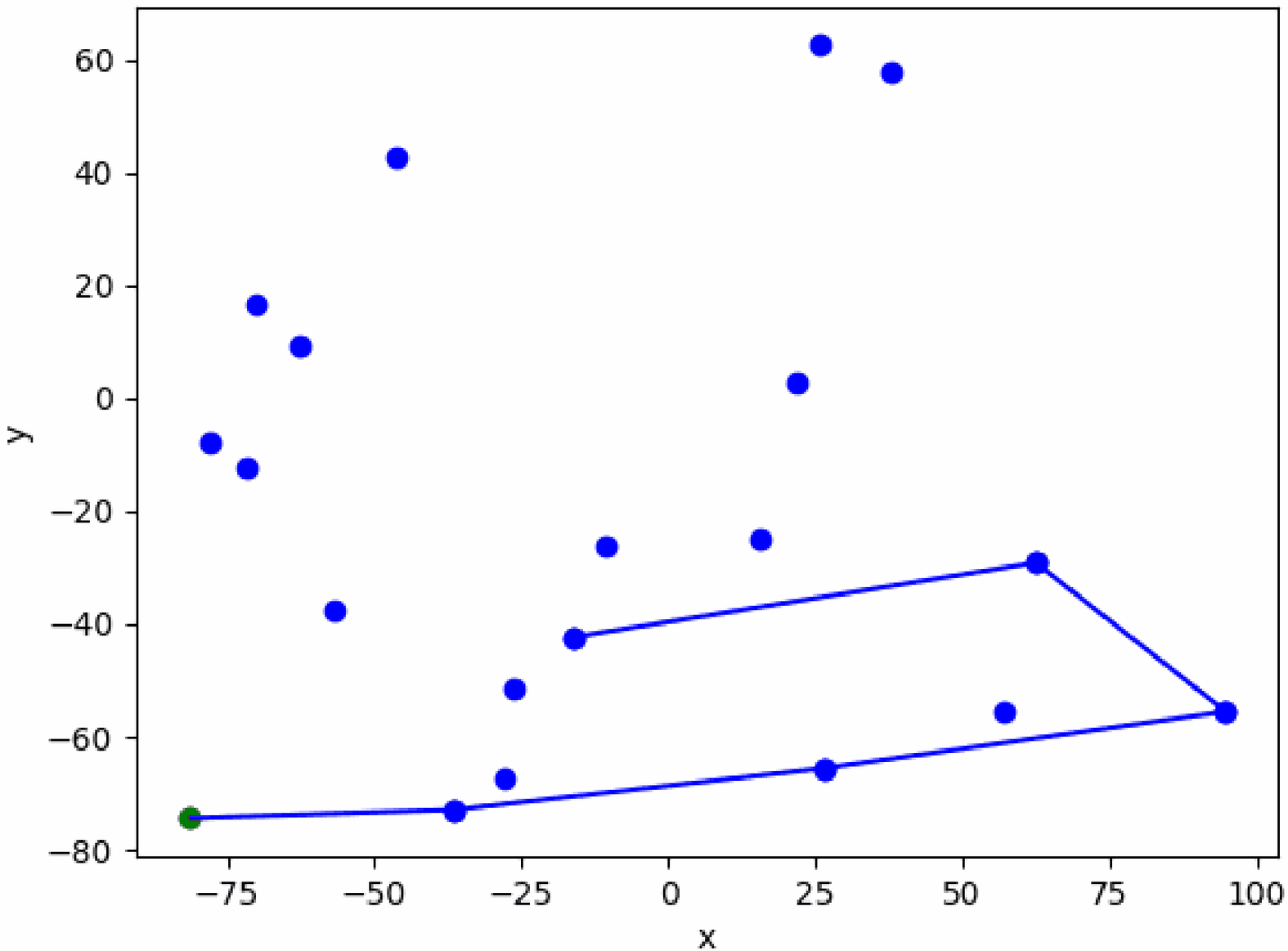

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

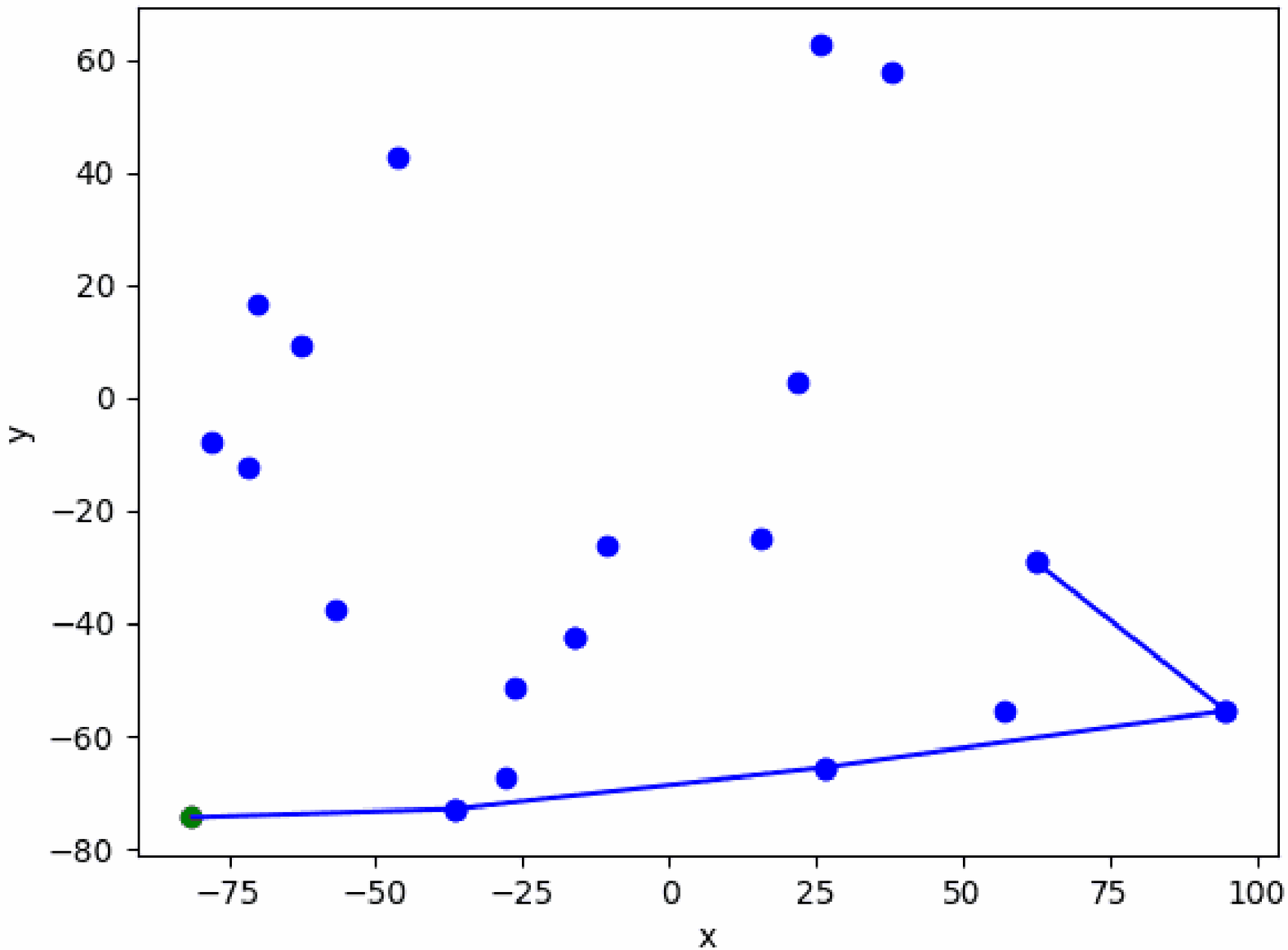

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

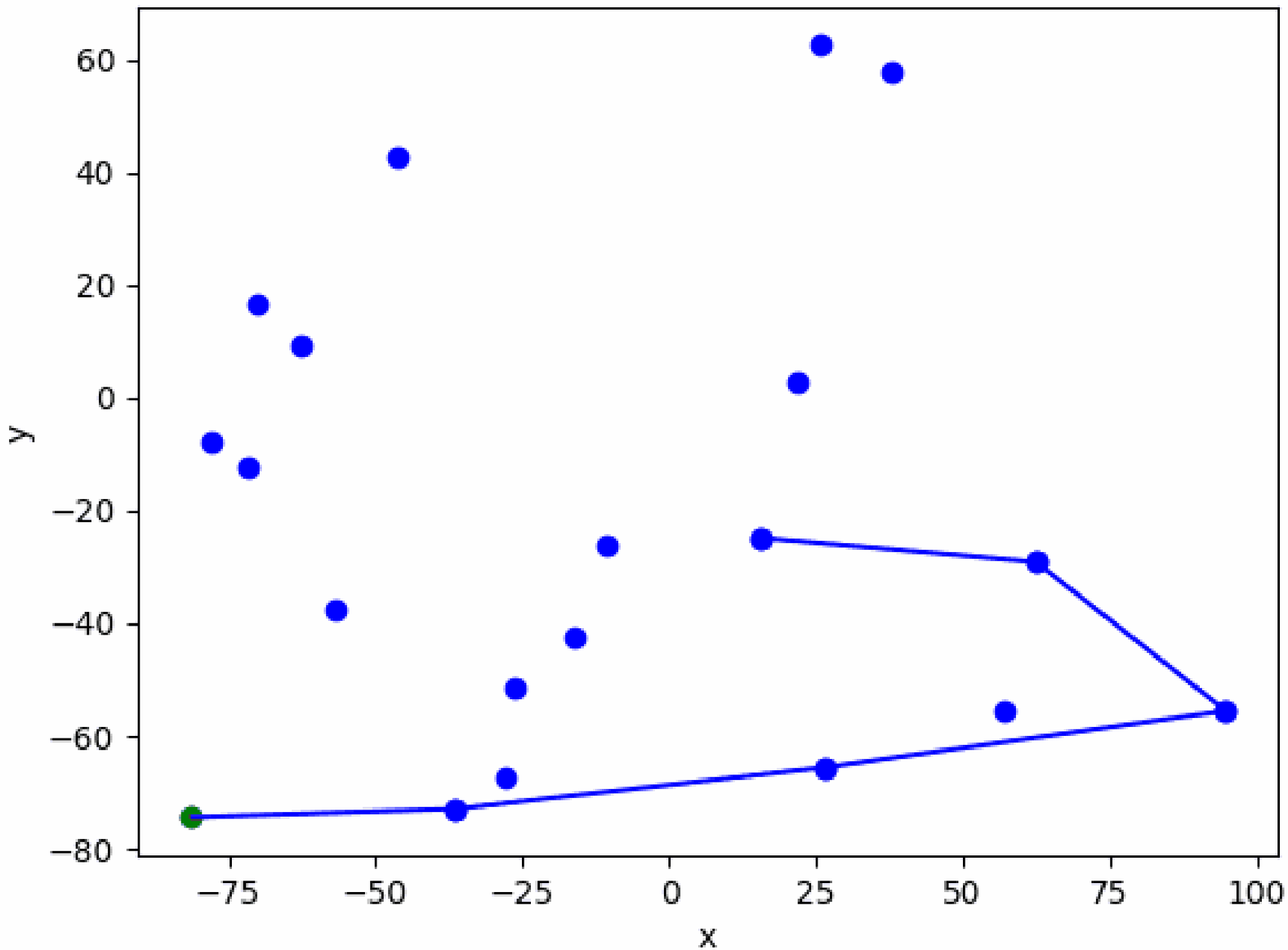

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

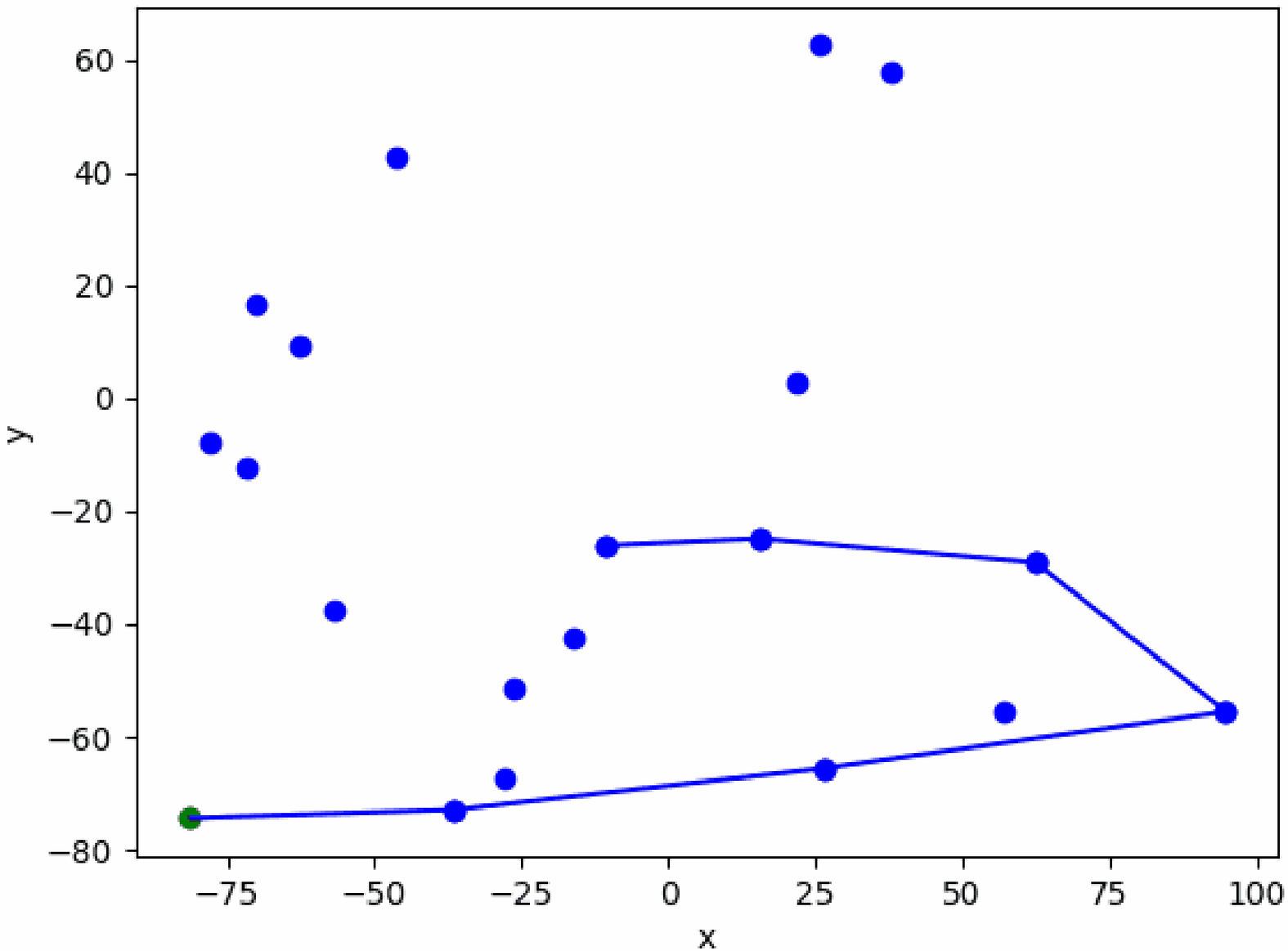

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

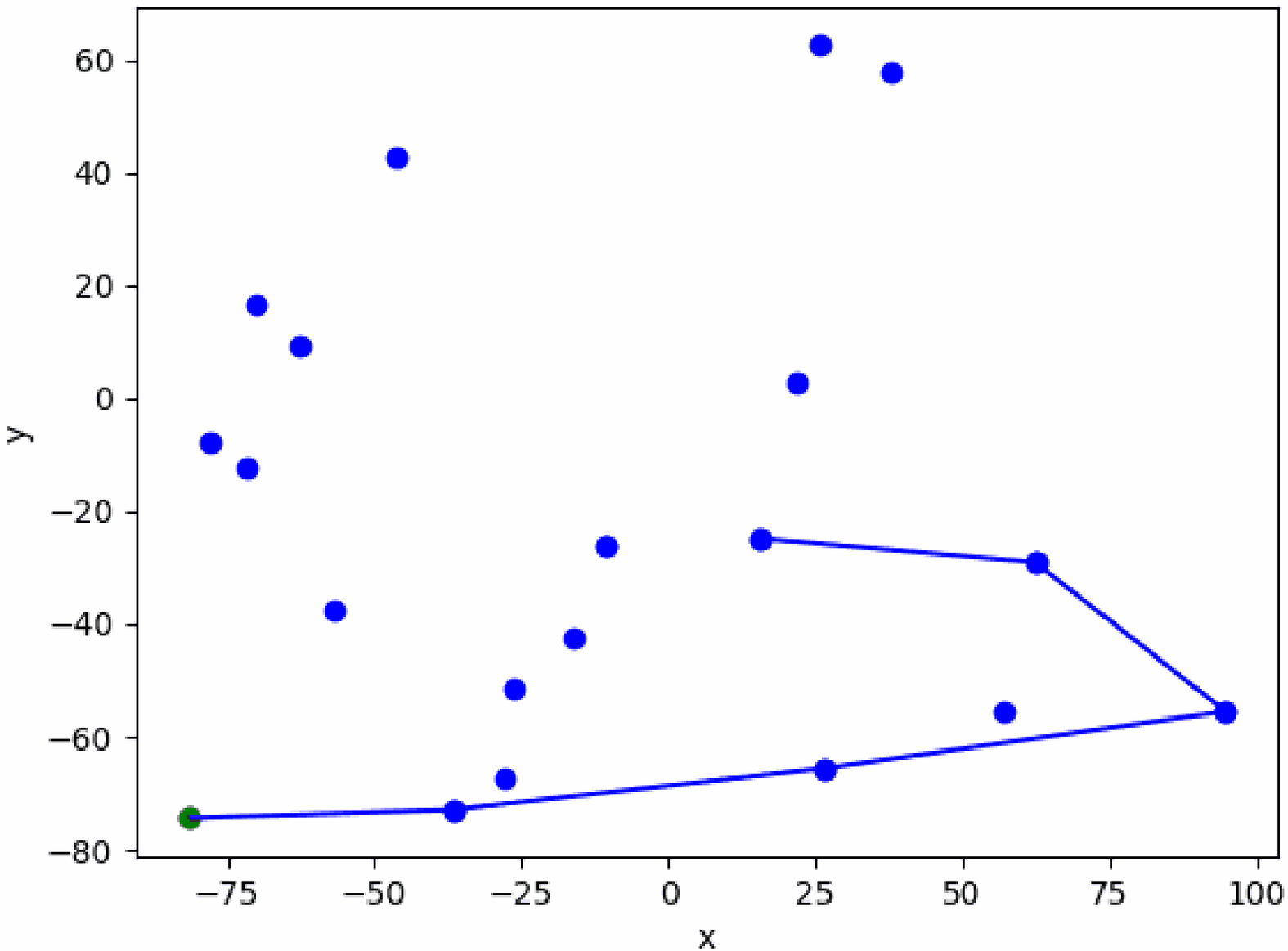

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

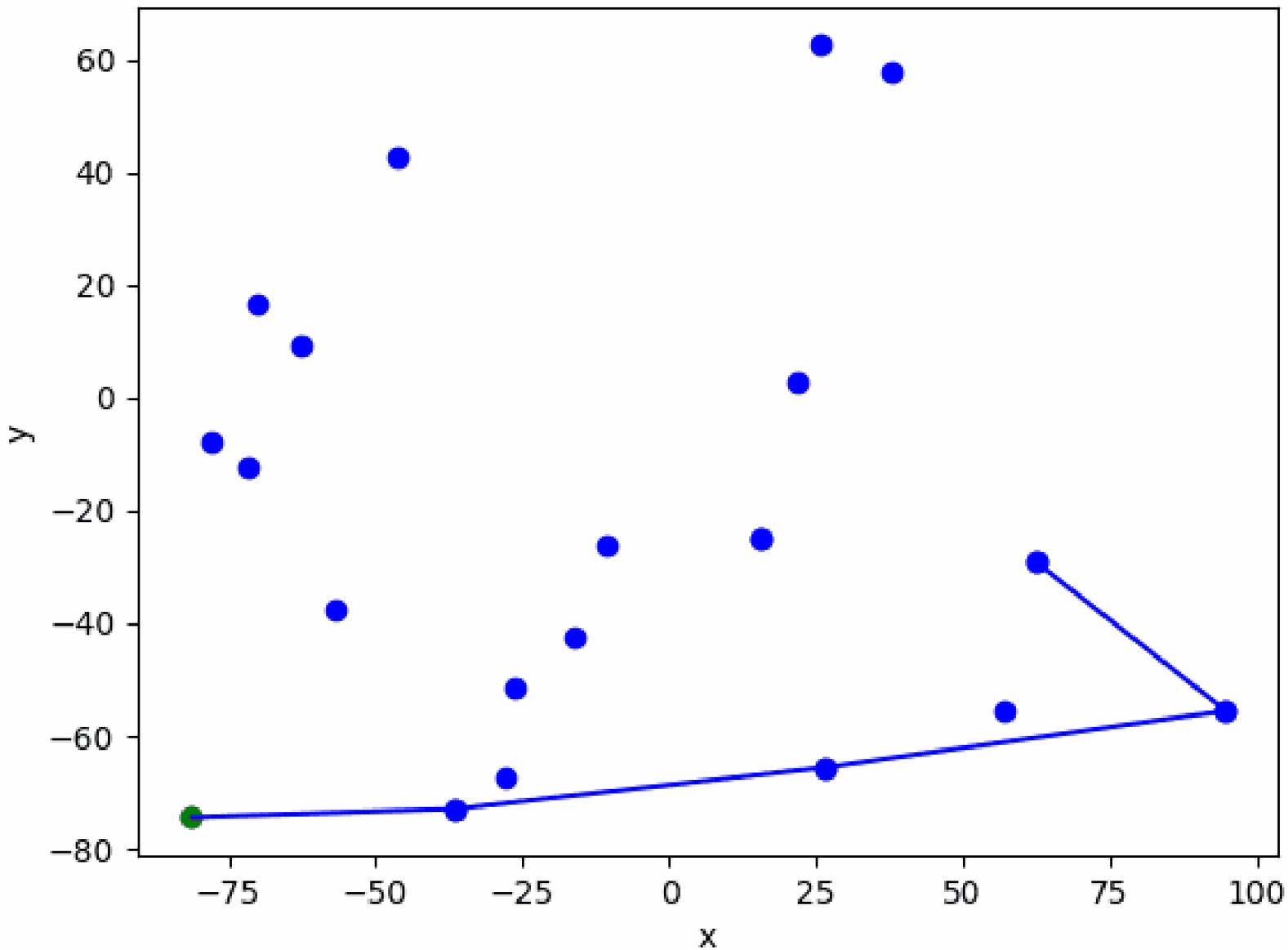

