Struktury danych i złożoność obliczeniowa projekt

Zadanie 1.

Jacek Czewonka — 241373

Prowadzący:
dr inż. Jarosław Mierzwa

Wrocław, 5 kwietnia 2019

Spis treści

1 Wstęp **2**
1.1 Założenia techniczne 2

2 Teoria **2**
2.1 Teoretyczne złożoności 2
2.1.1 Tablica i Lista 2
2.1.2 Kopiec 2
2.2 Tabele 3

3 Eksperyment **3**
3.1 Opis 3
3.2 Cele 3
3.3 Obserwacje 4
3.4 Wnioski 4

4 Wykresy **5**
4.0.1 Tablica 5
4.0.2 Lista 6
4.0.3 Kopiec 7

1 Wstęp

Celem projektu jest zaimplementowanie podstawowych struktur danych. Zadanie zostało zaprogramowane w języku c++ w wersji obiektowej. Program umożliwia wykonywanie operacji na strukturach.

Struktury:

- Tablica dynamiczna
- Lista podwójnie wiązana
- Kopiec(Sterta)

1.1 Założenia techniczne

Podstawowym typem danych jest 32 bitowa liczba ze znakiem (int). Do kompilacji kodu użyto kopilator g++ (gcc) oraz linker ld. Użyty standard języka to C++11. Programy kompilowane, uruchamiane i testowane pod systemem linux.

2 Teoria

2.1 Teoretyczne złożoności

Złożoność obliczeniową dzieli się na złożoność czasową i pamięciową. W zadaniu rozpatrzono i zmierzono złożoność czasową. Złożoność pamięciowa jest w tym wypadku bezpośrednio związana z wielkością struktury. Do oceny algorytmów najczęściej używa się notacji dużego O(f(n)). Oznacza ona, że funkcja złożoności algorytmu jest od góry ograniczona przez f dla prawie wszystkich argumentów.

2.1.1 Tablica i Lista

Tablicę i listę rozpatrywać można w podobny sposób ponieważ obie struktury są liniowe oraz umożliwiają wykonanie podobnych operacji. W przypadku tablicy nie implementowano amortyzacji kosztów. Rozpatrzone są pesymistyczne przypadki.

Operacja	Tablica	Lista dwu-kierunkowa
dodawanie na początku	O(n)	O(1)
dodawanie na końcu	O(n)	O(1)
dodawanie w środek	O(n)	O(n)
usuwanie z początku	O(n)	O(1)
usuwanie z końca	O(n)	O(1)
usuwanie ze środka	O(n)	O(n)
dostęp indeksowy	O(1)	O(n)
szukanie wartości	O(n)	O(n)

2.1.2 Kopiec

W przyjętej implementacji kopca pamięć jest realokowana z nadmiarem w celu amortyzacji kosztu czasu. Rezerwowany jest dwukrotny nadmiar pamięci. Zysk czasu kosztem pamięci.

Operacja	Kopiec
dodanie wartości	O(log(n))
usunięcie korzenia	O(log(n))
usunięcie klucza	O(n log(n))
szukanie wartości	O(n)

2.2 Tabele

3 Eksperyment

3.1 Opis

W celu przebadania czasu trwania operacji użyto standardowej biblioteki c++ <chrono>. Biblioteka ta w standardzie c++11 udostępnia zegar wysokiej rozdzielczości. Sprawdzono czas dodawania losowych

Metoda wykonywanie pomiaru:

```
1         std::chrono::high_resolution_clock::time_point t1;
2         std::chrono::high_resolution_clock::time_point t2;
3
4         t1 = std::chrono::high_resolution_clock::now();
5
6         //mierzona operacja
7
8         t2 = std::chrono::high_resolution_clock::now();
9         std::chrono::duration<double> pomiar =
10                        duration_cast<duration<double>>(t2 - t1);
```

Sprawdzono czas dodawania losowych liczb do struktur. Początkowo sprawdzono czy jest różnica między dodawanie konkretnych wartości takich jak: 0, -1 (binarnie same jedynki 0xffffff). Czasy bardzo zbliżone do siebie. W przypadku kopca rozpatrzono również dodawanie, a następnie usuwanie ciągów malejących oraz rosnących.

