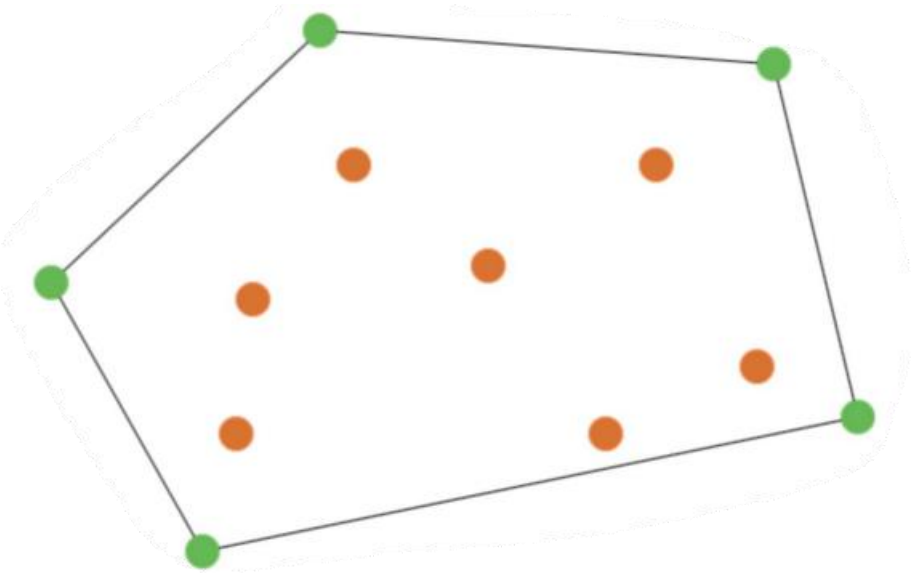
OUTLINE

PROGRAM DO ZNAJDOWANIA OTOCZKI WYPUKŁEJ ZBIORU PUNKTÓW W PREZESTRZENI 2-WYMIAROWEJ

Bartosz Polak | bartosz_polak@outlook.com | 29.10.2023



Opis programu

Stworzony program ma na celu znalezienie spośród zbioru punktów na przestrzeni dwu wymiarowej, taki podzbiór który otacza wszystkie inne punkty i jednocześnie powstały wielokąt posiada największe pole powierzchni.

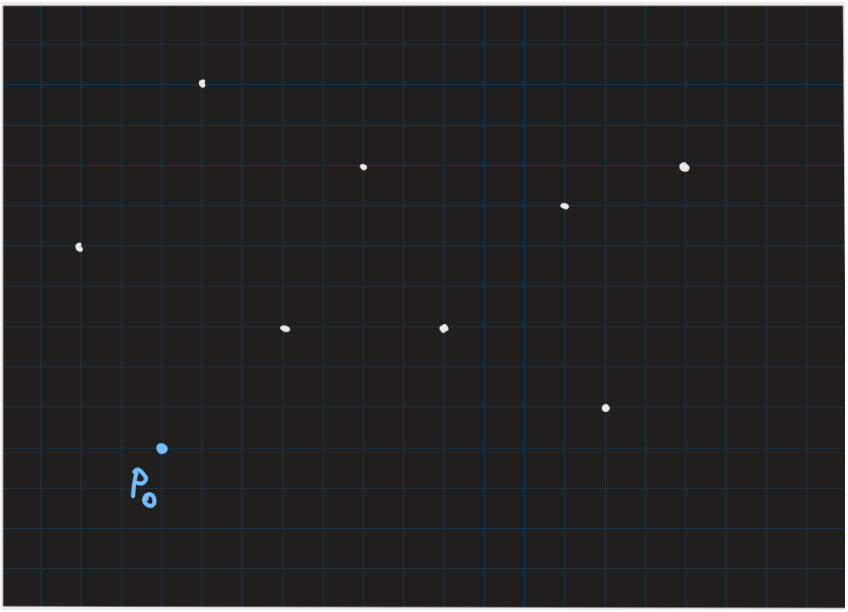
Program został oparty o algorytm *Chana,* jest to obecnie najlepiej zoptymalizowany algorytm do przeprowadzenia danego zadania. Algorytm ten składa się z dwóch innych algorytmów *Graham scan* i *Jarvis march*. Dzięki połączeniu tych dwóch metod uzyskuje się czasy działania **O(nlog(h))**, gdzie n to liczba punktów a h to liczba wierzchołków otoczki.

