UNIWERSYTET GDAŃSKI

WYDZIAŁ MATEMATYKI, FIZYKI I INFORMATYKI

Marcin Belicki

numer albumu: 273417

Kierunek studiów: Informatyka

OTOCZKI WYPUKŁE

Praca magisterska

wykonana

pod kierunkiem

dr inż. Arkadiusz Mirakowski

Gdańsk 2023

Spis treści

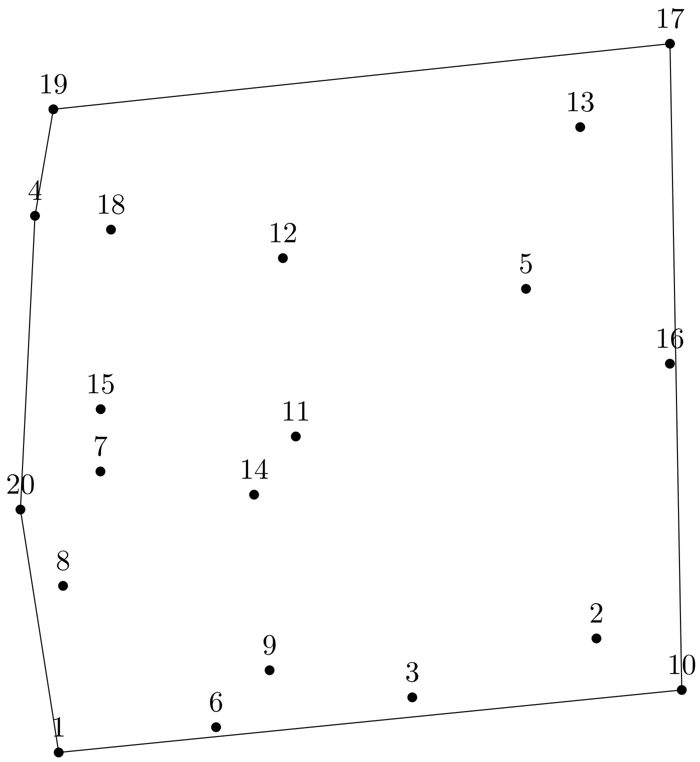
Wstęp

Celem niniejszej pracy jest przedstawienie definicji otoczki wypukłej oraz algorytmów wykorzystywanych do jej wyznaczania. Porównanie będzie miało charakter teoretyczny (Rozdział I) oraz praktyczny (Rozdział III), gdzie dogłębnie została omówiona implementacja algorytmów w języku Scala. Od strony teoretycznej celem będzie znajdywanie i uzasadnianie rozwiązań, które pozwolą w jak najmniej złożony obliczeniowo sposób wyznaczyć otoczkę wypukłą. Przez co strona praktyczna powinna w założeniu odzwierciedlać w jakiś sposób założenia strony teoretycznej i jej celem będzie też niejako potwierdzenie teoretycznych założeń.

Założeniem pracy jest także uzasadnienie istotności zagadnienia otoczek wypukłych we wielu dziedzinach mniej lub bardziej związanych z informatyką. Należą do nich między innymi: kartografia [1], detekcja istotnych obiektów [2]. Celem tej części będzie nie tylko wymienienie i opisanie tych pól nauki, ale również zbadanie jak wykorzystanie różnych algorytmów wpływa na wydajność konkretnych rozwiązań.

Rozdział I
Otoczka wypukła na płaszczyźnie

Otoczka wypukła zbioru punktów w swojej najbardziej podstawowej postaci jest wielokątem wypukłym obejmującym wszystkie punkty ze zbioru punktów leżących na płaszczyźnie w taki sposób, aby wielokąt ten miał jak najmniejsze pole.



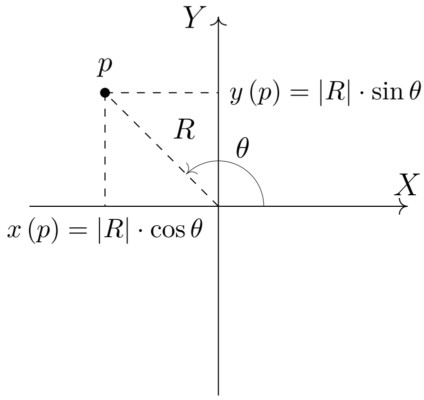
Rysunek 1: Otoczka wypukła na płaszczyźnie
Źródło: opracowanie własne

Dobrą reprezentacją otoczki wypukłej w świecie fizycznym może być grupa gwoździ przybita do płaskiej powierzchni i następnie opleciona ciasno sznurkiem. Gwoździe stykające się ze sznurkiem stanowić będą wierzchołki otoczki wypukłej tej grupy gwoździ.

1. Otoczka wypukła zbioru punktów

W celu wyznaczenia otoczki wypukłej dla najbardziej ogólnego przypadku — nieuporządkowanego zbioru punktów na płaszczyźnie, możemy wykorzystać dwa najbardziej popularne algorytmy algorytm Jarvisa oraz algorytm Grahama.

W wykorzystywanych algorytmach istotnym elementem jest sortowanie punktów względem wartości, które zostały przedstawione na rysunku 2. W zależności od tego, w jakiej postaci określone będą dane punkty, należy dokonać odpowiednich obliczeń.



Rysunek 2: Punkt na układzie współrzędnych
Źródło: opracowanie własne

Gdzie poszczególne symbole oznaczają:

p	–	rozpatrywany punkt
$x(p)$	–	odcięta punktu p
$y(p)$	–	rzędna punktu p
R	–	długość wektora wodzącego punktu p
θ	–	kąt nachylenia wektora wodzącego punktu p do osi OX

Istotną komplikację z punktu widzenia obliczeń w algorytmach może stanowić wyznaczenie wartości kąta θ (ze względu na potrzebę wykorzystania funkcji trygonometrycznych), którego dokładną wartość można wyliczyć za pomocą wzoru 1.

$$\theta(p) = \begin{cases} \operatorname{arctg}\frac{y(p)}{x(p)} & \text{dla} \quad x(p) > 0; \\ \operatorname{arctg}\frac{y(p)}{x(p)} + \pi & \text{dla} \quad x(p) < 0 \quad \wedge \quad y(p) \geqslant 0; \\ \operatorname{arctg}\frac{y(p)}{x(p)} - \pi & \text{dla} \quad x(p) < 0 \quad \wedge \quad y(p) < 0; \\ \frac{\pi}{2} & \text{dla} \quad x(p) = 0 \quad \wedge \quad y(p) < 0; \\ -\frac{\pi}{2} & \text{dla} \quad x(p) = 0 \quad \wedge \quad y(p) > 0. \end{cases} \tag{1}$$

Należy jednak zauważyć, że do celów sortowania wystarczy zastosować funkcję $\alpha(p)$, taką, że dla dowolnej pary punktów p_1, p_2 spełnione będą warunki 2 oraz 3.

$$\alpha(p_1) < \alpha(p_2) \Leftrightarrow \theta(p_1) < \theta(p_2) \tag{2}$$

$$\alpha(p_1) = \alpha(p_2) \Leftrightarrow \theta(p_1) = \theta(p_2) \tag{3}$$

Przykładowa funkcja $\alpha(p)$ zachowująca spełniająca warunki 2 i 3 została przedstawiona za pomocą wzoru 4.

$$\alpha(p) = \begin{cases} \frac{y(p)}{d(p)} & \text{dla} \quad x(p) \geqslant 0 \quad \wedge \quad y(p) \geqslant 0; \\ 2 - \frac{y(p)}{d(p)} & \text{dla} \quad x(p) < 0 \quad \wedge \quad y(p) \geqslant 0; \\ 2 + \frac{|y(p)|}{d(p)} & \text{dla} \quad x(p) < 0 \quad \wedge \quad y(p) < 0; \\ 4 - \frac{|y(p)|}{d(p)} & \text{dla} \quad x(p) \geqslant 0 \quad \wedge \quad y(p) < 0. \end{cases} \tag{4}$$

