

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

Кафедра інформатики та програмної інженерії

Звіт

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

«Проектування та аналіз алгоритмів внутрішнього сортування»

Варіант 5

Виконав студент

ІП-15, Буяло Дмитро Олександрович  
(шифр, прізвище, ім'я, по батькові)

Перевірів

Соколовський Владислав Володимирович  
(прізвище, ім'я, по батькові)

Київ 2022

## **Лабораторна робота 1**

### **Проектування та аналіз алгоритмів внутрішнього сортування**

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

#### **Індивідуальне завдання**

#### **Варіант 5**

#### **Завдання**

На прикладі алгоритмів сортування бульбашкою та гребінцем, виконати аналіз алгоритму внутрішнього сортування на відповідність властивостям стійкості<sup>1</sup>, природності поведінки<sup>2</sup>, необхідності додаткової пам'яті, необхідності в знаннях про структуру даних та визначити чи базується на порівняннях.

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

Провести ряд випробувань алгоритму на масивах різної розмірності 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.

<sup>1</sup> Стійкість – здатність не виконувати обмін однакових елементів.

<sup>2</sup> Природність – здатність ефективно обробляти вже відсортовані чи частково відсортовані масиви. Повністю природний алгоритм на відсортованому масиві робить  $n$  порівнянь та жодного обміну.

### Виконання

#### 1.1 Аналіз алгоритму сортування бульбашкою на відповідність властивостям

| Властивість                                | Сортування бульбашкою  |
|--|--|
| Стійкість                                  | Алгоритм є стійким   |
| «Природність» поведінки (Adaptability)     | Алгоритм є природним   |
| Базуються на порівняннях                   | Алгоритм базується на порівняннях                              |
| Необхідність в додатковій пам'яті (об'єм)  | Додаткової пам'яті алгоритм не потребує                        |
| Необхідність в знаннях про структури даних | Необхідні знання про базові та лінійні структури даних (масив) |

#### 1.2 Псевдокод алгоритму

##### Початок

isSorted = false; k = -1;

##### Повторити

##### Поки !isSorted

isSorted = true; k++

##### Повторити для i від lengthOfArr-1 до k

Якщо arr[i-1] > arr[i]

то

isSorted = false

buf = arr[i-1]

arr[i-1] = arr[i]

arr[i] = buf

##### Все якщо

##### Все повторити

##### Все повторити

##### Кінець

### 1.3 Аналіз часової складності

Найкращий випадок:  $O(n)$

Найгірший випадок:  $O(n^2)$

Середній випадок:  $O(n^2)$

### 1.4 Програмна реалізація алгоритму; Вихідний код

```
Main.java x
5  public class Main {
6  public static void main(String[] args) {
7      Scanner in = new Scanner(System.in);
8      int n = in.nextInt();
9      int[] arr = new int[n]; // кол-во елементів в масиві
10     for(int i = 0; i < n; i++) {
11         arr[i] = (int)(Math.random() * (double)(n + 1)); // in.nextInt(); arr[i] = i; //
12     }
13     int buf, comparison = 0, swap = 0, k = -1;
14     boolean isSorted = false; // перевірка на відсортованість, залог естественности
15     while(!isSorted) { // ідеальний бульбурек лічної розробки
16         isSorted = true;
17         k++; // самодельний счєтчик
18         for (int i = arr.length - 1; i > k; i--) {
19             comparison++; // кол-во сравнієніє
20             if (arr[i - 1] > arr[i]) {
21                 isSorted = false; // єсли без ієпользованиє третєєї переменной:
22                 buf = arr[i - 1]; // arr[j] += arr[j-1]
23                 arr[i - 1] = arr[i]; // arr[j-1] = arr[j] - arr[j-1]
24                 arr[i] = buf; // arr[j] -= arr[j-1]
25                 swap++; // кол-во перєстановок
26             }
27         }
28     }
29     System.out.println("\n\nComparisons = " + comparison + "; Swaps = " + swap);
30 }
31 }
```

