ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1994
Seria: INFORMATYKA z. 25 Nr kol. 1236

Robert SADŁOWSKI

ALGORYTM WYZNACZANIA NAJBARDZIEJ ODLEGŁYCH PUNKTÓW

Streszczenie. W artykule przedstawiono sposoby znajdowania dwóch najbardziej odległych punktów spośród danego skończonego zbioru punktów na płaszczyźnie. Opisano algorytm oparty na redukcji przestrzeni poszukiwań.

Przedstawiono ideę modyfikacji tego algorytmu polegającą na znalezieniu wspólnej części regionów powstałych w wyniku redukcji przestrzeni poszukiwań. Dzięki temu zmniejszeniu uległa liczba punktów, pomiędzy którymi badane są odległości. W części doświadczalnej pokazano, przy jakich rozkładach punktów na płaszczyźnie zmodyfikowana wersja algorytmu działa efektywniej.

AN ALGORITHM TO DETERMINE THE TWO MOST DISTANT POINTS

Summary. The article presents a method of finding the two most distant points in a given finite planar set of points. The proposed algorithm is based on the principle of reducing the search space.

The algorithm has been modified by further restricting the search area to the intersection of the regions determined by the original search-space reduction step. Practical tests show for what kinds of point distribution the modified algorithm is faster than the original one.

UN ALGORITHME DE RECHERCHE DES POINTS LES PLUS ELOIGNES

Résumé. L'article présente quelques méthodes de rècherche des deux points les plus eloignés dans un donné ensemble fini de points sur le plan. Un algorithme est proposé dont le fonctionnement est basé sur une rèduction de l'espace de recherche.

L'auteur propose aussi une modification de l'algorithme par la limitation de l'espace de recherche à la part commune des régions trouvées pendent le pas de rèduction. Cette opération diminue le nombre des points entre lesquels les distances doivent être calculées. La part expérimentale discute pour quels distributions des points la version modifiée de l'algorithme est plus efficace.

1. Wstęp

Jednym z problemów geometrii obliczeniowej jest znajdowanie najbardziej odległych w danym zbiorze punktów. Problem ten występuje często podczas komputerowej analizy obrazów, np. w metalurgii (w trakcie pomiarów uszkodzeń próbki metalu) [4], analizie teleradiogramów (wykrywanie nieprawidłowości budowy anatomicznej) [5], w biologii przy wyznaczeniu długości obiektu biologicznego, a także w technice wojskowej w celu precyzyjnego określenia toru pocisków.

Niech $S = \{P_a, P_b, P_c, ...P_n\}$ będzie skończonym zbiorem punktów na płaszczyźnie. Każdy punkt opisany jest współrzędnymi kartezjańskimi x_i, y_i. Maksymalną odległość między punktami nazwano średnicą zbioru punktów i oznaczono przez $DIAM(S)$. Odległość tę zdefiniowano następująco:

$$DIAM(S) = \max_{(i,j)}[d(P_i, P_j)],$$

gdzie $d(P_i, P_j)$ jest euklidesową miarą odległości pomiędzy punktami, czyli

$$d(P_i, P_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \quad P_i, P_j \in S, \quad i, j = 1, 2, ..., n.$$

Oczywistą metodą znajdowania najbardziej odległych punktów na płaszczyźnie jest metoda „każdy z każdym" (tzn. należy obliczyć odległości pomiędzy wszystkimi parami punktów i wybrać parę najbardziej odległą). W tym celu należy wyznaczyć $\frac{n*(n-1)}{2}$ odległości pomiędzy punktami. Złożoność tej metody wynosi zatem $O(\frac{n*(n-1)}{2})$ operacji.

W pracy [10] udowodniono:

Twierdzenie 1. *Średnica danej figury jest równa średnicy powłoki wypukłej tej figury.*

Problem wyznaczania najbardziej odległych punktów na płaszczyźnie sprowadza się zatem do problemu wyznaczenia powłoki wypukłej danego zbioru punktów oraz analizy odległości pomiędzy punktami tej powłoki. Najbardziej znanymi algorytmami wyznaczania powłoki wypukłej zbioru punktów są:

1. Algorytm Grahama [7].

   Sposób postępowania według tego algorytmu jest następujący:

   - spośród danych punktów wybierany jest punkt $P_0 = (x_0, y_0)$ o najmniejszej wartości współrzędnej y_i czyli:

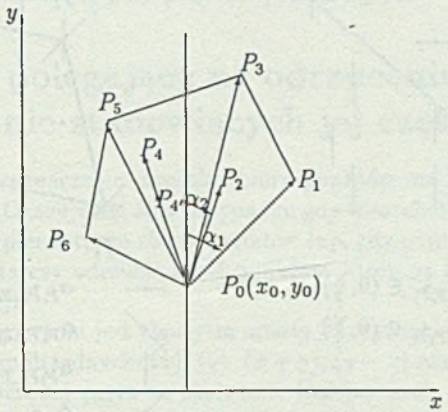
     $$y_0 = \min_i y_i, \quad x_0 = x_i.$$

     Pozostałe punkty traktowane są jako wektory względem punktu P_0,
   - kąty, które tworzą wektory badanych punktów $(\vec{P_iP_0})$ z osią y, porządkowane są malejąco [1].

[1] W przypadku jeśli wartość kąta α_i jest taka sama dla kilku wektorów badanych punktów, do dalszych obliczeń wybierany jest punkt skrajny (najbardziej odległy od punktu P_0).

- po uporządkowaniu badane są pary wektorów: $P_{i-1}\vec{P}_i, P_i\vec{P}_{i+1}$, i jeśli para wektorów jest dodatnio zorientowana, to wierzchołek P_i jest dopisywany do listy wierzchołków powłoki wypukłej.

Sposób tworzenia uporządkowanego zbioru punktów według tego algorytmu można prześledzić na rys. 1.