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

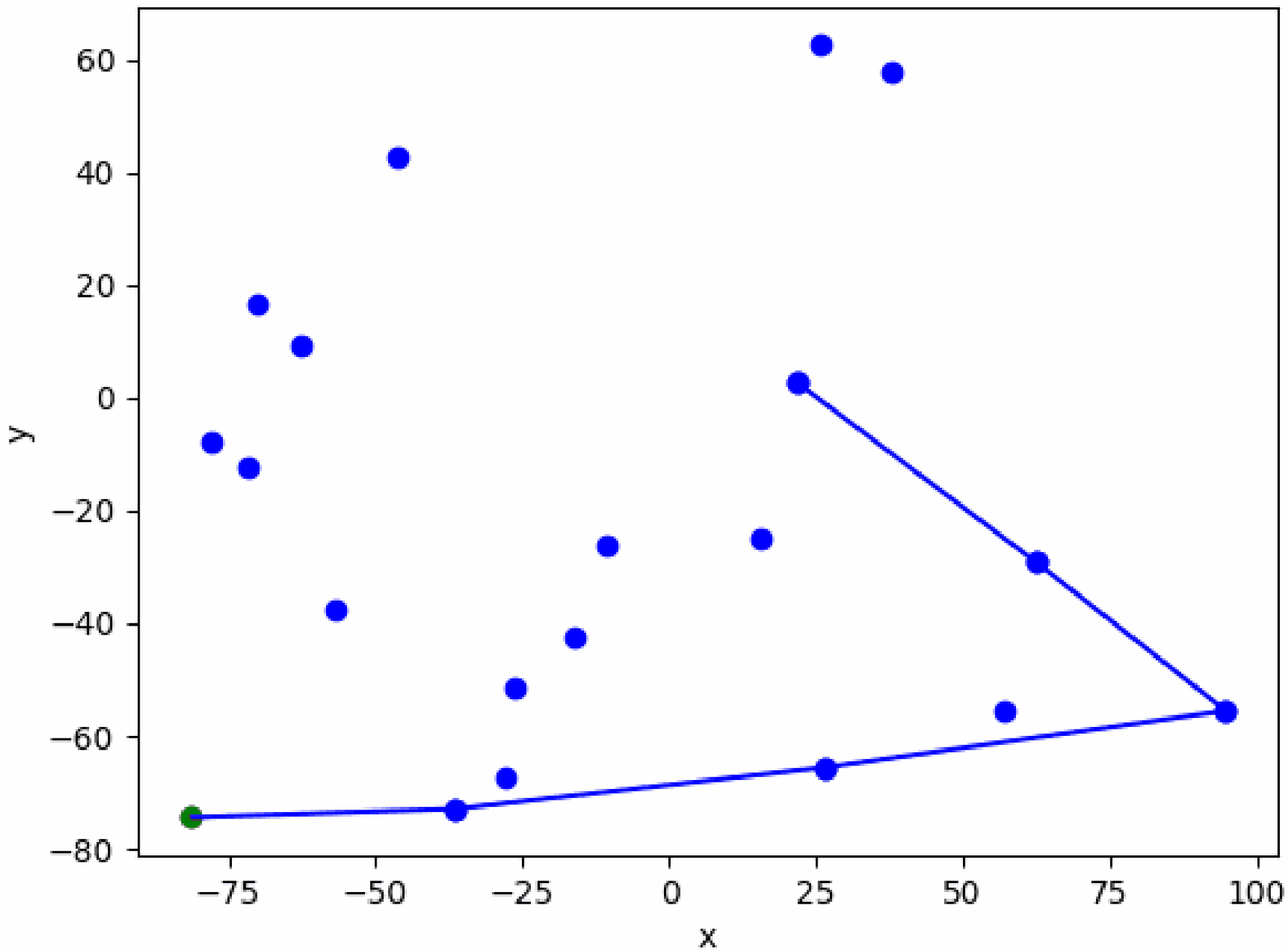

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

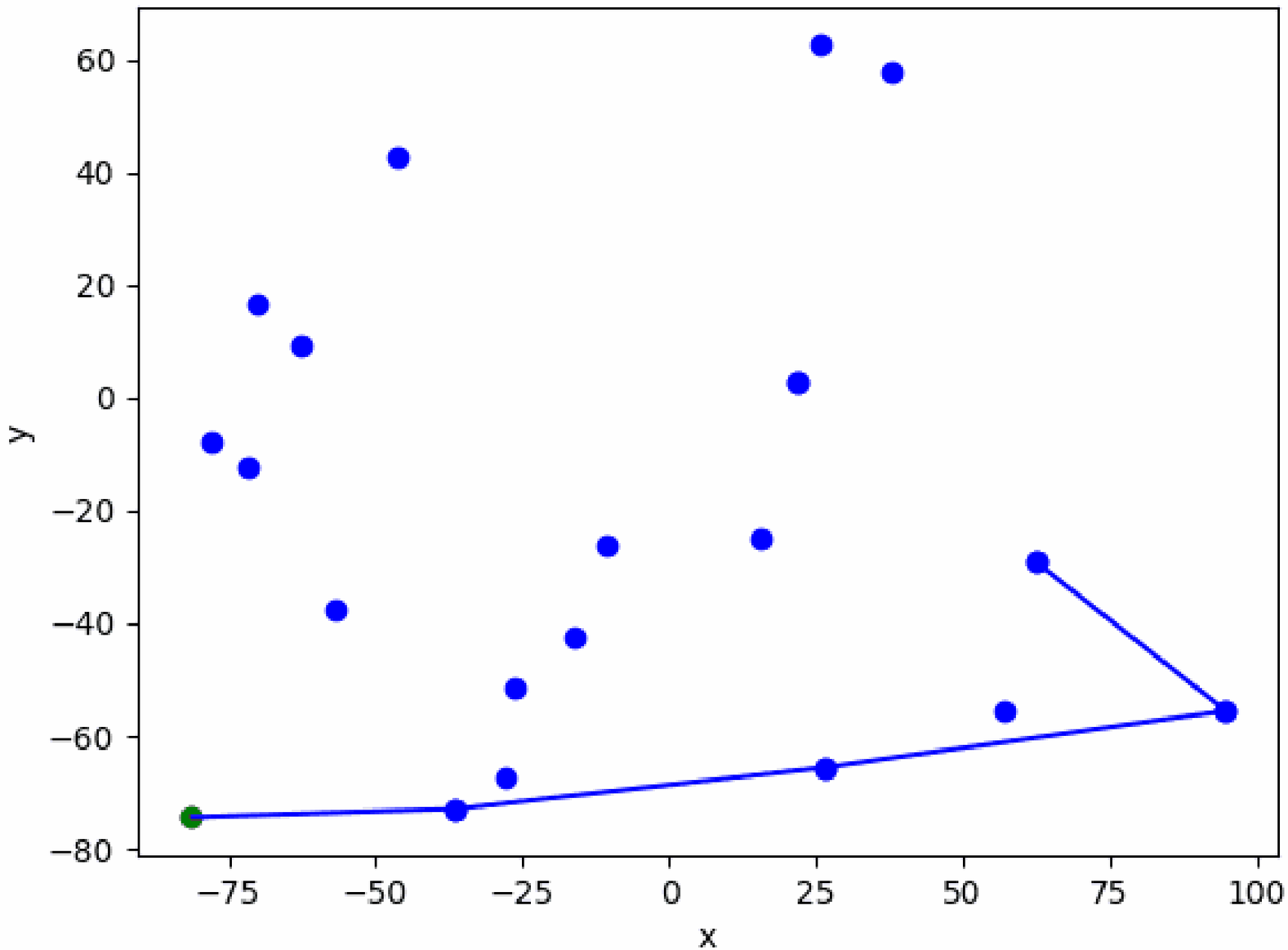

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

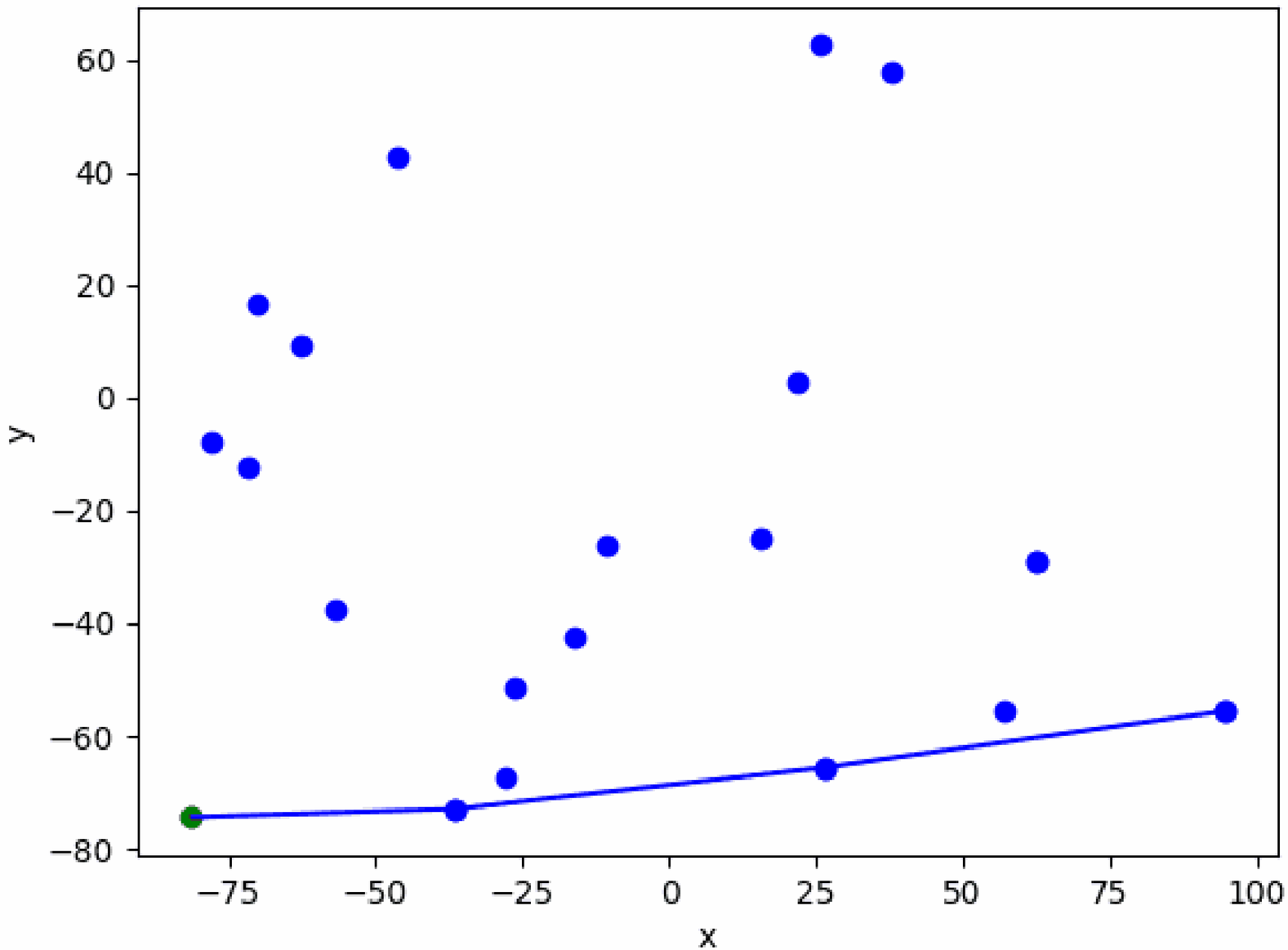

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

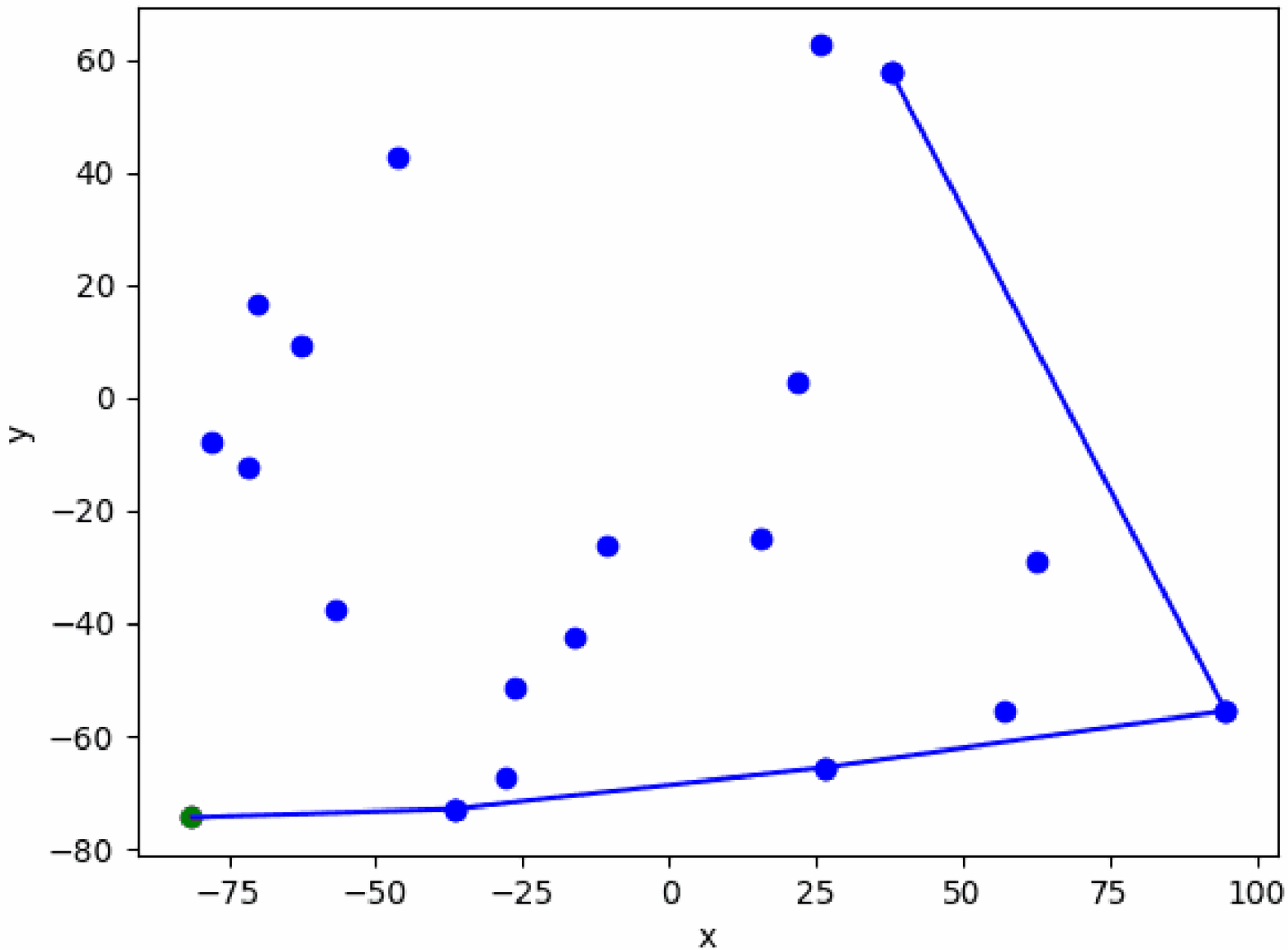

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

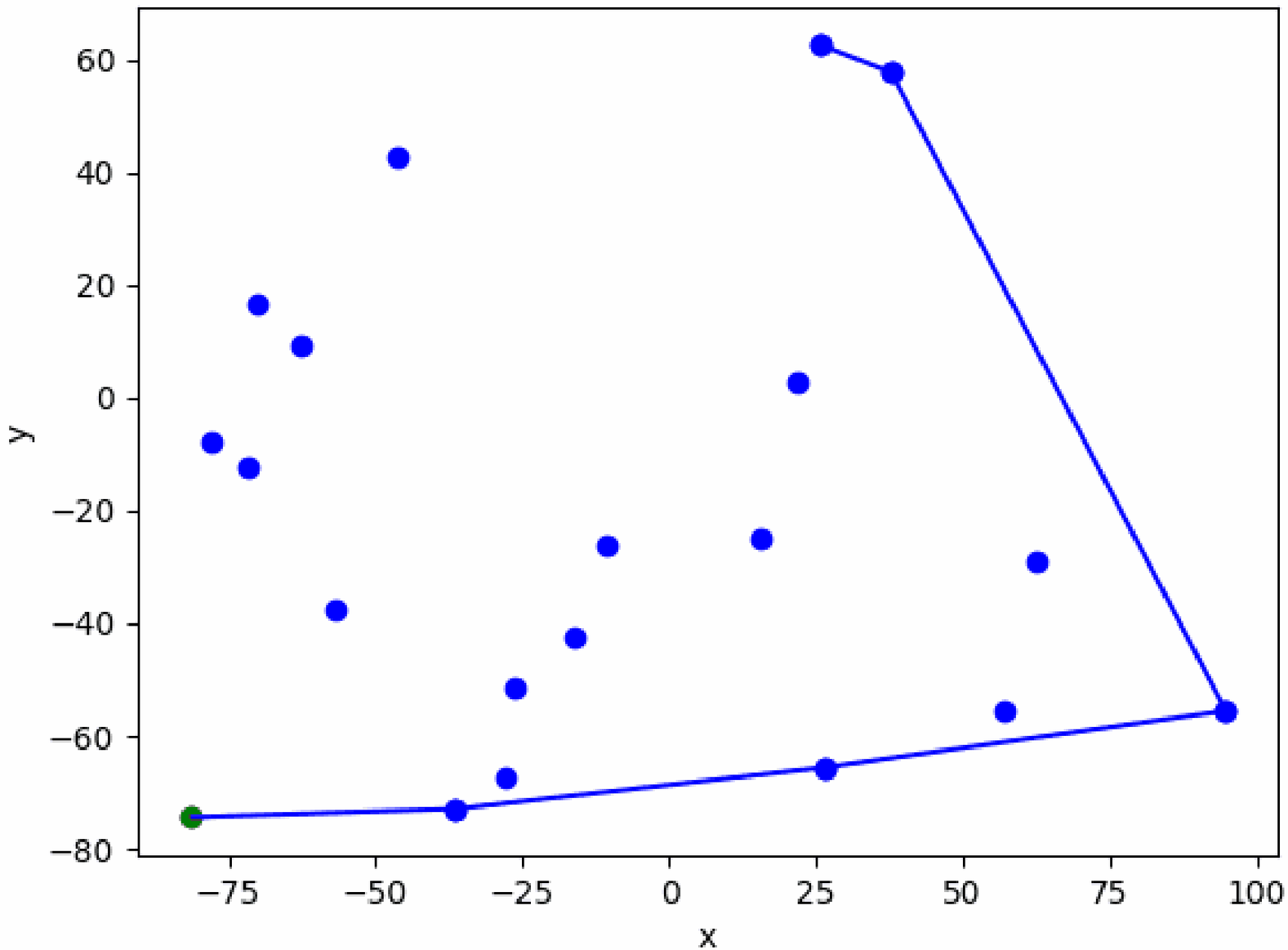

