Marcin Całkowski
nr 218191
Śr. 13.15

# Struktury danych i złożoność obliczeniowa

## Zadanie projektowe nr 3

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

Prowadzący: Zbigniew Buchalski

Politechnika Wrocławska

# 1. Wstęp

Tematem projektu jest przedstawienie działania algorytmów rozwiązujących dwa różne problemy: Dyskretny problem plecakowy (knapsack) oraz problem komiwojażera (travelling salesman problem). Dla każdego z problemów należało wybrać w wariancie minimalnym jeden algorytm optymalizacyjny dla wybranego problemu. W obydwu przypadkach wybrałem przegląd zupełny (bruteforce). Dodatkowymi założeniami projekty była konieczność dynamicznego alokowania danych, weryfikacja poprawności rozwiązania oraz przetestowania czasu działania algorytmów dla różnych wielkości instancji testowanych.

# 2. Podstawowe zagadnienia teoretyczne

**Dyskretny problem plecakowy** (ang. discrete knapsack problem) jest jednym z najczęściej poruszanych problemów optymalizacyjnych. Nazwa zagadnienia pochodzi od maksymalizacyjnego problemu wyboru przedmiotów, tak by ich sumaryczna wartość była jak największa i jednocześnie mieściły się w plecaku. Przy podanym zbiorze elementów o podanej wadze i wartości, należy wybrać taki podzbiór by suma wartości była możliwie jak największa, a suma wag była nie większa od danej pojemności plecaka. Parametrem plecaka jest skończony zbiór elementów $A = \{ a1, a2, ..., an \}$, z których każdy ma określony rozmiar $s(ai) > 0$ i wartość $w(ai) > 0$ oraz pojemność plecaka $b > 0$. Rozwiązaniem jest podzbiór elementów $A' \subset A$, który maksymalizuje łączną wartość wybranych elementów przy warunku nie przekroczenia dopuszczalnej pojemności plecaka.

**Problem komiwojażera** (ang. travelling salesman problem, TSP) - zagadnienie optymalizacyjne, polegające na znalezieniu minimalnego cyklu Hamiltona w pełnym grafie ważonym. Nazwa pochodzi od typowej ilustracji problemu, przedstawiającej go z punktu widzenia wędrownego sprzedawcy (komiwojażera): dane jest n miast, które komiwojażer ma odwiedzić, oraz odległość pomiędzy każdą parą miast. Celem jest znalezienie najkrótszej drogi łączącej wszystkie miasta, zaczynającej się i kończącej się w określonym punkcie. Parametrem problemu jest skończony zbiór miast $C = \{ c1, c2, ..., cn \}$ oraz odległości $d_{ij}$ z miasta ci do miasta $c_j$ (nie ma wymogu $d_{ij} = d_{ji}$). W celu rozwiązania problemu należy określić kolejność odwiedzania wszystkich miast (ich permutację) < ci[1], ci[2], …, ci[n] >, aby sumaryczna trasa była jak najkrótsza przy założeniu, że każde miasto zostało odwiedzone dokładnie jeden raz i nastąpił powrót do miasta początkowego.

**Algorytm Brute force** jest bardzo prymitywnym, ale przez to i prostym algorytmem. Załóżmy, że mamy 3 przedmioty w sklepie, $P = \{ a, b, c \}$, o masach odpowiednio $m = \{ 2, 4, 8 \}$ i wartościach $w = \{ 10, 20, 30 \}$ Złodziej, może zabrać ze sobą masę = 10. Jak łatwo się przekonamy, sumując wszystkie masy, złodziej nie może w tym wypadku zabrać wszystkiego, musi dokonać pewnych kompromisów. Nasza implementacja algorytmu operuje na permutacjach zbioru P, czyli elementy ustawiane są w każdy możliwy tylko sposób ( abc, acb, bac, bca, cab, cba itd. ) Każde z tych możliwych ustawień jest sprawdzane, wg. kolejności elementów czy zmieści się w plecaku, gdy element się nie mieści, nie jest on wkładany do plecaka i na końcu liczona jest wartość całkowita. Dla przykładu dla pierwszego i drugiego ustawienia pierwsze dwa się mieszczą, a ostatni nie. Dla ustawienia cba, mieści się tylko pierwszy element. Jeżeli wartość całkowita plecaka okazuje się być większa niż poprzednia największa, ta druga jest nadpisywana pierwszą.

## 3. Plan eksperymentu

Eksperyment został wykonany w moim własnym programie którego kod został oparty na algorytmach znalezionych w materiałach z zajęć oraz informacjach z sieci. Podstawą jego wykonania jest zbadanie czasu wykonywania algorytmu bruteforce dla każdego problemu. Dane do testów czasowych zostały każdorazowo wylosowane aby dać pewność średnich wyników. Dla każdej struktury test przeprowadzany jest **100 razy**, uzyskany wynik czasowy jest średnią z wszystkich uzyskanych wyników.

