Projekt ze
Projektowania Efektywnych Algorytmów

Jakub Grelowski - 262754
grupa K00-58e
środa - 11:15

23 listopada 2022

prowadzący: dr inż. Jarosław Mierzwa

Zadanie projektowe nr 1

Rozwiązanie problemu komiwojażera za pomocą przeglądu zupełnego oraz metody podziału i ograniczeń

Spis treści

1 Wstęp **3**
- 1.1 Cel projektu . . . 3
- 1.2 Sposób przechowywania grafów . . . 3
- 1.3 Testowane algorytmy . . . 3
- 1.4 Implementacja . . . 3

2 Reprezentacja grafu w komputerze **3**
- 2.1 Macierz incydencji . . . 3

3 Problem komiwojażera **3**
- 3.1 Asymetryczny problem komiwojażera . . . 3

4 Algorytmy **4**
- 4.1 Przegląd zupełny . . . 4
  - 4.1.1 Implementacja . . . 4
  - 4.1.2 Krok po kroku . . . 4
- 4.2 Metoda podziału i ograniczenia . . . 4
  - 4.2.1 Implementacja . . . 4
  - 4.2.2 Krok po kroku . . . 5
  - 4.2.3 Przykład praktyczny . . . 5

5 Pomiary czasu **7**
- 5.1 Sposób pomiaru . . . 7
- 5.2 Przebieg . . . 7

6 Wyniki pomiarów **7**
- 6.1 Metoda przeglądu zupełnego . . . 7
  - 6.1.1 Tabela . . . 7
  - 6.1.2 Wykres . . . 7
- 6.2 Metoda podziału i ograniczeń . . . 8
  - 6.2.1 Tabela . . . 8
  - 6.2.2 Wykres . . . 8
- 6.3 Porównanie . . . 8

7 Wnioski **9**

8 Literatura **9**

1 Wstęp

1.1 Cel projektu

Celem projektu było napisanie programu umożliwiającego zbadanie złożoności obliczeniowej algorytmów dokładnych rozwiązujących asymetryczny problem komiwojażera.

1.2 Sposób przechowywania grafów

- Macierz incydencji

1.3 Testowane algorytmy

- Przegląd zupełny (ang. Brute force) - złożoność $O(n!)$
- Metoda podziału i ograniczeń (ang. Branch and bound) - złożoność $O(n^2)$

1.4 Implementacja

Do wykonania projektu wykorzystany został język C# oraz środowisko programistyczne .NET Framework w wersji 4.8.

2 Reprezentacja grafu w komputerze

Istnieje kilka różnych sposobów reprezentacji grafu w pamięci komputera. W ramach projektu wykorzystana została macierz incydencji

2.1 Macierz incydencji

Macierz incydencji jest macierzą kwadratową o boku V, w której wartość i–tego wiersza i j–tej kolumny jest równa wadze krawędzi A_{ij}. Jeśli między dwoma wierzchołkami nie występuje krawędź, wartość na indeksach reprezentujących ją przyjmuje umownie ∞, która w programie jest reprezentowana przez 0 bądź -1.

3 Problem komiwojażera

Problem komiwojażera to problem obliczeniowy polegający na znalezieniu w grafie cyklu Hamiltona o najmniejszej wadze. Cykl Hamiltona to taki cykl, w którym każdy wierzchołek pojawia się dokładnie raz.

3.1 Asymetryczny problem komiwojażera

Kiedy przeszukiwany graf jest skierowany - czyli waga ścieżki z punktu A do punktu B jest taka sama jak waga ścieżki z punktu B do punktu A - mówimy wtedy o symetrycznym problemie komiwojażera. Tematem projektu jest jednak asymetryczny problem komiwojażera, czyli sytuacja przeciwna - waga ścieżki z punktu A do punktu B nie jest równa wadze ścieżki z punktu B do punktu A.

4 Algorytmy

4.1 Przegląd zupełny

Przegląd zupełny polega na sprawdzeniu każdego możliwego rozwiązania. W przypadku problemu komiwojażera rozwiązań jest tyle, ile możliwych jest różnych kombinacji wierzchołków. Tak więc jeśli mamy N wierzchołków, algorytm będzie musiał sprawdzić $N!$ różnych kombinacji tych wierzchołków, po czym wybrać ścieżkę z najmniejszą wagą.

