Projektowanie Efektywnych Algorytmów

Projekt

15/11/2022

259126 Maciej Fras

(2) Algorytm Helda Karpa

spis treści	strona

1. Sformułowanie zadania

Zadane polega na opracowaniu, implementacji i zbadaniu efektywności algorytmu Helda-Karpa rozwiązującego problem komiwojażera w wersji optymalizacyjnej.

Problem komiwojażera (ang. Travelling salesman problem, TSP) to zagadnienie optymalizacyjne, polegające na znalezieniu minimalnego cyklu Hamiltona w pełnym grafie ważonym, gdzie:

Cykl Hamiltona – to taki cykl w grafie, w którym każdy wierzchołek grafu odwiedzany jest dokładnie raz (oprócz pierwszego wierzchołka)

Graf Ważony Pełny – struktura składająca się ze zbioru wierzchołków oraz zbioru krawędzi. Każdej krawędzi przypisana jest pewna wartość liczbowa.

Algorytm *Helda Karpa* Został opracowany przez R.E. Bellmana, R.M. Karpa oraz M. Helda w 1962 r. Wykorzystuje strategię programowania dynamicznego. Zakłada ona podział rozwiązywanego problemu mniejsze na podproblemy wraz z zachowaniem własności optymalnej podstruktury. Strategia postępowania z wykorzystaniem tej metody została szerzej opisana w *punkcie* 2, natomiast opis działania zaimplementowanego algorytmu – w *punkcie* 3.

Przy wykorzystaniu algorytmu *Helda Karpa* spodziewamy się złożoności obliczeniowej wynoszącej w przybliżeniu $O(2^n n^2)$. Prawdopodobnie przy wykorzystaniu tej metody większym ograniczeniem okaże się ilość dostępnej pamięci operacyjnej, której złożoność wynosi $O(2^n n)$.

Czasowa złożoność obliczeniowa jest niestety gorsza od wielomianowej, natomiast dużo lepsza od wariantu sprawdzającego wszystkie rozwiązania, gdzie wynosi ona O(n!). Z tego powodu możemy spodziewać się krótszych czasów wykonania algorytmu a także możliwości rozwiązania problemu dla instancji większych niż w przypadku algorytmu *brute force* w czasie do ok. 30 minut.

