

Jakub Plona

Struktury Danych i Złożoność Obliczeniowa

Zadanie projektowe nr 1: Badanie efektywności operacji dodawania, usuwania oraz wyszukiwania elementów w różnych

strukturach danych

07/04/2019

Spis treści

[1. Wstęp teoretyczny 3](#_Toc5560312)

[1.1 Tablica dynamiczna 3](#_Toc5560313)

[1.2. Lista dwukierunkowa 4](#_Toc5560314)

[1.3. Kopiec maksymalny 5](#_Toc5560315)

[1.4. Drzewo czerwono-czarne 6](#_Toc5560316)

[2. Plan eksperymentu 7](#_Toc5560317)

[3. Wyniki 8](#_Toc5560318)

[3.1. Tablica 8](#_Toc5560319)

[3.1.1. Wstawianie elementu na początek 8](#_Toc5560320)

[3.1.2. Wstawianie elementu na koniec 9](#_Toc5560321)

[3.1.3. Wstawianie elementu na losową pozycję 11](#_Toc5560322)

[3.1.4. Usuwanie elementu z początku 12](#_Toc5560323)

[3.1.5. Usuwanie elementu z końca 14](#_Toc5560324)

[3.1.6. Usuwanie elementu z losowej pozycji 15](#_Toc5560325)

[3.1.7. Wyszukiwanie elementu 17](#_Toc5560326)

[3.2. Lista 18](#_Toc5560327)

[3.2.1. Wstawianie elementu na początek 18](#_Toc5560328)

[3.2.2. Wstawianie elementu na koniec 20](#_Toc5560329)

[3.2.3. Wstawianie elementu na losową pozycję 21](#_Toc5560330)

[3.2.4. Usuwanie elementu z początku 23](#_Toc5560334)

[3.2.5. Usuwanie elementu z końca 24](#_Toc5560335)

[3.2.6. Usuwanie elementu z losowej pozycji 26](#_Toc5560336)

[3.2.7. Wyszukiwanie elementu 27](#_Toc5560337)

[3.3. Kopiec 29](#_Toc5560338)

[3.3.1. Dodawanie klucza 29](#_Toc5560339)

[3.3.2. Usuwanie klucza 30](#_Toc5560340)

[3.3.3. Wyszukiwanie klucza 32](#_Toc5560341)

[3.4. Drzewo czerwono-czarne 33](#_Toc5560342)

[3.4.1. Dodawanie klucza 33](#_Toc5560343)

[3.4.2. Usuwanie klucza 35](#_Toc5560344)

[3.4.3. Wyszukiwanie klucza 36](#_Toc5560345)

[4. Podsumowanie 38](#_Toc5560346)

# Wstęp teoretyczny



Przeprowadzony eksperyment miał na celu zbadanie efektywności podstawowych operacji (często nazywanych też słownikowymi) na strukturach danych takich, jak tablica dynamiczna, lista (dwukierunkowa z wartownikiem), kopiec (maksymalny) oraz drzewo czerwono-czarne. Są to: dodawanie elementu do struktury, usuwanie elementu ze struktury oraz wyszukiwanie elementu w strukturze.

Do opisu czasu działania tychże operacji posłużono się asymptotyczną notacją dużego O.

## Tablica dynamiczna

Tablica jest reprezentowana w pamięci komputera jako spójny obszar pamięci. Aby uzyskać dostęp do takiej struktury, koniecznym jest posiadanie adresu jej początku. Z założenia elementy w tablicy są tego samego typu, dlatego dostęp do elementu jest natychmiastowy (gdy znamy jego indeks), gdyż adres tego elementu jest wyliczany na podstawie wzoru:

adres początkowy + rozmiar elementu \* indeks elementu

W przeprowadzonych badaniach zaimplementowano i posłużono się tablicą dynamiczną o zamortyzowanym koszcie operacji na jej końcu równym O(1).

* Wstawianie

Zaimplementowana tablica dynamiczna zwiększa swój rozmiar dwukrotnie, jeżeli podczas próby wstawienia elementu okaże się, że struktura jest pełna. Zamortyzowany koszt tej operacji wynosi O(1).

• Początek tablicy

Jeżeli tablica nie jest pusta, to wszystkie jej elementy są przesuwane o jedną pozycję do przodu. Po operacji warunkowej element jest wstawiany na pozycję o indeksie 0. Złożoność tej operacji z racji przesuwania tablicy wynosi O(n).

• Koniec tablicy

Element zostaje dodany na koniec tablicy po jej ewentualnym rozszerzeniu.

Koszt tej operacji wynosi O(1).

• Losowe miejsce tablicy

Element zostaje wstawiony na losowo wybraną pozycję tablicy (elementy od wybranej pozycji począwszy są przesuwane w prawo o jedno miejsce).

Koszt tej operacji wynosi O(n).

* Usunięcie

Zaimplementowana tablica dynamiczna zmniejsza swój rozmiar dwukrotnie, jeżeli podczas próby usunięcia elementu okaże się, że struktura jest w ¾ pusta. Zamortyzowany koszt tej operacji wynosi O(1).

• Początek tablicy

Jeżeli tablica nie jest pusta, to wszystkie jej elementy są przesuwane o jedną pozycję do tyłu. Złożoność tej operacji z racji przesuwania tablicy wynosi O(n).

• Koniec tablicy

W tym przypadku zostaje zmniejszony jedynie rozmiar tablicy. Koszt tej operacji wynosi O(1).

• Losowe miejsce tablicy

Element zostaje usunięty z losowo wybranej pozycji tablicy poprzez przesunięcie elementów dalszych o jedną pozycję w lewo. Koszt tej operacji wynosi O(n).

* Wyszukiwanie

Zaimplementowano klasyczne wyszukiwanie liniowe. Koszt znalezienia elementu jest opisywany przez O(n).

## Lista dwukierunkowa

List dwukierunkowa ze strażnikiem jest wskaźnikową strukturą danych. Kolejno dodawane do niej elementy poza kluczem zawierają jeszcze wskaźniki na elementy: poprzedni, następny. Wartownik jest specjalnym elementem tablicy zawsze w niej obecnym – jest to element pośredni pomiędzy pierwszym, a ostatnim wpisem na liście. Z racji obecności wartownika, lista jest tak naprawdę cykliczna, gdyż strażnik łączy ze sobą element pierwszy i ostatni.

Koszt obsługi strażnika jest określony przez O(1).

* Wstawianie

• Początek listy

Operacja ta polega na przepisaniu wskaźników w taki sposób, aby wstawiany element był wskazywany jako następnym przez strażnika, zaś poprzedni pierwszy element był zaraz za nowo wstawianym. Złożoność tej operacji wynosi O(1).

• Koniec listy

Operacja ta jest identyczna (z dokładnością do symetrii) z wstawianiem na początek listy, gdyż ostatni element jest wskazaniem poprzedniego w polu strażnika. Złożoność tej operacji wynosi O(1).

• Losowe miejsce na liście

Różnica między wstawianiem w losowe miejsce a wstawianiem na początek jest taka, że w tym wypadku element bazowy najpierw trzeba znaleźć (nie jest już nim strażnik). Złożoność tej operacji wynosi O(n)

* Usunięcie

• Początek listy

Operacja ta polega na przepisaniu wskaźników w taki sposób, aby następnik i poprzednik (strażnik) usuwanego elementu był ze sobą połączone. Złożoność tej operacji wynosi O(1).

• Koniec listy

Operacja ta jest identyczna (z dokładnością do symetrii) z usuwaniem z początku listy. Różnica jest taka, iż strażnik jest teraz następnikiem. Złożoność tej operacji wynosi O(1).

• Losowe miejsce na liście

Przed wykonaniem tej operacji należy najpierw znaleźć element do usunięcia. Pozostała część algorytmu jest taka sama, jak w przypadku usuwania z końca listy. Złożoność tej operacji wynosi O(n).

* Wyszukanie

Aby znaleźć element należy przeszukać całą listę. Złożoność tej operacji wynosi O(n).

## Kopiec maksymalny

Kopiec jest tablicową strukturą danych która posiada tzw. własność kopca, zaś jej elementy są indeksowane w specyficzny sposób. Dla kopca maksymalnego własność kopca brzmi następująco: wartość przodka jest niemniejsza od wartości potomka. Indeksowanie elementów polega na wyliczaniu indeksów rodziców i dzieci węzła na podstawie określonych wzorów. I tak dla rodzica jest to |\_i/2\_|, dla lewego dziecka 2i, zaś dla prawego dziecka 2i + 1, gdzie i jest indeksem węzła z zakresu [1…n]. Dzięki takiemu indeksowaniu kopiec można przedstawiać w formie drzewa. Do implementacji kopca użyto zaimplementowanej wcześniej tablicy dynamicznej.

* Wstawianie

Wstawienie elementu do kopca polega na dołączeniu go na koniec i pięciu się w górę drzewa w celu znalezienia takiej pozycji dla nowego węzła, że wstawiając go na nią zachowana będzie własność kopca.

Złożoność tej operacji wynosi O(h), gdzie h oznacza wysokość kopca liczoną ze wzoru:   
lgn (gdzie n – liczba elementów w kopcu).

* Usuwanie

Operacja ta polega na przepisaniu ostatniego węzła do usuwanego i przywróceniu własności kopca (element może poruszać się w górę, bądź w dół zależnie od sytuacji).

Złożoność tej operacji wynosi O(h).

* Wyszukiwanie

Kopiec musi być przeszukany w całości, przy usprawnieniu, iż nie ma potrzeby przeszukiwać poddrzew węzła jeżeli jego wartość jest mniejsza od poszukiwanego elementu (w kopcu maksymalnym głębiej znajdziemy jedynie elementy mniejsze z uwagi na własność kopca). Złożoność tej operacji wynosi jednak O(n), gdyż usprawnienie może jedynie zmienić stały współczynnik w funkcji złożoności obliczeniowej.

## Drzewo czerwono-czarne

Drzewo czerwono-czarne jest drzewem poszukiwań binarnych, które spełnia tzw. warunki drzewa czerwono-czarnego.

Warunki te są następujące:

1. Wszystkie węzły w drzewie są czerwone albo czarne
2. Korzeń drzewa jest zawsze czarny
3. Liście drzewa są zawsze czarne
4. Jeśli węzeł jest czerwony, to obaj jego synowie są czarni
5. Każda prosta ścieżka od danego węzła do jego liści potomnych zawiera tę samą liczbę węzłów czarnych.

Drzewo poszukiwań binarnych spełniających ww. warunki jest drzewem zrównoważonym. W przypadku drzewa czerwono-czarnego oznacza to, że jego wysokość nie przekroczy dwukrotnej wartości wysokości minimalnej.

* Wstawianie

Wstawienie elementu do drzewa czerwono-czarnego polega na wstawieniu elementu zgodnie z algorytmem wstawiania dla drzewa poszukiwań binarnych oraz przywrócenia warunków drzewa czerwono-czarnego. Czas potrzebny na wykonanie wstawienia do drzewa poszukiwań binarnych jest ograniczony przez O(lgn) (n – liczba elementów w drzewie), zaś przywrócenie warunków drzewa jest wykonywane w czasie stałym. Dlatego złożoność obliczeniowa tej operacji wynosi O(lgn).

* Usuwanie

Operacja ta polega na usunięciu elementu z drzewa za pomocą algorytmu usuwania elementu z drzewa poszukiwań binarnych oraz przywróceniu własności drzewa czerwono-czarnego. Tak jak w przypadku wstawiania pierwsza część operacji jest wykonywana w czasie O(lgn), zaś druga jest ograniczona przez O(1).

Złożoność tej operacji wynosi więc O(lgn).

* Wyszukiwanie

Wyszukiwanie w drzewie czerwono-czarnym jest tym samym algorytmem, który jest stosowany w przypadku drzewa poszukiwań binarnych. Polega na przemieszczaniu się w głąb drzewa na podstawie decyzji wynikłych z porównań poszukiwanego klucza i klucza w aktualnym węźle (jest to możliwe ze względu na warunek drzewa poszukiwań binarnych: lewy potomek jest mniejszy niż przodek, zaś prawy potomek jest niemniejszy niż przodek). Złożoność tej operacji wynosi O(lgn).