4.1.1 Implementacja

W implementacji przeglądu zupełnego największą trudnością było generowanie nowych kombinacji. W tym celu wykorzystany został porządek leksykograficzny. W trakcie działania algorytm generuje leksykograficznie większą permutację od poprzedniej. Żeby tak zrobić, najpierw przeszukując od przedostatniego wierzchołka szukamy pierwszego wystąpienia takiego, że $A[i] < A[i+1]$. Następnie w podzbiorze po prawej szukamy pierwszej wartości mniejszej od $A[i]$. Następnie zamieniamy tą wartość z $A[i]$ i odwracamy kolejność wartości pomiędzy nimi.

4.1.2 Krok po kroku

1. Wygeneruj początkową permutację (wszystkie wierzchołki poza startowym).
2. Oblicz wartości ścieżki obecnej permutacji (uwzględniając wierzchołek startowy na początku i końcu ścieżki).
3. Jeśli obecna waga jest mniejsza od poprzedniej, zapamiętaj ją oraz ścieżkę.
4. Wygeneruj kolejną permutację.
5. Jeśli nie istnieje następna permutacja, zwróć ostatnio zapisaną ścieżkę oraz wagę. Jeśli istnieje, powtórz kroki 2-4.

4.2 Metoda podziału i ograniczenia

Metoda podziału i ograniczenia polega na analizie drzewa przestrzeni stanów. W przypadku problemu komiwojażera stany będą reprezentować wszystkie możliwe ścieżki w grafie. By algorytm był mniej kosztowny, każdy wierzchołek drzewa posiada ograniczenie, na podstawie którego można oszacować, czy jest on obiecujący.

4.2.1 Implementacja

By sprawdzić, czy przejście do konkretnego wierzchołka jest obiecujące, należy dokonać *redukcji macierzy kosztów*. Oznaczamy wszystkie wartości w wierszu o indeksie wierzchołka z którego wychodzimy oraz w kolumnie o indeksie wierzchołka do którego wchodzimy jako ∞ (czyli -1). Następnie należy w każdym wierszu macierzy znaleźć najmniejszą wartość różną od nieskończoności i potem odjąć tą wartość od każdego elementu wierszu. Tą samą czynność powtarzamy następnie dla każdej kolumny. Następnie dodajemy wcześniej znalezione najmniejsze wartości kolumn oraz wierszy. Ta operacja daje nam zredukowaną macierz oraz koszt jej redukcji. Programie stany są przechowywane w kolekcji przechowującej struktury składające się z zredukowanej macierzy, ścieżki oraz wagi tej ścieżki (koszty redukcji, przejścia oraz poprzednich redukcji).

4.2.2 Krok po kroku

1. Wstępna redukcja macierzy.
2. Wyznaczenie potencjalnych kolejnych wierzchołków.
3. Dla każdego potencjalnego wierzchołka:
   (a) Redukcja macierzy kosztów.
   (b) Obliczenie kosztu przejścia do następnego wierzchołka.
   (c) Zapisanie stanu.
4. Znalezienie dolnej granicy.
5. Jeśli ścieżka z najmniejszą wagą ma długość końcowej ścieżki, zwróć ją. Jeżeli nie, powtórz kroki 2-4 korzystając z znalezionej granicy.

4.2.3 Przykład praktyczny

Przykładowa macierz:

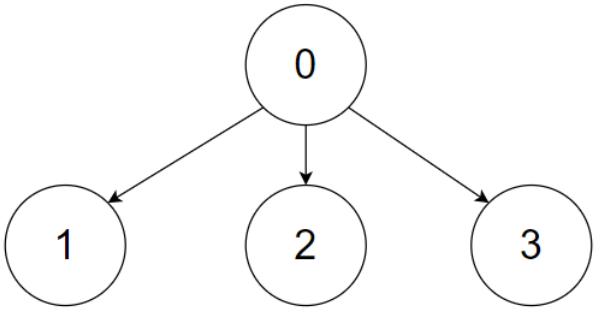
$$\begin{matrix} -1 & 2 & 5 & 3 \\ 3 & -1 & 6 & 2 \\ 1 & 4 & -1 & 4 \\ 2 & 1 & 3 & -1 \end{matrix}$$

Rozpoczynamy początkową redukcję:

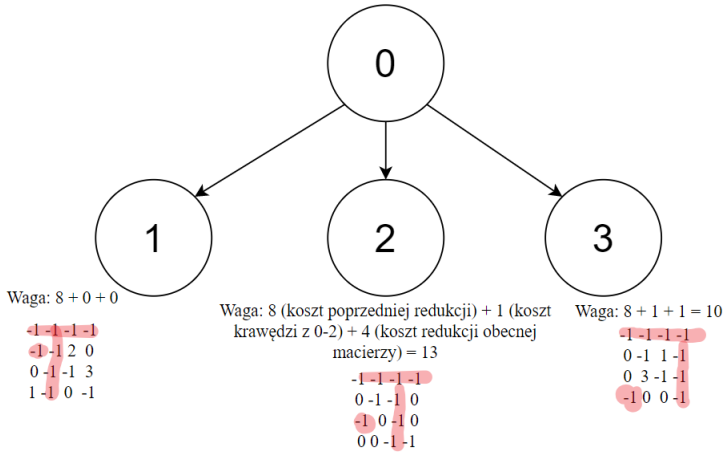
$$\begin{matrix} -1 & 2 & 5 & 3 & |(-2) \\ 3 & -1 & 6 & 2 & |(-2) \\ 1 & 4 & -1 & 4 & |(-1) \\ 2 & 1 & 3 & -1 & |(-1) \end{matrix} \qquad \begin{matrix} -1 & 0 & 3 & 1 \\ 1 & -1 & 4 & 0 \\ 0 & 3 & -1 & 3 \\ 1 & 0 & 2 & -1 \\ - & - & - & - \\ 0 & 0 & -2 & 0 \end{matrix} \qquad \begin{matrix} -1 & 0 & 1 & 1 \\ 1 & -1 & 2 & 0 \\ 0 & 3 & -1 & 3 \\ 1 & 0 & 0 & -1 \end{matrix}$$

Początkowy koszt: $2 + 2 + 1 + 1 + 2 = 8$

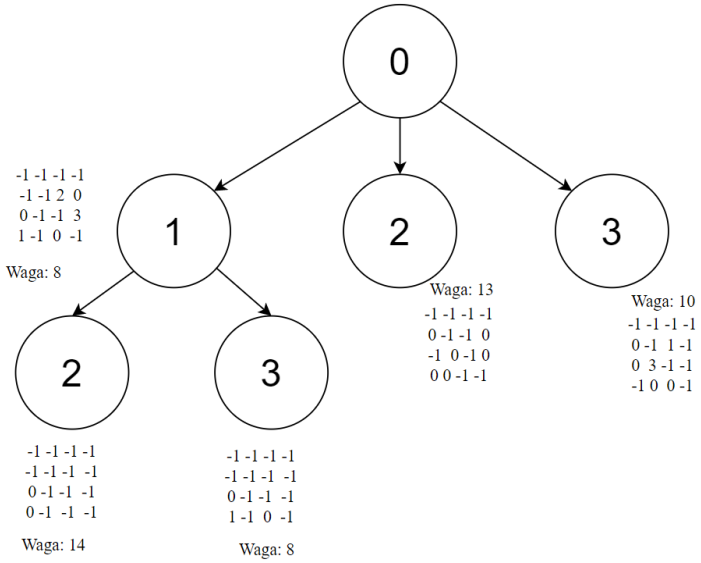
Dostępne do odwiedzenia wierzchołki:



Po zredukowaniu (na czerwono zaznaczono pozycje oznaczone jako ∞ w trakcie redukcji):



Rozwijamy macierze dla wierzchołków dostępnych dla ścieżki 0-1:



Ponieważ ścieżka 0-1-3 ma długość o jeden mniejszą niż rozmiar macierzy, oraz ma najmniejszą wagę, ostateczną ścieżką będzie 0-1-3-2 (pozostały wierzchołek) - 0(wierzchołek startowy).

5 Pomiary czasu

5.1 Sposób pomiaru

Do pomiarów czasu wykorzystana została klasa Stopwatch z biblioteki *System*. Umożliwia ona dokonania pomiarów z dokładnością do milisekundy.

5.2 Przebieg

1. Wygeneruj graf (rozmiar podany przez użytkownika).
2. Wykonaj 50 rozwiązań (bez zapisania wyniku).
3. Wykonaj 100 rozwiązań zapisując wyniki.
4. Podaj średnią z zapisanych rozwiązań

W przypadku metody Brute force dla większych macierzy liczba powtórzeń została ograniczona do 10 (bez rozgrzewki).