### 1.5 Приклад роботи програми сортування масиву на 100 елементів

```
100
Random array:
[8, 54, 18, 84, 45, 4, 7, 31, 31, 24, 73, 88, 66, 90, 75, 44, 66, 61, 3, 64, 34, 66, 35, 12, 42, 68, 54, 95,
99, 82, 48, 3, 91, 30, 49, 46, 94, 61, 48, 97, 48, 79, 77, 59, 47, 79, 97, 55, 28, 19, 59, 79, 84, 8, 68,
88, 11, 10, 3, 29, 71, 70, 66, 66, 20, 38, 58, 69, 88, 18, 30, 17, 67, 36, 8, 96, 31, 45, 55, 87, 87, 38,
11, 89, 72, 78, 43, 57, 8, 63, 37, 97, 20, 6, 78, 64, 17, 68, 59, 5]

Sorted array:
[3, 3, 3, 4, 5, 6, 7, 8, 8, 8, 8, 10, 11, 11, 12, 17, 17, 18, 18, 19, 20, 20, 24, 28, 29, 30, 30, 31,
31, 31, 34, 35, 36, 37, 38, 38, 42, 43, 44, 45, 45, 46, 47, 48, 48, 48, 49, 54, 54, 55, 55, 57, 58, 59, 59,
59, 61, 61, 63, 64, 64, 66, 66, 66, 66, 66, 67, 68, 68, 68, 69, 70, 71, 72, 73, 75, 77, 78, 78, 79, 79, 79,
82, 84, 84, 87, 87, 88, 88, 88, 89, 90, 91, 94, 95, 96, 97, 97, 97, 99]

Comparisons = 4779; Swaps = 2454
```

