Projektowanie Efektywnych Algorytmów
Projekt
7/11/2023

263974 Szymon Kluska

(2) Algorytm Helda-Karpa

Spis treści	strona

1. Sformułowanie zadania

Zadanie polega na opracowaniu, implementacji i zbadaniu efektywności algorytmu Helda-Karpa (nazywanego również algorytmem Bellmana-Helda-Karpa) rozwiązującego problem komiwojażera metodą programowania dynamicznego w wersji optymalizacyjnej.
Problem komiwojażera (eng. travelling salesman problem, TSP) to zagadnienie polegające na znalezieniu minimalnego cyklu Hamiltona w pełnym grafie ważonym. Nazwa wywodzi się z ilustracji problemu: wędrowny handlarz (komiwojażer) chce odwiedzić x miast w jak najoptymalniejszy dla niego sposób i wrócić do domu (początkowego miasta).
Cykl Hamiltona to cykl w grafie, w którym każdy wierzchołek grafu jest odwiedzany dokładnie raz oprócz wierzchołka startowego. Rozwiązanie polega na znalezieniu w grafie ścieżki o najmniejszym koszcie, która łączy wszystkie wierzchołki.
Problem komiwojażera dzieli się na symetryczny i asymetryczny. Symetryczny problem komiwojażera zakłada, że odległość między dowolnymi dwoma wierzchołkami V1 i V2 jest taka sama jak odległość między V2 i V1. Asymetryczny problem komiwojażera zakłada, że te odległości mogą być różne.

2. Metoda

Algorytm Helda-Karpa [1][5][6] jest algorytmem dokładnym (eng. exact) opartym na metodzie programowania dynamicznego. Algorytm polega na wyliczaniu wartości funkcji D(S, p) dla poszczególnych podproblemów i wyborze wartości optymalnej wedle przyjętego kryterium, która następnie jest zapamiętywana w pamięci i wykorzystywana w kolejnych etapach.
Funkcja D(S, p) to optymalna długość ścieżki wychodzącej z pierwszego miasta (wierzchołka, punktu) przechodząca przez miasta w zbiorze S i kończąca się w mieście p.

- Jeśli S.size = 1 to funkcja D(S, p) = $d_{1,p,}$ gdzie $d_{i,j}$ oznacza koszt drogi z miasta *i* do miasta *j*
- Jeśli S.size > 1 to funkcja D(S, p) = $\min_{i\in(S-\{p\})}[D(S-\{p\}, i) + d_{i,p}]$.

Złożoność czasowa
Wyznaczanie wartości funkcji D(S, p) dla |S| = 1 wyznaczana jest n-1 razy. Dla każdego podzbioru należącego do przedziału {2, 3, …, n-1} rozważane są wszystkie k-elementowe kombinacje ze zbioru n-1 elementowego, co daje wzór

$$\sum_{k=2}^{n-1} k(k-1)\binom{n-1}{k} + n - 1 = (n-1)(n-2) * 2^{n-3} + (n-1)$$

Wartość funkcji D(S, p) wyznacza się w czasie liniowym, a więc łączna złożoność obliczeniowa czasu wynosi $\boldsymbol{O(2^n n^2)}$ [4].

Złożoność pamięciowa
Przechowywanie wszystkich wartości funkcji D(S, p) dla podzbiorów o rozmiarze k wymaga zapamiętywania liczby wartości wynikającej ze wzoru:

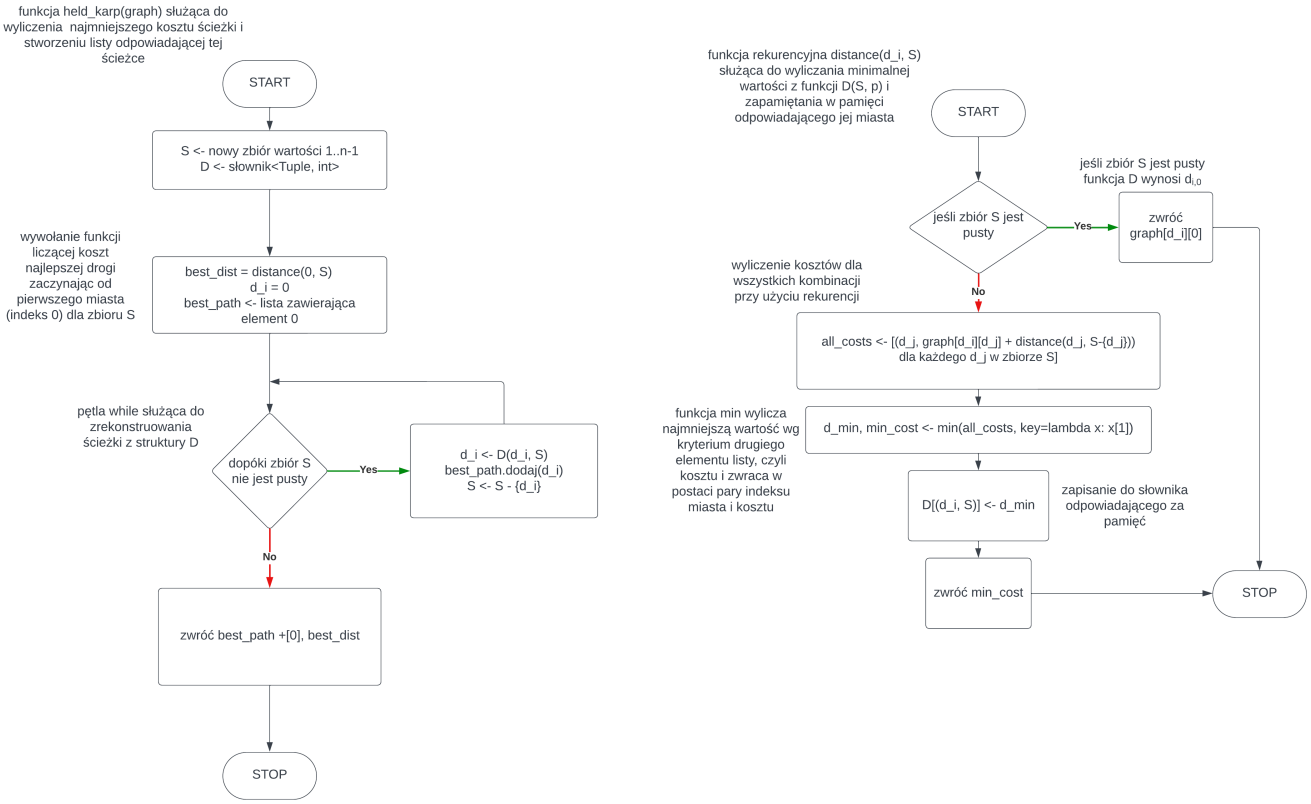
$$(n-k-1)\binom{n-1}{k} = (n-1)\binom{n-2}{k}$$

Dla wszystkich elementów w tablicy:

$$\sum_{k=0}^{n-2}(n-1)\binom{n-2}{k} = (n-1)2^n$$

Co daje złożoność pamięciową $\boldsymbol{O(n2^n)}$ [5].

3. Algorytm



Rysunek 1 Schemat blokowy algorytmu Helda-Karpa

4. Dane testowe

Do sprawdzenia poprawności działania algorytmu wybrano następujący zestaw danych instancji:

1. tsp_6_1.txt 20 132 [0, 1, 2, 3, 4, 5, 0]
2. tsp_6_2.txt 20 80 [0, 5, 1, 2, 3, 4, 0]
3. tsp_10.txt 20 212 [0, 3, 4, 2, 8, 7, 6, 9, 1, 5, 0]
4. tsp_12.txt 20 264 [0, 1, 8, 4, 6, 2, 11, 9, 7, 5, 3, 10, 0]

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

Do wykonania badań wybrano następujący zestaw instancji:

1. tsp_6_1.txt 20 132 [0, 1, 2, 3, 4, 5, 0]
2. tsp_6_2.txt 20 80 [0, 5, 1, 2, 3, 4, 0]
3. tsp_10.txt 20 212 [0, 3, 4, 2, 8, 7, 6, 9, 1, 5, 0]
4. tsp_12.txt 20 264 [0, 1, 8, 4, 6, 2, 11, 9, 7, 5, 3, 10, 0]
5. tsp_13.txt 20 269 [0, 10, 3, 5, 7, 9, 11, 2, 6, 4, 8, 1, 12, 0]
6. tsp_14.txt 20 282 [0, 10, 3, 5, 7, 9, 13, 11, 2, 6, 4, 8, 1, 12, 0]
7. tsp_15.txt 20 291 [0, 10, 3, 5, 7, 9, 13, 11, 2, 6, 4, 8, 14, 1, 12, 0]
8. tsp_17.txt 20 39 [0, 2, 13, 1, 9, 10, 12, 5, 6, 14, 15, 3, 4, 8, 16, 7, 11, 0]

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

9. gr21.txt 10 2707 [0, 6, 7, 5, 15, 4, 8, 2, 1, 20, 14, 13, 12, 17, 9, 16, 18, 19, 10, 3, 11, 0]

http://softlib.rice.edu/pub/tsplib/tsp/

5. Procedura badawcza

Należało zbadać zależność czasu rozwiązania problemu i zużycia pamięci od wielkości instancji. W przypadku algorytmu Helda-Karpa nie występowały parametry programu, które mogły wpłynąć na czas i jakość uzyskanego wyniku. Procedura badawcza polegała na uruchomieniu programu sterowanego plikiem konfiguracyjnym test_atsp.ini. Każda instancja programu, wykonująca się w czasie mniejszym niż godzina, została wykonana 20 razy.
Program został napisany w języku Python 3.10, a generowanie wyników procedury badawczej zostało przeprowadzone na MacBooku Pro o procesorze Apple M1, 16GB RAM i systemie macOS Sonoma 14.1.
Do pomiaru czasu została użyta biblioteka time [2], a do pomiaru zużycia pamięci biblioteka memory_profiler [3].

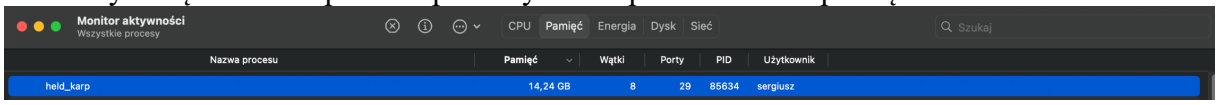
```
def perform_method(file_name: str, method: ()):
    from time import perf_counter
    graph = file_to_graph(file_name)
    start = perf_counter()
    memory_profiler.profile()
    path, dist = method(graph)
    mem_usage = memory_profiler.memory_usage()
    stop = perf_counter()
    diff = stop - start
    return path, dist, diff, mem_usage
```

Rysunek 2 Fragment kodu odpowiadający za pomiar czasu i pamięci w module pea_utils.py

Do pliku wyjściowego test_atsp_out.csv najpierw zapisywane były: nazwa pliku zestawu instancji, ilość powtórzeń, otrzymane rozwiązanie (koszt ścieżki) oraz ścieżka (numery kolejnych węzłów). Następnie program zapisywał otrzymywane wyniki: czas wykonania funkcji algorytmu [w sekundach] oraz zużycie pamięci [w MiB]. Plik wyjściowy był zapisywany w formacie csv.
Poniżej przedstawiono fragment zawartości pliku wyjściowego.

tsp_6_1.txt 20 132 [0, 1, 2, 3, 4, 5, 0]
0.10521912499098107 [45.75]
0.10557779099326581 [45.828125]
0.10515754198422655 [45.859375]
…
gr21.txt 10 2707 [0, 6, 7, 5, 15, 4, 8, 2, 1, 20, 14, 13, 12, 17, 9, 16, 18, 19, 10, 3, 11, 0]
115.13943995797308 [2706.109375]
…
134.89760229096282 [1868.09375]

Wyniki zostały opracowane w programie Microsoft Excel.
Zostało również przeprowadzone badanie dla instancji n = 24 (dla pliku gr24.txt), które zakończyło się zabiciem procesu przez system z powodu braku pamięci.



