**PROJEKTOWANIE EFEKTYWNYCH ALGORYTMÓW**

PROJEKT

07/10/2021

252736 Hutnik Szymon

Held-Karp (2)

|  |  |
| --- | --- |
| Strona | Spis treści |
| 2 | Treść zadania |
| 3 | Opis metody |
| 4 | Opis algorytmu |
| 6 | Dane testowe |
| 7 | Procedura badawcza |
| 9 | Wyniki |
| 11 | Analiza |

1. Treść zadania

Opracować, napisać, zbadać rozwiązanie problemu komiwojażera w wersji optymalizacyjnej algorytmem programowania dynamicznego.

Problem komiwojażera (*Travelling Salesman Problem*) polega na znalezieniu minimalnego cyklu Hamiltona (przejście przez wszystkie wierzchołki tylko raz, startując i kończąc w tym samym punkcie) w pełnym grafie ważonym.

1. Opis metody

Metoda programowania dynamicznego (eng. *dynamic programming*) polega na podzieleniu problemu na mniejsze części, a następnie rozwiązanie każdej z nich. Każdy następny etap wykorzystuje otrzymane wcześniej wyniki co pozwala skrócić czas obliczenia całości, bo nie ma potrzeby wykonywania kilkukrotnie tych samych obliczeń. Metoda jest trudniejsza do zaimplementowania i zazwyczaj limitem jest pamięć, a nie czas. W tym problemie sprowadza się ona do znalezienia wszystkich możliwych podścieżek, a następnie „rozbudowywaniu” ich o kolejne węzły, aż otrzymame pełny cykle i wybierzemy najtańszy z nich.

1. Opis algorytmu

Rozwiązanie zaimplementowano w postaci programu opisanego przez poniższe diagramy:

Diagram

Description automatically generated

Rysunek 1: Ogólny diagram czynności programu

Najpierw inicjalizowane są zmienne, najlepsza ścieżka jest ustawiona na maksymalną wartość, maksymanlna maska jest zerowana (później otrzyma wartość 2*n*-1). Po wczytaniu danych z konsoli następuje uruchomienie właściwej części algorytmu, następnie wypisywany jest wynik oraz czas wykonania właściwego algorytmu.

Diagram

Description automatically generated

Rysunek 2: Szczegółowy diagram czynności algorytmu

1. Dane testowe

Dane, na których była badana efektywność algorytmu pochodzą ze zbioru udostępnionego przez dr Rudego. Do badania użyto wartości z następujących plików:

* m15.astp
* gr17.tsp
* gr21.tsp
* ulysses22.stp
* gr24.stp
* fri26.stp
* bayg29.stp

http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/files/pea/instances.zip

1. Procedura badawcza

Należało zbadać zależność czasu rozwiązania problemu od wielkości instancji. W przypadku algorytmu realizującego przegląd zupełny przestrzeni rozwiązań dopuszczalnych nie występowały parametry programu, które mogły mieć wpływ na czas i jakość uzyskanego wyniku. W związku z tym procedura badawcza polegała na uruchomieniu programu i wklejeniu do niego danych z plików wybranych do badania.

Każda z instancji została wykonana do 10 razy, aby uśrednić czasy, limitem okazało się n=29, mimo usuwania poprzednich wyników zajęta pamięć (według Visual Studio 2019) przekroczyła 22GB, co poskutkowało memory overflow i niepowodzeniem testu. Wyniki były zapisywane w Excelu, następnie na ich podstawie została przeprowadzona analiza.

Pomiar czasu został zbadany przy użyciu bilbioteki chrono. Po otrzymaniu wyniku należy go podzielić przez liczbę powtórzeń wywołań algorytmu.

A screenshot of a computer

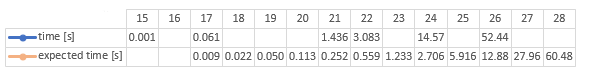
Description automatically generated with medium confidence



1. Wyniki

Wyniki opracowano w programie Excel:

Rysunek 3: Czasy działania algorytmu dla *n* wierzchołków - graf



Rysunek 4: Czasy działania algorytmu dla *n* wierzchołków - tabela

Rysunek 5: Porównanie czasu działania dotychczas opracowanych algorytmów dla *n* wierzchołków - graf

\*dane testowe nie są kolejnymi wartościami n stąd połączenie znanych wyników linią

1. Analiza

Krzywa wzrostu czasu względem wielkości instancji ma charakter wykładniczy (Rysunek 3). Nałożenie krzywej expected time ()potwierdza, że badany algorytm wyznacza rozwiązania problemu komiwojażera dla badanych instancji w czasie zależnym od wielkości instancji (obie krzywe są zgodne co do kształtu). Złożoność czasowa opracowanego algorytmu wynosi O(). Warto zwrócić uwagę, że nawet wykorzystując dynamiczne struktury danych złożoność pamięciowa skutecznie ogranicza użyteczność tego algorytmu.

Porównanie dotychczas zrealizowanych algorytmów pokazuje jak dużą przewagę ma podejście dynamiczne nad siłowym. Czas wykonania Helda-Karpa zauważalnie niższy jednak jest to okupione wysokim zużyciem pamięci.