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

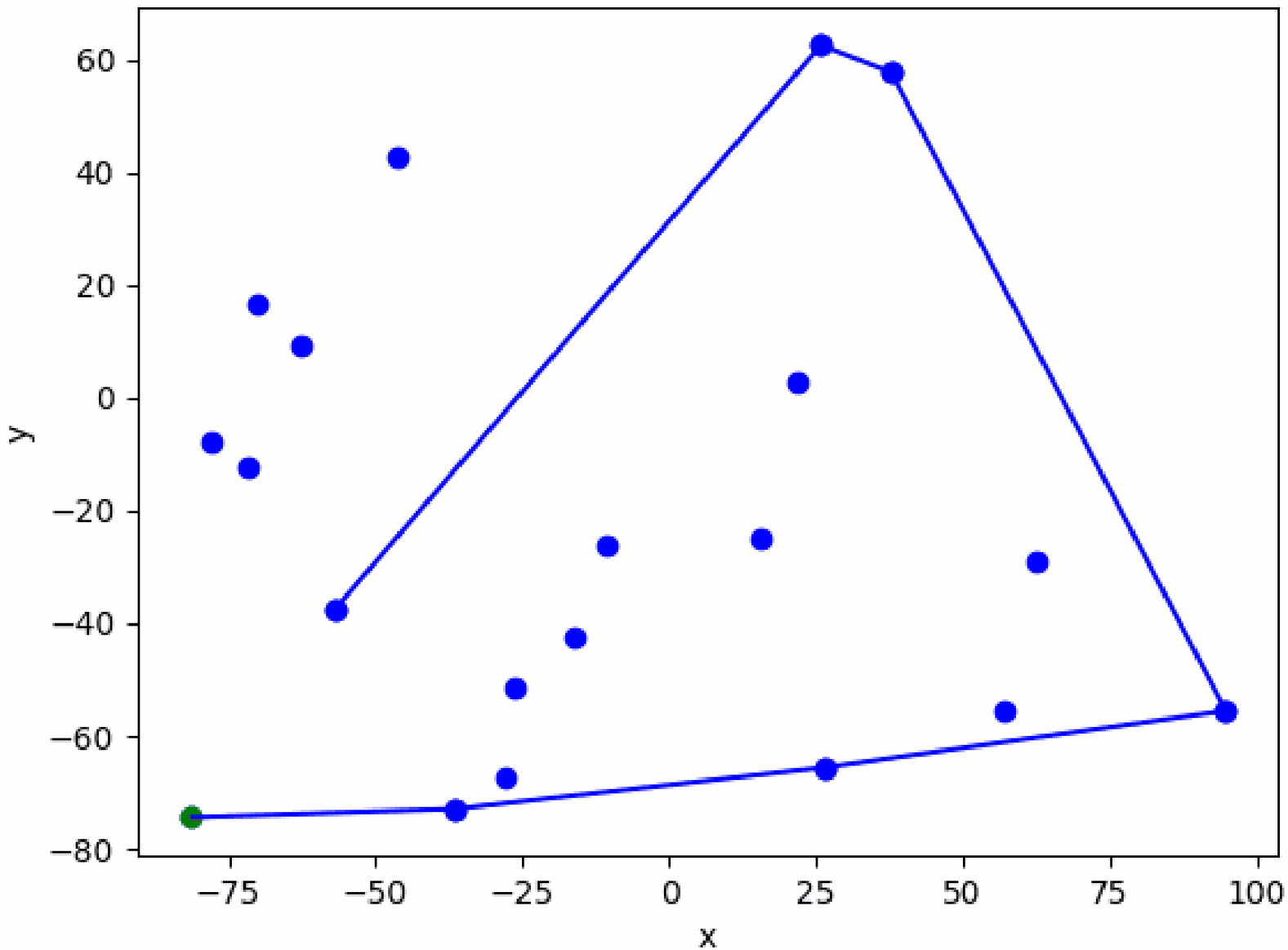

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

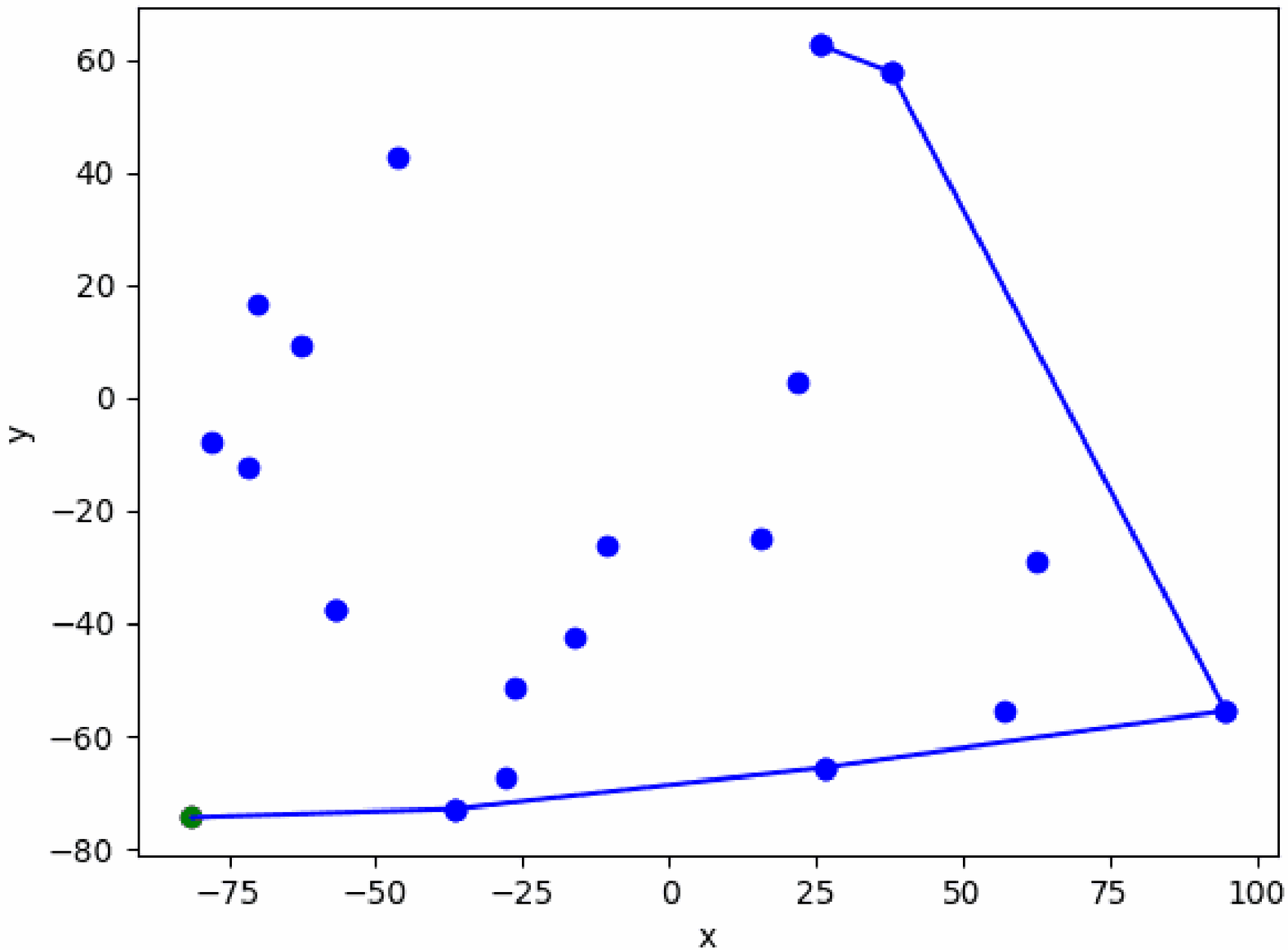

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

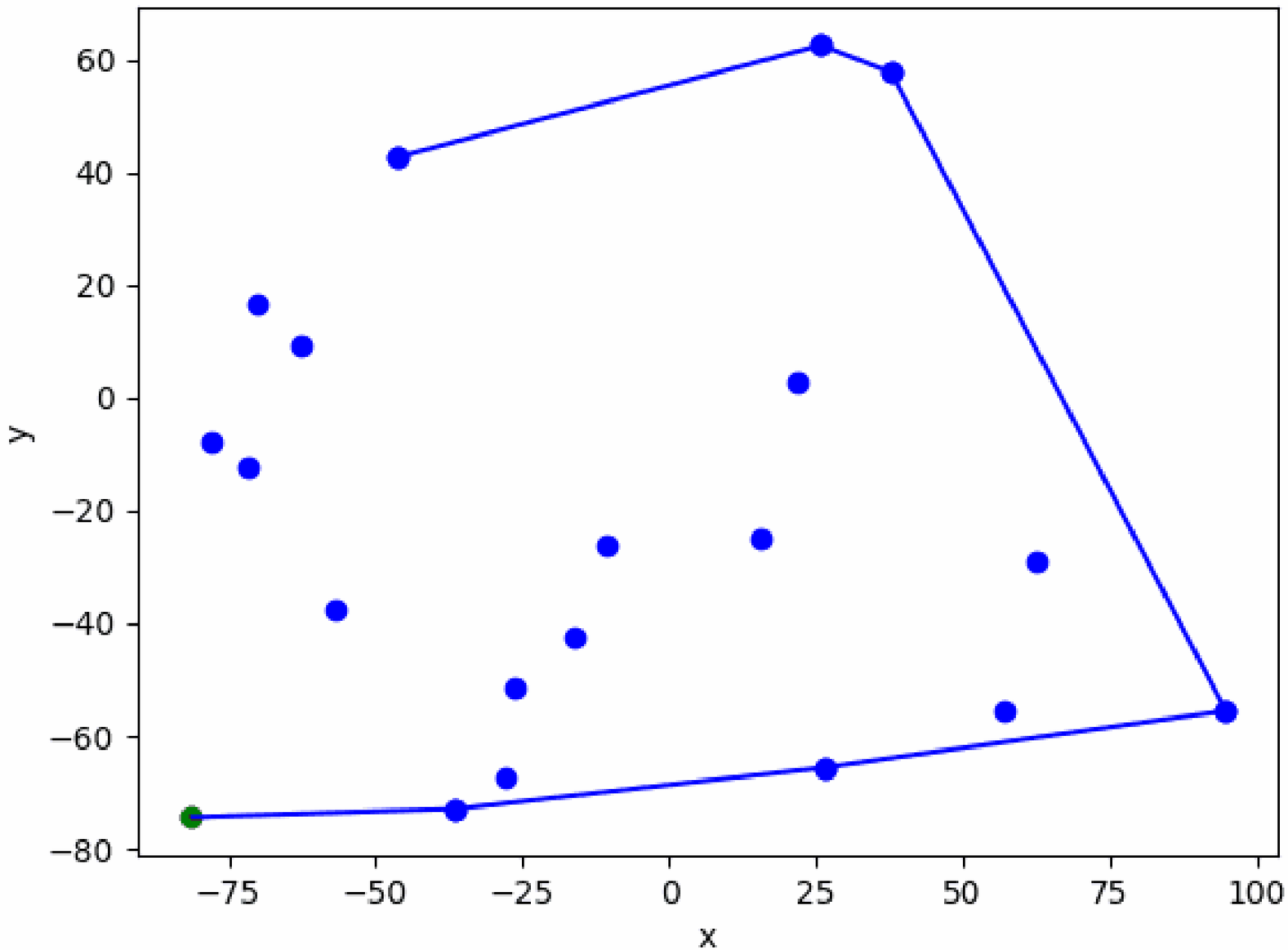

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

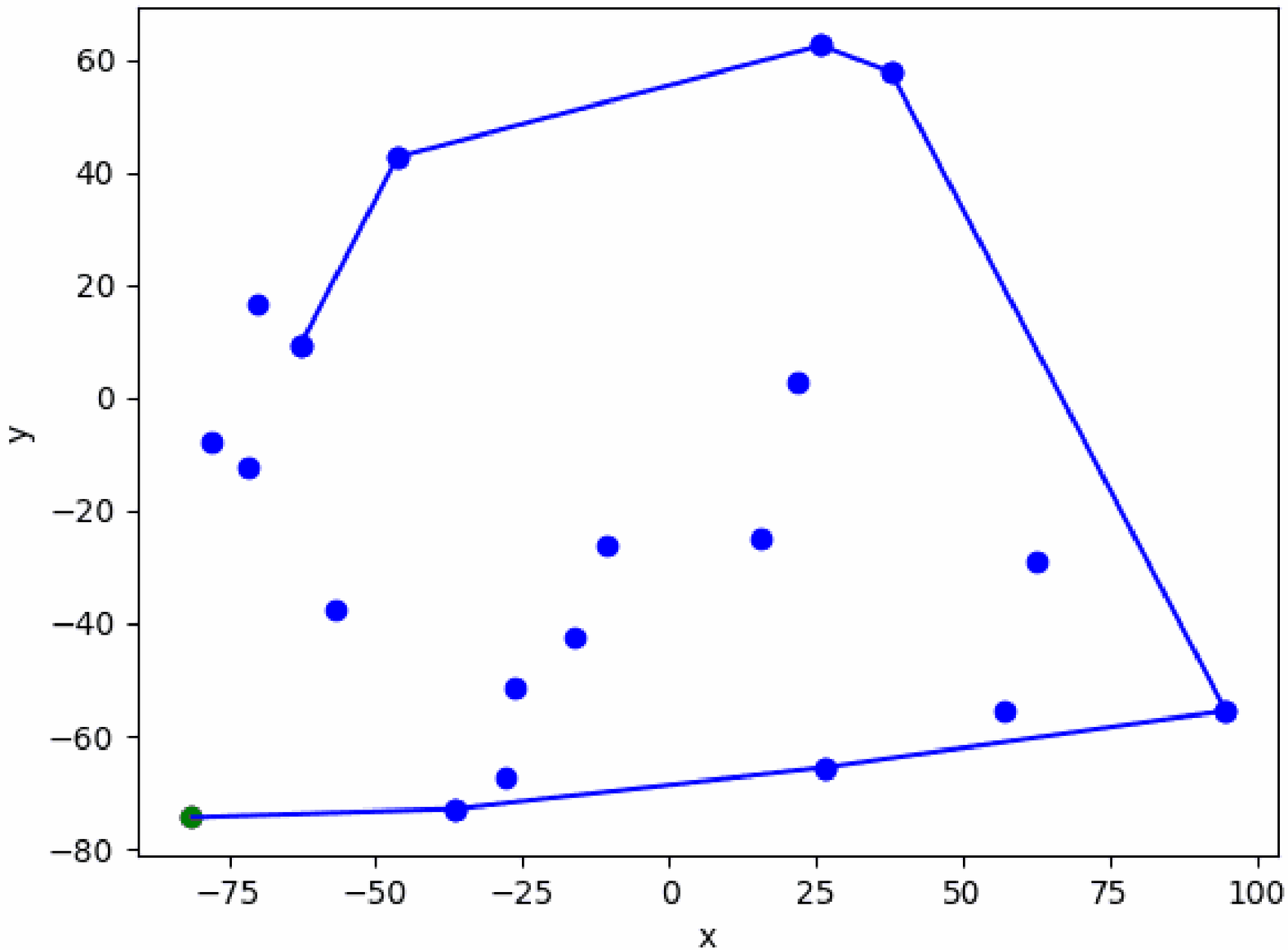

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

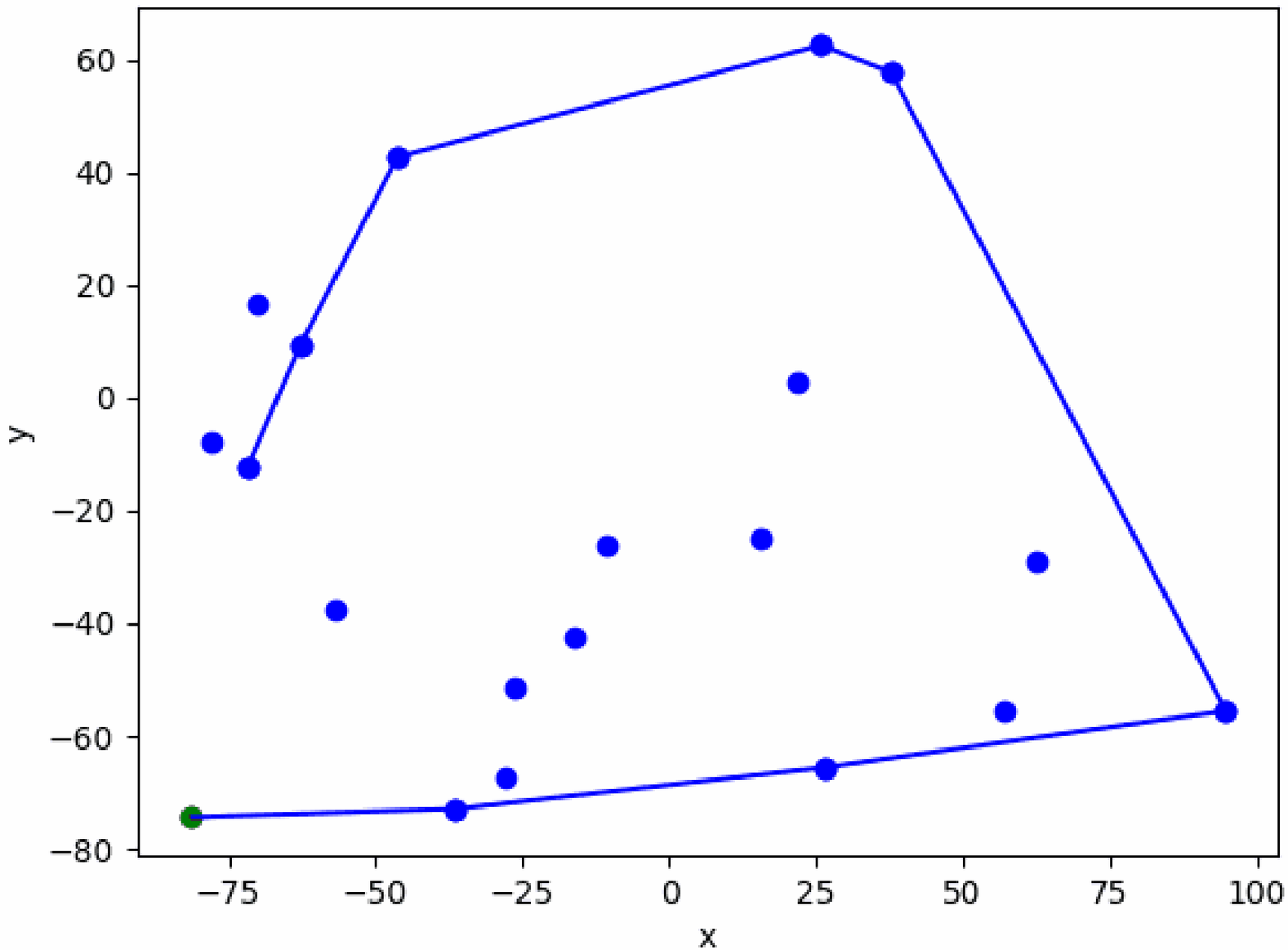

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

