PROJEKTOWANIE EFEKTYWNYCH ALGORYTMÓW

PROJEKT

07/10/2021

252736 Hutnik Szymon

Branch & Bound (3)

1. Treść zadania

Opracować, napisać, zbadać rozwiązanie problemu komiwojażera w wersji optymalizacyjnej algorytmem *Branch & Bound*.

Problem komiwojażera (eng. *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.

2. Opis metody

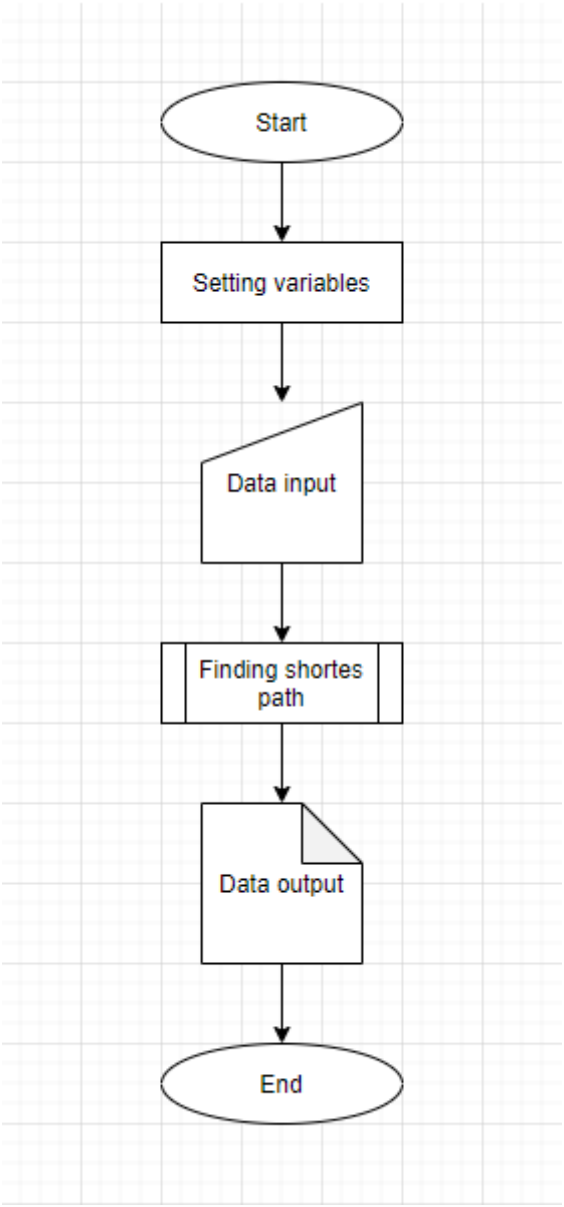
Metoda podziału i ograniczeń (eng. *Branch & Bound*) polega analizie drzewa rozwiązań, które obrazuje kazdy możliwy sposób otrzymania rozwiązania końcowego. Przejście i wygenerowanie całego drzewa jest bardzo kosztowne, więc ogranicza się je w wybrany sposób. Wyróżniamy 3 strategie przeszukujące drzewo:

- Breadth search – przeszukiwanie drzewa w szerz
- Depth search – przeszukiwanie drzewa w głąb
- Min/Max Cost – Best Search

W projekcie zbadam wariant Best Search, ponieważ pozostałe 2 są skrajnie nieoptymalne nawet w porównaniu do algorytmu Brute Force ze względu na ogromną złożoność pamięciową około $O((n-1)! * n^2)$. W przypadku wykorzystania metody Best Search nie przechodzimy po wszystkich wierzchołkach drzewa, ale korzystając z kolejki priorytetowej wybieramy najbardziej obiecującą ścieżkę. Następnie po sprawdzeniu i nowych ścieżek do kolejki ponownie wybieramy najlepszy element i powtarzamy, aż do skutku.