# Plan eksperymentu



* Wykorzystany język programowania to C++11
* Badanie poszczególnych operacji zostało przeprowadzone na losowo generowanych zestawach danych zawierających klucze z przedziałów:   
  *[0, INT\_MAX/2]*, *[INT\_MAX/2, INT\_MAX]*, *[0, INT\_MAX]*, *[0,100] oraz [INT\_MAX – 100, INT\_MAX]*
* Losowe liczby z podanych przedziałów były generowane przy użyciu biblioteki *random*
* Badania przeprowadzano dla kolejnych rozmiarów struktur: *0, 1, 2, …, 20000*
* Ilość powtórzeń wykonania każdej operacji dla danego rozmiaru i wyniosła 100 (generowano nową populację)
* Otrzymanym wynikiem była suma zmierzonych czasów ze wszystkich powtórzeń dla danej operacji podzielona przez ilość powtórzeń (średni czas dla wykonanej operacji przy ustalonym rozmiarze dla różnych populacji)
* Otrzymane wyniki zaprezentowano w postaci surowej (pełne dane na wykresie) oraz uwzględniając jedynie tendencje wzrostowe danych (wykorzystano linie trendu)
* Czas wykonania operacji mierzony był przy użyciu funkcji QueryPerformanceFrequency() oraz QeryPerformanceCounter()
* Wszystkie struktury były alokowane dynamicznie

# Wyniki



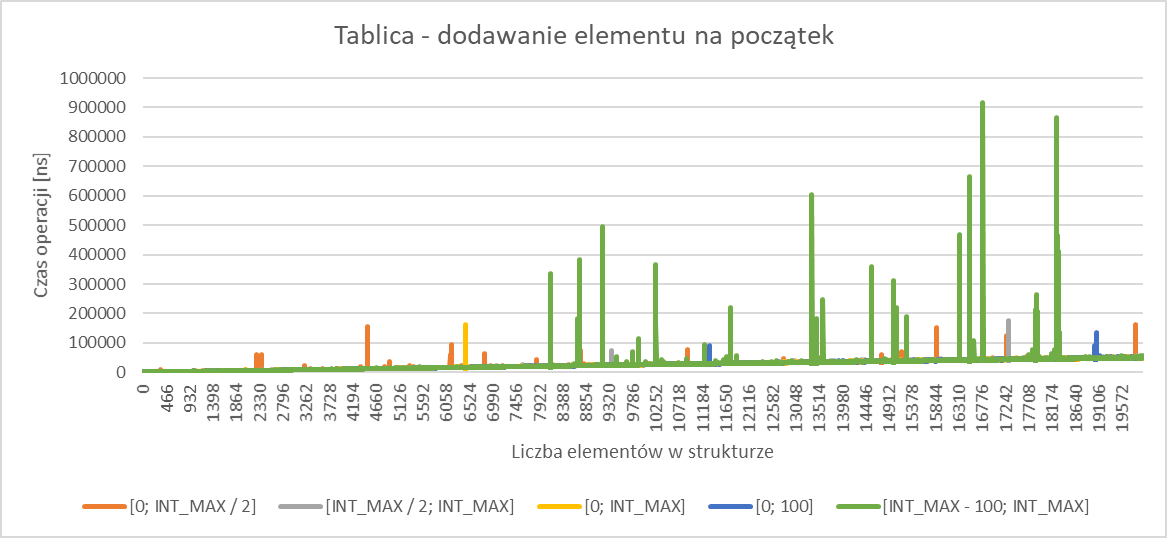
Czasy wykonywania poszczególnych operacji podane zostały w nanosekundach.

## Tablica

### Wstawianie elementu na początek

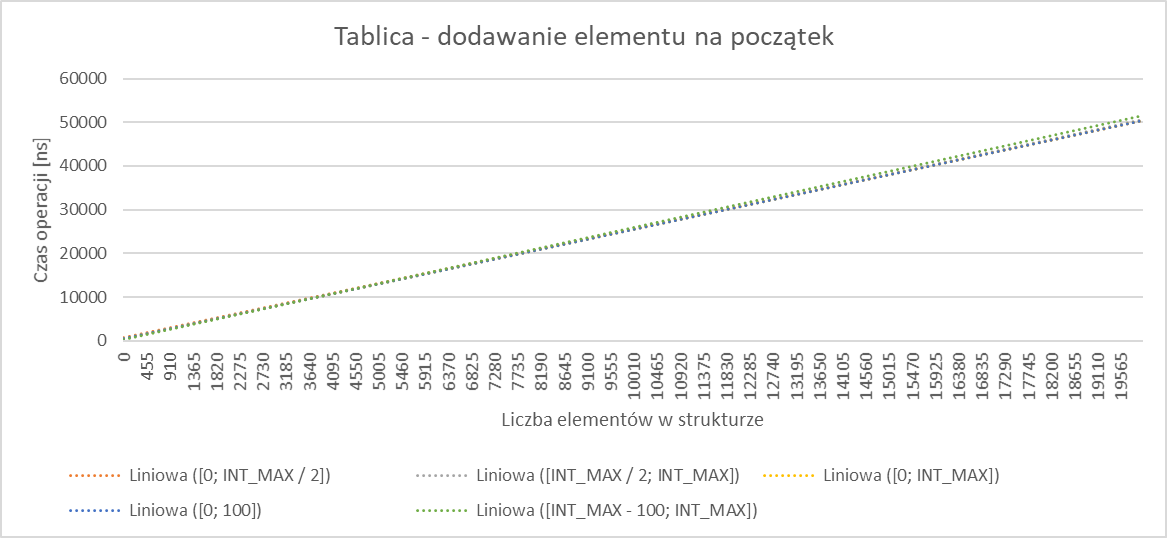






Rysunek 1 Wykres zawierający wszystkie punkty pomiarowe dla operacji wstawiania elementu na początek tablicy

Tabela 1 Czas [ns] wstawiania elementu na początek tablicy w zależności od l. elementów i wartości kluczy (równomierny wybór 20 punktów pomiarowych)



Rysunek 2 Linie trendu dla operacji wstawiania elementu na początek tablicy

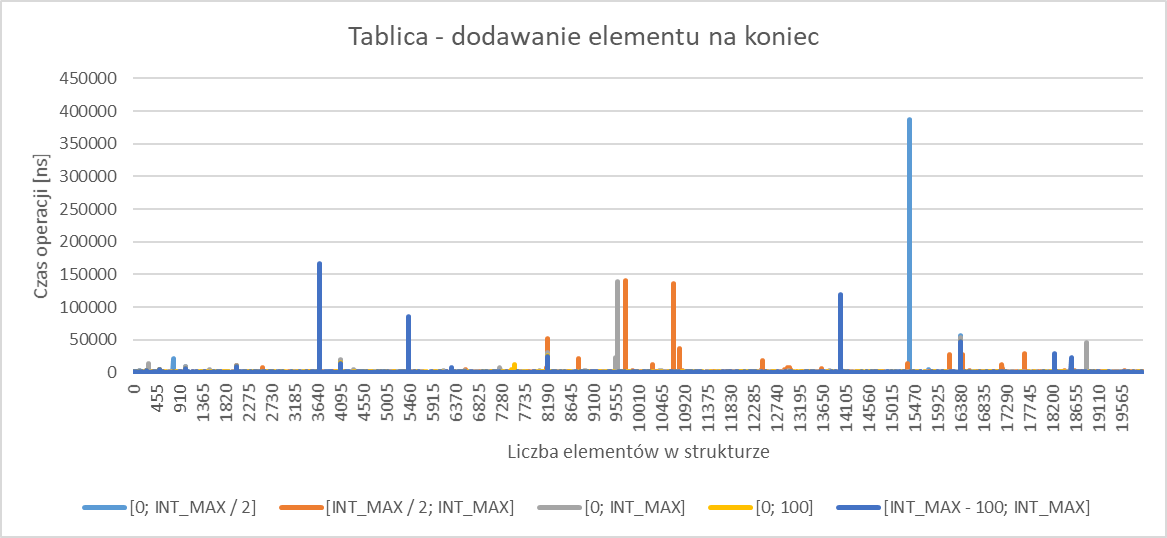
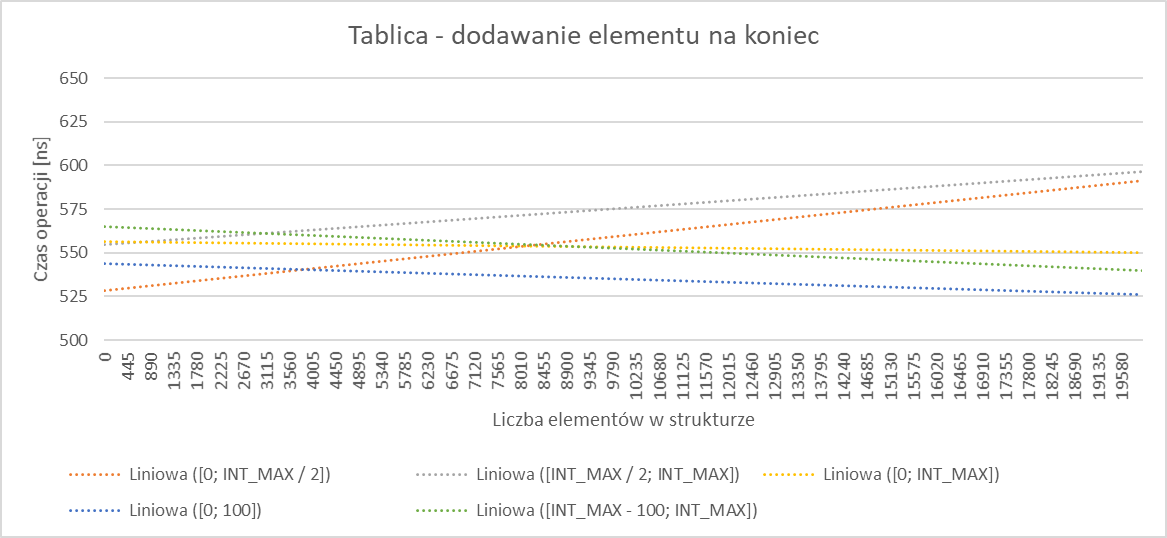
Otrzymane wyniki są zgodne z założeniami teoretycznymi. Brak zauważalnych różnic dla różnych zakresów kluczy.

### Wstawianie elementu na koniec





Tabela 2 Czas [ns] wstawiania elementu na koniec tablicy w zależności od l. elementów i wartości kluczy (równomierny wybór 20 punktów pomiarowych)



Rysunek 3 Linie trendu dla operacji wstawiania elementu na koniec tablicy

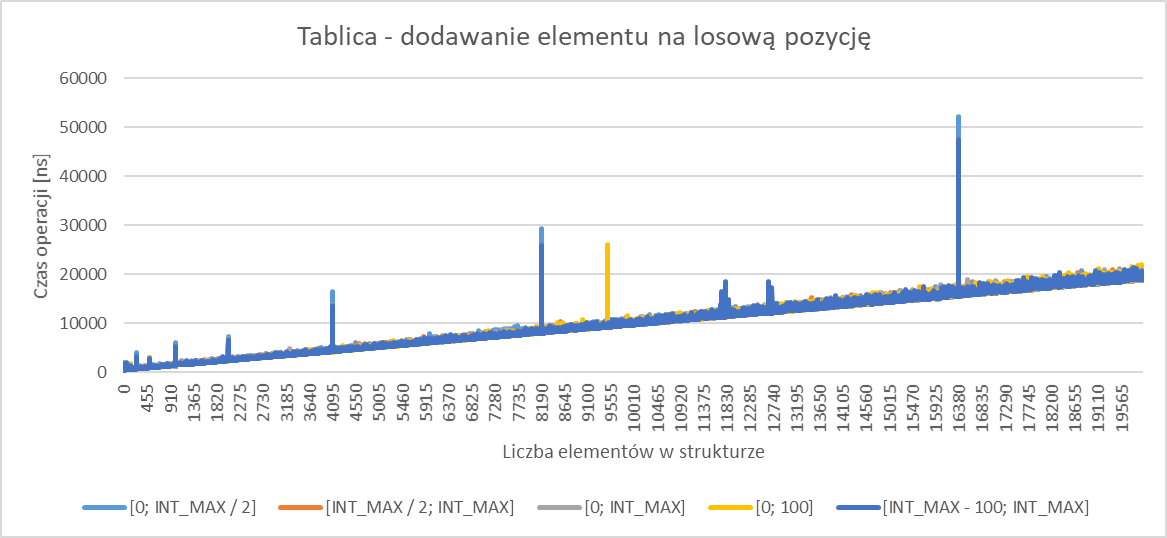
Rysunek 4 Wykres zawierający wszystkie punkty pomiarowe dla operacji wstawiania elementu na koniec tablicy

Otrzymane wyniki są zgodne z założeniami teoretycznymi. Brak zauważalnych różnic dla różnych zakresów kluczy.

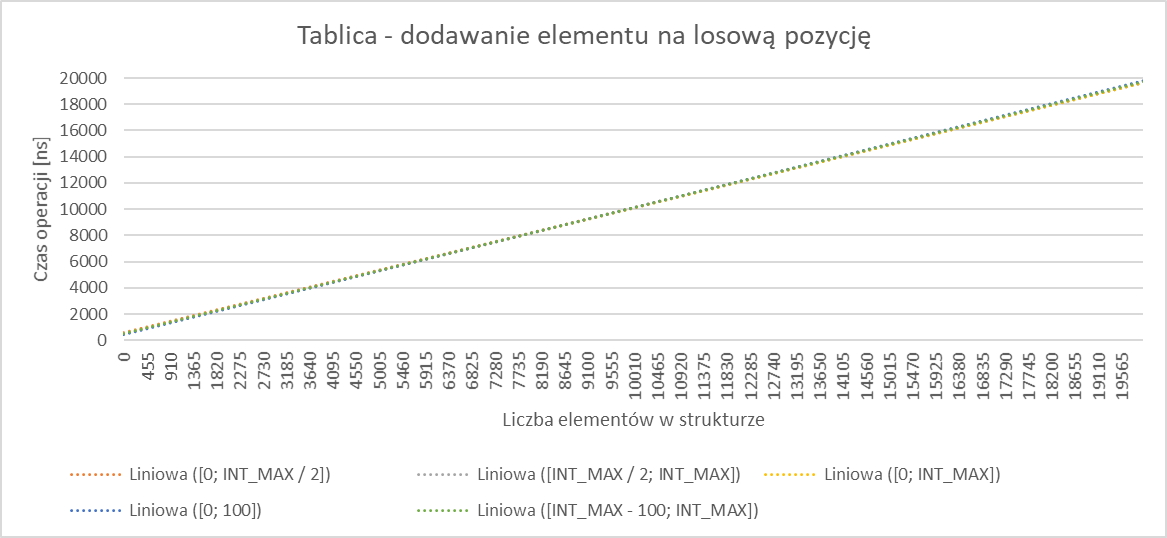
### Wstawianie elementu na losową pozycję



Tabela 3 Czas [ns] wstawiania elementu w losowe miejsce tablicy w zależności od l. elementów i wartości kluczy (równomierny wybór 20 punktów pomiarowych)



Rysunek 5 Wykres zawierający wszystkie punkty pomiarowe dla operacji wstawiania elementu w losowe miejsce tablicy



