Projekt 3

do samodzielnego wykonania

I.

1. Porównaj szybkość działania 4 metod sortowania: *Insertion Sort*, *Selection Sort*, *Heap Sort*, *Cocktail Sort* dla tablicy liczb całkowitych (rzędu 50k - 200k elementów) generowanych w postaci: losowej, rosnącej, malejącej, stałej, v-kształtnej.

Przedstaw wykresy *t = f(n)* dla każdej z metod w zależności od postaci tablicy wejściowej, gdzie: t - czas sortowania; n - liczba elementów tablicy.

Liczbę elementów należy dobrać w taki sposób, aby możliwe było wykonanie pomiarów. Wyniki przedstawić na czterech wykresach (przynajmniej 15 punktów pomiarowych).

2. Sformułuj wnioski odnośnie:

- złożoności obliczeniowej badanych metod i ich związku z efektywnością sortowania oraz zajętością pamięciową każdej z nich,
- wpływu postaci ciągów wejściowych na czas sortowania (najgorsze, najlepsze przypadki).

II.

1. Dla różnych typów danych wejściowych porównaj efektywność działania powyższych algorytmów.
   Przedstaw wykresy *t = f(n)* dla każdego typu danych wejściowych i różnych metod sortowania (5 wykresów).
   Liczbę elementów należy dobrać w taki sposób, aby możliwe było wykonanie pomiarów.
2. Sformułować wnioski odnośnie złożoności obliczeniowej i efektywności wykonywania oraz zachowania się algorytmów dla poszczególnych typów danych wejściowych.

III.

1. Zaimplementuj algorytm *Quicksort* w dwóch wersjach: rekurencyjnie oraz iteracyjnie (z własną implementacją stosu).
   Porównaj obie wersje na wspólnym wykresie przy sortowaniu ciągu losowego. Następnie porównaj różne sposoby wyboru klucza do porównania: skrajnie prawego, środkowego co do położenia, losowo wybranego. Utwórz wykres porównujący efektywność działania algorytmu (iteracyjnego lub rekurencyjnego) w zależności od wyboru różnego klucza dla A-kształtnego ciągu wejściowego (przynajmniej 15 punktów pomiarowych).
2. Sformułuj wnioski dotyczące złożoność badanych algorytmów. Jak wybór klucza wpływa na działanie algorytmu (najgorsze, najlepsze przypadki)?

Tablica malejąca:

Funkcja:

```
static int[] Malejace(int[] tab)
{
    int tab2 = tab.Length;
    for (int i = 0; i < tab.Length; i++)
    {
        tab[i] = tab2;
        tab2 -= 1;
    }
    return tab;
}
```

Wywołanie funkcji:

```
Console.WriteLine("tablica malejąca");
Console.WriteLine("Rozmiar" + " " + "HeapSort" + " " + "coctailSort" + " " + "Insertsort"+ " " +"  Selectsort ");
for (int j = 50000; j < 200000; j += 10000)
{

    int[] tab1 = new int[j];
    int[] tab2 = new int[j];
    int[] tab3 = new int[j];
    int[] tab4 = new int[j];

    Malejace(tab1);
    Malejace(tab2);
    Malejace(tab3);
    Malejace(tab4);
```

Tablica rosnąca:

Funkcja

```
static int[] Rosnaca(int[] tab)
{
    int tab2 = tab.Length;
    for (int i = 0; i < tab.Length; i++)
    {
        tab[i] = tab2;
        tab2 += 1;
    }
    return tab;
}
```

Wywołanie funkcji

```
Console.WriteLine("Tablica rosnąca");
Console.WriteLine("rozmiar" + " " + "HeapSort" + " " + "CocktailSort" + " " + "InsertSort" + " " + "SelectSort");
for (int j = 50000; j < 200000; j += 10000)
{

    int[] tab1 = new int[j];
    int[] tab2 = new int[j];
    int[] tab3 = new int[j];
    int[] tab4 = new int[j];

    Rosnaca(tab1);
    Rosnaca(tab2);
    Rosnaca(tab3);
    Rosnaca(tab4);
```

Tablica losowa:

Funkcja

```
static int[] Randomowa(int[] tab)
{
    Random random = new Random();
    int tab2 = tab.Length;
    for (int i = 0; i < tab.Length; i++)
    {
        tab[i] = tab2;
        tab2 = random.Next(50000, 200000);


    }
    return tab;
}
```

Wywołanie funkcji

```
Console.WriteLine("Tablica randomowa");
Console.WriteLine("rozmiar" + " " + "HeapSort" + " " + "CocktailSort" + " " + "InsertSort" + " " + "SelectSort");
Random random = new Random();

for (int j = 50000; j < 200000; j += 10000)
{

    int[] tab1 = new int[j];
    int[] tab2 = new int[j];
    int[] tab3 = new int[j];
    int[] tab4 = new int[j];

    Randomowa(tab1);
    Randomowa(tab2);
    Randomowa(tab3);
    Randomowa(tab4);
```

Tablica stała

Funkcja

```
static int[] Stala(int[] tab)
{
    int tab2 = tab.Length;
    for (int i = 0; i < tab.Length; i++)
    {
        tab[i] = 5;


    }
    return tab;
}
```

Wywołanie funkcji

```
Console.WriteLine("Tablica stała");
Console.WriteLine("rozmiar" + " " + "HeapSort" + " " + "CocktailSort" + " " + "InsertSort" + " " + "SelectSort");

for (int j = 50000; j < 200000; j += 10000)
{

    int[] tab1 = new int[j];
    int[] tab2 = new int[j];
    int[] tab3 = new int[j];
    int[] tab4 = new int[j];

    Stala(tab1);
    Stala(tab2);
    Stala(tab3);
    Stala(tab4);
```

Tablica V-kształtna

Funkcja

```
static int[] Vksztaltna(int[] tab)
{

    int rozmiar = tab.Length;
    int przyrost = rozmiar / 2;
    int licznik = 1;
    int x = 1;
    tab[przyrost] = 1;
    while (licznik < rozmiar)
    {
        if (przyrost - x >= 0)
            tab[przyrost - x] = ++licznik;
        if (przyrost + x < rozmiar)
            tab[przyrost + x++] = ++licznik;
    }
    return tab;

}
```