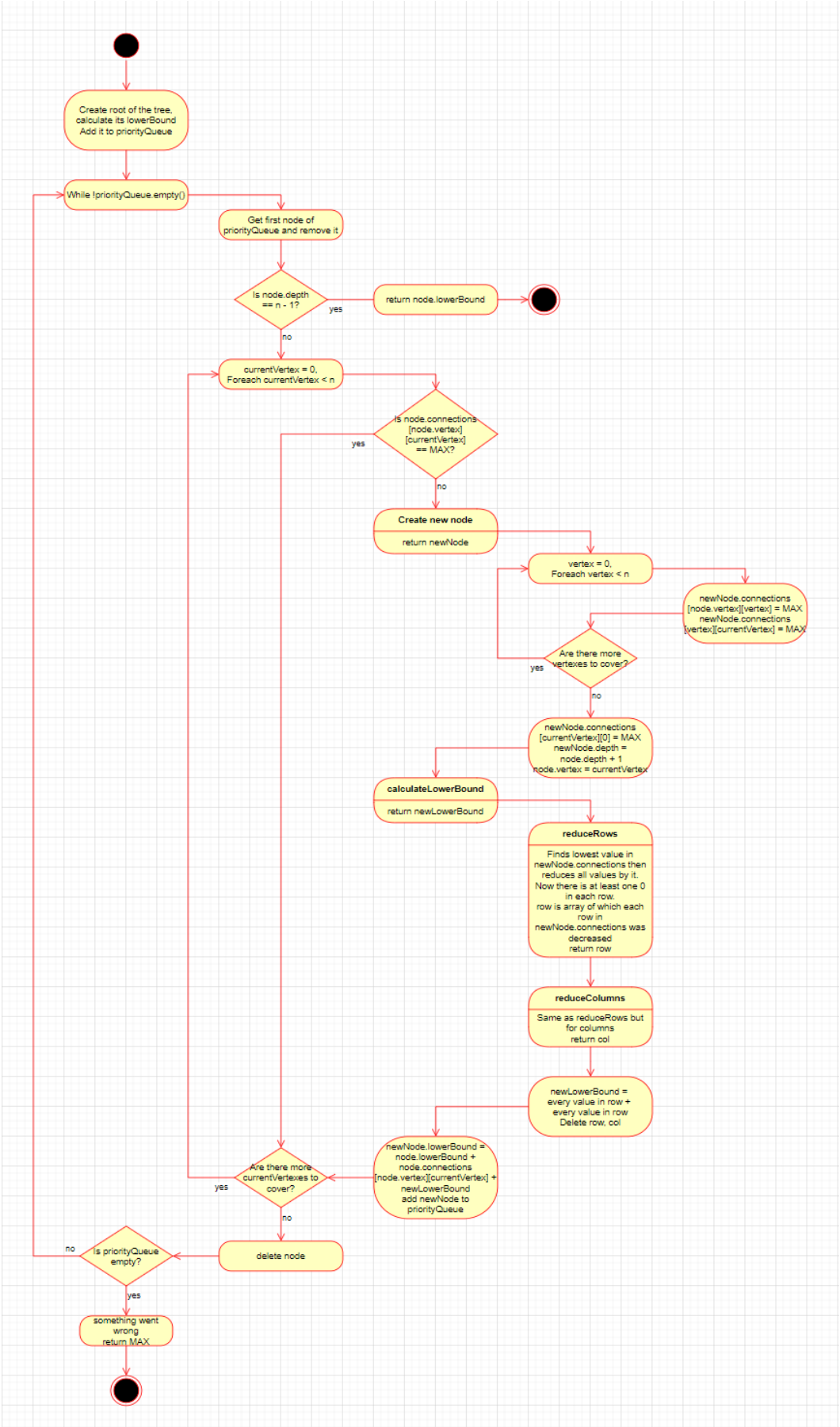
3. Opis algorytmu

Rozwiązanie zaimplementowano w postaci programu opisanego przez poniższe diagramy:



Rysunek 1: Ogólny diagram czynności programu

Najpierw inicjalizowane są zmienne, najlepsza ścieżka jest ustawiona na maksymalną wartość. Po wczytaniu danych z konsoli następuje uruchomienie właściwej części algorytmu, następnie wypisywany jest wynik oraz czas wykonania właściwego algorytmu.