Rys. 1. Uporządkowanie punktów w algorytmie Grahama
Fig. 1. Ordering of points in the Graham algorithm

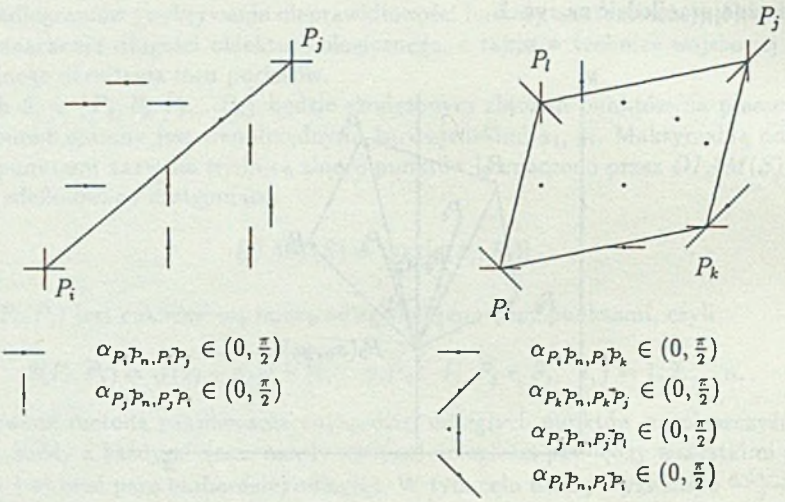
2. Algorytm Greena i Silvermana.

Metoda Greena i Silvermana polega na wyznaczeniu dwóch dowolnych punktów powłoki wypukłej (np. punktów P_i i P_j o największej i najmniejszej wartości współrzędnej x), a następnie na:

- wyznaczeniu wszystkich punktów P_n takich, że kąt $\alpha_{i,n}$ pomiędzy wektorem badanego punktu $P_i\vec{P}_n$ a wektorem $P_i\vec{P}_j$ jest dodatni,
- wyznaczeniu (spośród punktów P_n) punktu P_k najbardziej odległego od wektora $P_i\vec{P}_j$,
- wyznaczeniu wszystkich punktów P_m takich, że kąt $\alpha_{j,m}$ pomiędzy wektorem badanego punktu $P_j\vec{P}_m$ a wektorem $P_j\vec{P}_i$ jest dodatni,
- wyznaczeniu (spośród punktów P_m) punktu P_l najbardziej odległego od wektora $P_j\vec{P}_i$,
- wyznaczeniu punktów zewnętrznych do wielokąta: $P_i...P_k...P_j...P_l...P_i$, znalezieniu punktów najbardziej odległych od danego wektora, ... aż do momentu powstania powłoki $P_i...P_k...P_j...P_l...P_i$, która nie będzie posiadać żadnego punktu zewnętrznego. Punkty $P_i...P_k...P_j...P_l...$ tworzą wierzchołki powłoki wypukłej,

- uporządkowaniu wektorów łączących kolejne wierzchołki powłoki wypukłej odwrotnie do ruchu wskazówek zegara.

Sposób wyznaczenia powłoki wypukłej metodą Greena i Silvermana przedstawiono na rys. 2.



Rys. 2. Pierwsze kroki tworzenia powłoki wypukłej metodą Greena i Silvermana
Fig. 2. The first steps of creating the convex hull by the Green and Silverman method

Przed przedstawieniem własności figur wypukłych przytoczę definicję pojęcia prostej wspornikowej wykorzystywanej do wyznaczania punktów najbardziej odległych:

Definicja. *Prosta „wspornikowa" do figury F jest to prosta l przechodząca przez wierzchołek figury F taka, że cała figura F znajduje się po jednej stronie prostej l. Punkty figury, przez które można poprowadzić równoległe proste „wspornikowe", nazywają się punktami przeciwległymi.*

Okazuje się, że w przypadku figur wypukłych istnieje szybszy algorytm (tzn. algorytm, którego złożoność obliczeniowa jest mniejsza niż $O(\frac{n*(n-1)}{2})$ operacji) wyznaczenia wierzchołków najbardziej odległych. Algorytm ten opiera się na własności figur wypukłych przedstawionej w [11] jako:

Twierdzenie 2. *Średnica wypukłej figury jest równa najdłuższej odległości pomiędzy dwoma punktami przeciwległymi tej figury.*

W celu znalezienia punktów przeciwległych należy wybrać dowolny wierzchołek figury wypukłej P_i, a następnie, poruszając się przeciwnie do ruchu wskazówek zegara, wyznaczyć punkt $Q_R^{(i)}$ najbardziej odległy od prostej $\overrightarrow{P_{i-1}P_i}$ [2]. Analogicznie należy wyznaczyć wierzchołek $Q_L^{(i)}$ najbardziej odległy od prostej P_iP_{i+1} poruszając się w kierunku zgodnym z ruchem wskazówek zegara.

Punkty $Q_S^{(i)}$, do których zaliczamy $Q_R^{(i)}$, $Q_L^{(i)}$, oraz wszystkie punkty pomiędzy tymi wierzchołkami tworzą z punktem P_i pary punktów przeciwległych.

2. Algorytmy polegające na odrzuceniu wierzchołków figury nie stanowiących jej części wypukłej

Algorytmy oparte na wyznaczeniu powłoki zbioru punktów mają złożoność obliczeniową rzędu $O(n \log n)$ [3]. Okazuje się, że w przypadku gdy wierzchołki powłoki wypukłej stanowią niewielki procent pierwotnego zbioru punktów (np. gdy punkty są rozmieszczone niezależne od siebie), wystarczy odrzucić część punktów, które na pewno nie są wierzchołkami powłoki wypukłej.

Przykładem takiego algorytmu jest algorytm oparty na wyznaczeniu punktów o ekstremalnych wartościach współrzędnych: (x), (y), $(x+y)$, $(y-x)$ oraz punktów leżących na zewnątrz figury wyznaczonej przez te ekstrema. Kolejny krok, to analiza odległości pomiędzy punktami nie stanowiącymi wnętrza figury wyznaczonej przez te ekstrema. Sposób wyznaczenia punktów o ekstremalnych wartościach współrzędnych przedstawiono na rys. 3.