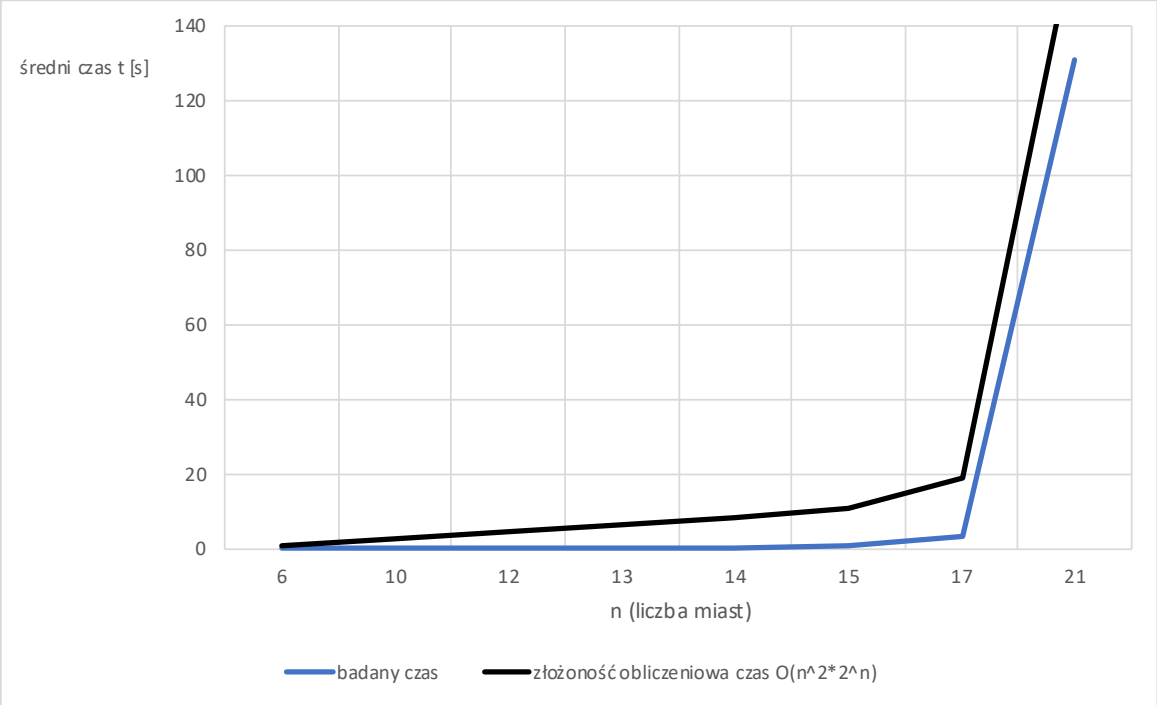
Rysunek 3 Zrzut ekranu z Monitora aktywności podczas wykonywania programu dla instancji o n = 24