Gdzie funkcja $d(p)$ określona jest zgodnie ze wzorem 5.

$$d(p) = |x(p)| + |y(p)| \tag{5}$$

W celu udowodnienia, że funkcja $\alpha(p)$ spełnia warunki 2 i 3, musimy wykazać, że funkcja $\gamma(\theta)$ (przedstawiona we wzorze 6), która stanowi przekształcenie funkcji $\alpha(p)$ takie, że jest w pełni zależna od wartości θ, a wartości $x(p(\theta))$ oraz $y(p(\theta))$ są określone zgodnie ze wzorem 7, jest rosnąca w każdym swoim przedziale.

$$\gamma(\theta) = \alpha(p(\theta)) \tag{6}$$

$$p(\theta) = (|R| \cdot \cos\theta, |R| \cdot \sin\theta) \tag{7}$$

Należy zauważyć, że poszczególne przypadki opisane we wzorze 4 dzielą układ współrzędnych na cztery ćwiartki. Zatem z rysunku 2 wynikają zależności 8, 9, 10 oraz 11.

$$x(p) \geqslant 0 \ \wedge \ y(p) \geqslant 0 \ \Leftrightarrow \ \theta \in \left\langle 0; \frac{\pi}{2} \right\rangle \tag{8}$$

$$x(p) < 0 \ \wedge \ y(p) \geqslant 0 \ \Leftrightarrow \ \theta \in \left(\frac{\pi}{2}; \pi \right\rangle \tag{9}$$

$$x(p) < 0 \ \wedge \ y(p) < 0 \ \Leftrightarrow \ \theta \in \left(\pi; \frac{3\pi}{2} \right) \tag{10}$$

$$x(p) \geqslant 0 \ \wedge \ y(p) < 0 \ \Leftrightarrow \ \theta \in \left\langle \frac{3\pi}{2}; 2\pi \right) \tag{11}$$

Zatem w celu udowodnienia spełnienia warunków 2 i 3 przez funkcję $\alpha(p)$ (4) należy udowodnić, że funkcja $\gamma(\theta)$ (6) jest rosnąca (jej pochodna jest większa od zera) we wszystkich przedziałach opisanych w zależnościach 8, 9, 10 oraz 11.

Obliczenia dla tych czterech przedziałów przedstawiają się w następujący sposób. Dla każdego z przedziałów możliwe jest uproszczenie funkcji $d(p)$, tak aby pozbyć się modułu, przez co różniczkowanie funkcji $\gamma(\theta)$ staje się łatwiejsze.

1° $x(p) \geqslant 0 \ \wedge \ y(p) \geqslant 0 \ \Leftrightarrow \ \theta \in \left\langle 0; \frac{\pi}{2} \right\rangle$

$$\alpha(p) = \frac{y(p)}{d(p)} = \frac{y(p)}{|x(p)| + |y(p)|} = \frac{y(p)}{x(p) + y(p)}$$

$$\gamma(\theta) = \alpha(p(\theta)) = \frac{|R| \cdot \sin\theta}{|R| \cdot \cos\theta + |R| \cdot \sin\theta} = \frac{\sin\theta}{\sin\theta + \cos\theta}$$

$$\begin{aligned}
\frac{d}{d\theta}\gamma(\theta) &= \frac{d}{d\theta}\left(\frac{\sin\theta}{\sin\theta + \cos\theta}\right) \\
&= \frac{\frac{d}{d\theta}(\sin\theta) \cdot (\sin\theta + \cos\theta) - \sin\theta \cdot \frac{d}{d\theta}(\sin\theta + \cos\theta)}{(\sin\theta + \cos\theta)^2} \\
&= \frac{\cos\theta \cdot (\sin\theta + \cos\theta) - \sin\theta \cdot (\cos\theta - \sin\theta)}{\sin^2\theta + 2\sin\theta\cos\theta + \cos^2\theta} \\
&= \frac{\cos\theta\sin\theta + \cos^2\theta - \sin\theta\cos\theta + \sin^2\theta}{1 + 2\sin\theta\cos\theta} = \frac{1}{1 + 2\sin\theta\cos\theta} \\
&= \frac{1}{1 + \sin 2\theta}
\end{aligned}$$

$$\begin{aligned}
\frac{d}{d\theta}\gamma(\theta) &> 0 \\
\frac{1}{1 + \sin 2\theta} &> 0 \\
1 + \sin 2\theta &> 0 \\
\sin 2\theta &> -1 \\
\sin 2\theta &\neq -1 \\
2\theta &\neq \frac{3}{2}\pi + 2k\pi \\
\theta &\neq \frac{3}{4}\pi + k\pi; \ k \in \mathbb{Z}
\end{aligned}$$

$$\left.\begin{array}{l} k = 0 \ \Rightarrow \ \theta \neq \frac{3}{4}\pi > \frac{\pi}{2} \\ k = -1 \ \Rightarrow \ \theta \neq -\frac{1}{4}\pi < 0 \end{array}\right\} \Rightarrow \frac{d}{d\theta}\gamma(\theta) > 0 \text{ dla } \theta \in \left\langle 0; \frac{\pi}{2} \right\rangle$$

$$\left.\begin{array}{rcl} \gamma(0) & = & \frac{\sin 0}{\sin 0 + \cos 0} = \frac{0}{0+1} = 0 \\ \gamma\left(\frac{\pi}{2}\right) & = & \frac{\sin\frac{\pi}{2}}{\sin\frac{\pi}{2} + \cos\frac{\pi}{2}} = \frac{1}{1+0} = 1 \end{array}\right\} \Rightarrow \gamma(\theta) \in \langle 0; 1 \rangle \text{ dla } \theta \in \left\langle 0; \frac{\pi}{2} \right\rangle$$

$2^\circ\;\; x(p) < 0 \;\wedge\; y(p) \geqslant 0 \;\Leftrightarrow\; \theta \in \left(\frac{\pi}{2};\pi\right\rangle$

$$\begin{aligned}
\alpha(p) &= 2 - \frac{y(p)}{d(p)} = 2 - \frac{y(p)}{|x(p)| + |y(p)|} = 2 - \frac{y(p)}{-x(p) + y(p)} \\
&= 2 - \frac{y(p)}{y(p) - x(p)} \\
\gamma(\theta) &= \alpha(p(\theta)) = 2 - \frac{|R| \cdot \sin\theta}{|R| \cdot \sin\theta - |R| \cdot \cos\theta} = 2 - \frac{\sin\theta}{\sin\theta - \cos\theta} \\
\frac{d}{d\theta}\gamma(\theta) &= \frac{d}{d\theta}\left(2 - \frac{\sin\theta}{\sin\theta - \cos\theta}\right) \\
&= -\frac{\frac{d}{d\theta}(\sin\theta) \cdot (\sin\theta - \cos\theta) - \sin\theta \cdot \frac{d}{d\theta}(\sin\theta - \cos\theta)}{(\sin\theta - \cos\theta)^2} \\
&= -\frac{\cos\theta \cdot (\sin\theta - \cos\theta) - \sin\theta \cdot (\cos\theta + \sin\theta)}{\sin^2\theta - 2\sin\theta\cos\theta + \cos^2\theta} \\
&= -\frac{\cos\theta\sin\theta - \cos^2\theta - \sin\theta\cos\theta - \sin^2\theta}{1 - 2\sin\theta\cos\theta} \\
&= -\frac{-\cos^2\theta - \sin^2\theta}{1 - 2\sin\theta\cos\theta} = \frac{\sin^2\theta + \cos^2\theta}{1 - \sin 2\theta} = \frac{1}{1 - \sin 2\theta}
\end{aligned}$$