2. Opis metody

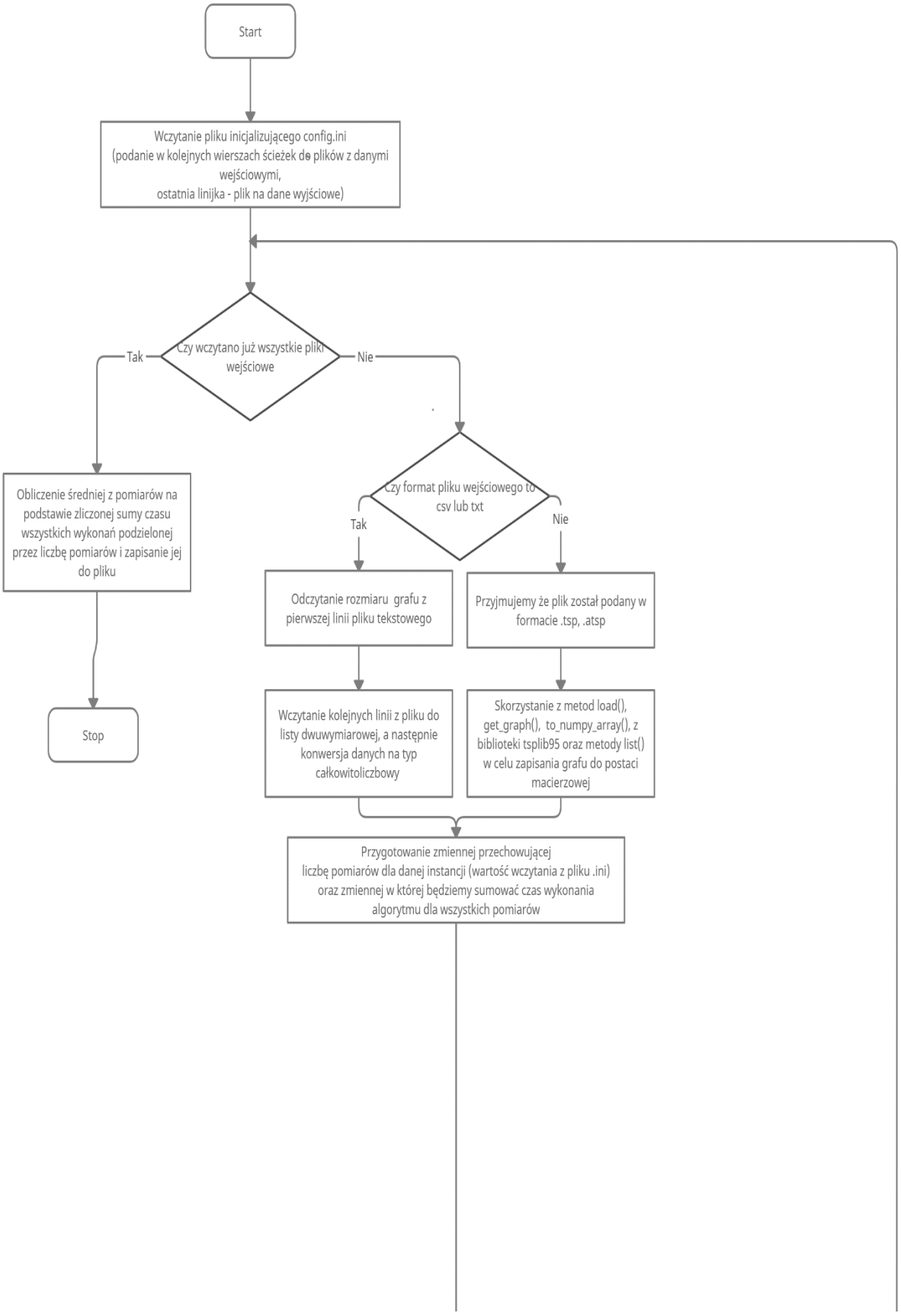
Algorytm Helda Karpa wykorzystuje metodę programowania dynamicznego. Jest ona alternatywą dla niektórych zagadnień optymalizacyjnych za pomocą algorytmów zachłannych. Zakłada ona rozwiazywanie podproblemów i zapamiętywaniu ich wyników. W technice tej, podobnie jak w metodzie dziel i zwyciężaj, problem dzielony jest na mniejsze podproblemy. Wyniki rozwiązywania podproblemów są jednak zapisywane w tabeli, dzięki czemu w przypadku natrafienia na ten sam podproblem nie trzeba go ponownie rozwiązywać. Zazwyczaj jednym z parametrów definiujących podproblemy jest liczba elementów znajdujących się w rozpatrywanym problemie, drugim jest pewna wartość liczbowa, zmieniająca się w zakresie od 0 do największej stałej występującej w problemie. Możliwe są jednak bardziej skomplikowane dobory parametrów, a także większa ich liczba. Ponieważ jednak uzyskiwany algorytm zazwyczaj wymaga pamięci (i czasu) proporcjonalnego do iloczynu maksymalnych wartości wszystkich parametrów, stosowanie większej ilości parametrów niż 3-4 rzadko bywa praktyczne.

Wykorzystując programowanie dynamiczne można zastosować *metodę zstępującą z zapamiętywaniem* lub *metodę wstępującą*.