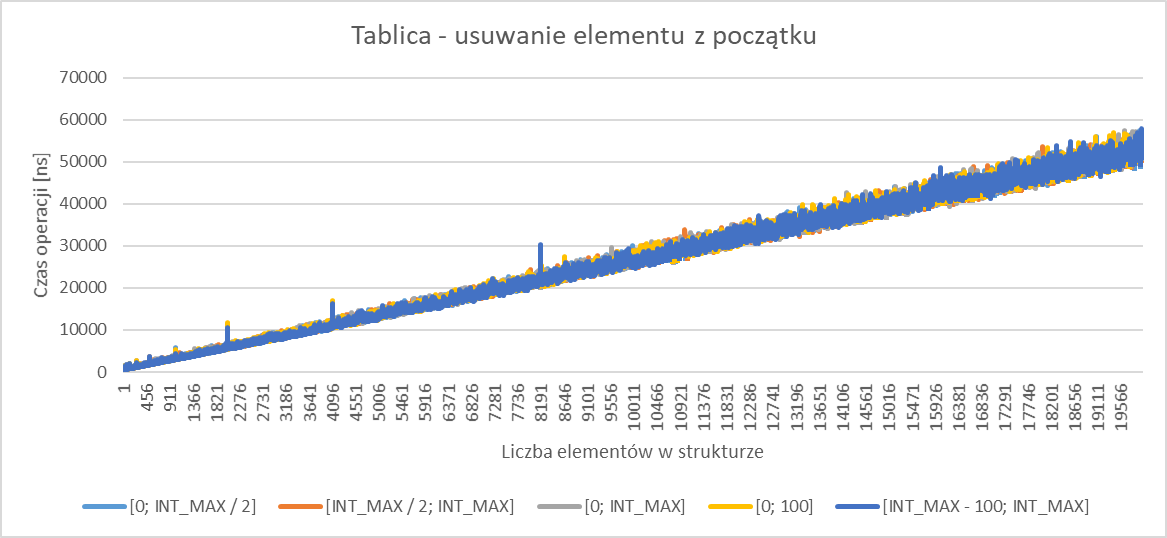
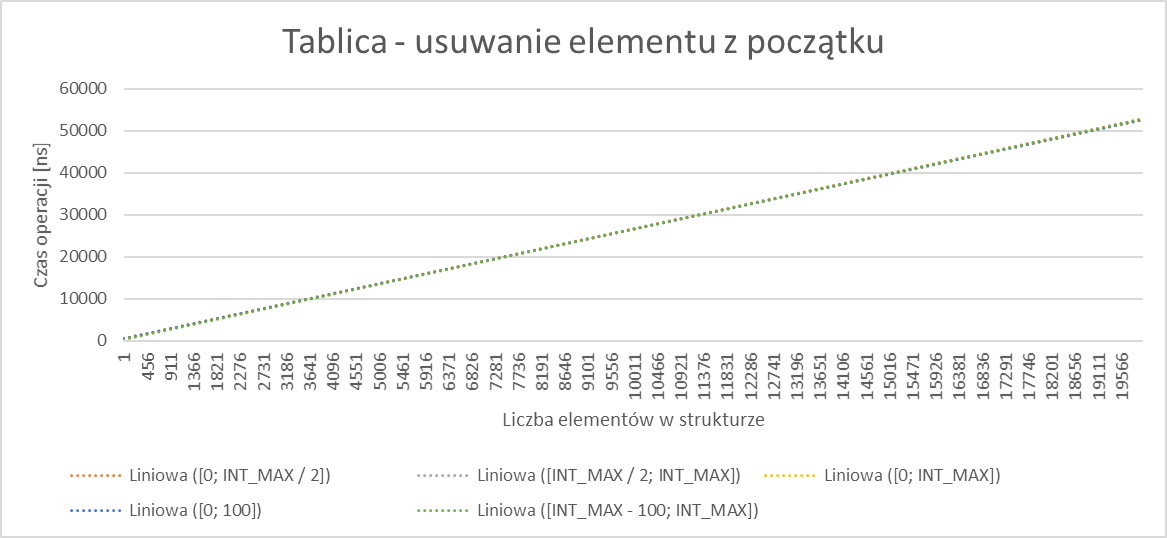
Rysunek 6 Linie trendu dla operacji wstawiania elementu w losowe miejsce tablicy

Otrzymane wyniki są zgodne z założeniami teoretycznymi. Brak zauważalnych różnic dla różnych zakresów kluczy.

### Usuwanie elementu z początku



Tabela 4 Czas [ns] usuwania elementu z początku tablicy w zależności od l. elementów i wartości kluczy (równomierny wybór 20 punktów pomiarowych)



Rysunek 7 Linie trendu dla operacji usuwania elementu z początku tablicy

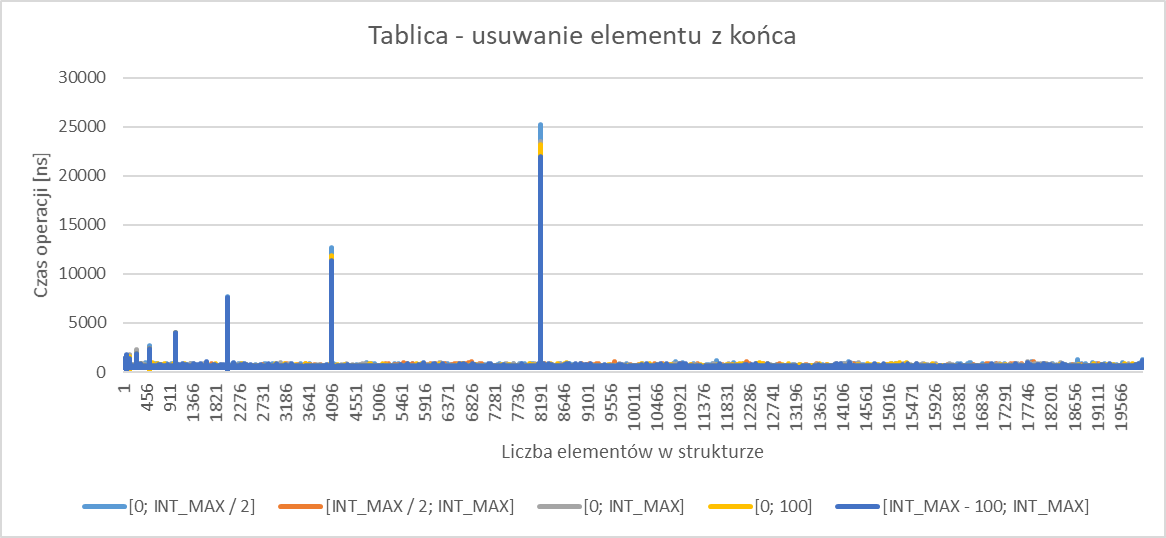
Rysunek 8 Wykres zawierający wszystkie punkty pomiarowe dla operacji usuwania elementu z początku tablicy

Otrzymane wyniki są zgodne z założeniami teoretycznymi. Brak zauważalnych różnic dla różnych zakresów kluczy.

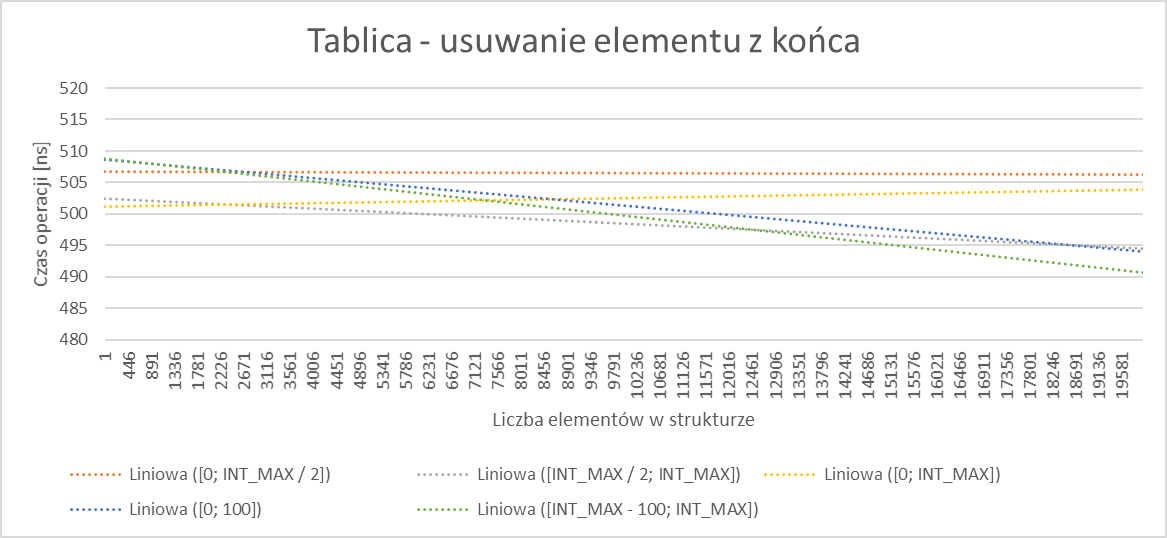
### Usuwanie elementu z końca



Tabela 5 Czas [ns] usuwania elementu z końca tablicy w zależności od l. elementów i wartości kluczy (równomierny wybór 20 punktów pomiarowych)



Rysunek 9 Wykres zawierający wszystkie punkty pomiarowe dla operacji usuwania elementu z końca tablicy



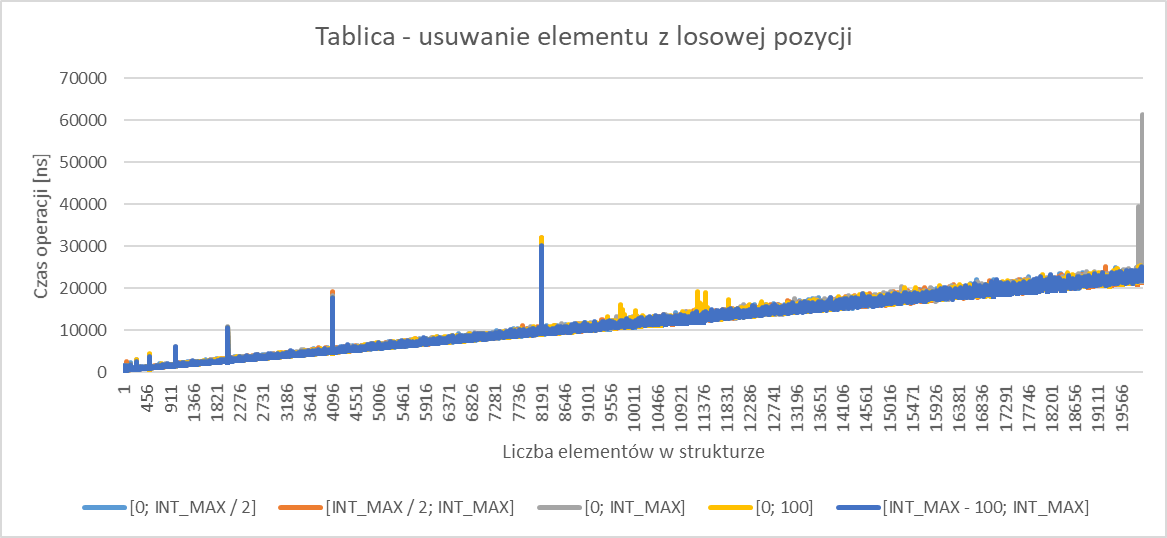
Rysunek 10 Linie trendu dla operacji usuwania elementu z końca tablicy

Otrzymane wyniki są zgodne z założeniami teoretycznymi. Brak zauważalnych różnic dla różnych zakresów kluczy.

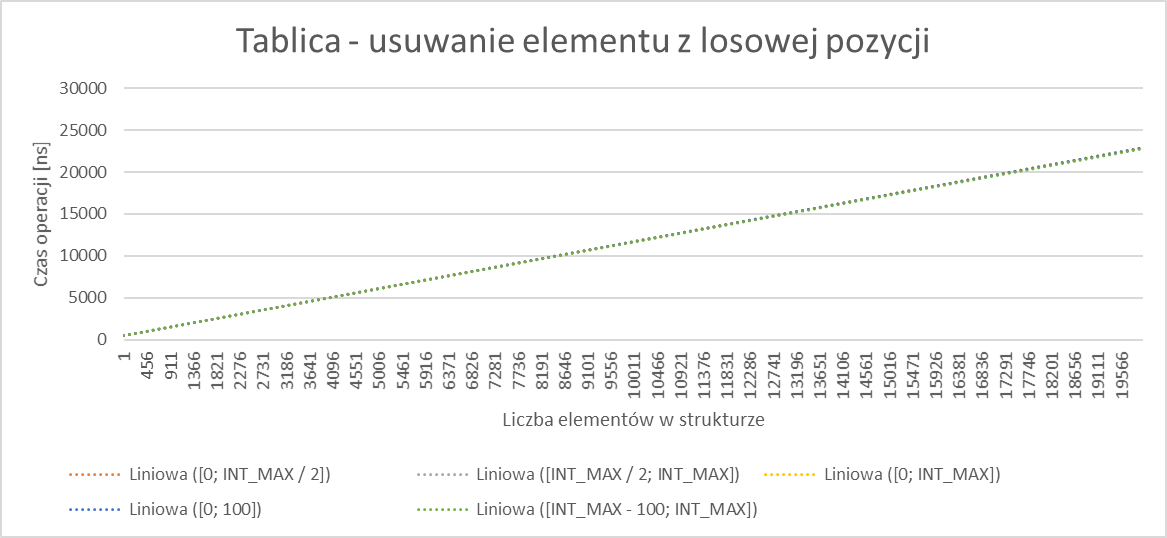
### Usuwanie elementu z losowej pozycji



Tabela 6 Czas [ns] usuwania elementu z losowego miejsce tablicy w zależności od l. elementów i wartości kluczy (równomierny wybór 20 punktów pomiarowych)



Rysunek 11 Wykres zawierający wszystkie punkty pomiarowe dla operacji usuwania elementu z losowego miejsca w tablicy



