

**Міністерство освіти і науки України**  
**Національний технічний університет України «Київський політехнічний**  
**інститут імені Ігоря Сікорського»**  
**Факультет інформатики та обчислювальної техніки**

**Кафедра ІІІ**

**Звіт**

з лабораторної роботи № 1 з дисципліни  
«Алгоритми та структури даних 2. Структури даних»

**„Проектування і аналіз алгоритмів внутрішнього сортування”**

**Виконав(ла)**

ІІІ-13 Пархомчук Ілля Вікторович  
(шифр, прізвище, ім'я, по батькові)

**Перевірив**

Сопов Олексій Олександрович  
(прізвище, ім'я, по батькові)

Київ 2022

## ЗМІСТ

<b>1</b>	<b>МЕТА ЛАБОРАТОРНОЇ РОБОТИ .....</b>	<b>3</b>
<b>2</b>	<b>ЗАВДАННЯ .....</b>	<b>4</b>
<b>3</b>	<b>ВИКОНАННЯ.....</b>	<b>5</b>
3.1	АНАЛІЗ АЛГОРИТМУ НА ВІДПОВІДНІСТЬ ВЛАСТИВОСТЯМ .....	5
3.2	ПСЕВДОКОД АЛГОРИТМУ .....	6
3.3	АНАЛІЗ ЧАСОВОЇ СКЛАДНОСТІ.....	7
3.4	ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ .....	8
3.4.1	<i>Вихідний код.....</i>	8
3.4.2	<i>Приклад роботи .....</i>	8
3.5	ТЕСТУВАННЯ АЛГОРИТМУ .....	10
3.5.1	<i>Часові характеристики оцінювання.....</i>	10
3.5.2	<i>Графіки залежності часових характеристик оцінювання від розмірності масиву .....</i>	13
	<b>ВИСНОВОК .....</b>	<b>15</b>

## 1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи – вивчити основні методи аналізу обчислювальної складності алгоритмів внутрішнього сортування і оцінити поріг їх ефективності.

## 2 ЗАВДАННЯ

Виконати аналіз алгоритму внутрішнього сортування на відповідність наступним властивостям (таблиця 2.1):

- стійкість;
- «природність» поведінки (Adaptability);
- базуються на порівняннях;
- необхідність додаткової пам'яті (об'єму);
- необхідність в знаннях про структуру даних.

Записати алгоритм внутрішнього сортування за допомогою псевдокоду (чи іншого способу по вибору).

Провести аналіз часової складності в гіршому, кращому і середньому випадках та записати часову складність в асимптотичних оцінках.

Виконати програмну реалізацію алгоритму на будь-якій мові програмування з фіксацією часових характеристик оцінювання (кількість порівнянь, кількість перестановок, глибина рекурсивного поглиблення та інше в залежності від алгоритму).

Провести ряд випробувань алгоритму на масивах різної розмірності (10, 100, 1000, 5000, 10000, 20000, 50000 елементів) і різних наборів вхідних даних (впорядкований масив, зворотно упорядкований масив, масив випадкових чисел) і побудувати графіки залежності часових характеристик оцінювання від розмірності масиву, нанести на графік асимптотичну оцінку гіршого і кращого випадків для порівняння.

Зробити порівняльний аналіз двох алгоритмів.

Зробити узагальнений висновок з лабораторної роботи.

Таблиця 2.1 – Варіанти алгоритмів

№	Алгоритм сортування
1	Сортування бульбашкою
2	Сортування гребінцем («розчіскою»)

### 3 ВИКОНАННЯ

#### 3.1 Аналіз алгоритму на відповідність властивостям

Аналіз алгоритму сортування бульбашкою на відповідність властивостям наведено в таблиці 3.1.1

Таблиця 3.1.1 – Аналіз алгоритму на відповідність властивостям

<b>Властивість</b>	<b>Сортування бульбашкою</b>
Стійкість	Так
«Природність» поведінки (Adaptability)	Так
Базуються на порівняннях	Так
Необхідність в додатковій пам'яті (об'єм)	Ні
Необхідність в знаннях про структури даних	Так(Масив)

Аналіз алгоритму сортування гребінцем на відповідність властивостям наведено в таблиці 3.1.2