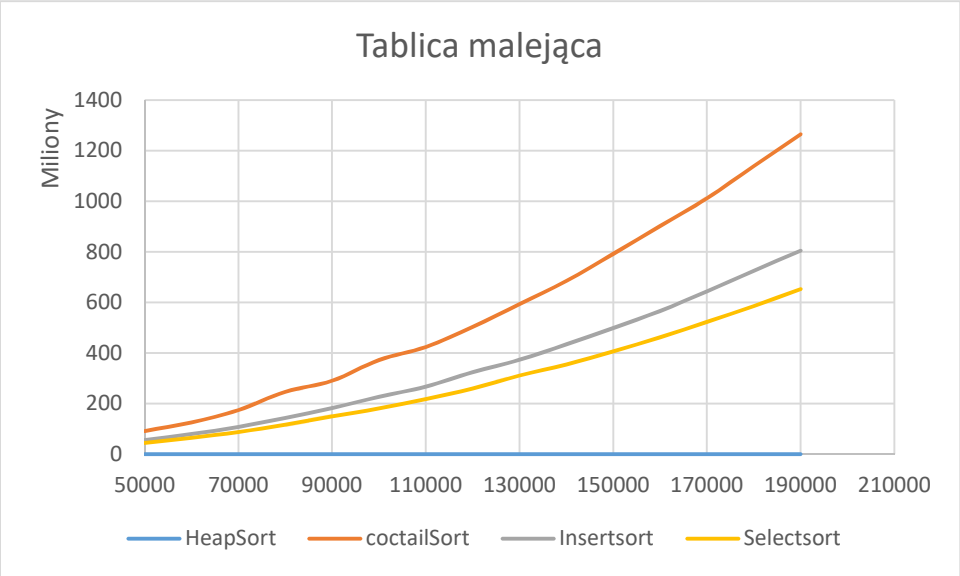
Wywołanie funkcji

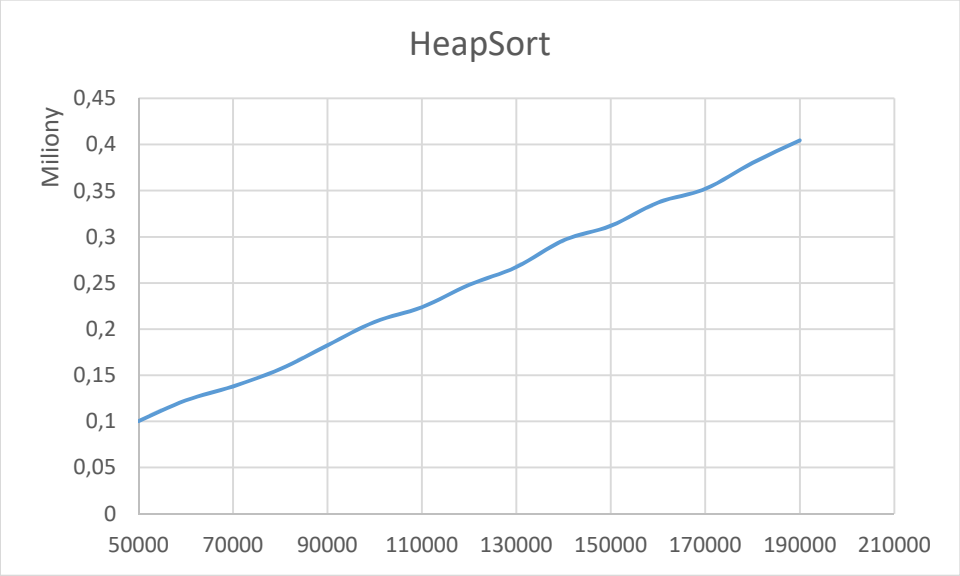
```
static void Main(string[] args)
{
    Console.WriteLine("Tablica Vksztaltna");
    Console.WriteLine("rozmiar" + " " + "HeapSort" + " " + "CocktailSort" + " " + "InsertSort" + " " + "SelectSort");
    for (int j = 50000; j < 200000; j += 10000)
    {

        int[] tab1 = new int[j];
        int[] tab2 = new int[j];
        int[] tab3 = new int[j];
        int[] tab4 = new int[j];

        Vksztaltna(tab1);
        Vksztaltna(tab2);
        Vksztaltna(tab3);
        Vksztaltna(tab4);
```

Wykresy:

Tablica malejąca

Rozmiar	HeapSort	coctailSort	Insertsort	Selectsort
50000	100251	91827330	56484053	44966082
60000	122745	126151612	79865920	65461182
70000	137953	174497535	108541095	88136644
80000	156677	246782188	143530021	117343207
90000	182469	290594466	182532306	149407722
100000	207909	372488263	227295001	181393120
110000	223776	423961911	267332372	218483397
120000	247983	503360350	324296545	259888230
130000	267267	593407525	373387297	311249324
140000	296209	684849428	434175422	354956548
150000	311964	791490611	498872894	406615395
160000	337150	901741907	566060817	462287758
170000	351941	1011464935	643767117	522846889
180000	380083	1138229368	725023232	585854158
190000	404551	1264500185	803976995	652494412