$$\frac{d}{d\theta}\gamma(\theta) > 0$$

$$\frac{1}{1 - \sin 2\theta} > 0$$

$$1 - \sin 2\theta > 0$$

$$-\sin 2\theta > -1$$

$$\sin 2\theta < 1$$

$$\sin 2\theta \neq 1$$

$$2\theta \neq \frac{\pi}{2} + 2k\pi$$

$$\theta \neq \frac{\pi}{4} + k\pi;\;\; k \in \mathbb{Z}$$

$$\left.\begin{array}{lll} k = 0 & \Rightarrow & \theta \neq \frac{\pi}{4} < \frac{\pi}{2} \\ k = 1 & \Rightarrow & \theta \neq \frac{5}{4}\pi > \pi \end{array}\right\} \Rightarrow \frac{d}{d\theta}\gamma(\theta) > 0 \text{ dla } \theta \in \left(\frac{\pi}{2};\pi\right\rangle$$

$$\left.\begin{array}{rcl} \gamma\left(\frac{\pi}{2}\right) & = & 2 - \frac{\sin\frac{\pi}{2}}{\sin\frac{\pi}{2} - \cos\frac{\pi}{2}} = 2 - \frac{1}{1-0} = 1 \\ \gamma(\pi) & = & 2 - \frac{\sin\pi}{\sin\pi - \cos\pi} = 2 - \frac{0}{0-(-1)} = 2 \end{array}\right\} \Rightarrow \gamma(\theta) \in (1;2\rangle \;\text{ dla } \theta \in \left(\frac{\pi}{2};\pi\right\rangle$$

3° $x(p) < 0 \ \wedge \ y(p) < 0 \ \Leftrightarrow \ \theta \in \left(\pi; \frac{3\pi}{2}\right)$

$$\begin{aligned}
\alpha(p) &= 2 + \frac{|y(p)|}{d(p)} = 2 + \frac{-y(p)}{|x(p)| + |y(p)|} = 2 - \frac{y(p)}{-x(p) - y(p)} \\
&= 2 + \frac{y(p)}{x(p) + y(p)} \\
\gamma(\theta) &= \alpha(p(\theta)) = 2 + \frac{|R| \cdot \sin\theta}{|R| \cdot \cos\theta + |R| \cdot \sin\theta} = 2 + \frac{\sin\theta}{\sin\theta + \cos\theta} \\
\frac{d}{d\theta}\gamma(\theta) &= \frac{d}{d\theta}\left(2 + \frac{\sin\theta}{\sin\theta + \cos\theta}\right) \\
&= \frac{\frac{d}{d\theta}(\sin\theta) \cdot (\sin\theta + \cos\theta) - \sin\theta \cdot \frac{d}{d\theta}(\sin\theta + \cos\theta)}{(\sin\theta + \cos\theta)^2} \\
&= \frac{\cos\theta \cdot (\sin\theta + \cos\theta) - \sin\theta \cdot (\cos\theta - \sin\theta)}{\sin^2\theta + 2\sin\theta\cos\theta + \cos^2\theta} \\
&= \frac{\cos\theta\sin\theta + \cos^2\theta - \sin\theta\cos\theta + \sin^2\theta}{1 + 2\sin\theta\cos\theta} \\
&= \frac{\cos^2\theta + \sin^2\theta}{1 + 2\sin\theta\cos\theta} = \frac{1}{1 + \sin 2\theta}
\end{aligned}$$

$$\begin{aligned}
\frac{d}{d\theta}\gamma(\theta) &> 0 \\
\frac{1}{1 + \sin 2\theta} &> 0 \\
1 + \sin 2\theta &> 0 \\
\sin 2\theta &> -1 \\
\sin 2\theta &\neq -1 \\
2\theta &\neq \frac{3}{2}\pi + 2k\pi \\
\theta &\neq \frac{3}{4}\pi + k\pi; \ k \in \mathbb{Z}
\end{aligned}$$

$$\left.\begin{array}{lcl}
k = 0 & \Rightarrow & \theta \neq \frac{3}{4}\pi < \pi \\
k = 1 & \Rightarrow & \theta \neq \frac{7}{4}\pi > \frac{3\pi}{2}
\end{array}\right\} \Rightarrow \frac{d}{d\theta}\gamma(\theta) > 0 \text{ dla } \theta \in \left(\pi; \frac{3\pi}{2}\right)$$

$$\left.\begin{array}{rcl}
\gamma(\pi) & = & 2 + \frac{\sin\pi}{\sin\pi + \cos\pi} = 2 - \frac{0}{0-1} = 2 \\
\gamma\left(\frac{3\pi}{2}\right) & = & 2 + \frac{\sin\frac{3\pi}{2}}{\sin\frac{3\pi}{2} + \cos\frac{3\pi}{2}} \\
& = & 2 + \frac{-1}{-1+0} = 2 + 1 = 3
\end{array}\right\} \Rightarrow \gamma(\theta) \in (2; 3) \ \text{ dla } \theta \in \left(\pi; \frac{3\pi}{2}\right)$$

$4^\circ \ \ x(p) \geqslant 0 \ \wedge \ y(p) < 0 \ \Leftrightarrow \ \theta \in \left\langle \tfrac{3\pi}{2}; 2\pi \right)$

$$\alpha(p) = 4 - \frac{|y(p)|}{d(p)} = 4 - \frac{-y(p)}{|x(p)| + |y(p)|} = 4 + \frac{y(p)}{x(p) - y(p)}$$

$$\gamma(\theta) = \alpha(p(\theta)) = 4 + \frac{|R| \cdot \sin\theta}{|R| \cdot \cos\theta - |R| \cdot \sin\theta} = 4 + \frac{\sin\theta}{\cos\theta - \sin\theta}$$

$$\begin{aligned}
\frac{d}{d\theta}\gamma(\theta) &= \frac{d}{d\theta}\left(4 + \frac{\sin\theta}{\cos\theta - \sin\theta}\right) \\
&= \frac{\frac{d}{d\theta}(\sin\theta) \cdot (\cos\theta - \sin\theta) - \sin\theta \cdot \frac{d}{d\theta}(\cos\theta - \sin\theta)}{(\cos\theta - \sin\theta)^2} \\
&= \frac{\cos\theta \cdot (\cos\theta - \sin\theta) - \sin\theta \cdot (-\sin\theta - \cos\theta)}{\cos^2\theta - 2\cos\theta\cos\theta + \sin^2\theta} \\
&= \frac{\cos^2\theta - \cos\theta\sin\theta + \sin^2\theta + \sin\theta\cos\theta}{1 - 2\sin\theta\cos\theta} \\
&= \frac{\cos^2\theta + \sin^2\theta}{1 - 2\sin\theta\cos\theta} = \frac{1}{1 - \sin 2\theta}
\end{aligned}$$

$$\begin{aligned}
\frac{d}{d\theta}\gamma(\theta) &> 0 \\
\frac{1}{1 - \sin 2\theta} &> 0 \\
1 - \sin 2\theta &> 0 \\
-\sin 2\theta &> -1 \\
\sin 2\theta &< 1 \\
\sin 2\theta &\neq 1 \\
2\theta &\neq \frac{\pi}{2} + 2k\pi \\
\theta &\neq \frac{\pi}{4} + k\pi; \ k \in \mathbb{Z}
\end{aligned}$$

