Marcin Bajak 225984

Struktury Danych i Złożoność Obliczeniowa

Zadanie projektowe nr 1: Badanie efektywności operacji wstawiania, usuwania oraz wyszukiwania elementów w podstawowych strukturach danych

Spis treści

Wstęp teoretyczny

Celem przeprowadzonego eksperymentu było zbadanie efektywności podstawowych operacji takich jak wstawianie, wyszukiwanie oraz usuwanie. Aby tego dokonać, najpierw należało zaimplementować te struktury, czyli tablicę (z realokacją pamięci), listę dwukierunkową oraz kopiec binarny. Język programowania jaki został użyty przy programowaniu struktur jak i przy przeprowadzaniu testów to C++.

Badaną wielkością było asymptotyczne tempo wzrostu konkretnych operacji, zapisywane za pomocą notacji dużego O, w celu późniejszego opisu złożoności obliczeniowej tychże operacji.

1.1.Tablica z realokacją pamięci

Tablica jest strukturą danych zawierająca klucze, z których każdy jest identyfikowalny przez konkretny indeks tablicy. W celu uzyskania dostępu do konkretnego klucza potrzebujemy jego indeksu oraz adresu początkowego tablicy. Adres elementów wyliczany jest równaniem: adres_początkowy + rozmiar_klucza * rządany_indeks.

W przeprowadzonych badaniach zaimplementowana została tablica z realokacją pamięci przy usuwaniu oraz wstawianiu klucza w celu zaoszczędzenia pamięci.

- ✦ Wstawianie
  - Początek tablicy

    W celu wstawienia nowego klucza na początek tablicy tworzona jest nowa tablica o rozmiarze o 1 większym w której wszystkie klucze przesunięte zostają o jeden indeks do przodu, a nowy klucz zajmuje miejsce o indeksie 0.

    Złożoność obliczeniowa takiej operacji zależy od wielkości tablicy, ponieważ wszystkie dane muszą zostać przesunięte o jedną pozycję, czyli w zapisie notacji dużego O będzie to O(n) – złożoność obliczeniowa liniowa.

  - Środek tablicy

    Sytuacja podobna do wstawiania na początek, również trzeba realokować całą tablicę, z tą różnicą, że mniejszą liczbę danych trzeba przesunąć co daje krótszy czas wykonywania.

Złożoność obliczeniowa taka sama jak powyżej, czas wykonywania jest krótszy, jednak w tym przypadku również czas rośnie liniowo w zależności od ilości danych w tablicy. Złożoność O(n).

- Koniec tablicy

  W tym przypadku nie ma potrzeby przesuwania elementów tablicy, co skutkuje zmniejszeniem czasu wykonywania operacji, jednak nadal niezbędna jest realokacja pamięci.

  Złożoność obliczeniowa w tym przypadku również pozostaje bez zmian w związku z niezmienną potrzebą realokacji całej tablicy, co zajmuje stosunkowo więcej czasu w zależności od ilości danych. Złożoność O(n).

✦ Usuwanie

- Początek tablicy

  Usuwanie jest operacją podobną do wstawiania, z tą różnicą, że przy usuwaniu z początku tablicy, realokujemy pamięć o mniejszym rozmiarze o 1 i przesuwamy wszystkie klucze o jeden indeks w dół z pominięciem indeksu 0.

  Operacja ta jak i poprzednie ma złożoność czasową liniową ze względu na potrzebę przepisania wszystkich danych. Złożoność O(n).

- Środek tablicy

  Różnica taka jak przy wstawianiu – mniej wartości do przesunięcia, co daje krótszy czas wykonywania.

  Złożoność bez zmian – nadal liniowa - O(n).

- Koniec tablicy

  W tym przypadku również różnica jak przy wstawianiu. Czas wykonania jest krótszy, ponieważ nie trzeba przesuwać wartości, jednak czas realokacji pozostaje bez zmian.

  Złożoność bez zmian – nadal liniowa – O(n).

✦ Wyszukiwanie

- Przeszukiwanie całej tablicy

  Każdy kolejny klucz w tablicy porównywany jest z szukanym kluczem do momentu znalezienia szukanej wartości, czyli w najgorszym wypadku trzeba przeszukać całą tablicę, gdy wartość klucza szukanego nie znajduje się w tablicy.

  Złożoność obliczeniowa w tym wypadku jest liniowa – im więcej elementów, tym więcej porównań – złożoność O(n).

1.2. Lista dwukierunkowa

Lista dwukierunkowa jest strukturą danych składającą się z sekwencyjnie połączonych ze sobą elementów. Każdy z elementów, zawiera przechowywany klucz oraz dwa pola wskazujące na element poprzedni oraz następny w liście. Lista dwukierunkowa zawiera pola wskazujące na element pierwszy oraz ostatni na liście, których odpowiednio poprzednim oraz następnym elementem jest pewnego rodzaju oznaczenie wskazujące na brak elementu poprzedniego lub następnego (np. wskaźnik na element pusty).

✦ Wstawianie

- Początek listy

  W przypadku wstawiania na początek listy, nowy element staje się pierwszym elementem na liście, a poprzednio pierwszy element zostaje elementem kolejnym. Operacja ta nie jest czasochłonna, ze względu na to, że jedyne co trzeba zrobić, to przypisać wskazanie poprzedniego elementu do elementu pierwszego, a do elementu nowego wskazanie elementu następnego do adresu elementu poprzednio pierwszego.

  Złożoność czasowa takiej operacji jest stała, niezależna od ilości elementów, zawsze musimy wykonać jedynie operacje opisane powyżej. Złożoność O(1).

- Środek listy

  W przypadku, kiedy chcemy wstawić nowy element w środek listy, musimy najpierw odnaleźć element, który obecnie się tam znajduje, co jest tym bardziej czasochłonne, im więcej elementów znajduję się w liście, ponieważ musimy przejść przez każdy poprzedni element. Następnie tworzymy nowe połączenia między nowym elementem, a jego następnym i poprzednim elementem.

  Złożoność w tym wypadku nie jest już stała, jak miało to miejsce przy wstawianiu na początek listy, lecz jest liniowa, ze względu na iterowanie poprzez elementy listy, do żądanego miejsca na liście. Złożoność O(n).

- Koniec listy

  Przypadek podobny jak przy wstawianiu na początek listy, z tą różnicą, że musimy odwołać się do tzw. ogona listy i wykonać na nim operacje analogiczne do wstawiania na początku listy.

  Złożoność taka sama jak przy wstawianiu na początek – O(1).

✦ Usuwanie

- Początek listy

  Przy usuwaniu z początku listy, jedyne operacje jakie musimy wykonać to przemianowanie drugiego elementu listy na element pierwszy, poprzez ustawienie jego pola wskaźnika na poprzedni element na adres pusty oraz usunięcie poprzednio pierwszego elementu.

  Operacja ta ma złożoność obliczeniową stałą, liczba elementów listy nie wpływa na szybkość jej wykonywania. Złożoność O(1).

- Środek listy

  Tak samo jak w przypadku wstawiania elementu w środek listy tutaj również musimy odnaleźć element środkowy listy, co wraz z wzrostem elementów staje się coraz to bardziej czasochłonne. Po odnalezieniu żądanego elementu, tworzymy nowe wiązania między jego następnym i poprzednim elementem, a następnie element ten usuwamy.

  Złożoność obliczeniowa wzrasta wraz ze wzrostem liczby elementów ze względu na potrzebę iteracji przez kolejne elementy listy. Złożoność O(n).

- Koniec listy

  Przypadek podobny jak przy usuwaniu z początku listy, tutaj również musimy przemianować przedostatni element na element ostatni oraz zwolnić pamięć po uprzednio ostatnim.

  Złożoność taka sama jak przy usuwaniu z początku – O(1).

✦ Wyszukiwanie

- Przeszukiwanie całej listy

  Każdy klucz przechowywany przez elementy w liście, porównywany jest z szukanym kluczem do momentu znalezienia szukanej wartości, czyli w najgorszym wypadku trzeba przeszukać całą tablicę gdy wartość klucza szukanego nie znajduje się na liście.

  Złożoność obliczeniowa w tym wypadku jest liniowa – im więcej elementów, tym więcej porównań – złożoność O(n).

1.3. Kopiec binarny

Kopiec jest strukturą danych przybierającą formę drzewiastą. W zależności od rodzaju kopca, minimalny lub maksymalny, odpowiednio każde dziecko jest większe lub mniejsze od swojego rodzica. W skutek tego otrzymujemy kopiec maksymalny lub minimalny, gdzie odpowiednio w korzeniu przechowywana jest wartość największa lub najmniejsza. Kopiec jest jednym ze sposobów implementacji kolejki priorytetowej. Wysokość drzewa h = log2(n), gdzie n to ilość przechowywanych kluczy. Przy implementacji drzewa można wykorzystać tablicę lub listę. W tym przypadku zbadane zostało zastosowanie tablicy bez realokacji.

✦ Wstawianie

W przypadku kopca operacja wstawiania nie jest rozdzielana na konkretne pozycje, wartość wstawiana jest zawsze na miejsce za ostatnim elementem kopca, a następnie krok po kroku nowa wartość jest przesuwana w górę drzewa, poprzez porównanie nowego elementu z jego rodzicem i ich zamianę miejscami, jeżeli własności kopca nie są zachowane.

W najgorszym przypadku, gdy nowa wartość jest odpowiednio największa lub najmniejsza w kopcu, zależnie od rodzaju kopca, złożoność obliczeniowa operacji wstawiania to O(log2(n)), ponieważ tyle właśnie operacji zamiany elementów trzeba wykonać. Natomiast średnia złożoność obliczeniowa tej operacji to O(1) – czyli czas stały.

✦ Usuwanie

W przypadku usuwania również nie wyróżnia się usuwania z konkretnych pozycji, lecz jedynie usuwanie elementu odpowiednio największego lub najmniejszego, w zależności od rodzaju kopca. W celu usunięcia korzenia, zamienia się wartości korzenia z elementem ostatnim w drzewie, usuwa się ostatni element, który teraz ma wartość korzenia, a następnie przesuwa się nowy korzeń w dół drzewa, poprzez porównywanie go z jednym z jego dzieci, do momentu, kiedy własności kopca będą zachowane.

Operacja ta ma złożoność czasową logarytmiczną – O(log2(n)), ponieważ po zamianie korzenia z elementem ostatni, zazwyczaj trzeba przesunąć nową wartość korzenia na sam dół drzewa, czyli wykonać log2(n) operacji zamiany.

✦ Wyszukiwanie

- Przeszukiwanie całego kopca

  Każdy kolejny klucz przechowywany w kopcu porównywany jest z szukanym kluczem do momentu znalezienia szukanej wartości, czyli w najgorszym wypadku trzeba przeszukać cały kopiec, gdy wartość klucza szukanego nie znajduje się w kopcu.

  Złożoność obliczenia w tym przypadku jest liniowa – im więcej elementów tym więcej porównań – złożoność O(n).

