Projektowanie efektywnych algorytmów
Projekt

17/11/2021

248881 Mikołaj Szymczyk

(2) Held Karp

Spis treści

1 Sformułowanie zadania

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

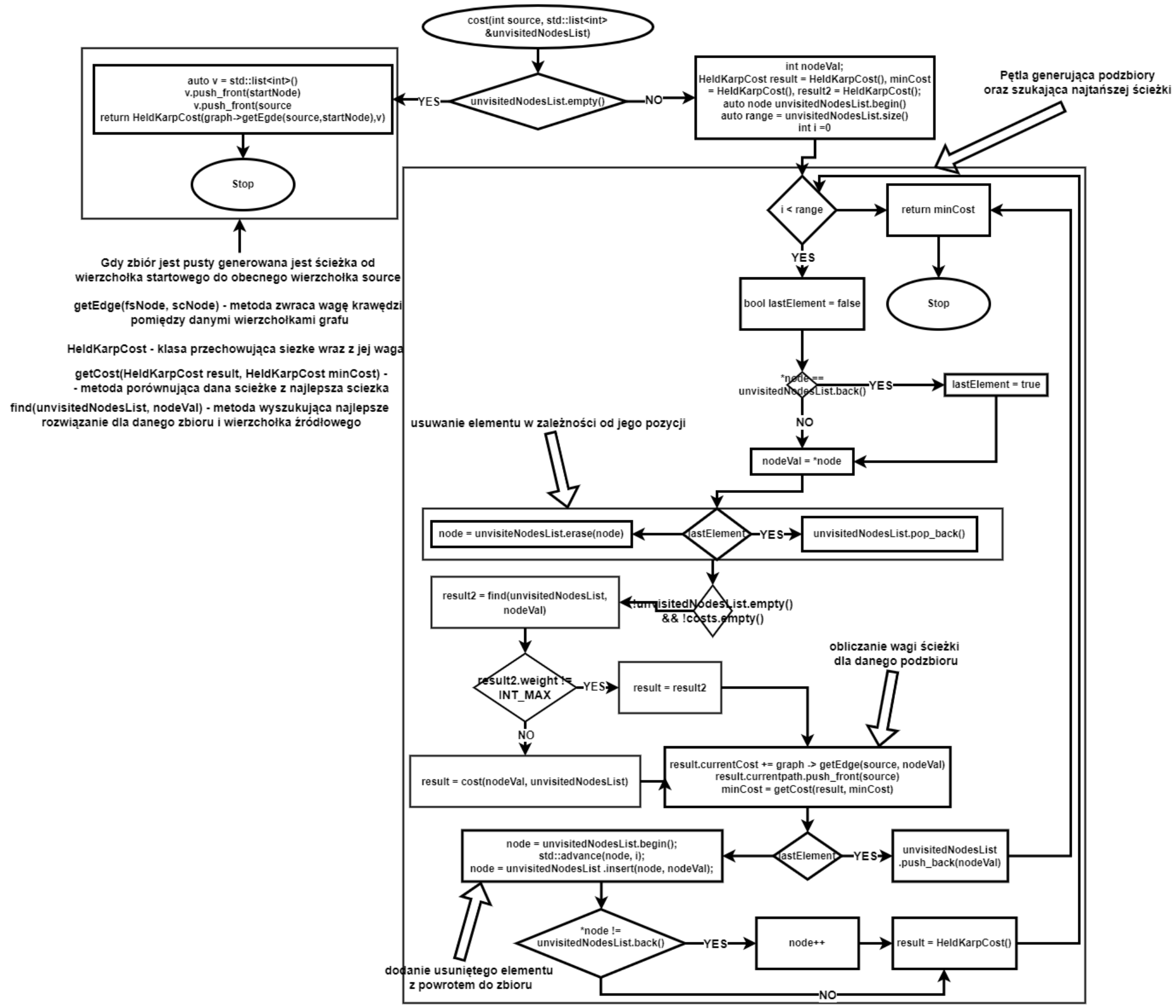
Problem komiwojażera (*eng. Traveling Salesman Problem*) polega na znalezieniu minimalnego cyklu Hamiltona (taki cykl, w którym każdy wierzchołek grafu odwiedzany jest dokładnie raz [oprócz początkowego]) w pełnym grafie ważonym.