Таблиця 3.1.2 – Аналіз алгоритму на відповідність властивостям

<b>Властивість</b>	<b>Сортування гребінцем</b>
Стійкість	Ні
«Природність» поведінки (Adaptability)	Так
Базуються на порівняннях	Так
Необхідність в додатковій пам'яті (об'єм)	Ні
Необхідність в знаннях про структури даних	Так(Масив)

### 3.2 Псевдокод алгоритму

#### Сортування бульбашкою

**BubbleSort(arr[], size)**

```
    для і від 0 до size - 1 , з кроком 1 повторити
        для j від 0 до size - 1, з кроком 1 повторити
            якщо  $\text{arr}[j] > \text{arr}[j + 1]$ 
                то
                     $\text{temp} := \text{arr}[j]$ 
                     $\text{arr}[j] := \text{arr}[j + 1]$ 
                     $\text{arr}[j+1] := \text{temp}$ 
            все якщо
        все повторити
    все повторити
```

#### Сортування гребінцем

**GetGap(gap)**

```
     $\text{gap} := \text{gap} / 1.3$ 
    якщо  $\text{gap} < 1$ 
        то
             $\text{gap} := 1$ 
    все якщо
    повернути gap
```

CombSort(arr[], size)

gap := size

swapped := *істина*

**повторити поки** gap > 1 **або** swapped

swapped := *хиба*

gap = GetGap(gap)

**для** i **від** 0 **до** size - gap , **з кроком** 1 **повторити**

**якщо** arr[i] > arr[i + gap]

**то**

temp := arr[i]

arr[i] := arr[i + gap]

arr[i+gap] := temp

swapped := *істина*

**все якщо**

**все повторити**

**все повторити**

### 3.3 Аналіз часової складності

Сортування бульбашкою

Випадок	Складність
Найгірший	$n^2$
Середній	$n^2$
Найкращий	$n^2$

Сортування гребінцем

Випадок	Складність
Найгірший	$n^2$
Середній	$n \log n$
Найкращий	$n \log n$

## 3.4 Програмна реалізація алгоритму

### 3.4.1 Вихідний код

#### Сортування бульбашкою

```
void BubbleSort(int* Arr, int size)
{
    int temp;
    for (int i = 0; i < size-1; i++)
    {
        for (int j = 0; j < size-1; j++)
        {
            if (Arr[j] > Arr[j + 1])
            {
                temp = Arr[j];
                Arr[j] = Arr[j + 1];
                Arr[j + 1] = temp;
            }
        }
    }
}
```

#### Сортування гребінцем

```
int GetGap(int gap)
{
    gap /= 1.3;
    if (gap < 1)
    {
        gap = 1;
    }
    return gap;
}

void CombSort(int* arr, int size)
{
    int temp, gap = size;
    bool swapped = true;
    while (gap > 1 || swapped)
    {
        swapped = false;
        gap = GetGap(gap);
        for (int i = 0; i < size - gap; ++i)
        {
            if (arr[i] > arr[i + gap])
            {
                temp = arr[i];
                arr[i] = arr[i + gap];
                arr[i + gap] = temp;
                swapped = true;
            }
        }
    }
}
```

### 3.4.2 Приклад роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми сортування масивів на 100 і 1000 елементів відповідно.



Microsoft Visual Studio Debug Console																													
Generated Array:																													
8926	3739	23879	9992	9769	23176	4951	26005	15758	15224	6764	29931	15115	29888	15509	31407	25157	24003	19640	13051										
8314	2582	15914	31143	183	15586	23165	30919	12363	11146	2457	8308	10107	21390	24590	19408	19063	26544	26537	14148										
5773	6229	22086	11276	12189	15149	9956	1270	25037	5654	18687	28774	27517	20798	23066	5022	9086	21640	7579	29500										
20350	5170	9947	17668	16071	4276	27006	3808	32042	3563	9216	14117	12853	17782	22673	27160	173	6241	18467	30897										
31288	1024	9153	5279	31577	24094	26658	3467	929	7256	6014	29601	24100	10947	17824	17229	11819	5589	10813	32322										
Sorted Array:																													
173	183	929	1024	1270	2457	2582	3467	3563	3739	3808	4276	4951	5022	5170	5279	5589	5654	5773	6014										
6229	6241	6764	7256	7579	8308	8314	8926	9086	9153	9216	9769	9947	9956	9992	10107	10813	10947	11146	11276										
11819	12189	12363	12853	13051	14117	14148	15115	15149	15224	15509	15586	15758	15914	16071	17229	17668	17782	17824	18467										
18687	19063	19408	19640	20350	20798	21390	21640	22086	22673	23066	23165	23176	23879	24003	24094	24100	24590	25037	25157										
26005	26537	26544	26658	27006	27160	27517	28774	29500	29601	29888	29931	30897	30919	31143	31288	31407	31577	32042	32322										