Czas został zmierzony przy pomocy biblioteki **chrono** (dostępnej od standardu c++11). Pozwala ona na uzyskanie dobrego pomiaru czasu bez komplikacji w kodzie, co znacząco ułatwia testowanie wszystkich struktur. Wszystkie wyniki czasowe przedstawione są w **milisekundach**. Maszyna testowa wyposażona jest w procesor **Intel i7** z taktowaniem **2.4 GHz** oraz **8GB** pamięci RAM, a program uruchamiany był w wersji 64 bitowej.

Testy wykonywane są struktur o następujących wielkościach (ze względu na różne czasy wykonywania):

- Problem plecakowy – ilość przedmiotów: **14, 15, 16, 17, 18** (stała pojemność – 200)
- Problem plecakowy – pojemność plecaka: **200, 400, 600, 800, 1000** (stała ilość – 15)
- Problem komiwojażera: **8, 9, 10, 11, 12**

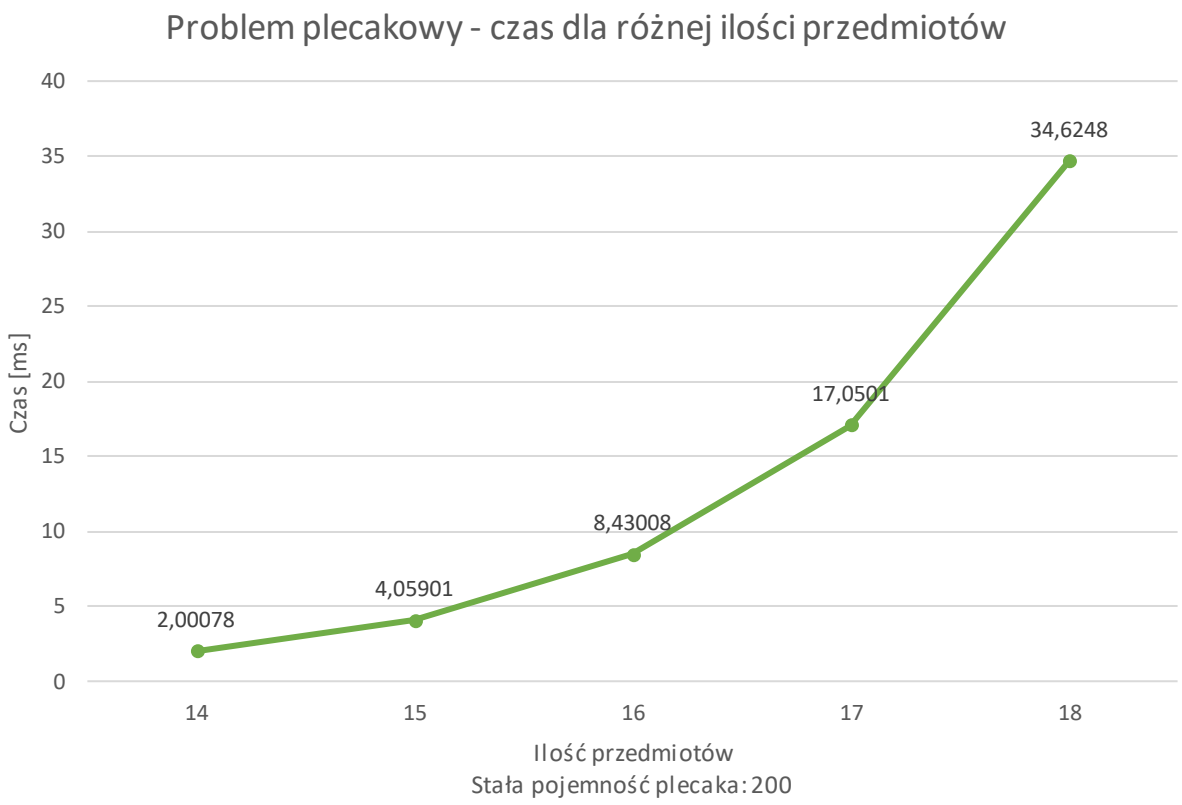
## 4. Wyniki eksperymentu

Na następnej stronie przedstawiono w tabelach oraz wykresach wyniki uzyskane podczas badania algorytmu buteforce dla danego problemu.