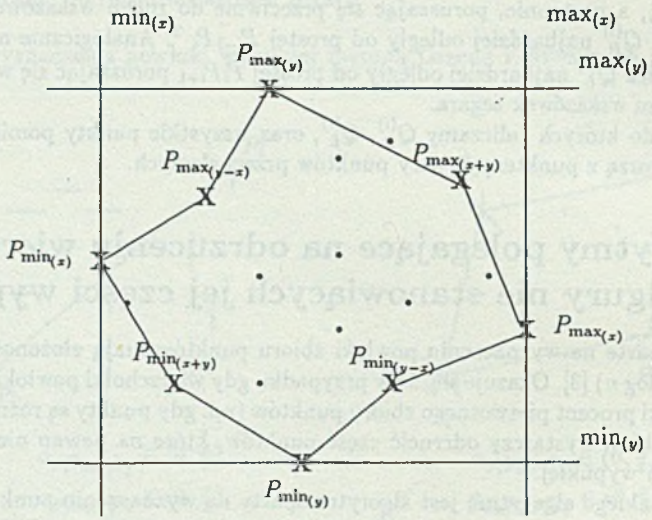
Algorytm ten cechuje „pesymistyczna" złożoność obliczeniowa rzędu $O(n^2)$ (oczywiście $O(n^2) > O(n \log n)$). Badania opisane w [1] wykazują jednak liniową zależność czasu wykonania programu od liczby punktów rozmieszczonych na płaszczyźnie.

2.1. Podział podzbioru punktów S na „naprzemianległe" regiony

Należy zwrócić uwagę, że punkty maksymalnie oddalone od siebie nie leżą w sąsiedztwie. Wystarczy więc pogrupować punkty w rejony i badać tylko pary punktów z tych rejonów, w których podejrzewamy istnienie punktów maksymalnie odległych. Naszym celem jest z jednej strony zredukowanie jak największej liczby punktów, z drugiej zaś wykonanie tej operacji w możliwie najkrótszym czasie. To rozumowanie prowadzi do następującej koncepcji algorytmu.

Spośród danych punktów należących do zbioru S wybieramy podzbiór E punktów o ekstremalnych wartościach współrzędnych: $(x), (y), (x+y), (y-x)$. Sposób wyznaczenia punktów o ekstremalnych wartościach współrzędnych przedstawiono na rys. 3.

[2] W przypadku gdy odległość punktów od prostej jest maksymalna dla dwóch wierzchołków (występują boki równoległe), należy wybrać pierwszy z badanych punktów. W celu wyznaczenia odległości punktu Q od prostej należy wyznaczyć pole trójkąta $P_{i-1}P_iQ$.



Rys. 3. Sposób wyznaczenia punktów o ekstremalnych wartościach współrzędnych x, y, $x+y$, $x-y$
Fig. 3. Method of determining points with extreme values of coordinates and coordinate combinations x, y, $x+y$, $y-x$

Przez P_1, P_2, P_3, P_4 oznaczmy punkty odpowiednio o maksymalnych i minimalnych wartościach współrzędnej x i y, tzn:

$$P_1 = \min_{\{x\}}(x), \min_{\{y\}}(y), \quad x, y \in (0, \infty) \tag{1}$$

$$P_2 = \max_{\{x\}}(x), \min_{\{y\}}(y), \quad x, y \in (0, \infty) \tag{2}$$

$$P_3 = \max_{\{x\}}(x), \max_{\{y\}}(y), \quad x, y \in (0, \infty) \tag{3}$$

$$P_4 = \min_{\{x\}}(x), \max_{\{y\}}(y), \quad x, y \in (0, \infty) \tag{4}$$

Czyli dane punkty należące do zbioru S wpisujemy w prostokąt $P_1P_2P_3P_4$. Należy zauważyć, że średnica zbioru S jest nie mniejsza niż średnica zbioru E, oczywiście średnica zbioru E jest większa lub równa długości najdłuższego boku prostokąta $P_1P_2P_3P_4$, czyli:

$$\max(d(P_1, P_2), d(P_2, P_3)) \leq DIAM(E) \leq DIAM(S).$$

Każdemu wierzchołkowi P_i odpowiada region R_{i+2} taki, że dla każdego punktu A_j należącego do R_{i+2} odległość:

$$d(P_i, A_j) \geq DIAM(E), \quad \text{gdzie } i = 1, 2, 3, 4 \quad \text{i} \quad R_5 \equiv R_1 \quad \text{i} \quad R_6 \equiv R_2.$$

Oznaczmy:

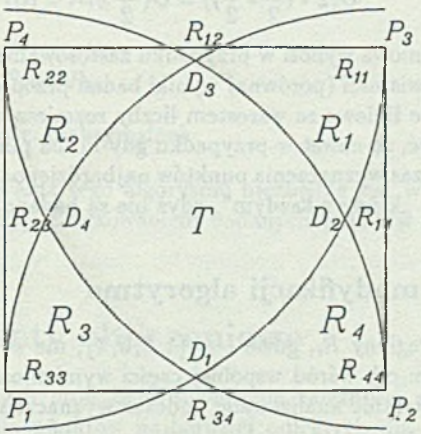
$$R = R_1 \cup R_2 \cup R_3 \cup R_4;$$
$$Q = \text{prostokąt } P_1P_2P_3P_4;$$
$$T = Q \setminus R.$$

Można zauważyć, że dla dowolnego punktu A_i należącego do zbioru T oraz dowolnego punktu A_j należącego do S odległość $d(A_i, A_j)$ jest mniejsza od średnicy zbioru E, gdzie $i = 1, 2$. Dzięki temu wszystkie punkty należące do T (wnętrza zbioru S) można odrzucić przy szukaniu maksymalnie odległych punktów figury.

W pracy [1] udowodniono twierdzenie umożliwiające dalszą redukcję podzbioru rozmieszczonych na płaszczyźnie punktów, wśród których poszukujemy punktów najbardziej odległych:

Twierdzenie 3. *Jeżeli odległość danej pary punktów jest średnicą zbioru S, to jeden punkt należy do obszaru R_i, a drugi punkt należy do obszaru R_{i+2}, gdzie $i = 1, 2$.*

Sposób tworzenia regionów R_i przedstawiono na rys. 4.



Rys. 4. Ilustracja tworzenia regionów naprzemianległych
Fig. 4. The illustration of the creation of alternate regions

Wersja podstawowa algorytmu opartego na podziale zbioru S na regiony składa się z następujących czynności:

1. Wyznaczenie zbioru ekstremalnych punktów względem współrzędnych: x, y, $x + y$, $y - x$.

2. Wyznaczenie wierzchołków prostokąta $P_1P_2P_3P_4$ „opisanego" na zbiorze S (sposób wyznaczenia punktów P_1, P_2, P_3, P_4 podano w równaniach (1) ÷ (4)).