Рисунок 3.1 – Сортунання масиву на 100 елементів

Microsoft Visual Studio Debug Console																																						
Generated Array:																																						
9301	27390	13874	24761	16297	15998	2435	17048	14047	29092	22522	2229	5182	32171	11703	26439	29034	12984	797	31412	10377	13418	5915	18621	8357	9394	16040	25078	14673	28901	19886	12082	5544	29967	31934	30940	26098	27019	
28836	3681	8704	20176	18116	876	16602	19342	19173	30809	10527	12538	14902	22990	12301	6501	14222	13103	20458	21780	21752	722	7586	6476	22127	4434	16817	9644	4582	24167	30877	21440	423	19734	23766	17618	21837	26423	
945	8707	27084	6866	10783	8859	10608	23981	31037	19514	13196	12402	10123	9120	19204	21923	38080	14809	28593	3846	24193	22396	1450	523	18148	11333	31682	26401	19929	24734	21185	25097	4403	11	16818	821	9284		
6867	15487	24161	24370	20820	12894	14413	32253	30633	30835	6300	1206	735	13308	1400	26020	23599	30830	13851	26179	20926	10472	12121	11766	14781	1854	20995	29774	12226	22169	25067	19905	8288	9287	24555				
26373	21971	9061	8392	25962	28680	28061	26526	8526	15703	3423	29321	15926	14562	569	17109	4366	1129	9653	20134	27231	20867	19521	15255	16847	24141	20726	26079	18853	15170	3604	25422	2757	3085	1643	1048	144		
5824	30835	28378	15026	3157	21701	27510	8009	18057	18766	21851	14929	23947	30015	749	4274	25450	21981	25258	25597	11333	25686	26708	31484	15755	17145	21842	22670	31092	19109	24660	21681	577	3202	54	19425	916	7152	
24103	22689	10313	13501	13866	19287	3656	4507	5148	21596	13505	1745	29399	5204	6814	13616	21527	1723	28468	26953	36	31425	30664	3413	16399	3672	28242	16533	15144	24459	15744	27635	19087	24664	15870	2499	1337	141	
22723	24937	1045	32137	13266	5084	15128	8187	18760	20955	19254	17310	5016	9386	10418	2218	11208	6779	7640	26901	977	2229	22905	26710	16214	3255	30966	5314	21492	14761	347	21145	17809	2291	24068	9831			
27371	15257	14923	18865	8185	12371	20286	10117	20870	2331	9555	7271	16631	12048	9585	16544	26865	23794	20961	27171	17830	27099	25452	25316	29737	17213	1085	17146	7237	25866	3387	19416	28257	8151	25520	25896	14053	14756	
31406	2121	22374	1173	26265	6025	1204	28275	18917	10288	19287	14334	32545	5150	8422	13957	9269	26667	27435	21429	27368	654	29900	16303	21157	30104	5418	2397	12158	675	3978	357	16781	20789	2041	26346	1212		
22614	31075	6538	25055	13941	29059	23591	16018	3837	3560	31508	1659	31093	30502	30626	20789	6760	14453	7142	30992	32526	14546	31677	10249	13032	22067	1806	12538	6060	10338	15170	3604	25422	2757	3085	1643	1048		
9250	16742	10922	18219	657	4144	19921	12197	3955	8148	24639	1927	24712	4807	8149	29406	7278	15681	17674	16981	7437	6601	11995	30726	17615	24899	20589	3728	26032	20870	862	13930	2570	624	6355	3219	2356	1470	
27397	5330	2916	855	8635	10507	19475	22212	16198	42	20912	20075	31800	21140	3319	8729	24518	4533	21610	30285	24193	4381	14657	15374	12774	17877	6424	25550	8402	4083	486	7017	10201	17004	14460	13475	22041	23850	
15805	20433	31019	11614	23411	7894	8890	18456	13139	21585	11062	4885	1858	4885	13179	20439	26247	29146	27054	23083	14882	26418	5480	8648	21006	12734	12055	22532	22468	4257	16297	10251	18044	24629	15690	1924	26055	2841	
5847	10248	8361	7899	26099	7630	6212	9859	6099	28905	6921	1840	8982	3718	11843	2710	24047	6685	13652	23831	32645	31049	7746	6201	20634	20951	14195	20824	20331	11804	4789	15	19160	15224	5306	21534	1453	22409	
13953	17280	19392	24185	14415	30585	3044	28955	20864	23018	30461	9243	23009	25053	27629	19940	13568	28756	25934	26293	495	20642	18264	12023	30788	12687	17669	6900	4046	5732	10082	3308	111	232	434	3083	18006	2104	8714