2. Plan eksperymentu

- Językiem programowania użytym w eksperymencie był C++ w standardzie ISO C++11
- Badanie poszczególnych operacji zostało przeprowadzone na losowo generowanych zestawach danych zawierających klucze z przedziałów: *[10,100]*, *[~(INT_MAX/2)]*, *[~INT_MAX]* oraz *[1, INT_MAX]*. ~ - rozrzut rzędu 100000000
- Losowe liczby z podanych przedziałów były generowane przy użyciu funkcji *rand().*
- Badania przeprowadzano dla kolejnych rozmiarów struktur: *16, 32, 64, 128, ...,* z ograniczeniem maksymalnej liczby danych, które w tym przypadku wynosiło *1GB/4B*
- Ilość powtórzeń wykonania każdej operacji to od *2500000* do *3000000* zależnie od struktury i operacji, podzielone przez rozmiar struktury, z minimalną liczbą operacji, również zależną od struktury i operacji w przedziale od *10* do *1000*
- Otrzymanym wynikiem jest zmierzony czas podzielony przez ilość wykonanych operacji
- Czas wykonania operacji mierzony był przy użyciu biblioteki *chrono* przy pomocy funkcji *std::chrono::high_resolution_clock*

3. Wyniki/Wnioski

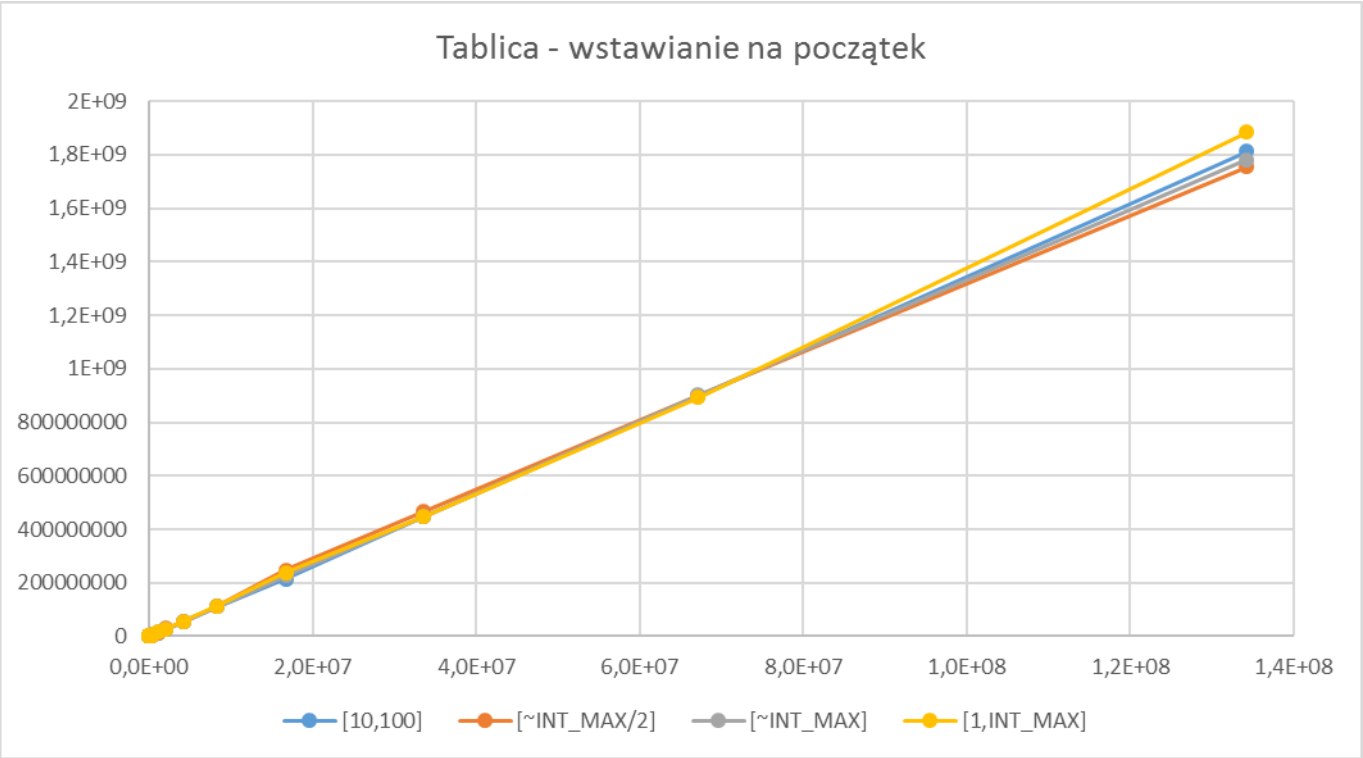
Czasy wykonywania poszczególnych operacji podane zostały w nanosekundach.

3.1. Tablica

3.1.1. Wstawianie na początek

Wstawianie na początek:				
l. elementów	czas operacji dla konkretnych przedziałów wartości kluczy			
[16,~maxMem]	*[10,100]*	*[~INT_MAX/2]*	*[~INT_MAX]*	*[1,INT_MAX]*
16	409	400	403	403
32	582	589	576	576
64	947	921	934	934
128	1613	1613	1613	1639
256	2970	2970	2970	3021
512	5836	5836	5534	5734
1024	11264	11264	11263	11273
2048	22959	21718	22554	22544
4096	45113	45088	45088	45088
8192	90170	95091	93432	88554
16384	177736	179506	181039	184184
32768	368381	361789	355302	355315
65536	737342	789947	762947	631473
131072	1579210	1527210	1473894	1447947
262144	3444000	3499555	3444222	3611666
524288	7374250	7316500	7520250	7124500
1048576	11885500	14002000	14252000	14252500
2097152	25753000	28767000	28003000	28496000
4194304	55001000	54860000	54051000	55499000
8388608	111521000	113501000	111514000	113515000
16777216	213527000	248990000	227022000	237547000
33554432	447314000	464845000	446823000	447064000
67108864	896628000	900615000	902877000	891455000
134217728	1813646000	1754810000	1781536000	1883490000

Tabela 1 Czas[ns] wstawiania na początek tablicy w zależności od l. elementów i wartości kluczy



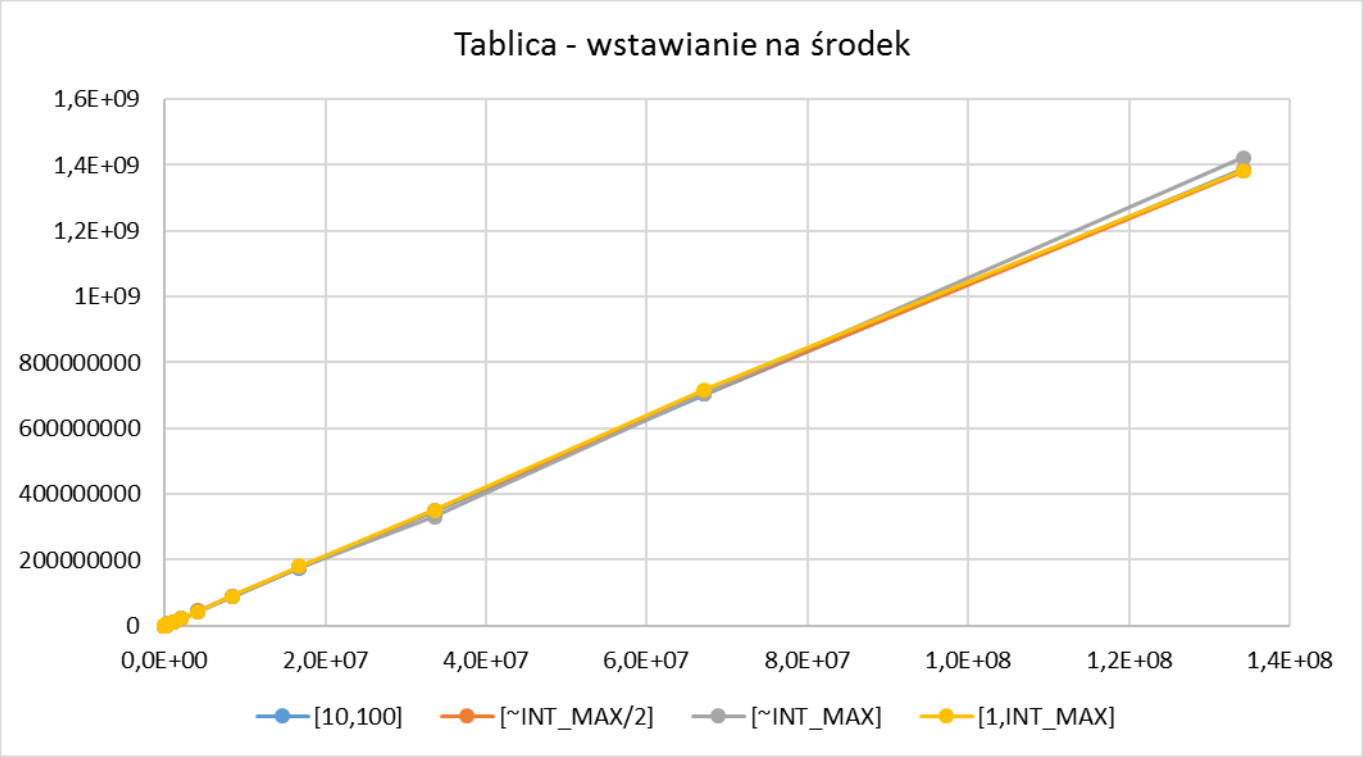
Wykres 1 Zależność między l. elementów a czasem wstawiania na początek tablicy dla konkretnych przedziałów wartości

Wyniki przedstawione powyżej dla wstawiania na początek tablicy zgadzają się z teorią, która mówi o liniowej zależności czasu wykonywania operacji od liczby elementów znajdujących się w strukturze. Brak zauważalnego wpływu wartości kluczy na czas wykonania operacji.

3.1.2. Wstawianie na środek

Wstawianie na środek:				
l. elementów	czas operacji dla konkretnych przedziałów wartości kluczy			
[16,~maxMem]	*[10,100]*	*[~INT_MAX/2]*	*[~INT_MAX]*	*[1,INT_MAX]*
16	355	446	518	355
32	486	486	492	486
64	768	755	755	755
128	1280	1280	1280	1279
256	2304	2303	2304	2354
512	4518	4482	4404	4610
1024	8396	8808	8399	9237
2048	16512	17214	17215	16804
4096	34432	33609	32791	35273
8192	73409	70501	70498	68836
16384	142177	139190	141414	141559
32768	289697	276250	276250	263197
65536	579210	539289	565868	526394
131072	1132157	1105000	1105789	1105368
262144	2666333	2777444	2777222	2720888
524288	5625750	5499000	5625750	5625750
1048576	11001500	10810000	11065000	11001000
2097152	22510000	22003000	22004000	22490000
4194304	44507000	44520000	43506000	42491000
8388608	88011000	89006000	90013000	89510000
16777216	176037000	177017000	176513000	180750000
33554432	348044000	349820000	330526000	351045000
67108864	704180000	707090000	704604000	716908000
134217728	1386857000	1381449000	1421711000	1382202000

Tabela 2 Czas[ns] wstawiania na środek tablicy w zależności od l. elementów i wartości kluczy



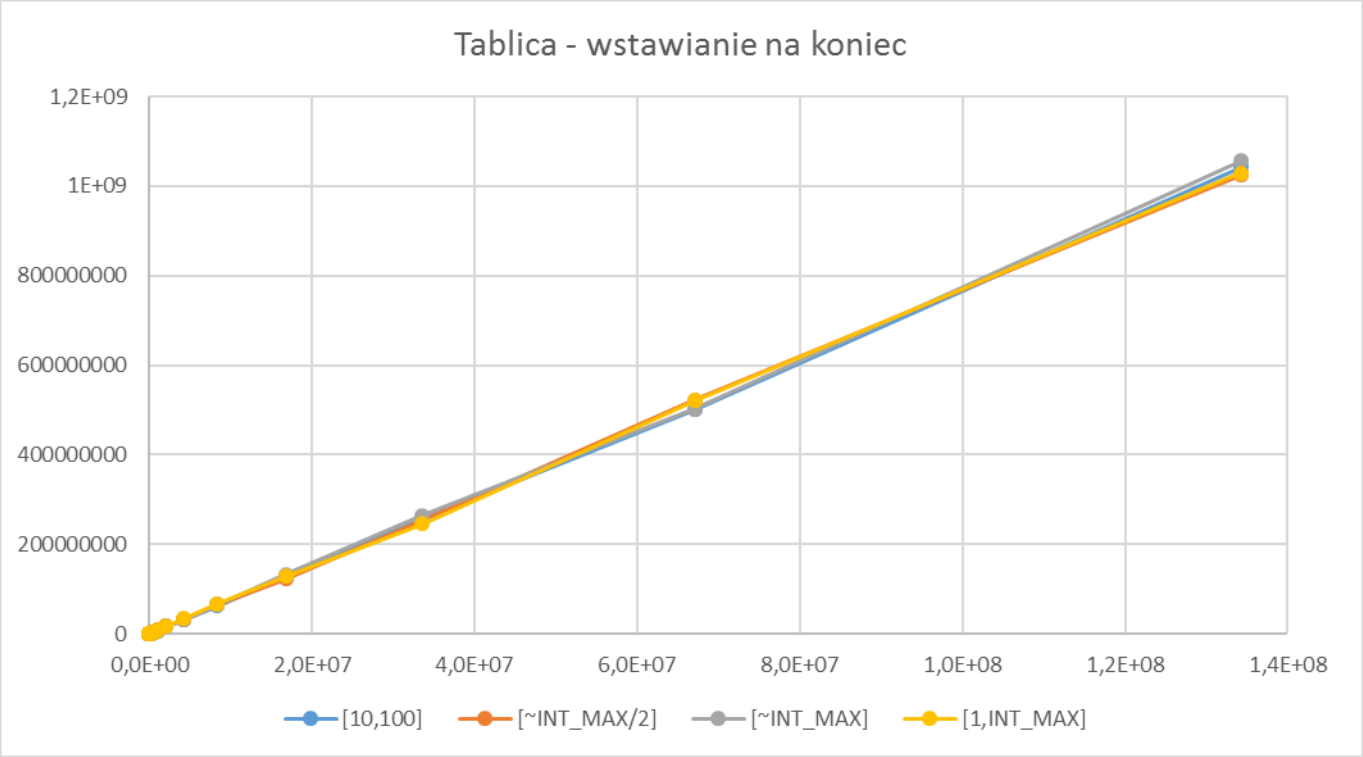
Wykres 2 Zależność między l. elementów a czasem wstawiania na środek tablicy dla konkretnych przedziałów wartości

Wyniki przedstawione powyżej dla wstawiania na środek tablicy zgadzają się z teorią, która mówi o liniowej zależności czasu wykonywania operacji od liczby elementów znajdujących się w strukturze. Zgodnie z założeniami w części teoretycznej czas wstawiania na środek jest krótszy od wstawiania na początek tablicy. Brak zauważalnego wpływu wartości kluczy na czas wykonania operacji.