JARVIS MARCH

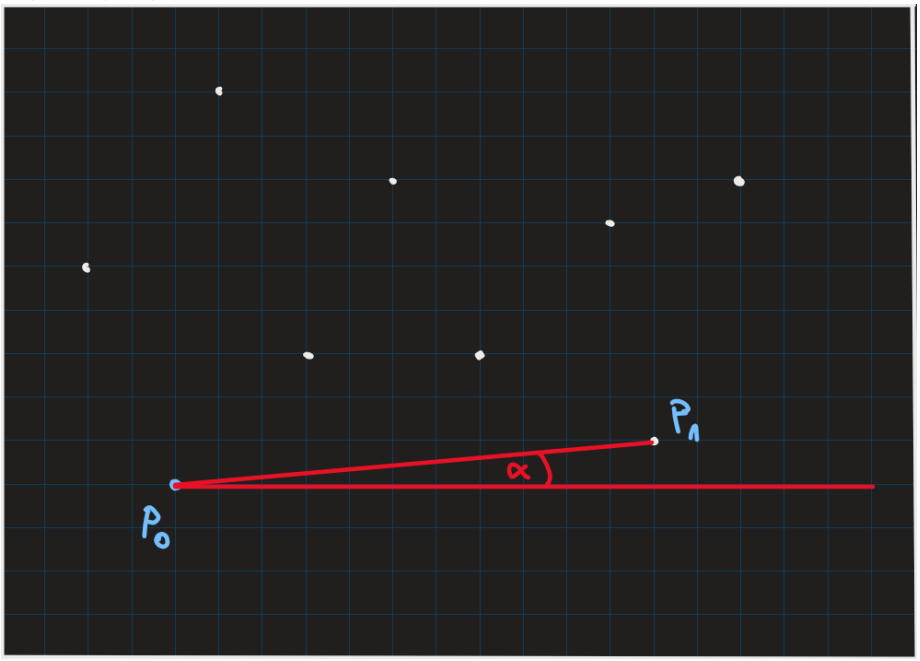
Jarvis march jest to algorytm do znajdowania otoczki wypukłej zbioru punktów z czasem działania **O(nh)**, więc w najgorszym przypadku(h = n) jego czas działania wynosi $\mathbf{O(n^2)}$.

ZASADA DZIAŁANIA

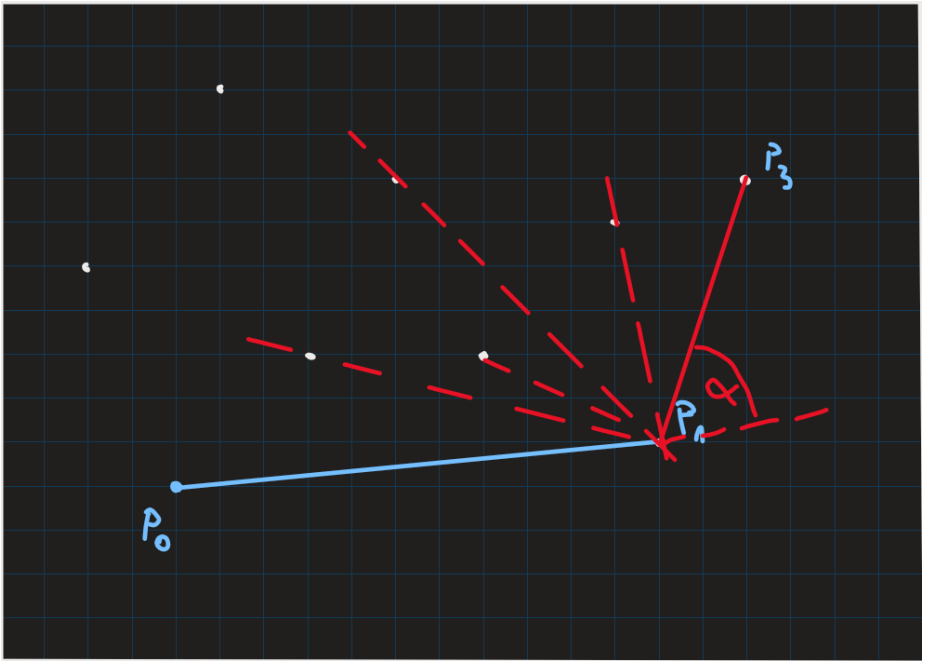
1. Znajdź punkt o najmniejszej współrzędnej y a w przypadku tych samych wartości ten o najmniejszej współrzędnej x.



