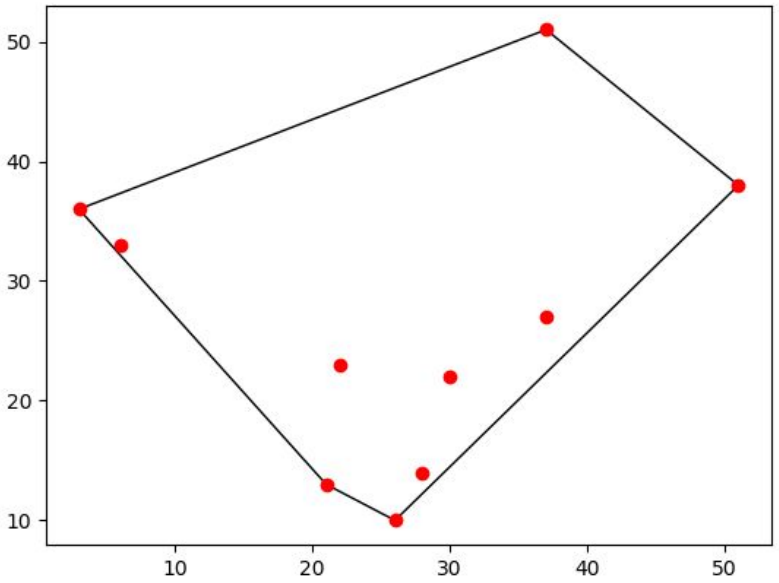
I. Wstęp teoretyczny

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



Do znajdowania takiej właśnie otoczki wypukłej możemy użyć między innymi algorytmu Grahama, pierwszy raz opublikowanego w 1972 przez Ronalda Grahama. Teoretyczna złożoność tego algorytmu wynosi $O(nlogn)$, gdzie n to wielkość badanego zbioru.

W algorytmie Grahama problem wyszukiwania otoczki wypukłej jest rozwiązywany z użyciem stosu, który zawiera kandydatów na wierzchołki otoczki. Każdy punkt ze zbioru wejściowego jest raz wkładany na stos, natomiast punkty nie będące wierzchołkami szukanej otoczki są usuwane. Algorytm po zakończeniu zwraca kolekcję punktów występujących na otoczce w kolejności odwrotnej do ruchu wskazówek zegara.

Pseudokod algorytmu

Wejście:

Q - wejściowy zbiór punktów

S = Ø - stos przechowywujący punkty otoczki

Algorytm:

1. Niech p_0 będzie punktem z Q o najmniejszej współrzędnej y. Jeżeli kilka punktów ma taką współrzędną y, wybieramy ten o najmniejszej współrzędnej x.
2. Niech $[p_1, ..., p_n]$ będzie zbiorem pozostałych punktów z Q, posortowanych po współrzędnej polarnej według punktu p_0
3. S.push(p_0)
4. S.push(p_1)
5. S.push(p_2)
6. for i=3; i < n; i++

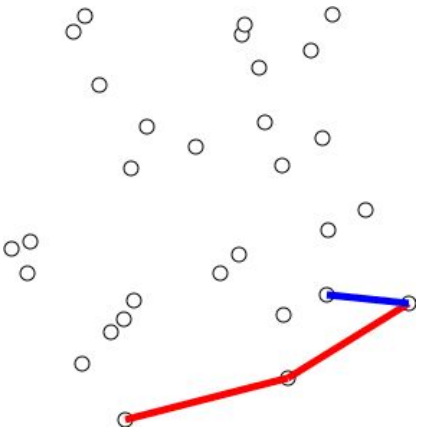
   while p_i jest na prawo od wektora $S[-2] \rightarrow S[-1]$ i S.length > 2

   S.pop()