3. Wyznaczenie średnicy zbioru punktów o ekstremalnych wartościach wzlędem współrzędnych x, y, $x+y$, $y-x$.

4. Wyznaczenie wszystkich punktów figury, które należą do kolejnych regionów R_i, gdzie $i = 1,2,3,4$ (przynależność punktów do poszczególnych regionów bada się wyznaczając odległość badanego punktu od kolejnych wierzchołków prostokąta $P_1P_2P_3P_4$ i sprawdza, czy odległość jest mniejsza niż średnica wyznaczona w poprzednim punkcie. Zauważmy, że badany punkt może należeć co najwyżej do dwóch regionów).

5. Wyznaczenie odległości pomiędzy wszystkimi parami punktów P i Q, gdzie $P \in R_i$ i $Q \in R_{i+2}$, $i = 1,2$.

6. Wyznaczenie odległości maksymalnej.

Najgorszy przypadek wystąpi, jeżeli wszystkie badane punkty znajdować się będą w regionach R_i, i połowa punktów należeć będzie do wspólnej części obszarów R_1 i R_2, a druga połowa do wspólnej części obszarów R_3 i R_4. Wtedy liczba niezbędnych obliczeń wyniesie:

$$O(2 * (\frac{n}{2} * \frac{n}{2})) = O(\frac{n^2}{2}).$$

Stąd też złożoność obliczeniowa wynosi w przypadku zastosowania tego algorytmu $O(\frac{n^2}{2})$ operacji. Jednak w rzeczywistości (porównaj wyniki badań przeprowadzonych w [1]) czas działania algorytmu rośnie liniowo ze wzrostem liczby rozmieszczonych na płaszczyźnie punktów. Należy zauważyć, że nawet w przypadku gdy liczba punktów należących do T będzie niewielka, to i tak czas wyznaczenia punktów najbardziej odległych będzie znacznie krótszy niż dla algorytmu „każdy z każdym", gdyż nie są badane punkty z sąsiedztwa.

2.2. Możliwości modyfikacji algorytmu

Należy zauważyć, że regiony R_i, gdzie $i \in \{1,2,3,4\}$, nie są rozłączne. W związku z tym pojawia się pytanie: czy wśród wspólnej części wymienionych regionów znajdują się obszary, które są dwukrotnie analizowane podczas wyznaczania odległości pomiędzy punktami?

Przez R_{ij} oznaczono część wspólną obszarów R_i i R_j. Z analizy rys. 4 wynika, że w przypadku porównywania punktów należących do obszarów:

- R_{21} i R_{43},
- R_{23} i R_{41},

te podzbiory są analizowane zarówno podczas analizy regionów R_1 i R_3, jak i regionów R_2 i R_4. Jeśli podzbiór R_x zdefiniowany jako: $R_x = (R_{21} \cap R_{43}) \cup (R_{23} \cap R_{41})$ nie jest pusty, to liczbę punktów, pomiędzy którymi należy wyznaczyć odległość można zmniejszyć.

Na powyższej obserwacji opiera się koncepcja zmodyfikowanego algorytmu, składającego się z następujących czynności:

1. Wyznaczenie zbioru ekstremalnych punktów względem współrzędnych: $x, y, x+y, y-x$.
2. Wyznaczenie wierzchołków prostokąta $P_1P_2P_3P_4$ „opisanego" na zbiorze S (sposób wyznaczenia punktów P_1, P_2, P_3, P_4 podano w równaniach (1) ÷ (4)).
3. Wyznaczenie średnicy zbioru punktów o ekstremalnych wartościach względem współrzędnych $x, y, x+y, y-x$.
4. Wyznaczenie wszystkich punktów figury, które należą do kolejnych regionów R_i, gdzie $i = 1,2,3,4$ (w przypadku regionów R_2 i R_4 wyznaczana jest przynależność punktów do regionów R_{ij}).
5. Wyznaczenie odległości pomiędzy parami punktów P i Q, gdzie $P \in R_1$ i $Q \in R_3$.
6. Wyznaczenie odległości pomiędzy parami punktów P i Q, gdzie:
   - $P \in R_{21}$ i $Q \in R_{41} \cup R_{44}$,
   - $P \in R_{22}$ i $Q \in R_4$,
   - $P \in R_{23}$ i $Q \in R_{43} \cup R_{44}$.
7. Wyznaczenie odległości maksymalnej.

W przypadku zastosowania tego algorytmu niezbędne jest wykonanie dodatkowych obliczeń związanych z przyporządkowaniem badanych punktów do kolejnych regionów R_{ij}.

3. Eksperymenty obliczeniowe

Celem przeprowadzonych prób było znalezienie zależności pomiędzy liczbą punktów a czasem wyznaczenia punktów najbardziej odległych dla różnych rozkładów na płaszczyźnie generowanych punktów. Należało również odpowiedzieć na pytanie, czy we wszystkich przypadkach efektywne jest stosowanie zmodyfikowanej wersji algorytmu.

3.1. Sposób przeprowadzenia eksperymentu obliczeniowego

Eksperyment przeprowadzono dla różnych rozkładów generowanych na płaszczyźnie punktów. Współrzędne x_i, y_i punktu P_i generowano niezależnie lub w przypadku dwóch ostatnich eksperymentów, współrzędną y_i obliczano w taki sposób, aby powstał obrys odpowiednio okręgu lub elipsy. W eksperymencie mierzono nie tylko całkowity czas wykonania programu dla różnej liczby punktów, ale również stosunek czasu potrzebnego na „wyszukiwanie" i obliczenie odległości oraz liczbę punktów należących do regionów

R_i [3]. W przypadku pomiarów czasu wykonania programu opartego na zmodyfikowanej wersji algorytmu mierzono również iloczyn (n_2) liczby punktów P i Q, gdzie: $P \in R_{21}$ i $Q \in R_{43}$ lub $P \in R_{23}$ i $Q \in R_{41}$. Odpowiada to parom punktów, pomiędzy którymi odległości nie będą powtórnie wyznaczane – czyli „zysk" wynikający z zastosowania tej wersji algorytmu.