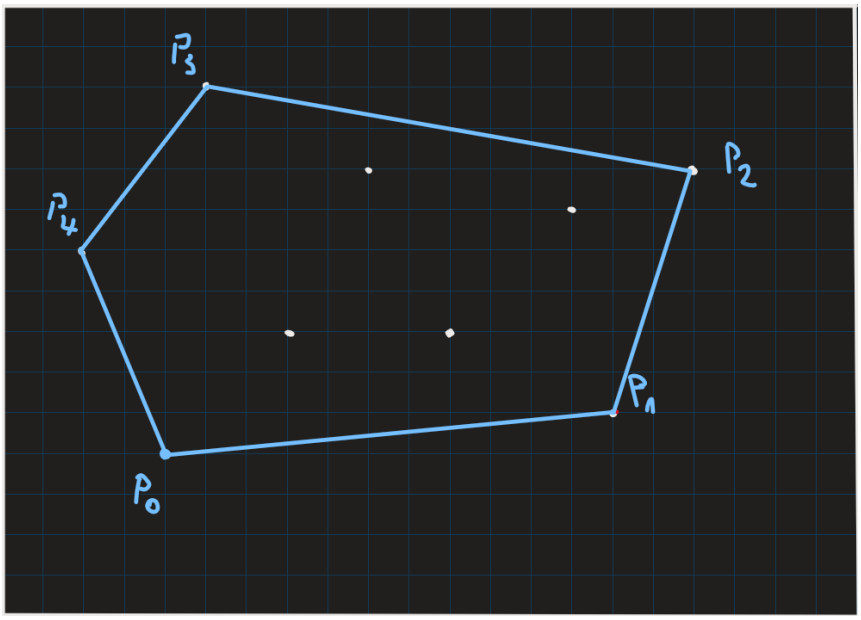
2. Oblicz kąt pomiędzy punktem po a każdym następnym i wybierz ten o najmniejszej wartości



3. Wykonuj powyższą operację aż do momentu powrotu do punktu początkowego



4. Zakończona praca algorytmu powinna przedstawić następujący wynik:

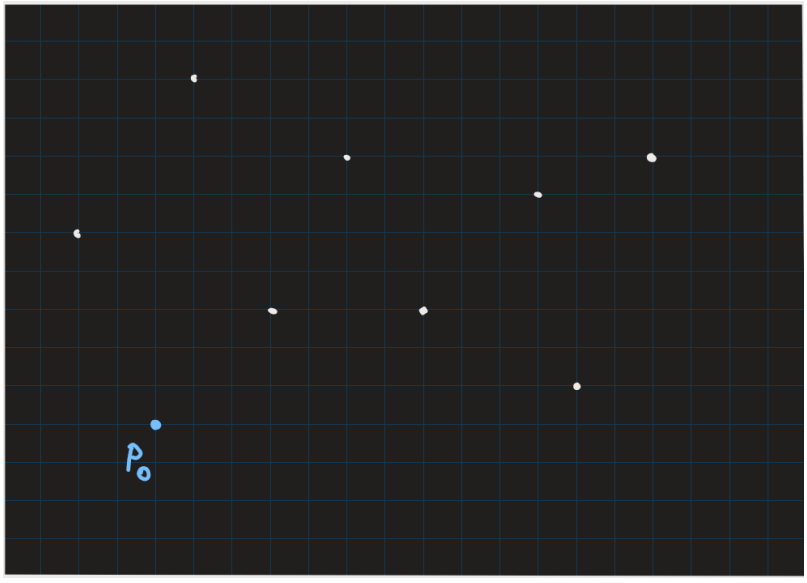


GRAHAM SCAN

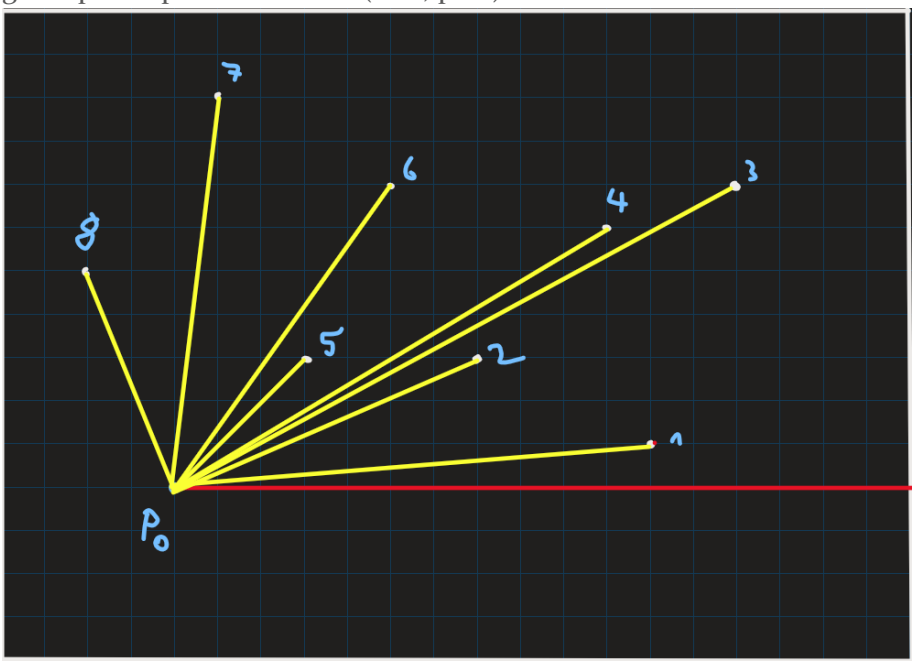
Graham scan jest algorytmem do znajdowania otoczki wypukłej o czasie złożoności **O(nlog(n))** co sprawie że jest lepszy od poprzedniego w przypadku gdy liczba punktów tworzących otoczkę jest znaczna.

