Algorytmy i struktury danych.

Autor: Patryk Waszkiewicz

Nr albumu: 179990

Grupa: 8

Opiekun przedmiotu: Dr inż. Mariusz Borkowski

Rzeszów 12.01.2025 r.

Spis treści

1. Wstęp

W zadaniu analizujemy n-kąt, którego wierzchołki są połączone odległościami między kolejnymi punktami na obwodzie. Odległości między wierzchołkami są przechowywane w tablicy T, w której każdy element reprezentuje odległość między dwoma kolejnymi wierzchołkami. Naszym zadaniem jest znalezienie pary wierzchołków, które są od siebie najbardziej oddalone, przy czym można poruszać się tylko po obwodzie n-kąta, nie przechodząc przez jego wnętrze. Zadanie wymaga zaprojektowania algorytmu, który obliczy dystans między wszystkimi parami punktów na obwodzie i znajdzie te, które mają maksymalną odległość. Algorytm ten zostanie zaimplementowany w języku C++ z wykorzystaniem tablic oraz operacji na nich. Przykładowe dane wejściowe i wyjściowe.

Wejście:

[1 2 3 4 3]

Wyjście:

[1 4]

[3 5]

2. Teoretyczny opis zagadnienia

Wielokąt o n wierzchołkach jest opisany przez tablicę T [1..n], gdzie T[i] to odległość między wierzchołkiem i a wierzchołkiem I+1, a T[n] to odległość między wierzchołkiem n a wierzchołkiem 1. Chcemy znaleźć pary punktów, które są od siebie najbardziej oddalone. Ponieważ możemy poruszać się tylko po obwodzie n-kąta, dystans między dowolnymi dwoma wierzchołkami j i k obliczamy dwoma możliwymi drogami:

1. Zgodnie z ruchem wskazówek zegara – od wierzchołka j do k, sumując odległości między kolejnymi punktami w obwodzie.
2. Przeciwnie do ruchu wskazówek zegara - obliczając dystans odwrotną drogą.

Obliczamy odległości w obu kierunkach i wybieramy najmniejszą z nich, aby uzyskać najbardziej optymalny wynik. Naszym celem jest znaleźć parę punktów, która posiada maksymalną odległość na obwodzie.

KROK	OPIS DZIAŁANIA	ZMIENNA/OPERACJA	WARTOŚĆ/WYNIKI
1. Wczytanie danych	**Program wczytuje liczbę testów i odpowiednie tablice z pliku tablice.txt.**	**liczbaTestow, tablice[t][i], rozmiary[t]**	**Dane wejściowe: (jak w przykładzie : [1, 2, 3, 4, 3]**
2. Oblicz dystanse	**Obliczanie całkowitego obwodu i sumy dystansów zgodnie z ruchem lub przeciwnie do ruchu.**	**calkowityDystans = sum(tablice), dystansZgodnyZRuchem, dystansPrzeciwny**	**Całkowity dystans: 13**
3. Porównanie pomiarów	**Porównanie dystansów dla każdej pary punktów i zapisanie maksymalnego dystansu.**	**maksymalnyDystans, wyniki**	**Maksymalny dystans: 7**
4. Znajdź pary	**Zapisanie par punktów o maksymalnym dystansie w tablicy wyników.**	**wyniki[liczbaPar][0], wyniki[liczbaPar][1]**	**Pary punktów: [1, 4], [3, 5]**
5. Zapis wyników	**Wyniki zostają zapisane do pliku wyniki.txt**	**plikWyjsciowy**	**Plik wyniki.txt**

3. Implementacja rozwiązania

1. Wczytywanie danych wejściowych.

Na początku programu dane wejściowe są wczytywane z pliku tablice.txt. Dane te zawierają liczbę testów oraz dla każdego testu liczbę wierzchołków i tablicę odległości między kolejnymi punktami.

```
ifstream plikWejsciowy("tablice.txt");
if (!plikWejsciowy) {
   cout << "Nie udalo sie otworzyc pliku wejsciowego 'tablice.txt'!" << endl;
   return 1;
}

int liczbaTestow;
plikWejsciowy >> liczbaTestow;

int tablice[100][100];
int rozmiary[100];

for (int t = 0; t < liczbaTestow; t++) {
   plikWejsciowy >> rozmiary[t];
   for (int i = 0; i < rozmiary[t]; i++) {
      plikWejsciowy >> tablice[t][i];
   }
}
plikWejsciowy.close();
```