3.2. Wyniki eksperymentu obliczeniowego

Opisane w pracy algorytmy zostały zaimplementowane w języku C. Do wyznaczenia współrzędnych badanych punktów posłużono się generatorem liczb pseudolosowych. Dla rozkładu normalnego współrzędne wyznaczano zgodnie z zależnością:

$$(x \text{ lub } y) = \frac{\sum_{i=1}^{n} x_i - nE(x)}{n\sigma},$$

gdzie $n = 12$, $\mu = \frac{E(x_i)}{2}$, x_i ma rozkład równomierny w przedziale $(0, 1)$.

Aby podpisy pod tablicami i wykresami były czytelne, program oparty na podstawowej wersji algorytmu oznaczano jako program 1, natomiast program oparty na zmodyfikowanej wersji jako program 2.

3.2.1. Eksperyment 1

W pierwszym eksperymencie punkty generowano na płaszczyźnie niezależnie, zgodnie z rozkładem normalnym:

- o wartości oczekiwanej $\mu = 250$ oraz odchyleniu standardowym $\sigma = 50$ dla współrzędnych x i y,
- o wartości oczekiwanej $\mu = 250$ oraz odchyleniach standardowych odpowiednio $\sigma_x = 50$ (dla współrzędnej x), $\sigma_y = 100$ (dla współrzędnej y).

Wyniki pomiarów czasu wykonania programu opartego na wersji podstawowej algorytmu przedstawiono w tablicy 1. W tablicy 2 przedstawiono wyniki pomiarów czasu wykonania programu dla rozkładu normalnego. Program oparty był na zmodyfikowanej wersji algorytmu. Czas wykonania programów w funkcji liczby generowanych na płaszczyźnie punktów dla przyjętego rozkładu pokazano na rys. 5. W obu badanych przypadkach można zaobserwować:

- liniową zależność czasu wykonania programów od liczby generowanych na płaszczyźnie punktów,
- nieliniowy wzrost liczby punktów należących do obszarów R_i (powolny wzrost liczby punktów należących do obszarów R_i gwarantuje liniowy wzrost czasu wykonania programów również dla większej próby),
- nieefektywność stosowania programu opartego na zmodyfikowanej wersji algorytmu.

[3] Należy zauważyć, że im mniej punktów należy do regionów R_i w stosunku do liczby badanych punktów, tym ilość wyznaczanych odległości zmniejszy się.

Tablica 1

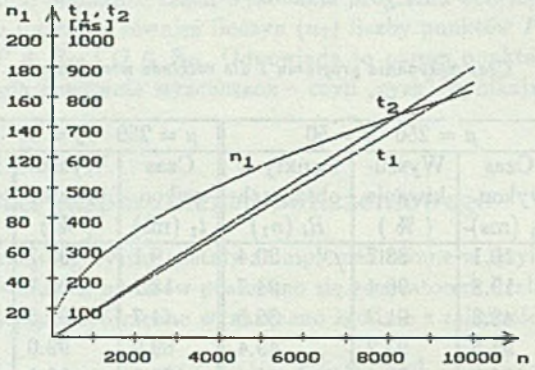
Czas wykonania programu 1 dla rozkładu normalnego

	$\mu = 250 \quad \sigma = 50$			$\mu = 250 \quad \sigma_x = 50 \quad \sigma_y = 100$		
Liczba (n)	Czas wykon. t_1 (ms)	Wyszukiwanie (%)	Punkty w obszarach R_i (n_1)	Czas wykon. t_1 (ms)	Wyszukiwanie (%)	Punkty w obszarach R_i (n_1)
100	10.1	88.7	20.4	9.1	98.7	6.0
200	19.8	90.4	24.7	18.1	98.7	8.8
500	48.8	91.2	36.5	44.7	99.6	9.0
1000	95.5	93.2	49.4	89.9	99.0	19.2
2000	186.5	95.4	66.4	179.5	99.1	28.8
3000	295.9	90.2	102.4	269.4	99.0	45.0
4000	392.7	90.6	111.0	359.6	98.9	43.8
5000	469.1	94.8	127.0	449.4	98.9	100.5
6000	618.3	86.3	151.0	541.0	98.6	119.0
8000	771.9	92.1	132.6	720.2	98.8	131.0
10000	971.4	91.5	187.4	919.6	96.7	231.0

Tablica 2

Czas wykonania programu 2 dla rozkładu normalnego

	$\mu = 250 \quad \sigma = 50$			
Liczba (n)	Czas wykon. t_1 (ms)	Wyszukiwanie (%)	Iloczyn punktów (n_2)	Zysk (strata) (%)
100	10.6	89.2	0.6	- 4.4
200	20.3	90.9	2.3	- 2.5
500	49.6	91.4	2.2	- 1.6
1000	96.5	93.3	2.0	- 1.1
2000	187.9	95.4	3.6	- 0.8
3000	298.0	90.3	7.4	- 0.7
4000	395.0	90.7	6.0	- 0.6
5000	471.9	94.8	1.8	- 0.6
6000	621.5	86.4	6.8	- 0.5
8000	774.8	92.2	3.2	- 0.4
10000	991.3	91.6	9+.8	- 0.4



Rys. 5. Czas wykonania programów dla rozkładu normalnego
Fig. 5. Program execution times for normal distribution

3.2.2. Eksperyment 2

W tym eksperymencie punkty generowane są na płaszczyźnie niezależnie, zgodnie z rozkładem równomiernym w prostokącie:

- o bokach $500 * 500$,
- o bokach $500 * 200$.

W tablicy 3 przedstawiono wyniki pomiarów czasu wykonania programu dla punktów, generowanych zgodnie z tym rozkładem. Program oparty był na wersji podstawowej algorytmu. W tablicy 4 przedstawiono wyniki pomiarów czasu wykonania programu dla rozkładu równomiernego. Program oparto na zmodyfikowanej wersji algorytmu. Czas wykonania programów w funkcji liczby generowanych na płaszczyźnie punktów dla przyjętego rozkładu przedstawiono na rys. 6.