ZASADA DZIAŁANIA

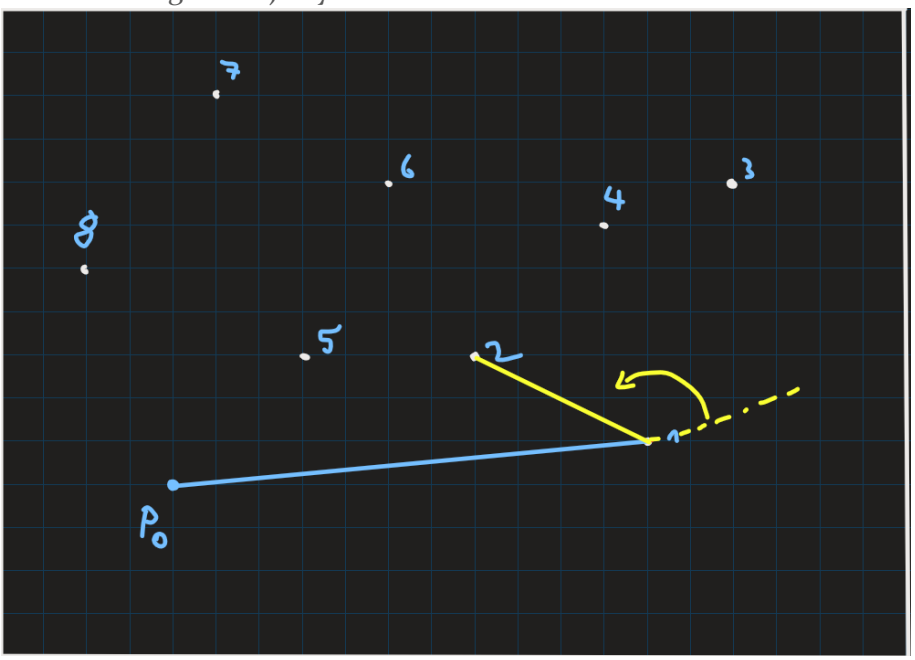
1. Znajdź punkt o najmniejszej współrzędnej y a w przypadku tych samych wartości ten o najmniejszej współrzędnej x.



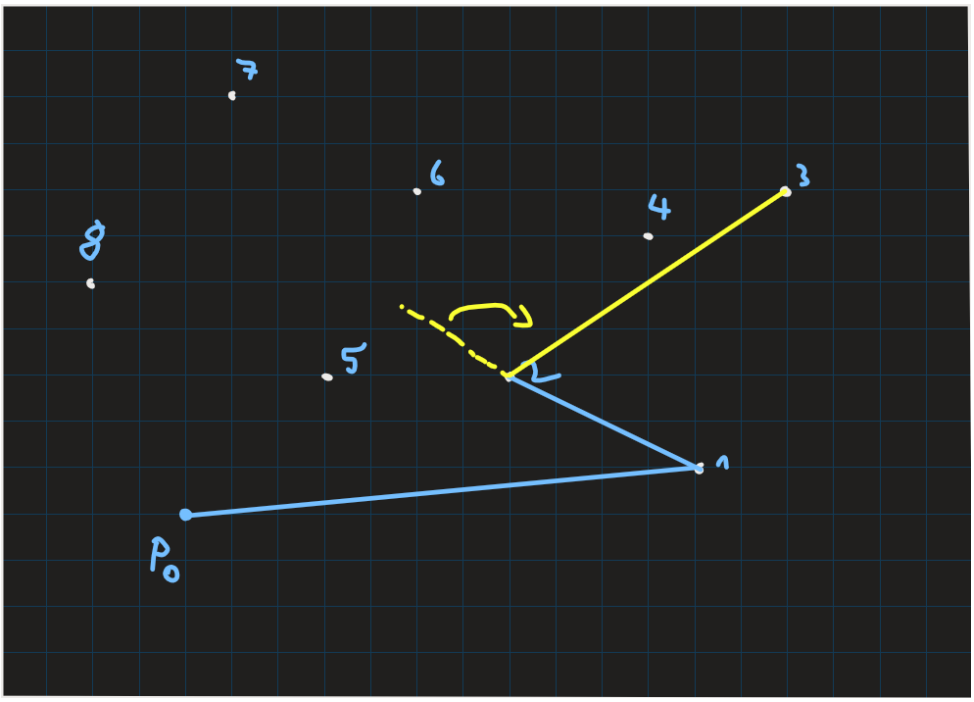
2. Posortuj wszystkie punkty względem kąta jaki tworzą pomiędzy **p0,p1,pn,** gdzie punkt p0 ma wartości (+inf, p0.x)