2 Metoda

Metoda programowania dynamicznego (*eng. dynamic programming*), polega na rozwiązaniu problemu przez odpowiednie złożenie rozwiązań pod problemów. „Programowanie" oznacza w tym kontekście tabelaryczną metodę rozwiązywania problemów, a nie pisanie programów komputerowych. Programowanie dynamiczne można stosować wtedy, kiedy pod problemy nie są zależne, tzn. kiedy pod problemy mogą zawierać te same pod problemy. W algorytmie opartym na programowaniu dynamicznym rozwiązuje się każdy pod problem tylko raz, po czym zapamiętuje się wynik w odpowiedniej strukturze , unikając w ten sposób wielokrotnych obliczeń dla tego samego pod problemu.

3 Algorytm

Diagram (rys. 1) prezentuje schemat blokowy rekurencyjnej metody cost(). Zadaniem tej metody jest znalezienie najtańszej ścieżki dla zadanego podzbioru.
Wykonanie tego zadnia polega na usuwaniu kolejnych elementów zbioru wywołaniu dla tego podzbioru metody cost, gdzie wierzchołkiem źródłowym będzie usunięty element zbioru. Następnie do zbioru wierzchołków dodawany jest usunięty element. Proces jest powtarzany dla wszystkich elementów zbioru wierzchołków. Jeżeli dany zbiór jest pusty wygenerowana jest ścieżka od wierzchołka startowego do wierzchołka źródłowego.



Rysunek 1: Diagram metody cost

4 Dane testowe

Do sprawdzenia poprawności działania algorytmu, wybrano następujący zestaw instancji:
<nazwa instancji><optymalny wynik>

1. tsp_6_1.txt 132 [0 1 2 3 4 5 0],
2. tsp_6_2.txt 80 [0 5 1 2 3 4 0],
3. tsp_10.txt 212 [0 3 4 2 8 7 6 9 1 5 0];
4. tsp_12.txt 264 [0 1 8 4 6 2 11 9 7 5 3 10 0]
5. tsp_13.txt 269 [0 10 3 5 7 9 11 2 6 4 8 1 12 0]
6. tsp_14.txt 282 [0 10 3 5 7 9 13 11 2 6 4 8 1 12 0]
7. tsp_15.txt 291 [0 12 1 14 8 4 6 2 11 13 9 7 5 3 10 0]
8. tsp_17.txt 39 [0 11 13 2 9 10 1 12 15 14 5 6 3 4 7 8 16 0]
9. tsp_21.txt 212 [0 3 4 2 8 7 6 9 1 5 0];

http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl

Do wykonania badań wybrano następujący zestaw instancji:

1. tsp_6.txt 132 [0 1 2 3 4 5 0],
2. tsp_10.txt 212 [0 3 4 2 8 7 6 9 1 5 0],
3. tsp_15.txt 10 291 [0 1 8 4 6 2 11 9 7 5 3 10 0]
4. tsp_17.txt 10 39 [0 10 3 5 7 9 11 2 6 4 8 1 12 0]
5. tsp_18.txt 10 2376 [0 6 7 5 15 4 8 2 1 14 13 12 17 9 16 10 3 11 0]
6. tsp_19.txt 10 2413 [0 6 7 5 15 4 8 2 1 14 13 12 17 9 16 18 10 3 11 0]
7. tsp_20.txt 10 2543 [0 6 7 5 15 4 8 2 1 14 13 12 17 9 16 18 19 10 3 11 0]
8. tsp_21.txt 1 2707 [0 6 7 5 15 4 8 2 1 20 14 13 12 17 9 16 18 19 10 3 11 0]

http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl
http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl

5 Procedura badawcza

Należało zbadać zależność czasu rozwiązania problemu od wielkości instancji.
W przypadku algorytmu Helda Karpa 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 inicjującym .INI (format pliku: nazwa_instancji liczba_wykonań rozwiązanie_optymalne [ścieżka optymalna]; nazwa pliku .csv)

Treść pliku .INI:

```
tsp_6.txt 10 132 [0 1 2 3 4 5 0]
tsp_10.txt 10 212 [0 3 4 2 8 7 6 9 1 5 0]
tsp_15.txt 10 291 [0 1 8 4 6 2 11 9 7 5 3 10 0]
tsp_17.txt 10 39 [0 10 3 5 7 9 11 2 6 4 8 1 12 0]
tsp_18.txt 10 2376 [0 6 7 5 15 4 8 2 1 14 13 12
17 9 16 10 3 11 0]
tsp_19.txt 10 2413 [0 6 7 5 15 4 8 2 1 14 13 12
17 9 16 18 10 3 11 0 ]
tsp_20.txt 10 2543 [0 6 7 5 15 4 8 2 1 14 13 12
17 9 16 18 19 10 3 11 0 ]
tsp_21.txt 1 2707 [0 6 7 5 15 4 8 2 1 20 14 13
12 17 9 16 18 19 10 3 11 0 ]
outputHeldKarp.csv
```