Nie mierzone były pojedyncze operacje lecz wiele operacji pod rząd (przykładowo 1000).

```
1         for(int j = 1000; j>0 ; j--)
2         {
3                 t1 = high_resolution_clock::now();
4                 for(int i=0; i<1000; i++)
5                         tablica.remove_end();
6
7                 t2 = high_resolution_clock::now();
8                 duration<double> pomiar =
9                                 duration_cast<duration<double>>(t2 - t1);
10                out_file<<(j)*1000<<","<<std::fixed<<std::setprecision (12)
11                        << pomiar.count()<<std::endl;
12        }
```

Wyniki były zapisywane do pliku .csv (coma-separated values)

Wielkości badanych struktur wahają się między 10.000 a 50.000.000 elementów zależnie od eksperymentu.

3.2 Cele

Badanie skupia się na pomiarze czasu dodawania i usuwania elementów. Inne operacje nie wymagają dynamicznej alokacji i kopiowania pamięci i z tego powodu wykonują się znacznie szybciej. Utrudnia to wykonanie wiarygodnego pomiaru.

3.3 Obserwacje

W przypadku tablic operacje zależą liniowo od aktualnego rozmiaru. Wynika to konieczności kopiowania tabli przy każdej operacji(założenie by tablica zajmowała minimum pamięci). Spodziewany wynik.

Lista ze względu na swoje właściwości (niejednorodne rozmieszczenie w pamięci) posiada czasy operacji niezależne od aktualnego rozmiaru. Wypełnienie całej dostępnej pamięci operacyjnej zajęłoby kilka sekund. Również spodziewany wynik.

Dodanie 50.000.000 elementów do listy zajmuje średnio 2.106700328 sekundy. Struktura przechowująca jeden węzeł ma 24 bajty. Pamięć zużyta do takiej operacji to ok. 1,25 GB.

3.4 Wnioski

Tablica - W przypadku tablic operacje zależą liniowo od aktualnego rozmiaru. Wynika to konieczności kopiowania tabli przy każdej operacji(założenie by tablica zajmowała minimum pamięci). Spodziewany wynik.

Lista ze względu na swoje właściwości (niejednorodne rozmieszczenie w pamięci) posiada czasy operacji niezależne od aktualnego rozmiaru. Wypełnienie całej dostępnej pamięci operacyjnej zajęłoby kilka sekund. Również spodziewany wynik.

W przypadku listy na wykresie widać pewne zaszumienie. Prawdopodobnie ma na nie wpływ przypadkowe obciążenie procesora lub pamięci(którego starano si unikać). Na pozostałych strykturach również występuje ów szum ale ze względu na stałą zależność (O(1)) operacji na liście, wygląda na bardziej chaotyczny.

Kopiec przy dodawaniu elementów rosnących zachowuje się zgodnie z przewidywaniami. Przy dodawaniu losowych oraz malejących trudno zauważyć jakąś tendencję.

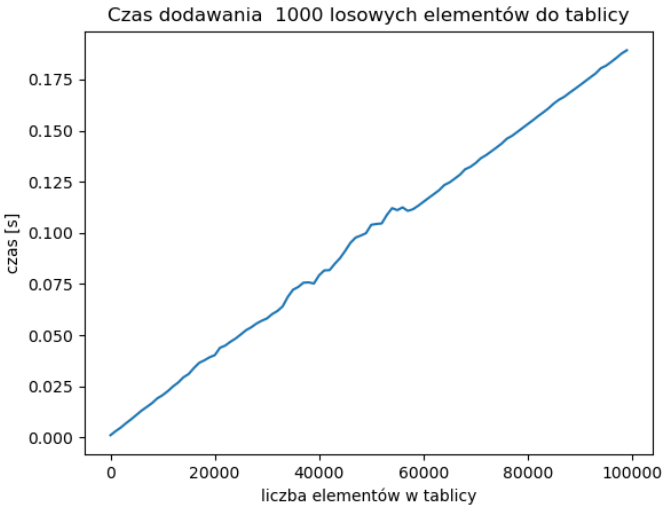
Przy usuwaniu elementów również można dostrzec logarytmiczną krzywą zależności od rozmiaru.