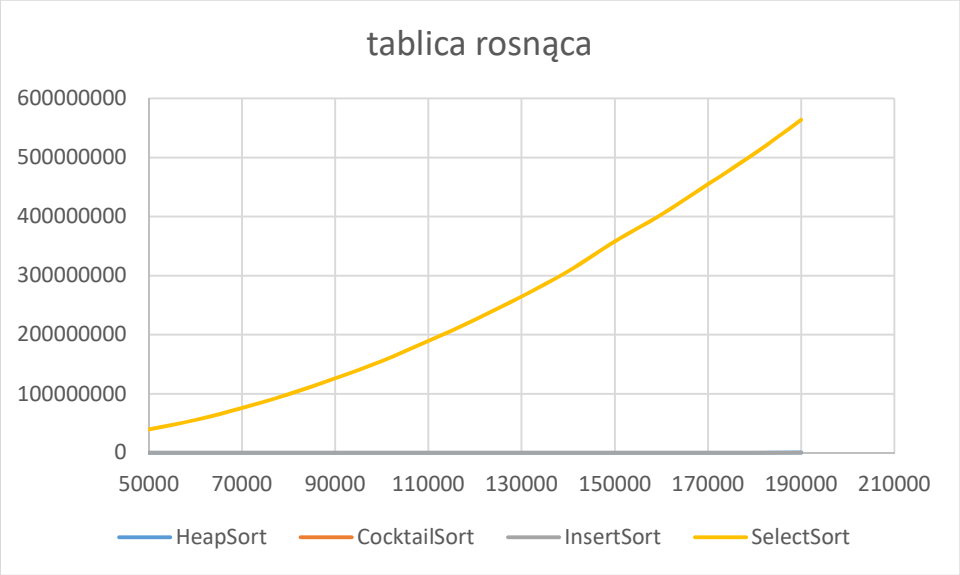
Wnioski: Heap Sort jest najszybszy Cocktail Sort najwolniejszy

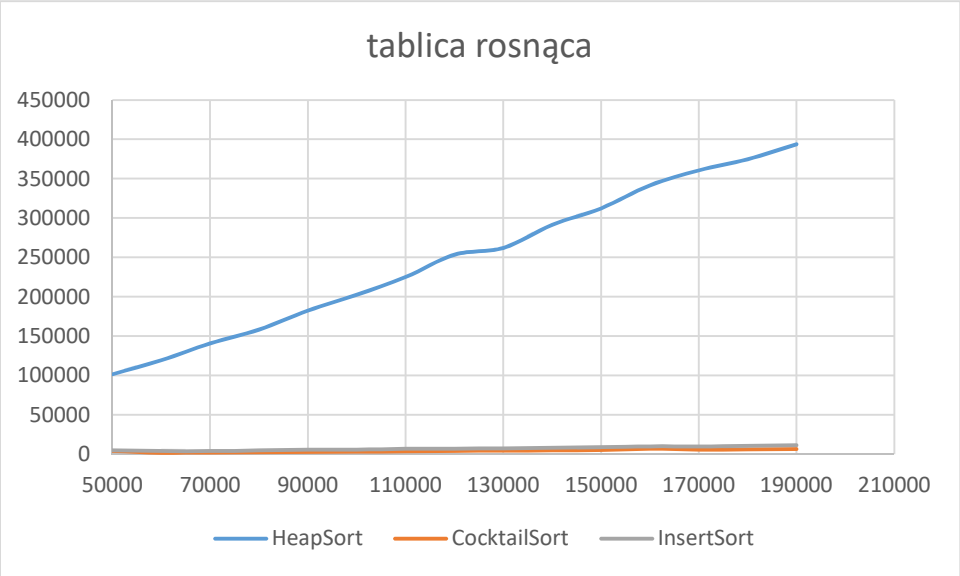
Stworzyłem nowy osobny wykres dla Heap Sortu bo skala była zbyt duża i nie było widać go na wykresie razem z innymi metodami sortowania.

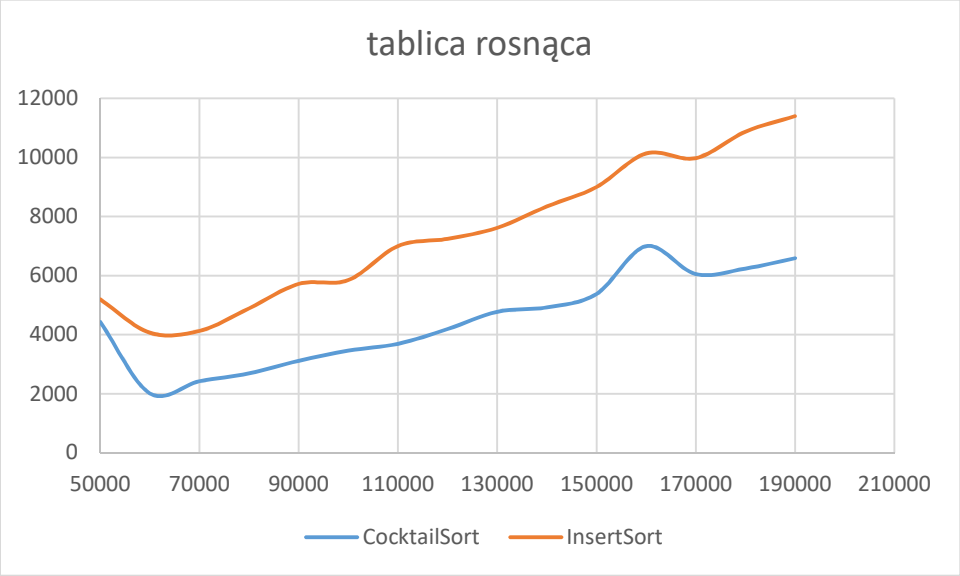
Tablica rosnąca

rozmiar	HeapSort	CocktailSc	InsertSort	SelectSort
50000	101718	4428	5192	39527338
60000	119483	2010	4064	55694924
70000	140807	2421	4124	75878723
80000	158354	2685	4890	99329292
90000	182464	3114	5717	126054757
100000	202619	3457	5847	155430906
110000	225107	3689	7000	189333289
120000	253517	4187	7243	225448367
130000	262108	4773	7615	264719258
140000	291443	4922	8340	307408120
150000	312343	5378	8998	357951882
160000	341589	6994	10135	403352351
170000	360616	6051	9974	454686096
180000	374699	6236	10871	506673280
190000	393653	6586	11397	563654796

Wykresy:



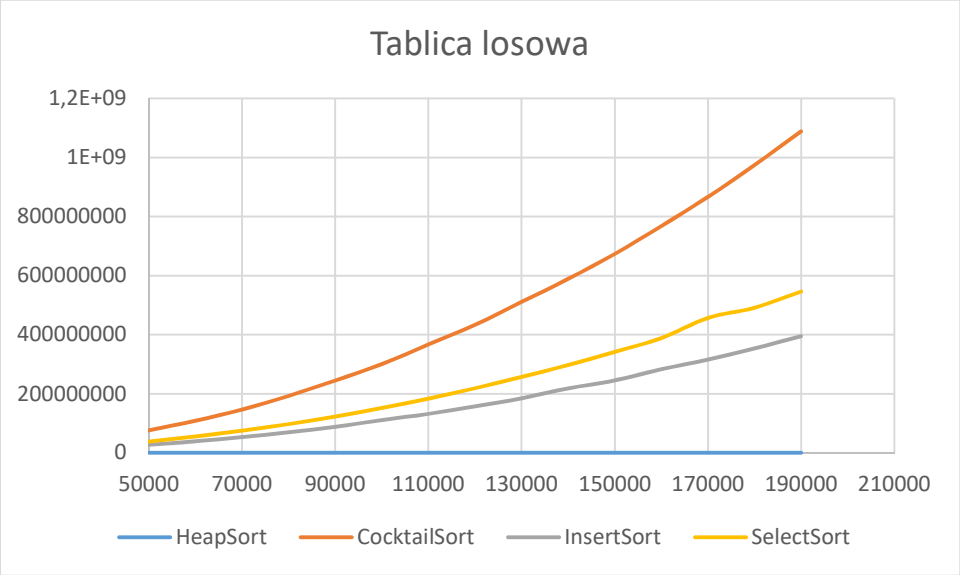


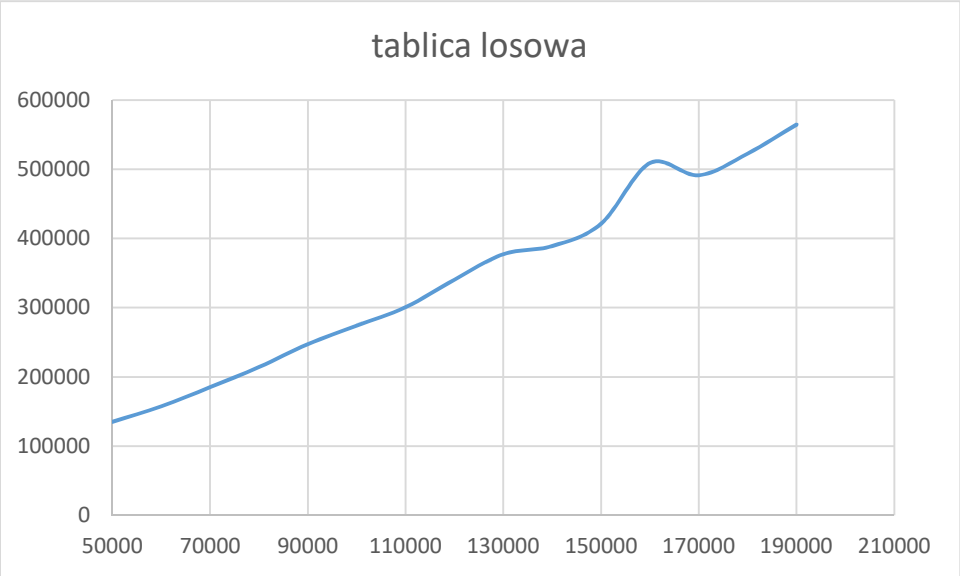


Wnioski: Select Sort najwolniejszy, Cocktail Sort najszybszy

Tablica losowa

rozmiar	HeapSort	CocktailSort	InsertSort	SelectSort
50000	135028	76116790	27188631	38044224
60000	157542	108753883	39510922	55230283
70000	185091	146620874	53176088	75071588
80000	214230	192862634	69725390	97251571
90000	247177	244944790	88405752	123071155
100000	274069	300438487	110784233	151600134
110000	300628	366671204	131969661	183517616
120000	340459	433570775	157230300	218779998
130000	377216	510952852	184696118	257272412
140000	388946	589169195	218209511	297235515
150000	421255	673554737	245001663	341935334
160000	508796	767915148	282852572	388514084
170000	491185	866622778	315838807	456729526
180000	522586	974484984	353450286	491269041
190000	564419	1088294267	394912191	546363601