### 1.6 Приклад роботи програми сортування масиву на 1000 елементів

```
1000
Random array:
[63, 468, 663, 476, 37, 417, 423, 39, 414, 559, 459, 271, 520, 298, 15, 250, 762, 831, 766, 312, 463, 646, 59, 852, 359, 898, 237, 833,
438, 567, 895, 741, 77, 225, 132, 308, 881, 747, 893, 935, 400, 229, 220, 346, 957, 970, 887, 737, 135, 364, 113, 284, 111, 194, 315,
706, 855, 892, 639, 246, 475, 3, 788, 295, 356, 276, 98, 654, 527, 489, 55, 305, 693, 391, 505, 459, 673, 896, 602, 516, 541, 982,
147, 588, 665, 577, 336, 139, 944, 83, 88, 366, 997, 100, 101, 955, 726, 318, 431, 908, 712, 545, 977, 280, 103, 281, 458, 746, 270,
328, 594, 128, 278, 439, 996, 188, 375, 111, 940, 208, 52, 257, 585, 172, 946, 615, 371, 622, 750, 462, 335, 595, 621, 394, 811, 405,
974, 669, 518, 405, 470, 916, 350, 288, 991, 783, 291, 936, 808, 466, 728, 290, 776, 888, 121, 240, 291, 747, 411, 51, 2, 548, 761,
442, 687, 194, 248, 8, 517, 735, 885, 337, 988, 354, 920, 806, 64, 788, 722, 769, 780, 873, 656, 566, 339, 138, 290, 518, 24, 921,
947, 905, 596, 708, 76, 715, 650, 123, 213, 809, 401, 189, 255, 36, 647, 302, 672, 762, 281, 645, 811, 171, 535, 602, 852, 755, 314,
115, 135, 330, 395, 814, 493, 38, 951, 178, 973, 790, 867, 974, 808, 772, 589, 658, 796, 731, 752, 531, 307, 931, 709, 905, 963, 873,
174, 681, 692, 101, 577, 366, 253, 657, 414, 645, 499, 58, 824, 367, 639, 988, 857, 998, 509, 229, 423, 299, 373, 777, 701, 336, 112,
449, 862, 658, 751, 804, 341, 863, 117, 611, 521, 633, 147, 642, 586, 701, 720, 116, 583, 163, 170, 652, 117, 5, 619, 232, 965, 686,
27, 298, 858, 193, 474, 744, 90, 984, 459, 302, 150, 848, 209, 532, 519, 850, 831, 363, 912, 121, 861, 957, 728, 502, 236, 475, 174,
703, 954, 162, 577, 930, 707, 79, 669, 664, 633, 702, 237, 626, 872, 563, 364, 750, 9, 97, 635, 397, 677, 846, 125, 932, 451, 215,
668, 177, 994, 82, 888, 293, 795, 595, 764, 54, 236, 707, 29, 557, 696, 95, 536, 946, 311, 345, 831, 819, 590, 502, 587, 603, 522,
333, 190, 323, 989, 968, 745, 907, 254, 857, 10, 318, 86, 408, 260, 627, 552, 960, 908, 330, 893, 251, 756, 96, 931, 929, 158, 412,
374, 145, 959, 489, 22, 803, 41, 187, 314, 524, 508, 645, 899, 21, 990, 752, 876, 141, 364, 629, 177, 452, 203, 753, 86, 978, 605,
472, 253, 311, 648, 209, 712, 112, 659, 778, 248, 730, 579, 896, 250, 481, 238, 144, 539, 967, 27, 348, 16, 610, 524, 584, 949, 525,
878, 808, 529, 733, 380, 390, 963, 668, 476, 685, 694, 911, 637, 140, 546, 103, 912, 335, 485, 17, 721, 774, 438, 829, 381, 233, 109,
759, 178, 935, 994, 386, 309, 855, 876, 53, 865, 33, 940, 54, 124, 830, 231, 453, 842, 698, 633, 700, 848, 186, 702, 444, 274, 577,
672, 318, 924, 753, 703, 781, 843, 669, 960, 505, 923, 32, 439, 222, 477, 636, 559, 360, 55, 117, 370, 300, 437, 772, 955, 592, 642,
460, 970, 998, 19, 587, 546, 806, 948, 564, 736, 471, 443, 193, 488, 251, 50, 917, 699, 192, 675, 924, 222, 513, 654, 725, 867, 135,
260, 886, 284, 918, 985, 276, 369, 891, 409, 394, 257, 641, 217, 983, 236, 840, 905, 462, 950, 787, 53, 429, 61, 763, 598, 800, 153,
934, 395, 4, 7, 509, 535, 16, 578, 900, 248, 899, 885, 611, 998, 953, 814, 738, 280, 574, 265, 971, 992, 349, 412, 208, 688, 996,
869, 997, 972, 288, 434, 236, 286, 198, 680, 820, 834, 95, 75, 107, 980, 12, 532, 470, 467, 649, 197, 405, 20, 3, 117, 136, 388,
279, 874, 941, 724, 949, 656, 899, 254, 717, 298, 190, 538, 416, 693, 550, 290, 759, 654, 979, 640, 265, 743, 394, 906, 613, 778, 419,
463, 127, 762, 150, 41, 100, 740, 373, 817, 189, 663, 334, 40, 158, 670, 917, 503, 847, 352, 751, 59, 347, 428, 580, 603, 38, 1,
982, 323, 284, 979, 360, 831, 336, 542, 988, 744, 210, 551, 757, 619, 92, 588, 205, 706, 119, 953, 345, 852, 232, 275, 781, 31, 159,
607, 630, 396, 489, 726, 795, 435, 221, 193, 185, 849, 153, 121, 635, 973, 934, 958, 342, 153, 665, 603, 872, 307, 479, 68, 921, 3,
589, 471, 286, 793, 953, 592, 761, 191, 825, 404, 908, 681, 169, 949, 61, 302, 78, 680, 207, 976, 146, 952, 773, 429, 951, 607, 824,
736, 675, 754, 551, 811, 173, 651, 442, 837, 691, 331, 37, 553, 841, 975, 707, 500, 368, 321, 444, 841, 681, 358, 307, 109, 152, 338,
726, 311, 691, 737, 605, 112, 144, 91, 819, 484, 703, 810, 430, 28, 549, 924, 723, 425, 701, 463, 937, 858, 123, 602, 404, 34, 946,
958, 636, 424, 812, 472, 805, 637, 228, 127, 987, 439, 11, 307, 431, 779, 828, 41, 182, 314, 164, 469, 295, 380, 968, 864, 0, 17,
614, 209, 209, 678, 866, 882, 849, 502, 912, 226, 682, 586, 668, 917, 388, 496, 24, 807, 923, 419, 849, 610, 361, 683, 134, 173, 716,
109, 888, 277, 505, 566, 958, 858, 151, 348, 292, 337, 673, 545, 797, 223, 484, 296, 280, 511, 552, 996, 750, 465, 613, 786, 687, 274,
595, 382, 768, 697, 944, 166, 921, 287, 58, 688, 522, 970, 628, 211, 22, 778, 263, 708, 244, 315, 442, 735, 160, 85, 467, 977, 1,
168, 808, 841, 105, 857, 966, 374, 90, 762, 702, 959, 525, 603, 319, 691, 495, 343, 468, 72, 151, 601, 812, 691, 588, 307, 779, 814,
138, 350, 537, 458, 537, 605, 256, 313, 176, 67, 744, 53, 90, 1, 497, 996, 188, 937, 297, 778, 688, 62, 514, 689, 660, 94, 410]
```