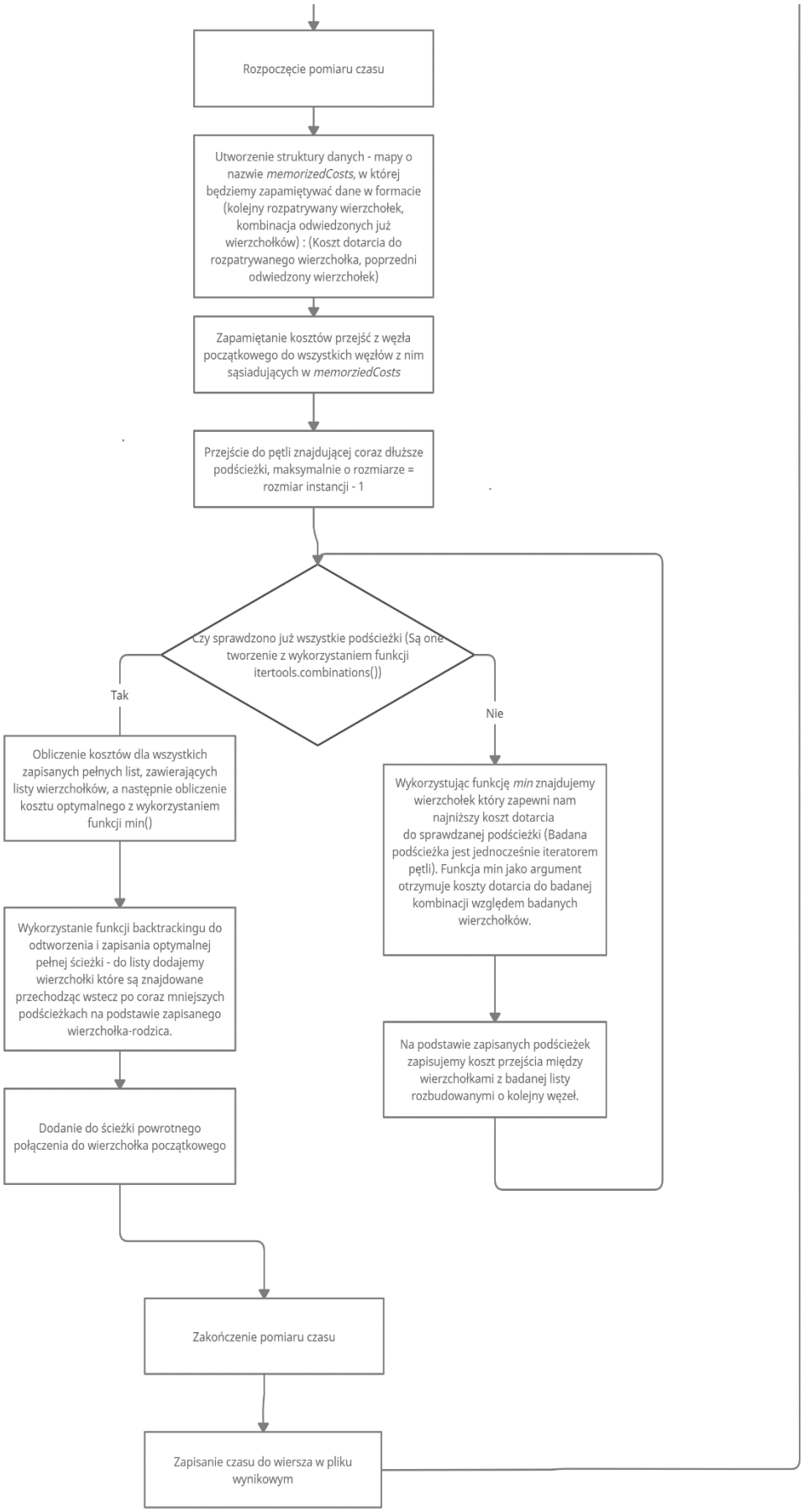
- *Metoda zstępująca z zapamiętywaniem* polega na rekurencyjnym wywoływaniu funkcji z zapamiętywaniem wyników. Metoda ta jest podobna do metody *dziel i zwyciężaj* – różni się od niej tym, że jeśli rozwiązanie danego problemu jest już w tabeli z wynikami, to należy je po prostu stamtąd odczytać.
- *Metoda wstępująca* polega na rozwiązywaniu wszystkich możliwych podproblemów, zaczynając od tych o najmniejszym rozmiarze. Wówczas w momencie rozwiązywania podproblemu na pewno są już dostępne rozwiązania jego podproblemów. W tym podejściu nie zużywa się pamięci na rekurencyjne wywołania funkcji. Może się jednak okazać, że część podproblemów została rozwiązana nadmiarowo (nie były one potrzebne do rozwiązania głównego problemu).

W przypadku zaimplementowanego algorytmu, opisywanego przez *Rysunek 2* wykorzystujemy *Metodę wstępującą*.

3. Opis algorytmu



Rysunek 1 - Pierwsza część schematu blokowego - obsługa programu i danych wejściowych



Rysunek 2 – druga część schematu blokowego – wykonanie algorytmu Helda Karpa dla problemu Komiwojażera

3. Dane testowe

Dane testowe są wczytywane zgodnie z procedurą przedstawioną na *rysunku* 1.