3.1.3. Wstawianie na koniec

Wstawianie na koniec:				
l. elementów	czas operacji dla konkretnych przedziałów wartości kluczy			
[16,~maxMem]	*[10,100]*	*[~INT_MAX/2]*	*[~INT_MAX]*	*[1,INT_MAX]*
16	296	381	307	307
32	403	403	396	396
64	576	563	563	588
128	921	921	947	921
256	1485	1638	1588	1587
512	2767	2970	2927	2969
1024	5121	5739	5533	5881
2048	10894	11464	11472	11488
4096	22954	22600	21339	21693
8192	45885	45452	45232	47593
16384	84894	88986	101940	98703
32768	203973	184407	197407	203684
65536	407736	381605	381657	406023
131072	842315	842947	790368	789631
262144	1999555	1999555	1944111	2071111
524288	3936500	4004250	4127750	4250750
1048576	8004000	8636000	7365000	8325550
2097152	16495000	16497000	17494000	16498000
4194304	32001000	31005000	31504000	34011000
8388608	63295000	66003000	66011000	67610000
16777216	128502000	123010000	132510000	128004000
33554432	257028000	253011000	264526000	246170000
67108864	500787000	522147000	502415000	521287000
134217728	1041788000	1025185000	1056554000	1029408000

Tabela 3 Czas[ns] wstawiania na koniec tablicy w zależności od l. elementów i wartości kluczy



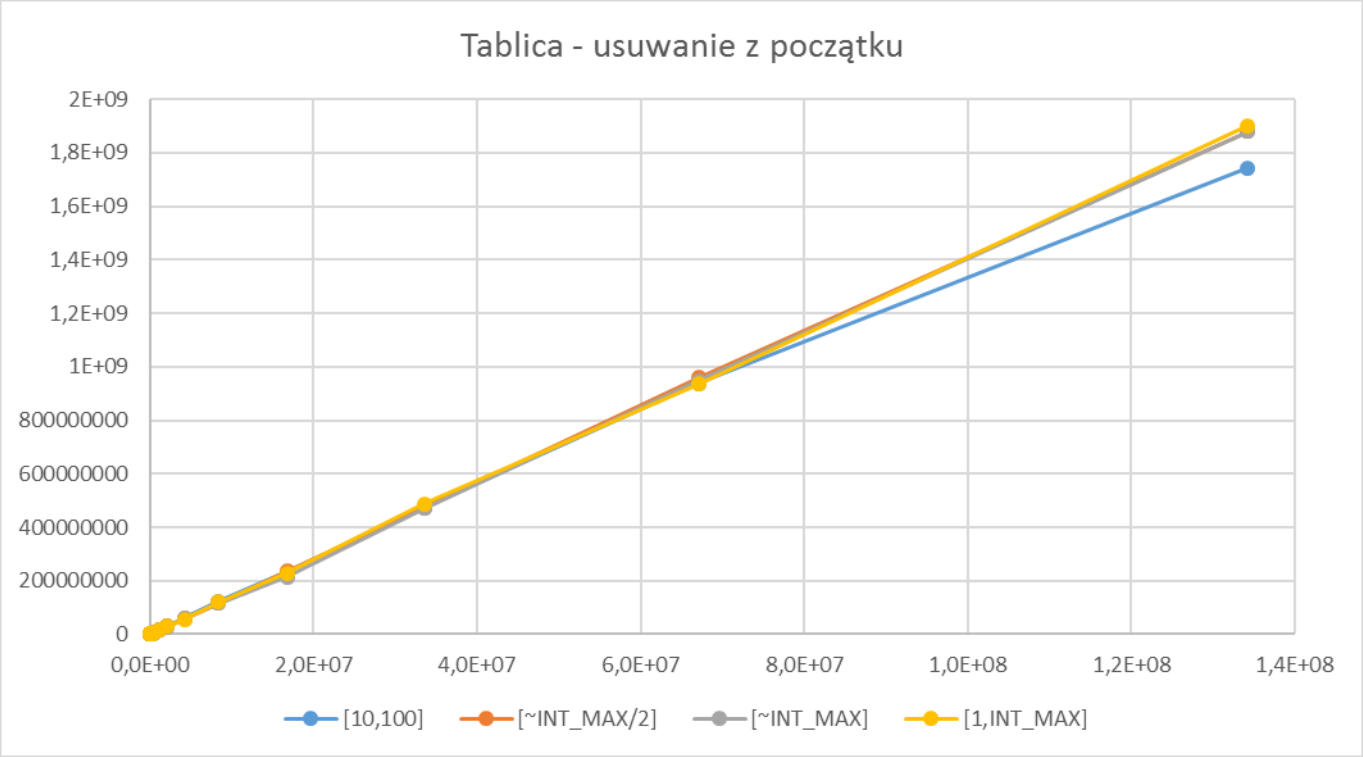
Wykres 3 Zależność między l. elementów a czasem wstawiania na koniec tablicy dla konkretnych przedziałów wartości

Wyniki przedstawione powyżej dla wstawiania na koniec tablicy zgadzają się z teorią, która mówi o liniowej zależności czasu wykonywania operacji od liczby elementów znajdujących się w strukturze. Zgodnie z założeniami w części teoretycznej czas wstawiania na koniec jest krótszy od wstawiania na początek oraz wstawiania na środek tablicy. Brak zauważalnego wpływu wartości kluczy na czas wykonania operacji

3.1.4. Usuwanie z początku

Usuwanie z początku:				
l. elementów	czas operacji dla konkretnych przedziałów wartości kluczy			
[16,~maxMem]	*[10,100]*	*[~INT_MAX/2]*	*[~INT_MAX]*	*[1,INT_MAX]*
16	316	316	367	360
32	480	486	480	486
64	844	870	870	844
128	1561	1510	1510	1561
256	2970	3022	3021	3071
512	6041	5940	5838	5942
1024	11378	11679	11472	11884
2048	23357	23299	23536	23358
4096	46716	45908	46714	47524
8192	95068	96432	92626	94245
16384	190861	190769	190717	183697
32768	400447	383434	338315	339644
65536	763263	763263	750105	710447
131072	1579157	1401105	1320052	1526473
262144	3611444	3667000	2936555	3734000
524288	7249000	7254750	7499250	7249250
1048576	14431500	14498000	14745000	14244500
2097152	30492000	29004000	28497000	28998000
4194304	60238000	58001000	60001000	57040000
8388608	123605000	116502000	117023000	121384000
16777216	233018000	235532000	215038000	229625000
33554432	475382000	473053000	470053000	487958000
67108864	939113000	960383000	947107000	936329000
134217728	1743043000	1880492000	1880746000	1902263000

Tabela 4 Czas[ns] usuwania z początku tablicy w zależności od l. elementów i wartości kluczy



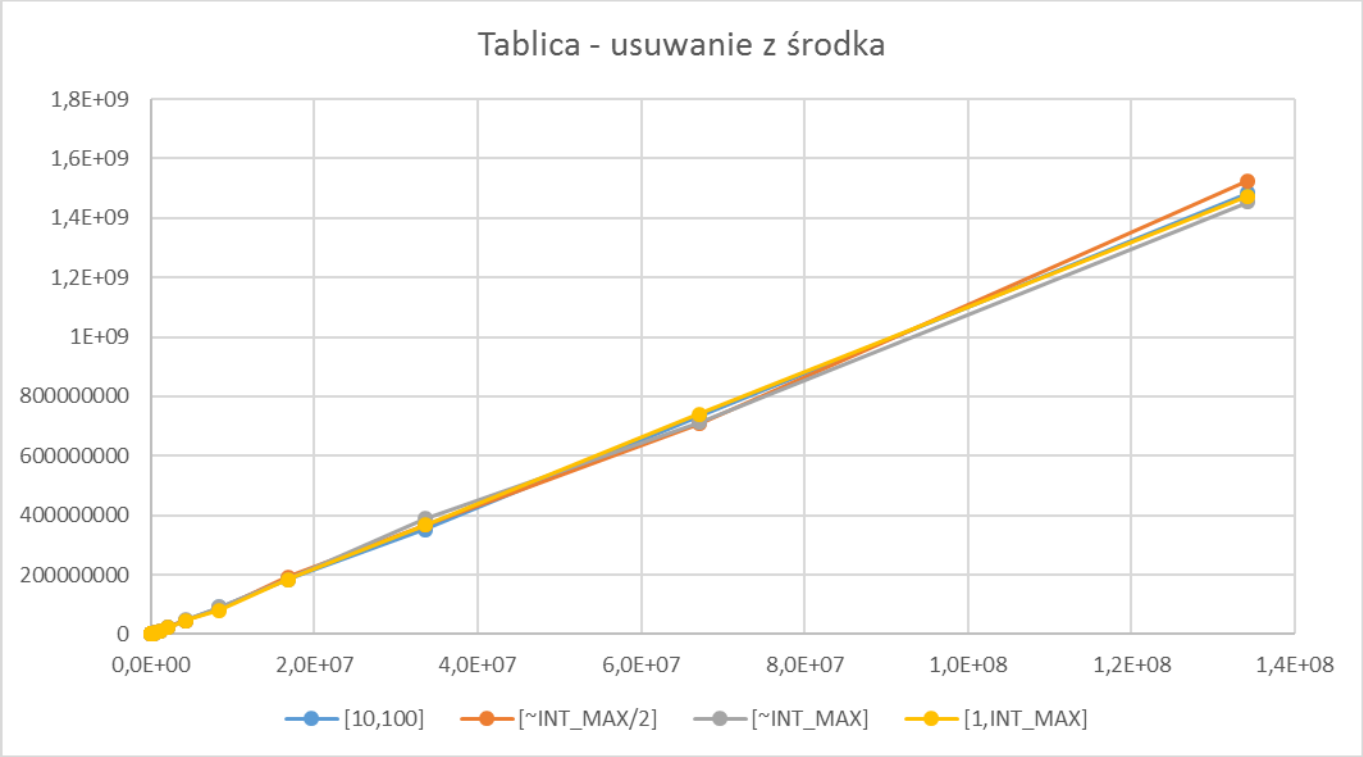
Wykres 4 Zależność między l. elementów a czasem usuwania z początku tablicy dla konkretnych przedziałów wartości

Wyniki przedstawione powyżej dla usuwania z początku tablicy zgadzają się z teorią, która mówi o liniowej zależności czasu wykonywania operacji od liczby elementów znajdujących się w strukturze. Brak zauważalnego wpływu wartości kluczy na czas wykonania operacji, odstępstwo dla wartości z przedziału [10,100] spowodowane prawdopodobnie niedokładnością pomiaru dla największej możliwej liczby elementów.