Rysunek 2: Szczegółowy diagram czynności algorytmu

4. Dane testowe

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

- m14.astp
- m15.astp
- ulysses16.tsp
- gr17.tsp
- gr21.tsp
- ulysses22.tsp
- gr24.tsp

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

5. 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=22, mimo usuwania poprzednich wyników zajęta pamięć (według Visual Studio 2019) przekroczyła 19GB, test został przerwany. Wyniki były zapisywane w Excelu, następnie na ich podstawie została przeprowadzona analiza.

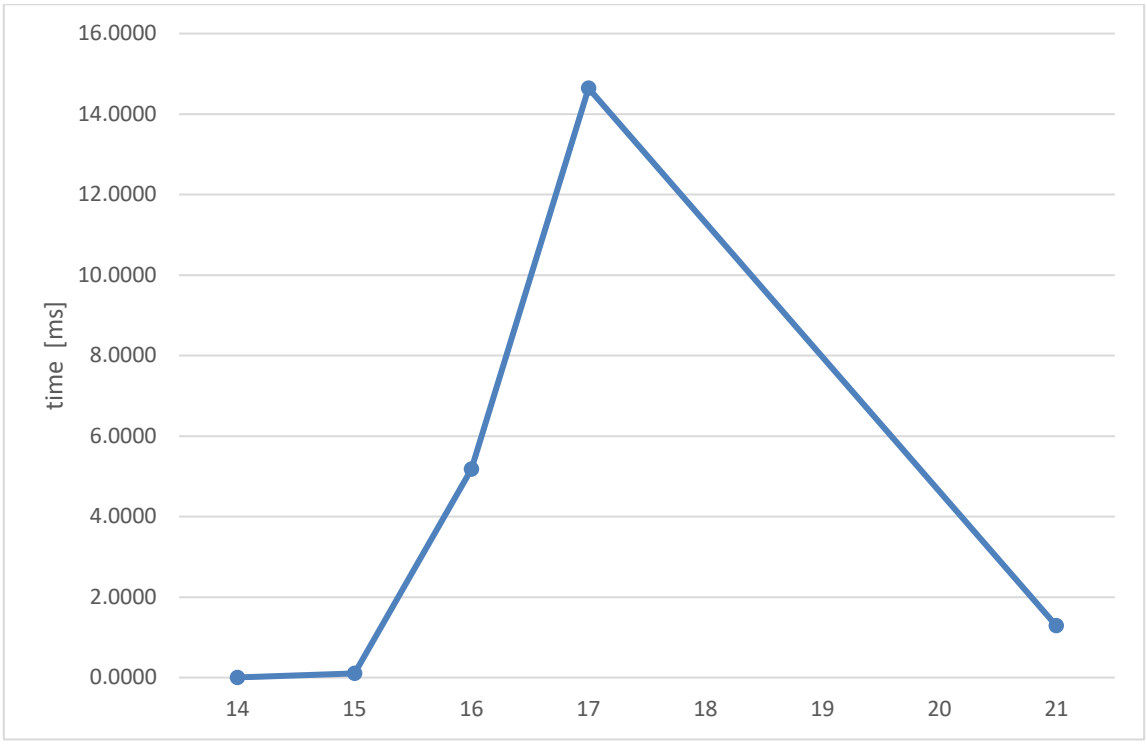
Pomiar czasu został wykonany przy użyciu bilbioteki chrono. Po otrzymaniu wyniku należy go podzielić przez liczbę powtórzeń wywołań algorytmu.

```
auto startTime = chrono::steady_clock::now();
auto resultTime = chrono::steady_clock::now() - startTime;

for (int q = 0; q < repeats; ++q)
{
    cout << "Iteration " << q << endl;
    shortest_path = SHRT_MAX;
    startTime = chrono::steady_clock::now();
    shortest_path = findShortestPath(connections);
    resultTime = resultTime + chrono::steady_clock::now() - startTime;
    while (!priorityQueue.empty())
    {
        Node* currentNode = priorityQueue.top();
        priorityQueue.pop();
        delete currentNode;
    }
}

cout << shortest_path << " - expected: " << result << endl;
cout << chrono::duration <double, milli>(resultTime).count() << "ms \n";
```

