Implementacja i analiza efektywności algorytmów optymalnych o pseudowielomianowej złożoności obliczeniowej dla wybranych problemów kombinatorycznych

1. Plan pomiarów

a. Generowanie przedmiotów do wyboru

Aby zagwarantować aby sumaryczny rozmiar przedmiotów w każdej instancji problemu był większy niż pojemność plecaka (min. 25%), pod koniec każdego losowania program obliczał ich stosunek, jeśli był mniejszy niż 1.25 zwiększał zakres wag jakie losował przedmiotom i ponawiał losowanie.

b. Metoda pomiaru czasu i platforma testowa

Do pomiaru czasu użyto funkcji QueryPerformanceCounter.
Program kompilowana w trybie realease(optymalizacja o2) w Visual Studio 2015.
Testy przeprowadzano na laptopie z procesorem Intel Core i7-4720HQ, 6MB cache, z taktowaniem ograniczone do 2,6 GHz.

Wszystkie pomiary zostały wykonane na 100 losowych instancjach problemu, chyba że obok wyników zostało napisane inaczej(dot. przeglądu zupełnego).

2. Przegląd zupełny(bruteforce)

a. Złożoność obliczeniowa i implementacja

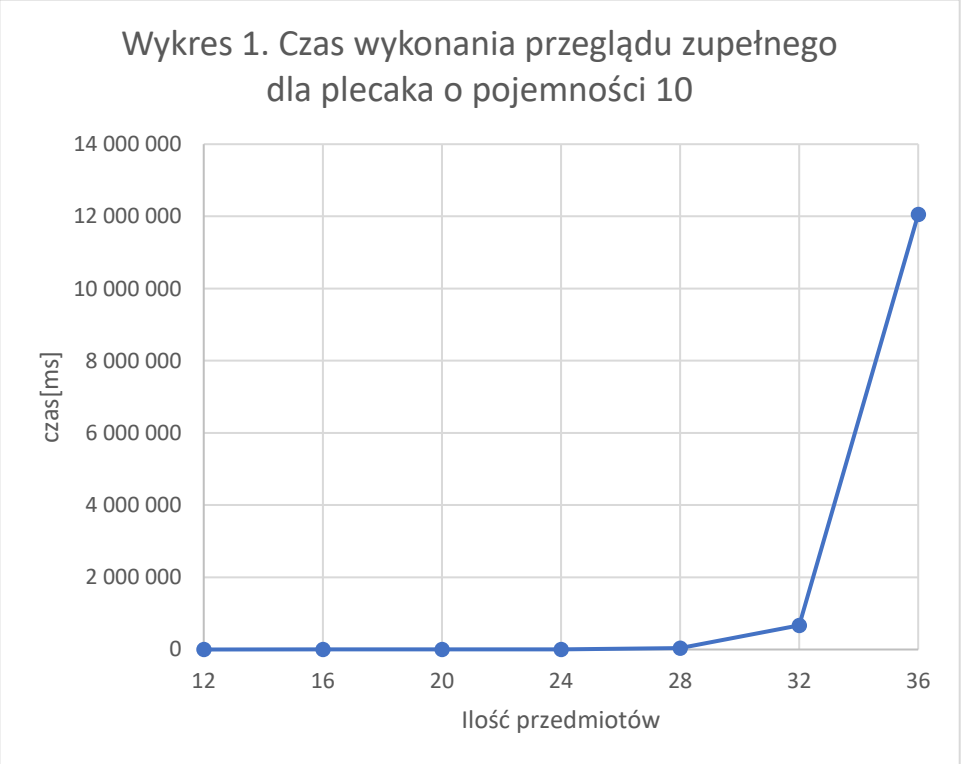
Złożoność obliczeniowa wynosi $O(2^n)$ ponieważ tyle jest możliwych kombinacji 0-1 dla n przedmiotów. Pojemność plecaka nie ma znaczenia. Wyniki pomiarów pokrywają się z teorią. W mojej implementacji nie użyłem rekurencji, zamiast tego inkrementuję wektor booleanów.

b. Wyniki pomiarów

Ze względu nas bardzo długi czas wykonania, pomiary dla plecaka o pojemności 28 litrów wykonano tylko 32-krotnie, a dla 32 i 36 l – jednokrotnie.