Usuwanie elementów dodanych malejąco wykazuje pewne anomalie(możliwe, że spowodowane innym(niezależnym) obciążeniem komputera).

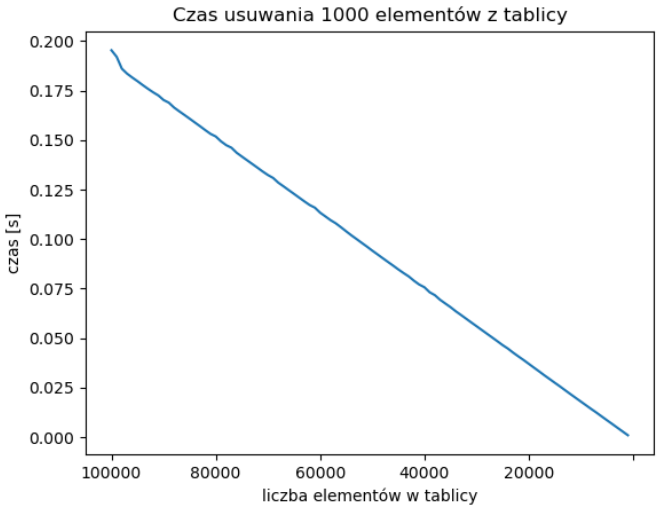
Na niektórych wykresach ucięto (”szpilki”) duże wartości wykraczające poza ogólną tendencję. Wyraźnie większe wartości wynikają z realokacji pamięci. Po przybliżeniu wykresu lepiej widać zależności.

4 Wykresy

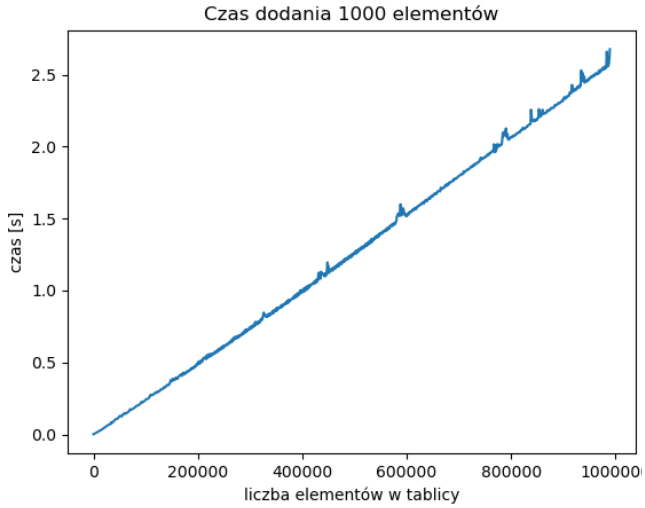
4.0.1 Tablica



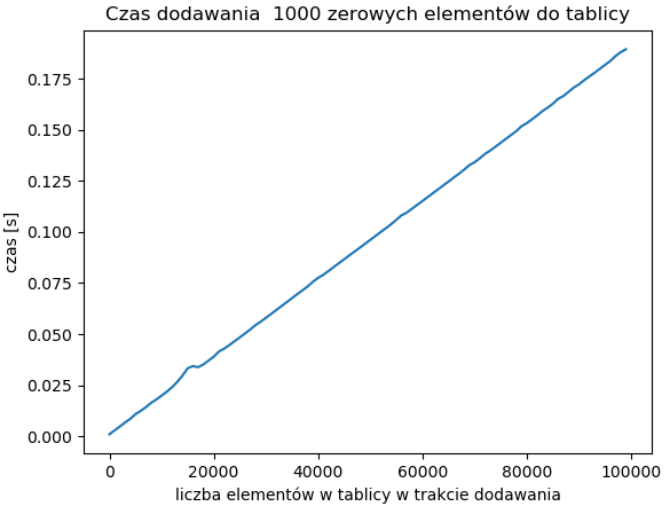
Rys. 1: Tablica - dodawanie losowych elementów