6. Wyniki

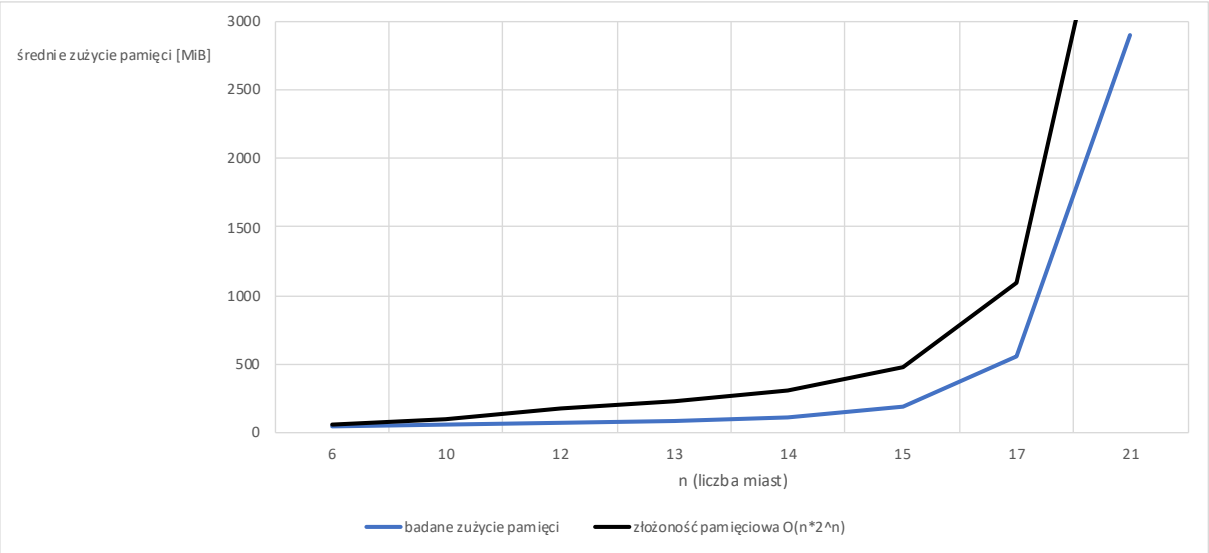
Wyniki przedstawione zostały w postaci wykresu zależności czasu uzyskania rozwiązania problemu od wielkości instancji i tabeli z dokładnymi wartościami.

Tabela 1 Wyniki badanego czasu i zużycia pamięci dla instancji n

n	średni czas [s]	średnie zużycie pamięci [MiB]
6	0,1044	46,07422
10	0,1263	52,33672
12	0,1642	67,66875
13	0,2273	81,45547
14	0,3664	111,42813
15	0,7400	193,95391
17	3,5391	561,01406
21	130,5848	2903,25156



Wykres 1 Wpływ wielkości instancji n na czas wykonania algorytmu Held-Karpa do rozwiązania problemu komiwojażera



Wykres 2 Wpływ wielkości instancji n na zużycie pamięci algorytmu Helda-Karpa do rozwiązania problemu komiwojażera

7. Analiza wyników i wnioski

Analiza wyników

Z Wykresu 1 i Tabeli 1 można odczytać, że algorytm dynamiczny Helda-Karpa wykonuje się relatywnie szybko dla badanych instancji $n < 17$. Dla $n = 21$ czas się zwiększa ponad 43-krotnie względem czasu otrzymanego dla n = 17. Krzywa o kolorze czarnym odpowiadająca za złożoność obliczeniową jest bliska pokrycia się z niebieską krzywą odpowiadającą za badany czas.

Z Wykresu 2 i Tabeli 1 można odczytać, że algorytm Helda-Karpa dla każdej kolejnej instancji n zużywa coraz więcej pamięci. Dla n = 15 zużycie osiąga prawie 200 MiB, a dla n = 21 średnio około 2900MiB, co jest równe około 3GB pamięci. Z Rysunku 3 można odczytać, że zużycie pamięci podczas uruchomienia programu dla instancji o wielkości n = 24 zużycie w trakcie wynosiło ponad 14GB pamięci RAM. Dalszy przebieg wykonywania algorytmu przekraczał 16GB przez co system musiał zabić proces.