Tabela 1. Zależność Pojemności plecaka i ilości przedmioty na czas wykonywania przeglądu zupełnego, czas w ms

Poj\il. przed	12	16	20	24	28	32	36
10	0,39315	6,992	123,281	2 169,46	38 719,41	668 538,45	12 055 142,17
20	0,41514	6,886	123,476	2 169,26	-	-	-
30	0,39820	7,173	123,293	2 169,70	-	-	-



3. Algorytm zachłanny(greedy)

a. Złożoność obliczeniowa i implementacja

Złożoność obliczeniowa wynosi O(n log n + n) – posortowanie wszystkich przedmiotów według jednego parametru (w mojej implementacji – ich ilorazu wartości do objętości, sortowaniem przez kocpowanie) + iteracja po nich aż do zapełnienia plecaka.

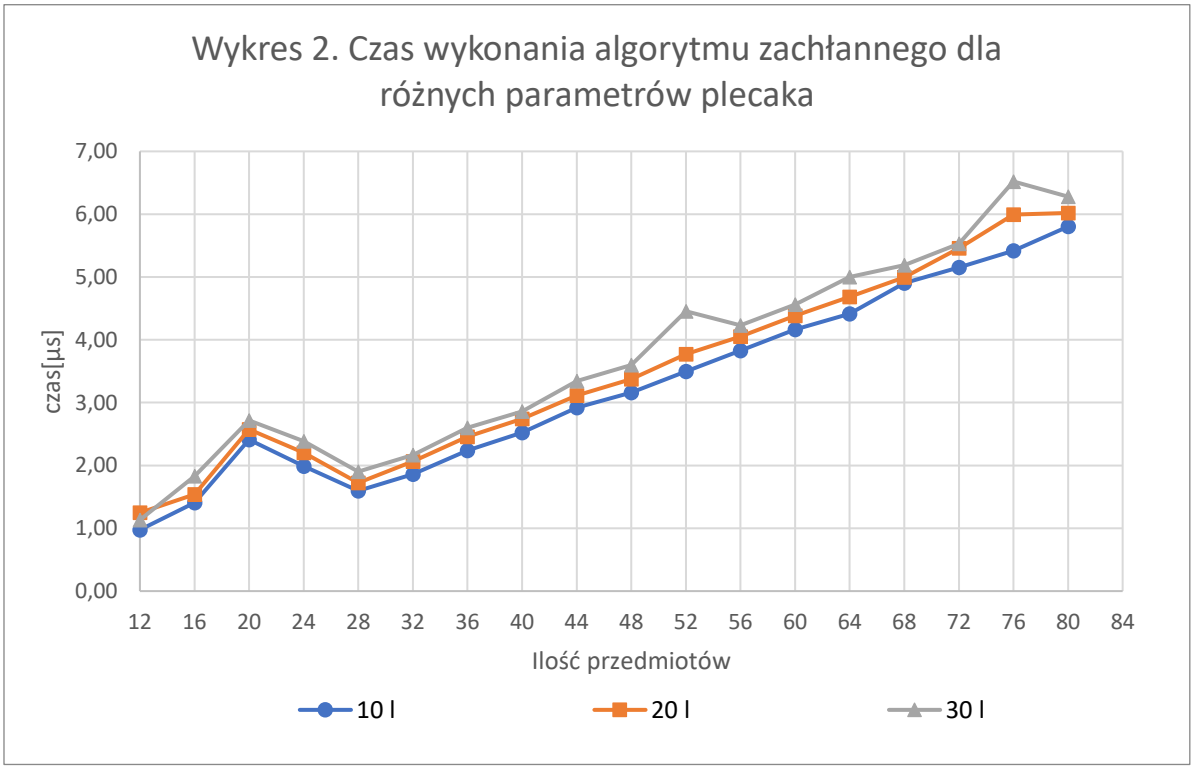
Pojemność plecaka ma drobny wpływa na czas rozwiązywania pojedynczej instancji, im jest ona większa tym więcej przedmiotów jest średnio rozpatrywanych zanim algorytm napełni plecak, ale nie ma wpływy na warunek pesymistyczny (np. gdy najmniej wartościowy przedmiot jako jedyny mieści się w prawie pełnym plecaku).

