Piotr Błoński nr Indexu: 225959 Prowadzący: Zbigniew Buchalski
Grupa projektowa: Pt 9:15

Projektowanie Efektywnych algorytmów

Projekt nr1:

Implementacja i analiza efektywności algorytmu podziału i ograniczeń i programowania dynamicznego.

Piotr Błoński nr Indexu: 225959 Prowadzący: Zbigniew Buchalski
Grupa projektowa: Pt 9:15

1.Wstęp teoretyczny

Problem komiwojażera (ang. Travelling salesman problem) jest problemem optymalizacyjnym. Należy do problemów NP-zupełnych I polega na znalezieniu cyklu Hamiltona o najniższym koszcie w grafie pełnym ważonym. Początki rozpatrywania tego problem wzięły się od przysłowiowego kupca który szukał najkrótszej trasy aby odwiedzić każdego klienta w najbardziej optymany sposób. Problem komiwojażera dzielimy na przypadki symetryczne I asymetryczne. Symetryczne pozwalają na podróż między miastami A i B po tym samym koszcie, natomiast Asymetryczny nie (typowy przykład z biletami lotniczymi, koszt trasy Wrocław->Tokio nie jest równy kosztowi trasy Tokio-Wrocław)

2.Wstęp teoretyczny

- Brute-force:
  W skrócie polega na siłowym obliczeniu każdej możliwej permutacji miast.
  Niestety jego złożoność to aż n! co przy n=14 daje aż 87 178 291 200 kombinacji.

- Programowanie dynamiczne Jest to strategia przy projektowaniu algorytmów stosowaną najczęściej do rozwiązywania zagadnień optymalizacyjnych. Generalnie chodzi o zapamiętywanie wyników poprzednich obliczeń aby użyć ich przy innej operacji za pomocą rekurencji. Przedstawiany w tym projekcie algorytm jest zwany algorytmem Helda-Karpa. Pozwala on obniżyć złożoność obliczeniową z n! na $O(2nn2)$ kosztem przestrzeni pamięci $O(2nn)$

- Branch&Bound Jest to znana metoda rozwiązywania problemów optymalizacyjnych od roku 1960. Opiera się na analizie drzewa przestrzeni stanów. Drzewo to reprezentuje wszystkie możliwe ścieżki jakimi może pójść algorytm rozwiązując problem. Algorytm rozpoczyna się w korzeniu drzewa i przechodząc do liścia konstruuje rozwiązanie. W każdym węźle oblicza się granice (ograniczenie), która pozwala określić go jako obiecujący węzeł lub nie. W dalszej fazie algorytm przegląda tylko potomków węzłów obiecujących. Pozwala to na zmniejszenie ilości odwiedzanych wierzchołków i szybsze rozwiązanie problemu. Jego złożoność bardzo zależy od ograniczeń które wybierzemy.

3.Przykłady algorytmów dla danego grafu S o wierzchołkach V=4:

0	**2**	**8**	**3**
1	0	5	4
3	1	0	5
4	3	2	0

Gdzie prawidło ścieżka to: 1-4-3-2-1 O koszcie: 7

- Programowanie dynamiczne: Poszukujemy najniższego kosztu do punktu startowego

$C(S, i) = \min\{C(S - \{i\}, j) + dis(j, i)\}$

gdzie j jest jednym z wierzchołków I nie jest w zbiorze i.
Na początku badamy dla wszystkich ścieżek 1 – j
Zapisujemy wyniki tych ścieżek w przygotowanej strukturze.
Następnie obliczamy koszt następnych wierzchołków rekurencyjnie.
Krok po kroku obliczenia wyglądały by w następujący sposób:
$C(2, phi) = 1$ $C(3, phi) = 3$ $C(4, phi) = 4$
Więc wiemy że z wierzchołków 2 3 4 do 1 idziemy po wyżej napisanych kosztach.
$C(2, \{3\} = d(2,3) + C(3, phi) = 5 + 3 = 8$
Widzimy że drogę 3 do 1 już mamy obliczoną!
Nie musimy jej liczyć co zaoszczędzi nam czasu.
Dalsze obliczenia:
$C(2, \{4\} = d(2,4) + C(4, phi) = 3 + 4 = 7$
$C(3, \{2\} = d(3,2) + C(2, phi) = 1 + 1 = 2$
$C(3, \{4\} = d(3,4) + C(3, phi) = 5 + 3 = 8$
$C(4, \{3\} = d(4,3) + C(4, phi) = 2 + 4 = 6$
$C(4, \{2\} = d(4,2) + C(4, phi) = 2 + 3 = 5$
Następnym krokiem będzie wykonanie obliczeń dla pozostałych ścieżek
$C(2, \{3,4\} = \min\{d(2,3) + C(3, \{4\}) , d(2,4) + C(4, \{3\})) = \min(13,10) = 10$
$C(3, \{2,4\} = \min\{d(3,2) + C(2, \{4\}) , d(3,4) + C(4, \{2\})) = \min(8,10) = 10$
$C(4, \{2,3\} = \min\{d(4,2) + C(2, \{3\}) , d(4,3) + C(3, \{2\})) = \min(10,4) = 4$
I obliczamy teraz drogę od pierwszego węzła do obliczonych ścieżek:
$C(1, \{2,3,4\} = \min\{d(1,2) + C(2, \{3,4\}) , d(1,3) + C(3, \{2,3\}, d(1,4) + C(4, \{2,3\})) =$
$\min(11,13,8) = 8$