Wnioski

Zakładając, że satysfakcjonującym wynikiem jest czas poniżej 5 sekund, algorytm Helda-Karpa jest efektywny dla instancji o $n \leq 17$. Dla instancji n $\leq$ 17 zużycie pamięci, które osiąga do 560 MiB, dla większości względnie nowoczesnych komputerów osobistych nie powinno być kryterium zniechęcającym do wykorzystania algorytmu. Algorytm Helda-Karpa oferuje względnie szybkie znalezienie optymalnego kosztu i ścieżki kosztem sporego zużycia pamięci. Dla instancji n > 17 zużycie pamięci i czas zaczynają być znacznie większe przez co algorytm Helda-Karpa może nie być najlepszym rozwiązaniem problemu komiwojażera.

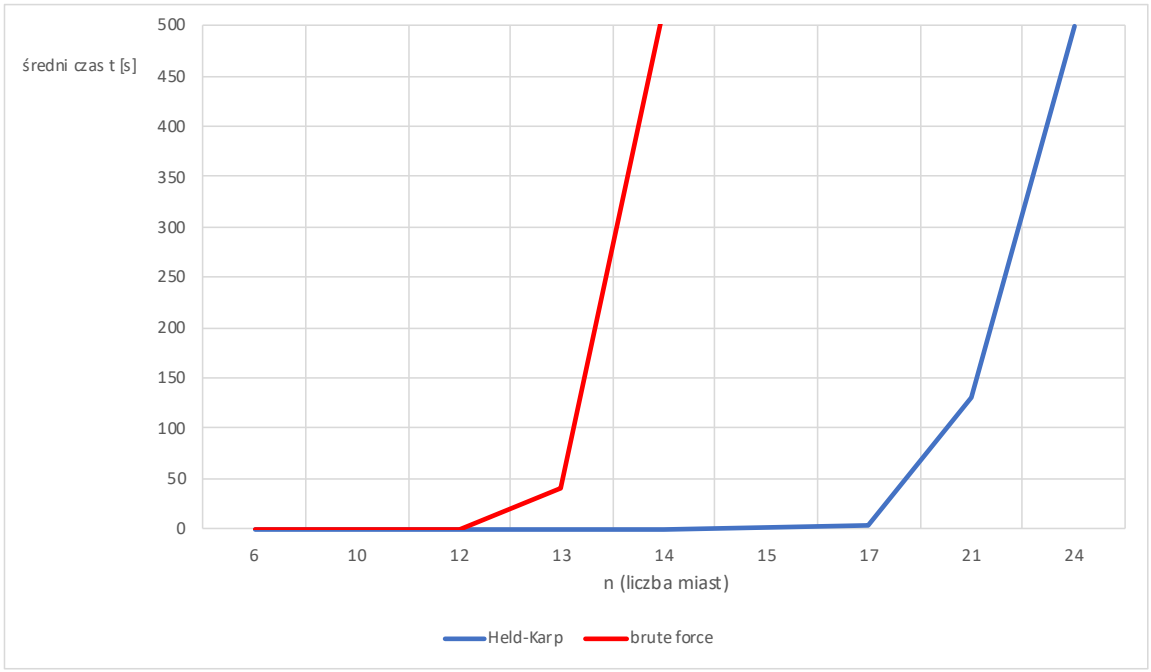
8. Źródła

[1] https://www.math.nagoya-u.ac.jp/~richard/teaching/s2020/Quang1.pdf
[2] https://docs.python.org/3/library/time.html
[3] https://pypi.org/project/memory-profiler/
[4] https://scholarworks.calstate.edu/downloads/xg94hr81q
[5] https://en.wikipedia.org/wiki/Held–Karp_algorithm
[6] http://algorytmy.ency.pl/artykul/algorytm_helda_karpa