Każda instancja rozwiązywana była zgodnie z liczbą jej wykonań, np. tsp_14.txt wykonana została 10 razy. Dla instancji tsp_21.txt zmniejszono liczbę powtórzeń z powodu zbyt długiego czasu wykonania metody. Do pliku wyjściowego outputHeldKarp.csv zapisywany był czas wykonania. Plik wyjściowy zapisywany był w formacie csv. Poniżej przedstawiono fragment zawartości pliku wyjściowego.

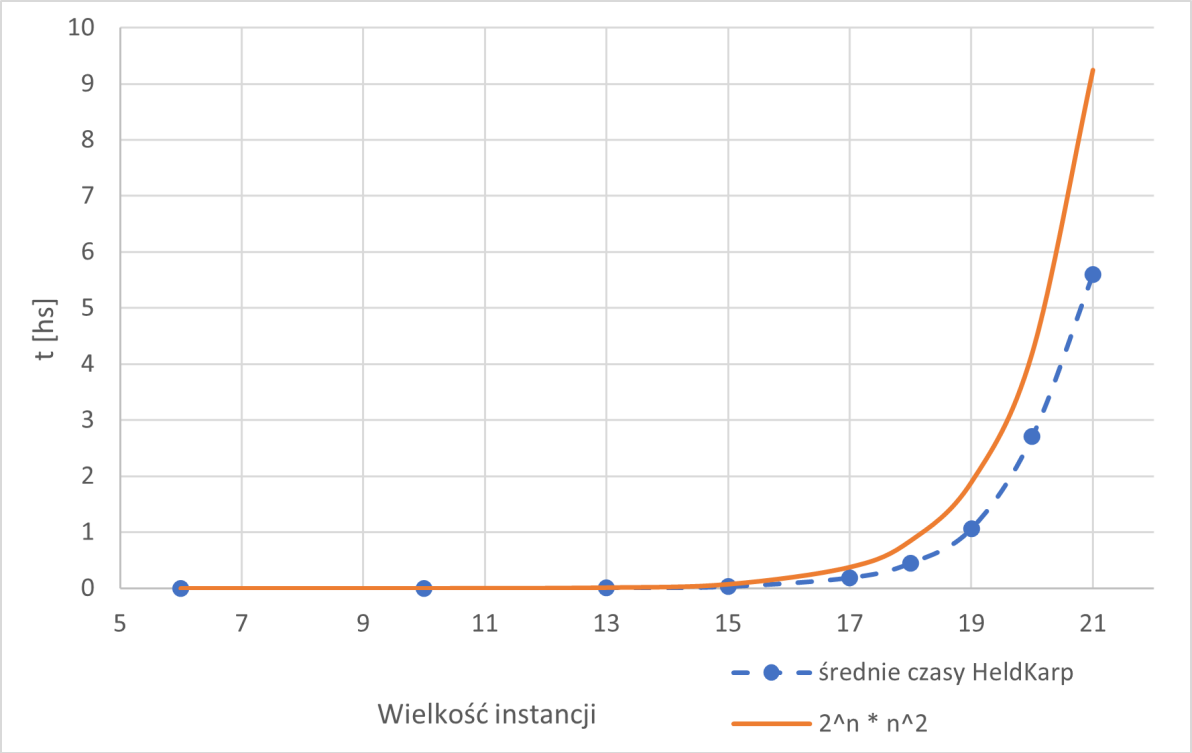
```
tsp_6.txt 10 132 [0 1 2 3 4 5 0 ]
0 132 [0 1 2 3 4 5 0 ]
...
0 132 [0 1 2 3 4 5 0 ]
tsp_10.txt 10 212 [0 3 4 2 8 7 6 9 1 5 0 ]
tsp_12.txt 10 264 [0 1 8 4 6 2 11 9 7 5 3 10 0 ]
tsp_13.txt 10 269 [0 10 3 5 7 9 11 2 6 4 8 1 12 0 ]
```