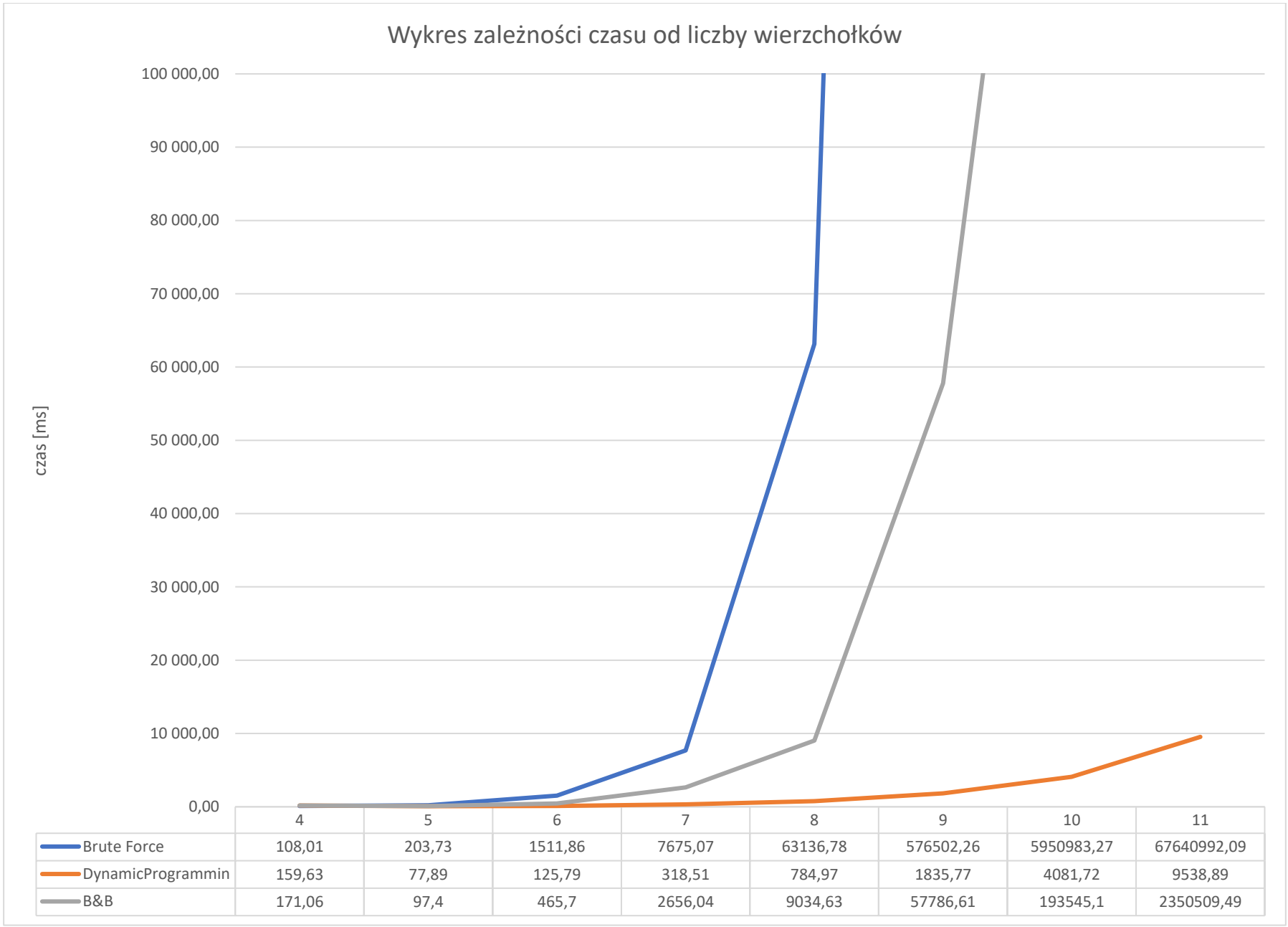
Naszą drogą jest 1 – 4 – 3 – 2 -1

Drogę rekonstruujemy na podstawie naszej struktury z pamięcią.

Piotr Błoński
Grupa projektowa: Pt 9:15

nr Indexu: 225959

Prowadzący: Zbigniew Buchalski



	4	5	6	7	8	9	10	11
Brute Force	108,01	203,73	1511,86	7675,07	63136,78	576502,26	5950983,27	67640992,09
DynamicProgrammin	159,63	77,89	125,79	318,51	784,97	1835,77	4081,72	9538,89
B&B	171,06	97,4	465,7	2656,04	9034,63	57786,61	193545,1	2350509,49

Piotr Błoński nr Indexu: 225959 Prowadzący: Zbigniew Buchalski
Grupa projektowa: Pt 9:15

Wnioski:

• Wybór sposobu ograniczenia ma ogromny wpływ na prędkość algorytmu typu Branch and Bound

• Algorytmy przeglądu zupełnego są niestety wolne, ale łatwe w implementacji

• Programowanie dynamiczne pozwala na rekurencyjne zwalczenie problemu optymalizacyjnego, ale jedynie w przypadkach jeżeli problemy są od siebie zależne

• Duży wpływ mają na pomiary inne procesy włączone równolegle na komputerze

• Opłaca się do wykonania pomiarów używać biblioteki excel, która pozwala wszystkie wyniki wrzucić od razu do arkusza kalkulacyjnego.

• Algorytmy typu Branch and Bound mają w najgorszym wypadku złożoność obliczeniową podobną do przeglądu zupełnego.

Pomiary zostały przeprowadzone na komputerze o następujących parametrach: Procesr: Intel(R) Core(TM) i7-6700K CPU @ 4.00GHz, 4008 MHz, Rdzenie: 4,
Procesory logiczne: 8
RAM : 16GB