9697	8036	29667	1278	28891	12009	4668	26745	6933	380	14598	25048	2563	14030	22779	28539	4312	10841	6317	970	8102	30195	17370	15471	28103	676	6862	6595	9085	7093	14153	19077	18133	14335	22890	2450	26625	20881	
25837	32081	1900	17338	15104	22959	6146	23089	8223	9035	22570	235	2406	7474	2307	806	617	2125	17800	21332	740	11481	31196	5041	24588	6168	6200	23986	14904	25188	13102	22424	319	4303	19	18502	16626	26805	
31891	7501	28387	17786	27567	5972	9482	18349	17252	4237	26267	28804	8317	4517	15464	32030	1264	10443	32692	5694	637	14086	16955	10307	29228	10184	30579	30554	10466	18119	11708	1734	14580	14028	4304	4976	3081	22076	
22894	28964	9488	15184	14583	15228	19311	30528	14918	20856	4099	6444	31688	13406	6474	28333	4484	29103	8721	23913	18519	26969	30429	31461	11908	17454	25226	22171	1389	11925	20801	10829	15	11336	624	2752	30213	2802	
26405	16844	25487	5329	31323	26363	28112	17890	16361	13136	15384	8820	12342	15266	21938	822	13697	9289	25702	26205	8832	19761	66	7250	31815	1069	26770	11431	7020	12685	20846	16722	1030	439	2902	2025	2704		
9045	561	1639	14231	18947	17016	9435	18844	18763	17752	918	1691	14783	19302	13864	29681	26840	17931	26272	22668	21363	20099	10536	2902	9017	2299	6145	4186	6227	23903	7085	7142	7127	7237	7240	12487	12841	14515	32165
7240	21110	1000	5159	21476	27010	30616	14305	28362	30460	2222	31023	17667	8107	30509	3035	11063	7964	5806	20424	4790	9131	13807	16864	22099	4753	4051	10644	22207	23903	7815	18116	25970	2719	4307	2434	27954	3161	
23288	5542	12718	18065	9573	26401	9017	22329	32207	4086	7863	24070	27606	29661	18768	14899	9093	15937	2583	15274	27676	15856	18153	30928	25619	10518	26776	21917	18011	23916	22709	25522	3351	28845	3047	5825	1663	9546	
24842	32487	1380	5982	24594	32061	9816	13796	151	9988	2911	23592	8708	18683	18146	4809	24243	17973	24084	9515	23043	14889	12285	5298	14834	7754	19054	31278	20315	24508	1774	15157	12861	7085	8177	5866	21681	301	
31420	14212	20819	15783	3284	24199	30951	21673	29000	7385	25411	3205	12976	21512	1897	28222	8518	21061	7923	8215	5267	1809	22708	23977	16081	23520	7836	13874	21376	1298	29484	1328	20875	24356	13535	22891	15084	1304	
31248	15583	31587	17778	18017	5737	7491	5525	28916	4355	23741	28302																											
Sorted Array:																																						
42	54	66	151	152	156	199	235	323	347	380	495	561	569	611	617	636	654	657	675	676	722	735	749	749	797	806	822	855	876	918	945	970	977	1002	1069	1085	1129	
1173	1203	1204	1206	1264	1278	1282	1298	1328	1357	1380	1389	1460	1480	1619	1659	1691	1697	1723	1745	1806	1809	1840	1845	1854	1858	1872	1874	1897	1918	1987	2082	2112	2125	2229	2291	2299	2307	
2318	2319	2333	2334	2356	2387	2406	2435	2563	2583	2710	2847	2982	2991	3141	3216	3046	3197	3202	3205	3255	3284	3319	3387	3423	3560	3680	3656	3672	3681	3718	3728	3835	3836	3864	3866	3955	3978	4007
4046	4057	4060	4061	4062	4063	4064	4065	4066	4067	4068	4069	4070	4071	4072	4073	4074	4075	4076	4077	4078	4079	4080	4081	4082	4083	4084	4085	4086	4087	4088	4089	4090	4091	4092	4093	4094	4095	
5041	5084	5148	5150	5159	5182	5204	5267	5298	5314	5320	5338	5341	5418	5480	5525	5542	5634	5694	5732	5737	5806	5824	5825	5847	5951	5972	5982	6060	6099	6145	6146	6168	6201	6212	6214	6276	6278	
6275	6280	6308	6311	6337	6355	6424	6444	6474	6476	6501	6530	6538	6544	6595	6601	6614	6685	6747	6779	6814	6862	6866	6867	6921	6933	7017	7020	7085	7093	7142	7157	7237	7240	7260	7261	7317	7385	
7437	7474	7491	7503	7586	7630	7646	7646	7646	7754	7836	7863	7894	7899	7923	7964	8009	8036	8059	8102	8148	8149	8151	8177	8185	8187	8187	8218	8218	8223	8241	8308	8317	8317	8357	8361	8392	8396	
8422	8434	8462	8518	8526	8535	8638	8692	8704	8707	8708	8721	8729	8781	8820	8832	8859	8860	8982	9017	9035	9045	9085	9093	9120														