Rys. 2: Tablica - Usuwanie

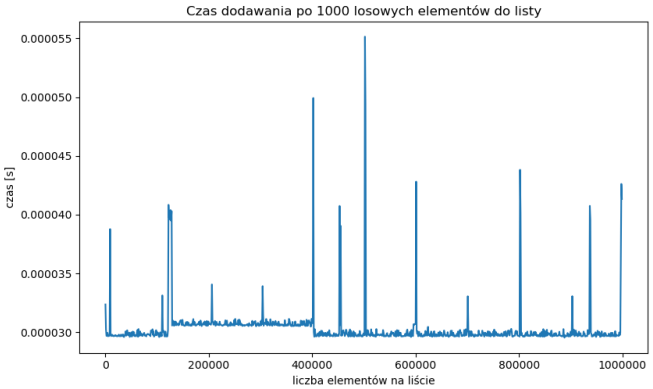


Rys. 3: Tablica - dodawanie losowych elementów

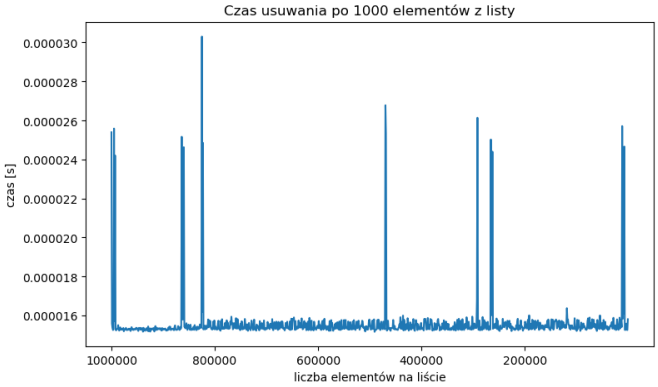


Rys. 4: Tablica - dodawanie zer

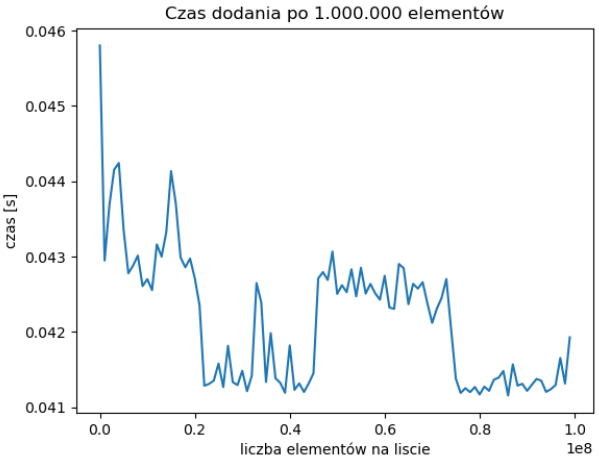
4.0.2 Lista



Rys. 5: Lista - dodawanie losowych

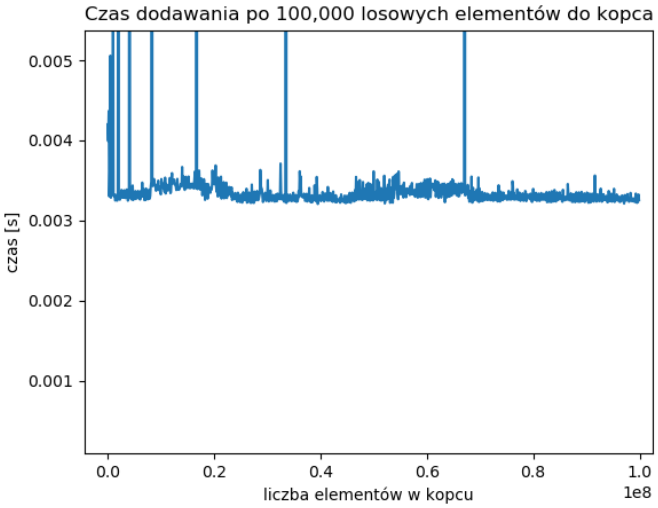