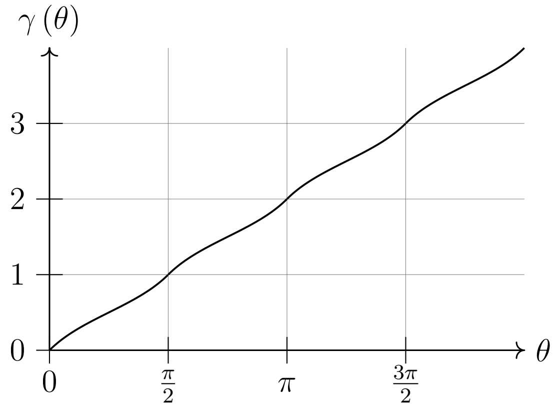
$$\left.\begin{aligned}
k = 1 \ &\Rightarrow \ \theta \neq \tfrac{5}{4}\pi > \tfrac{3\pi}{2} \\
k = 2 \ &\Rightarrow \ \theta \neq \tfrac{9}{4}\pi > 2\pi
\end{aligned}\right\} \Rightarrow \frac{d}{d\theta}\gamma(\theta) > 0 \text{ dla } \theta \in \left\langle \frac{3\pi}{2}; 2\pi \right)$$

$$\left.\begin{aligned}
\gamma\left(\tfrac{3\pi}{2}\right) &= 4 + \tfrac{\sin\frac{3\pi}{2}}{\cos\frac{3\pi}{2} - \sin\frac{3\pi}{2}} \\
&= 4 + \tfrac{-1}{0-(-1)} = 4 - 1 = 3 \\
\gamma(2\pi) &= 4 + \tfrac{\sin 2\pi}{\cos 2\pi - \sin 2\pi} \\
&= 4 + \tfrac{0}{1-0} = 4
\end{aligned}\right\} \Rightarrow \gamma(\theta) \in \langle 3; 4) \text{ dla } \theta \in \left\langle \frac{3\pi}{2}; 2\pi \right)$$

Z powyższych obliczeń wynika, że we wszystkich przedziałach z zależności 8, 9, 10 oraz 11 funkcja $\gamma(\theta)$ ma dodatnią pochodną, a tym samym jest rosnąca we wszystkich przedziałach, oraz każda z wartości osiąganych w przedziale jest większa od każdej wartości z poprzedniego przedziału, a tym samym funkcja $\alpha(p)$ spełnia warunki 2 i 3.

Postać funkcji $\gamma(\theta)$ może być opisana wzorem 12 a wykres jej przebiegu, przedstawiony na rysunku 3 wyraźnie potwierdza jej rosnącą monotoniczność we wszystkich przedziałach.

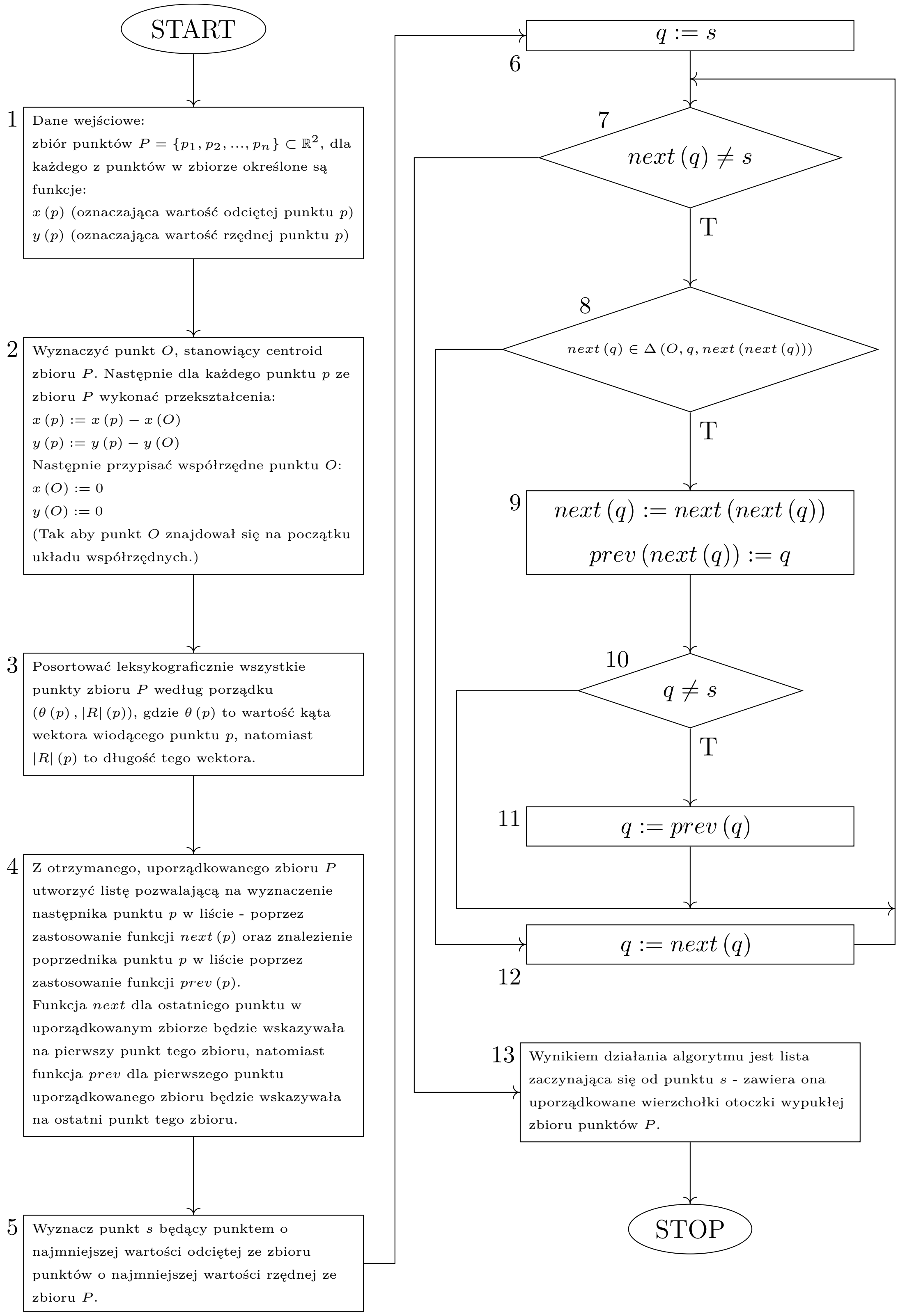
$$\gamma(\theta) = \begin{cases} \frac{\sin\theta}{\sin\theta+\cos\theta} & \text{dla} \quad \theta \in \left\langle 0; \frac{\pi}{2} \right\rangle \\ 2 - \frac{\sin\theta}{\sin\theta-\cos\theta} & \text{dla} \quad \theta \in \left(\frac{\pi}{2}; \pi \right\rangle \\ 2 + \frac{\sin\theta}{\sin\theta+\cos\theta} & \text{dla} \quad \theta \in \left(\pi; \frac{3\pi}{2} \right) \\ 4 - \frac{\sin\theta}{\sin\theta+\cos\theta} & \text{dla} \quad \theta \in \left\langle \frac{3\pi}{2}; 2\pi \right) \end{cases} \tag{12}$$