### 3.5 Тестування алгоритму

#### 3.5.1 Часові характеристики оцінювання

В таблиці 3.2.1 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування бульбашки для масивів різної розмірності, коли масив містить упорядковану послідовність елементів.

Таблиця 3.2.1 – Характеристики оцінювання алгоритму сортування бульбашки для упорядкованої послідовності елементів у масиві

Розмірність масиву	Число порівнянь	Число перестановок
10	81	0
100	9801	0
1000	998001	0
5000	24990001	0
10000	99980001	0
20000	399960001	0
50000	2499900001	0

В таблиці 3.2.2 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування гребінцем для масивів різної розмірності, коли масив містить упорядковану послідовність елементів.

Таблиця 3.2.2 – Характеристики оцінювання алгоритму сортування гребінцем для упорядкованої послідовності елементів у масиві

Розмірність масиву	Число порівнянь	Число перестановок
10	32	0
100	1003	0
1000	18713	0
5000	123386	0
10000	276739	0
20000	613402	0
50000	1683412	0

В таблиці 3.3.1 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування бульбашки для масивів різної розмірності, коли масиви містять зворотно упорядковану послідовність елементів.

Таблиця 3.3.1 – Характеристики оцінювання алгоритму сортування бульбашки для зворотно упорядкованої послідовності елементів у масиві.

Розмірність масиву	Число порівнянь	Число перестановок
10	81	45
100	9801	4950
1000	998001	499500
5000	24990001	12497500
10000	99980001	49995000
20000	399960001	199990000
50000	2499900001	1249975000

В таблиці 3.3.2 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування гребінцем для масивів різної розмірності, коли масиви містять зворотно упорядковану послідовність елементів.

Таблиця 3.3.2 – Характеристики оцінювання алгоритму сортування гребінцем для зворотно упорядкованої послідовності елементів у масиві.

Розмірність масиву	Число порівнянь	Число перестановок
10	41	7
100	1102	122
1000	19712	1582
5000	128385	9572
10000	286738	20078
20000	633401	42634
50000	1733411	116838

У таблиці 3.4.1 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування бульбашки для масивів різної розмірності, масиви містять випадкову послідовність елементів.

Таблиця 3.4.1 – Характеристика оцінювання алгоритму сортування бульбашки для випадкової послідовності елементів у масиві.