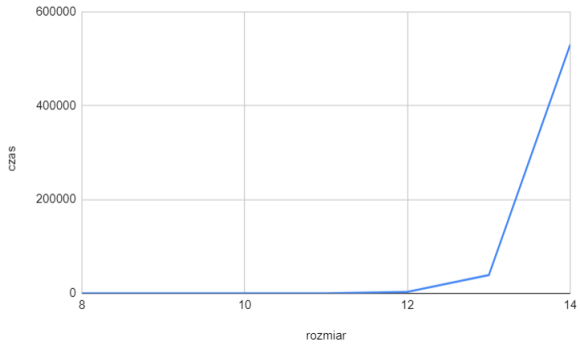
6 Wyniki pomiarów

6.1 Metoda przeglądu zupełnego

6.1.1 Tabela

rozmiar macierzy	czas [ms]
8	0,2
9	2,5
10	28
11	278
12	3192,2
13	38987,4
14	530580,8

6.1.2 Wykres

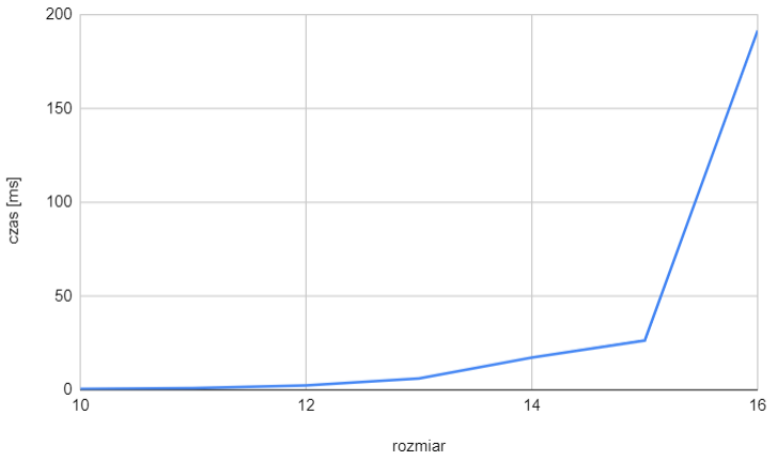


6.2 Metoda podziału i ograniczeń

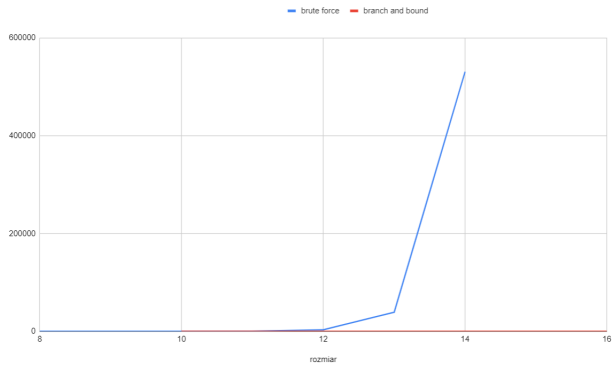
6.2.1 Tabela

rozmiar macierzy	czas [ms]
10	0,52
11	0,95
12	2,28
13	6,02
14	17,22
15	26,27
16	191,56

6.2.2 Wykres



6.3 Porównanie



7 Wnioski

Algorytmy zachowywały się zgodnie z oczekiwaniami. Ilość kombinacji do zbadania przez algorytm przeglądu zupełnego bardzo szybko wzrastała, przez co szukanie dla większych rozmiarów problemu stawało się czasochłonne. Niezależnie od instancji problemu, czas obliczeń zawsze był do siebie zbliżony, czego nie można powiedzieć o metodzie podziału i ograniczeń, która w zależnoście od danych początkowych miała bardzo duże wachania w czasie rozwiązania. Ponadto, kiedy daną wejściową był graf nieskierowany, czasy wykonania było często dużo wyższe niż dla grafów skierowanych.

8 Literatura

1. https://www.interviewbit.com/blog/next-permutation-problem/
2. https://www.youtube.com/watch?v=1FEP_sNb62k
3. https://iq.opengenus.org/travelling-salesman-branch-bound