Rysunek 3: Wykres funkcji $\gamma(\theta)$
Źródło: opracowanie własne

1.1 Algorytm Grahama

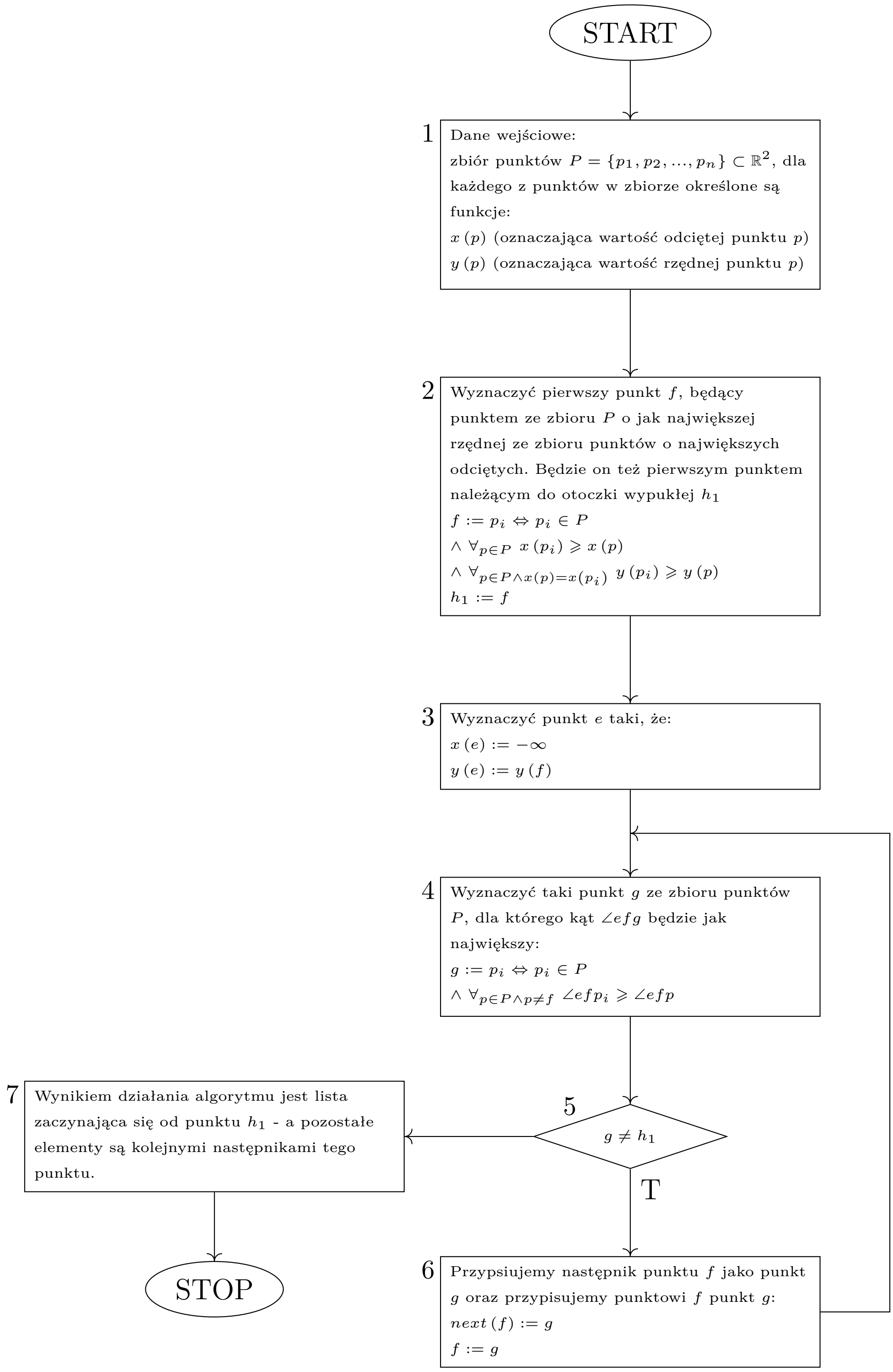
Jednym z ważniejszych algorytmów służących wyznaczaniu otoczki wypukłej zbioru punktów na płaszczyźnie jest algorytm Grahama.



Rysunek 4: Algorytm Grahama

Źródło: opracowanie własne

1.2 Algorytm Jarvisa



Rysunek 5: Algorytm Jarvisa

Źródło: opracowanie własne

2. Otoczka wypukła wielokąta prostego

[3]

3. Redukcja zbioru punktów do wielokąta prostego

Rozdział II
Zastosowania

Istotnym, jeśli nie najistotniejszym zagadnieniem dotyczącym otoczek wypukłych są ich zastosowania. Już od kilkudziesięciu lat problem ten temat znajduje swoje użycie w wielu dziedzinach kombinatoryki i informatyki. Dla przykładu problem sortowania elementów liczbowych listy można sprowadzić do problemu znalezienia otoczki wypukłej. W związku z tym rozwiązanie tego problemu może pomóc w rozwiązaniu innych problemów w bardziej symboliczny i graficzny sposób.

1. Generalizacja kartograficzna

W celu jak najbardziej informatywnego i niezłożonego przedstawienia danych geograficznych w sposób graficzny potrzebne jest zastosowanie algorytmu generalizującego informacje. Taka generalizacja w kartografii zaczęła być coraz bardziej potrzebna, kiedy w dobie cyfrowego przetwarzania danych mapy zaczęły być przechowywane cyfrowo. Przed tym generalizacją zajmowali się manualnie ludzie, którzy wyposażeni w ludzkie zmysły wykrywania wzorów i kształtów byli w stanie z łatwością narysować obrys zbioru drzew na płaszczyźnie i opisać go jako teren leśny. W przypadku przechowywania danych geograficznych w postaci cyfrowej zaistniała potrzeba wykonywania tej samej operacji w sposób zalgorytmizowany. [1]

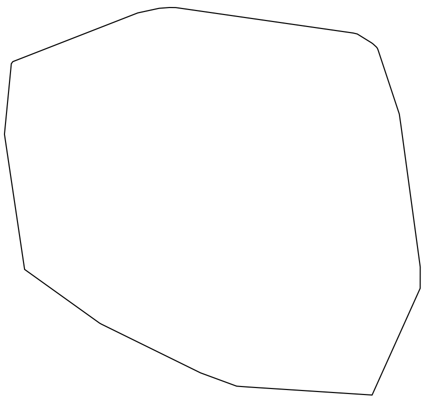
Do celu generalizacji można by wykorzystać otoczkę wypukłą w sposób bezpośredni [4], jednak taki obrys terenu może zawierać w sobie przestrzenie pozbawione obiektów, co byłoby niepomocne dla ludzkiej interpretacji przedstawianego mu obszaru.

Należy też zwrócić uwagę, że otoczka wypukła wielokąta prostego zawiera w sobie wszystkie punkty należące do wewnętrznej powierzchni tego wielokąta. [3] W związku z tym jeśli udowodni się, że dany punkt nie należy do wnętrza otoczki wypukłej będzie to implikowało, że punkt ten nie należy do wnętrza wielokąta prostego. Taka wiedza okazuje się być przydatna jeśli uwzględnimy dodatkowo fakt, że określanie czy punkt leży wewnątrz wielokąta wypukłego jest dużo łatwiejsze niż dla pozostałych wielokątów. [5]

Dla przykładu rozpatrując obrys granic Polski (Rysunek 6; źródło współrzędnych obrysu: [6]; współrzędne przedstawione zgodnie z projekcją EPSG:2180 [7] przy użyciu biblioteki GeoTools [8]) można zauważyć, że jego kształt jest bardzo skomplikowany. W istocie zawiera on 1336 krawędzi.

Rysunek 6: Obrys granic Polski
Źródło: opracowanie własne

Natomiast otoczka wypukła obrysu granic Polski zawiera tylko 28 krawędzi i została przedstawiona na rysunku 7.



Rysunek 7: Otoczka wypukła obrysu granic Polski
Źródło: opracowanie własne

W związku z tym, że złożoność obliczeniowa algorytmu sprawdzającego czy punkt należy do wnętrza danego wielokąta jest liniowa [9], możliwe jest dużo szybsze wykluczenie punktów znajdujących się poza otoczką.

2. Grafika komputerowa

Obiekty wykorzystywane w grafice komputerowej często mogą charakteryzować się skomplikowanymi kształtami. Im bardziej skomplikowany kształt, tym więcej mocy obliczeniowej potrzeba w celu wykonania danej operacji na tym kształcie. W niektórych przypadkach do uproszczenia graficznego danego obiektu używa się otoczki wypukłej jego kształtu. Ze względu na fakt, iż otoczka wypukła wielokąta zawsze będzie miała liczbę wierzchołków mniejszą lub równą liczbie wierzchołków samego wielokąta, powstała otoczka może posłużyć do wykonania mniejszej liczby obliczeń przy wykonywaniu operacji na danym obiekcie.