2. Obliczanie dystansu między dwoma wierzchołkami

Funkcja obliczDystans oblicza dystans między dwoma punktami na obwodzie, uwzględniając dwie możliwe ścieżki: zgodną z ruchem i przeciwną.

```
int obliczDystans(const int tablice[], int n, int start, int koniec, int calkowityDystans) {
   int dystansZgodnyZRuchem = 0;

   for (int i = start; i != koniec; i = (i + 1) % n) {
      dystansZgodnyZRuchem += tablice[i];
   }

   int dystansPrzeciwny = calkowityDystans - dystansZgodnyZRuchem;

   return min(dystansZgodnyZRuchem, dystansPrzeciwny);
}
```

3. Znajdowanie najbardziej oddalonych par

Funkcja znajdzNajdalszePary iteruje przez wszystkie możliwe pary wierzchołków i zapisuje te, które są najbardziej oddalone.

```
void znajdzNajdalszePary(const int tablice[], int n, int wyniki[][2], int& liczbaPar) {
  int calkowityDystans = 0;
  for (int i = 0; i < n; i++) {
    calkowityDystans += tablice[i];
  }

  int maksymalnyDystans = 0;
  liczbaPar = 0;

  for (int i = 0; i < n; i++) {
    for (int j = i + 1; j < n; j++) {
      int dystans = obliczDystans(tablice, n, i, j, calkowityDystans);

      if (dystans > maksymalnyDystans) {
        maksymalnyDystans = dystans;
        liczbaPar = 0;
        wyniki[liczbaPar][0] = i + 1;
        wyniki[liczbaPar][1] = j + 1;
        liczbaPar++;
      } else if (dystans == maksymalnyDystans) {
        wyniki[liczbaPar][0] = i + 1;
        wyniki[liczbaPar][1] = j + 1;
        liczbaPar++;
      }
    }
  }
}
```

4. Zapis wyników

Wyniki są zapisywane do pliku wyniki.txt

```
ofstream plikWyjsciowy("wyniki.txt");
if (!plikWyjsciowy) {
   cout << "Nie udalo sie otworzyc pliku wyjsciowego 'wyniki.txt'!" << endl;
   return 1;
}

for (int t = 0; t < liczbaTestow; t++) {
   int n = rozmiary[t];
   plikWyjsciowy << "Przypadek Testowy " << t + 1 << ":\n";
   plikWyjsciowy << "Dystanse: [";

   for (int i = 0; i < n; i++) {
      plikWyjsciowy << tablice[t][i] << (i < n - 1 ? ", " : "");
   }
   plikWyjsciowy << "]\n";

   znajdzNajdalszePary(tablice[t], n, wyniki, liczbaPar);

   plikWyjsciowy << "Najdalsze Pary: [";
   for (int i = 0; i < liczbaPar; i++) {
      plikWyjsciowy << "[" << wyniki[i][0] << ", " << wyniki[i][1] << "]";
      if (i < liczbaPar - 1) plikWyjsciowy << ", ";
   }
   plikWyjsciowy << "]\n";
   plikWyjsciowy << "----------------------------\n";
}

plikWyjsciowy.close();
```

Po uruchomieniu programu w konsoli widoczny jest komunikat:

1. Gdy plik się poprawnie otworzył:

```
Wyniki zostaly zapisane do pliku!

--------------------------------
Process exited after 0.2944 seconds with return value 0
Press any key to continue . . .
```