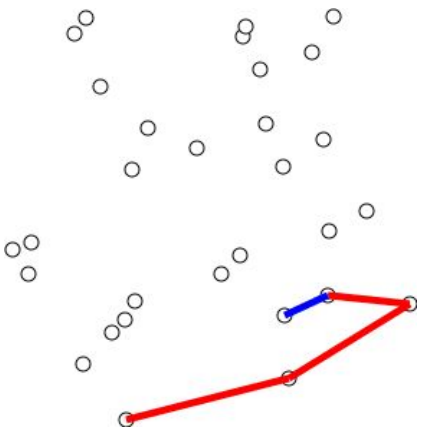
   S.push(p_i)
7. return S

Przykład działania algorytmu:

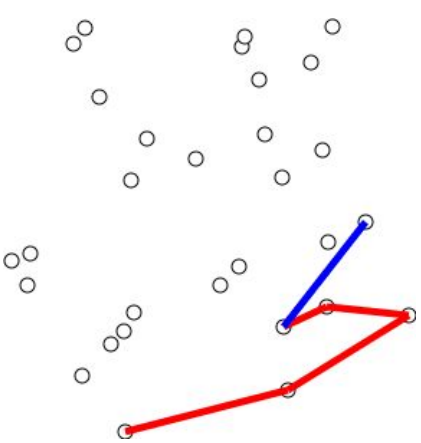
1)



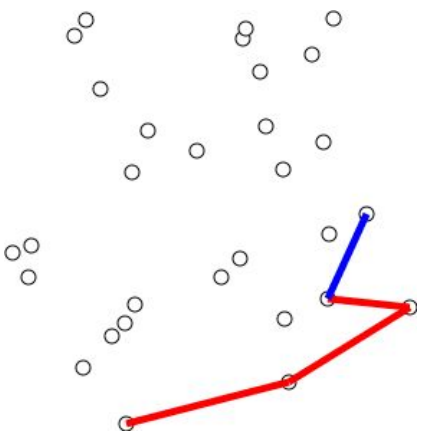
2)



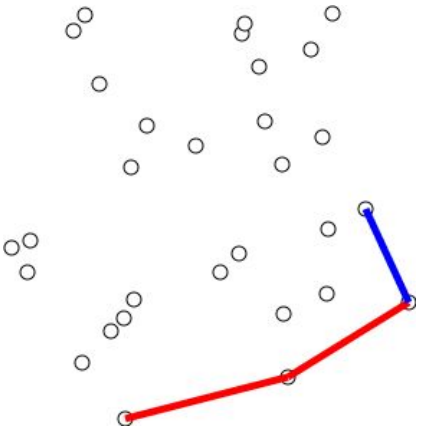
3)



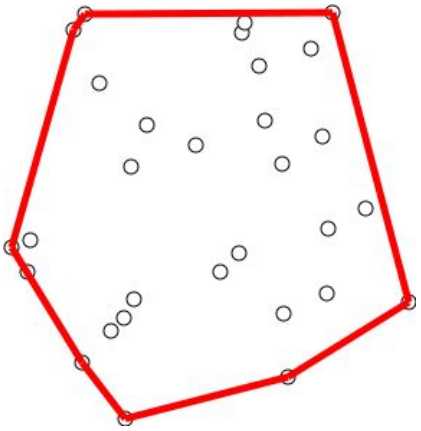
4)



5)



6) W kilku krokach:



II. Implementacja

W implementacji algorytmu wykorzystane zostały dodatkowe funkcje pomocnicze:

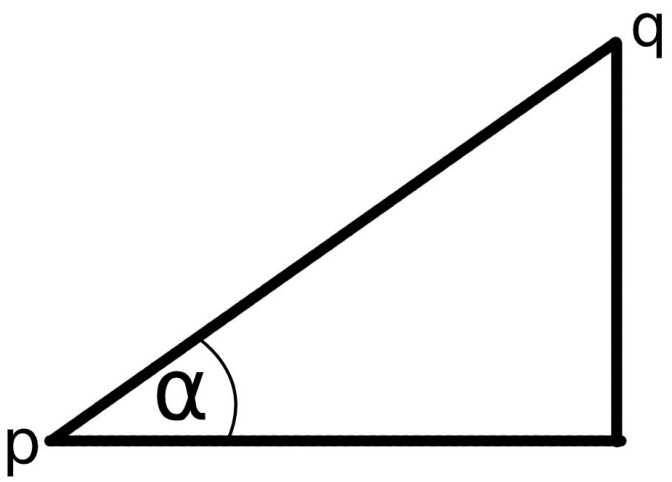
- findMinY() - Funkcja znajdująca punkt p_0 taki jak wyżej.
- quicksort(L, start, end, anchor) - Funkcja sortująca punkty zbioru wejściowego. Przyjmuje zbiór do posortowania, indeks pierwszego i ostatniego elementu oraz punkt według którego obliczane będą kąty polarne dla pozostałych punktów. W tym przypadku wykorzystany algorytm sortujący to quicksort który działa w czasie $O(nlogn)$. Złożoność algorytmu grahama to złożoność algorytmu sortowania użytego przy implementacji. Oznacza to, że jeśli chcemy osiągnąć złożoność zgodną ze złożonością teoretyczną, możemy wykorzystywać tylko te algorytmy sortujące, których złożoność wynosi $O(nlogn)$.
- det(p,q,r) - Funkcja licząca wyznacznik dla trzech punktów. Wyznacznik wykorzystywany jest do stwierdzenia czy dany punkt znajduje się po prawej czy po lewej stronie wektora utworzonego z dwóch pozostałych punktów. W praktyce

jeśli chcemy obliczyć czy dany punkt r leży na prawo czy na lewo od wektora $p \rightarrow q$ liczymy wyznacznik:

$$\det(p,q,r) = \det \begin{bmatrix} x_p & y_p & 1 \\ x_q & y_q & 1 \\ x_r & y_r & 1 \end{bmatrix}$$

Jeśli obliczony wyznacznik jest większy od zera to punkt r leży po lewej stronie danego wektora, jeśli jest większy od zera to po prawej stronie, a kiedy jest równy zero to punkt r jest współliniowy.

Dodatkowo punkt reprezentowany jest jako obiekt z wartościami x, y oraz z metodą, która dla danego punktu oblicza kąt polarny wg. drugiego punktu. Obliczenie kąta polarnego w tym przypadku sprowadza się do obliczenia $arctg(\alpha)$:



Sama implementacja algorytmu w moim przypadku wygląda następująco:

```
def find_hull(L):
    """ Znajduje i zwraca otoczke wypukla """
    anchor = findMinY(L)
    L.remove(anchor)
    quicksort(L, 0, len(L)-1, anchor)
    hull = [anchor, L[0]]
    for x in L:
        while len(hull)>1 and det(hull[-2], hull[-1], x) <= 0:
            hull.pop()
        hull.append(x)
    if det(hull[-2], hull[-1], anchor) == 0 and len(hull)>2:
        hull.pop()
    return hull
```

Dodatkowo w celach prezentacji danych użyto biblioteki matplotlib.

III. Testy

Do przetestowania algorytmu użyty został moduł unittest. Same testy zostały sprawdzone dla trzech zbiorów wejściowych:

1. [Point(0,0), Point(0,2), Point(0,4), Point(2,0), Point(2,2), Point(2,4), Point(4,0), Point(4,2), Point(4,4)]
2. [Point(0,0), Point(2,0), Point(4,0), Point(6,0)]
3. [Point(2,0), Point(1,0), Point(3,1), Point(2,2), Point(4,3), Point(4,2), Point(3,2), Point(1,4)]

Dla wszystkich zestawów testy przeszły pomyślnie, a czas ich wykonania był mniejszy niż 0,000s

IV. Bibliografia

- http://www.algorytm.org/geometria-obliczeniowa/znajdowanie-wypuklej-otoczki-algorytm-grahama.html
- http://wazniak.mimuw.edu.pl/index.php?title=Zaawansowane_algorytmy_i_struktury_danych/Wyk%C5%82ad_11
- Wprowadzenie do algorytmów - Ron Rivest i Thomas H. Cormen