3. Detekcja obiektów

Proces analizowania obrazu przez ludzkie organy jest wyjątkowo złożony złożony [10]. W celu imitacji tego procesu w sposób algorytmiczny, zostały przeprowadzone liczne badania naukowe [2]. Dzięki temu zaproponowano wiele podejść rozwiązania tego problemu. Istotnym aspektem ze względów praktycznych tych pomysłów jest złożoność obliczeniowa ich implementacji.

Jednym z działań wykonywanych przez ludzkie zdolności poznawcze po zarejestrowaniu obrazu jest wybranie miejsca w obrazie o największym znaczeniu (z ludzkiego punktu widzenia) [11]. Niektóre z rozwiązań tego problemu w informatyce opierały się nie tylko na samych danych pochodzących z obrazu, ale również tworząc modele z uwzględnieniem danych statycznych z badań dotyczących ludzkich umiejętności poznawczych [12]. Jednak spora część z nich opiera się na przetwarzaniu samych danych pozyskanych z obrazów, bez potrzeby dostarczania danych statystycznych. Aby komputer był w stanie wykonać tego typu operację potrzebny jest szereg złożonych obliczeniowo operacji [2], [12].

W celu zmniejszenia kosztu obliczeniowego tego typu procesu można wykonać preprocesowanie danych wejściowych. Jednym z nich może być wykorzystanie algorytmu wyznaczającego otoczkę wypukłą tak zwanych punktów kluczowych [2]. W celu znalezienia lokalizacji punktów kluczowych można skorzystać z wyznaczania lokalnej energii piksela za pomocą wzoru (13) na wielopoziomową energię Harrisa

[2], [13].

$$\mathbf{E}_{WPH}(p) = \sum_{l=0}^{L} \mathbf{R}(l)(p) \tag{13}$$

Gdzie poszczególne symbole oznaczają:

p – rozpatrywany piksel obrazu

$\mathbf{R}(l)(p)$ – energia piksela p wyznaczona ze wzoru Harrisa (14) dla l - tego poziomu obrazu

L – liczba poziomów dla których ma być wyznaczona suma energii, przyjęte jako $L = 5$ [2]

W celu wyznaczenia wielopoziomowej Harrisa dla danego piksela p musimy znać też wzór (14) na energię piksela p dla pojedynczego poziomu l.

$$\mathbf{R}(l)(p) = |\mathbf{M}(l)(p)| + k \cdot \mathrm{tr}(\mathbf{M}(l)(p)) \tag{14}$$

Gdzie poszczególne symbole oznaczają:

k – współczynnik skalarny

$\mathbf{M}(l)(p)$ – macierz autokorelacji dla danego piksela p i poziomu obrazu l wyznaczana ze wzoru 15 [2], [13], [14]

$\mathrm{tr}(\mathbf{A})$ – ślad macierzy kwadratowej $\mathbf{A}$, czyli suma elementów należących do głównej przekątnej macierzy $\mathbf{A}$, obliczany według wzoru 16 [15], [16]

$$\mathbf{M}(l)(p) = \mathbf{G}_\sigma \cdot \begin{bmatrix} \mathbf{I}_x^2(l)(p) & \mathbf{I}_x(l)(p) \cdot \mathbf{I}_y(l)(p) \\ \mathbf{I}_y(l)(p) \cdot \mathbf{I}_x(l)(p) & \mathbf{I}_y^2(l)(p) \end{bmatrix} \tag{15}$$

Gdzie poszczególne symbole oznaczają:

$\mathbf{G}_\sigma$ – prostokątna macierz o liczbie kolumn 2 stanowiąca okno czasowe Gaussa o odchyleniu standardowym σ; poszczególne elementy tej macierzy można wyznaczyć według wzoru 17 [2], [13]

$\mathbf{I}_x(l)(p)$ – wartość pierwszej składowej gradientu l - tego poziomu obrazu $\mathbf{I}$ dla piksela p, obliczana według wzoru

$\mathbf{I}_y(l)(p)$ – wartość drugiej składowej gradientu l - tego poziomu obrazu $\mathbf{I}$ dla piksela p, obliczana według wzoru

$$\mathrm{tr}(\mathbf{A}) = \sum_{i=1}^{n} \mathbf{A}(i,i) \tag{16}$$

$$\mathbf{G}_\sigma (u, v) = \exp \frac{-\left(u^2 + v^2\right)}{2\sigma^2} \tag{17}$$

4. Wyznaczanie obwiedni sygnału

Sygnał to zapis informacji zmieniającej się w zależności od czasu. Taką informacją może być natężenie/napięcie prądu, natężenie pola elektromagnetycznego, dźwięk utworu muzycznego itd. W przypadku, jeśli zmienną informację da się reprezentować w sposób liczbowy, możliwe jest graficzne przedstawienie sygnału. Zapisanie informacji w ten sposób może być łatwiejsze do interpretacji przez człowieka. Jednak w przypadku, gdy mamy do czynienia ze skomplikowanym przebiegiem, na

Algorytm wyznaczający otoczkę wypukłą wielokąta prostego może posłużyć znajdowaniu obwiedni sygnału.

Rozdział III
Implementacja w języku Scala

1. Pojęcia ogólne

W tej sekcji przedstawione zostaną listingi zawierające kod w języku Scala definiujący pojęcia potrzebne do implementacji poszczególnych algorytmów wyznaczających otoczkę wypukłą.

Klasa `Point` (Listing 1: linia 4) posłuży jako podstawowa klasa reprezentująca punkt w przestrzeni dwuwymiarowej. W celu łatwiejszego manipulowania danymi. Zostały w jej ramach zaimplementowane 3 metody pomocnicze.

Odejmowanie (Listing 1: linia 7) - zaimplementowane jako odejmowanie od siebie odpowiadających sobie współrzędnych dwu punktów.

Dodawanie (Listing 1: linia 8) - dodawanie do siebie odpowiadających sobie współrzędnych dwu punktów.

Dzielenie przez skalar (Listing 1: linia 9) - zaimplementowane jako dzielenie obu współrzędnych przez wskazaną liczbę. Zaimplementowane dzięki użyciu domniemanemu argumentowi `numeric: Numeric[T]`. W języku Scala tego typu konstrukcja umożliwia korzystanie z metody `/` w tedy i tylko wtedy jeśli zaimportowany zostanie do aktualnego kontekstu obiekt typu `Numeric[T]`. Domyślnie typ ten jest zaimplementowany dla `T` będącym jednym z podstawowych typów liczbowych: `Float`, `Int`, `Double` etc. Więc w domyśle tego typu konstrukcja służy niejako „udowodnieniu”, że wskazywany argument jest w istocie liczbą.

```
1 package utilities.geometry
2
3 case class Point(
4                   x: Double,
5                   y: Double
6                 ) {
7   def -(that: Point): Point = Point(x - that.x, y - that.y)
8   def +(that: Point): Point = Point(x + that.x, y + that.y)
9   def /[T](number: T)(implicit numeric: Numeric[T]): Point = {
10    val double = numeric.toDouble(number)
```

```
    Point(x / double, y / double)
  }
}
```

Listing 1: Point.scala
Źródło: opracowanie własne

Kolejnym istotnym elementem wykorzystanym w implementacji algorytmów wyznaczających otoczkę wypukłą zbioru punktów na płaszczyźnie są metody wyliczające określone właściwości punktów.

Metoda `calculateCentroid(points: Points): Points` (Listing 2: linia 11) wyznacza centroid zbioru punktów `points`.

Metoda `distanceFromCenterSquared(p: Point): Double` (Listing 2: linia 17) wyznacza kwadrat odległości wskazanego punktu od środka układu współrzędnych.

Na podstawie wyżej wymienionej metody zostały zaimplementowane trzy metody związane z odległością między punktami.

Metoda `distanceSquared(a: Point, b: Point): Double` (Listing 2: linia 13) wyznacza kwadrat odległości pomiędzy dwoma punktami.