Tablica 3

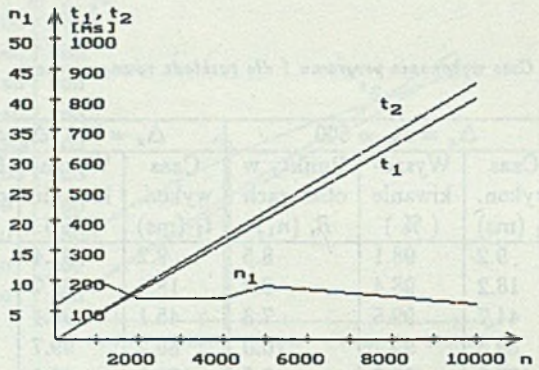
Czas wykonania programu 1 dla rozkładu równomiernego

	$\Delta_x = \Delta_y = 500$			$\Delta_x = 500$ $\Delta_y = 200$		
Liczba (n)	Czas wykon. t_1 (ms)	Wyszukiwanie (%)	Punkty w obszarach R_i (n_1)	Czas wykon. t_1 (ms)	Wyszukiwanie (%)	Punkty w obszarach R_i (n_1)
100	9.2	98.1	8.5	9.2	97.4	7.8
200	18.2	98.4	9.2	18.1	98.7	8.6
500	44.7	99.5	7.3	45.1	99.1	14.5
1000	89.3	99.7	10.0	89.2	99.7	9.7
2000	178.0	99.8	7.3	178.0	99.9	8.6
3000	266.9	99.7	7.0	266.9	99.95	7.2
4000	355.8	99.9	7.2	356.1	99.85	12.7
5000	444.8	99.9	9.0	444.9	99.90	10.3
6000	533.6	99.95	7.3	533.9	99.90	11.3
8000	711.3	99.98	7.2	711.4	99.97	6.7
10000	889.1	99.99	5.9	889.3	99.96	9.0

Tablica 4

Czas wykonania programu 2 dla rozkładu równomiernego

	$\Delta_x = \Delta_y = 500$			
Liczba (n)	Czas wykon. t_1 (ms)	Wyszukiwanie (%)	Iloczyn punktów (n_2)	Zysk (strata) (%)
100	9.3	98.1	0	- 2.0
200	18.4	98.4	0	- 1.1
500	44.9	99.5	0	- 0.4
1000	89.5	99.7	0	- 0.25
2000	178.2	99.9	0	- 0.10
3000	267.1	99.9	0	- 0.06
4000	355.9	99.95	0	- 0.05
5000	445.0	99.95	0	- 0.04
6000	533.7	99.97	0	- 0.03
8000	711.5	99.98	0	- 0.02
10000	889.2	99.99	0	- 0.01



Rys. 6. Czas wykonania programów dla rozkładu równomiernego
Fig. 6. Program execution times for uniform distribution

W tych eksperymentach można zaobserwować:

- liniową zależność czasu wykonania programów od liczby generowanych na płaszczyźnie punktów,
- niezależność liczby puktów należących do obszarów R_i od liczby generowanych punktów,
- nieefektywność stosowania programu opartego na zmodyfikowanej wersji algorytmu.

3.2.3. Eksperyment 3

W tym eksperymencie współrzędną x generowano na płaszczyźnie niezależnie, zgodnie z rozkładem równomiernym na przedziale (-500,500), a współrzędną y obliczano zgodnie z wzorami:

$y = \sqrt{r^2 - x^2}$, $r = 500$ (połowa punktów),
$y = -\sqrt{r^2 - x^2}$, $r = 500$ (połowa punktów),
dzięki temu powstał obrys okręgu o promieniu $r = 500$ i środku w punkcie $(0, 0)$ układu współrzędnych.

W tablicy 5 przedstawiono wyniki pomiarów czasu wykonania programu w przypadku, gdy program oparty był na podstawowej wersji algorytmu. Dodatkowo pokazano czas działania programu, gdy punkty badane są metodą „każdy z każdym". W tablicy 6 przedstawiono wyniki pomiarów czasu działania programu opartego na zmodyfikowanej wersji algorytmu. Czas wykonania programów w funkcji liczby generowanych na płaszczyźnie punktów dla przyjętego rozkładu przedstawiono na rys. 7.

Tablica 5

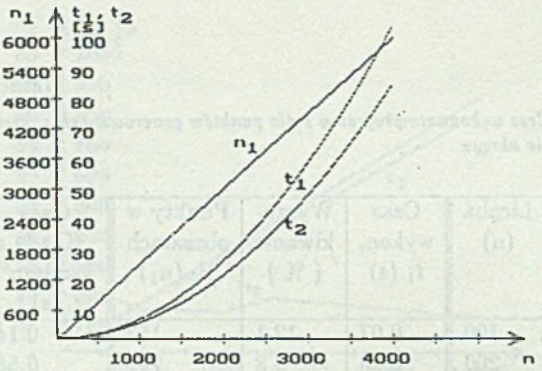
Czas wykonania programu 1 dla punktów generowanych w kształcie okręgu

Liczba (n)	Czas wykon. t_1 (s)	Wyszukiwanie (%)	Punkty w obszarach R_i (n_1)	Czas „Każdy z każdym" (s)
100	0.07	12.1	153	0.14
200	0.26	6.8	297	0.55
500	1.61	2.8	748	3.46
1000	6.40	1.4	1507	13.87
2000	25.16	0.7	2999	55.53
3000	56.04	0.5	4481	125.00
4000	100.32	0.35	5999	222.17

Tablica 6

Czas wykonania programu 2 dla punktów generowanych w kształcie okręgu

Liczba (n)	Czas wykon. t_1 (ms)	Wyszukiwanie (%)	Iloczyn punktów (n_2)	Zysk (strata) (%)
100	0.6	19.0	561	+ 12.2
200	0.23	10.5	1661	+ 11.5
500	1.36	4.5	11339	+ 15.1
1000	5.43	2.3	45331	+ 15.2
2000	21.35	1.1	177811	+ 15.2
3000	48.00	0.8	366549	+ 14.4
4000	86.61	0.6	607902	+ 13.7



Rys. 7. Czas wykonania programów dla punktów rozmieszczonych na okręgu
Fig. 7. Program execution times for circular arrangement of points

W tym eksperymencie można zaobserwować:

- niezadowalające wyniki działania programów – liczba punktów w obszarach R_i jest nawet większa niż liczba badanych punktów. W tym eksperymencie nie udało się odrzucić żadnego z badanych punktów,
- zależność czasu działania programów jest proporcjonalna do kwadratu wzrostu liczby generowanych na płaszczyźnie punktów (chociaż i tak czas wykonania programów jest około trzykrotnie krótszy niż w przypadku metody „każdy z każdym" – spowodowane jest to faktem, że w stosowanej metodzie nie ma potrzeby badać punktów sąsiadujących ze sobą),
- efektywność stosowania programu opartego na zmodyfikowanej wersji algorytmu – oszczędność czasu wykonania programu dochodzi do kilkunastu sekund,
- liczba punktów należących do obszarów R_i wzrasta proporcjonalnie do liczby generowanych punktów.