Do sprawdzenia poprawności działania algorytmu oraz do wykonania badań wybrano następujący zestaw instancji, dla których podano również najlepsze znane rozwiązanie:

Dane pochodzące ze strony Dr. Jarosława Mierzwy

http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl/pea-stud/tsp/

tsp_10.txt - 212
tsp_12.txt - 264
tsp_13.txt -269
tsp_15.txt – 291

Dane z biblioteki tsp-lib

http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html

burma14.tsp – 3323
gr17.tsp - 2085
gr21.tsp - 2707
gr24.tsp – 1272

Do sprawdzenia złożoności pamięciowej algorytmu wybrano następujący zestaw instancji:

http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl/pea-stud/tsp/

tsp_10.txt - 212
tsp_12.txt - 264
tsp_13.txt -269
tsp_15.txt – 291
tsp_17.txt – 39

http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html

gr21.tsp - 2707
gr24.tsp – 1272
bays29.tsp - 2020

5. Procedura badawcza

Należało zbadać zależność czasu rozwiązania problemu od wielkości instancji. W przypadku algorytmu Helda Karpa rozwiązującego problem komiwojażera nie występowały parametry programu, które mogły mieć wpływ na czas i jakość uzyskanego wyniku. W związku z tym procedura badawcza polegała na uruchomieniu programu sterowanego plikiem konfiguracyjnym *conf.ini*

Przykładowa zawartość gotowego pliku:

```
Nazwa pliku, Liczba testów, Optymalny koszt, Optymalna ścieżka
tsp_10.txt 5 212 [0 3 4 2 8 7 6 9 1 5 0]
gr24.tsp 1 1272 []
wynikiii.csv
```

Do pliku wyjściowego zapisywany był czas wykonania oraz otrzymane rozwiązanie (koszt ścieżki) oraz ścieżka (numery kolejnych węzłów). Plik wyjściowy zapisywany był w formacie txt lub csv. Poniżej przedstawiono zawartość pliku wyjściowego dla przykładowych pomiarów.

['tsp_10.txt', '5', '212', '[0', '3', '4', '2', '8', '7', '6', '9', '1', '5', '0]']

0.004224538803100586 212 [0, 3, 4, 2, 8, 7, 6, 9, 1, 5, 0]
0.002405405044555664 212 [0, 3, 4, 2, 8, 7, 6, 9, 1, 5, 0]
0.004010677337646484 212 [0, 3, 4, 2, 8, 7, 6, 9, 1, 5, 0]
0.0037081241607666016 212 [0, 3, 4, 2, 8, 7, 6, 9, 1, 5, 0]
0.003726482391357422 212 [0, 3, 4, 2, 8, 7, 6, 9, 1, 5, 0]

Uśredniony wynik pomiaru dla 5 instancji: 0.0036150455474853516
--
['gr17.tsp', '1', '1272', '[]']

1.566316843032837 2085.0 [0, 3, 12, 6, 7, 5, 16, 13, 14, 2, 10, 9, 1, 4, 8, 11, 15, 0]

Uśredniony wynik pomiaru dla 1 instancji: 1.566316843032837
--

Pomiar czasu został wykonany za pomocą funkcji time() z modułu time języka Python. Funkcja ta zwraca czas POSIX, czyli liczbę sekund, które upłynęły od 1 stycznia 1970 r. Procedura pomiaru czasu składa się z dwukrotnego wywołania tejże funkcji – przed rozpoczęciem wykonywania algorytmu oraz zaraz po jego ukończeniu. Wynikiem pomiaru czasu jest różnica tych wywołań.

Pomiar czasu odbywa się dla każdego wykonania funkcji rozwiązującej problem komiwojażera. Każdy pojedynczy wynik pomiar zapisywany jest do pliku wyjściowego. Czasy wszystkich wywołań są sumowane do jednej zmiennej, a gdy zakończy się pętla sprawdzająca jak wiele razy problem ma być rozwiązany suma czasów dzielona jest przez liczbę wywołań dzięki czemu uzyskujemy średnią z pomiarów, która zapisywana jest do wynikowego pliku .csv lub .txt.

Pomiarów dokonano z wykorzystaniem interpretera języka Python o nazwie *PyPy* w celu zwiększenia wydajności algorytmu. Interpreter ten okazał się być 5x – 10x szybszy w stosunku do domyślnego rozwiązania *cpython*.