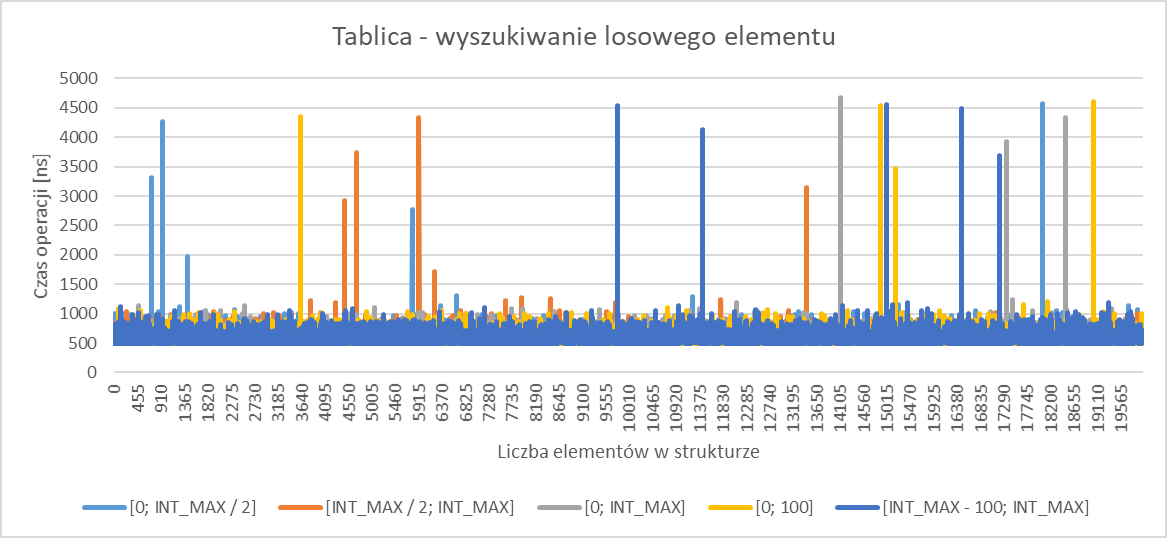
Rysunek 12 Linie trendu dla operacji usuwania elementu z losowego miejsca w tablicy

Otrzymane wyniki są zgodne z założeniami teoretycznymi. Brak zauważalnych różnic dla różnych zakresów kluczy.

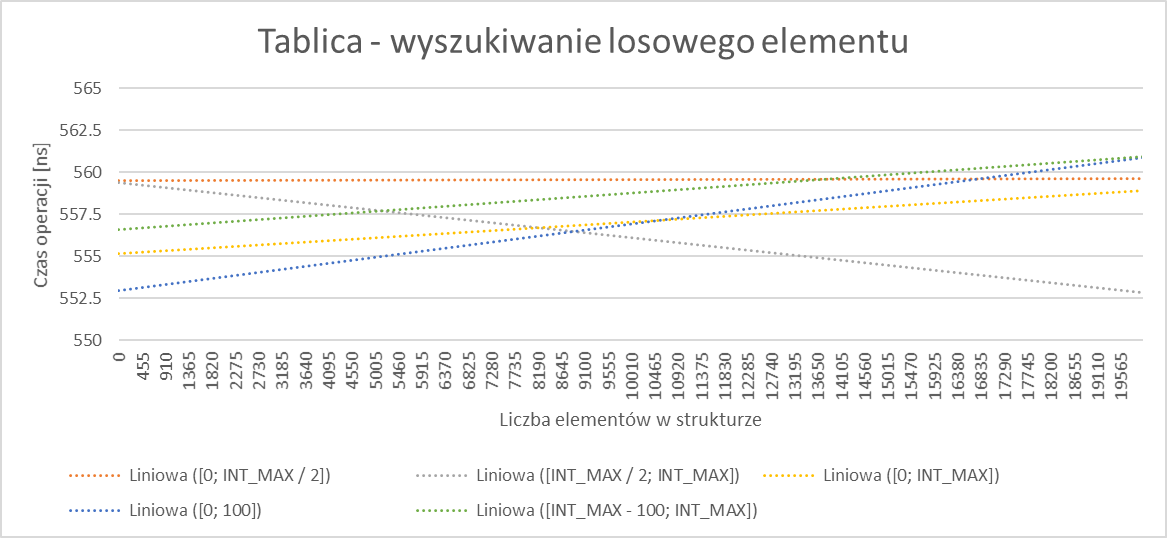
### Wyszukiwanie elementu



Tabela 7 Czas [ns] wyszukiwania elementu w tablicy w zależności od l. elementów i wartości kluczy (równomierny wybór 20 punktów pomiarowych)



Rysunek 13 Wykres zawierający wszystkie punkty pomiarowe dla operacji wyszukiwania elementu w tablicy



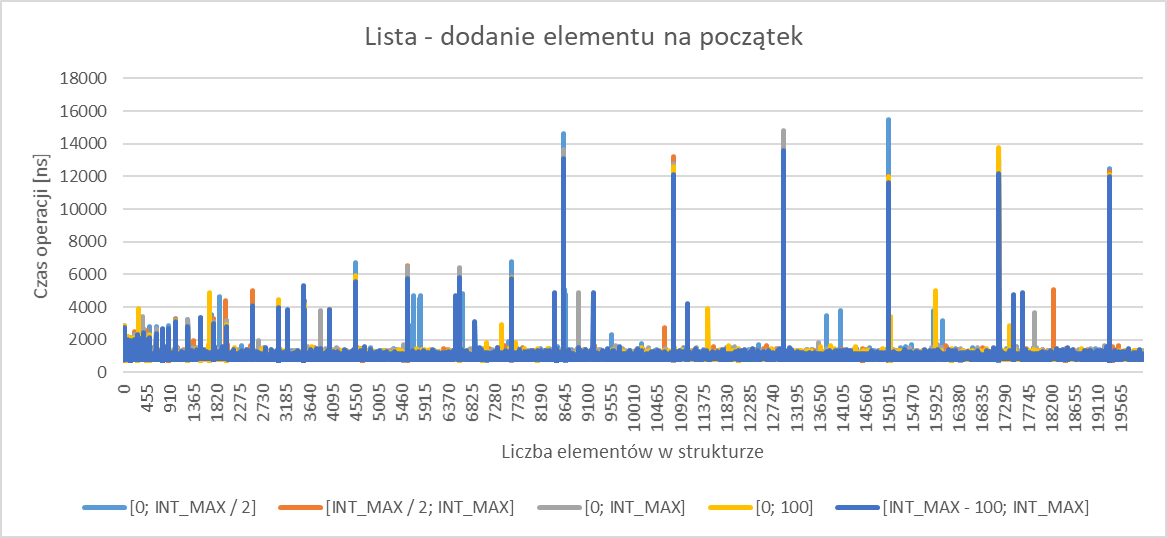
Rysunek 14 Linie trendu dla operacji wyszukiwania elementu w tablicy

Otrzymane wyniki różnią się od założeń teoretycznych. Czas wyszukiwania elementu w tej strukturze okazał być się stały (O(1)) - w granicach niepewności pomiarowych. Pomiary powtórzono dla 1.000.000 elementów, jednak uzyskane wyniki nadal wskazywały na stały czas operacji. Może być to skutek zastosowanych rozwiązań w architekturze procesora (Intel), które pozwoliły na bardzo wydajny odczyt danych z pamięci (wyszukiwanie nie modyfikuje elementów w strukturze).

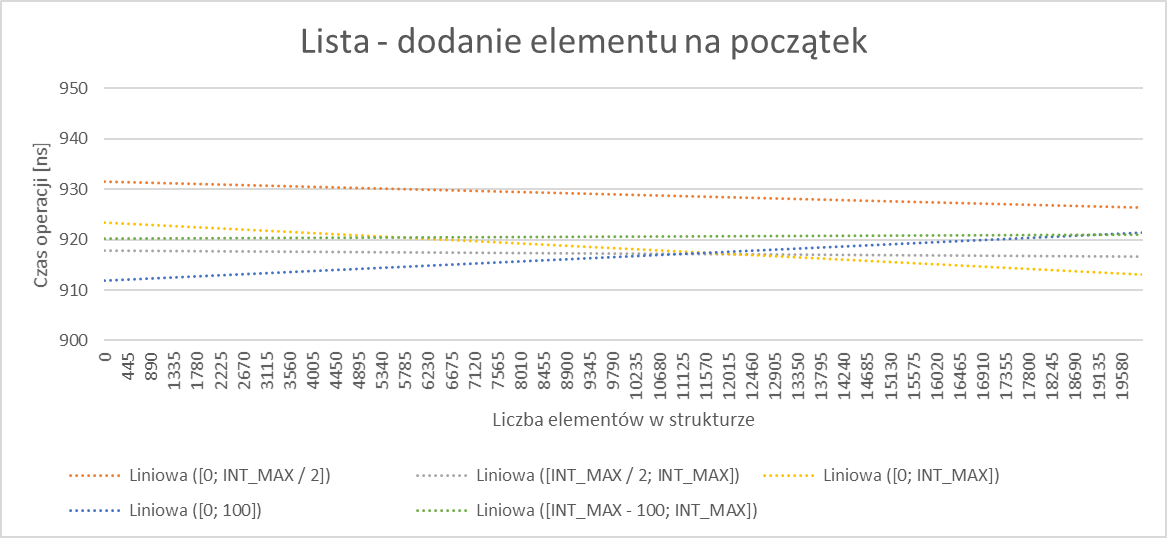
## Lista

### Wstawianie elementu na początek

Tabela 8 Czas [ns] wstawiania elementu na początek listy w zależności od l. elementów i wartości kluczy (równomierny wybór 20 punktów pomiarowych)



Rysunek 15 Wykres zawierający wszystkie punkty pomiarowe dla operacji wstawiania elementu na początek listy



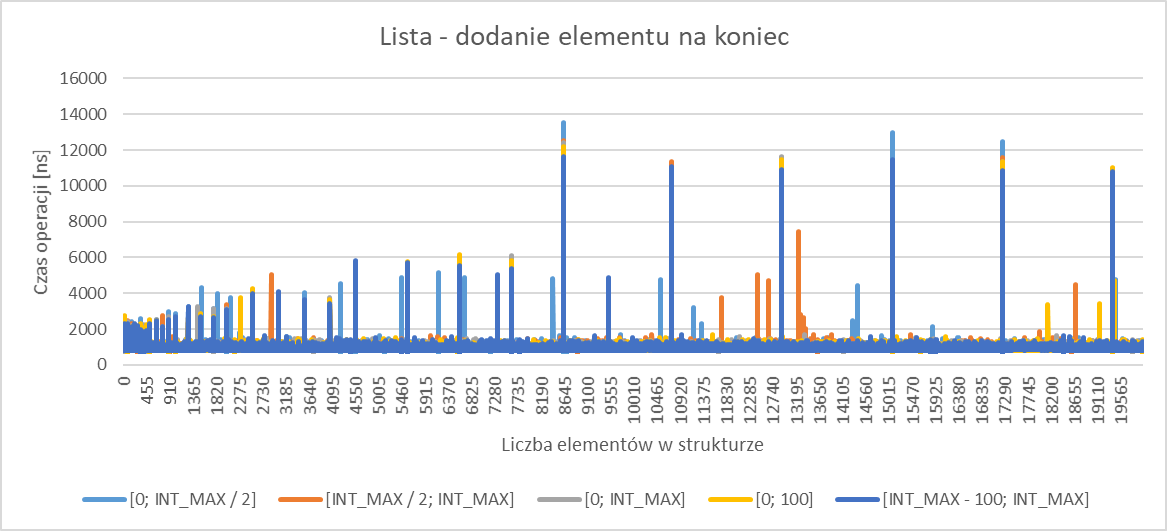
Rysunek 16 Linie trendu dla operacji wstawiania elementu na początek listy

Otrzymane wyniki są zgodne z założeniami teoretycznymi. Brak zauważalnych różnic dla różnych zakresów kluczy.