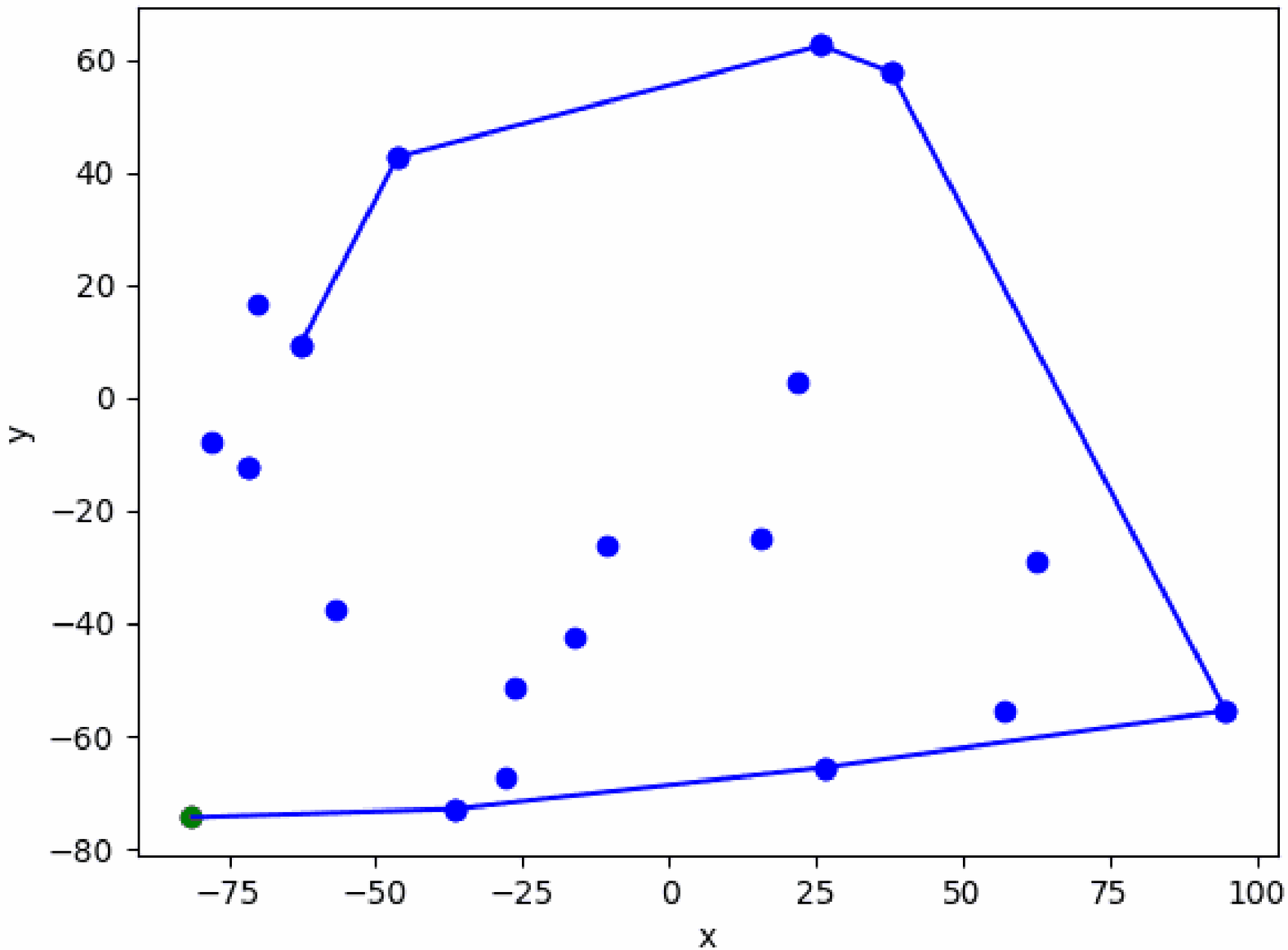

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

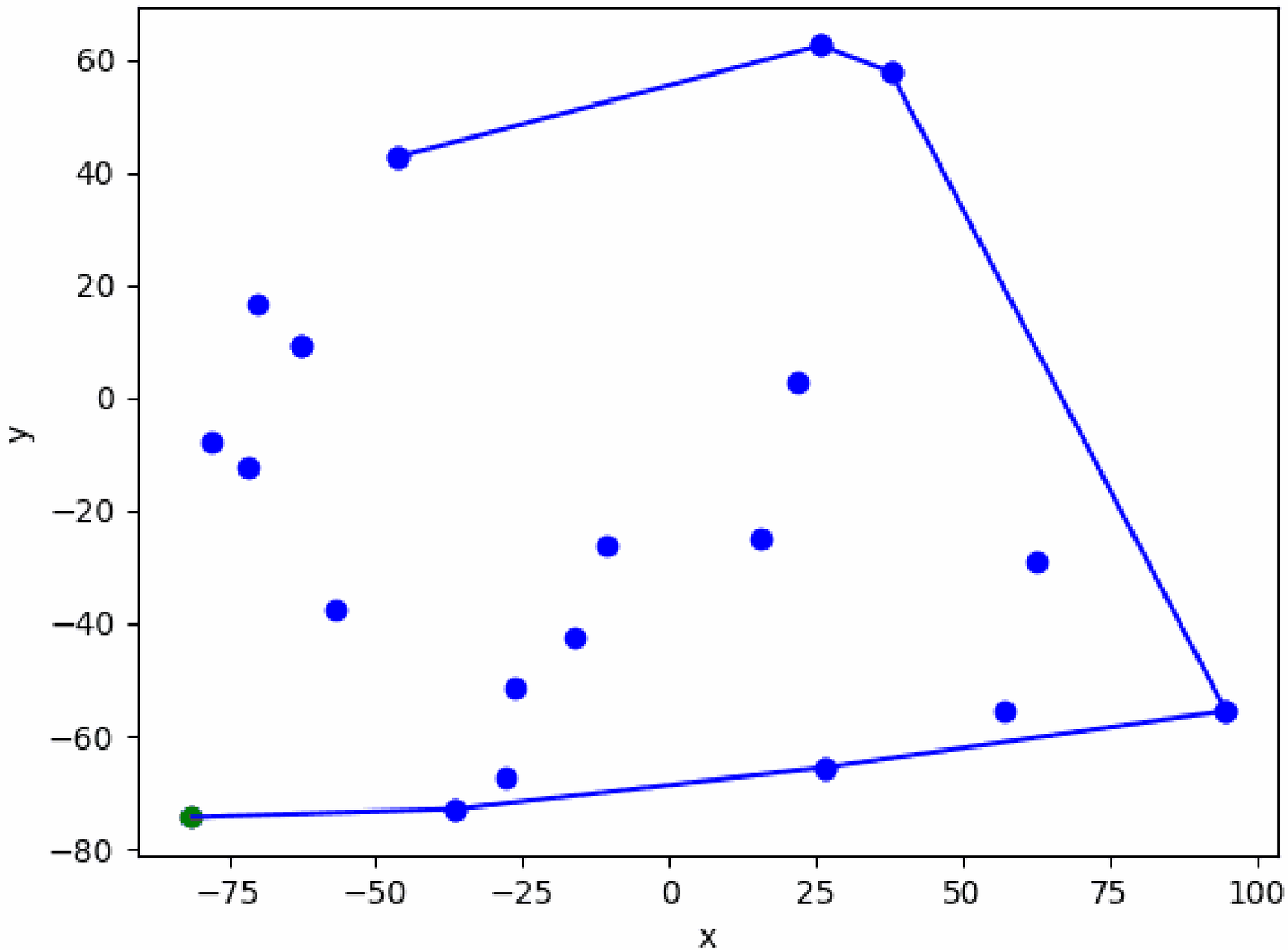

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

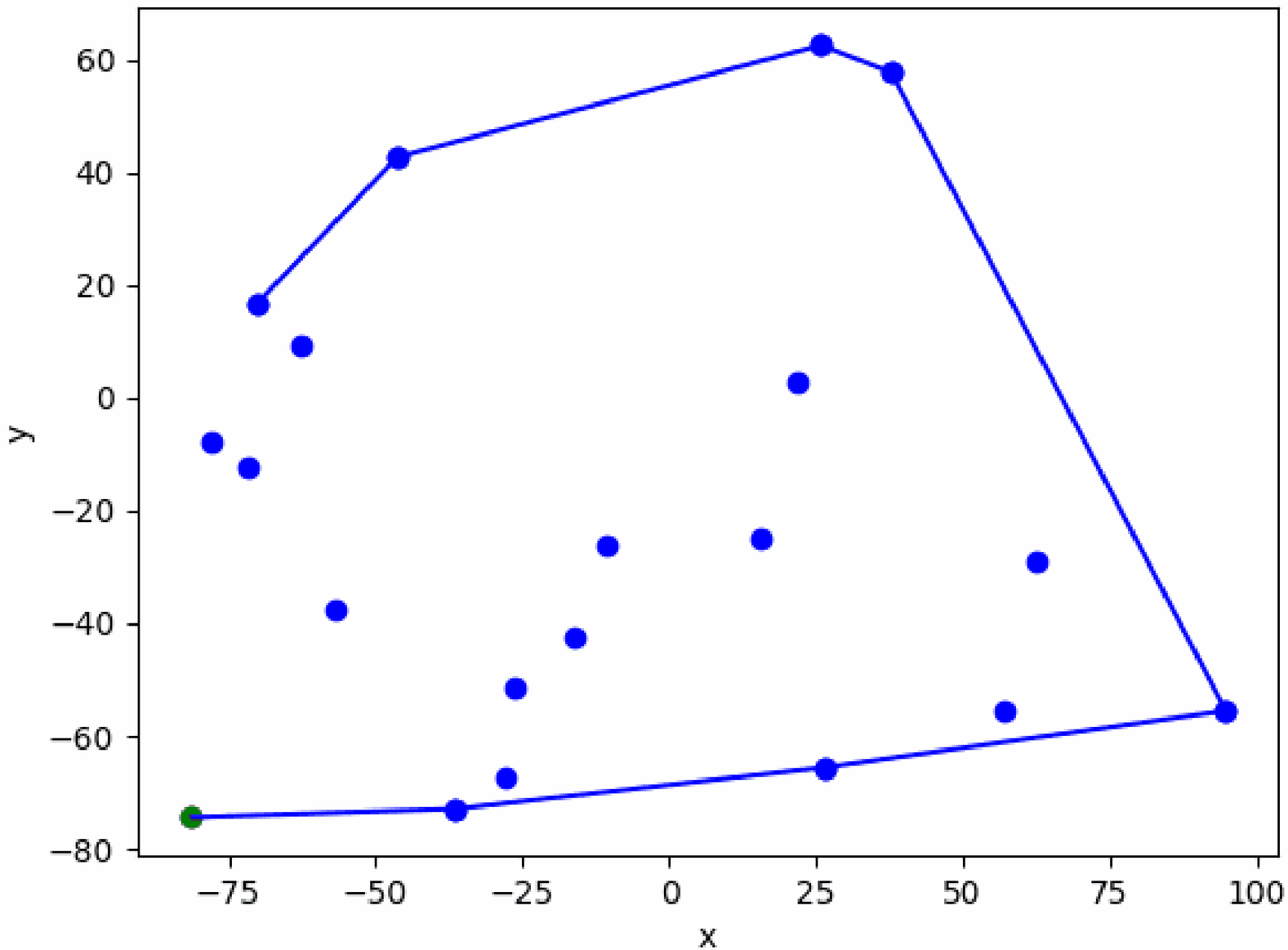

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

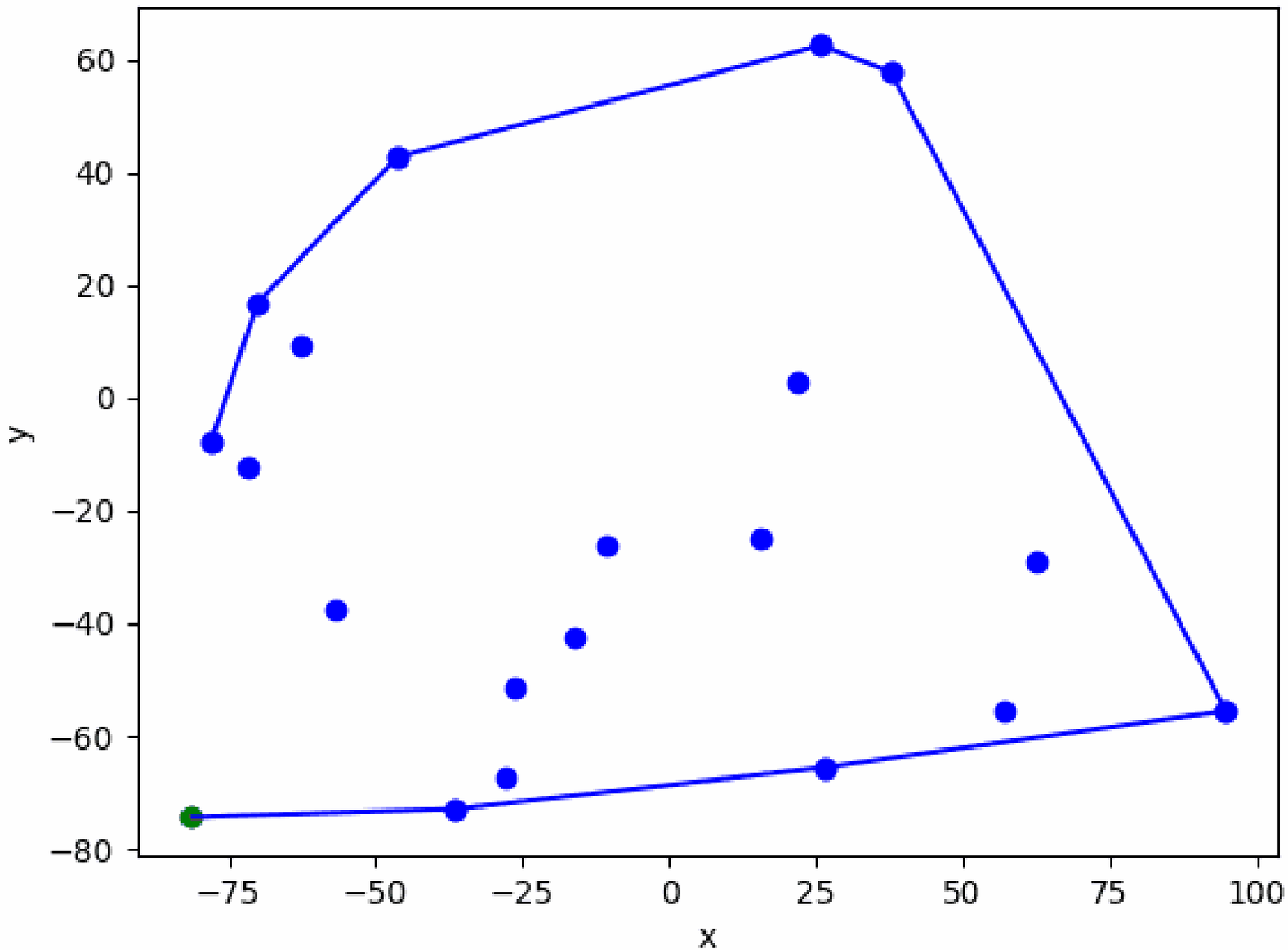

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

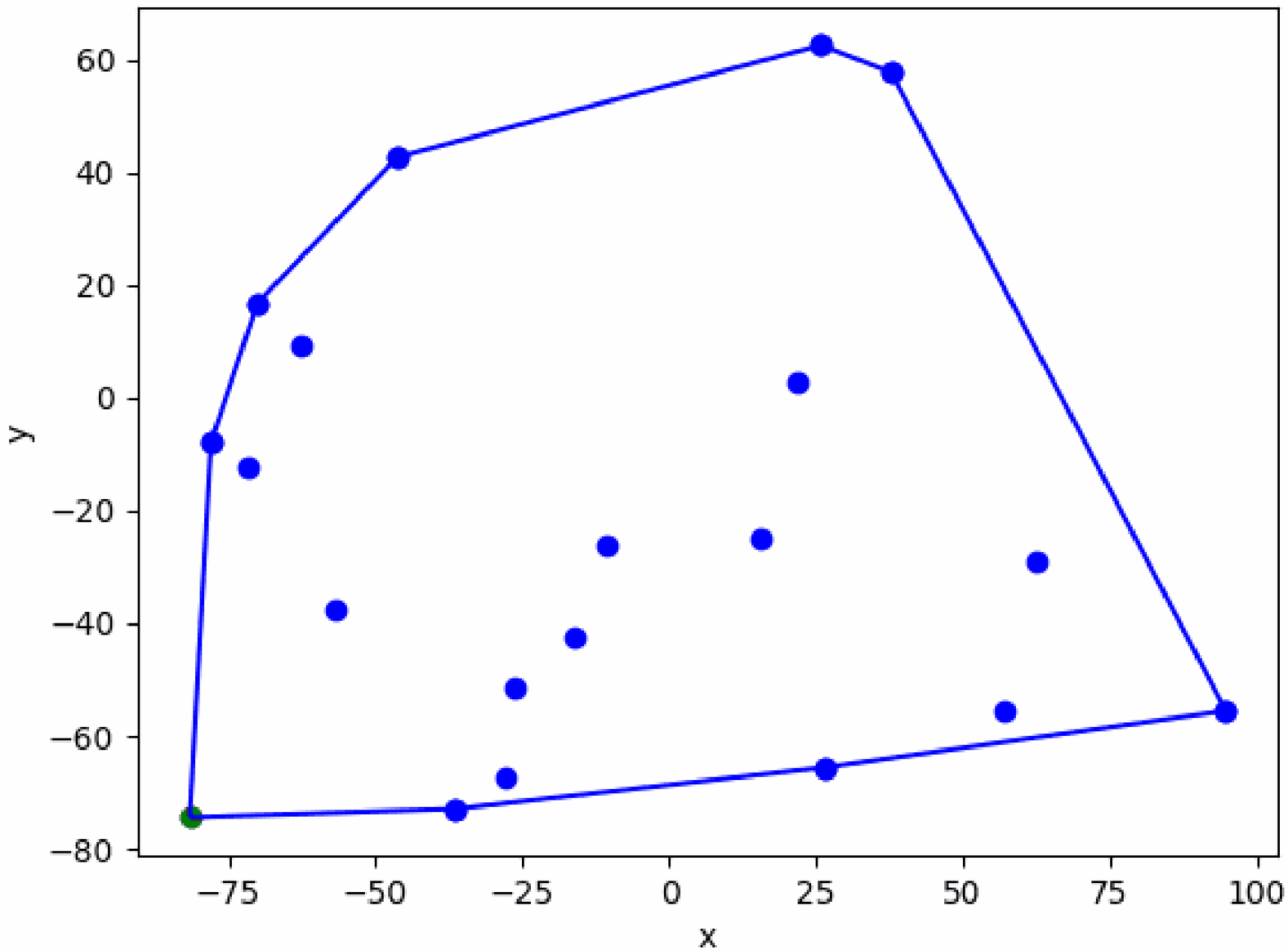

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x