## Основи програмування – 2. Алгоритми та структури даних

Sorted array:

```
[0, 1, 1, 1, 2, 3, 3, 3, 4, 5, 7, 8, 9, 10, 11, 12, 15, 16, 16, 17, 17, 19, 20, 21, 22, 22, 24, 24,
27, 27, 28, 29, 31, 32, 33, 34, 36, 37, 37, 38, 38, 39, 40, 41, 41, 41, 50, 51, 52, 53, 53, 53, 54, 54, 55,
55, 58, 58, 59, 59, 61, 61, 62, 63, 64, 67, 68, 72, 75, 76, 77, 78, 79, 82, 83, 85, 86, 86, 88, 90, 90, 90,
91, 92, 94, 95, 95, 96, 97, 98, 100, 100, 101, 101, 103, 103, 105, 107, 109, 109, 109, 109, 111, 111, 112, 112, 112, 113, 115, 116,
117, 117, 117, 117, 119, 121, 121, 121, 123, 123, 124, 125, 127, 127, 128, 132, 134, 135, 135, 135, 136, 138, 138, 139, 140, 141, 144,
144, 145, 146, 147, 147, 150, 150, 151, 151, 152, 153, 153, 153, 158, 158, 159, 160, 162, 163, 164, 166, 168, 169, 170, 171, 172, 173,
173, 174, 174, 176, 177, 177, 178, 178, 182, 185, 186, 187, 188, 188, 189, 189, 190, 190, 191, 192, 193, 193, 193, 194, 194, 197, 198,
203, 205, 207, 208, 208, 209, 209, 209, 209, 210, 211, 213, 215, 217, 220, 221, 222, 222, 223, 225, 226, 228, 229, 229, 231, 232, 232,
233, 236, 236, 236, 236, 237, 237, 238, 240, 244, 246, 248, 248, 248, 250, 250, 251, 251, 253, 253, 254, 254, 255, 256, 257, 257, 260,
260, 263, 265, 265, 270, 271, 274, 274, 275, 276, 276, 277, 278, 279, 280, 280, 280, 281, 281, 284, 284, 284, 286, 286, 287, 288, 288,
290, 290, 290, 291, 291, 292, 293, 295, 295, 296, 297, 298, 298, 298, 299, 300, 302, 302, 302, 302, 305, 307, 307, 307, 307, 308, 309,
311, 311, 311, 312, 313, 314, 314, 314, 315, 315, 318, 318, 318, 319, 321, 323, 323, 328, 330, 330, 331, 333, 334, 335, 335, 336, 336,
336, 337, 337, 338, 339, 341, 342, 343, 345, 345, 346, 347, 348, 348, 349, 350, 350, 352, 354, 356, 358, 359, 360, 360, 361, 363, 364,
364, 364, 366, 366, 367, 368, 369, 370, 371, 373, 373, 374, 374, 375, 380, 380, 381, 382, 386, 388, 388, 390, 391, 394, 394, 394, 395,
395, 396, 397, 400, 401, 404, 404, 405, 405, 405, 408, 409, 410, 411, 412, 412, 414, 414, 416, 417, 419, 419, 423, 423, 424, 425, 428,
429, 429, 430, 431, 431, 434, 435, 437, 438, 438, 439, 439, 439, 442, 442, 442, 443, 444, 444, 449, 451, 452, 453, 458, 458, 459, 459,
459, 460, 462, 462, 463, 463, 465, 466, 467, 467, 468, 468, 469, 470, 470, 471, 471, 472, 472, 474, 475, 475, 476, 476, 477, 479,
481, 484, 484, 485, 488, 489, 489, 489, 493, 495, 496, 497, 499, 500, 502, 502, 502, 503, 505, 505, 505, 505, 509, 509, 511, 513, 514,
516, 517, 518, 518, 519, 520, 521, 522, 522, 524, 524, 525, 525, 527, 529, 531, 532, 532, 535, 535, 536, 537, 537, 538, 539, 541, 542,
545, 545, 546, 546, 548, 549, 550, 551, 551, 552, 552, 553, 557, 559, 559, 563, 564, 566, 566, 567, 574, 577, 577, 577, 577, 578, 579,
580, 583, 584, 585, 586, 586, 587, 587, 588, 588, 588, 589, 589, 590, 592, 592, 594, 595, 595, 595, 596, 598, 601, 602, 602, 602, 603,
603, 603, 603, 605, 605, 605, 607, 607, 610, 610, 611, 611, 613, 613, 614, 615, 619, 619, 621, 622, 626, 627, 628, 629, 630, 633, 633,
633, 635, 635, 636, 636, 637, 637, 639, 639, 640, 641, 642, 642, 645, 645, 645, 646, 647, 648, 649, 650, 651, 652, 654, 654, 654, 656,
656, 657, 658, 658, 659, 660, 663, 663, 664, 665, 665, 668, 668, 668, 669, 669, 669, 670, 672, 672, 673, 673, 675, 675, 677, 678, 680,
680, 681, 681, 681, 682, 683, 685, 686, 687, 687, 688, 688, 688, 689, 691, 691, 691, 691, 692, 693, 693, 694, 696, 697, 698, 699, 700,
701, 701, 701, 702, 702, 702, 703, 703, 703, 706, 706, 707, 707, 707, 708, 708, 709, 712, 712, 715, 716, 717, 720, 721, 722, 723, 724,
725, 726, 726, 726, 728, 728, 730, 731, 733, 735, 735, 736, 736, 737, 737, 738, 740, 741, 743, 744, 744, 744, 745, 746, 747, 747, 750,
750, 750, 751, 751, 752, 752, 753, 753, 754, 755, 756, 757, 759, 759, 761, 761, 762, 762, 762, 762, 763, 764, 766, 768, 769, 772, 772,
773, 774, 776, 777, 778, 778, 778, 778, 779, 779, 780, 781, 781, 783, 786, 787, 788, 788, 790, 793, 795, 795, 796, 797, 800, 803, 804,
805, 806, 806, 807, 808, 808, 808, 808, 809, 810, 811, 811, 811, 812, 812, 814, 814, 814, 817, 819, 819, 820, 824, 824, 825, 828, 829,
830, 831, 831, 831, 831, 833, 834, 837, 840, 841, 841, 841, 842, 843, 846, 847, 848, 848, 849, 849, 849, 850, 852, 852, 852, 855, 855,
857, 857, 857, 858, 858, 858, 861, 862, 863, 864, 865, 866, 867, 867, 869, 872, 872, 873, 873, 874, 876, 876, 878, 881, 882, 885, 885,
886, 887, 888, 888, 888, 891, 892, 893, 893, 895, 896, 896, 898, 899, 899, 899, 900, 905, 905, 905, 906, 907, 908, 908, 908, 911, 912,
912, 912, 916, 917, 917, 917, 918, 920, 921, 921, 921, 923, 923, 924, 924, 924, 929, 930, 931, 931, 932, 934, 934, 935, 935, 936, 937,
937, 940, 940, 941, 944, 944, 946, 946, 946, 947, 948, 949, 949, 949, 950, 951, 951, 952, 953, 953, 953, 954, 955, 955, 957, 957, 958,
958, 958, 959, 959, 960, 960, 963, 963, 965, 966, 967, 968, 968, 970, 970, 970, 971, 972, 973, 973, 974, 974, 975, 976, 977, 977, 978,
979, 979, 980, 982, 982, 983, 984, 985, 987, 988, 988, 988, 989, 990, 991, 992, 994, 994, 996, 996, 996, 996, 997, 997, 998, 998, 998]
```