### 4.1. Problem plecakowy – ilość przedmiotów w plecaku

| Ilość przedmiotów | 14 | 15 | 16 | 17 | 18 |
|---|---|---|---|---|---|
| czas[ms] | 2,00078 | 4,05901 | 8,43008 | 17,0501 | 34,6248 |

Tabela 1: Wyniki czasowe algorytmu Bruteforce dla problemu plecakowego w zależności od ilości dostępnych przedmiotów (stała pojemność plecaka: 200).

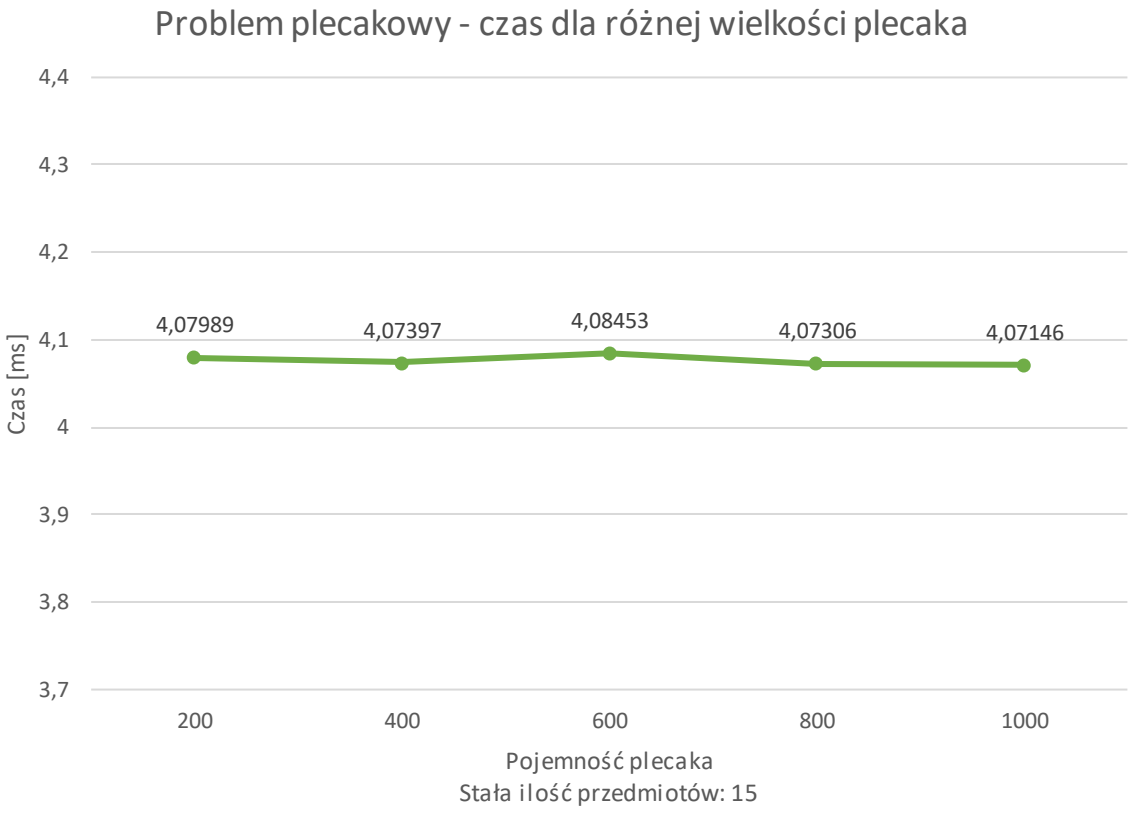



Wykres 1: Wykres wyników czasowych dla danych przedstawionych w tabeli 1.

### 4.2. Problem plecakowy – pojemność plecaka

| Pojemność plecaka | 200 | 400 | 600 | 800 | 1000 |
|---|---|---|---|---|---|
| czas[ms] | 4,07989 | 4,07397 | 4,08453 | 4,07306 | 4,07146 |

Tabela 2: Wyniki czasowe algorytmu Bruteforce dla problemu plecakowego w zależności od pojemności plecaka (stała ilość przedmiotów: 15).