3.1.5. Usuwanie z środka

Usuwanie z środka:				
l. elementów	czas operacji dla konkretnych przedziałów wartości kluczy			
[16,~maxMem]	*[10,100]*	*[~INT_MAX/2]*	*[~INT_MAX]*	*[1,INT_MAX]*
16	252	246	246	246
32	396	384	390	396
64	691	652	652	665
128	1229	1203	1177	1178
256	2253	2303	2253	2201
512	4609	4199	4404	4403
1024	9008	8293	8792	9501
2048	17209	17619	18030	17209
4096	35239	34419	35273	34431
8192	72118	70452	70478	68865
16384	134881	141467	138177	141453
32768	282842	269842	282934	269960
65536	539552	552710	565868	565868
131072	1185157	1210736	1157789	1211473
262144	2778111	2890888	2779888	2721777
524288	5749250	5375750	5750750	5749500
1048576	11001500	11251500	11748000	11255500
2097152	23721000	23510000	22988000	22996000
4194304	46521000	45999000	47021000	45959000
8388608	90924000	87149000	91505000	81049000
16777216	185517000	191666000	183023000	184032000
33554432	351108000	367772000	389494000	366317000
67108864	732585000	706831000	712832000	740140000
134217728	1483496000	1525666000	1454264000	1472598000

Tabela 5 Czas[ns] usuwania z środka tablicy w zależności od l. elementów i wartości kluczy



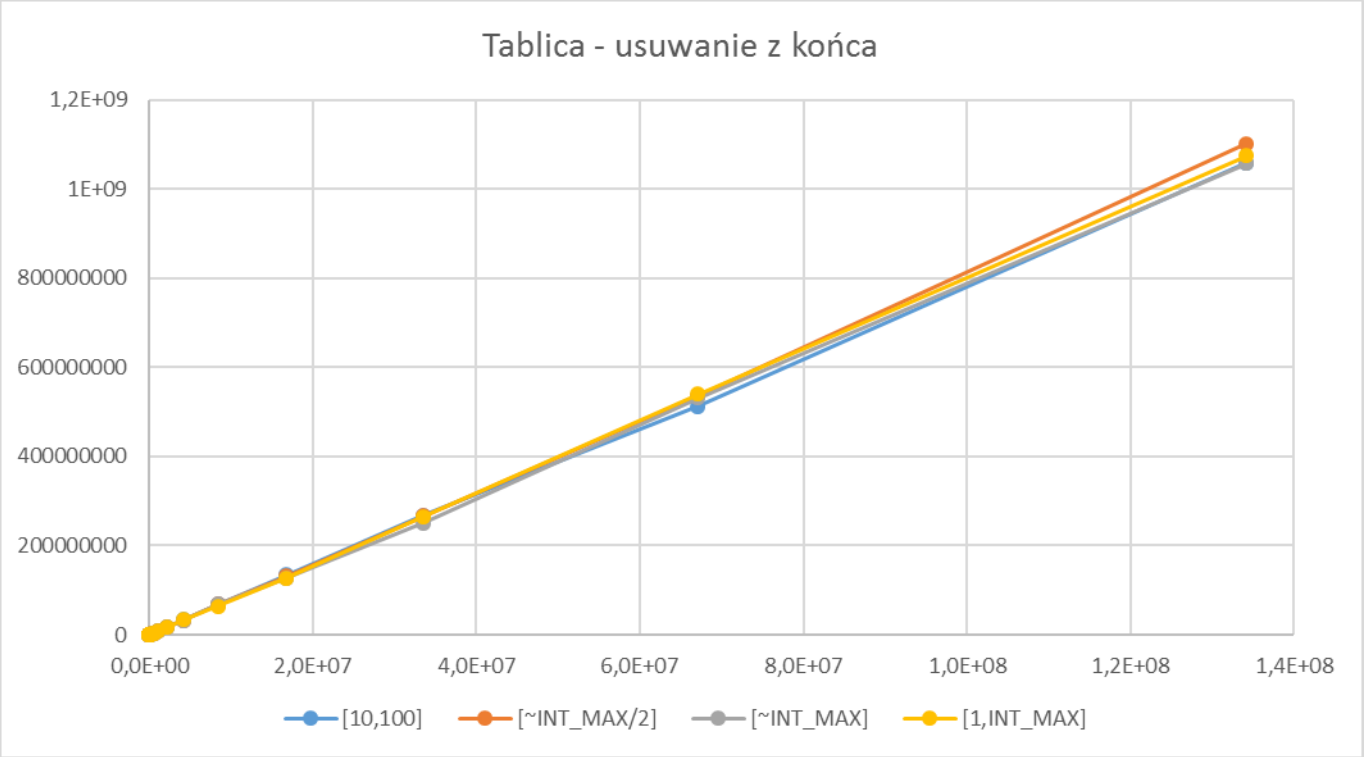
Wykres 5 Zależność między l. elementów a czasem usuwania z środka tablicy dla konkretnych przedziałów wartości

Wyniki przedstawione powyżej dla usuwania z środka tablicy zgadzają się z teorią, która mówi o liniowej zależności czasu wykonywania operacji od liczby elementów znajdujących się w strukturze. Zgodnie z założeniami w części teoretycznej czas usuwania z środka jest krótszy od czasu usuwania z początku tablicy. Brak zauważalnego wpływu wartości kluczy na czas wykonania operacji.

3.1.6. Usuwanie z końca

Usuwanie z końca:				
l. elementów	czas operacji dla konkretnych przedziałów wartości kluczy			
[16,~maxMem]	*[10,100]*	*[~INT_MAX/2]*	*[~INT_MAX]*	*[1,INT_MAX]*
16	214	201	294	211
32	313	294	288	294
64	499	473	492	486
128	793	793	820	819
256	1591	1640	1360	1587
512	2715	2970	2970	2973
1024	5528	5528	5823	5938
2048	11321	11488	11067	11477
4096	23773	22954	23773	24604
8192	45862	44222	45865	45862
16384	88875	95506	95407	91269
32768	185907	184934	199013	184250
65536	415263	368315	355289	407763
131072	868263	762894	816000	789578
262144	2113000	2111222	1889222	2111444
524288	4375500	4377500	4004250	3873750
1048576	7743500	8248500	8247500	8247500
2097152	16997000	17010000	16008000	17013000
4194304	33498000	32005000	33011000	33496000
8388608	68001000	68001000	68016000	64502000
16777216	134024000	132023000	128024000	128024000
33554432	268244000	266814000	251367000	266029000
67108864	512503000	535566000	530689000	539562000
134217728	1058854000	1102192000	1056951000	1074462000

Tabela 6 Czas[ns] usuwania z końca tablicy w zależności od l. elementów i wartości kluczy



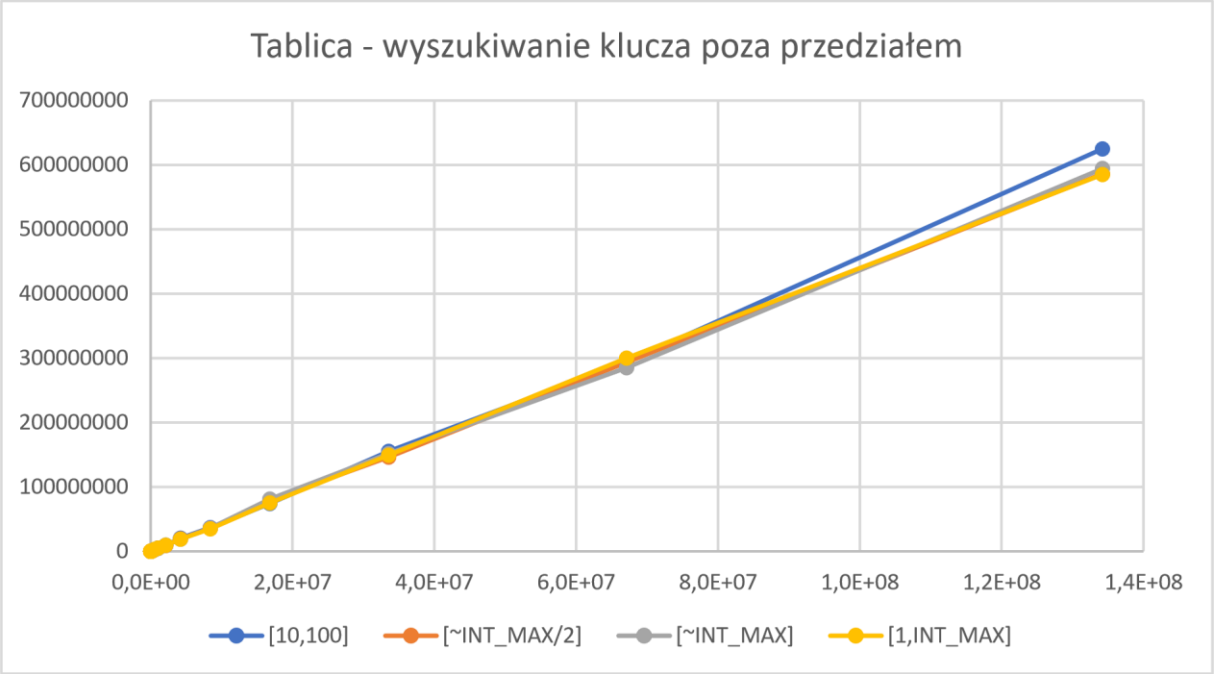
Wykres 6 Zależność między l. elementów a czasem usuwania z końca tablicy dla konkretnych przedziałów wartości

Wyniki przedstawione powyżej dla usuwania z końca tablicy zgadzają się z teorią, która mówi o liniowej zależności czasu wykonywania operacji od liczby elementów znajdujących się w strukturze. Zgodnie z założeniami w części teoretycznej czas usuwania z końca jest krótszy od czasu usuwania z początku oraz z środka tablicy. Brak zauważalnego wpływu wartości kluczy na czas wykonania operacji.

3.1.7. Wyszukiwanie klucza poza przedziałem

Wyszukiwanie - klucz poza przedziałem:				
l. elementów	czas operacji dla konkretnych przedziałów wartości kluczy			
[16,~maxMem]	*[10,100]*	*[~INT_MAX/2]*	*[~INT_MAX]*	*[1,INT_MAX]*
16	83	76	76	120
32	140	147	134	185
64	268	224	256	281
128	562	441	561	475
256	1126	1190	1138	977
512	2253	2048	2150	2047
1024	4091	4299	4094	4299
2048	8197	9010	8801	8602
4096	16393	17754	18036	18024
8192	31127	32773	32786	32770
16384	59177	62513	69092	65796
32768	118618	134763	138157	131618
65536	276315	263157	276342	263184
131072	553421	605263	579368	616684
262144	1332888	1110444	1112888	1110333
524288	2477750	2556250	2500250	2750250
1048576	4747000	4743500	4500500	4754000
2097152	9501000	9001000	9315000	9515000
4194304	20221000	19496000	19017000	19002000
8388608	37004000	36006000	36004000	35013000
16777216	73995000	79647000	81299000	75004000
33554432	155334000	146503000	151520000	149019000
67108864	293947000	293529000	285244000	300046000
134217728	625032000	586212000	594290000	585212000

Tabela 7 Czas[ns] wyszukiwania klucza spoza przedziału w zależności od l. elementów i wartości kluczy



Wykres 7 Zależność między l. elementów a czasem wyszukiwania klucza spoza przedziału dla konkretnych przedziałów wartości