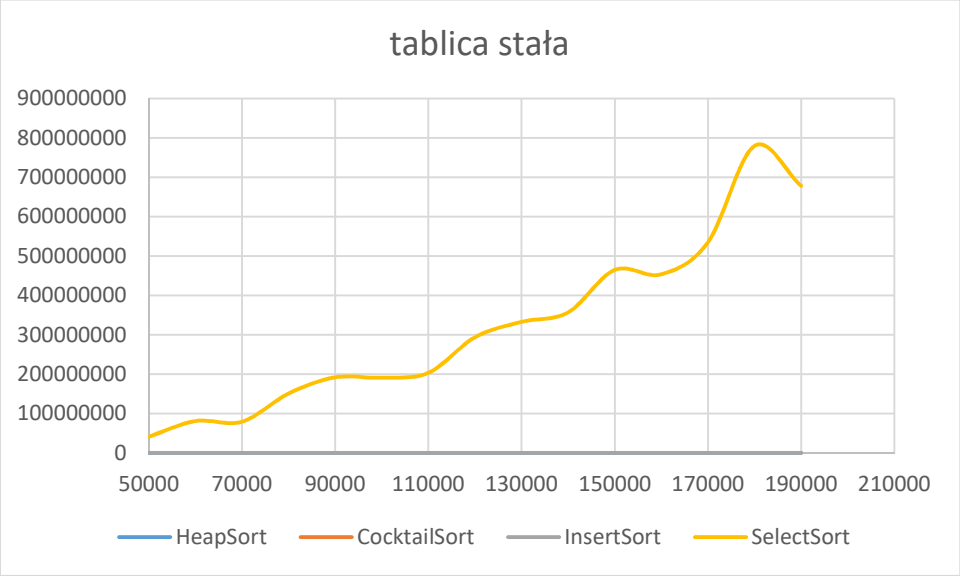


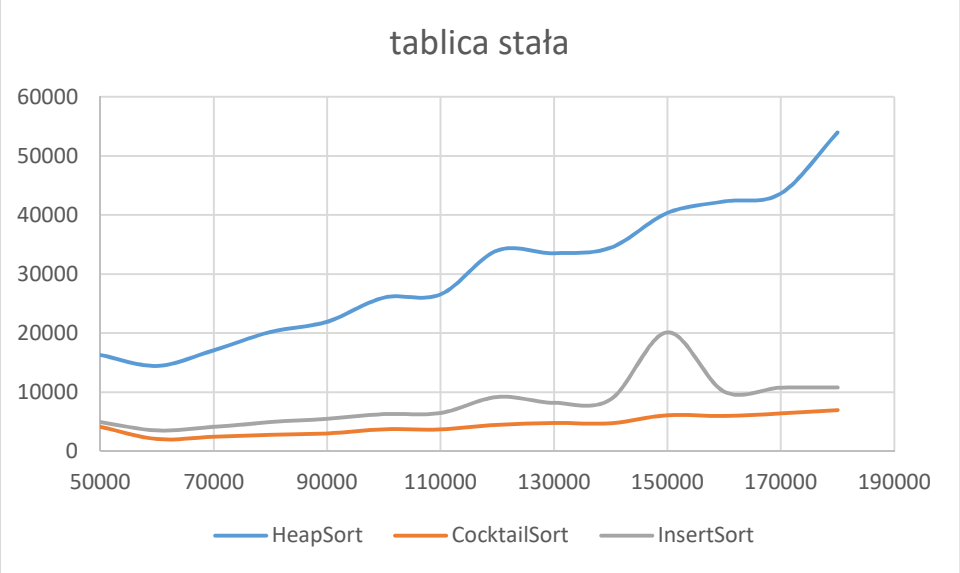
Wnioski: Cocktail Sort najwolniejszy Heap Sort najszybszy

Tablica Stała

rozmiar	HeapSort	CocktailSc	InsertSort	SelectSort
50000	16311	4131	4946	40983153
60000	14455	2074	3519	80753275
70000	17099	2444	4119	79265476
80000	20198	2767	4935	150864678
90000	21929	3014	5498	191961199
100000	26009	3722	6289	190873359
110000	26552	3704	6481	203302558
120000	33992	4474	9185	293590645
130000	33534	4792	8207	332756109
140000	34465	4744	8791	356773039
150000	40347	6112	20130	464615972
160000	42296	5959	10106	453368484
170000	43657	6412	10777	534613827
180000	53975	6978	10794	779499859
190000	50627	6953	11358	678022528

Wykresy:

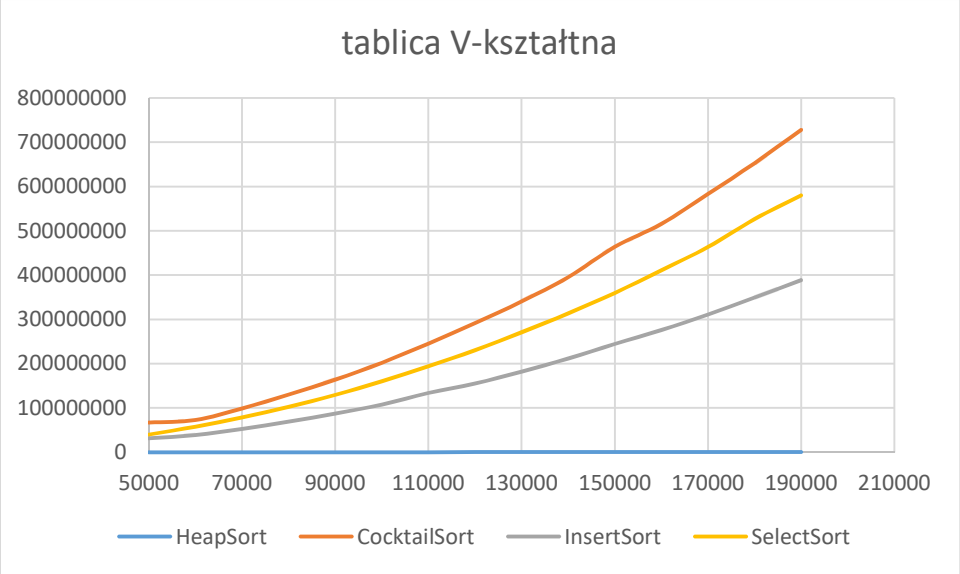


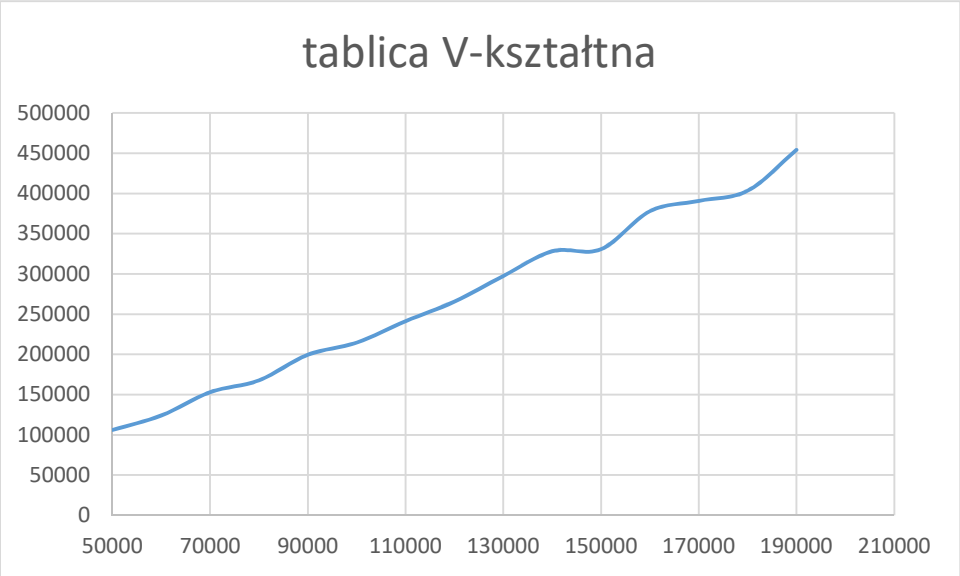


Wnioski: Najwolniejszy Select Sort najszybszy Cocktail Sort

Tablica V-kształtna

rozmiar	HeapSort	CocktailSort	InsertSort	SelectSort
50000	106063	67131489	31353728	39994133
60000	124117	72791849	38800161	57753099
70000	153039	99040548	52844740	78423654
80000	167801	130371777	69068988	102510595
90000	199526	163984028	87210903	129533011
100000	214699	202064607	107811858	160326031
110000	241326	245517285	133827933	194201688
120000	265574	291483230	155256203	230518813
130000	297447	340711186	182155654	270865788
140000	328187	395268586	211539512	313542631
150000	330734	463759695	244565034	359746467
160000	377933	515667437	275929098	410803659
170000	390760	582839952	311170704	463347831
180000	403426	652739508	349068444	526437028
190000	454037	727649886	388633855	580078952



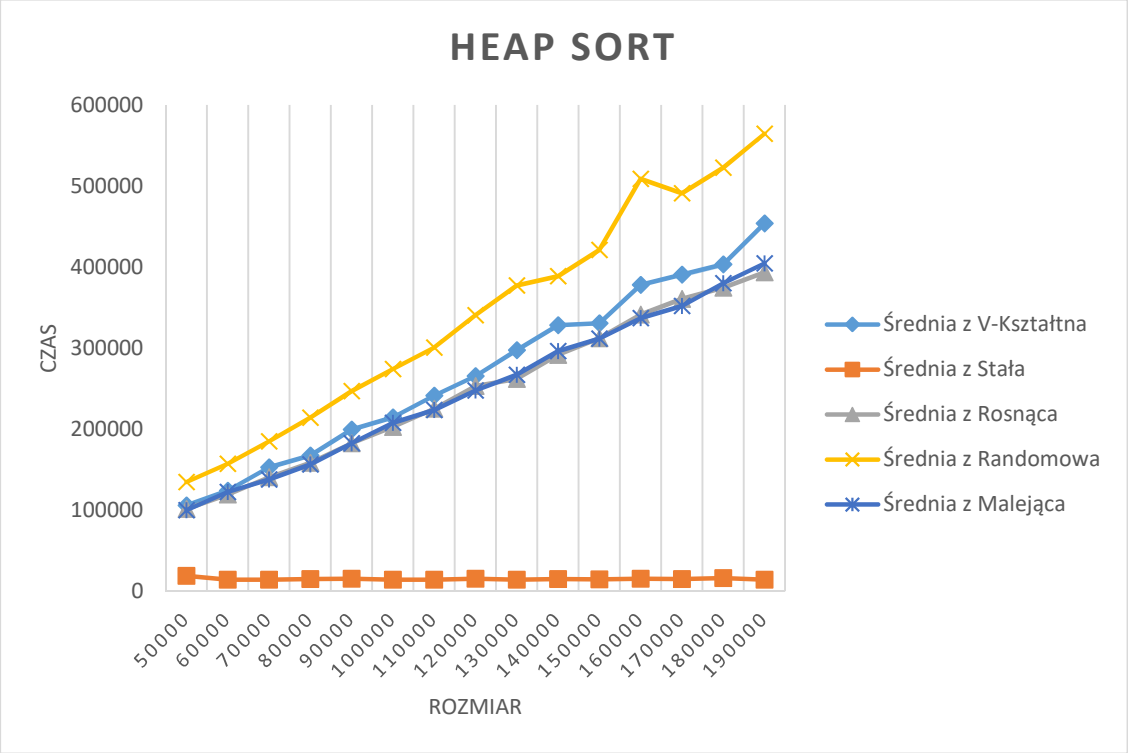


Wnioski: Najwolniejszy jest Cocktail Sort a najszybszy Heap Sort

Heap Sort

rozmiar	V-Kształtn	Malejąca	Rosnąca	Randomowa	Stała
50000	106063	100251	101718	135028	16311
60000	124117	122745	119483	157542	14455
70000	153039	137953	140807	185091	17099
80000	167801	156677	158354	214230	20198
90000	199526	182469	182464	247177	21929
100000	214699	207909	202619	274069	26009
110000	241326	223776	225107	300628	26552
120000	265574	247983	253517	340459	33992
130000	297447	267267	262108	377216	33534
140000	328187	296209	291443	388946	34465
150000	330734	311964	312343	421255	40347
160000	377933	337150	341589	508796	42296
170000	390760	351941	360616	491185	43657
180000	403426	380083	374699	522586	53975
190000	454037	404551	393653	564419	50627

