Zaprojektować i zaimplementować algorytm dokładny znajdujący najtańszą ścieżkę rozpinającą w grafie pełnym oparty o strategię dziel i zwyciężaj o złożoności $O(4^n n^{logn})$

Anna Bekas
Bartosz Woźniak

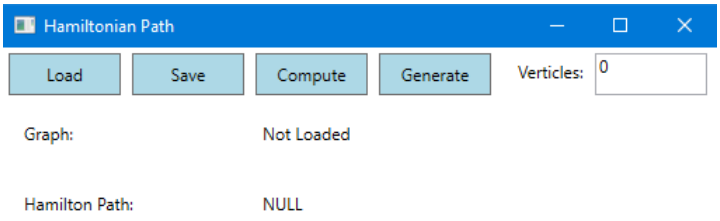
1 października 2017

1 Zmiany w stosunku do dokumentacji wstępnej

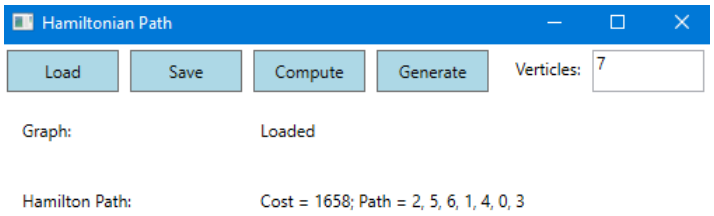
Algorytm został zaimplementowany zgodnie z założeniami opisanymi w dokumentacji wstępnej.

2 Instrukcja obsługi

2.1 Wygląd aplikacji



Rysunek 1: Wygląd aplikacji po uruchomieniu



Rysunek 2: Wygląd aplikacji po zakończeniu obliczeń

2.2 Załadowanie pliku wejściowego

W celu załadowania pliku i rozpoczęcia pracy z programem, należy wybrać przycisk *Load*, a następnie wybrać plik w formacie opisanym w sekcji *3.1*. Po poprawnym załadowaniu pliku, tekst *Not Loaded* zmieni się na *Loaded*.

2.3 Obliczenia

Wykonywanie obliczeń możliwe jest po wczytaniu pliku wejściowego oraz wybraniu przycisku *Compute*. Uzyskany wynik, wypisany zostanie w polu *Hamilton Path*.

2.4 Zapis obliczeń

Aby zapisać uzyskany rezultat, wybrać należy przycisk *Save* oraz lokalizację i nazwę pliku. Zapisywany jest graf, dla którego wykonywane były obliczenia, uzyskany koszt ścieżki oraz kolejne wierzchołki tworzące ścieżkę.

2.5 Generowanie przykładów

Wygenerowanie nowego przykładu następuje przez podanie w polu *Verticles* liczby wierzchołków z których ma składać się garf oraz wybraniu przycisku *Generate*. Następnie pojawi się okno dialogowe, gdzie należy wybrać nazwę oraz lokalizację, w której ma zostać zapisany wygenerowany przykład. Graf zapisywany jest w formacie pozwalającym na wczytanie go przez program (sekcja *2.1*).

3 Opis plików wejściowych

3.1 Reprezenacja grafu

Graf reprezentowany jest jako macierz o wymiarach nxn, gdzie n jest liczbą wierzchołków w grafie. Każdej krawędzi z wierzchołka a_i do a_j odpowiada komórka $A[i, j]$, gdzie $i, j < n$ są numerami wierzchołków grafu.
Zawartość komórki reprezentuje wagę krawędzi, czyli koszt związany z jej wyborem. Wartość 0 oznacza, że krawędź między rozważanymi wierzchołkami nie istnieje. Jednak ponieważ algorytm operuje na grafach pełnych, takie wartości występują jedynie na przekątnej macierzy. Oznacza to, że nie akceptowalne są pętle w grafie.
Algorytm bada grafy nieskierowane, w związku z czym wymaga się aby macierz była symetryczna.

3.2 Przykładowy plik

```
0 1 2 3 4
1 0 9 8 7
2 9 0 1 2
3 8 1 0 8
4 7 2 8 0
```

4 Opis wyników

4.1 Zawartość pliku

W rezultacie zapisu wyników, otrzymujemy graf (reprezentowany za pomocą macierzy) dla którego przeprowadzane były obliczenia jak i rezultat.
Wynik zawiera informacje o koszcie wygenerowanej ścieżki oraz indeksach kolejnych wierzchołków na drodze od wierzchołka startowego do końcowego.

4.2 Przykładowy plik

```
Graph:
0 1 2 3 4
1 0 9 8 7
2 9 0 1 2
3 8 1 0 8
4 7 2 8 0
Hamilton Path:
Cost = 7; Path = 1, 0, 3, 2, 4
```

5 Generowanie przykładów

Aplikacja posiada możliwość wygenerowania losowych przykładów. Dla podanej przez użytkownika liczby wierzchołków, generowany jest graf pełny wraz z wagami krawędzi. Wagi są liczbami naturalnymi takimi, że $\forall i, j, i \neq j \; 0 < A[i, j] < 1000$.
Graf zapisywany jest w pliku, którego nazwę i lokalizację ustala użytkownik. Plik jest w formacie, który pozwala na poprawne wczytanie go do programu w celu rozpoczęcia obliczeń.