Rys. 6: Lista - usuwanie

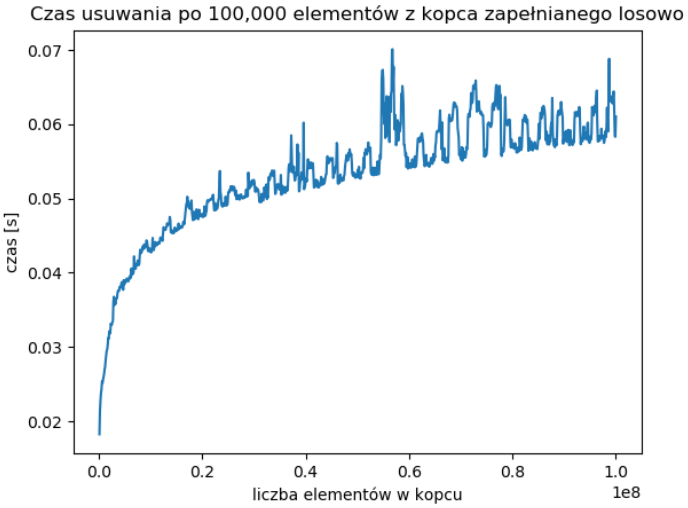


Rys. 7: Lista - dodawanie po 1.000.000 losowychelementów

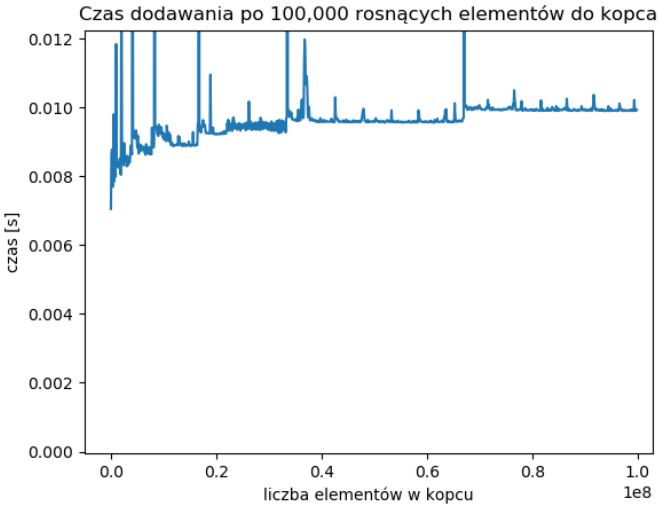
4.0.3 Kopiec



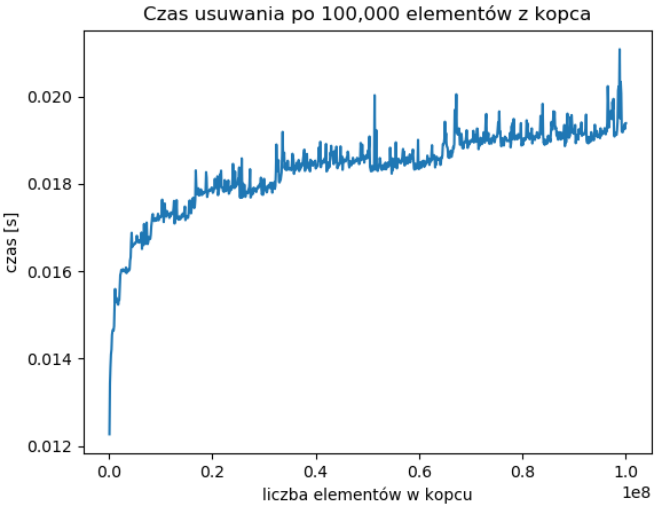
Rys. 8: Kopiec - dodawanie losowych



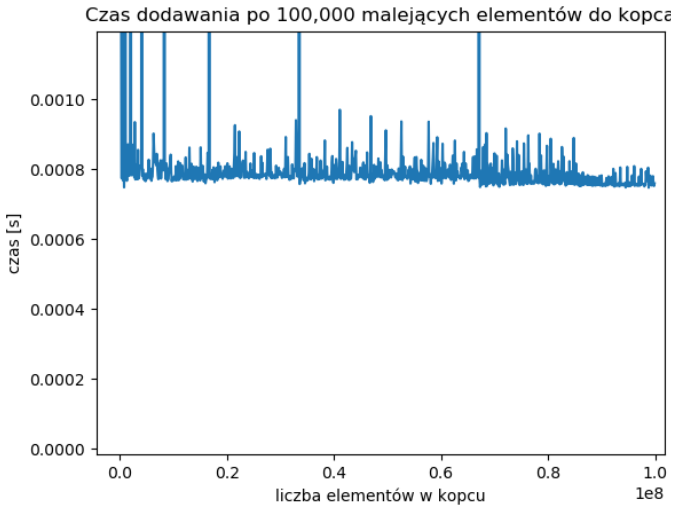
Rys. 9: Kopiec - usuwanie losowych