Comparisons = 495930; Swaps = 248902

### 1.7 Часові характеристики оцінювання

В таблиці 1.7.1 наведені характеристики оцінювання алгоритму сортування бульбашки для впорядкованої послідовності елементів у масиві.

Таблиця 1.7.1

| Розмірність масиву | Число порівнянь | Число перестановок |
|--------------------|-----------------|--------------------|
| 10                 | 9               | 0                  |
| 100                | 99              | 0                  |
| 1000               | 999             | 0                  |
| 5000               | 4999            | 0                  |
| 10000              | 9999            | 0                  |
| 20000              | 19999           | 0                  |
| 50000              | 49999           | 0                  |

В таблиці 1.7.2 наведені характеристики оцінювання алгоритму сортування бульбашки для зворотно впорядкованої послідовності елементів у масиві.

Таблиця 1.7.2

| Розмірність масиву | Число порівнянь | Число перестановок |
|--------------------|-----------------|--------------------|
| 10                 | 45              | 45                 |
| 100                | 4950            | 4950               |
| 1000               | 499500          | 499500             |
| 5000               | 12497500        | 12497500           |
| 10000              | 49995000        | 49995000           |
| 20000              | 199990000       | 199990000          |
| 50000              | 1249975000      | 1249975000         |



В таблиці 1.7.3 наведені характеристики оцінювання алгоритму сортування бульбашки для випадкової послідовності елементів у масиві.

Таблиця 1.7.3

| Розмірність масиву | Число порівнянь | Число перестановок |
|--------------------|-----------------|--------------------|
| 10                 | 45              | 21                 |
| 100                | 4872            | 2294               |
| 1000               | 499122          | 257972             |
| 5000               | 12477199        | 6173360            |
| 10000              | 49982120        | 24789372           |
| 20000              | 199983559       | 101022631          |
| 50000              | 1249898364      | 621786381          |

## 1.8 Графіки залежності часових характеристик оцінювання від розмірності масиву

На рисунку 1.8.1 та 1.8.2 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 на рисунках 1.8.3 та 1.8.4 показані асимптотичні оцінки гіршого (фіолетовий графік) і кращого (жовтий графік) випадків для порівняння.