Розмірність масиву	Число порівнянь	Число перестановок
10	81	14
100	9801	2401
1000	998001	249351
5000	24990001	6344409
10000	99980001	25124386
20000	399960001	100130948
50000	2499900001	624639797

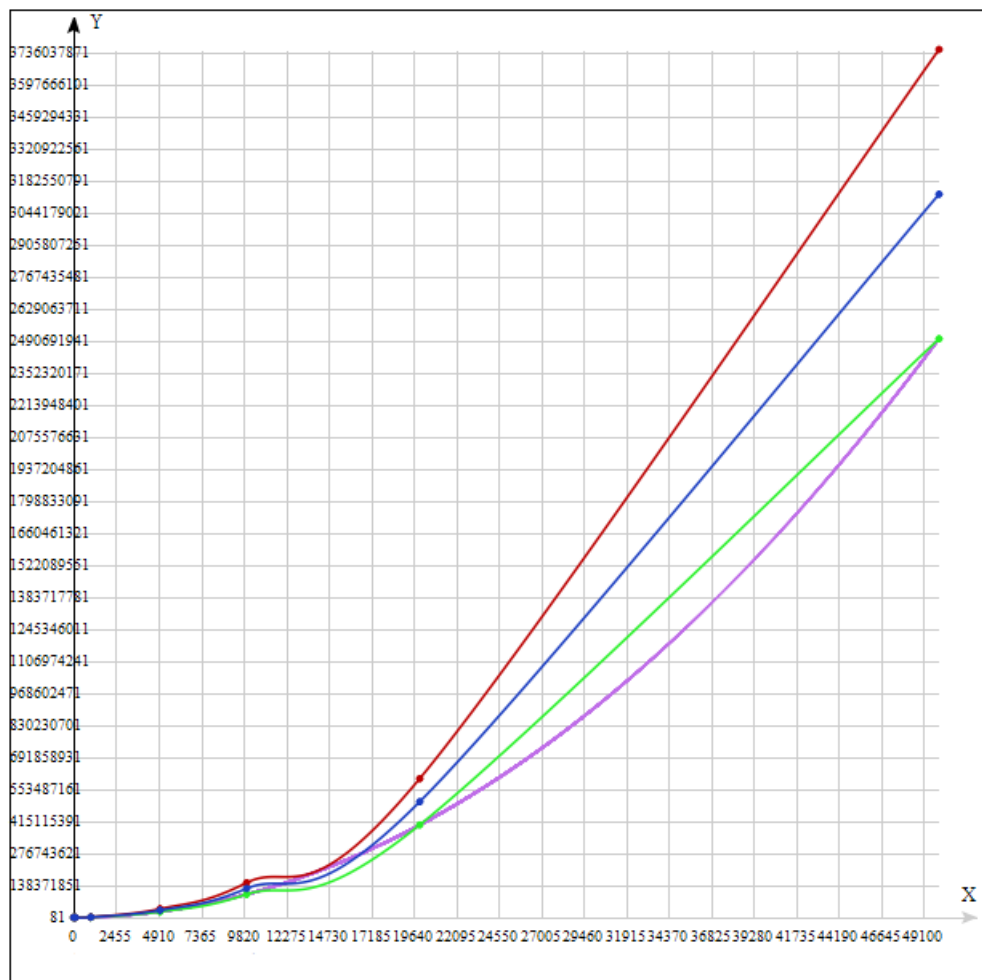
У таблиці 3.4.2 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування гребінцем для масивів різної розмірності, масиви містять випадкову послідовність елементів.

Таблиця 3.4.2 – Характеристика оцінювання алгоритму сортування гребінцем для випадкової послідовності елементів у масиві.