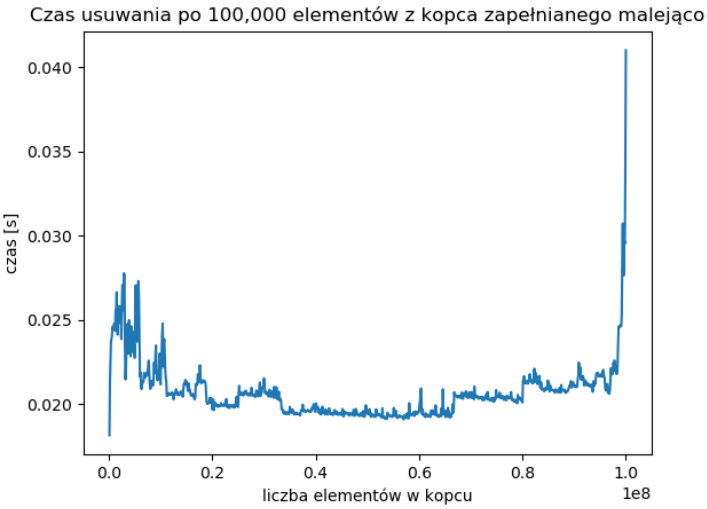
Rys. 10: Kpiec - dodawanie rosnących liczb



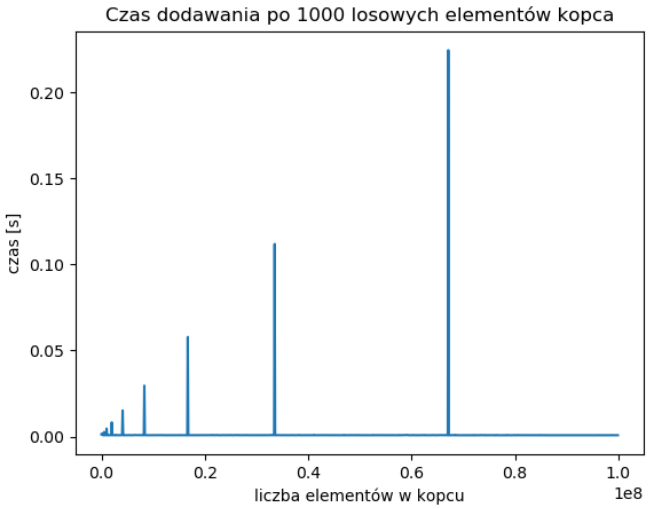
Rys. 11: Kopiec - usuwanie el. dodanych rosnąco



Rys. 12: Kpiec - dodawanie malejących liczb



Rys. 13: Kopiec - usuwanie el. dodanych malejąco



Rys. 14: Kopiec - widzoczna realokacja