6 Testy algorytmu

6.1 Opis testów wydajnościowych

W celu sprawdzenia działania algorytmu przeprowadzone zostały testy wydajnościowe. Za pomocą generatora stworzonych zostało po 10 przykładów dla grafów o liczbie wierzchołków od 2 do 15. Następnie, za pomocą zaimplementowanego algorytmu, obliczono ścieżki Hamiltona. Grafy oraz wyniki zostały zapisane do oddzielnych plików, a czasy wykonania poszczególnych przykładów do pliku *.csv*.
Uzyskane wyniki zostały porównane ze złożonością wynikającą ze specyfiki algorytmu. Jako stała, została przyjęta wartość $\alpha = 0,0001$.

6.2 Parametry sprzętowe

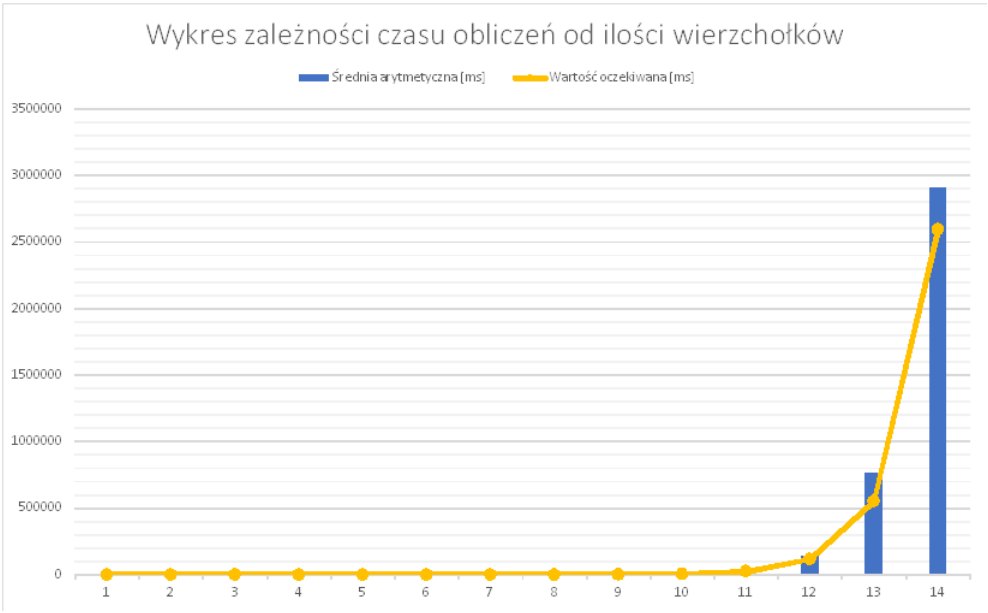
Testy prowadzone były na komputerze o następujących parametrach:

- **Procesor**: Intel Core i5-3230M 2.60GHz
- **Architektura**: x64
- **Pamięć RAM**: 8GB
- **System operacyjny**: Windows 10

6.3 Wyniki

Liczba wierzchołków	Średni czas wykonania [ms]	Oczekiwany czas wykonania [ms]
2	0	0
3	0	0
4	0	0
5	0	0
6	2	2
7	9	8
8	36	43
9	248	213
10	1284	1049
11	5702	5095
12	33815	24510
13	141481	116856
14	766743	552649
15	2907909	2594720

Tabela 1: Średni i szacowany zas wykonania obliczeń w zależności od liczby wierzchołków w grafie



Rysunek 3: Wykres zależności czasu wykonania od liczby wierzchołków w grafie

6.4 Wnioski

- Dla grafów o małej liczbie wierzchołków, czas wykonania programu jest bardzo mały. Średnią zaburzają tylko pewne anomalie.
- Wraz ze wzrostem liczby wierzchołków, wykładniczo rośnie czas wykonania algorytmu, co ilustruje wykres. Jest to przewidziane zachowanie, ze względu na oczekiwaną złożoność algorytmu.
- Dla garfów o 15 wierzchołkach, wykonanie zajęło około 40 minut, co jest spodziewanym czasem obliczeń.
- Testy wydajnościowe nie były przeprowadzane dla większej liczby wierzchołków, ze względu na słabą wydajność maszyn, na których były prowadzone.

7 Podział pracy

1. Anna Bekas
   - Dokumentacja wstępna (opis algorytmu)
   - Testy programu
   - Dokumentacja końcowa
2. Bartosz Woźniak
   - Implementacja algorytmu
   - Testy programu