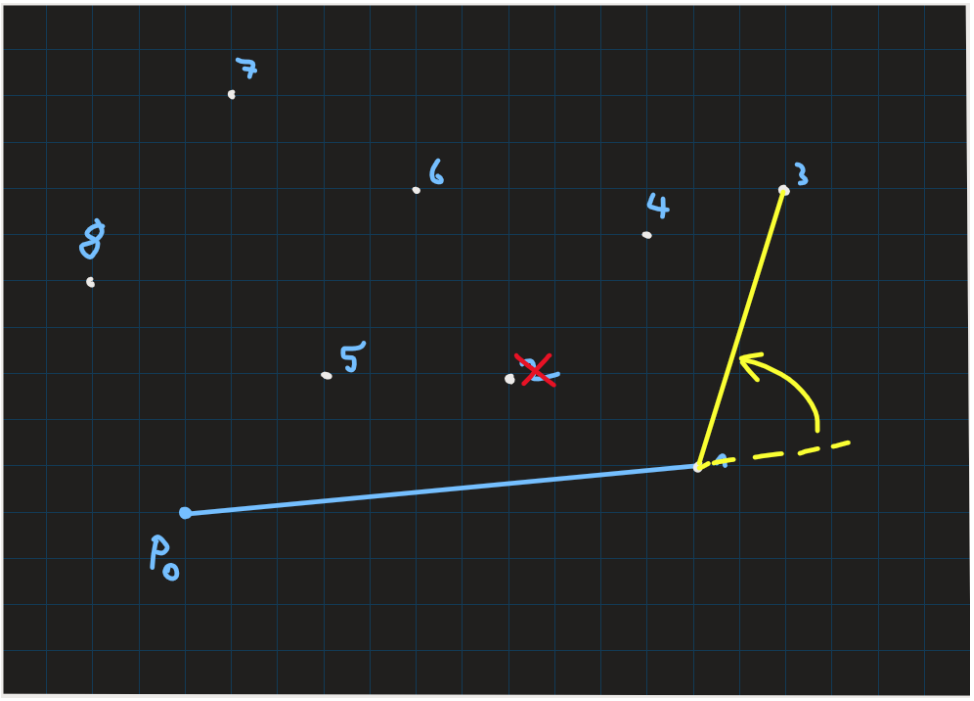
3. Wybierz punkt 1 i zacznij obliczać kąt który tworzą punkty p0, p1, pn, gdy jest on w *lewo* to go dodajemy



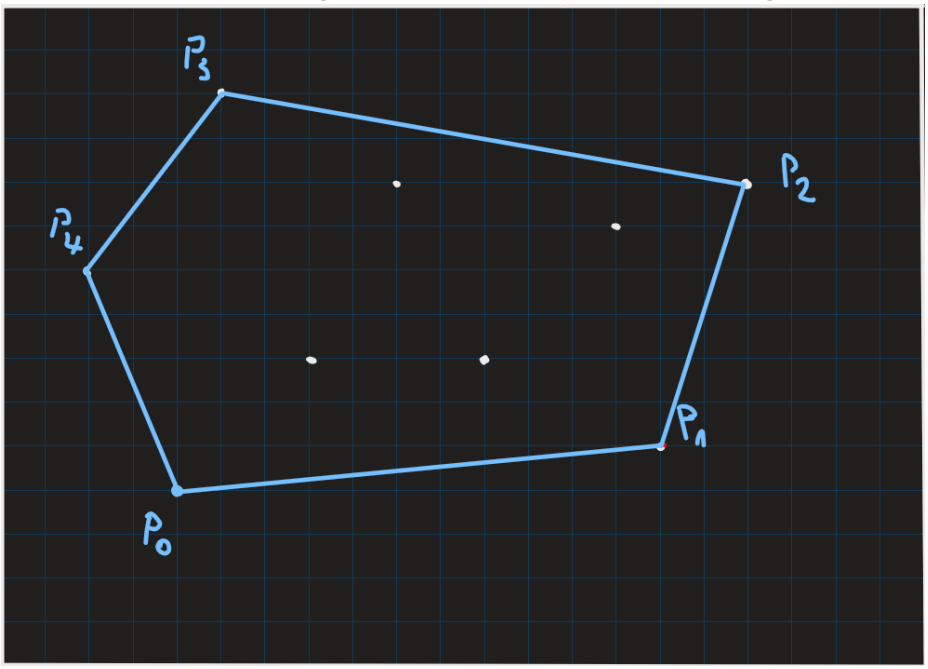
4. Natomiast gdy kolejny punkt daje kąt w *prawo* to zabieramy poprzedni punkt z listy