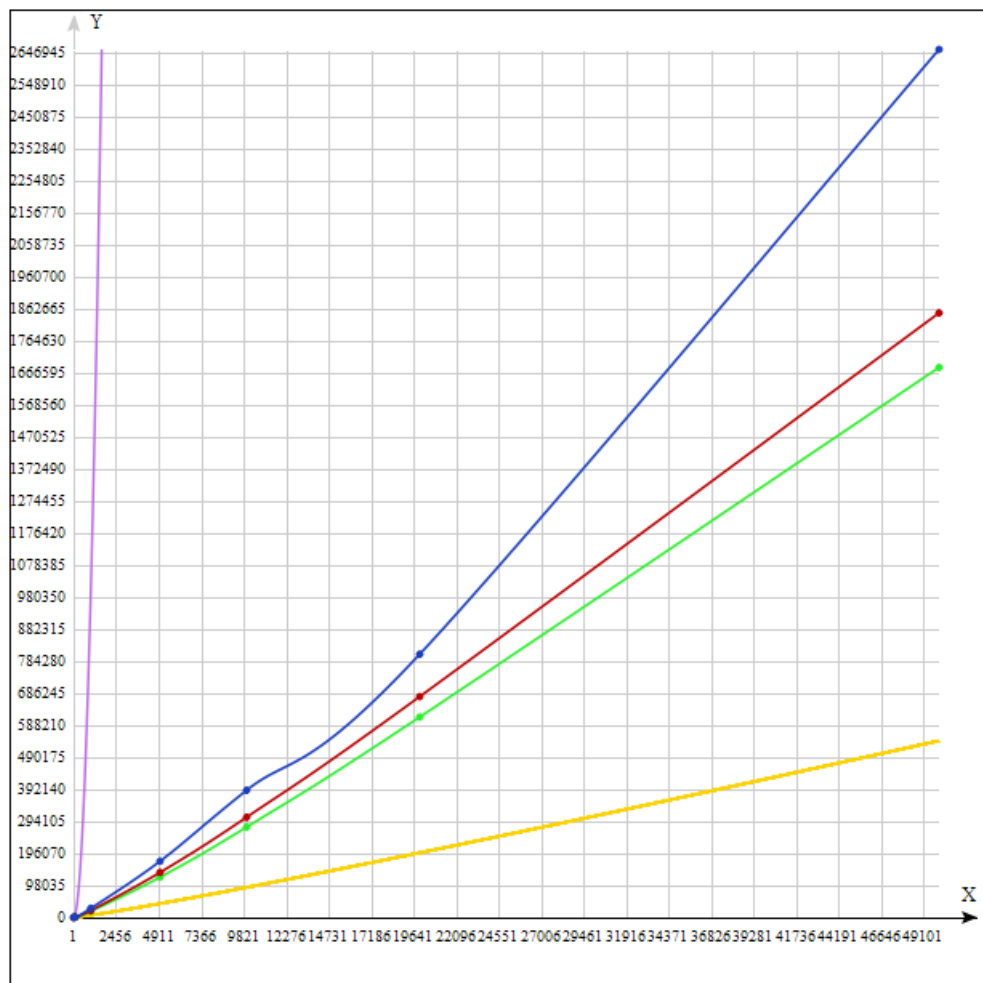
Розмірність масиву	Число порівнянь	Число перестановок
10	41	6
100	1300	267
1000	23708	4322
5000	143382	28741
10000	326734	62564
20000	673399	132277
50000	2283400	373305

### 3.5.2 Графіки залежності часових характеристик оцінювання від розмірності масиву

На рисунку 3.3 показані графіки залежності часових характеристик оцінювання від розмірності масиву для випадків, коли масиви містять упорядковану послідовність елементів (зелений графік), коли масиви містять зворотно упорядковану послідовність елементів (червоний графік), коли масиви містять випадкову послідовність елементів (синій графік), також показані асимптотичні оцінки гіршого (фіолетовий графік) і кращого (жовтий графік) випадків для порівняння.



Сортування бульбашкою



Сортування гребінцем

Рисунок 3.3 – Графіки залежності часових характеристик оцінювання

## ВИСНОВОК

При виконанні даної лабораторної роботи я ознайомився з нотацією великого  $O$ . Навчився оцінювати швидкодію алгоритмів бульбашки та гребінцем. Мною був написаний програмний код та проведено тестування алгоритмів з фіксуванням кількості порівнянь та кількості перестановок. На основі отриманих даних було побудовано графіки із залежністю кількості операцій від кількості елементів.

Я помітив, що сортування бульбашкою у всіх випадках веде себе майже однаково й очікувано. Однак, сортування гребінцем на прямо та зворотно упорядкованій послідовності елементів веде себе краще, чим на випадковій послідовності. Це пояснюється основою ідеєю алгоритму, щоб великі елементи зразу ставити ближче до кінця, а маленькі – ближче до початку.