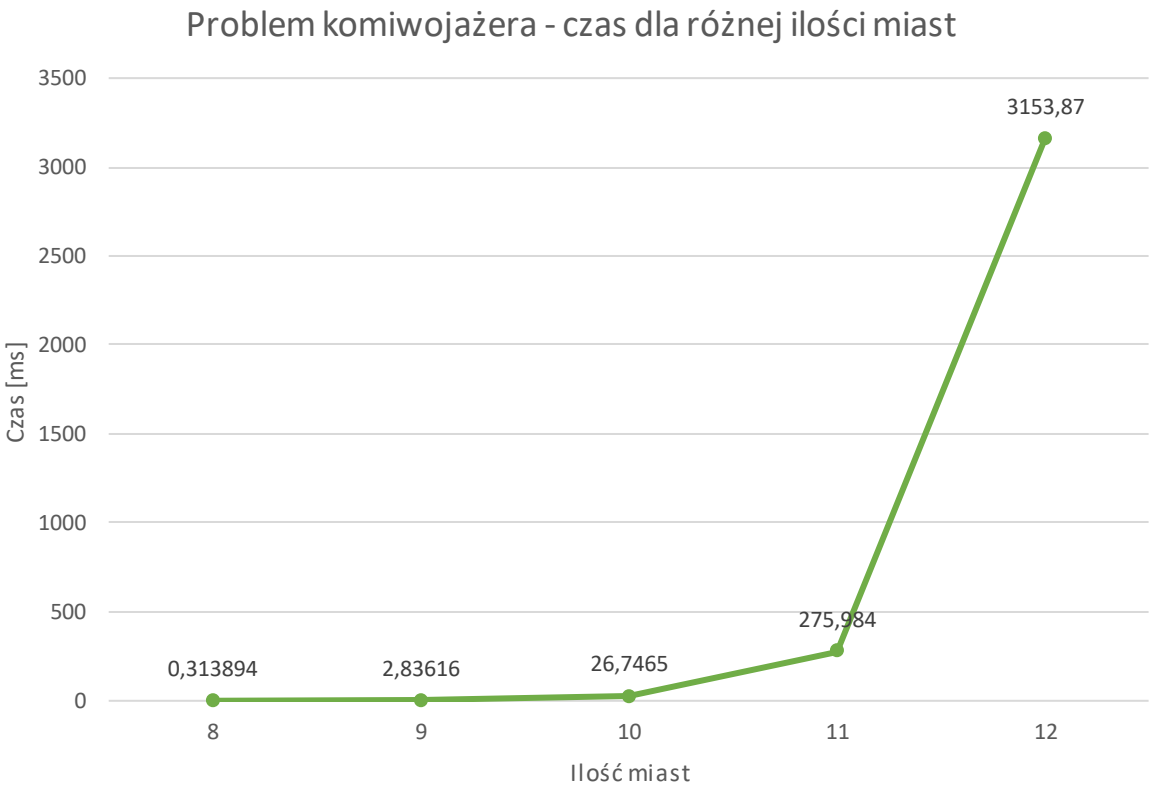



Wykres 2: Wykres wyników czasowych dla danych przedstawionych w tabeli 2.

### 4.3. Problem komiwojażera – ilość miast

| Ilość miast | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|
| czas[ms] | 0,313894 | 2,83616 | 26,7465 | 275,984 | 3153,87 |

Tabela 3: Wyniki czasowe algorytmu Bruteforce dla problemu komiwojażera w zależności od ilości miast.




Wykres 3: Wykres wyników czasowych dla danych przedstawionych w tabeli 3.

## 5. Wnioski

Przeprowadzony eksperyment zgadza się ze złożonością obliczeniową jaka przyznawana jest dobremu, chodź mało efektywnemu algorytmowi jakim jest Bruteforce. Jak widać po wynikach w tabeli 1, algorytm dla problemu plecakowego w zależności od ilości dostępnych ten osiąga niemalże dokładnie złożoność obliczeniową rzędu $2^n$, niestety w przypadku problemu komiwojażera czas dla każdego kolejnego rzędu wzrasta znacząco.

O ile w przypadku problemu plecakowego przedmiotów przyrost czasowy pozwala na przejrzeniu 18 elementów w 18 ms (tabela 1), to w przypadku problemu TSP przegląd już 12 miast zajmuje ponad 3 sekundy, gdzie dla 11 było to „zaledwie" 275 ms, czyli około 10 razy wolniej (tabela oraz wykres 3). Różnica ta jest spowodowana złożonością działania algorytmu dla obydwu problemów, w tym przede wszystkim koniecznością przeglądania tablicy dwuwymiarowej dla problemu TSP. Wywnioskować z tego można, że o ile jeszcze dla problemu plecakowego algorytm Bruteforce jeszcze daje przyzwoite wyniki (chodź i tak jest gorszy od jakichkolwiek innych algorytmów), to ten sam algorytm zastosowany dla problemu TSP wypada fatalnie.

Warto również zauważyć, że dla problemu plecakowego rozmiar plecaka nie wpływa na czas działania algorytmu. Jak pokazuje Tabela oraz Wykres 2, czas dla instancji o wielkości 15 jest niemalże identyczny, a wszelkie rozbieżności czasowe w tysięcznych milisekundy są najprawdopodobniej spowodowane błędami pomiarowymi. Widać więc, że pojemność plecaka nie ma znaczącego wpływu na działanie algorytmu, a ten zależy wyłącznie od ilości przeglądanych elementów.