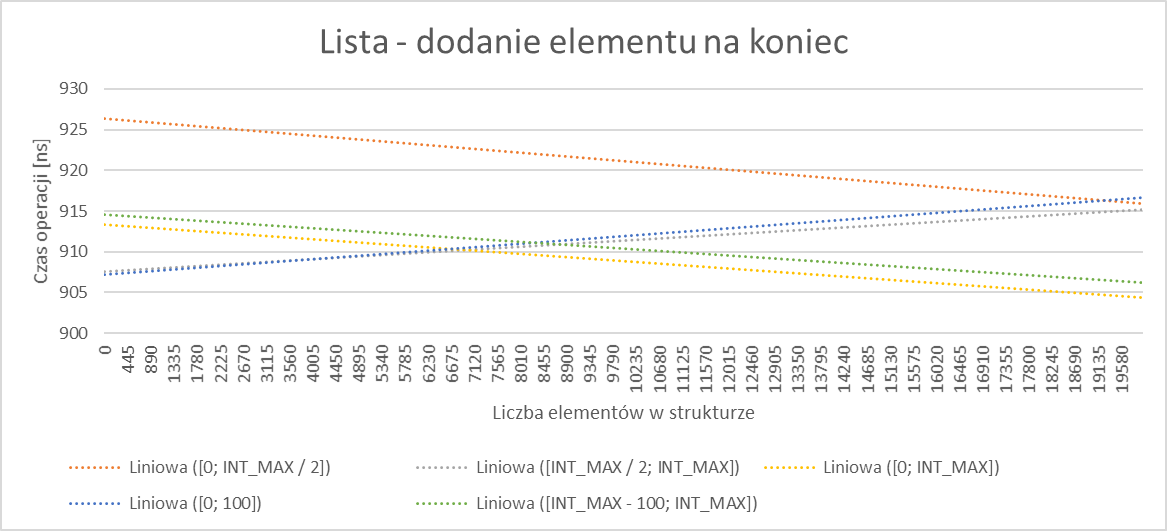
### Wstawianie elementu na koniec



Tabela 9 Czas [ns] wstawiania elementu na koniec listy w zależności od l. elementów i wartości kluczy (równomierny wybór 20 punktów pomiarowych)



Rysunek 17 Wykres zawierający wszystkie punkty pomiarowe dla operacji wstawiania elementu na koniec listy

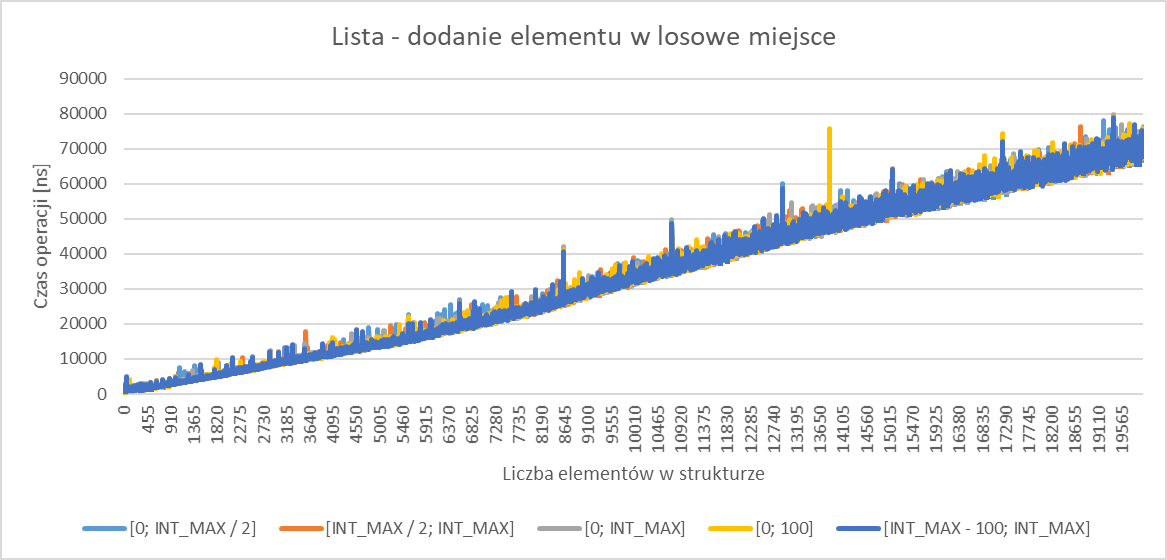
Otrzymane wyniki są zgodne z założeniami teoretycznymi. Brak zauważalnych różnic dla różnych zakresów kluczy.

Rysunek 18 Linie trendu dla operacji wstawiania elementu na koniec listy

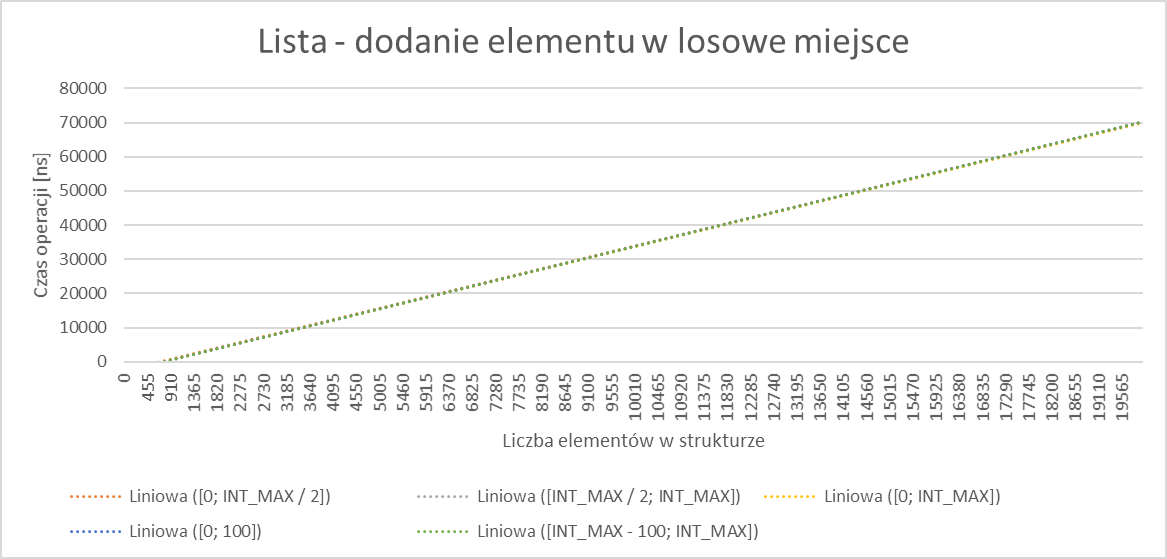
### Wstawianie elementu na losową pozycję



Tabela 10 Czas [ns] wstawiania elementu na losową pozycję listy w zależności od l. elementów i wartości kluczy (równomierny wybór 20 punktów pomiarowych)



Rysunek 19 Wykres zawierający wszystkie punkty pomiarowe dla operacji wstawiania elementu na losową pozycję listy



Rysunek 20 Linie trendu dla operacji wstawiania elementu na losową pozycję listy

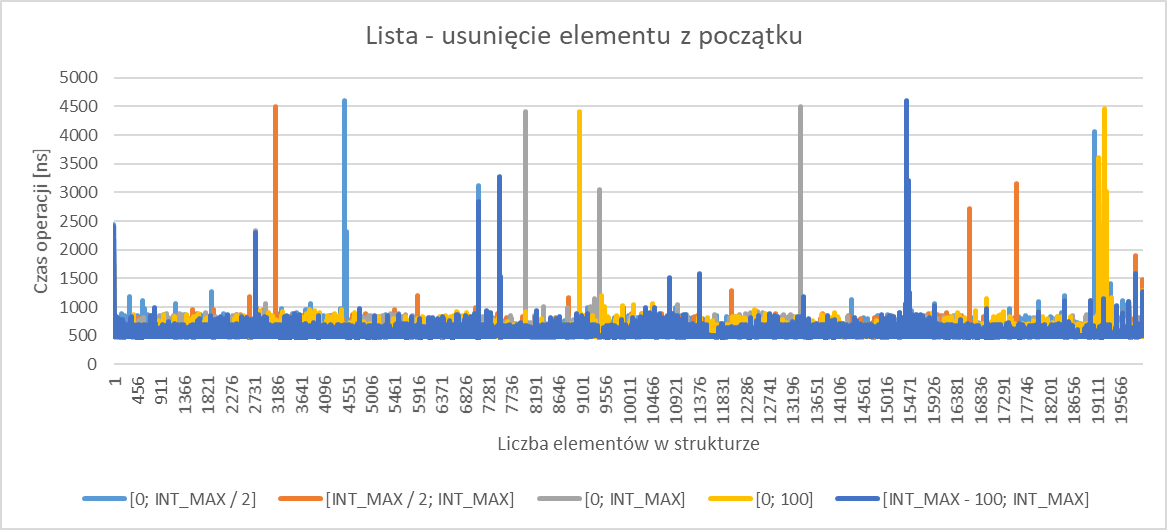
Otrzymane wyniki są zgodne z założeniami teoretycznymi. Brak zauważalnych różnic dla różnych zakresów kluczy.



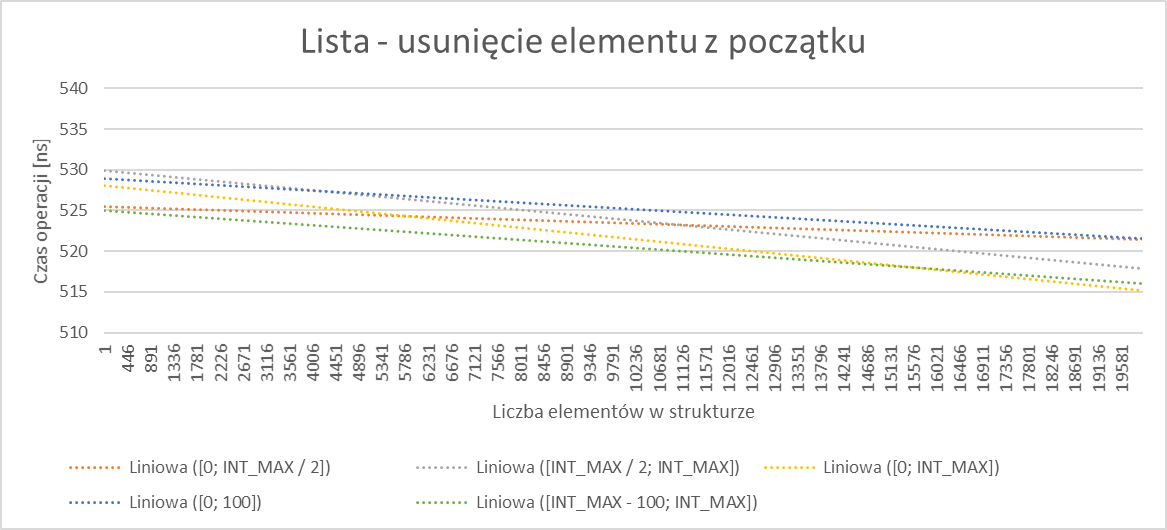
### Usuwanie elementu z początku



Tabela 11 Czas [ns] usuwania elementu z początku listy w zależności od l. elementów i wartości kluczy (równomierny wybór 20 punktów pomiarowych)



Rysunek 21 Wykres zawierający wszystkie punkty pomiarowe dla operacji usuwania elementu z początku listy

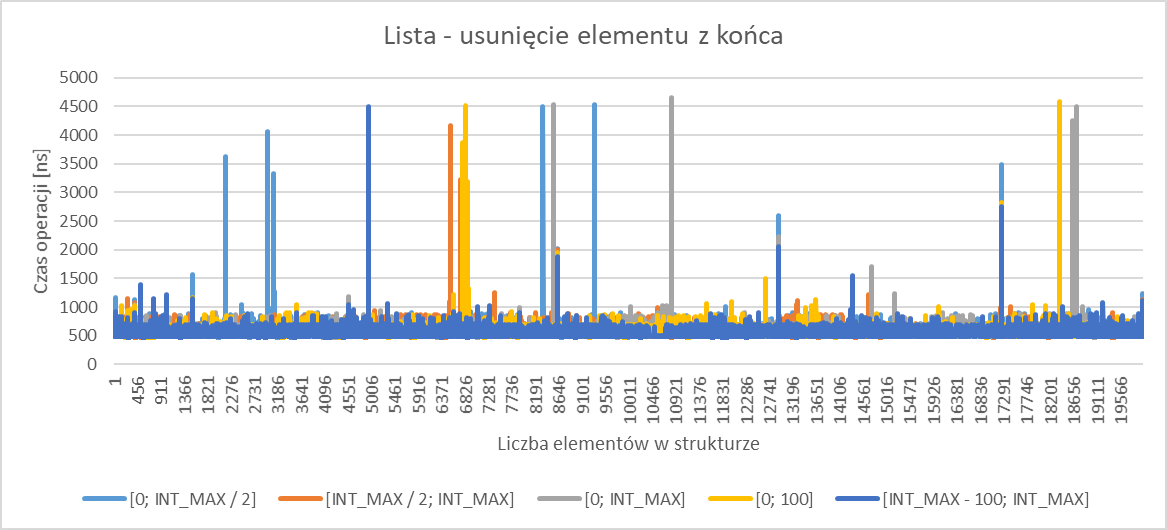
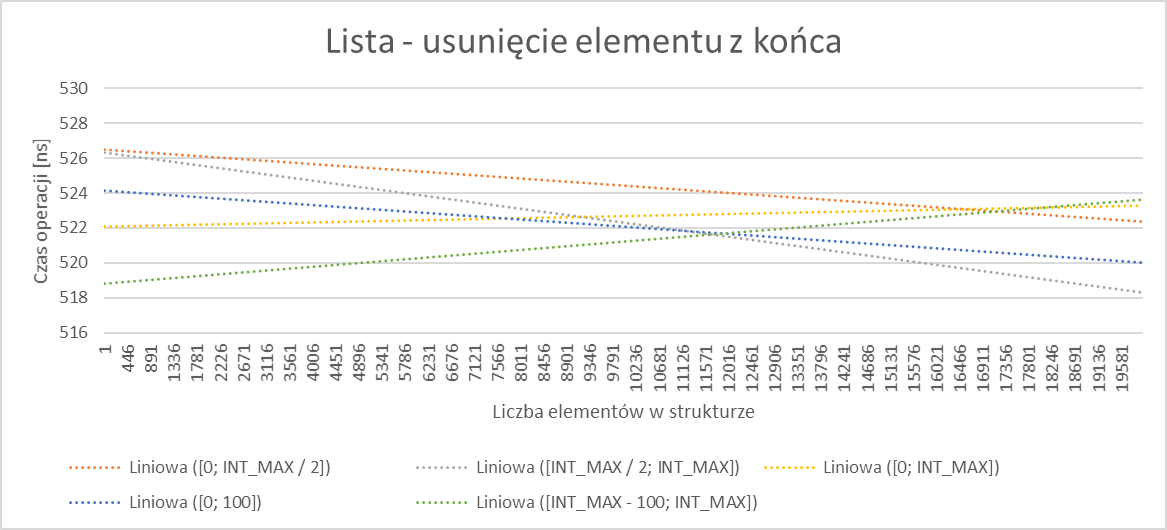


Rysunek 22 Linie trendu dla operacji usuwana z początku listy

Otrzymane wyniki są zgodne z założeniami teoretycznymi. Brak zauważalnych różnic dla różnych zakresów kluczy.