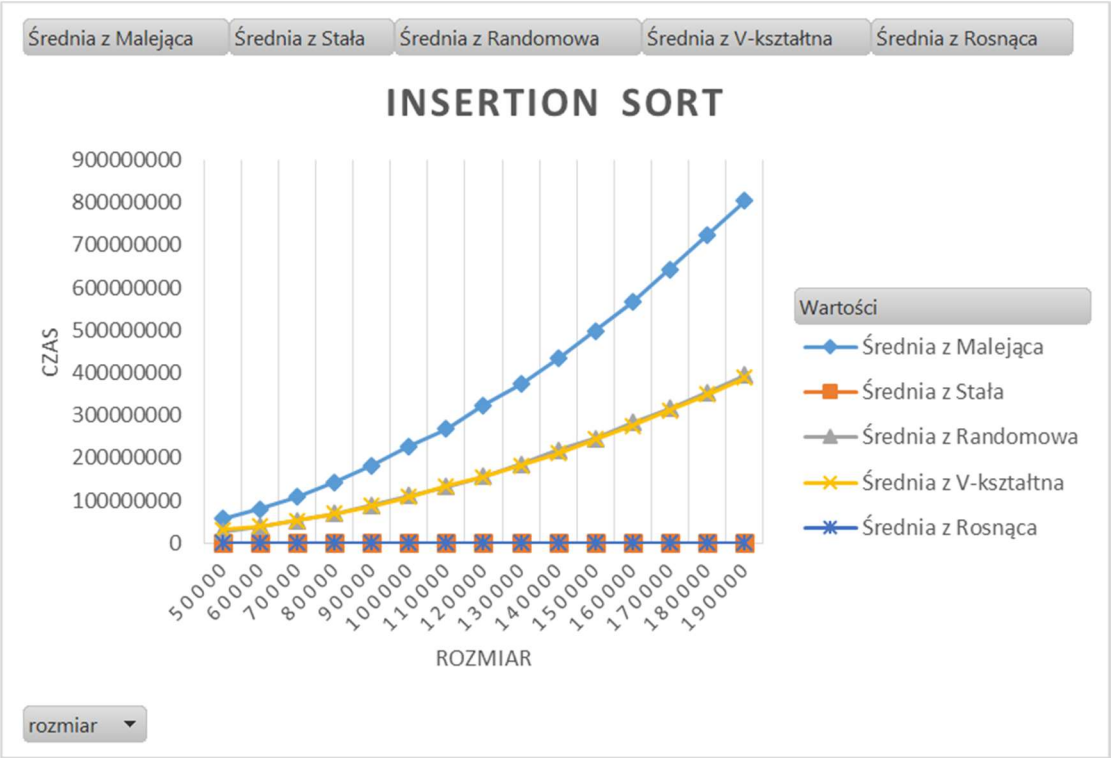
Wykresy



Wnioski: Najszybsza dla stałej tablicy najwolniejsza dla losowej

Insertion Sort

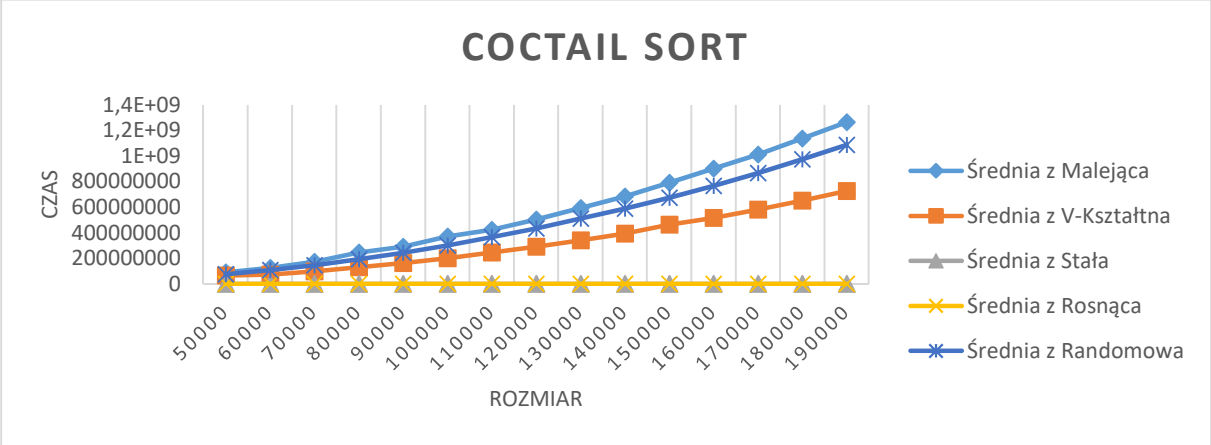
rozmiar	Malejąca	Rosnąca	Randomowa	Stała	V-kształtna
50000	56484053	5192	27188631	4946	31353728
60000	79865920	4064	39510922	3519	38800161
70000	108541095	4124	53176088	4119	52844740
80000	143530021	4890	69725390	4935	69068988
90000	182532306	5717	88405752	5498	87210903
100000	227295001	5847	110784233	6289	107811858
110000	267332372	7000	131969661	6481	133827933
120000	324296545	7243	157230300	9185	155256203
130000	373387297	7615	184696118	8207	182155654
140000	434175422	8340	218209511	8791	211539512
150000	498872894	8998	245001663	20130	244565034
160000	566060817	10135	282852572	10106	275929098
170000	643767117	9974	315838807	10777	311170704
180000	725023232	10871	353450286	10794	349068444
190000	803976995	11397	394912191	11358	388633855



Wnioski: Najwolniejsza dla malejącej tablicy najszybsza dla tablicy stałej

Cocktail Sort

Rozmiar	Malejąca	Rosnąca	Randomowa	Stała	V-Kształtna
50000	91827330	4428	76116790	4131	67131489
60000	126151612	2010	108753883	2074	72791849
70000	174497535	2421	146620874	2444	99040548
80000	246782188	2685	192862634	2767	130371777
90000	290594466	3114	244944790	3014	163984028
100000	372488263	3457	300438487	3722	202064607
110000	423961911	3689	366671204	3704	245517285
120000	503360350	4187	433570775	4474	291483230
130000	593407525	4773	510952852	4792	340711186
140000	684849428	4922	589169195	4744	395268586
150000	791490611	5378	673554737	6112	463759695
160000	901741907	6994	767915148	5959	515667437
170000	1011464935	6051	866622778	6412	582839952
180000	1138229368	6236	974484984	6978	652739508
190000	1264500185	6586	1088294267	6953	727649886