3.2.4. Eksperyment 4

W tym eksperymencie współrzędną x generowano na płaszczyźnie niezależnie, zgodnie z rozkładem równomiernym w przedziale $(-500, 500)$, a współrzędną y obliczano zgodnie z wzorami:
$y = \sqrt{b^2 - \frac{b^2 - x^2}{a^2}}$, $a = 500$, $b = 200$ (połowa punktów),
$y = -\sqrt{b^2 - \frac{b^2 - x^2}{a^2}}$, $a = 500$, $b = 200$ (połowa punktów),
dzięki temu powstał obrys elipsy o osiach: wielkiej $a = 500$ i małej $b = 200$, oraz środku w punkcie $(0, 0)$ układu współrzędnych.

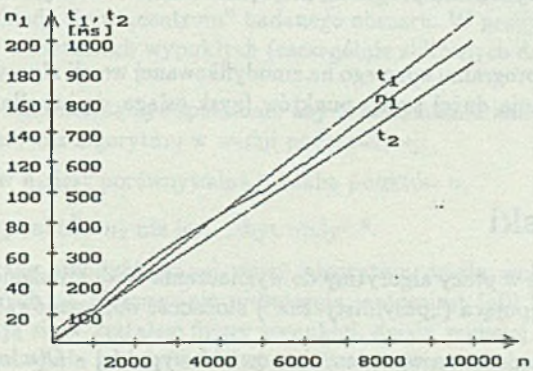
W tablicy 7 przedstawiono wyniki pomiarów czasu wykonania programu w przypadku, gdy program oparty był na podstawowej wersji algorytmu.

Tablica 7

Czas wykonania programu 1 dla punktów generowanych w kształcie elipsy

Liczba (n)	Czas wykon. t_1 (ms)	Wyszukiwanie (%)	Punkty w obszarach R_i (n_1)
100	9.1	98.8	6.3
200	18.1	98.9	8.3
500	45.4	98.1	17.2
1000	91.3	97.4	29.3
2000	184.5	96.4	48.2
3000	276.8	96.3	60.3
4000	377.9	94.1	89.8
5000	479.4	92.7	112.2
6000	575.9	92.6	123.2
8000	800.5	88.9	180.0
10000	1015.4	87.5	213.6

W tablicy 8 przedstawiono wyniki pomiarów czasu wykonania programu w przypadku, gdy program opierał się na zmodyfikowanej wersji algorytmu. Czas wykonania programów w funkcji liczby generowanych na płaszczyźnie punktów dla przyjętego rozkładu przedstawiono na rys. 8.



Rys. 8. Czas wykonania programów dla punktów rozmieszczonych na elipsie
Fig. 8. Program execution times for elliptical arrangement of points

Eksperyment wykazał:

- liniową zależność czasu wykonania programów od liczby generowanych na płaszczyźnie punktów,

Tablica 8

Czas wykonania programu 2 dla punktów generowanych w kształcie elipsy

Liczba (n)	Czas wykon. t_1 (ms)	Wyszukiwanie (%)	Iloczyn punktów (n_2)	Zysk (strata) (%)
100	9.2	99.3	1.7	- 1.1
200	18.2	99.3	2.8	- 0.7
500	45.5	98.7	11	- 0.3
1000	91.3	98.2	31	+ 0.04
2000	183.9	97.3	77	+ 0.3
3000	275.8	97.2	106	+ 0.4
4000	374.0	95.6	222	+ 1.0
5000	472.7	94.6	415	+ 1.4
6000	567.1	94.5	520	+ 1.5
8000	782.0	91.5	1014	+ 2.3
10000	987.1	90.5	1488	+ 2.8

- zależność liczby punktów w obszarach R_i od liczby generowanych punktów dla małej próby punktów (tzn. do około 2000 punktów) jest zależnością typu pierwiastkowego, natomiast dla dużej próby punktów (powyżej 4000) jest to zależność prawie liniowa. Fakt ten sugeruje, że dla bardzo dużej próby punktów – rzędu kilkudziesięciu tysięcy, czas wykonania programu nie będzie liniowo zależny od liczby generowanych punktów [4],

- stosowanie programu opartego na zmodyfikowanej wersji algorytmu ma sens w przypadku badania dużej próby punktów (zysk osiąga się już dla próby rzędu tysiąca punktów).

4. Wnioski

Przedstawione w pracy algorytmy do wyznaczania średnicy skończonego zbioru punktów cechuje następująca („pesymistyczna") złożoność obliczeniowa:

1. Algorytmy oparte na wyznaczeniu powłoki wypukłej – $O(n \log n)$ operacji.

2. Algorytmy oparte na redukcji przestrzeni poszukiwań – $O(\frac{n^2}{2})$ operacji.

Część obliczeniową algorytmu opartego na redukcji przestrzeni poszukiwań można podzielić na dwa etapy:

[4]Odległości pomiędzy punktami należącymi do obszarów R_i badane są metodą „każdy z każdym", dlatego liniowy wzrost liczby punktów należących do obszarów R_i powoduje wzrost liczby obliczeń odległości zgodnie z kwadratem wzrostu liczby tych punktów.

- „wyszukiwanie", czyli czynności zmierzające do wyznaczenia wierzchołków prostokąta opisanego na badanej figurze oraz przyporządkowanie punktów do regionów T lub R_i. Wymienione tutaj czynności polegają na sprawdzeniu, czy dany punkt P spełnia określone warunki i złożoność tej części jest liniowo zależna od liczby badanych punktów oraz niezależna od ich rozkładu na płaszczyźnie,
- „porównywanie odległości pomiędzy badanymi punktami". Ponieważ punkty są badane metodą każdy z regionu R_i z każdym z regionu R_{i+2}, złożoność tej części wynosi $O(n^2)$ operacji.