Pomiarów dokonano na laptopie Lenovo Legion 15 o specyfikacji:

Pamięć Ram: 16GB DDR4
Procesor: Intel Core i5 8300H 4,0GHz (4 rdzenie, 8 wątków)
Dysk: SSD 512gb nvme
Karta graficzna: GeForce GTX 1050Ti

6. Wyniki

Wyniki zgromadzone zostały w plikach:

wyniki_tsp10.csv,
wyniki_tsp12.csv,
wyniki_tsp13.csv,
wyniki_burma14.csv,
wyniki_tsp15.csv,
wyniki_gr17.csv,
wyniki_gr21.csv,
wyniki_ gr24.csv

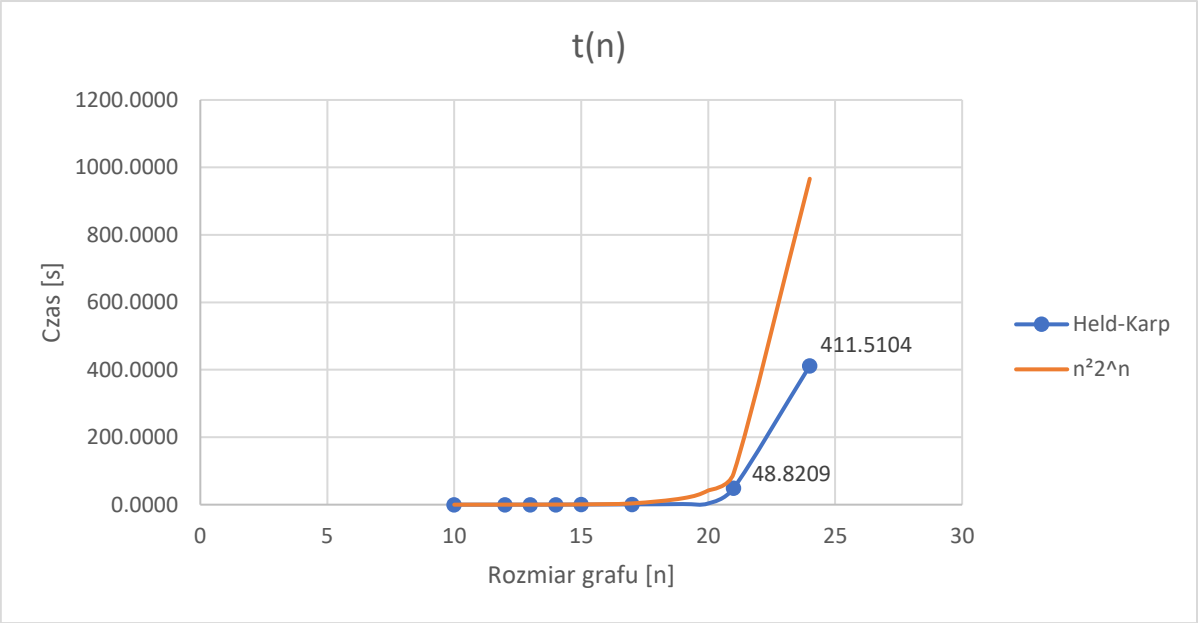
Wszystkie ww. pliki zostały dodane na eportal wraz ze źródłami.

Tabela 1 - Średnie wyniki pomiarów czasu rozwiązania problemu Komiwojażera wykorzystując algorytm Karpa-Rabina

Nazwa instancji	Wielkość instancji	Liczba pomiarów	Średni czas wykonania
tsp_10.txt	10	1000	0.0046
tsp_12.txt	12	1000	0.0252
tsp_13.txt	13	1000	0.0591
burma14.tsp	14	100	0.1345
tsp_15.txt	15	100	0.3161
gr17.tsp	17	10	0.7854
gr21.tsp	21	5	48.8208
gr24.tsp	24	1	411.5104