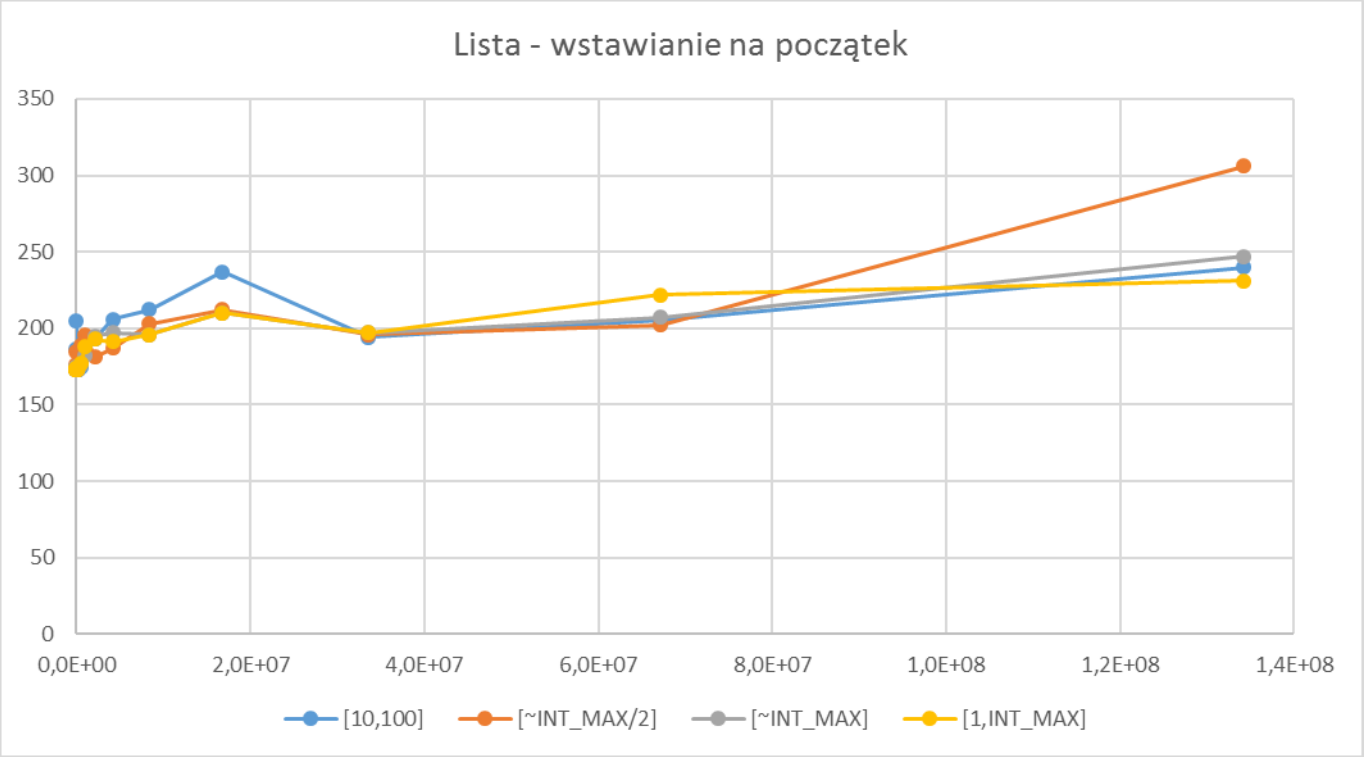
Wyniki przedstawione powyżej dla wyszukiwania elementu spoza przedziału zgadzają się z teorią, która mówi o liniowej zależności czasu wykonywania operacji od liczby elementów znajdujących się w strukturze. Brak zauważalnego wpływu wartości kluczy na czas wykonania operacji, odstępstwo dla wartości z przedziału [10,100] spowodowane prawdopodobnie niedokładnością pomiaru dla największej możliwej liczby elementów.

3.2. Lista dwukierunkowa

3.2.1. Wstawianie na początek

Wstawianie na początek:				
l. elementów	czas operacji dla konkretnych przedziałów wartości kluczy			
[16,~maxMem]	*[10,100]*	*[~INT_MAX/2]*	*[~INT_MAX]*	*[1,INT_MAX]*
16	205	185	175	173
32	173	174	174	173
64	173	174	174	174
128	174	174	175	173
256	173	174	174	173
512	174	176	174	173
1024	174	174	174	173
2048	173	174	174	174
4096	173	175	174	173
8192	173	174	174	173
16384	186	173	173	173
32768	173	173	174	173
65536	173	173	174	173
131072	173	174	174	173
262144	173	175	174	175
524288	175	189	176	177
1048576	189	196	182	188
2097152	193	181	195	193
4194304	206	187	197	191
8388608	212	203	196	196
16777216	237	212	210	210
33554432	194	196	197	197
67108864	205	202	207	222
134217728	240	306	247	231

Tabela 8 Czas[ns] wstawiania na początek listy w zależności od l. elementów i wartości kluczy



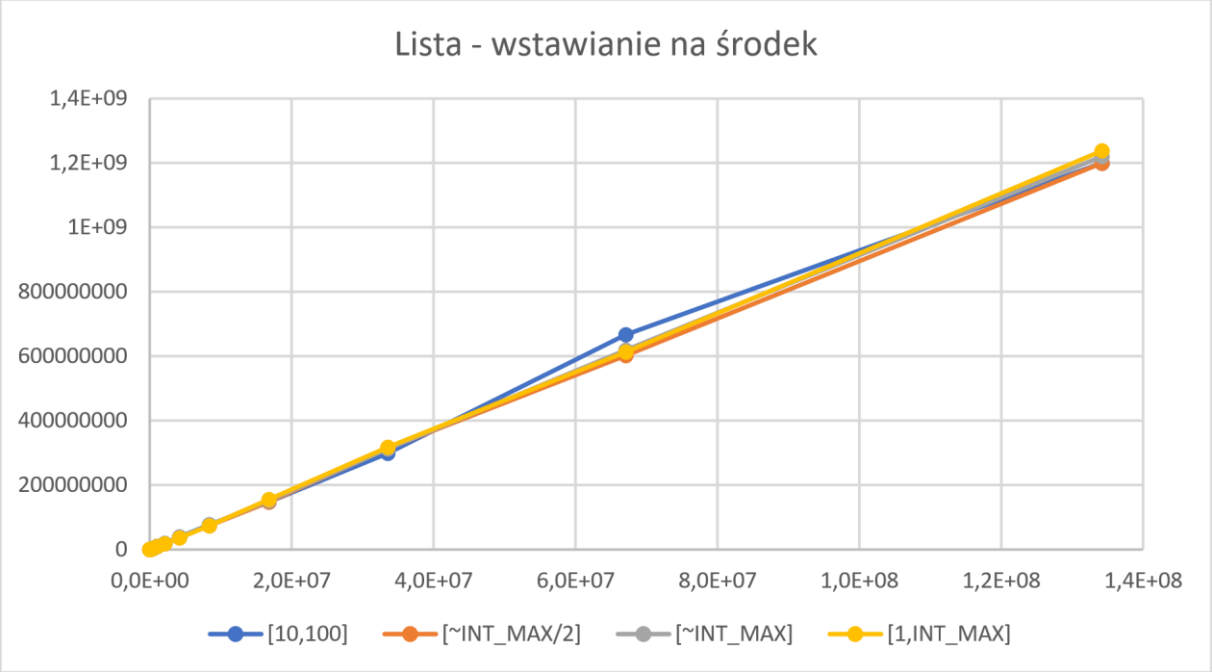
Wykres 8 Zależność między l. elementów a czasem wstawiania na początek listy dla konkretnych przedziałów wartości

Wyniki przedstawione powyżej dla wstawiania na początek listy zgadzają się z teorią, która mówi o stałym czasie wykonywania operacji, niezależnym od liczby elementów. Brak zauważalnego konkretnego wpływu wartości kluczy na czas wykonania operacji, wszelkie zauważalne odstępstwa są spowodowanie jedynie niedokładnością przeprowadzonych pomiarów.

3.2.2. Wstawianie na środek

Wstawianie na środek:				
l. elementów	czas operacji dla konkretnych przedziałów wartości kluczy			
[16,~maxMem]	*[10,100]*	*[~INT_MAX/2]*	*[~INT_MAX]*	*[1,INT_MAX]*
16	164	163	169	164
32	206	207	196	206
64	259	260	270	271
128	366	367	409	366
256	622	622	622	622
512	1053	1391	1394	1134
1024	1988	1988	1988	1988
2048	4722	4038	4038	4722
4096	9501	9519	10188	12250
8192	24516	24516	21800	24530
16384	54595	49087	49246	46393
32768	109841	120841	109830	109896
65536	567028	577895	645206	599984
131072	1273212	1181758	1136394	1092985
262144	2410303	2364303	2454667	2272758
524288	4600540	4798940	4600140	4802940
1048576	10001440	9755940	9499940	8999940
2097152	19510440	18525440	18523440	18012440
4194304	39023940	37525440	38022940	36518940
8388608	77027440	74023440	76525940	74533940
16777216	148045940	148074440	152073440	155572440
33554432	298656440	314115440	313080940	318108440
67108864	666722440	602696940	619211940	613188440
134217728	1199955440	1199980940	1220560440	1237995940

Tabela 9 Czas[ns] wstawiania na środek listy w zależności od l. elementów i wartości kluczy



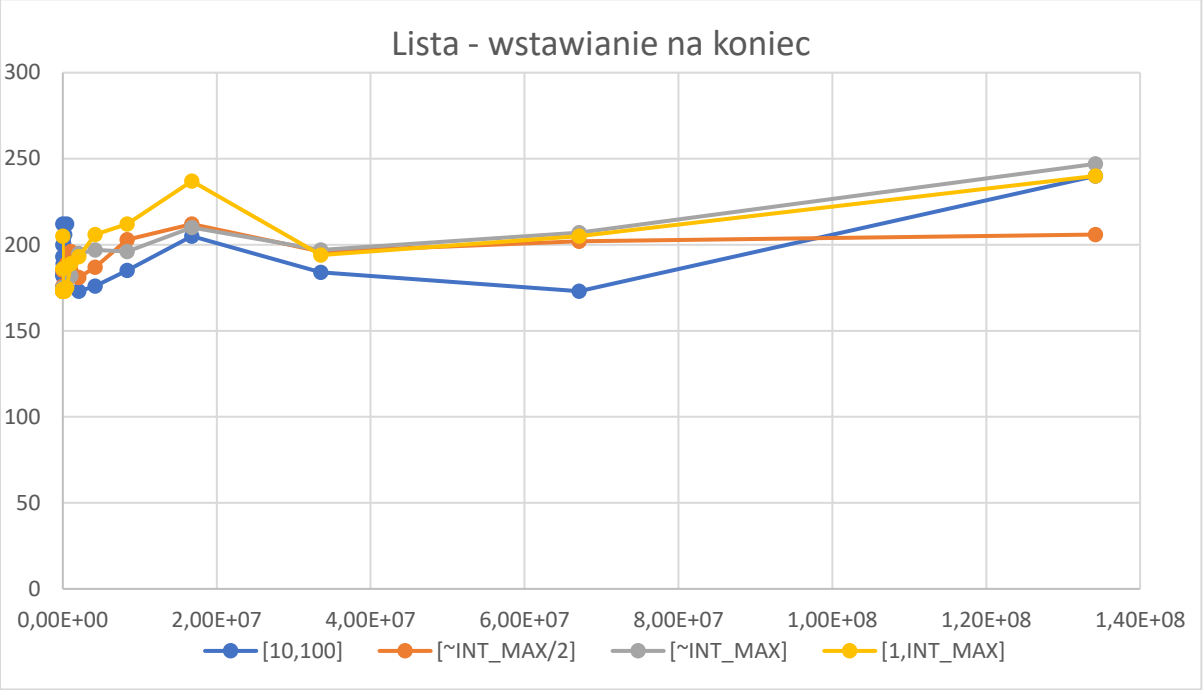
Wykres 9 Zależność między l. elementów a czasem wstawiania na środek listy dla konkretnych przedziałów wartości

Wyniki przedstawione powyżej dla wstawiania na środek listy zgadzają się z teorią, która mówi o liniowej zależności czasu wykonywania operacji od liczby elementów znajdujących się w strukturze. Brak zauważalnego wpływu wartości kluczy na czas wykonania operacji.