2. Gdy plik nie otworzył się poprawnie:

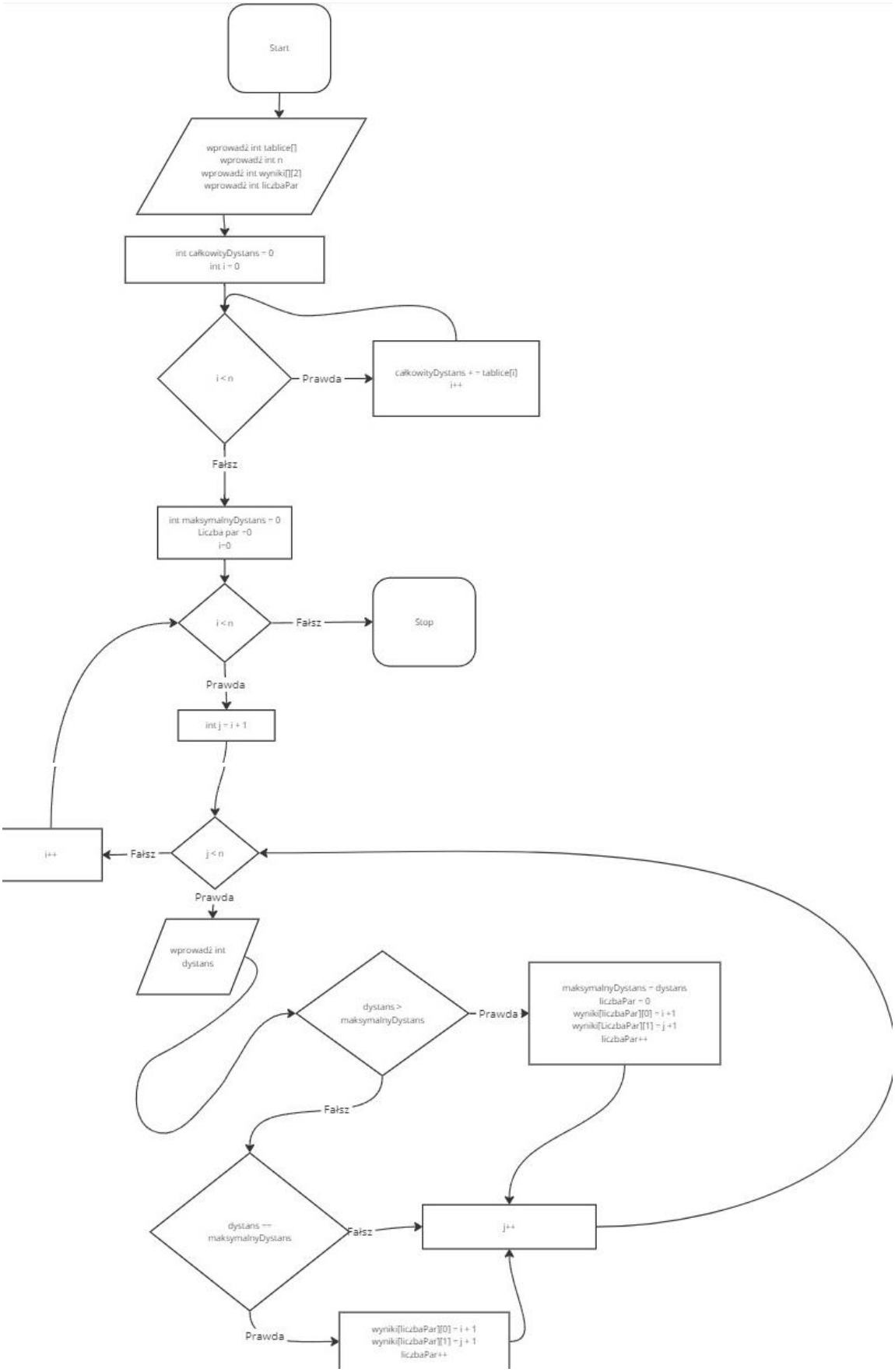
```
Nie mozna otworzyc pliku!

--------------------------------
Process exited after 0.2609 seconds with return value 1
Press any key to continue . . .
```

3. Widok pliku tekstowego po poprawnym uruuchomieniu się pliku:

```
Przypadek testowy: 1:
Dystanse: [1, 2, 3, 4, 3]
Najdalsze pary: [[1, 4], [3, 5]]
----------------------------
Przypadek testowy: 2:
Dystanse: [2, 3, 5, 7, 11]
Najdalsze pary: [[2, 5]]
----------------------------
Przypadek testowy: 3:
Dystanse: [1, 1, 1, 1, 1, 1]
Najdalsze pary: [[1, 4], [2, 5], [3, 6]]
----------------------------
Przypadek testowy: 4:
Dystanse: [10, 20, 30, 40]
Najdalsze pary: [[2, 4]]
----------------------------
Przypadek testowy: 5:
Dystanse: [1, 5, 2, 3, 8, 6]
Najdalsze pary: [[3, 6]]
----------------------------
```

4. Schemat blokowy



5. Pseudokod

```
wczytaj(plikWejsciowy)
wczytaj(tablica[n])
wczytaj(rozmiaryTestow)
wczytaj(wyniki)

inicjuj calkowityDystans ← 0
inicjuj maksymalnyDystans ← 0
inicjuj liczbaPar ← 0
inicjuj wynikiTablica[100][2]

// Obliczanie dystansow miedzy parami punktow
for i ← 0 do n - 1 powtarzaj
   for j ← i + 1 do n powtarzaj
      dystans ← obliczDystans(tablica, n, i, j, calkowityDystans)
      jeśli dystans > maksymalnyDystans to
         maksymalnyDystans ← dystans
         liczbaPar ← 0
         wynikiTablica[liczbaPar][0] ← i + 1
         wynikiTablica[liczbaPar][1] ← j + 1
         liczbaPar ← liczbaPar + 1
      w przeciwnym razie jeśli dystans = maksymalnyDystans to
         wynikiTablica[liczbaPar][0] ← i + 1
         wynikiTablica[liczbaPar][1] ← j + 1
         liczbaPar ← liczbaPar + 1

// Zapis wynikow
for t ← 0 do liczbaTestow - 1 powtarzaj
   wczytaj(rozmiarTablicy)
   wczytaj(dystanse[n])
   znajdzNajdalszePary(tablica, n, wynikiTablica, liczbaPar)
   zapisz(wynikiTablica, liczbaPar)
```

6. Testy obliczeniowe

Testy zostały wykonane na podstawie badanych przypadków.