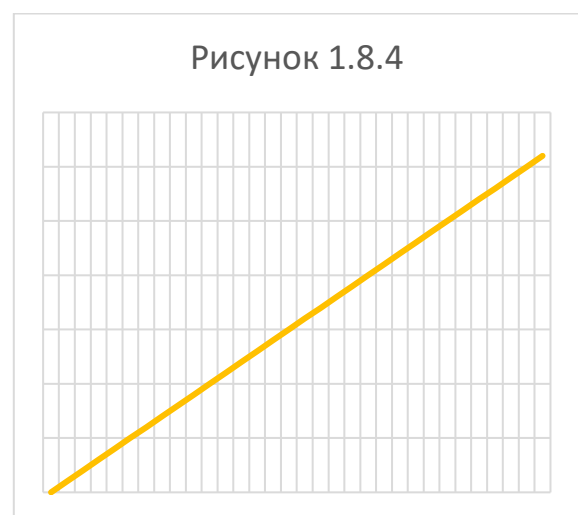




Рисунок 1.8.1

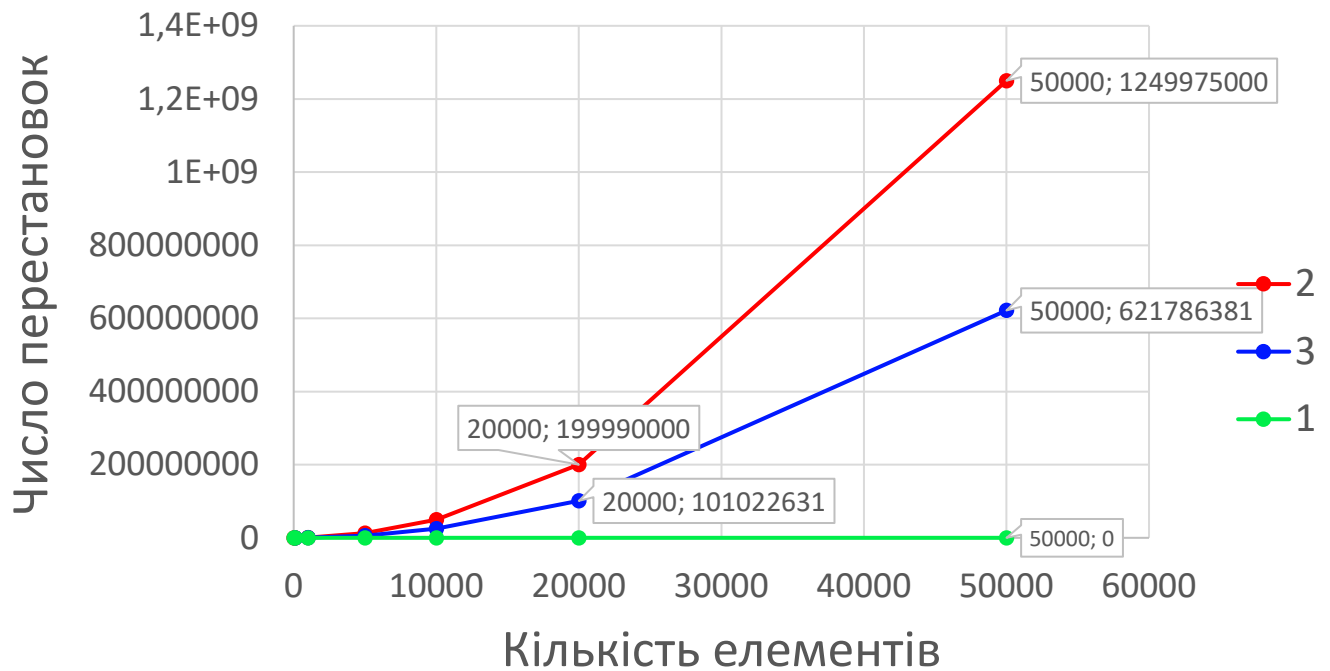
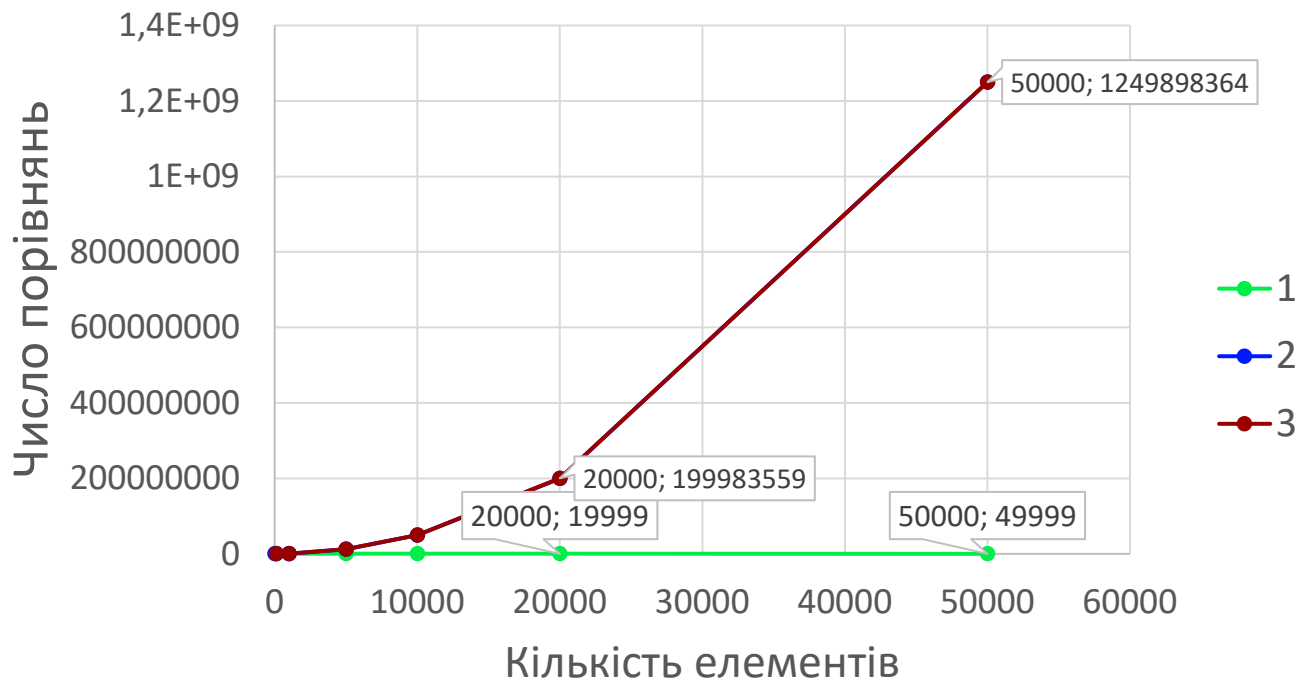


Рисунок 1.8.2



## 2.1 Аналіз алгоритму сортування бульбашкою на відповідність властивостям

| Властивість                                | Сортування бульбашкою  |
|--|--|
| Стійкість                                  | Алгоритм не є стійким  |
| «Природність» поведінки (Adaptability)     | Алгоритм є природним   |
| Базуються на порівняннях                   | Алгоритм базується на порівняннях                              |
| Необхідність в додатковій пам'яті (об'єм)  | Додаткової пам'яті алгоритм не потребує                        |
| Необхідність в знаннях про структури даних | Необхідні знання про базові та лінійні структури даних (масив) |

## 2.2 Псевдокод алгоритму

### Початок

step = lengthOfArr

constant = 1.24733095

### Повторити

Поки step > 1

step = step / constant

Повторити для i від 0 до lengthOfArr - step

Якщо arr[i] > arr[i+step]