60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x


60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

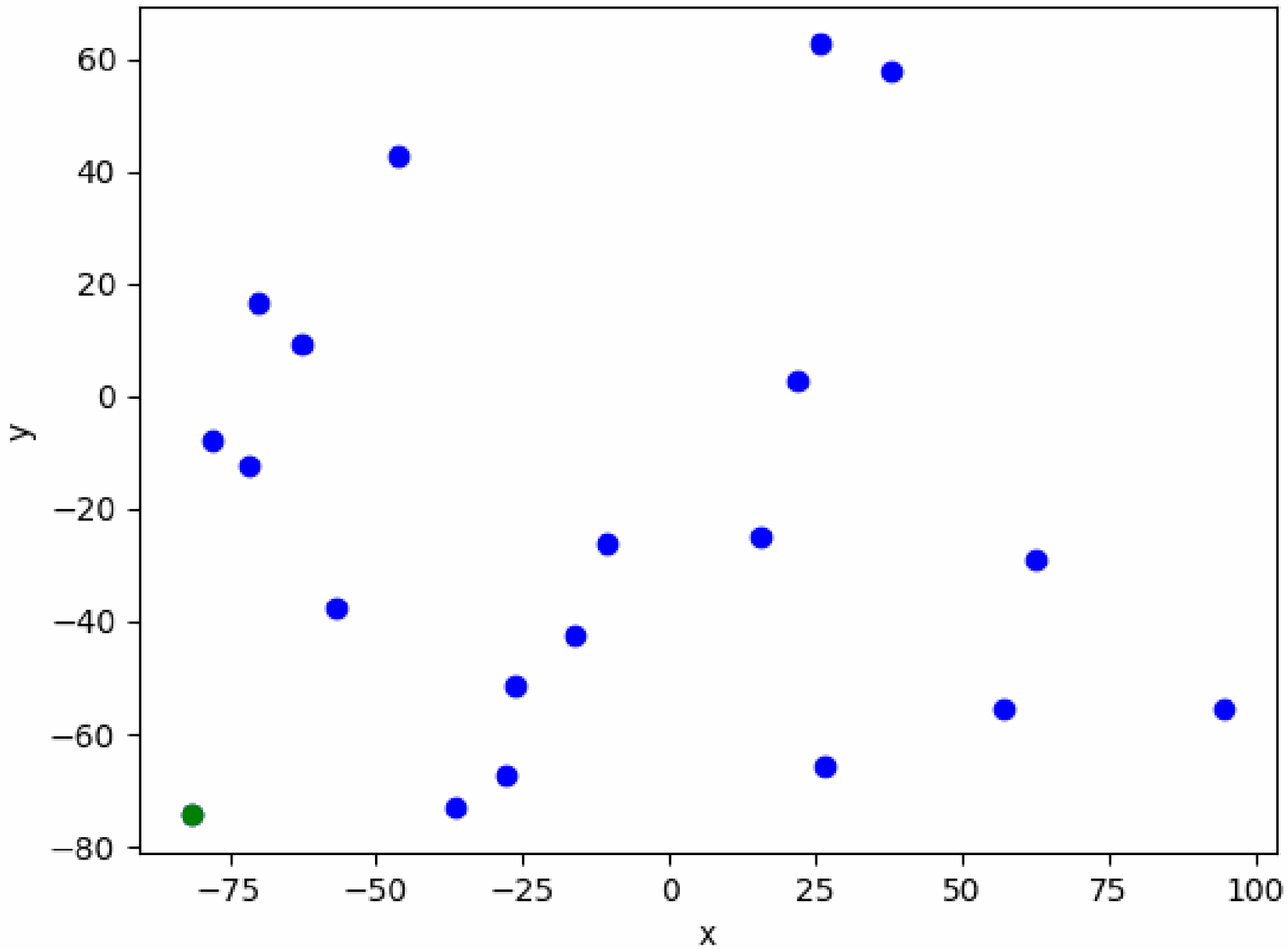

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

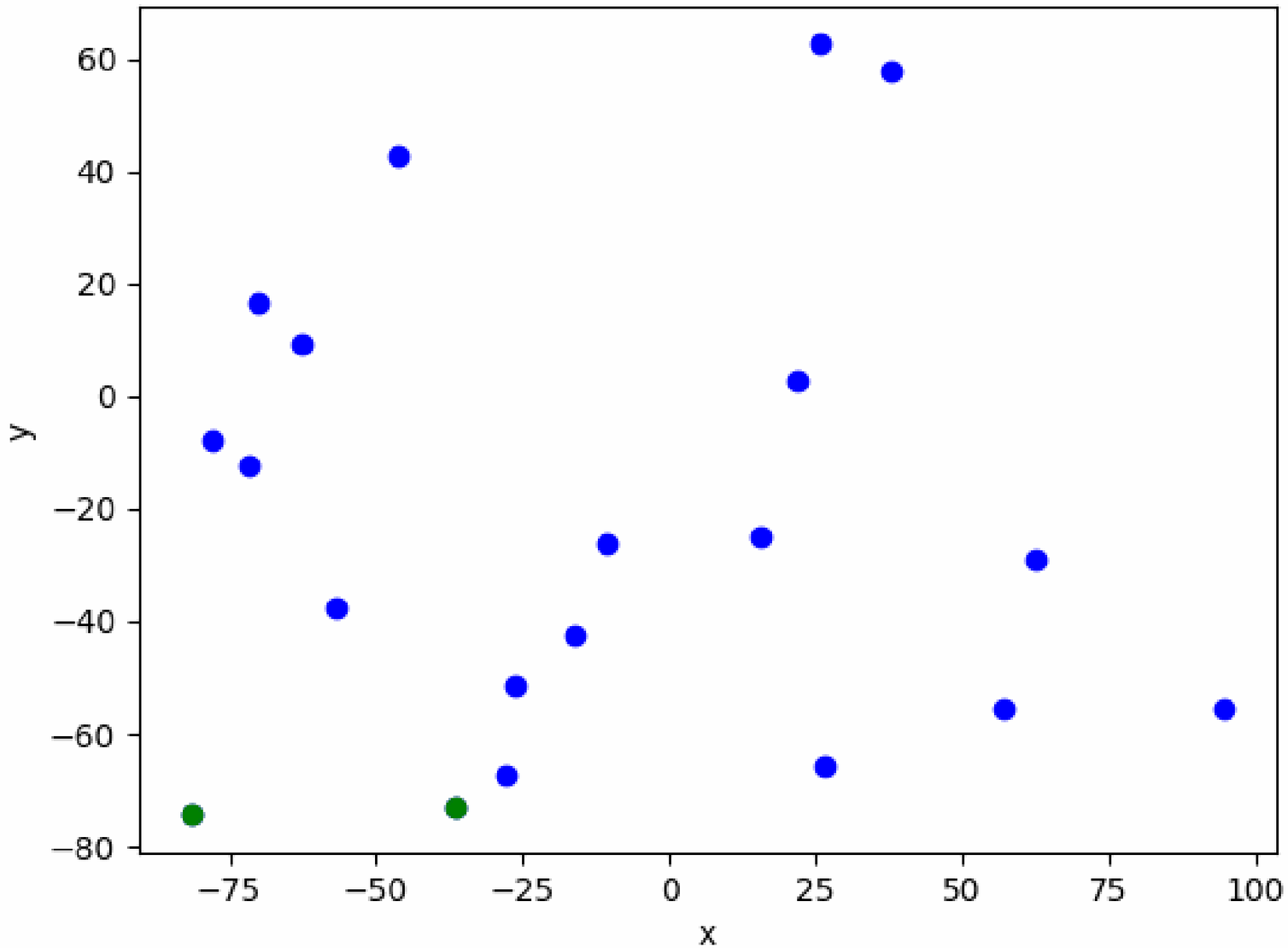

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

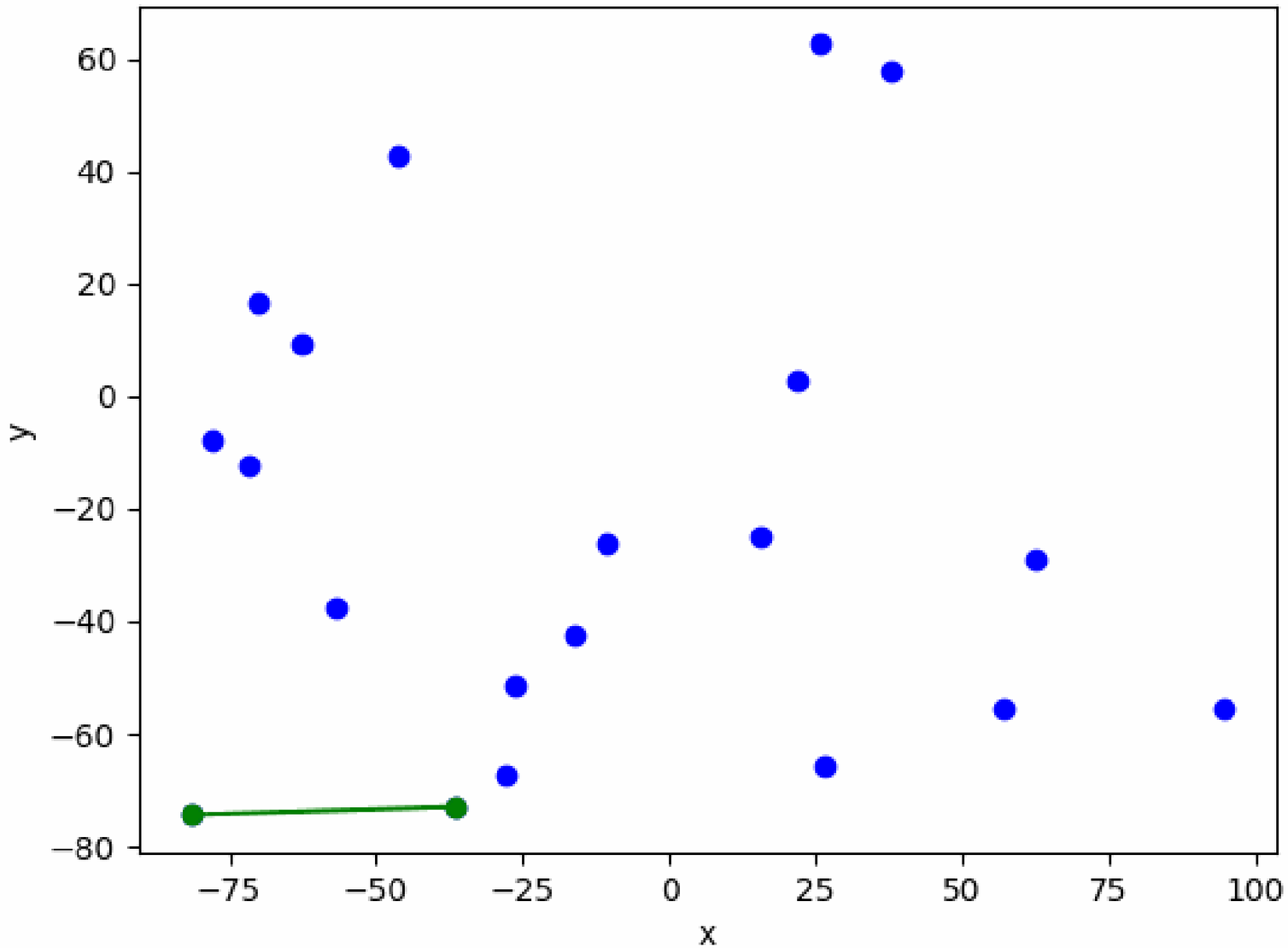

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

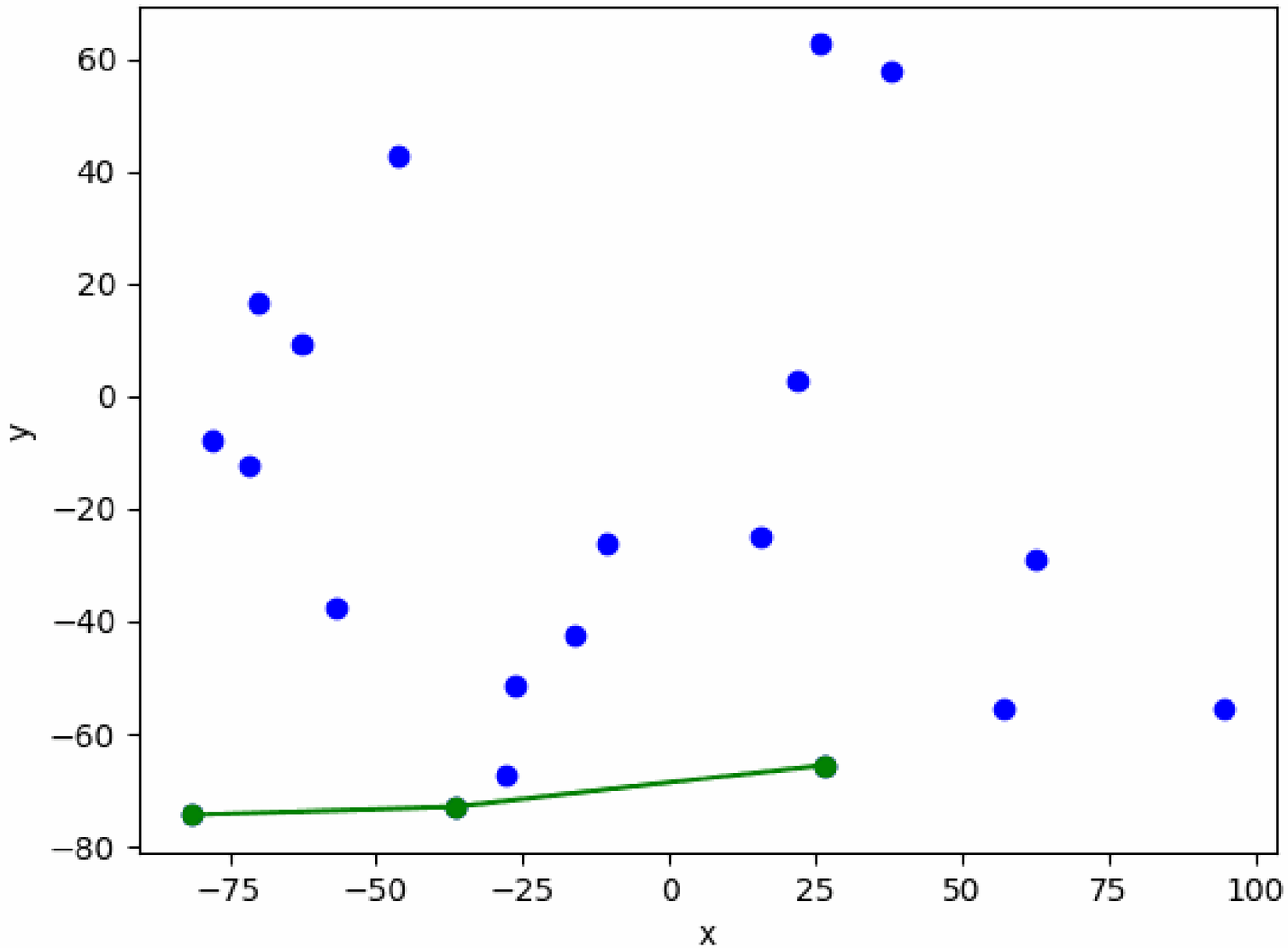
3 Algorytm Jarvisa

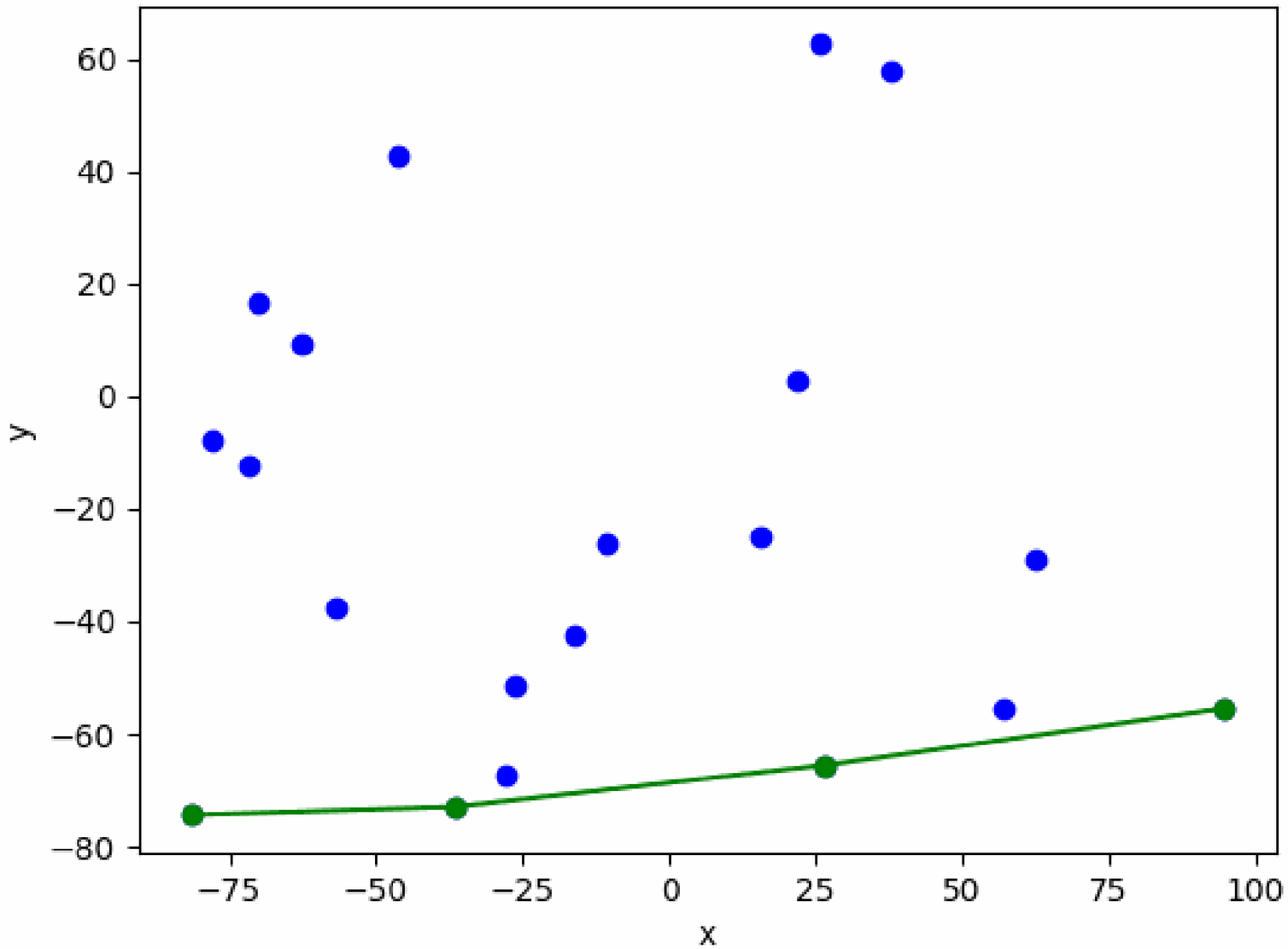
- Wyznaczamy najniżej położony punkt spośród wszystkich. Jeżeli istnieje więcej niż jeden taki punkt, to wybieramy ten o najmniejszej współrzędnej x.
- Powtarzaj, dopóki następny wykryty punkt jest różny od pierwszego punktu:
  - Znajdź punkt, dla którego kąt liczony przeciwnie do ruchu wskazówek zegara w odniesieniu do ostatniej krawędzi otoczki jest najmniejszy. Ten punkt jest kolejnym punktem należącym do otoczki.
- Znalezione punkty tworzą otoczkę wypukłą.
- Algorytm ma złożoność $O(nh)$, gdzie h jest liczbą punktów należących do otoczki.