Algorytm jest bardzo szybki, ale nie rozwiązuje problemu znalezienia optymalnego zestawu przedmiotów prawie zawsze daje gorsze wyniki od algorytmu zachłannego i dynamicznego, średnio o 10-20%.

b. Wyniki pomiarów

Tabela 2. Zależność Pojemności plecaka i ilości przedmioty na czas wykonywania przeglądu zupełnego, czas w µs

Poj\il. przed	12	16	20	24	28	32	36	40
10 l	0,98	1,41	2,41	1,99	1,60	1,86	2,24	2,52
20 l	1,25	1,54	2,57	2,20	1,72	2,07	2,46	2,74
30 l	1,13	1,83	2,72	2,38	1,90	2,17	2,60	2,86
Poj\il. przed	**48**	**52**	**56**	**60**	**64**	**68**	**72**	**76**
10 l	3,16	3,50	3,83	4,17	4,41	4,90	5,15	5,42
20 l	3,38	3,77	4,05	4,38	4,69	4,99	5,46	5,99
30 l	3,60	4,45	4,23	4,56	5,00	5,19	5,53	6,52



4. Algorytm oparty o programowanie dynamiczne(dynamic)

a. Złożoność obliczeniowa i implementacja

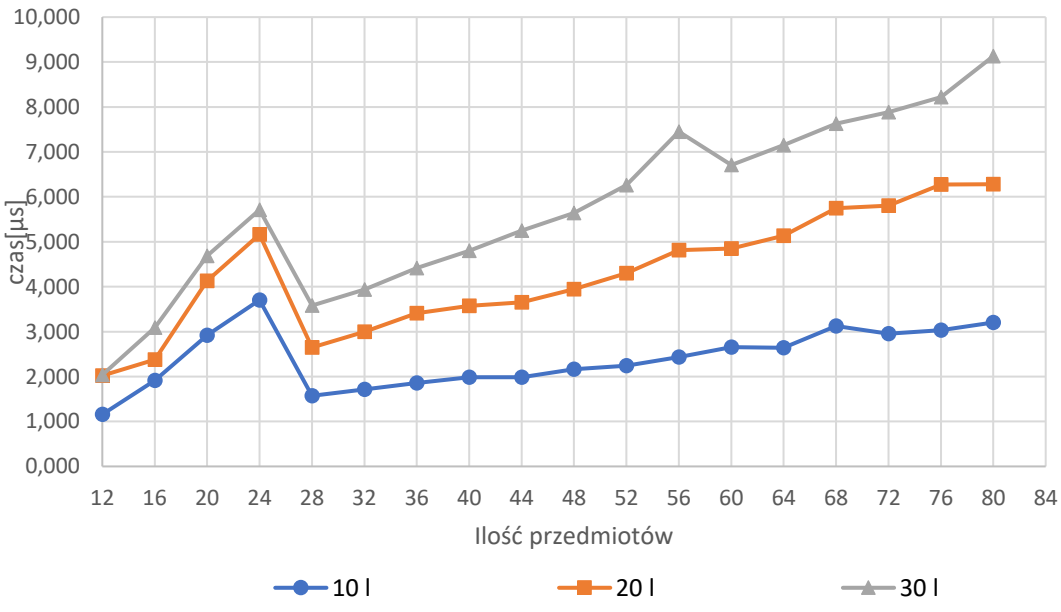
Złożoność obliczeniowa wynosi O(n*c), gdzie n oznacza ilość przedmiotów, c - pojemność plecaka, wynika to z faktu że do rozwiązania problemu rozwiązujemy podproblem, efektywnie tworząc tabelę rozwiązań instancji o rozmiarach od 1 do n przedmiotów i od 1 c jednostek pojemności.

b. Wyniki pomiarów

Tabela 3. Zależność Pojemności plecaka i ilości przedmioty na czas wykonywania algorytmu dynamicznego, czas w µs

Poj\il. przed	12	16	20	24	28	32	36	40
10 l	1,157	1,915	2,921	3,703	1,571	1,717	1,855	1,986
20 l	2,021	2,376	4,129	5,163	2,649	2,996	3,407	3,576
30 l	2,037	3,091	4,682	5,708	3,584	3,94	4,417	4,796
Poj\il. przed	48	52	56	60	64	68	72	76
10 l	1,986	2,163	2,238	2,432	2,653	2,637	3,126	2,953
20 l	3,651	3,944	4,299	4,812	4,852	5,136	5,748	5,803
30 l	5,25	5,641	6,257	7,449	6,707	7,153	7,631	7,887