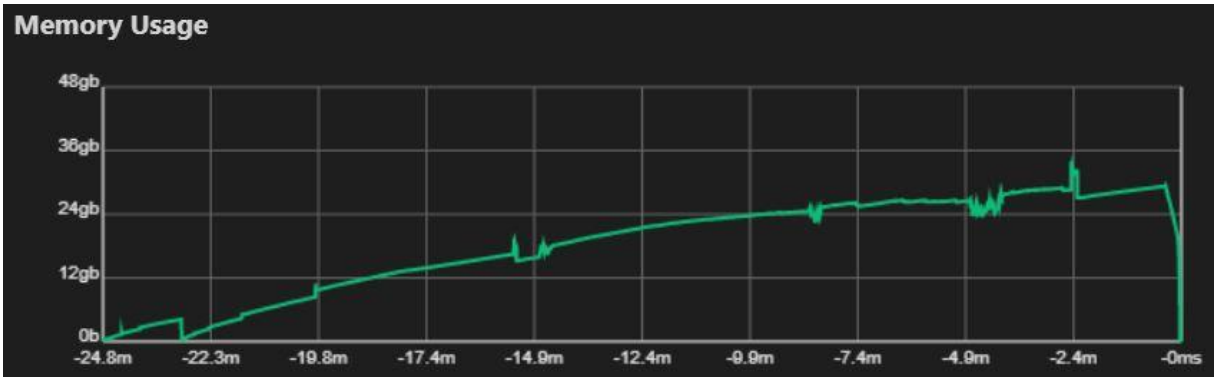
Wyniki przedstawione zostały w postaci wykresu zależności czasu uzyskania rozwiązania problemu od wielkości instancji na *rysunku 3* na podstawie danych z *tabeli 1*. W celu oceny poprawności wykonania algorytmu, obok wyników pomiaru naniesiona została również krzywa $n^2 2^n$ skorygowana o współczynnik równy 7.00E-08, tak aby móc czytelnie porównać tę zależność z otrzymanym rezultatem. Krzywa ta obrazuje teoretyczny, wzorcowy rozkład zależności czasu wykonania algorytmu od wielkości instancji.

Wyniki opracowane zostały w programie MS Excel.



Rysunek 3: Wpływ wielkości instancji na czas uzyskania rowiązania problemu komiwojażera algorytmem Karpa-Rabina oraz porównanie charakterystyki wykresu ze złożonością $O(n^2 2^n)$

7. Złożoność pamięciowa



Rysunek 4 - Wykres przedstawiający zużycie pamięci podczas wykonywania algorytmu Helda-Karpa

Rysunek 4 przedstawia zależność czasu wykonania algorytmu Helda-Karpa od wykorzystanej pamięci. Wykres powstał na podstawie pomiarów dla instancji podanych w punkcie 4. *Dane testowe*.

Czasy wykonań algorytmu podczas badania złożoności pamięciowej są wyższe niż podczas testów wydajności, ponieważ w celu stworzenia wykresu zużycia pamięci posłużyłem się wykonaniem programu w wersji debug ograniczającej wydajność programu.

Oddzielnie wykonany został pomiar dla instancji 29 elementowej. Dla takiej wielkość grafu wejściowego nie udało mi się zapisać zależności czasu od wykorzystanej pamięci. Wynika to z faktu wykorzystania całej dostępnej pamięci komputera – po czasie 2 godzin od rozpoczęcia wykonywania pomiaru doszło do zatrzymania procesu programu rozwiązującego algorytm. Wykorzystanie całej dostępnej pamięci możemy zaobserwować na *rysunku 5*

Rysunek 5 - maksymalne wykorzystanie dostępnej pamięci komputera przy wykonywaniu algorytmu dla instancji 29-elementowej

Do wykonania pomiarów wykorzystano rozszerzenie edytora *Visual Studio Code* o nazwie *Python Resources.*

8. Analiza wyników i wnioski

Krzywa wzrostu czasu względem wielkości instancji ma charakter wykładniczy (rysunek 3), jednakże wzrost czasu wykonania algorytmu metodą Helda-Karpa jest łagodniejszy niż w przypadku wykorzystania metody Brute Force, co jest zgodne z założeniami. Ograniczeniem, przez które nie udało mi się wykonać pomiarów dla większych instancji problemu okazała się pamięć, odnośnie do której szczegółowe informacje podano w *punkcie* 7. Nałożenie krzywej $n^2 2^n$ potwierdza, że badany algorytm wyznacza rozwiązania problemu komiwojażera dla badanych instancji w czasie $n^2 2^n$ zależnym względem wielkości instancji (obie krzywe są zgodne co do kształtu). Złożoność czasowa opracowanego algorytmu wynosi $O(n^2 2^n)$