3.2.3. Wstawianie na koniec

Wstawianie na koniec:				
l. elementów	czas operacji dla konkretnych przedziałów wartości kluczy			
[16,~maxMem]	*[10,100]*	*[~INT_MAX/2]*	*[~INT_MAX]*	*[1,INT_MAX]*
16	212	185	175	205
32	174	174	174	173
64	200	174	174	173
128	183	174	175	174
256	174	174	174	173
512	173	176	174	174
1024	182	174	174	174
2048	173	174	174	173
4096	175	175	174	173
8192	193	174	174	173
16384	175	173	173	186
32768	173	173	174	173
65536	189	173	174	173
131072	193	174	174	173
262144	206	175	174	173
524288	212	189	176	175
1048576	194	196	182	189
2097152	173	181	195	193
4194304	176	187	197	206
8388608	185	203	196	212
16777216	205	212	210	237
33554432	184	196	197	194
67108864	173	202	207	205
134217728	240	206	247	240

Tabela 10 Czas[ns] wstawiania na koniec listy w zależności od l. elementów i wartości kluczy



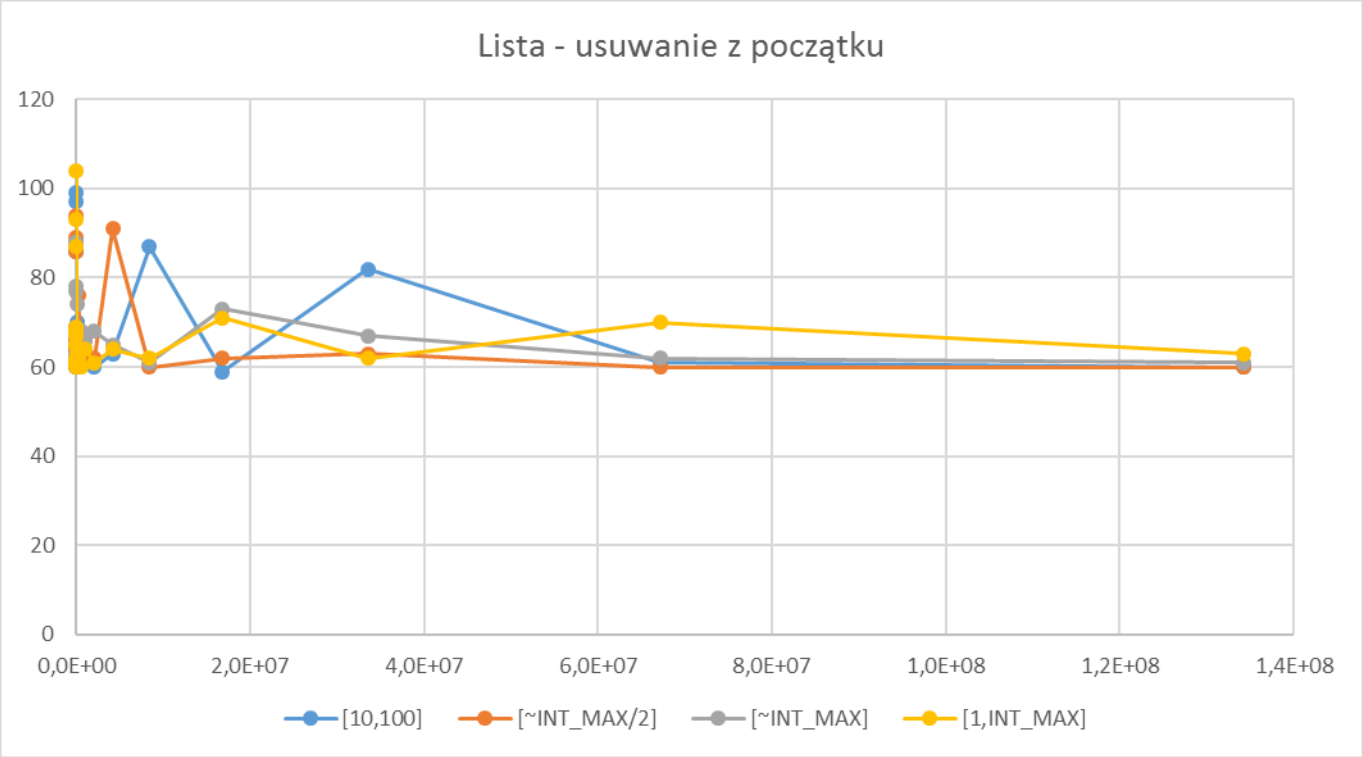
Wykres 10 Zależność między l. elementów a czasem wstawiania na koniec listy dla konkretnych przedziałów wartości

Wyniki przedstawione powyżej dla wstawiania na koniec listy zgadzają się z teorią, która mówi o liniowej zależności czasu wykonywania operacji od liczby elementów znajdujących się w strukturze. Zgodnie z założeniami w części teoretycznej czas wstawiania na koniec jest dłuższy od czasu wstawiania na środek listy. Brak zauważalnego wpływu wartości kluczy na czas wykonania operacji.

3.2.4. Usuwanie z początku

Usuwanie z początku:				
l. elementów	czas operacji dla konkretnych przedziałów wartości kluczy			
[16,~maxMem]	*[10,100]*	*[~INT_MAX/2]*	*[~INT_MAX]*	*[1,INT_MAX]*
16	60	64	65	62
32	61	63	61	65
64	64	66	67	61
128	62	60	63	60
256	60	64	69	67
512	62	62	61	68
1024	65	69	65	69
2048	64	66	77	104
4096	99	89	88	87
8192	86	86	88	93
16384	97	94	78	61
32768	61	62	62	65
65536	60	61	74	61
131072	70	68	66	63
262144	64	76	64	62
524288	61	67	68	60
1048576	66	63	66	64
2097152	60	62	68	61
4194304	63	91	65	64
8388608	87	60	61	62
16777216	59	62	73	71
33554432	82	63	67	62
67108864	61	60	62	70
134217728	60	60	61	63

Tabela 11 Czas[ns] usuwania z początku listy w zależności od l. elementów i wartości kluczy



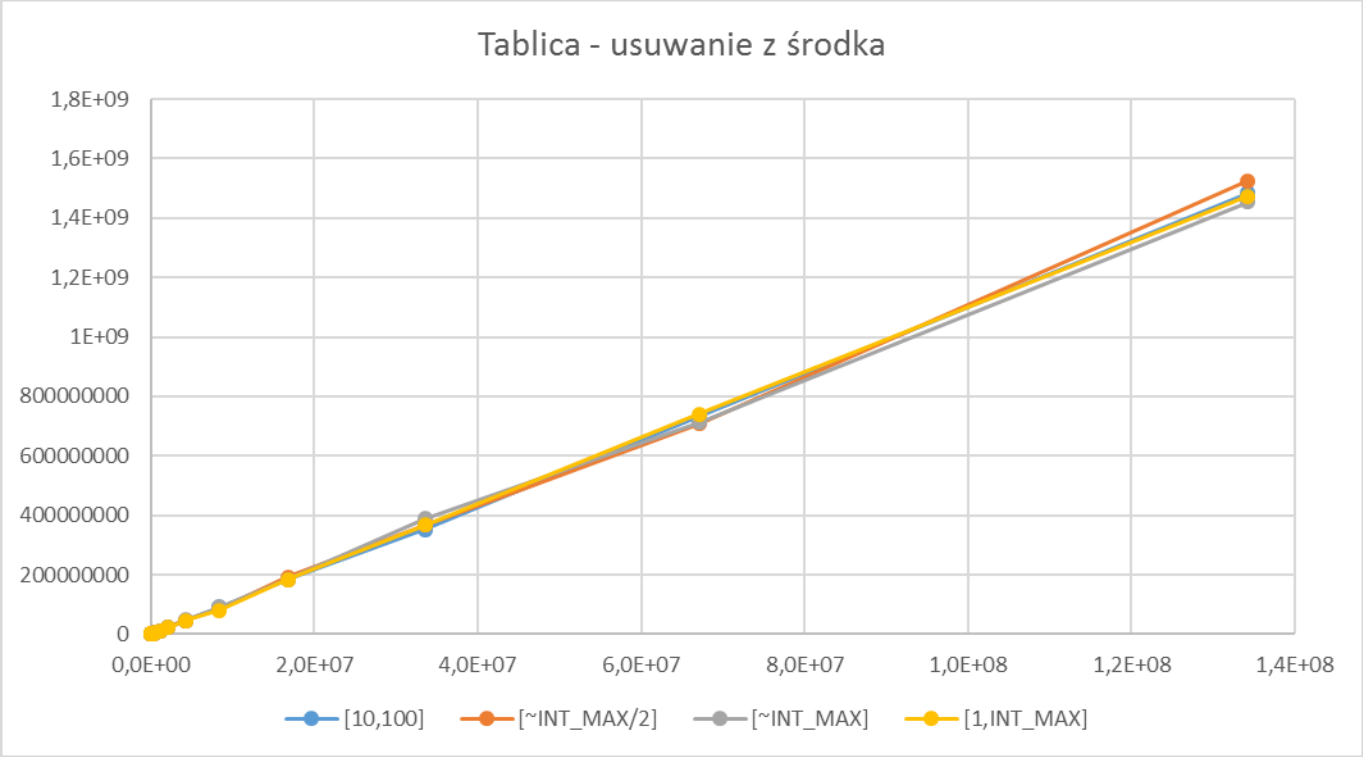
Wykres 11 Zależność między l. elementów a czasem usuwania z początku listy dla konkretnych przedziałów wartości

Wyniki przedstawione powyżej dla usuwania z początku listy zgadzają się z teorią, która mówi o stałym czasie wykonywania operacji, niezależnym od liczby elementów w strukturze. Brak zauważalnego wpływu wartości kluczy na czas wykonania operacji, widoczne odstępstwa od wartości stałej trwania operacji spowodowane są niedokładnością przeprowadzonych badań.

3.2.5. Usuwanie z środka

Usuwanie z środka:				
l. elementów	czas operacji dla konkretnych przedziałów wartości kluczy			
[16,~maxMem]	*[10,100]*	*[~INT_MAX/2]*	*[~INT_MAX]*	*[1,INT_MAX]*
16	100	97	97	100
32	132	132	137	174
64	180	201	180	190
128	329	307	308	371
256	713	585	585	884
512	1225	1054	1141	1140
1024	2591	2079	2079	1908
2048	4295	3622	3958	4637
4096	9424	10107	10105	9433
8192	25819	23064	23092	25835
16384	49078	49018	46313	49078
32768	109761	109838	115189	120750
65536	577548	577726	555637	566615
131072	1181769	1159132	1249723	1317905
262144	2637223	2591132	3091223	2591223
524288	5500460	5399260	4901860	4900460
1048576	9997860	9251360	10250860	10004360
2097152	20248860	19752360	21252360	20502860
4194304	45256360	39005360	38508360	38758360
8388608	77756360	77509360	84760360	79758360
16777216	158022860	157019860	160023360	156269360
33554432	318540360	312039360	321544360	307042360
67108864	639329860	647079360	740357860	674588360
134217728	1349667860	1406927860	1258656360	1288910360

Tabela 12 Czas[ns] usuwania z środka listy w zależności od l. elementów i wartości kluczy



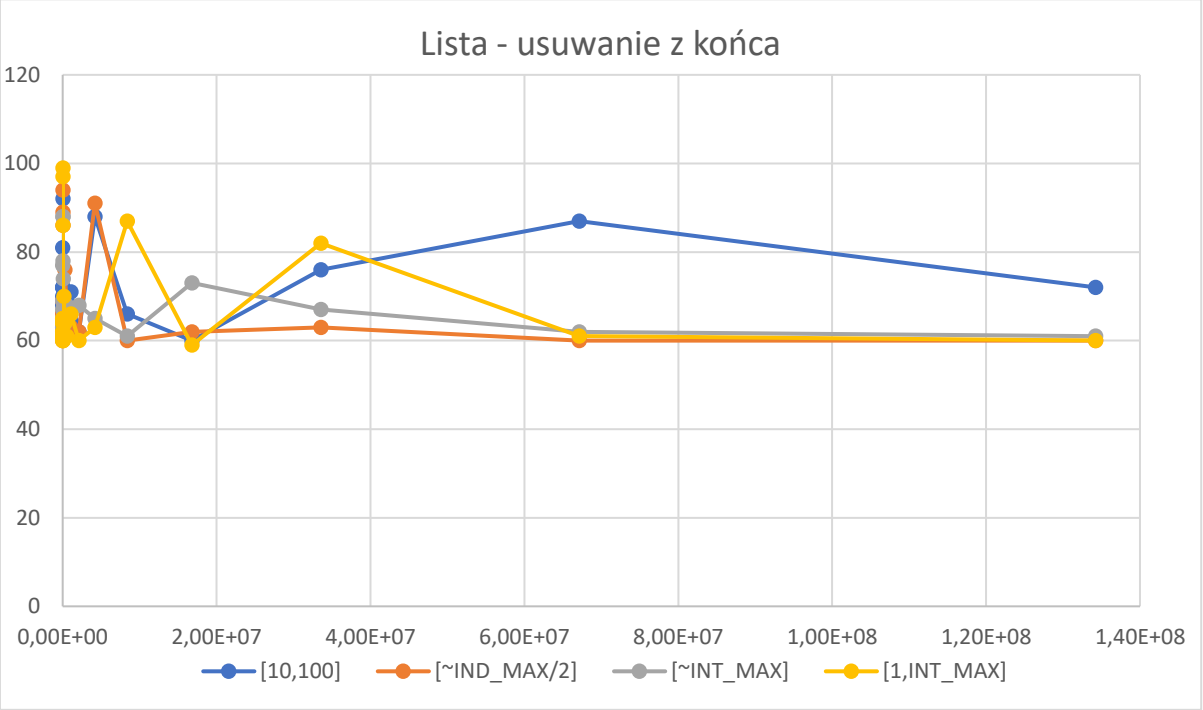
Wykres 12 Zależność między l. elementów a czasem usuwania z środka listy dla konkretnych przedziałów wartości

Wyniki przedstawione powyżej dla usuwania z środka listy zgadzają się z teorią, która mówi o liniowej zależności czasu wykonywania operacji od liczby elementów znajdujących się w strukturze. Brak zauważalnego wpływu wartości kluczy na czas wykonania operacji.