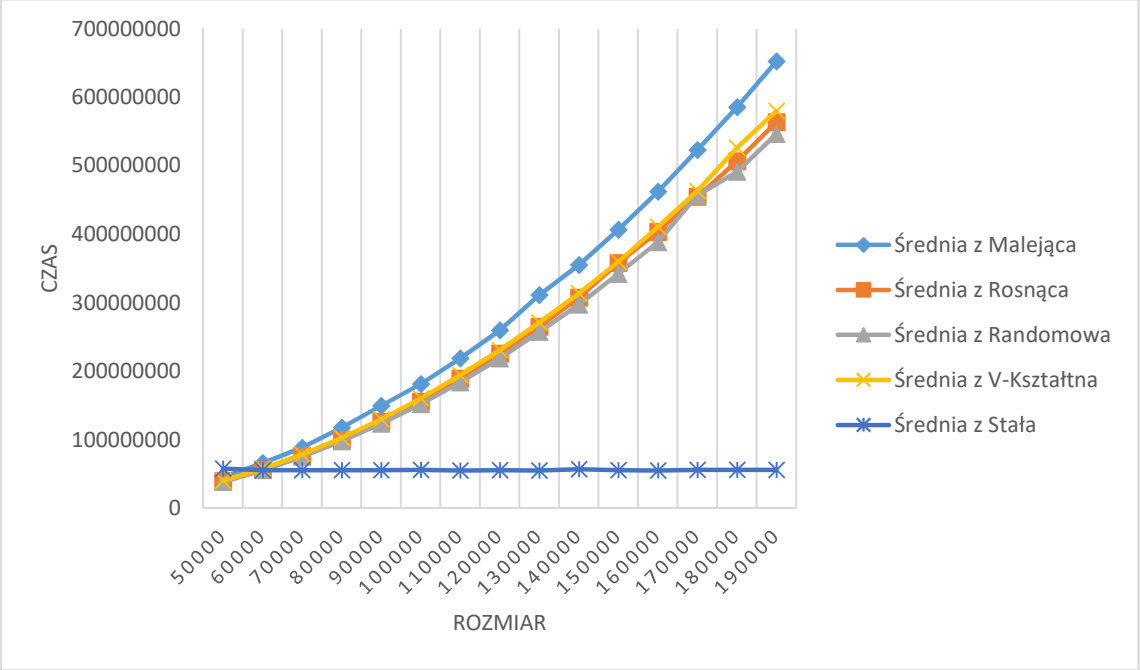


Wnioski: Najwolniejsze dla malejącej tablicy najszybsze dla stałej

Selection Sort

rozmiar	Malejąca	Rosnąca	Randomowa	Stała	V-Kształtna
50000	44966082	39527338	38044224	40983153	39994133
60000	65461182	55694924	55230283	80753275	57753099
70000	88136644	75878723	75071588	79265476	7842365
80000	117343207	99329292	97251571	150864678	10251059
90000	149407722	126054757	123071155	191961199	12953301
100000	181393120	155430906	151600134	190873359	16032603
110000	218483397	189333289	183517616	203302558	19420168
120000	259888230	225448367	218779998	293590645	23051881
130000	311249324	264719258	257272412	332756109	27086578
140000	354956548	307408120	297235515	356773039	31354263
150000	406615395	357951882	341935334	464615972	35974646
160000	462287758	403352351	388514084	453368484	41080365
170000	522846889	454686096	456729526	534613827	46334783
180000	585854158	506673280	491269041	779499859	52643702
190000	652494412	563654796	546363601	678022528	58007895

Wykres:

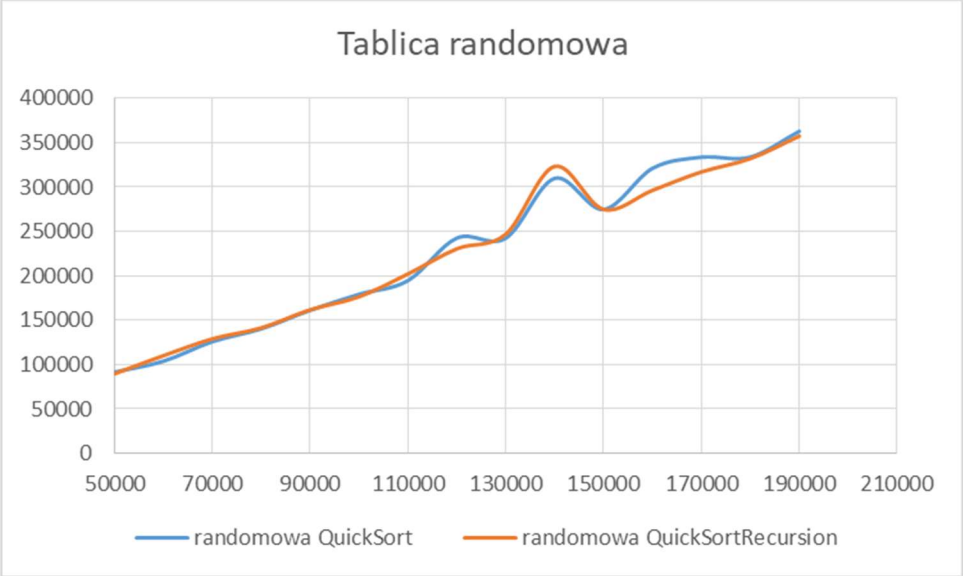


Wnioski: Najwolniejszy dla tablicy malejącej najszybszy dla tablicy stałej

Quick Sort

Tablica	randomowa	
rozmiar	QuickSort	QuickSortRecursion
50000	91511	89606
60000	103769	110019
70000	125856	128994
80000	140317	141330
90000	161191	161707
100000	179250	176401
110000	194564	202090
120000	242574	230494
130000	242520	246988
140000	309900	323321
150000	274844	274862
160000	321174	296453
170000	333676	317137
180000	333807	332080
190000	362908	357445

Wykres:



Wnioski:

Szybsza jest metoda Quick Sort z rekurencją lecz o nie wiele

Komputer na którym robiono pomiary:

Wersja systemu Windows

Windows 10 Home

System

Procesor:	Intel(R) Core(TM) i5-3210M CPU @ 2.50GHz 2.50 GHz
Zainstalowana pamięć (RAM):	8,00 GB (dostępne: 7,89 GB)
Typ systemu:	64-bitowy system operacyjny, procesor x64
Pióro i dotyk:	Brak obsługi pióra i wprowadzania dotykowego dla tego ekranu