### Usuwanie elementu z końca

Tabela 12 Czas [ns] usuwania elementu z końca listy w zależności od l. elementów i wartości kluczy (równomierny wybór 20 punktów pomiarowych)



Rysunek 23 Linie trendu dla operacji usuwana z końca listy

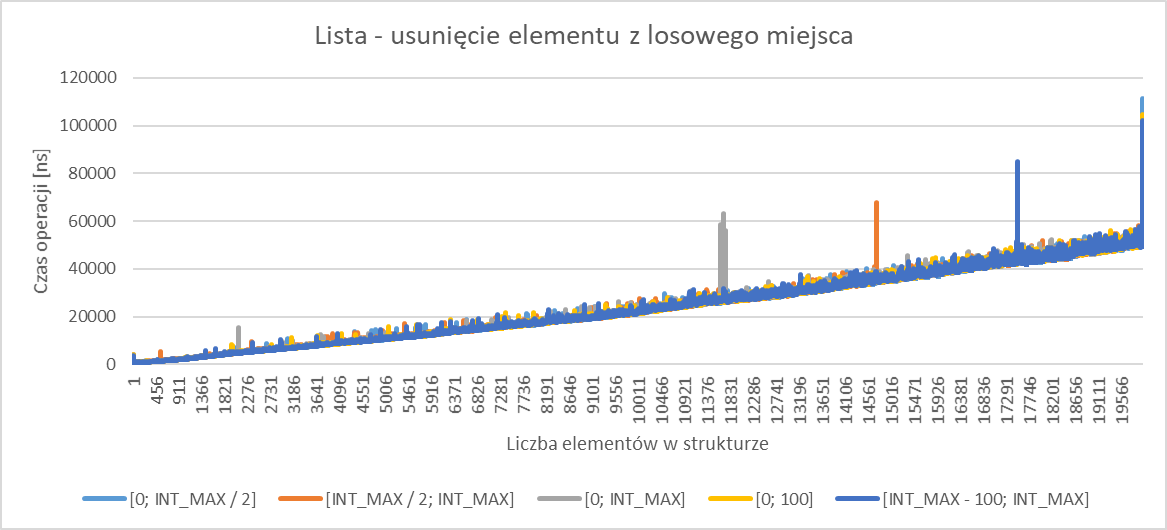
Rysunek 24 Wykres zawierający wszystkie punkty pomiarowe dla operacji usuwania elementu z końca listy

Otrzymane wyniki są zgodne z założeniami teoretycznymi. Brak zauważalnych różnic dla różnych zakresów kluczy.

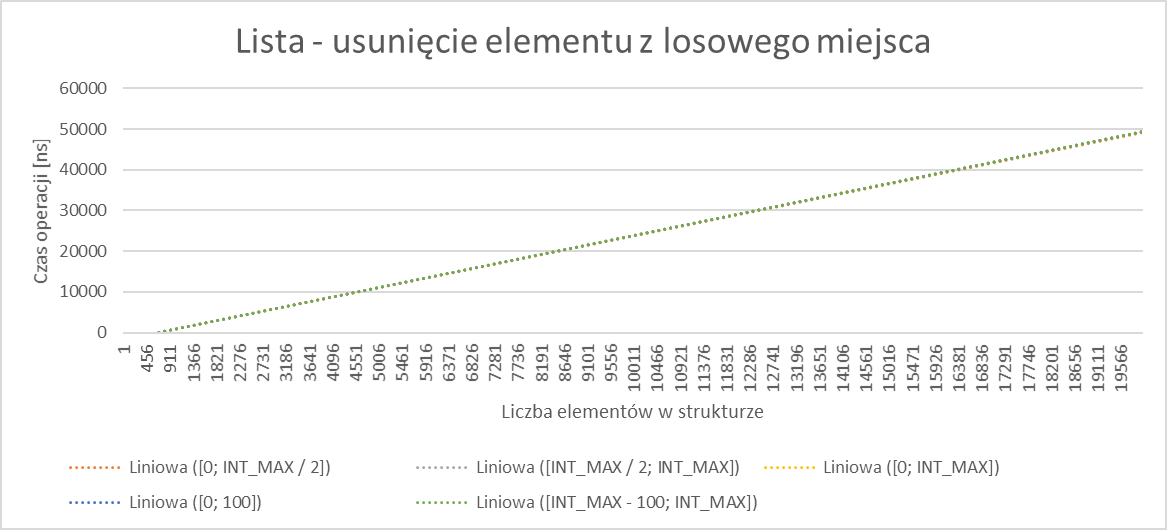
### Usuwanie elementu z losowej pozycji



Tabela 13 Czas [ns] usuwania elementu z losowej pozycji listy w zależności od l. elementów i wartości kluczy (równomierny wybór 20 punktów pomiarowych)



Rysunek 25 Wykres zawierający wszystkie punkty pomiarowe dla operacji usuwania elementu z losowej pozycji listy



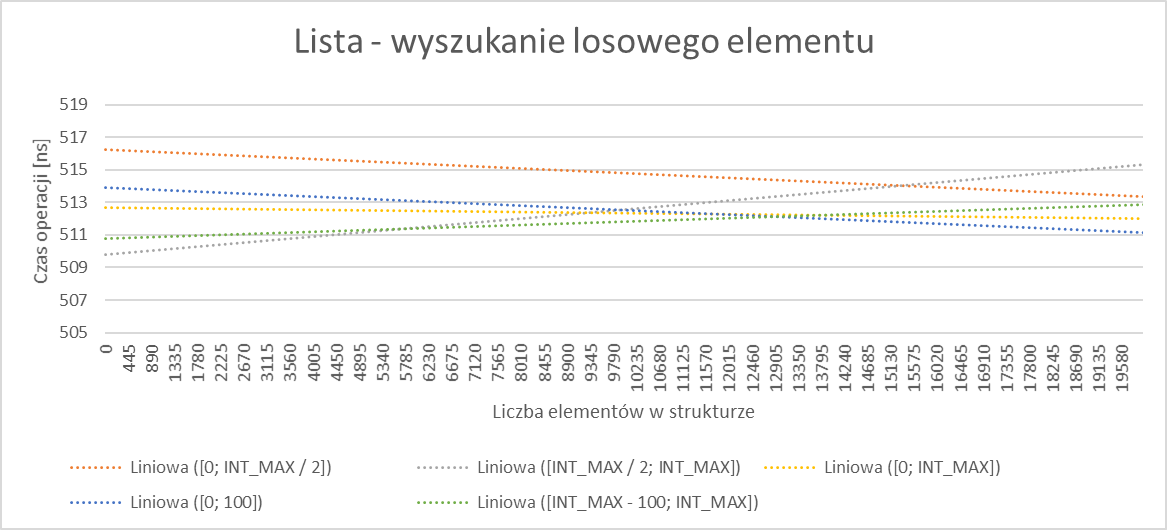
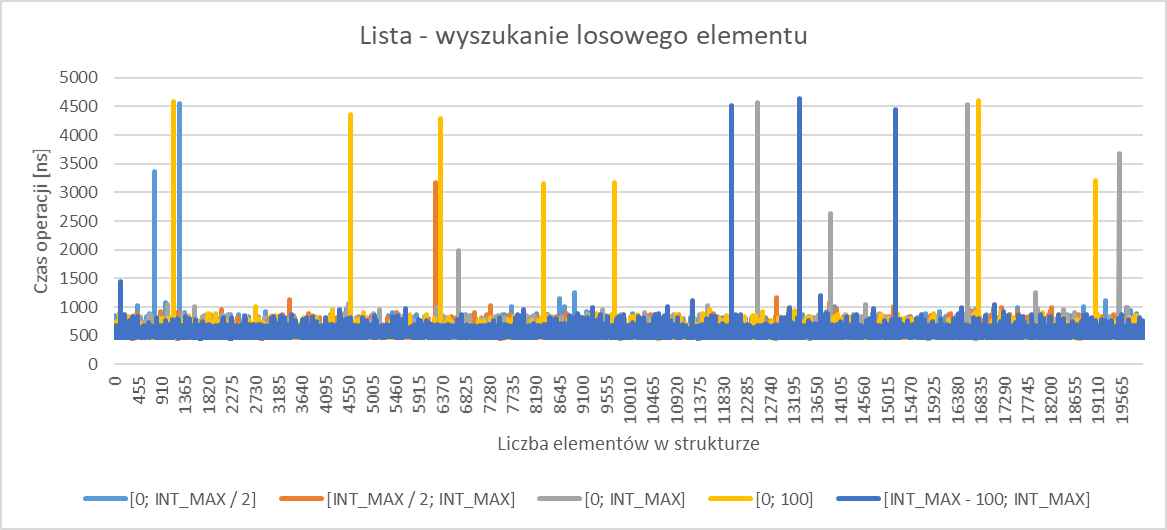
Rysunek 26 Linie trendu dla operacji usuwana z losowej pozycji listy

Otrzymane wyniki są zgodne z założeniami teoretycznymi. Brak zauważalnych różnic dla różnych zakresów kluczy.

### Wyszukiwanie elementu



Tabela 14 Czas [ns] wyszukiwania elementu na liście w zależności od l. elementów i wartości kluczy (równomierny wybór 20 punktów pomiarowych)

Otrzymane wyniki różnią się od założeń teoretycznych. Czas wyszukiwania elementu w tej strukturze okazał być się stały (O(1)) - w granicach niepewności pomiarowych. Pomiary powtórzono dla 1.000.000 elementów, jednak uzyskane wyniki nadal wskazywały na stały czas operacji. Może być to skutek zastosowanych rozwiązań w architekturze procesora (Intel), które pozwoliły na bardzo wydajny odczyt danych z pamięci (wyszukiwanie nie modyfikuje elementów w strukturze).

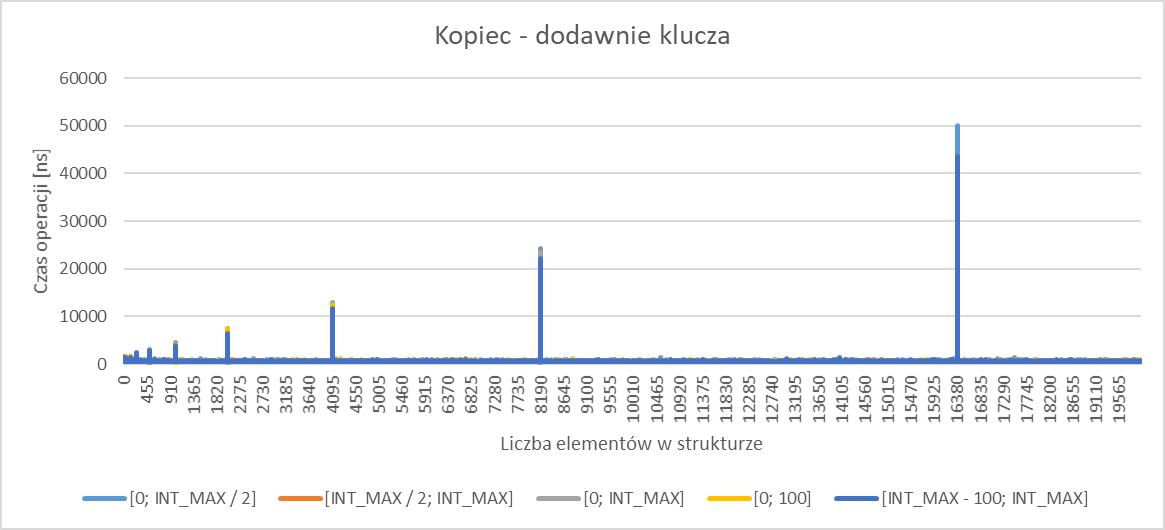
Rysunek 27 Linie trendu dla operacji wyszukiwania elementu na liście

Rysunek 28 Wykres zawierający wszystkie punkty pomiarowe dla operacji wyszukiwania elementu na liście

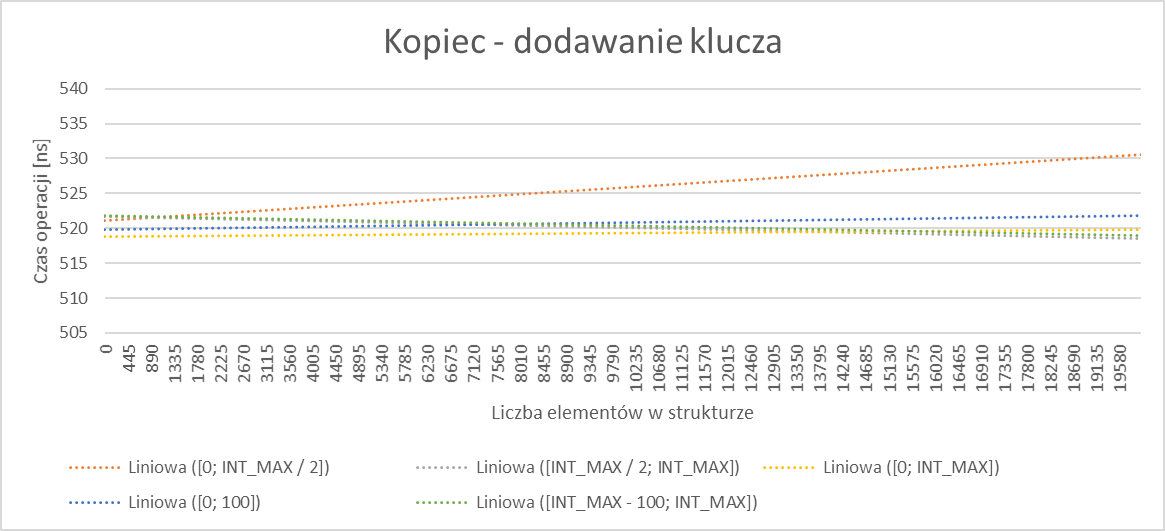
## Kopiec

### Dodawanie klucza

Tabela 15 Czas [ns] dodawania klucza do kopca w zależności od l. elementów i wartości kluczy (równomierny wybór 20 punktów pomiarowych)



Rysunek 29 Wykres zawierający wszystkie punkty pomiarowe dla operacji dodawania klucza do kopca