Metoda `distanceFromCenter(p: Point): Double` (Listing 2: linia 19) wyznacza odległość od środka układu współrzędnych.

Metoda `distance(a: Point, b: Point): Double` (Listing 2: linia 15) wyznacza odległość pomiędzy dwoma punktami.

W ramach obiektu `PointsUtils` (Listing 2: linia 5) zostały również zaimplementowane metody służące do wyznaczania wartości umożliwiających sortowanie punktów względem kąta występującego pomiędzy horyzontalną osią układu współrzędnych, a wektorem łączącym początek układu współrzędnych z punktem (kąt θ zaznaczony na rysunku 2).

Metoda `phase(p: Point): Double` (Listing 2: linia 21) wyznacza dokładną wartość szukanego kąta. Matematycznie ta funkcja została zapisana we wzorze 1.

Metoda `alpha(p: Point): Double` (Listing 2: linia 32) wyznacza wartość umożliwiającą porównanie kąta dla wskazanego punktu z innymi punktami. Matematycznie ta funkcja została zapisana we wzorze 4.

```
1 package utilities.geometry
2
3 import java.lang.Math._
```

```scala

object PointsUtils {

  type Points = List[Point]

  private lazy val HALF_PI = PI * .5

  def calculateCentroid(points: Points): Point = points.reduceLeft(
    _ + _) / points.length

  def distanceSquared(a: Point, b: Point): Double =
    distanceFromCenterSquared(a - b)

  def distance(a: Point, b: Point): Double = distanceFromCenter(a -
    b)

  def distanceFromCenterSquared(p: Point): Double = pow(p.x, 2) +
    pow(p.y, 2)

  def distanceFromCenter(p: Point): Double = sqrt(
    distanceFromCenterSquared(p))

  def phase(p: Point): Double = {
    import p.{x, y}
    (x.sign, y.sign) match {
      case (1, _) => atan(y / x)
      case (-1, 0 | 1) => atan(y / x) + PI
      case (-1, -1) => atan(y / x) - PI
      case (0, -1) => HALF_PI
      case (0, 1) => -HALF_PI
    }
  }

  def alpha(p: Point): Double = {
    import p.{x, y}
    val absX = abs(x)
    val absY = abs(y)
    val d = absX + absY

```

```
    (x.sign, y.sign) match {
      case (0 | 1, 0 | 1) => y / d
      case (-1, 0 | 1) => 2 - y / d
      case (-1, -1) => 2 + absY / d
      case (0 | 1, -1) => 4 - absY / d
    }
  }
}
```

Listing 2: PointsUtils.scala

Źródło: opracowanie własne

W celu zapewnienia odpowiedniego sortowania punktów został utworzony `trait OrientationOrdering`. Język Scala umożliwia wykorzystywanie zaimplementowanie własnego obiektu `Ordering[T]`, który będzie następnie używany w metodzie `sorted` dla kolekcji `C[T]`.

Trait `OrientationOrdering` (Listing 3: linia 6) ma zaimplementowaną metodę `compare(x: Point, y: Point)` (Listing 3: linia 7) w taki sposób, aby w pierwszej kolejności porównywane były wartości funkcji reprezentujących fazę tych punktów, a następnie porównywane wartości funkcji reprezentujących odległość od środka układu współrzędnych.

W ramach `OrientationOrdering` zostały zaimplementowane dwa obiekty.

Obiekt `Exact` (Listing 3: linia 19) służy do dokładnego porównania fazy punktów, wyznaczanej ze wzoru 1 oraz odległości od środka układu współrzędnych.

Obiekt `Indicator` (Listing 3: linia 24) służy do porównywania fazy punktów za pomocą funkcji wyznaczanej ze wzoru 4.

```
1  package utilities.geometry.ordering
2
3  import utilities.geometry.{Point, PointsUtils}
4  import Ordering.Double.TotalOrdering
5
6  trait OrientationOrdering extends Ordering[Point] {
7    final override def compare(x: Point, y: Point): Int = {
8      TotalOrdering.compare(phase(x), phase(y)) match {
9        case 0 => TotalOrdering.compare(distance(x), distance(y))
10       case phaseCompared => phaseCompared
11     }
12   }
13
14   protected def distance(p: Point): Double
15   protected def phase(p: Point): Double
16 }
17
18 object OrientationOrdering {
19   implicit object Exact extends OrientationOrdering {
20     override protected def distance(p: Point): Double = PointsUtils
       .distanceFromCenter(p)
21     override protected def phase(p: Point): Double = PointsUtils.
```

```
    phase(p)
  }

  implicit object Indicator extends OrientationOrdering {
    override protected def distance(p: Point): Double = PointsUtils
    .distanceFromCenterSquared(p)
    override protected def phase(p: Point): Double = PointsUtils.
    alpha(p)
  }
}
```

Listing 3: OrientationOrdering.scala

Źródło: opracowanie własne

```scala
package utilities.geometry.convexhull.algorithms

import utilities.geometry.PointsUtils.Points
import utilities.geometry.ordering.OrientationOrdering

trait ConvexHullAlgorithm extends Product {
  def calculate(points: Points)(implicit ordering:
    OrientationOrdering): Points
}
```

Listing 4: ConvexHullAlgorithm.scala

Źródło: opracowanie własne

```scala
package utilities.geometry.convex.hull.algorithms

import org.scalatest.Assertion
import org.scalatest.wordspec.AnyWordSpec
import utilities.geometry.Point
import utilities.geometry.PointsUtils.Points
import AlgorithmTest.{EXPECTED_CONVEX_HULL, INPUT_POINTS,
  ONE_INPUT_POINT}
import utilities.geometry.convexhull.algorithms.ConvexHullAlgorithm
import utilities.geometry.ordering.OrientationOrdering

abstract class AlgorithmTest(convexHullAlgorithm:
  ConvexHullAlgorithm)(implicit ordering: OrientationOrdering)
  extends AnyWordSpec {
  private def convexHullAlgorithmName: String = convexHullAlgorithm
  .productPrefix

  private def assertExpectedOutput(input: Points, expectedOutput:
  Points): Assertion = {
    val actualConvexHull = convexHullAlgorithm.calculate(input)
    assert(actualConvexHull == expectedOutput)
  }

  s"$convexHullAlgorithmName" must {
    "throw the IllegalArgumentException for empty" in {
      assertThrows[IllegalArgumentException](convexHullAlgorithm.
  calculate(Nil))
    }

    "return the same thing for one input point" in {
      assertExpectedOutput(ONE_INPUT_POINT, ONE_INPUT_POINT)
    }

    "properly calculate convex hull" in {
      assertExpectedOutput(INPUT_POINTS, EXPECTED_CONVEX_HULL)
    }
  }
}
```

```scala

object AlgorithmTest {
  val INPUT_POINTS: Points = List(
    Point(1, 3),
    Point(2, 4),
    Point(2, 2),
    Point(4, 6),
    Point(3, 4),
    Point(4, 4),
    Point(4, 3),
    Point(2, 6),
    Point(5, 2),
    Point(6, 4),
    Point(5, 5)
  )

  val EXPECTED_CONVEX_HULL: Points = List(
    Point(6, 4),
    Point(4, 6),
    Point(2, 6),
    Point(1, 3),
    Point(2, 2),
    Point(5, 2)
  )

  val ONE_INPUT_POINT: Points = List(
    Point(1, 2)
  )
}
```

Listing 5: AlgorithmTest.scala

Źródło: opracowanie własne

2. Algorytm Grahama

```scala
package utilities.geometry.convexhull.algorithms
import utilities.geometry.PointsUtils.Points
import utilities.geometry.ordering.OrientationOrdering
import utilities.geometry.{PointsCycle, PointsUtils}


case object Graham extends ConvexHullAlgorithm {

  override def calculate(points: Points)(implicit ordering:
    OrientationOrdering): Points = {
    require(points.nonEmpty, "You can not calculate convex hull of
    an empty points set.")
    val centroid = PointsUtils.calculateCentroid(points)
    val pointsSorted = points.map(_ - centroid).sorted
    val pointsCycle = new PointsCycle(pointsSorted)

    pointsCycle.getHull.map(_ + centroid)
  }
}
```