3.2.6. Usuwanie z końca

Usuwanie z końca:				
l. elementów	czas operacji dla konkretnych przedziałów wartości kluczy			
[16,~maxMem]	*[10,100]*	*[~INT_MAX/2]*	*[~INT_MAX]*	*[1,INT_MAX]*
16	63	64	65	60
32	66	63	61	61
64	70	66	67	64
128	68	60	63	62
256	72	64	69	60
512	63	62	61	62
1024	81	69	65	65
2048	72	66	77	64
4096	68	89	88	99
8192	72	86	88	86
16384	63	94	78	97
32768	92	62	62	61
65536	64	61	74	60
131072	64	68	66	70
262144	68	76	64	64
524288	64	67	68	61
1048576	71	63	66	66
2097152	62	62	68	60
4194304	88	91	65	63
8388608	66	60	61	87
16777216	60	62	73	59
33554432	76	63	67	82
67108864	87	60	62	61
134217728	72	60	61	60

Tabela 13 Czas[ns] usuwania z końca listy w zależności od l. elementów i wartości kluczy



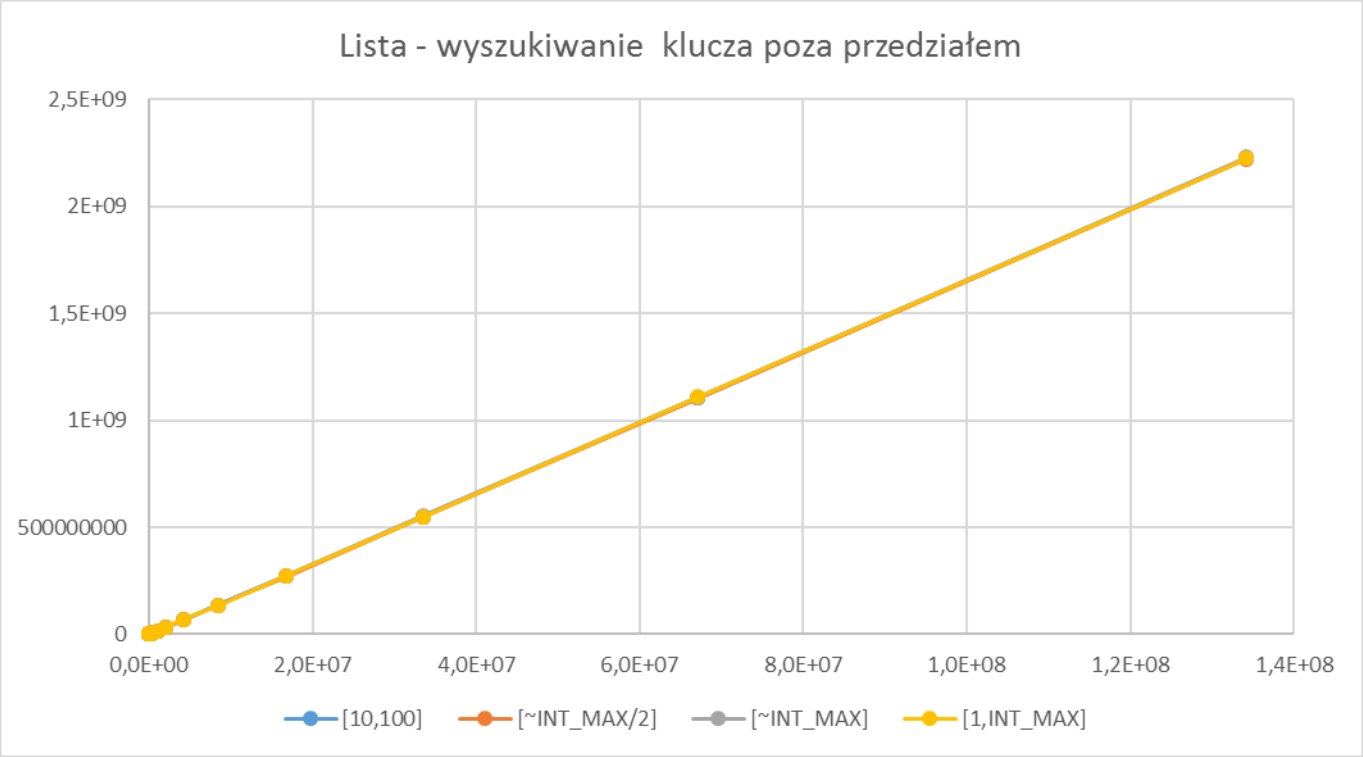
Wykres 13 Zależność między l. elementów a czasem usuwania z końca listy dla konkretnych przedziałów wartości

Wyniki przedstawione powyżej dla usuwania z końca listy zgadzają się z teorią, która mówi o liniowej zależności czasu wykonywania operacji od liczby elementów znajdujących się w strukturze. Zgodnie z założeniami w części teoretycznej czas usuwania z końca jest dłuższy od czasu usuwania z środka listy. Brak zauważalnego wpływu wartości kluczy na czas wykonania operacji.

3.2.7. Wyszukiwanie klucza poza przedziałem

Wyszukiwanie - klucz poza przedziałem:				
l. elementów	czas operacji dla konkretnych przedziałów wartości kluczy			
[16,~maxMem]	*[10,100]*	*[~INT_MAX/2]*	*[~INT_MAX]*	*[1,INT_MAX]*
16	80	74	77	74
32	133	133	138	133
64	256	255	245	256
128	490	490	490	490
256	938	981	938	938
512	1878	2133	2048	2133
1024	4095	3928	3926	3924
2048	9222	8539	8538	8881
4096	19808	19129	19812	19137
8192	39603	38240	39620	39620
16384	112076	114770	117502	114770
32768	511065	494648	494571	489076
65536	933400	988888	1044711	955688
131072	2045681	1841136	2341181	1909045
262144	3909545	4046000	4136272	4182454
524288	8301400	8501000	8401000	8602400
1048576	16505500	16752000	16752000	16252500
2097152	33501000	33755000	32764000	33013000
4194304	66047500	67026500	67759000	69509500
8388608	135786000	136041000	136267500	133805000
16777216	272309000	271561000	273284000	273781500
33554432	549569500	547816000	554320500	549596000
67108864	1106940000	1102714500	1105169000	1109673500
134217728	2225134000	2222172000	2228362000	2225100500

Tabela 14 Czas[ns] wyszukiwania klucza spoza przedziału w zależności od l. elementów i wartości kluczy



Wykres 14 Zależność między l. elementów a czasem wyszukiwania klucza spoza przedziału dla konkretnych przedziałów wartości

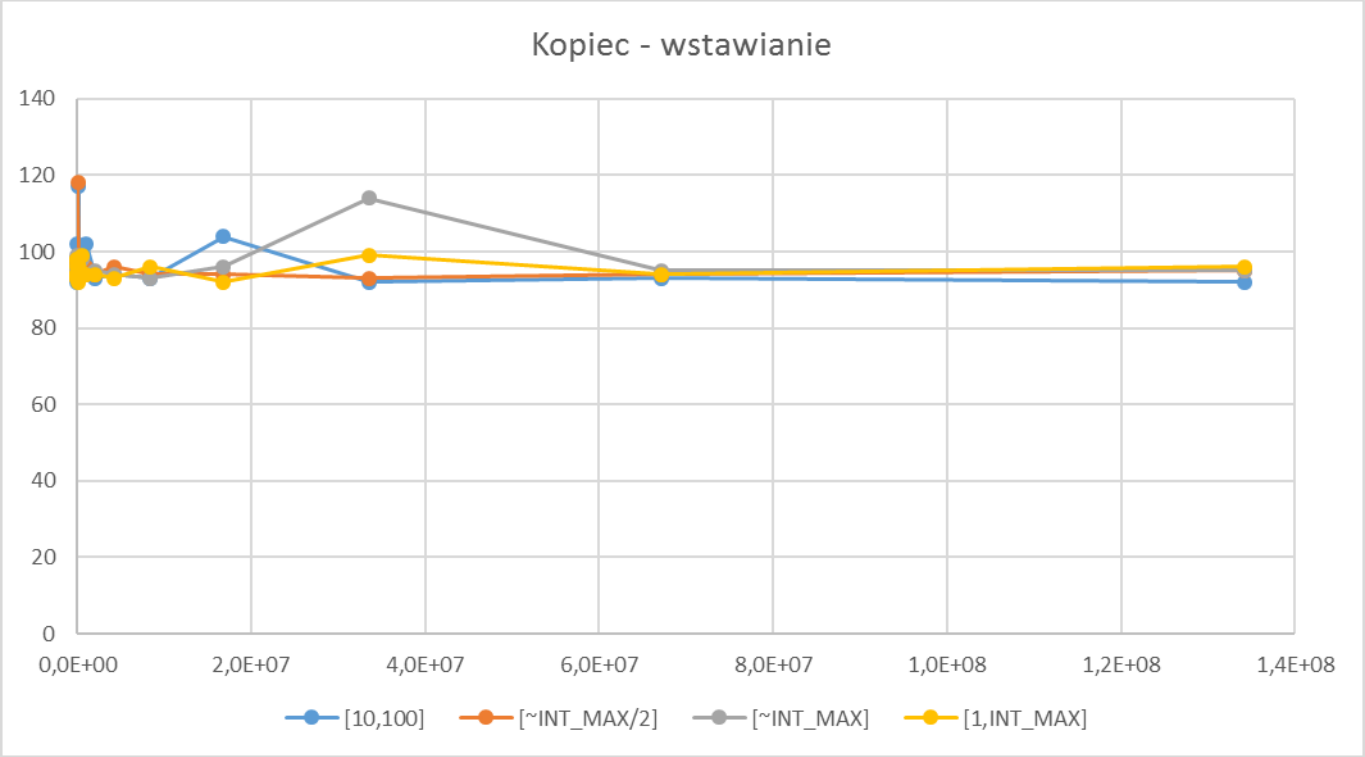
Wyniki przedstawione powyżej dla wyszukiwania elementu spoza przedziału zgadzają się z teorią, która mówi o liniowej zależności czasu wykonywania operacji od liczby elementów znajdujących się w strukturze. Brak zauważalnego wpływu wartości kluczy na czas wykonania operacji.