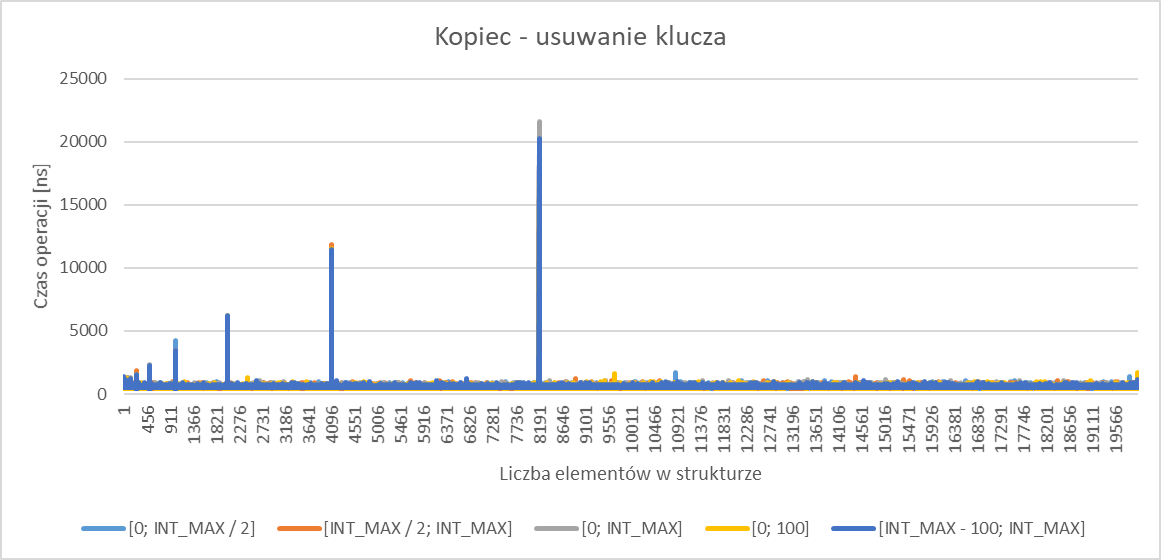
Rysunek 30 Linie trendu dla operacji dodawania klucza do kopca

Otrzymane wyniki różnią się od założeń teoretycznych. Czas dodawania elementu w tej strukturze okazał być się stały (O(1)) - w granicach niepewności pomiarowych. Pomiary powtórzono dla 1.000.000 elementów, jednak uzyskane wyniki nadal wskazywały na stały czas operacji. Może być to skutek powolnego tempa wzrostu czasu wykonywania operacji o złożoności logarytmicznej.

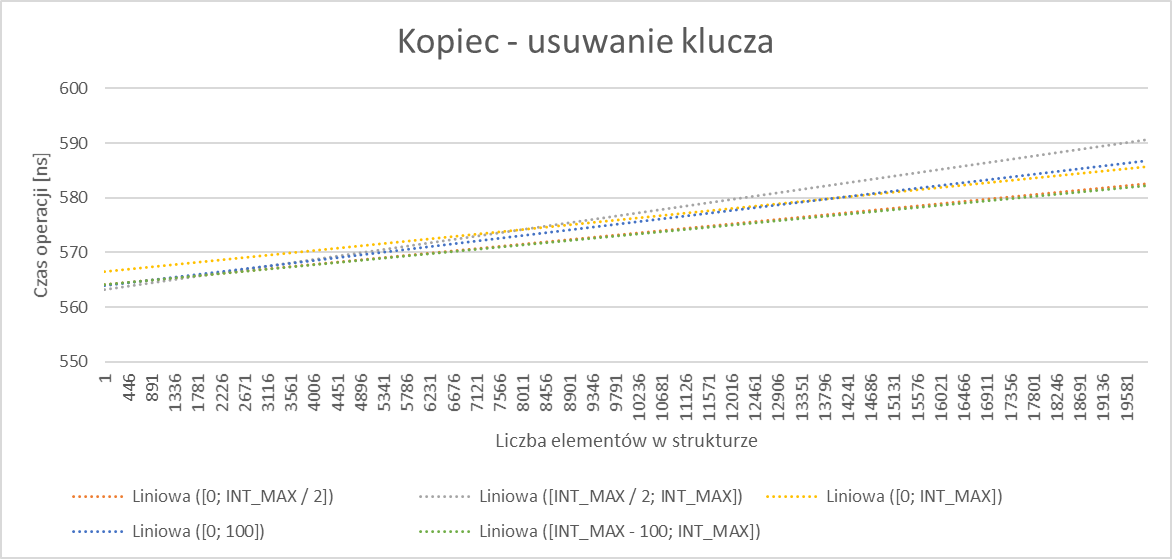
### Usuwanie klucza



Tabela 16 Czas [ns] usuwania klucza z kopca w zależności od l. elementów i wartości kluczy (równomierny wybór 20 punktów pomiarowych)



Rysunek 31 Wykres zawierający wszystkie punkty pomiarowe dla operacji usuwania klucza z kopca



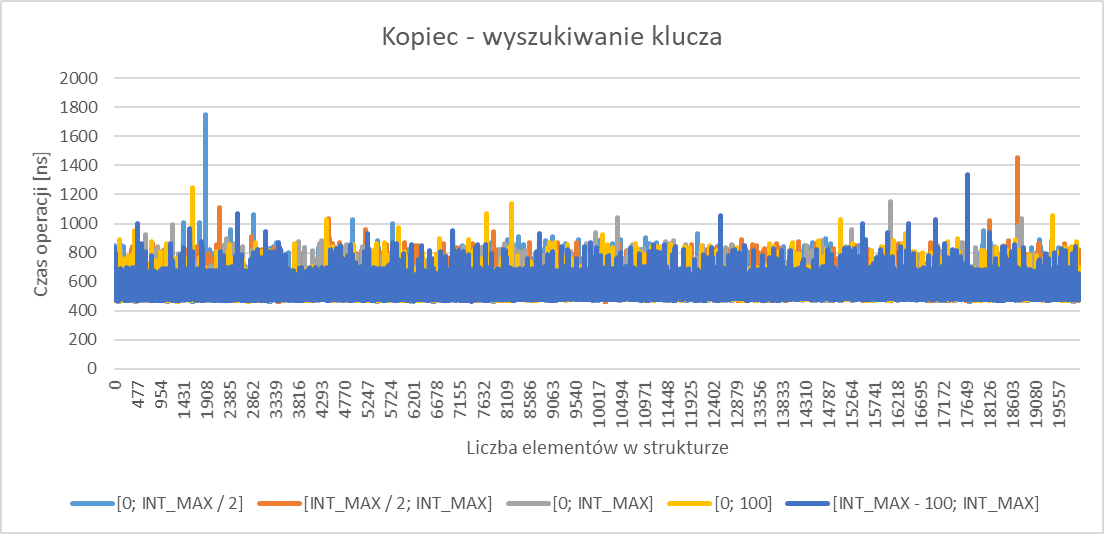
Rysunek 32 Linie trendu dla operacji usuwania klucza z kopca

Otrzymane wyniki różnią się od założeń teoretycznych. Czas dodawania elementu w tej strukturze okazał być się stały (O(1)) - w granicach niepewności pomiarowych. Pomiary powtórzono dla 1.000.000 elementów, jednak uzyskane wyniki nadal wskazywały na stały czas operacji. Może być to skutek powolnego tempa wzrostu czasu wykonywania operacji o złożoności logarytmicznej. W przeciwieństwie do operacji dodawania można zauważyć tutaj małą tendencję wzrostową.

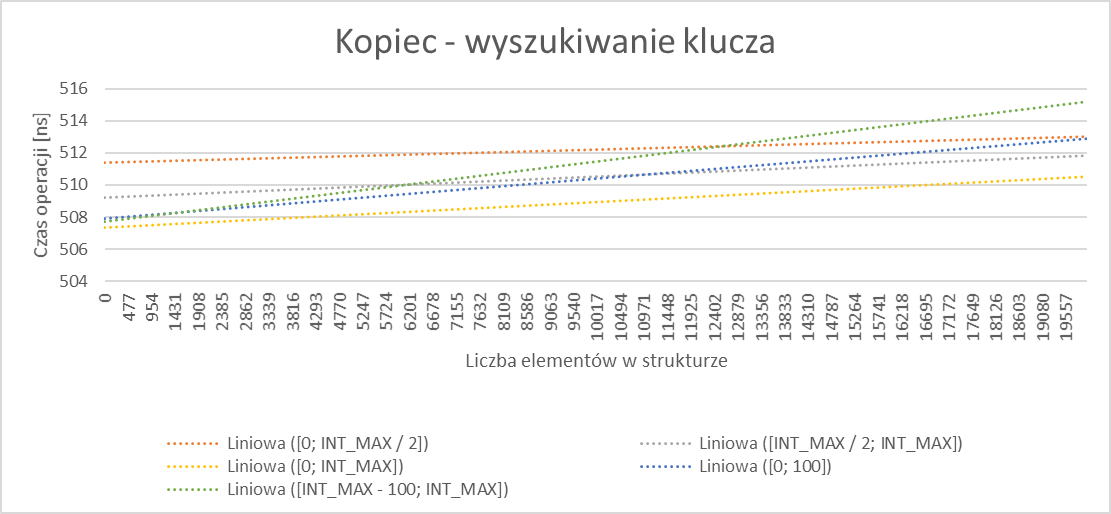
### Wyszukiwanie klucza



Tabela 17 Czas [ns] wyszukiwania klucza w kopcu w zależności od l. elementów i wartości kluczy (równomierny wybór 20 punktów pomiarowych)



Rysunek 33 Wykres zawierający wszystkie punkty pomiarowe dla operacji wyszukiwania klucza w kopcu



Rysunek 34 Linie trendu dla operacji wyszukiwania klucza w kopcu

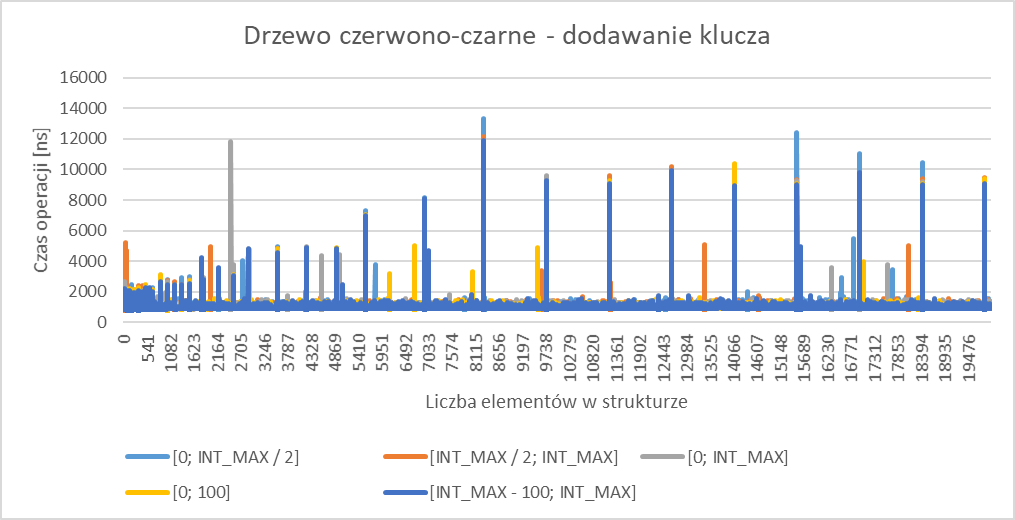
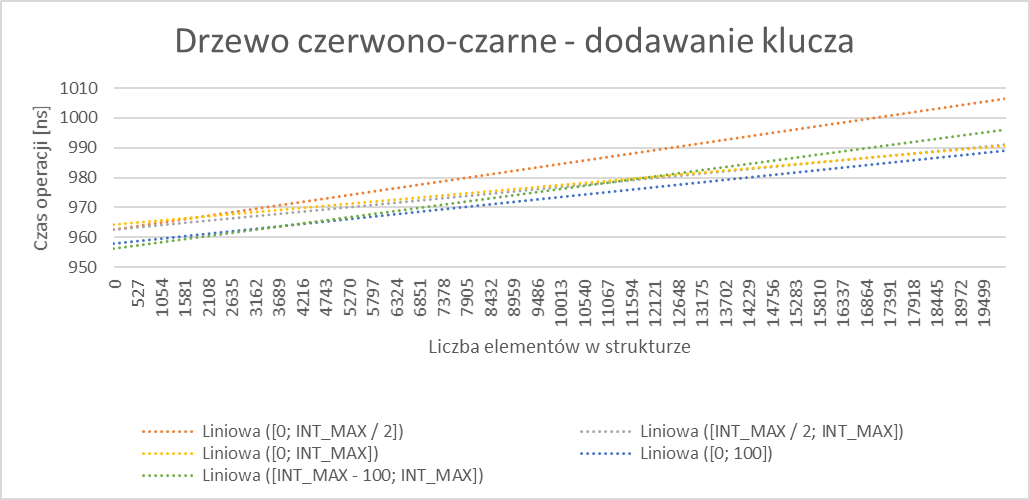
Otrzymane wyniki różnią się od założeń teoretycznych. Czas wyszukiwania elementu w tej strukturze okazał być się stały (O(1)) - w granicach niepewności pomiarowych. Pomiary powtórzono dla 1.000.000 elementów, jednak uzyskane wyniki nadal wskazywały na stały czas operacji. Może być to skutek zastosowanych rozwiązań w architekturze procesora (Intel), które pozwoliły na bardzo wydajny odczyt danych z pamięci (wyszukiwanie nie modyfikuje elementów w strukturze).