Listing 6: Graham.scala

Źródło: opracowanie własne

```
package utilities.geometry

import utilities.geometry.PointsUtils.Points

class PointsCycle(points: Points) extends Iterable[Point] {

  private lazy val first = NoPointRef.next

  override def iterator: Iterator[Point] = new Iterator[Point] {
    private var currentPointRef: AbstractPointRef = NoPointRef

    override def hasNext: Boolean = currentPointRef.isNotLast

    override def next(): Point = {
      val next = currentPointRef.next
      currentPointRef = next
      next.point
    }
  }

  private def createPointRefs(points: Points): AbstractPointRef =
    points.foldLeft[AbstractPointRef](NoPointRef)(_.addPoint(_)).
   addLastPointAndReturn()

  createPointRefs(points)

  trait AbstractPointRef {
    var next: PointRef = _
    var prev: PointRef = _

    def isNotFirst: Boolean
    def isNotLast: Boolean

    def addPoint(point: Point): PointRef

    protected def addLastPoint(): Unit

    final def addLastPointAndReturn(): AbstractPointRef = {
      addLastPoint()
```

```
      this
    }
  }

  class PointRef(val point: Point) extends AbstractPointRef {

    final def isNotFirst: Boolean = this ne first
    final def isNotLast: Boolean = next ne first

    override def addPoint(point: Point): PointRef = {
      val pointRef = new PointRef(point)
      pointRef.prev = this
      next = pointRef
      pointRef
    }

    def isInTriangle(t2: PointRef, t3: PointRef): Boolean =
      Orientation.calculateThreePointsOrientation(t2.point, t3.
  point, point) == Orientation.Left

    override protected def addLastPoint(): Unit = {
      next = first
      first.prev = this
    }
  }

  object NoPointRef extends AbstractPointRef {

    override def addPoint(point: Point): PointRef = {
      val pointRef = new PointRef(point)
      next = pointRef
      pointRef
    }

    override def isNotFirst: Boolean = first ne null
    override def isNotLast: Boolean = first ne null

    override protected def addLastPoint(): Unit = ()
  }
```

```scala

  def getHull: Points = {

    if (NoPointRef.isNotLast) {
      var q = NoPointRef.next
      while (q.isNotLast) {
        if (q.next.isInTriangle(q, q.next.next)) {
          q.next = q.next.next
          q.next.prev = q
          if (q.isNotFirst) q = q.prev
        } else q = q.next
      }
    }
    toList
  }

}
```

Listing 7: PointsCycle.scala

Źródło: opracowanie własne

```scala
package utilities.geometry.convex.hull.algorithms

import utilities.geometry.convexhull.algorithms.Graham
import utilities.geometry.ordering.OrientationOrdering.Indicator

class GrahamAlgorithmTest extends AlgorithmTest(Graham)
```

Listing 8: GrahamAlgorithmTest.scala
Źródło: opracowanie własne

3. Algorytm Jarvisa

Dzięki wykorzystaniu pattern matchingu dostępnego w Scali, możliwe jest przejrzyste zaimplementowanie algorytmu Jarvisa, który, jak widać jest dużo mniej skomplikowany od algorytmu Grahama. Mniejsze skomplikowanie implementacji wiąże się jednak z dużo większą złożonością obliczeniową.

```scala
package utilities.geometry.convexhull.algorithms

import utilities.geometry.{Point, PointsUtils}
import utilities.geometry.PointsUtils.Points

case class Jarvis() extends ConvexHullAlgorithm {

  private val FIRST_POINT_ORDERING = new Ordering[Point] {
    override def compare(x: Point, y: Point): Int = {
      val byX = Ordering.Double.TotalOrdering.compare(x.x, y.x)
      lazy val byY = Ordering.Double.TotalOrdering.compare(x.y, y.y
)
      if (byX == 0) byY
      else byX
    }
  }


  override protected def nonEmptyCalculate(points: Points): Points
 = {
    val firstPoint = points.max(FIRST_POINT_ORDERING)
    val pZero = Point(firstPoint.x - 1, firstPoint.y)

```

```scala
    var pI = firstPoint
    var pIMinusOne = pZero

    val pointsBuilder = List.newBuilder[Point]
    pointsBuilder += firstPoint

    while (true) {
      points
        .filterNot(_ == pI)
        .maxByOption(PointsUtils.angleBetweenThreePoints(pIMinusOne
  , pI, _)) match {
        case Some(‘firstPoint‘) => return pointsBuilder.result()
        case Some(pIPlusOne) =>
          pointsBuilder += pIPlusOne
          pIMinusOne = pI
          pI = pIPlusOne
        case _ => return pointsBuilder.result()
      }
    }
    Nil
  }

}
```

Listing 9: Jarvis.scala
Źródło: opracowanie własne

```scala
package utilities.geometry.convex.hull.algorithms

import utilities.geometry.convexhull.algorithms.Jarvis

class JarvisAlgorithmTest extends AlgorithmTest(Jarvis())
```

Listing 10: JarvisAlgorithmTest.scala
Źródło: opracowanie własne

Rozdział IV
Dynamiczna otoczka wypukła

1. Algorytm

2. Implementacja w języku Scala

Rozdział V
Podsumowanie

Bibliografia

[1] J. Jourban i Y. Gabay, *A Method for Construction of 2D Hull For Generalized Cartographic Representation*. 2000.

[2] N. Singh, R. Arya i R. Agrawal, *A convex hull approach in conjunction with Gaussian mixture model for salient object detection*. 2016.

[3] R. L. Graham i F. Yao, *Finding the Convex Hull of a Simple Polygon*. 1981.

[4] R. Laurini, *Geographic Knowledge Infrastructure*. 2017, s. 139–156.

[5] P. Heckbert, *Graphics Gems IV* (Graphics gems series). 1994, s. 30.

[6] Datopian, *Country Polygons as GeoJSON — datahub.io*, `https://datahub.io/core/geo-countries#resource-countries`, [Data dostępu: 13:34 20.08.2023].

[7] *EPSG:2180*, `https://epsg.io/2180`, [Data dostępu: 00:05 02.09.2023].

[8] *GeoTools*, `https://www.geotools.org/`, [Data dostępu: 00:21 02.09.2023].

[9] C.-W. Huang i T.-Y. Shih, *On The Complexity Of Point-In-Polygon Algorithms*. 1996, s. 109–118.

[10] I. Biederman, *Human image understanding: Recent research and a theory*. 1985, s. 30.

[11] D. Ndayikengurukiye i M. Mignotte, *CoSOV1Net: A Cone- and Spatial-Opponent Primary Visual Cortex-Inspired Neural Network for Lightweight Salient Object Detection*. 2023.

[12] A. Borji, M.-M. Cheng, Q. Hou, H. Jiang i J. Li, *Salient object detection: A survey - computational visual media*. 2019.

[13] C. Harris i M. Stephens, *A Combined Corner And Edge Detector*. 1988.

[14] K. Bala, *CS4670: Computer Vision: Lecture 7: Harris Corner Detection*, [Data dostępu: 15:02 02.09.2023], 2015.

[15] *ślad macierzy, Encyklopedia PWN*, `https://encyklopedia.pwn.pl/haslo/;3983965`, [Data dostępu: 15:41 02.09.2023].

[16] *Matrix Trace – from Wolfram MathWorld*,
`https://mathworld.wolfram.com/MatrixTrace.html`,
[Data dostępu: 22:11 02.09.2023].

Spis rysunków

Spis listingów