Oznaczono przez:

- n – całkowita liczba generowanych punktów,
- n_1 – liczba punktów należących do obszarów R_i,
- n_2 – iloczyn liczby punktów $n_1^- * n_1^-$, gdzie: n_1^- są to punkty P_i, n_1^- są to punkty Q_j spełniające warunek: $P_i \in R_{21}$ i $Q_j \in R_{43}$ lub $P_i \in R_{23}$ i $Q_j \in R_{41}$.

Można pokazać, że spełnienie warunku, aby całkowita złożoność obliczeniowa była rzędu n, zachodzi, jeśli:

- liczba obliczeń związanych z „wyszukiwaniem" jest znacznie większa niż liczba obliczeń związanych z wyznaczeniem odległości,
- liczba punktów n_1 „wolno" wzrasta wraz ze wzrostem liczby generowanych punktów. Teoretycznie, aby czas działania programu rósł liniowo wraz ze wzrostem liczby generowanych punktów, liczba punktów w obszarach R_i powinna spełniać warunek [5]: $n_1 \leq 2 * \sqrt{n}$.

Powyższe warunki spełnione są, jeśli punkty są generowane na płaszczyźnie niezależnie i większość z nich układa się w „centrum" badanego obszaru. W przypadku badania rzeczywistych obrazów o kształtach wypukłych (szczególnie zbliżonych do okręgu), warunki te nie są spełnione.

Następujące warunki muszą być spełnione, aby czas działania zmodyfikowanej wersji algorytmu był krótszy niż algorytmu w wersji podstawowej:

- liczba punktów n_1 jest porównywalna z liczbą punktów n,
- iloczyn liczby punktów n_2 nie jest „zbyt mały" [6].

Program oparty na zmodyfikowanej wersji algorytmu działa wolniej w przypadku punktów generowanych na płaszczyźnie niezależnie, natomiast jeśli badane punkty na płaszczyźnie układają się w kształcie figury wypukłej, działa szybciej. Należy zauważyć, że w przypadku stosowania zmodyfikowanej wersji algorytmu tracimy tylko na wyszukiwaniu (strata jest liniowo związana z liczbą generowanych punktów i nie przekracza milisekundy). Natomiast zysk dotyczy liczby obliczeń odległości i jest proporcjonalny do kwadratu liczby punktów należących do obszarów R_i (co w praktyce oznacza nawet kilka sekund).

[5] Zależność jest prawdziwa tylko w przypadku równomiernego rozkładu punktów w regionach R_i.

[6] Określenie „zbyt mała" trudno jednoznacznie zdefiniować – w praktyce oznacza to taką liczbę punktów, aby czas zyskiwany na obliczaniu odległości był większy niż czas tracony na „wyszukiwaniu".

LITERATURA

[1] Binay K. Bhattacharya, Godfried T. Toussaint. *Fast algorithms for computing the diameter of a finite planar set.* The Visual Computer (1988) 3, Springer – Verlag.

[2] Bentley J. L., Kung H. T., Schkolnick M., Tompson C. D. *On the average number of maxima in a set of vectors and applications.* J ACM (1978).

[3] Michał Jankowski. *Elementy grafiki komputerowej.* WNT, Warszawa 1990.

[4] M. Karaczyn, L. Luchowski, A. Mrózek, M. Porwoł, W. Rebajn, R. Winiarczyk. *Interaktywny system analizy obrazów mikroskopowych BIOLAR – VIDEO.* Archiwum Informatyki Teoretycznej i Stosowanej, tom 1 z 1 – 4, 1989.

[5] R. Sadłowski, A. Michalik, A. Pisulska – Otremba. *Komputerowa metoda pomiarów teleradiogramów głowy dzieci z rozszczepami podniebienia pierwotnego i wtórnego w projekcji bocznej.* Ogólnopolska konferencja naukowa radiologii stomatologicznej szczękowo – twarzowej, Opole 1991.

[6] F. Preparata, M. Shamos. *Wyczislitielnaja Gieometria: Wwiedienie.* Izdatielstwo „Mir", Moskwa 1989.

[7] R. L. Graham. *An efficient algorithm for determining the convex hull of a planar set.* Inf. Proces. Letters 1972, 1.

[8] D. E. Knuth. *The art of computer programming.* Reading, Mass., Addison–Wesley 1973.

[9] P. J. Green, B. W. Silverman. *Constructing the convex hull of a set of points in the plane.* Computer Journal 1979, 22.

[10] J. G. Hocking, G. S. Young. *Topology.* Addison–Wesley, Reading, MA, 1961.

[11] I. M. Yaglom, V. G. Boltyanskii. *Convex Figures.* Holt, Rinehart and Winston, New York 1961.

Recenzent: Dr hab. inż. Stanisław Wołek

Wpłynęło do Redakcji 26 sierpnia 1993 r.

Abstract

There exist many methods to determine the two most distant points in a given finite planar set of points. Some of them rely on the fact that the diameter of a set of points is equal to the diameter of its convex hull (Theorem 1). Methods to find the convex hull have been developed by:

- Graham (Fig. 1),
- Green and Silverman (Fig. 2).

Another group of algorithms reduce the original set by discarding some points which lie in its center. Fig. 3 illustrates one such method of discarding some points certain not to be on the convex hull.

Another algorithm is based on the principle of dividing the set of points into alternate regions. The way of finding such regions is illustrated in Fig. 4. This algorithm has been modified by further restricting the search area to the intersection of the regions.

The experimental part covers practical tests, which determined how the time required to find the diameter relates to the size of the set of points, and for what kinds of point distribution the modified algorithm is faster than the original one. Experiments 1 and 2 examined the case where the X and Y coordinates of points were generated independently, with a:

- normal distribution (Fig. 5),
- uniform distribution (Fig. 6).

Experiment 3 examined the case where the points were arranged on a circle. Relation beetwen processing time and the size of the set is illustated on Fig. 7. The case explored in Experiment 4 was a set of points arranged on an ellipse (Fig. 8).