5. Powtarzamy proces



6. Po powrocie do pierwszego punktu otrzymujemy otoczkę tego zbioru punktów

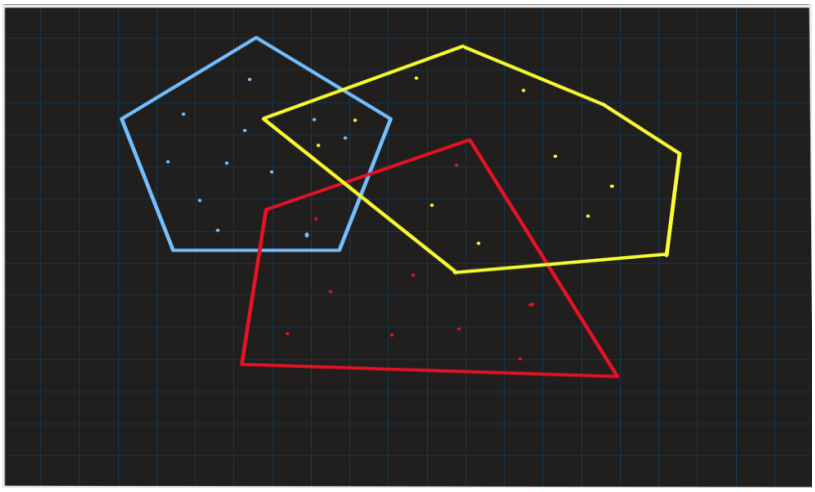


CHAN'S ALGORITHM

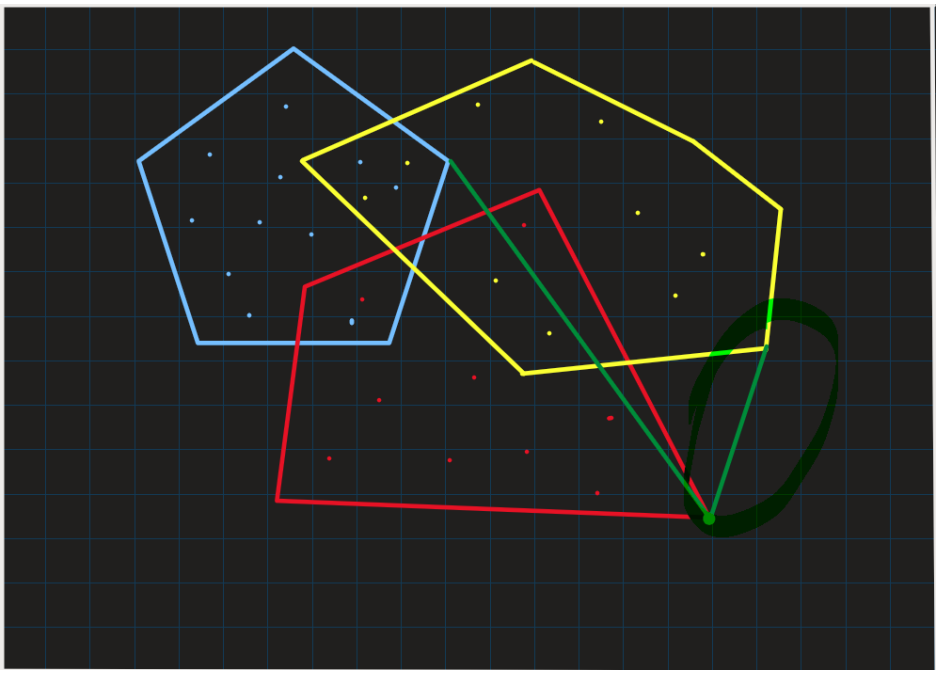
Algorytm Chana polega na podzieleniu zbioru punktów na mniejsze podzbiory, gdzie liczba podzbiorów powinna być równa liczbie wierzchołków, aby to osiągnąć i zachować złożoność obliczeniową **O(nlog(h))** wykonuje się ponowne iteracje gdzie za każdym razem liczba, która określa ilość punktów w każdym podzbiorze **m** jest powiększana zgodnie z zasadą: (t jest zwiększane o 1 po każdym przejściu algorytmu)

$$m = 2^{2^t}$$

Następnie dla każdego podzbioru jest przeprowadzany *Graham scan:*



Następnie przy wykorzystaniu *Jarvis march* jest wybierany kolejny punkt otoczki głównej poprzez wybranie z każdego podzbioru punktu o najmniejszym kącie a z nich najmniejszym w całości.



Warto dodać że punkty które tworzą poszczególne otoczki są już posortowane więc do znalezienia kolejnego punktu można zastosować wyszukiwanie binarne i skrócić czas działania algorytmu.