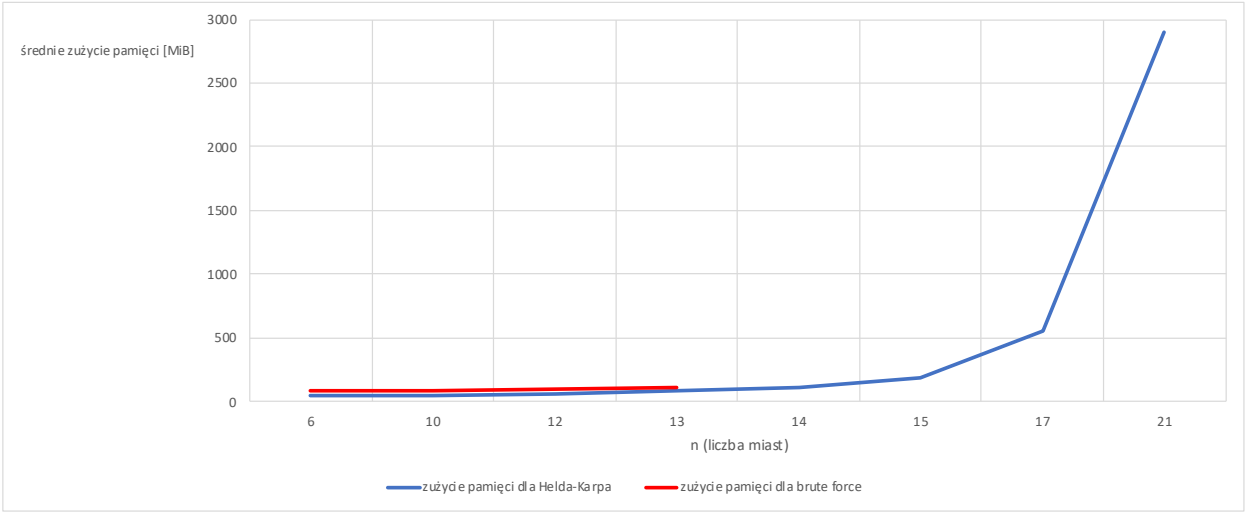
Dodatek A
Porównanie algorytmów brute force i Held-Karpa

Tabela A.1 Porównanie średnich wyników algorytmu Held-Karpa i brute force

	ALGORYTM HELDA-KARPA		ALGORYTM BRUTE FORCE	
n	średni czas [s]	średnie zużycie pamięci [MiB]	średni czas [s]	średnie zużycie pamięci [MiB]
6	0,10440424	46,07422	0,000092	35,60673828
10	0,12625138	52,33672	0,000082	35,61796875
12	0,16419182	67,66875	0,323168	35,37285156
13	0,22732475	81,45547	40,347227	35,12890625
14	0,36638765	111,42813	516,349511	
15	0,73998868	193,95391	7098,565880	
17	3,53910684	561,01406		
21	130,58478244	2903,25156		



Wykres A.1 Porównanie czasu wykonania algorytmu brute force z algorytmem Held-Karpa



Wykres A.2 Porównanie zużycia pamięci algorytmu brute force z algorytmem Held-Karpa

Analiza i wnioski
Z Tabeli A.1 można wywnioskować, że algorytm brute force jest efektywniejszy dla instancji $n \leq 10$. Z Wykresu A.1 można odczytać, że algorytm brute force dla instancji n > 12 osiąga o wiele gorsze czasy od algorytmu Helda-Karpa. Z wykresu A.2 można odczytać, że algorytm brute force ma liniowe zużycie pamięci, zgodnie z jego złożonością pamięciową O(n). Zgodnie z danym w Tabeli A.1, dla instancji n = {6, 10, 12, 13} zużycie pamięci wynosi stałe około 35 MiB. Zużycie pamięci brute force jest znacznie mniejsze od zużycia pamięci algorytmu Helda-Karpa.

Algorytm brute force będzie lepszym wyborem od algorytmu Helda-Karpa dla problemu komiwojażera w przypadkach, gdy:

- Liczba miast wynosi $n \leq 10$ (mniejszy średni czas wykonania)
- Zasoby pamięci są ograniczone i ważniejsze jest mniejsze zużycie pamięci niż czas wykonania (przy czym górna granica wynosi n = 17)

W pozostałych przypadkach algorytm Helda-Karpa będzie optymalniejszy niż algorytm brute force.