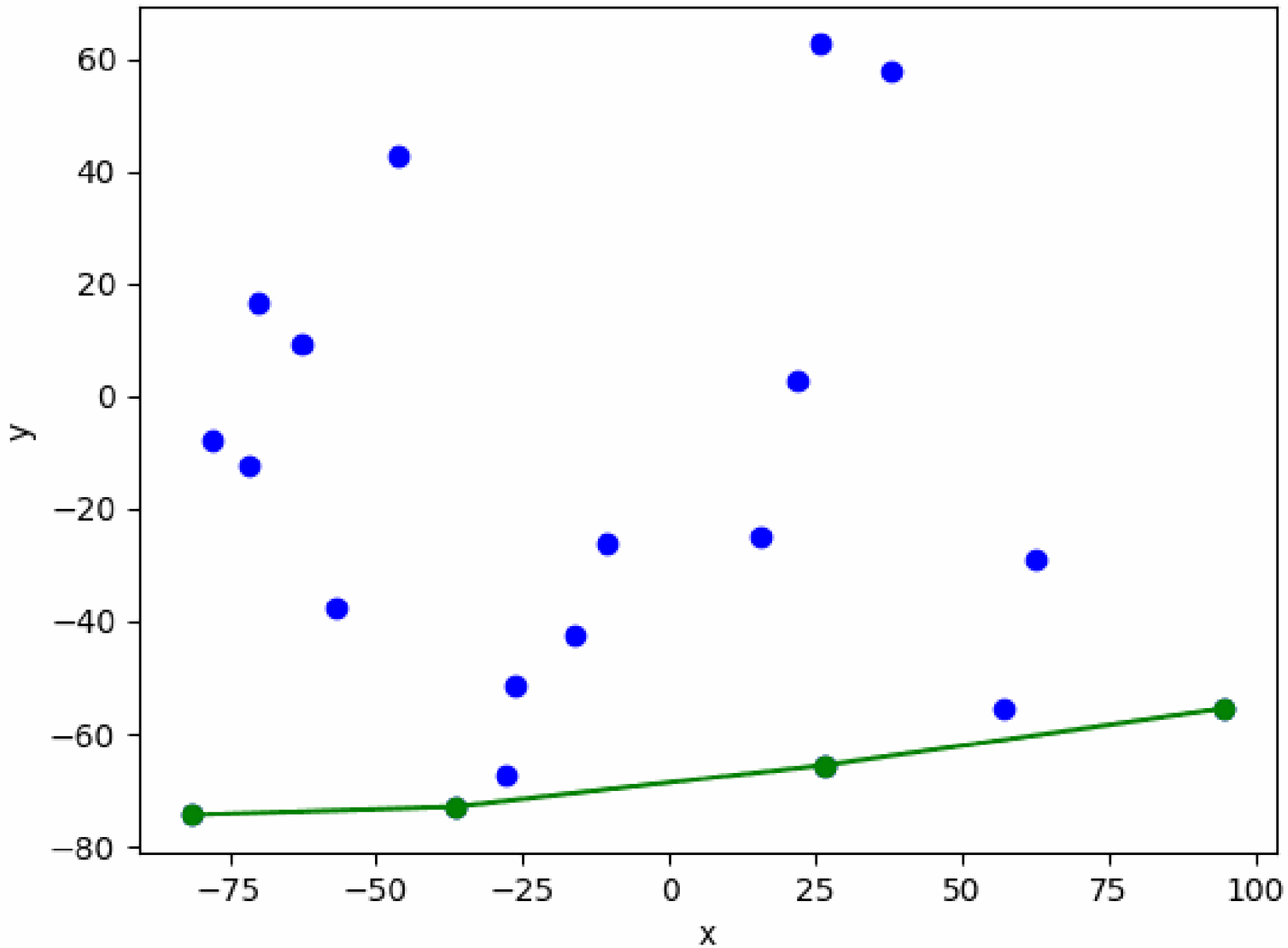

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

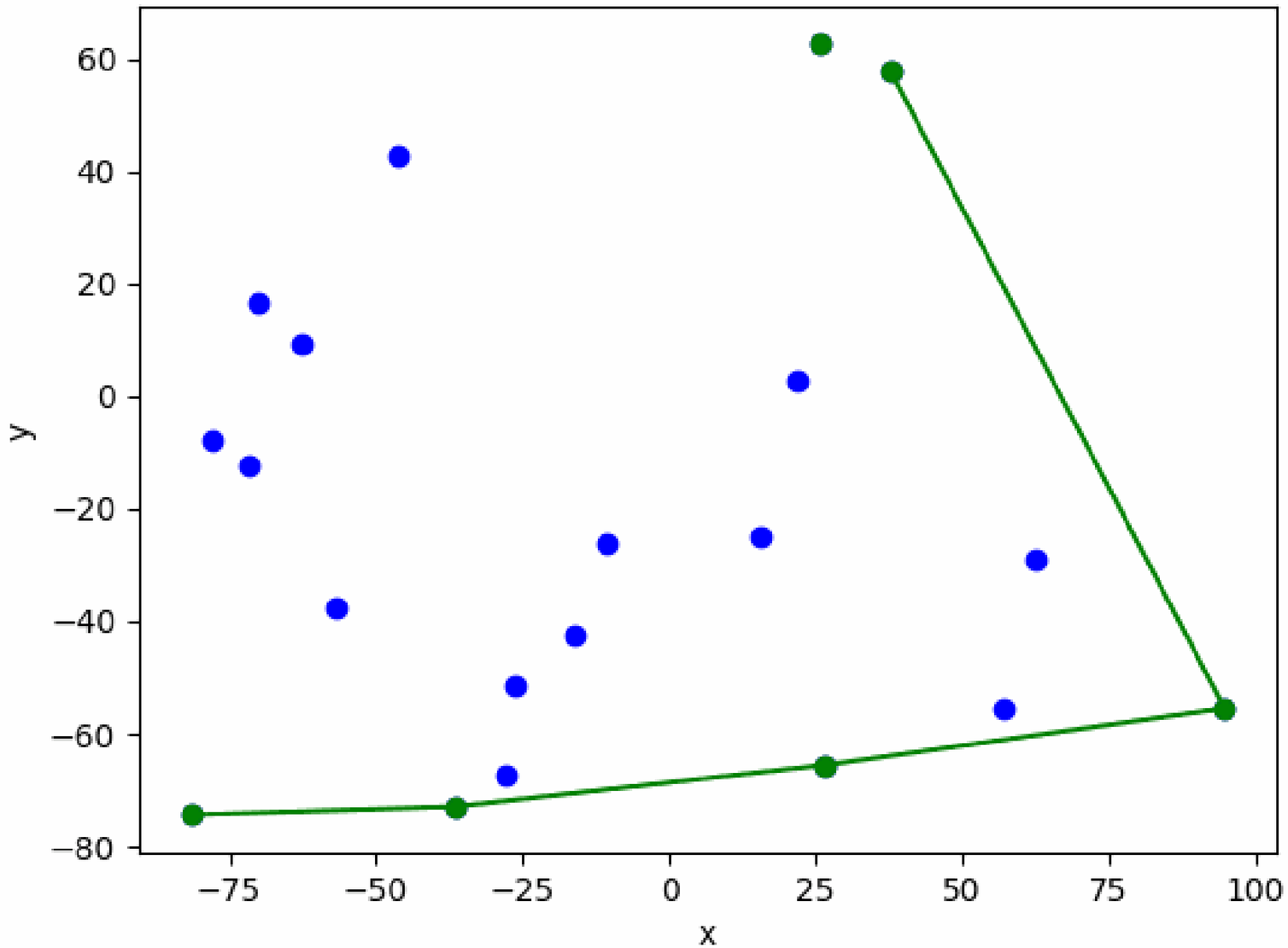

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

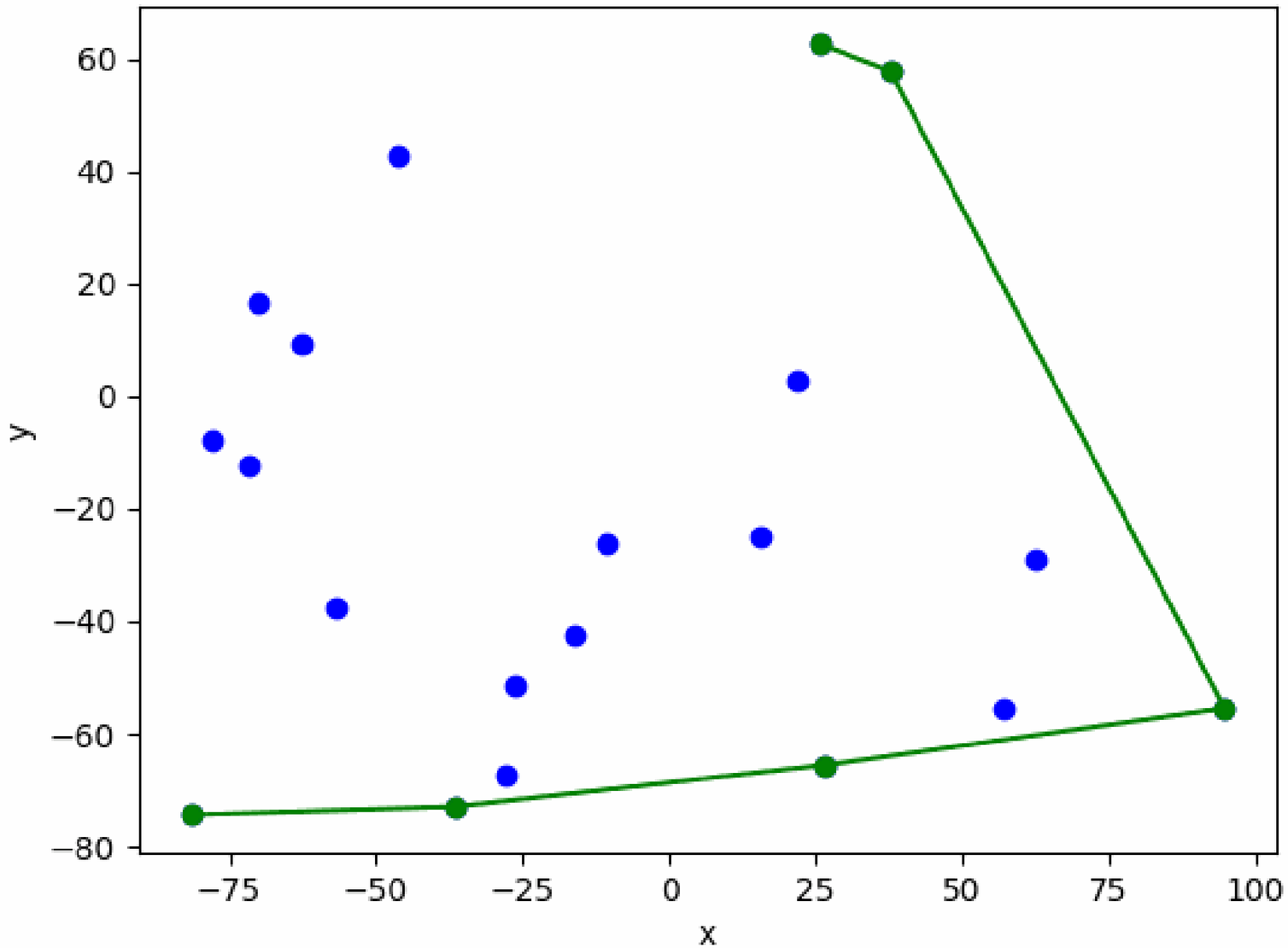

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

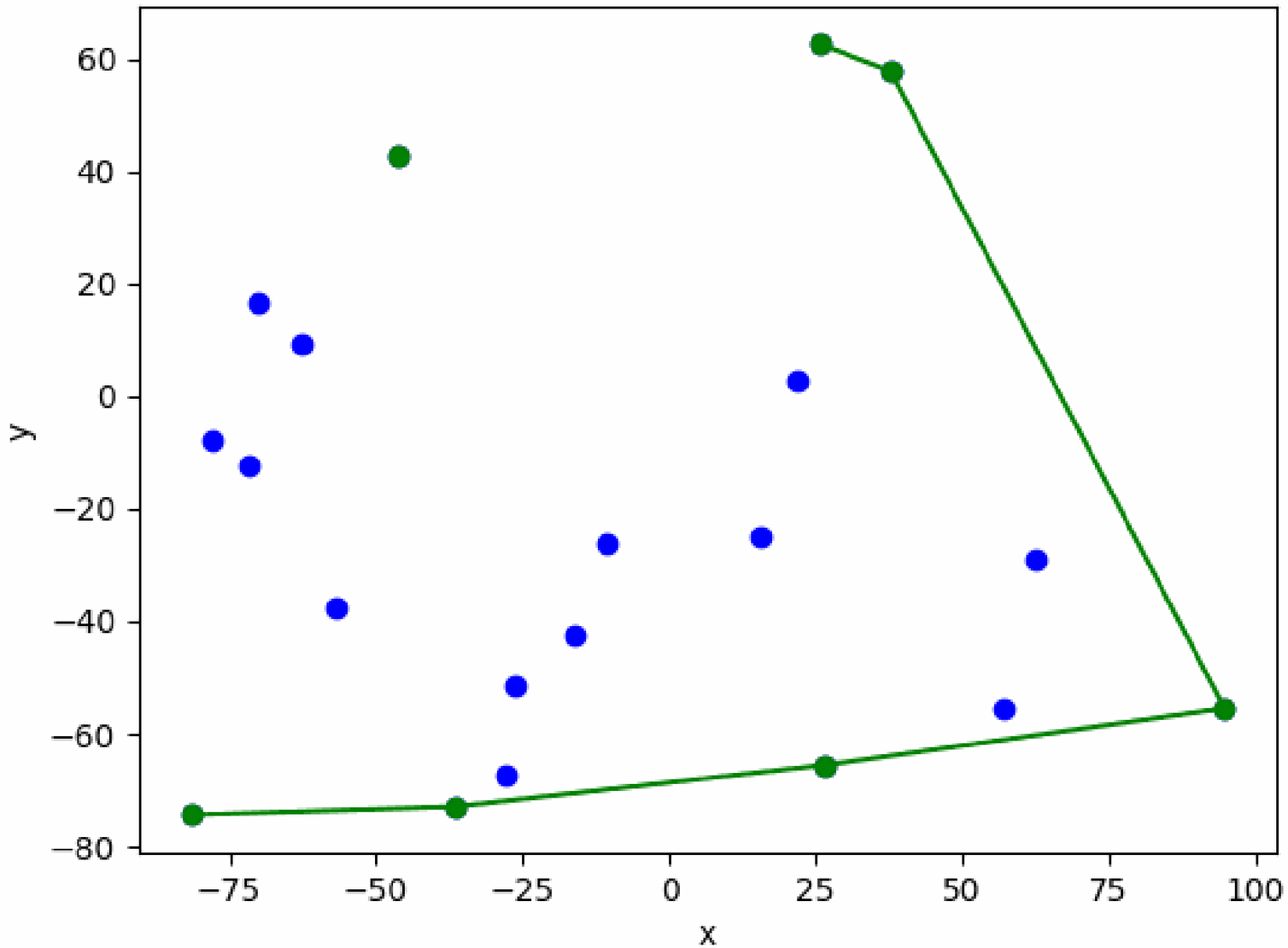

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

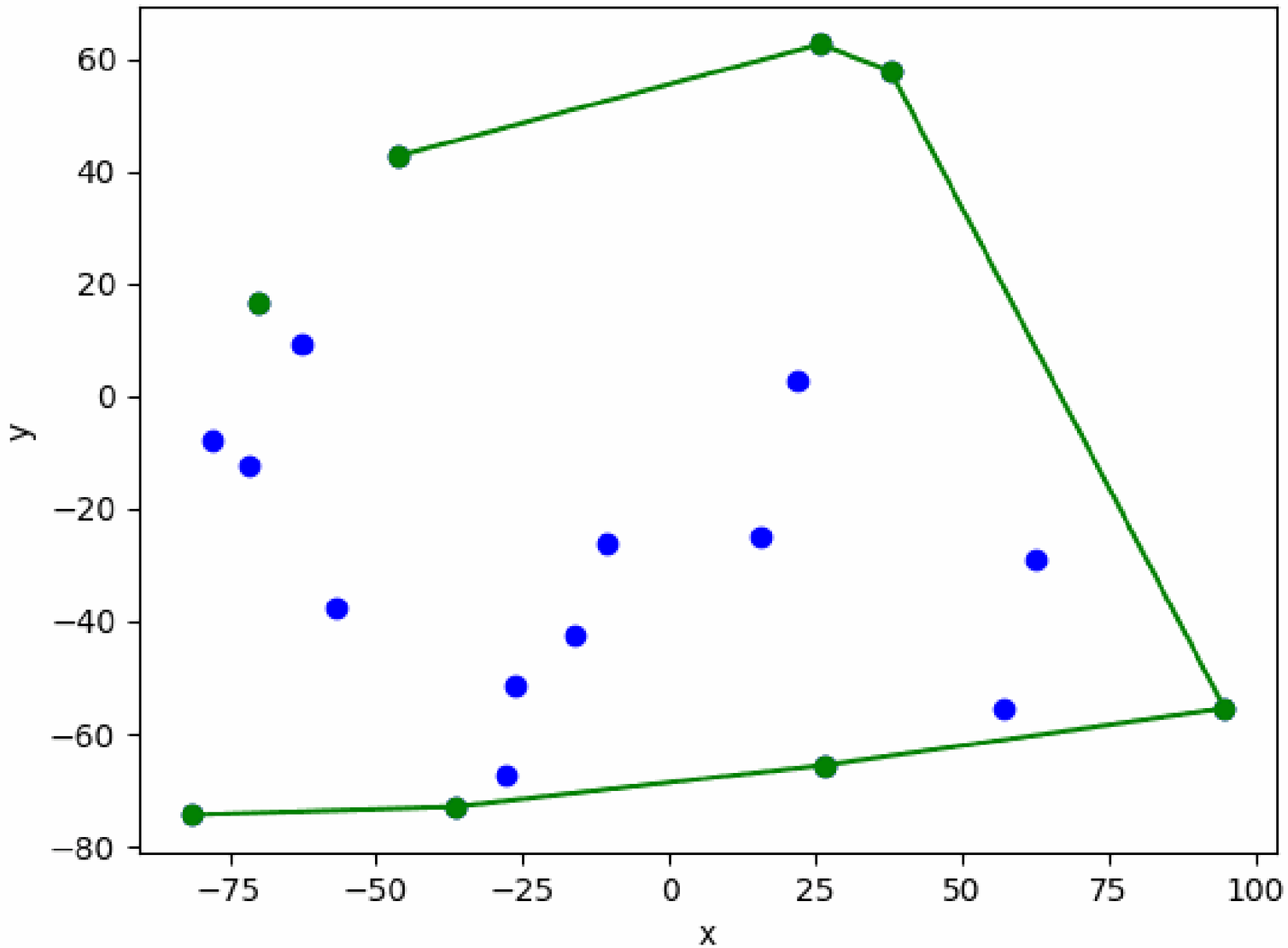

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

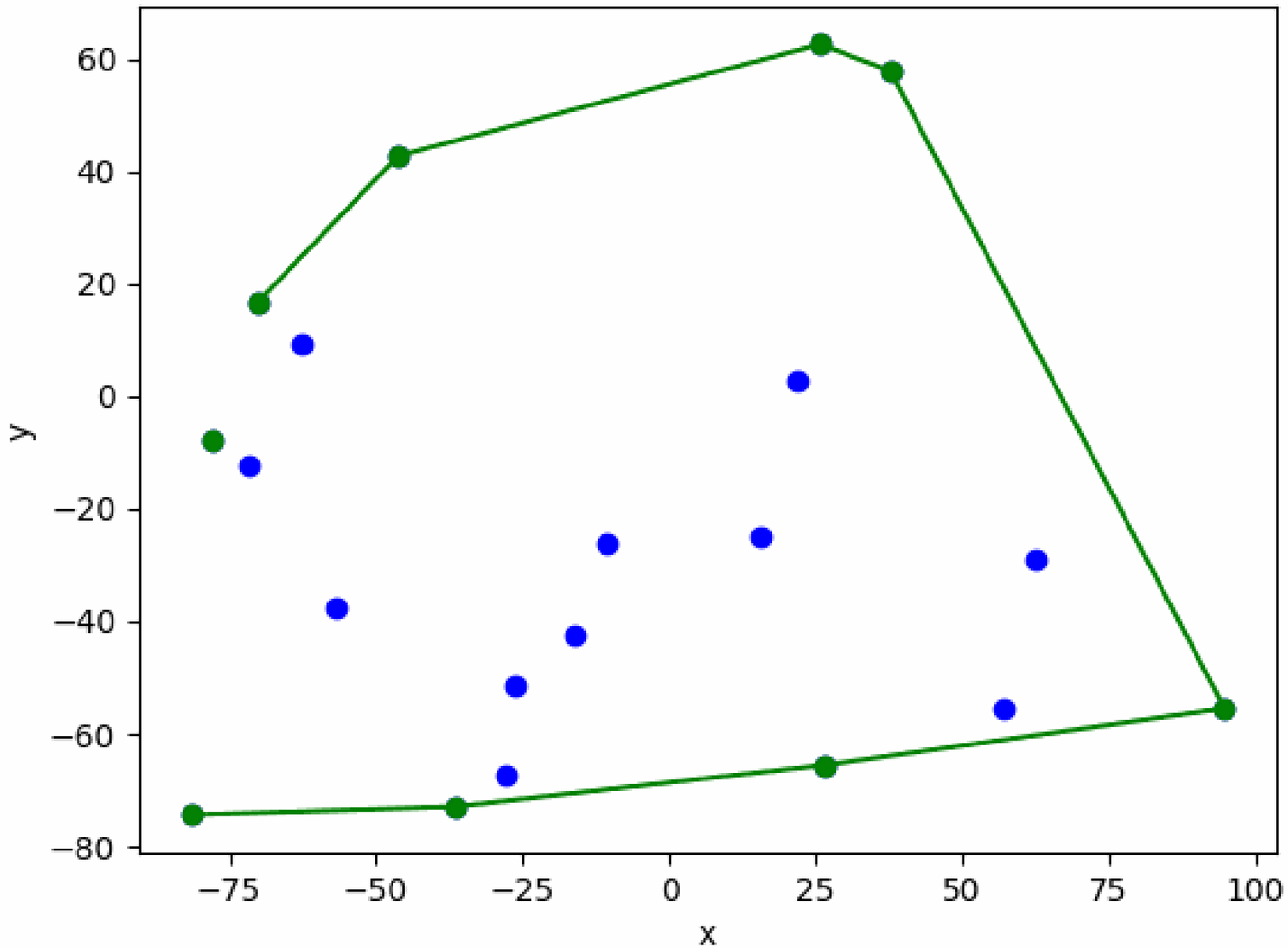

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

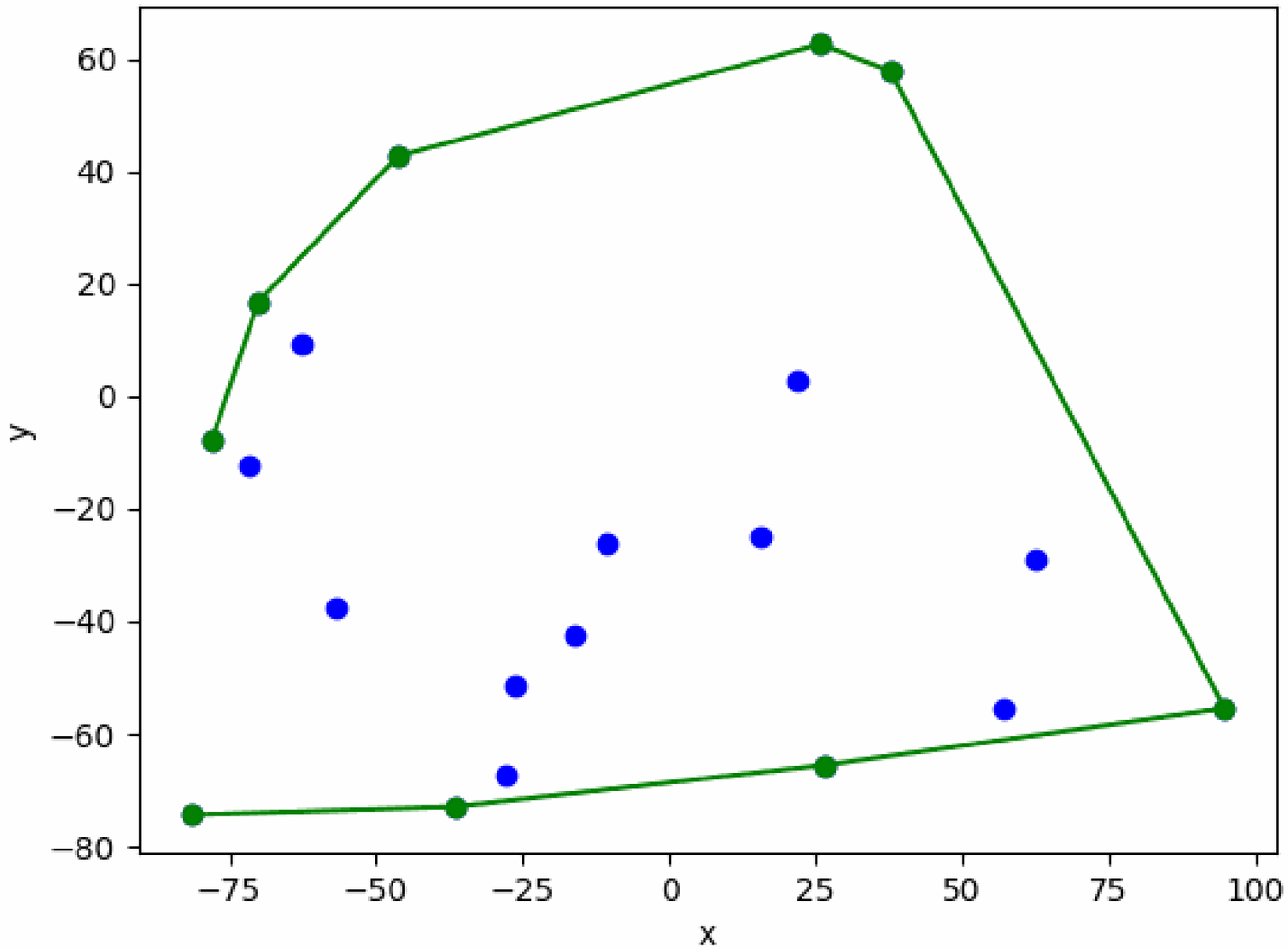

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

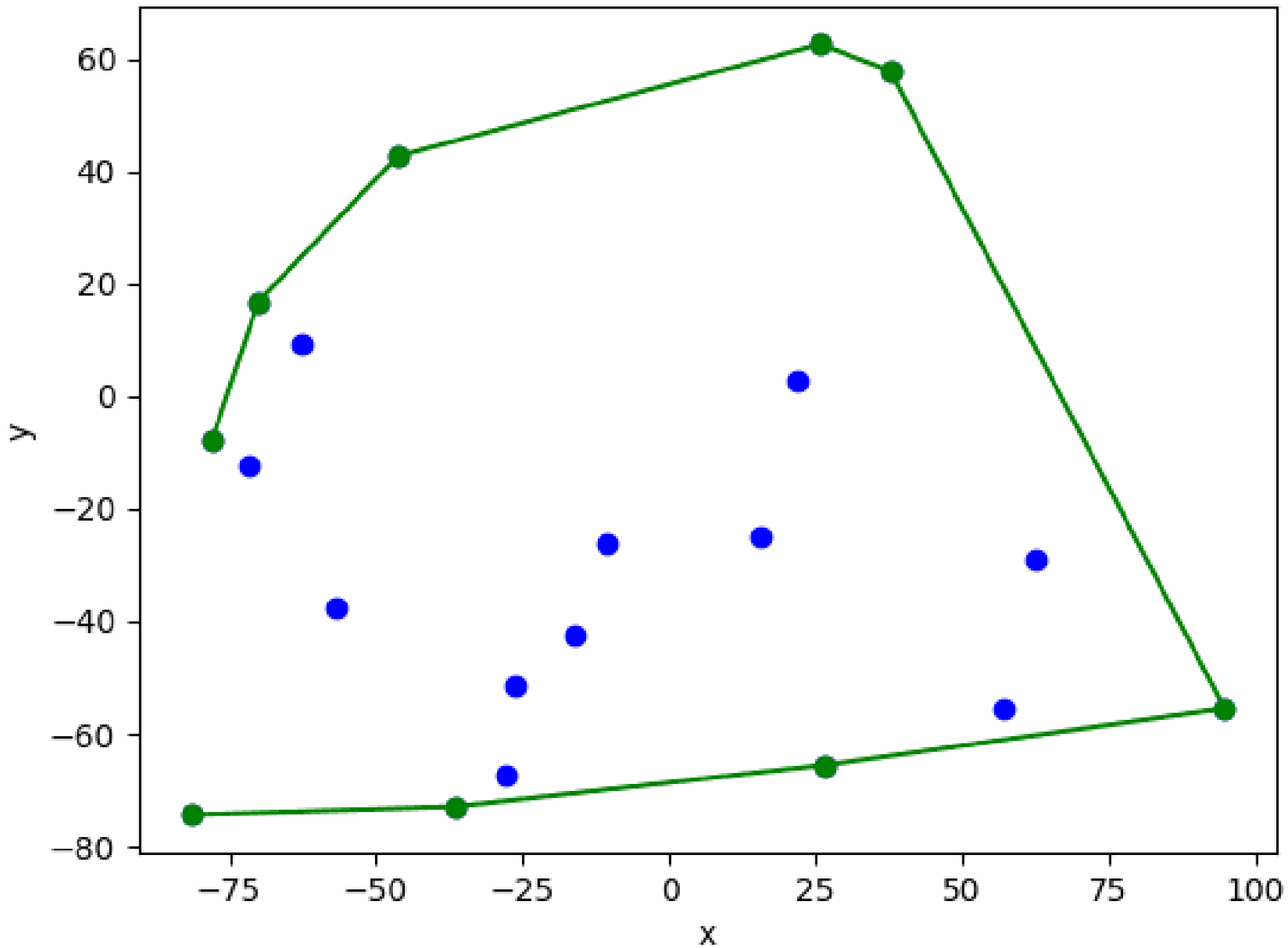

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

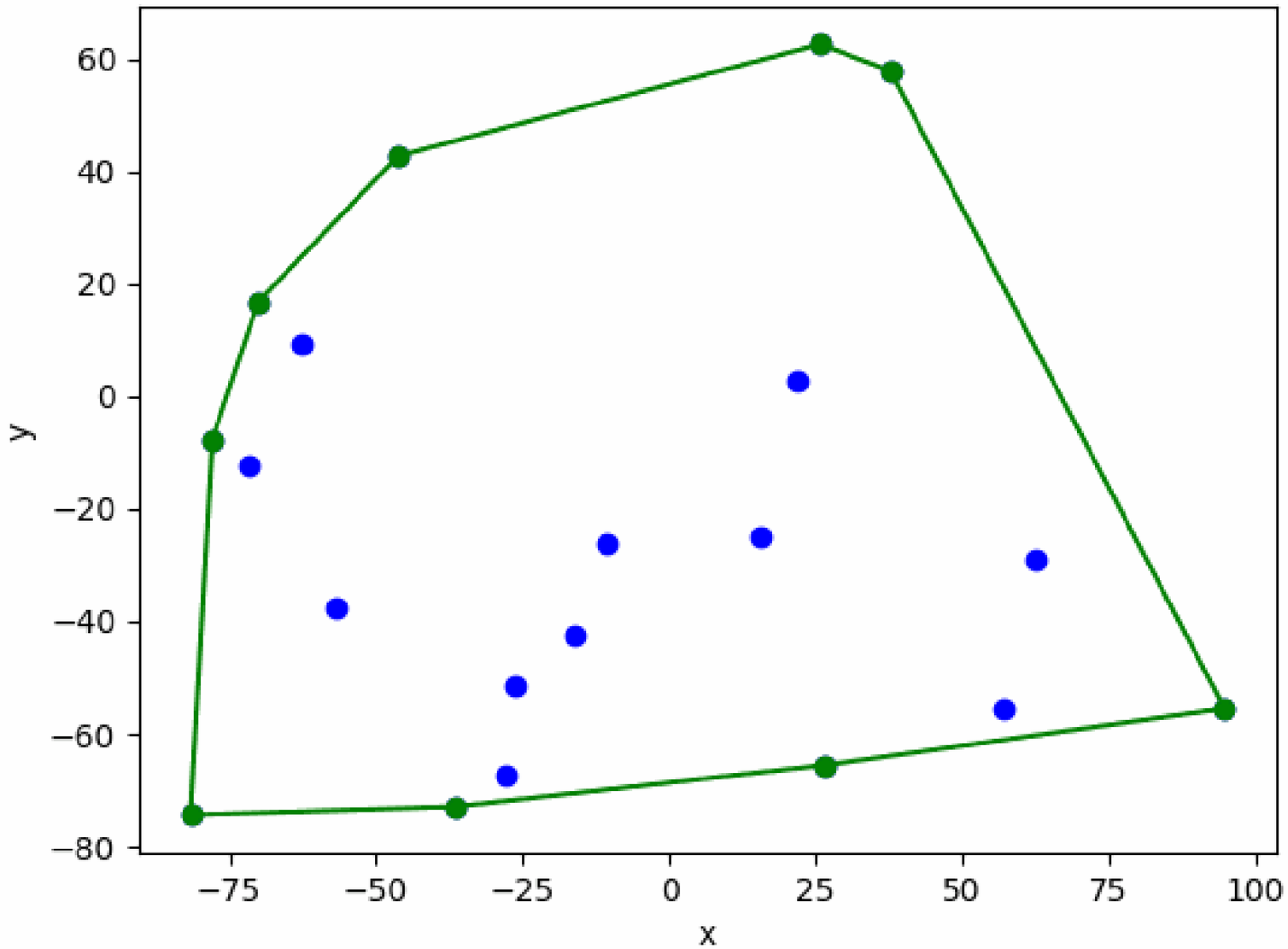

60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x


60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x


60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x


60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x


60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x


60
40
20
0
−20
−40
−60
−80
y
−75
−50
−25
0
25
50
75
100
x

4 Algorytm Chana

- Podziel wejściowy zbiór punktów na podzbiory o w miarę równych ilosciach punktów w nich zawartych, z których żaden nie zawiera więcej niż dane m.
- Wyznacz otoczki dla każdego takiego zbioru.
- Analogicznie jak w algorytmie Jarvisa wybierz najniższy punkt należący do danego zbioru punktów.
- Mając wierzchołek należący do otoczki możemy wyznaczyć następny:
  - Dla każdej podotoczki wyznaczamy punkt styczny do tej otoczki.
  - Spośród zbioru takich punktów oraz kolejnego punktu z podotoczki, do której dany punkt należy wybieramy taki punkt, że wszystkie pozostałe punkty znajdują się na lewo od odcinka utworzonego z wierzchołka należącego do otoczki głównej i niego.
  - Tak wybrany punkt jest kolejnym punktem otoczki głównej.
- Wyznaczamy kolejne punkty otoczki, dopóki nie znajdziemy punktu początkowego. Jeżeli nie znajdziemy punktu początkowego w m iteracjach to przerywamy wykonywanie algorytmu.
- Algorytm ma złożoność $O(n \log n)$

5 Algorytm dziel i rządź

- Sortujemy zbiór punktów.
- Dzielimy zbiór na mniejsze względem mediany zbioru. Powtarzamy czynność dopóki wielkość dowolnego zbioru jest mniejsza niż zadane k.
- Dla każdego małego zbioru wyznaczamy otoczki wypukłe innym algorytmem znajdowania otoczki wypukłej.