Dokumentacja Techniczna

Opis funkcji i kodu:

```
bool readFile(const std::string fileName, std::vector<Point>& points);

Funkcja służy odczytu zapisanych punktów z pliku .txt, zapisuje je jako vector
punktów, w przypadku niepowodzenia zwraca false, przeciwnie true.

bool calculateConvexHull(std::vector<Point>& points, int m, int n,
std::vector<Point>& outLinePoints);

Funkcja służy do wyznaczenia punktów otoczki wypukłej, w przypadku
niepowodzenia zwraca false, przeciwnie true i zapisuje punkty otoczki do
vectora outLinePoints.
Zmienne:
m – to liczba punktów w podzbiorze
n – to liczba wszystkich punktów
points – to vector wszystkich punktów
```

```
void GrahamScan(std::vector<Point>&points);
```

Funkcja służy do wytyczenia otoczki wypukłej z zastosowaniem *Graham scan*. Vector wejściowy z punktami na wyjściu zostaje pomniejszony punktów jedynie tworzących otoczkę.

```
Point jarvisMarch(std::vector<Point>& points, Point p0, Point p);
```

Funkcja służy do wytyczenia otoczki wypukłej z zastosowaniem *Jarvis march*, na wyjściu daje punkt który tworzy najmniejszy kąt [**p0, p, pn**] gdzie **pn** to punkt ze zbioru punktów, zawartych w vektorze **points**.

```
float calculateAngle(Point a, Point b);
```

Funkcja zwraca kąt jaki tworzy funkja liniowa utworzona z punktów **a** i **b** z osią **X** w radianach.

```
float calculateAngle(Point a, Point b, Point c);
```

Funckja zwraca wartość kąta [**a, b, c**] w stopniach.

```
bool compare_angle(Point a, Point b);
```

Funkcja składowa sortowania porównuje dwa kąty i zwraca true gdy kąt a < b.

```
float distance(Point a, Point b);
```

Funkcja zwraca dystans pomiędzy dwoma punktami w przestrzeni 2-wymiarowej.

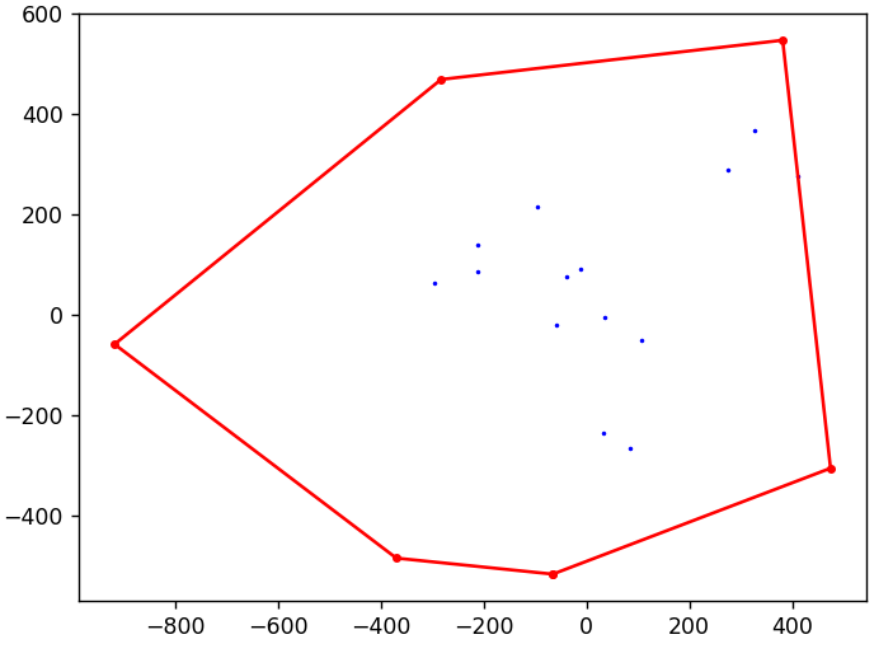
```
float angle_dir(Point a, Point b, Point c);
```

funkcja zwraca liczbę większą od 0 dla kąta[**a, b, c**] gdy jest to kąt w *lewo* i liczbę mniejszą od zera gdy kąt jest w *prawo* oraz 0 gdy punkty są w jednej linii. Jest do tego wykorzystany mnożenie wektorowe.

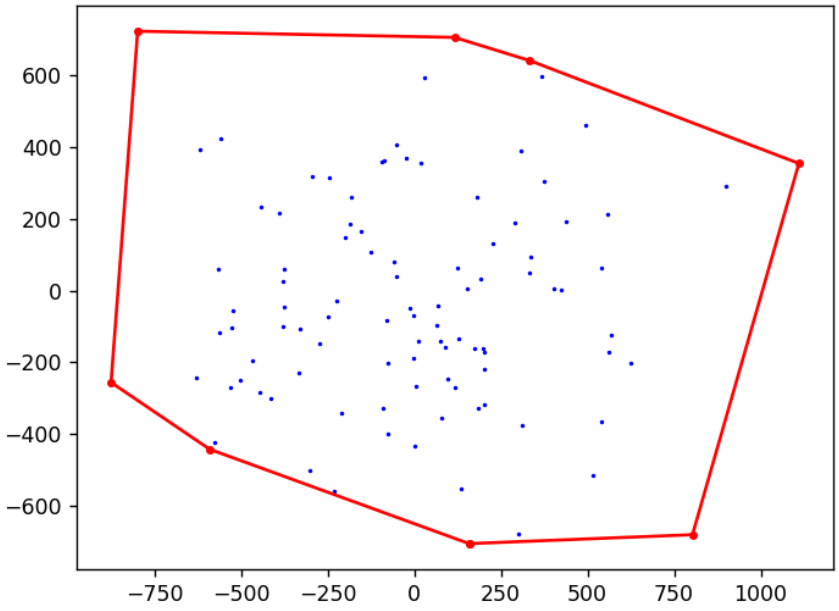
Fukcje Graham scan i Jarvis march działają zgodnie z opisem i wyjaśnienem dodanym wcześniej.