Item	Value
OS Name	Microsoft Windows 11 [redacted]
Version	10.0.22471 Build 22471
Other OS Description	Not Available
OS Manufacturer	Microsoft Corporation
System Name	SUPERCIUPERPC
System Manufacturer	Micro-Star International Co., Ltd.
System Model	MS-7C95
System Type	x64-based PC
System SKU	To be filled by O.E.M.
Processor	AMD Ryzen 5 3600 6-Core Processor, 3600 Mhz, 6 Core(s), 12 Logical Proces...
BIOS Version/Date	American Megatrends International, LLC. 2.82, 22/06/2021
SMBIOS Version	2.8
Embedded Controller Version	255.255
BIOS Mode	UEFI
BaseBoard Manufacturer	Micro-Star International Co., Ltd.
BaseBoard Product	B550M PRO-VDH (MS-7C95)
BaseBoard Version	1.0
Platform Role	Desktop
Secure Boot State	Off
PCR7 Configuration	[redacted]
Windows Directory	
System Directory	
Boot Device	
Locale	United Kingdom
Hardware Abstraction Layer	[redacted]
Username	
Time Zone	
Installed Physical Memory (RAM)	16.0 GB
Total Physical Memory	15.9 GB
Available Physical Memory	8.21 GB
Total Virtual Memory	18.3 GB
Available Virtual Memory	6.31 GB
Page File Space	2.38 GB
Page File	[redacted]
Kernel DMA Protection	Off
Virtualisation-based security	Not enabled
Device Encryption Support	Elevation Required to View
Hyper-V - VM Monitor Mode E...	Yes
Hyper-V - Second Level Addres...	Yes
Hyper-V - Virtualisation Enable...	Yes
Hyper-V - Data Execution Prote...	Yes

6. Wyniki

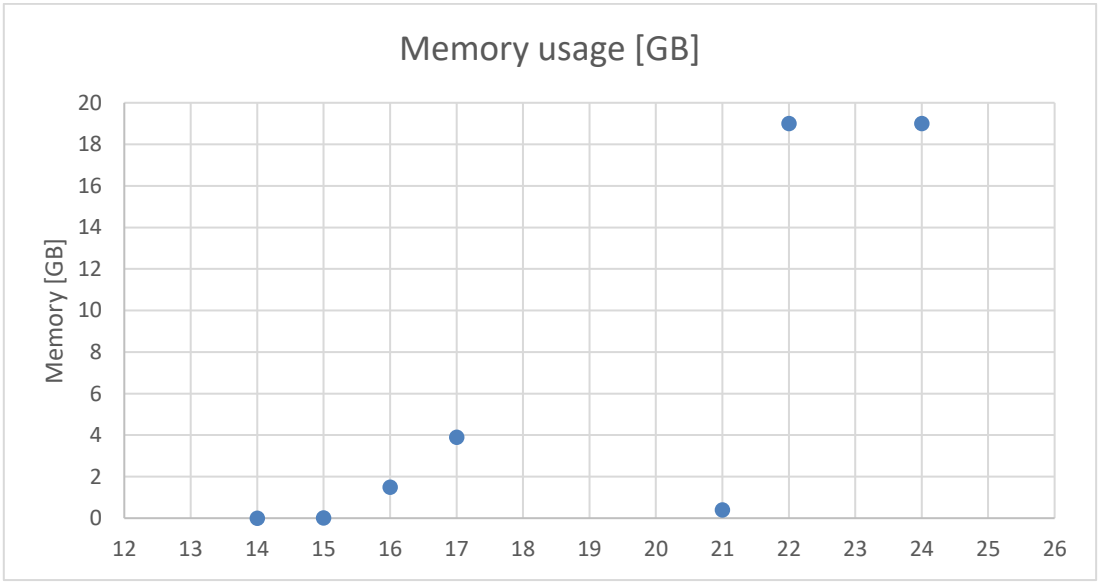
Wyniki opracowano w programie Excel:



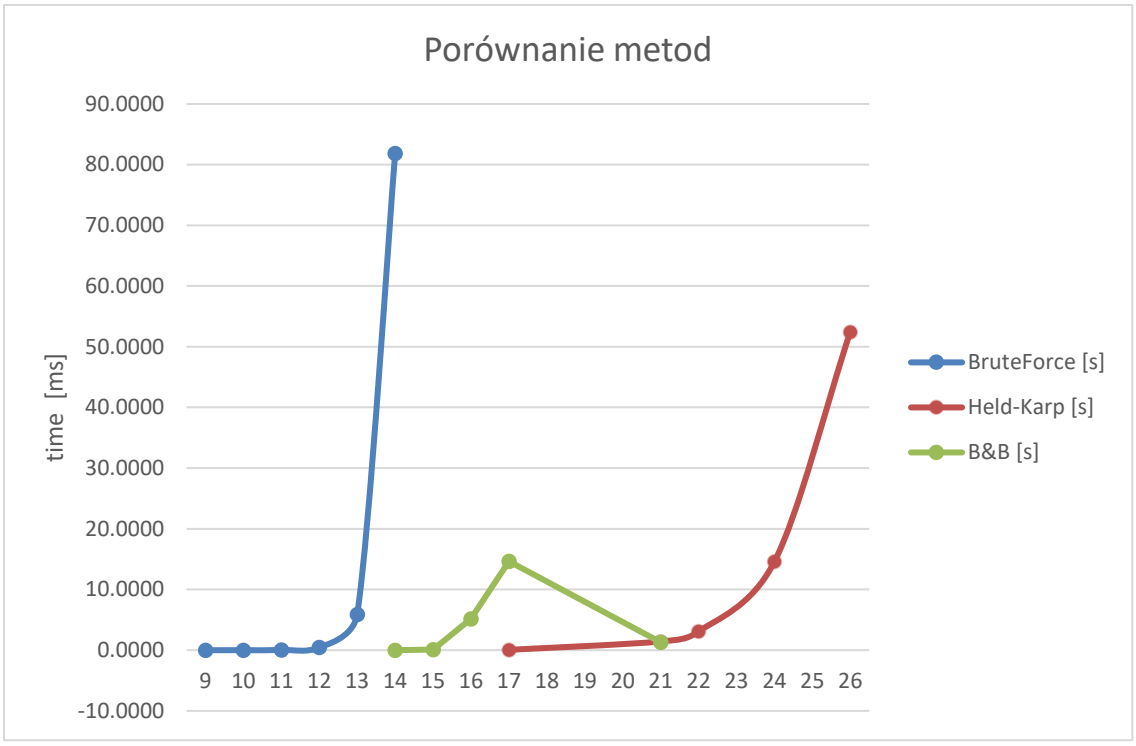
Rysunek 3: Czasy działania algorytmu dla *n* wierzchołków - graf

	14	15	16	17	18	19	20	21
time [s]	0.0030	0.1033	5.1815	14.641				1.2869

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



Rysunek 5: Zużycie pamięci algorytmu dla *n* wierzchołków - graf



Rysunek 6: Porównanie czasu działania dotychczas opracowanych algorytmów dla *n* wierzchołków - graf

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

7. Analiza

Krzywa wzrostu czasu względem wielkości instancji ma charakter losowy (Rysunek 3). Wynika to z faktu, że algorytm B&B ma złożoność czasową od O(n) do O($n!$), tak niska złożoność wynika z faktu, że jeśli jedna ścieżka będzie tańsza od kosztu wejścia do każdego innego pierwszego wierzchołka niż wybrany to zrealizujemy tylko ją i będziemy mieć pewność, że otrzymaliśmy najlepszy wynik. Widać to na wykresach, gdzie czas wykonania n=17 jest dużo wyższy niż dla n=21. Warto zwrócić uwagę, że nawet wykorzystując dynamiczne struktury danych użyteczność algorytmu jest ograniczona, ponieważ złożoność pamięciowa wynosi od O($n*n^2$) do O($n!*n^2$).

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. W przypadku B&B mamy szansę na ogromną poprawę czasu, ale również większą szansę na przekroczenie pamięci, co w moim odczuciu stawia to podejście poniżej algorytmu Helda-Karpa ze względu na jego zawodność.