- Łączymy otoczki wypukłe w jedną otoczkę, tak aby zachować złożoność algorytmu $O(n \log n)$.
- Algorytm ma złożoność $O(n \log n)$

6 Algorytm przyrostowy

- Z trzech pierwszych punktów tworzymy otoczkę wypukłą.
- Iterujemy po pozostałych punktach wykonując następujące czynności:
  - Jeżeli punkt należy do wnętrza wyznaczonej już otoczki to przechodzimy do kolejnego punktu.
  - Jeżeli punkt nie należy do wnętrza otoczki to znajdujemy styczne do otoczki z danego punktu i aktualizujemy otoczkę o znalezione krawędzie.
- Zwracamy punkty otoczki.
- Algorytm ma złożoność $O(n \log n)$

7 Algorytm górnej i dolnej otoczki

- Sortujemy punkty rosnąco po współrzędnych x. W przypadku takich samych pierwszych współrzędnych, porównujemy drugie współrzędne.
- Pierwsze dwa punkty z posortowanego zbioru wpisujemy do zbioru punktów otoczki górnej oraz dolnej.
- Iterujemy po zbiorze punktów zaczynając od punktu trzeciego:
  - Dopóki górna (dolna) otoczka ma co najmniej 2 punkty oraz bieżący punkt nie znajduje się po prawej (lewej) stronie odcinka skierowanego utworzonego przez ostatnie dwa punkty otoczki, to usuwamy punkt z górnej (dolnej) otoczki. W przeciwnym przypadku dodajemy punkt do otoczki górnej (dolnej).
- Odwracamy kolejność wierzchołków w otoczce dolnej.
- Łączymy zbiory punktów otoczki górnej oraz dolnej i zwracamy złączony zbiór punktów otoczki.
- Algorytm ma złożoność $O(n \log n)$

8 Algorytm Quickhull

- Znajdź dwa punkty skrajne A oraz B - pierwszy o najmniejszej pierwszej współrzędnej, drugi o największej pierwszej współrzędnej.
- Uruchamiamy rekurencyjną funkcję znajdowania łuku należącego do otoczki między danymi punktami należącymi do tej otoczki p, q na prawo od odcinka $|pq|$. Otoczka jest sumą punktów w wyniku działania funkcji rekurencyjnej dla odcinka $|ab|$ oraz dla odcinka $|ba|$.
- Funkcja rekurencyjna polega na:
  - Wyznaczeniu najbardziej odległego punktu na prawo od $|pq|$, jeśli takich punktów nie ma to zwracamy pustą tablicę.
  - Punkty p, q należą do otoczki, to znaleziony punkt r musi do niej należeć. Usuwamy wszystkie wierzchołki wewnątrz trójkąta pqr
  - Szukany łuk to suma działania funkcji rekurencyjnej dla punktów p, r oraz dla punktów r, q.
  - Na koniec zwracamy znaleziony łuk.
- Algorytm ma złożoność oczekiwaną rzędu $O(n \log n)$, natiomiast złożoność pesymistyczną rzędu $O(n^2)$