Wizualizacja

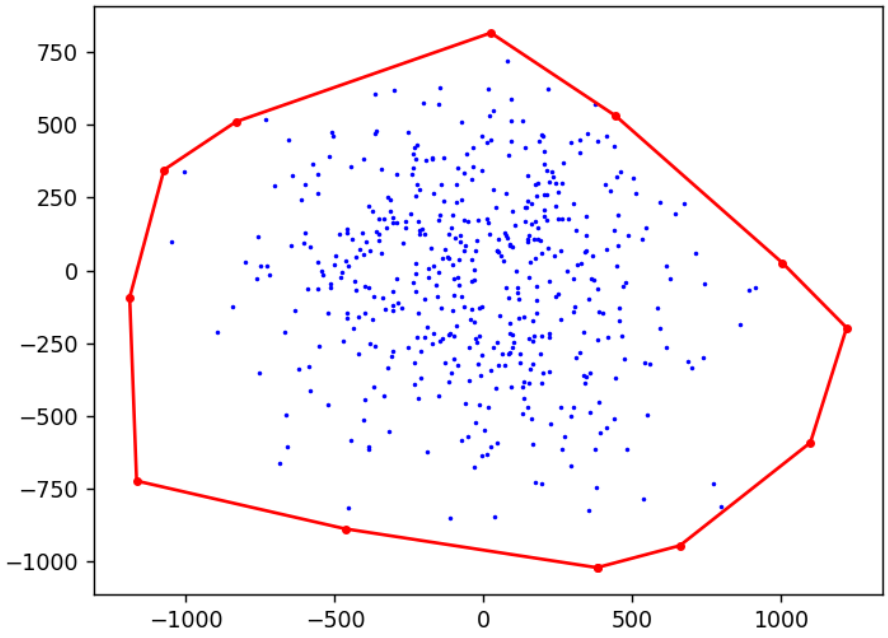
Graficzna wizualizacja kilku przypadków wyników programu dla różnych ilości punktów: (wizualizacja została utworzona przy pomocy skryptu z *visualization*)



20 Punktów



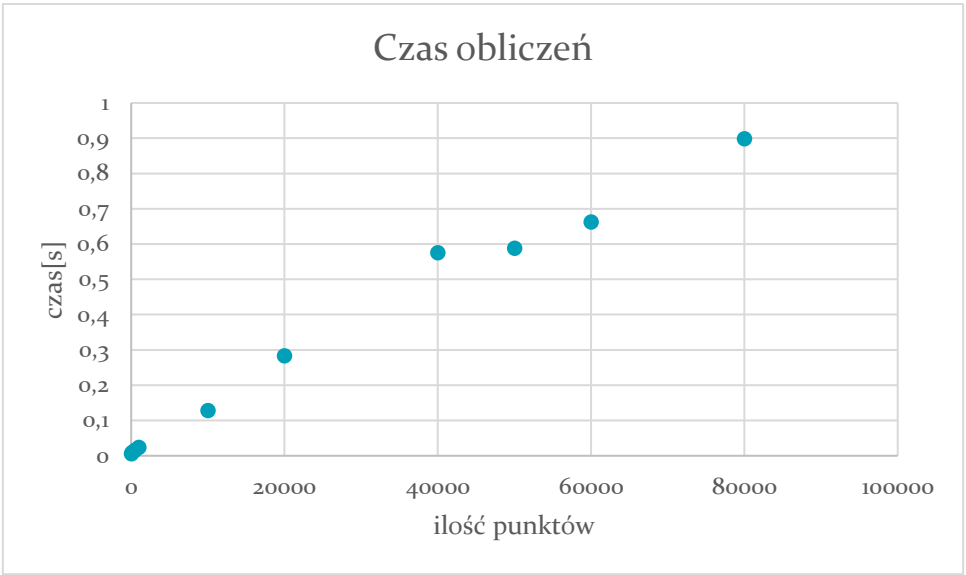
100 punktów



500 punktów

Wykresy czasu obliczeń dla różnych ilości punktów:

(Punkty są dobrane w sposób losowy, z zastosowaniem rozkładu Gausa)



BARTOSZ POLAK

junior developer, future engineer

bartosz_polak@outlook.com

+48 784 502 001

Github:

https://github.com/AitenAndGo

@_bartosz_polak_