6 Wyniki

6.1 Pomiar złożoności czasowej

Wyniki zgromadzone zostały w pliku: - „outputHeldKarp.csv”

Wyniki przedstawione zostały w postaci wykresu zależności czasu uzyskania rozwiązania problemu od wielkości instancji (rys. 2):



Rysunek 2: Wykres zależności czasu działania algorytmu Helda Karpa od liczby wierzchołków grafu

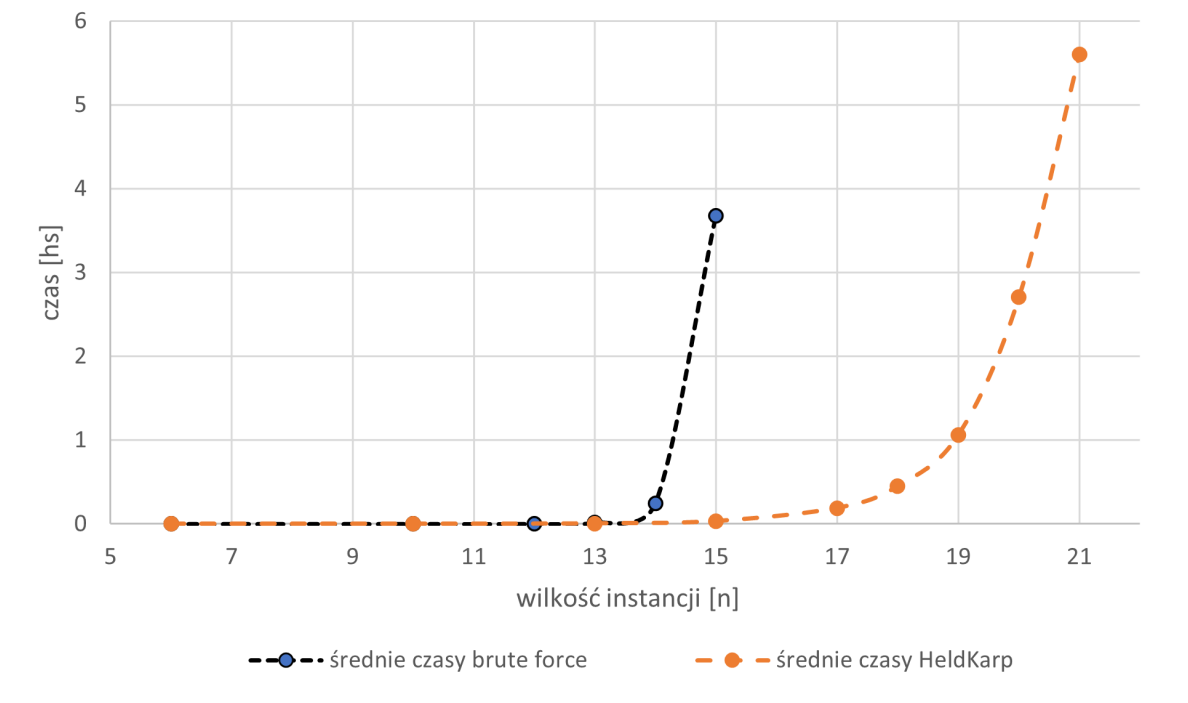
Wyniki opracowane zostały w programie MS Excel.

6.2 Pomiar złożoności pamięciowej

7 Analiza wyników i wnioski

Wyniki przedstawione na wykresie(rys 2) przedstawiają czas(niebieski) potrzebny na rozwiązanie problemu komiwojażera algorytmem Helda Karpa w zależności od ilości wierzchołków w zadanym grafie. Dodatkowo w celu porównania z teoretyczną złożonością przedstawiony został również przeskalowany wykres $2^n * n^2$ (pomarańczowy). W związku z tym udało się zaimplementować algorytm helda karpa z złożonością czasową przybliżoną do tej teoretycznej

Poniżej załączono porównanie złożoności obliczeniowej algorytmu brute force oraz algorytmu Held Karp. Z wykresu (rys 3) wynika, że algorytm Helda Karpa jest znacznie lepszy od metody przeszukiwania zupełnego, ale otrzymanie wyniku dla większych instancji (¿ 21) zajmuje zbyt dużo czasu.



Rysunek 3: Wykres zależności czasu działania algorytmu Helda Karpa oraz Brute force w zależności od liczby wierzchołków grafu