## Drzewo czerwono-czarne

### Dodawanie klucza



Tabela 18 Czas [ns] dodawania klucza w drzewie czerwono-czarnym w zależności od l. elementów i wartości kluczy (równomierny wybór 20 punktów pomiarowych)



Rysunek 35 Wykres zawierający wszystkie punkty pomiarowe dla operacji dodawania klucza w drzewie czerwono-czarnym

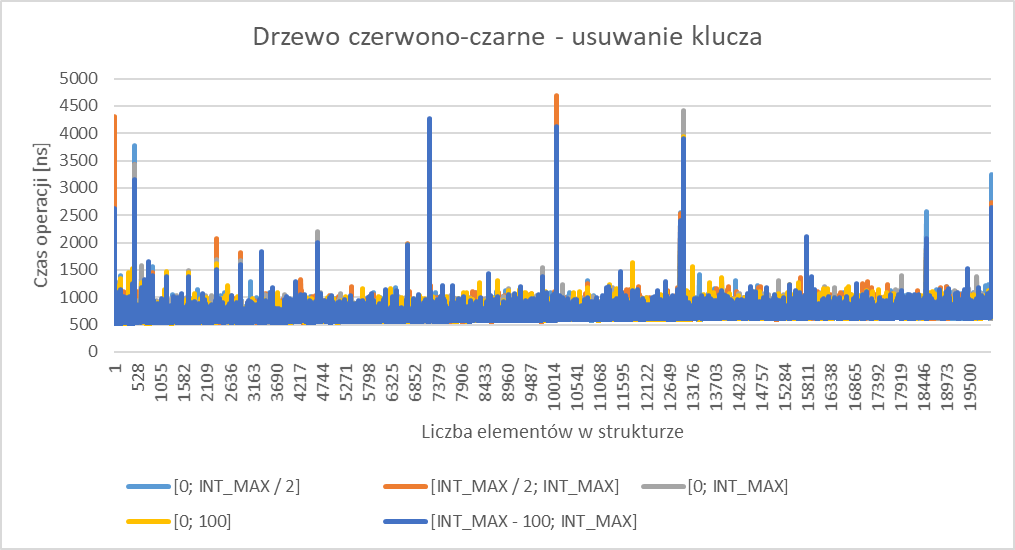
Rysunek 36 Linie trendu dla operacji dodawania klucza w drzewie czerwono-czarnym

Otrzymane wyniki różnią się od założeń teoretycznych. Czas dodawania elementu w tej strukturze okazał być się stały (O(1)) - w granicach niepewności pomiarowych. Pomiary powtórzono dla 1.000.000 elementów, jednak uzyskane wyniki nadal wskazywały na stały czas operacji. Może być to skutek powolnego tempa wzrostu czasu wykonywania operacji o złożoności logarytmicznej. Można zaobserwować jednak małą tendencję wzrostową, która może sugerować złożoność logarytmiczną operacji (jest to wolny wzrost).

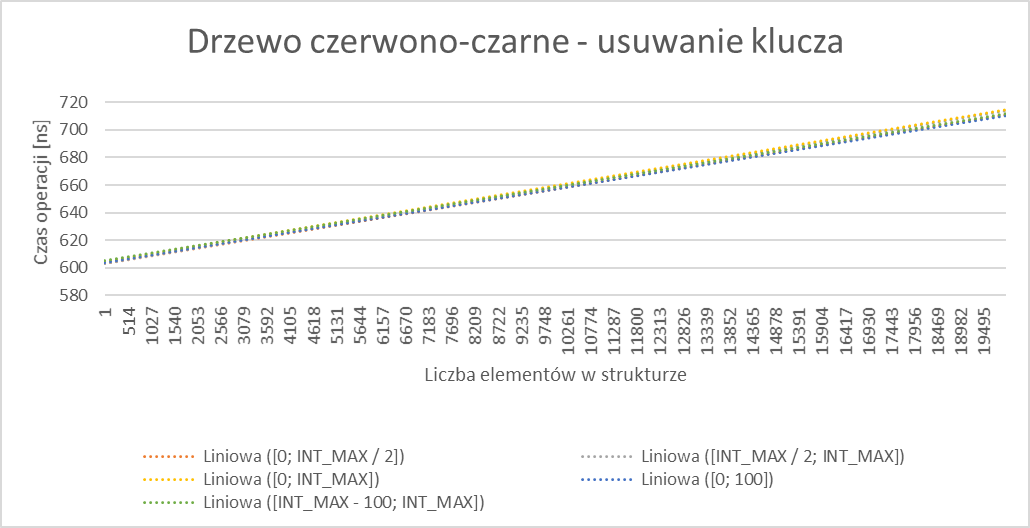
### Usuwanie klucza



Tabela 19 Czas [ns] usuwanie klucza w drzewie czerwono-czarnym w zależności od l. elementów i wartości kluczy (równomierny wybór 20 punktów pomiarowych)



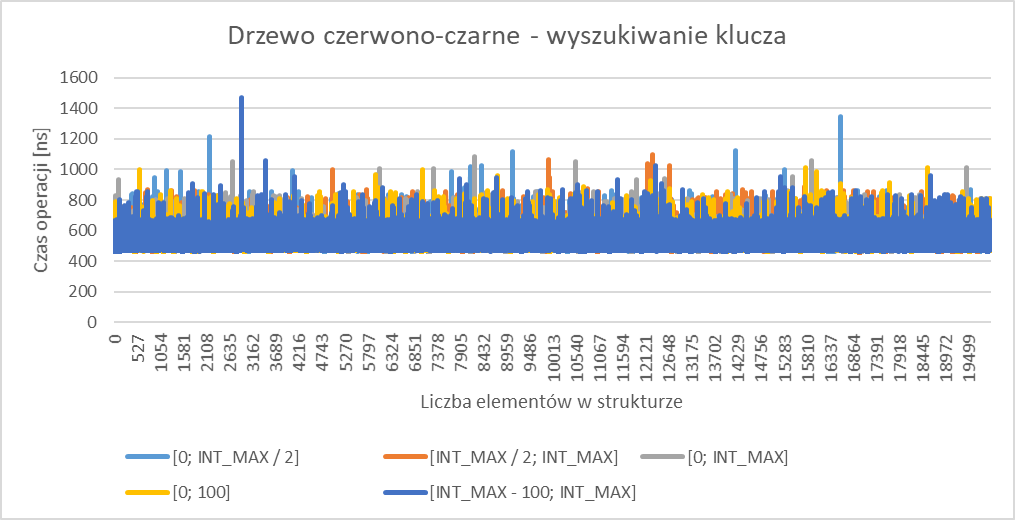
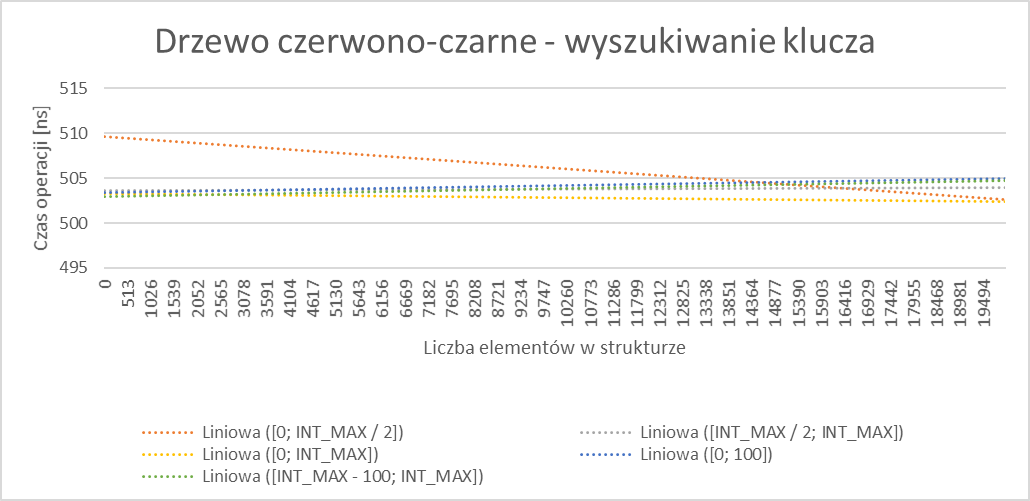
Rysunek 37 Wykres zawierający wszystkie punkty pomiarowe dla operacji usuwanie klucza w drzewie czerwono-czarnym

Otrzymane wyniki różnią się od założeń teoretycznych. Czas dodawania elementu w tej strukturze okazał być się stały (O(1)) - w granicach niepewności pomiarowych. Pomiary powtórzono dla 1.000.000 elementów, jednak uzyskane wyniki nadal wskazywały na stały czas operacji. Może być to skutek powolnego tempa wzrostu czasu wykonywania operacji o złożoności logarytmicznej. Można zaobserwować jednak tendencję wzrostową, która może sugerować złożoność logarytmiczną operacji (jest to wolny wzrost).

Rysunek 38 Linie trendu dla operacji usuwanie klucza w drzewie czerwono-czarnym

### Wyszukiwanie klucza

Tabela 20 Czas [ns] wyszukiwania klucza w drzewie czerwono-czarnym w zależności od l. elementów i wartości kluczy (równomierny wybór 20 punktów pomiarowych)



Rysunek 39 Linie trendu dla operacji wyszukiwania klucza w drzewie czerwono-czarnym

Rysunek 40 Wykres zawierający wszystkie punkty pomiarowe dla operacji wyszukiwania klucza w drzewie czerwono-czarnym

Otrzymane wyniki różnią się od założeń teoretycznych. Czas wyszukiwania elementu w tej strukturze okazał być się stały (O(1)) - w granicach niepewności pomiarowych. Pomiary powtórzono dla 1.000.000 elementów, jednak uzyskane wyniki nadal wskazywały na stały czas operacji. Może być to skutek zastosowanych rozwiązań w architekturze procesora (Intel), które pozwoliły na bardzo wydajny odczyt danych z pamięci (wyszukiwanie nie modyfikuje elementów w strukturze).

1. Podsumowanie



Badane struktury można podzielić na dwie grupy:

* działające w czasie liniowym w pesymistycznych przypadkach (tablica, lista)
* działające w czasie logarytmicznym w pesymistycznych przypadkach (kopiec, drzewo czerwono – czarne)

Lista okazała się górować nad tablicą, gdyż w operacjach dodawania na oraz usuwania z początku uzyskała czas O(1), gdzie tablica potrzebuje czasu O(n) na wykonanie ww. operacji. Dodanie/usunięcie elementu w losowym miejscu wymaga liniowego czasu w obu strukturach– jest to spowodowane koniecznością wyszukania elementu w pierwszej kolejności (lista) lub przesunięcia danych (tablica). Dodanie/usuniecie elementu z końca w obu strukturach okazało być się operacją stałą. Wyszukiwanie wymaga teoretycznego czasu liniowego zarówno w tablicy jaki i liście, jednak wykazano, że w pewnych okolicznościach (szukanie losowego elementu z zakresu, przy odpowiednio małej wielkości struktury oraz na specyficznym sprzęcie) może być operacją w praktyce (średnio) stałą (chociaż jest to sprzeczne z teorią). Lista wydaje się być odpowiednim kandydatem na implementację kolejki, zaś tablica sprawdziłaby się w implementacji stosu. Warto nadmienić, że przewagą tablicy nad listą jest natychmiastowy dostęp do elementu o znanym indeksie, który to w wielu przypadkach decyduje o słuszności zastosowania właśnie tablicy, a nie listy do przechowywania danych.

Kopiec i drzewo czerwono – czarne potrzebują w teorii logarytmicznego czasu na wykonanie wszystkich badanych operacji słownikowych (z wyjątkiem operacji wyszukiwania w kopcu – ta ma złożoność O(n)). Badania wykazały jednak, że przy odpowiednich rozmiarach danych i konfiguracji sprzętu operacje takie, jak dodawanie, usuwanie czy wyszukiwanie elementu można traktować jako działające (średnio) w stałym czasie. Wynika to z faktu, iż przyrost czasu potrzebnego no wykonanie ww. operacji dla coraz większej ilości danych jest wolny. Niemniej jednak należy podkreślić, że drzewo czerwono – czarne ma zasadniczą przewagę nad kopcem: jest to zrównoważone drzewo poszukiwań binarnych gwarantujące górne ograniczenie na operację wyszukiwania elementu równe O(lgn). Kosztem tego stanu rzeczy jest spory czynnik stały w operacjach dodawania i usuwania drzewa, co uwidacznia się na wykresach w postaci wyższej dolnej granicy przedziału wartości wymaganego na wykonanie operacji czasu. Kopiec wydaje się być dobrym kandydatem na bazę dla kolejki priorytetowej ze względu na swoją własność kopca. Drzewo czerwono – czarne pozwoli natomiast na szybkie wyszukiwanie elementu; jego implementacja ma jednak sens, gdy ilość danych jest względnie duża.

Warto zauważyć, iż na wykresach wystąpiły nagłe skoki wartości. Ich geneza może być różna.

W tablicy i kopcu wynikają one z czasu potrzebnego na realokację danych przy zwiększaniu rozmiaru, chociaż mogą być to również opóźnienia wynikłe z działania systemu operacyjnego, który podczas wykonywania testów nie zawsze przydzielał najwyższy priorytet zadaniom programu testującego. W przypadku listy i drzewa czerwono – czarnego nieciągłości tych jest więcej w porównaniu z tablicą i kopcem. Może to wynikać z faktu, iż do operacji na tych strukturach używa się w dużej mierze wskaźników, przez co pojawia się wiele żądań pośredniego odczytu z pamięci podczas działania programu, co może skutkować zwiększonym wykorzystaniem magistral adresowych i w ostateczności w zaobserwowany sposób wpłynąć na czas wykonywania operacji.

Charakterystyczne są również duże zagęszczenia w dolnych przedziałach czasowych operacji wyszukiwania we wszystkich badanych strukturach. Jest to następstwem wyszukiwania kluczy z rozkładu równomiernego i uwidacznia się to na wykresach w postaci „równomiernych” czasów potrzebnych na znalezienie danego klucza w strukturze.