Wykres 3. Czas wykonania algorytmu dynamicznego dla różnych parametrów plecaka



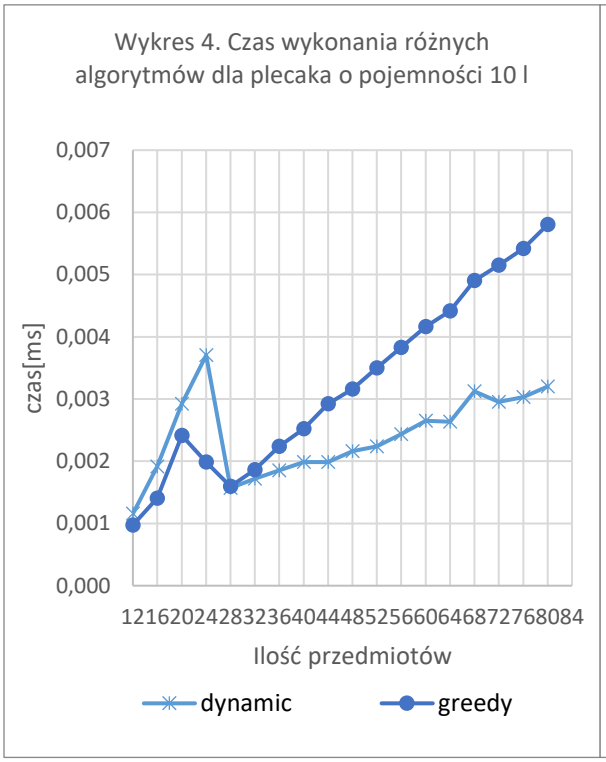
5. Konkluzje

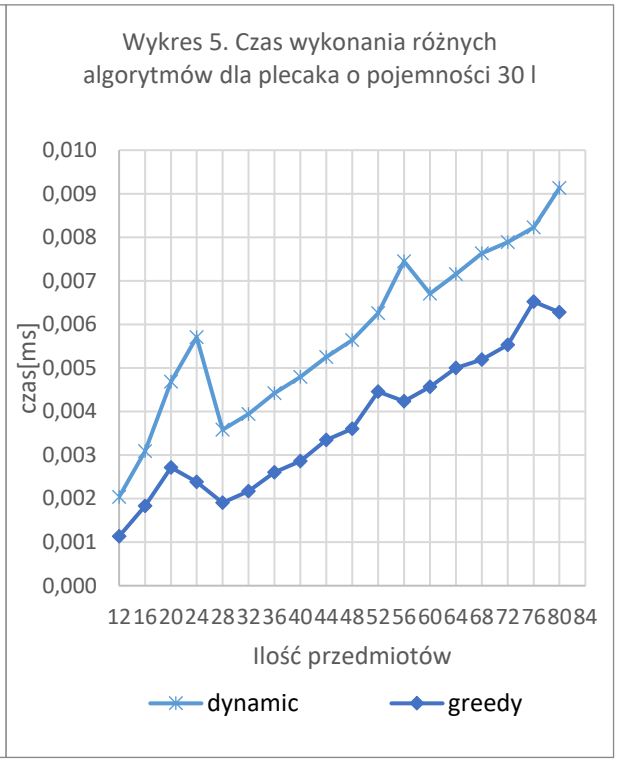
Przegląd zupełny dla problemu plecakowego bardzo szybko staje się absurdalnie kosztowny.

Algorytm zachłanny w niektórych przypadkach daje bardzo dobre przybliżenie rozwiązania instancji problemu plecakowego, ale to tylko przybliżenie.

Algorytm oparty o programowanie dynamiczne jest stosunkowo szybki dla rozważanego dyskretnego problemu plecakowego, niestety jego złożoność pamięciowa i obliczeniowa rośnie wraz oboma parametrami.

Dla bardzo małych pojemności plecaka, algorytm zachłanny jest wolniejszy od dynamicznego.





6. Literatura

https://www.wikipedia.org/

http://www-users.mat.uni.torun.pl/~henkej/knapsack.pdf