Funkcja ObliczDystans

Funkcja oblicza dystans między dwoma punktami na podstawie tablicy dystanse. Jej złożoność wynika z iteracji między punktami start i koniec, co w najgorszym przypadku obejmuje wszystkie elementy tablicy.

ZnajdzNajdalszePary

Funkcja ta znajduje wszystkie pary punktów o maksymalnym dystansie, używając funkcji obliczDystans.

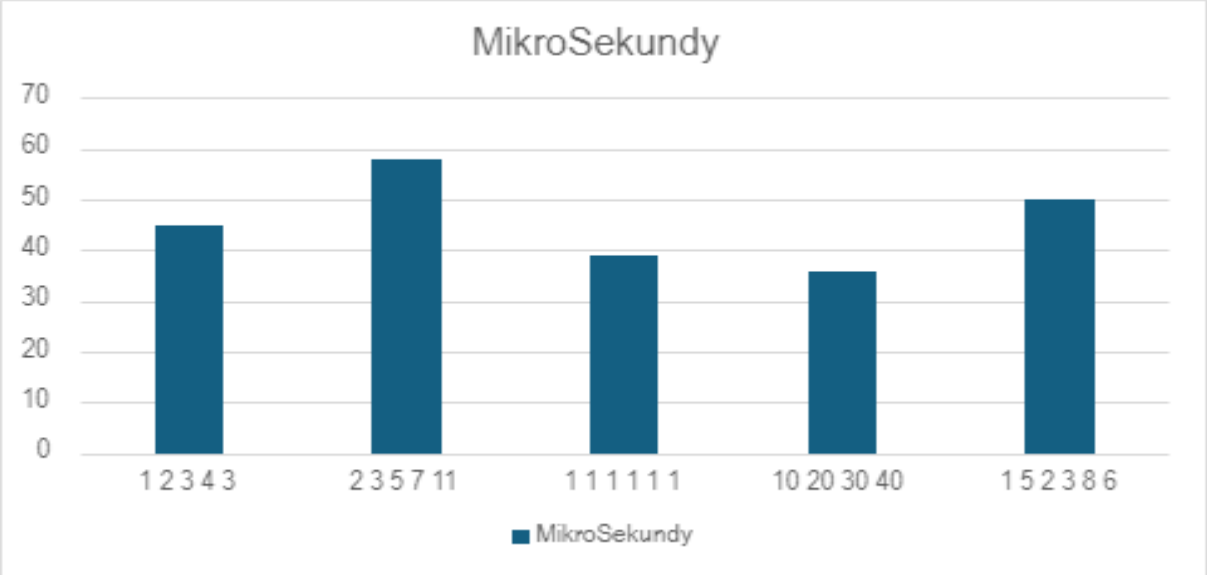
Main

W przypadku funkcji main:

- Odczytywana jest liczba punktów n oraz tablica dystansów dystanse.
- Wywoływana jest funkcja znajdzNajdalszePary
- Wyniki zapisywane są do pliku

Podsumowanie złożoności obliczeniowej

Kod charakteryzuje się złożonością czasową $O(T*n^3)$, gdzie T to liczba przypadków testowych, a n to liczba punktów w jednym przypadku testowym. Wynika to z faktu, że dla każdego przypadku testowego funkcja znajdzNajdalszePary wykonuje zagnieżdżone pętle iterujące przez wszystkie pary punktów, a obliczenia dystansu między punktami w najgorszym przypadku wymagają O(n). Złożoność pamięciowa wynosi O(n), ponieważ program przechowuje tablicę dystansów oraz zmienne pomocnicze, a tablica najdalsze Pary ma stały rozmiar niezależny od danych wejściowych.


MikroSekundy
70
60
50
40
30
20
10
0
1 2 3 4 3
2 3 5 7 11
1 1 1 1 1 1
10 20 30 40
1 5 2 3 8 6
MikroSekundy

7. Podsumowanie

Program poprawnie rozwiązuje problem znajdowania najdalszych par na okręgu, zarówno pod względem logiki, jak i efektywności. Warto rozważyć optymalizację algorytmu dla większych danych wejściowych, ponieważ czas wykonania rośnie wraz z liczbą punktów. Oszacowane wyniki testów pokazują, że wielkość przypadku testowego jest zależna od ilości kątów badanego przypadku, a także długości między następnymi wierzchołkami.