3.3. Kopiec maksymalny

3.3.1. Wstawianie

Wstawianie na początek:				
l. elementów	czas operacji dla konkretnych przedziałów wartości kluczy			
[16,~maxMem]	*[10,100]*	*[~INT_MAX/2]*	*[~INT_MAX]*	*[1,INT_MAX]*
16	205	185	175	173
32	173	174	174	174
64	173	174	174	174
128	174	174	185	173
256	173	174	174	173
512	174	176	174	173
1024	174	174	174	173
2048	173	174	174	174
4096	173	175	184	193
8192	173	174	174	173
16384	186	183	173	178
32768	173	173	174	173
65536	173	173	174	173
131072	173	174	174	173
262144	173	175	174	185
524288	175	189	176	177
1048576	189	196	182	188
2097152	193	181	195	193
4194304	206	187	197	191
8388608	212	203	196	196
16777216	237	212	190	210
33554432	194	176	197	197
67108864	205	202	207	222
134217728	240	266	227	231

Tabela 15 Czas[ns] wstawiania do kopca w zależności od l. elementów i wartości kluczy



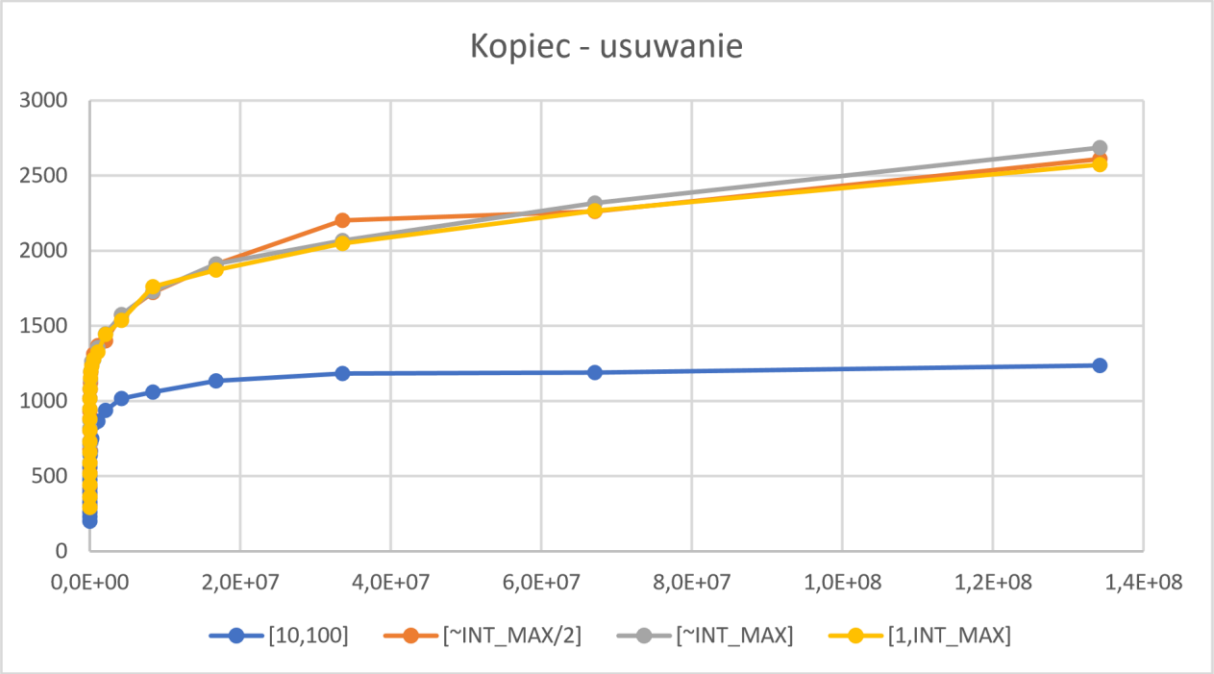
Wykres 15 Zależność między l. elementów a czasem wstawiania do kopca dla konkretnych przedziałów wartości

Wyniki przedstawione powyżej dla wstawiania do kopca zgadzają się z teorią, która mówi o stałym czasie średnim wykonywania operacji, nie zależym od liczby elementów. Brak zauważalnego konkretnego wpływu wartości kluczy na czas wykonania operacji, wszelkie zauważalne odstępstwa są spowodowanie jedynie niedokładnością przeprowadzonych pomiarów.

3.3.2. Usuwanie

Usuwanie:				
l. elementów	czas operacji dla konkretnych przedziałów wartości kluczy			
[16,~maxMem]	*[10,100]*	*[~INT_MAX/2]*	*[~INT_MAX]*	*[1,INT_MAX]*
16	200	295	292	293
32	231	364	364	363
64	260	438	442	443
128	301	522	515	515
256	327	589	591	589
512	357	675	710	659
1024	390	734	731	729
2048	421	804	819	806
4096	476	881	873	881
8192	506	933	947	947
16384	559	1020	1017	1014
32768	639	1076	1083	1081
65536	667	1122	1144	1154
131072	726	1189	1199	1199
262144	751	1244	1264	1232
524288	848	1314	1276	1280
1048576	866	1368	1357	1327
2097152	939	1400	1447	1443
4194304	1016	1571	1575	1538
8388608	1060	1722	1729	1762
16777216	1134	1911	1912	1871
33554432	1183	2203	2068	2049
67108864	1189	2263	2317	2266
134217728	1237	2611	2687	2574

Tabela 16 Czas[ns] usuwania z kopca w zależności od l. elementów i wartości kluczy



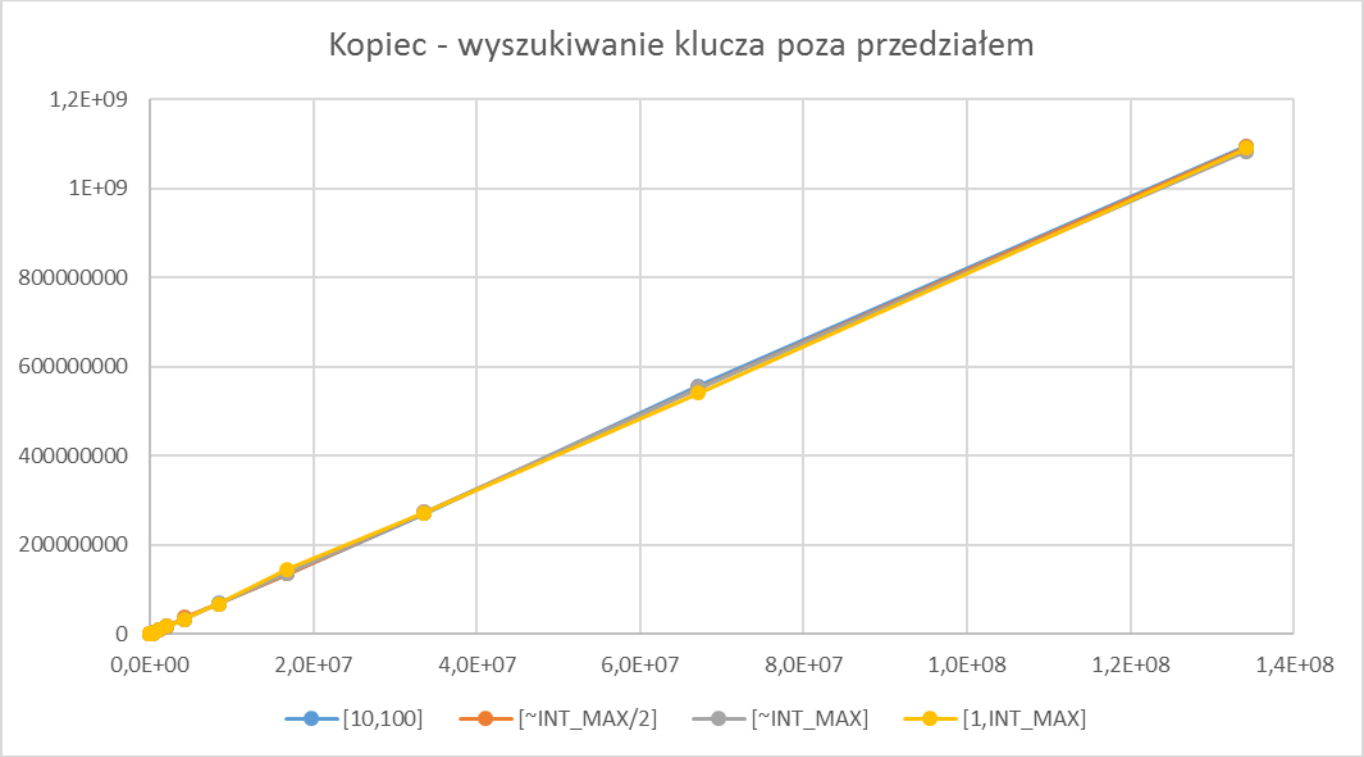
Wykres 16 Zależność między l. elementów a czasem usuwania z kopca dla konkretnych przedziałów wartości

Wyniki przedstawione powyżej dla usuwania z kopca zgadzają się z teorią, która mówi o logarytmicznej zależności czasu wykonywania operacji od liczby elementów znajdujących się w strukturze. Usuwanie wykonywane jest widocznie szybciej dla kluczy o wartości z przedziału [10,100], co wynika z małego rozrzutu wartości, co przy większej liczbie elementów drzewa sprowadza średnią liczbę wykonywanych porównań coraz to bliżej wartości stałej.

3.3.3. Wyszukiwanie klucza poza przedziałem

Wyszukiwanie - klucz poza przedziałem:				
l. elementów	czas operacji dla konkretnych przedziałów wartości kluczy			
[16,~maxMem]	*[10,100]*	*[~INT_MAX/2]*	*[~INT_MAX]*	*[1,INT_MAX]*
16	138	138	137	137
32	259	257	258	256
64	509	522	511	500
128	998	1031	988	988
256	1951	1977	2000	1956
512	3954	3933	3923	4199
1024	7786	7775	7744	7754
2048	16183	15692	15652	15490
4096	31633	31306	31139	30940
8192	63512	65068	62287	61796
16384	124256	124579	123947	123949
32768	251571	250925	250624	251926
65536	504818	502183	505464	502871
131072	1231196	1017509	1050123	1021296
262144	2100805	2106278	2121652	2111363
524288	4212610	4222915	4222831	4212557
1048576	8492531	8599085	8472297	8471829
2097152	17442130	17005260	16895478	17181000
4194304	33786272	37188818	34009000	33920545
8388608	69121600	67678600	68131800	68034200
16777216	135911000	135456100	136607600	145751400
33554432	270386200	272867900	272915500	271525400
67108864	556468800	552314900	553188400	540853600
134217728	1094173100	1093638300	1083530200	1089659400

Tabela 17 Czas[ns] wyszukiwania klucza spoza przedziału w zależności od l. elementów i wartości kluczy



Wykres 17 Zależność między l. elementów a czasem wyszukiwania klucza spoza przedziału dla konkretnych przedziałów wartości

Wyniki przedstawione powyżej dla wyszukiwania elementu spoza przedziału zgadzają się z teorią, która mówi o liniowej zależności czasu wykonywania operacji od liczby elementów znajdujących się w strukturze. Brak zauważalnego wpływu wartości kluczy na czas wykonania operacji.

4. Podsumowanie

- Przedział wartości kluczy nie ma widocznego wpływu na szybkość wykonywania operacji. Jest to stwierdzenie prawdziwe dla wszystkich testowanych operacji za wyjątkiem usuwania z kopca, gdzie przez zastosowany algorytm, który jest zbliżony swoim działaniem do operacji sortowania, czas wykonywania operacji dla małego przedziału danych zbliża się do czasu stałego wraz ze wzrostem liczby danych.

- Ze względu na różne czasy wykonywania poszczególnych operacji, różne struktury nadają się do innych celów np.
    - Operacje na tablicy z realokacją pamięci przy dużych ilościach danych zajmują stosunkowo dużo czasu, lecz dzięki indeksowaniu danych, możliwości odniesienia się do nich bezpośrednio. Struktura ta sprawdza się, gdy nie ulega nadmiernej modyfikacji
    - W przypadku listy dwukierunkowej najszybciej wykonywanymi operacjami są operacje wstawiania i usuwania z początku oraz końca listy. Przykładowym zastosowaniem są struktury dynamicznie zmieniające się jak np. stos, gdzie ostatni dodany element jest zawsze pierwszym pozyskiwanym elementem (LIFO)
    - Zastosowaniem kopca może być kolejka priorytetowej, gdyż w korzeniu kopca zawsze mamy wartość największą lub najmniejszą(zależnie od implementacji), a czas pozyskania jej nieznacznie rośnie wraz ze wzrostem liczby elementów. Wstawianie do kopca również nie zajmuje dużo czasu, a średnio przyjmuje wartość stałą czasu wykonywania operacji