то

buf = arr[i]

arr[i] = arr[i+step]

arr[i+step] = buf

Все якщо

Все повторити

Все повторити

Кінець

### 2.3 Аналіз часової складності

Найкращий випадок:  $O(n \log n)$

Найгірший випадок:  $O(n^2)$

Середній випадок:  $O(n^2 / 2^p)$

### 2.4 Програмна реалізація алгоритму; Вихідний код

```
Main.java x BubbleSort.java x
7 public class Main {
8     public static void main(String[] args) {
9         Scanner in = new Scanner(System.in);
10        int n = in.nextInt();
11        int[] arr = new int[n]; // кол-во елементів в масиві
12        for(int i = 0; i < n; i++) {
13            arr[i] = (int)(Math.random() * (double)(n + 1)); //in.nextInt(); //arr[i] = n-i;
14        }
15        System.out.print("Random array:"); output(n, arr);
16        //bubbleSort(arr);
17        int buf, comparison = 0, swap = 0, step = arr.length;
18        double constant = 1.24733095;
19        while (step > 1) {
20            step = (int) (step / constant);
21            for(int i = 0; step + i < arr.length; i++) {
22                comparison++;
23                if(arr[i] > arr[i + step]) {
24                    buf = arr[i];
25                    arr[i] = arr[i + step];
26                    arr[i + step] = buf;
27                    swap++;
28                }
29            }
30        }
31        System.out.print("\nSorted array:"); output(n, arr);
32        System.out.println("\nComparisons = " + comparison + "; Swaps = " + swap);
33    }
```

### 2.4 Приклад роботи програми сортування масиву на 100 елементів

```
100
Random array:
[83, 29, 11, 63, 77, 64, 51, 91, 29, 3, 58, 4, 40, 43, 86, 97, 73, 14, 72, 39, 95, 90, 7, 28, 65, 70, 68, 42,
59, 24, 31, 70, 84, 31, 49, 82, 96, 99, 8, 44, 69, 45, 55, 86, 44, 35, 46, 79, 0, 4, 76, 28, 35, 100, 21,
78, 87, 11, 53, 7, 53, 0, 94, 28, 50, 37, 26, 23, 43, 26, 86, 38, 15, 18, 67, 86, 92, 54, 98, 55, 37, 24,
5, 15, 18, 98, 76, 35, 27, 81, 62, 9, 55, 83, 47, 65, 68, 48, 24, 83]

Sorted array:
[0, 0, 3, 4, 4, 5, 7, 7, 8, 9, 11, 11, 14, 15, 15, 18, 18, 21, 23, 24, 24, 24, 26, 26, 27, 28, 28, 28,
29, 29, 31, 31, 35, 35, 35, 37, 37, 38, 39, 40, 42, 43, 43, 44, 44, 45, 46, 47, 48, 49, 50, 51, 53, 53, 54,
55, 55, 55, 58, 59, 62, 63, 64, 65, 65, 67, 68, 68, 69, 70, 70, 72, 73, 76, 76, 77, 78, 79, 81, 82, 83, 83,
83, 84, 86, 86, 86, 86, 87, 90, 91, 92, 94, 95, 96, 97, 98, 98, 99, 100]

Comparisons = 1229; Swaps = 242
```

### 2.5 Приклад роботи програми сортування масиву на 1000 елементів

```
1000
Random array:
[9, 772, 495, 180, 684, 853, 81, 138, 466, 108, 640, 413, 192, 111, 13, 821, 308, 483, 706, 365, 971, 537, 76, 329, 38, 316, 413, 900,
690, 812, 592, 87, 347, 506, 194, 929, 347, 709, 528, 694, 343, 538, 759, 113, 709, 843, 456, 419, 332, 771, 210, 352, 68, 193, 619,
139, 737, 682, 872, 875, 49, 303, 362, 611, 14, 40, 226, 9, 410, 258, 504, 305, 667, 290, 641, 941, 89, 839, 390, 228, 680, 633,
781, 15, 951, 471, 705, 428, 14, 562, 870, 431, 964, 854, 285, 118, 963, 866, 213, 351, 783, 393, 516, 624, 504, 338, 640, 786, 948,
824, 48, 581, 975, 716, 773, 782, 183, 637, 912, 78, 675, 573, 294, 762, 520, 209, 282, 702, 247, 210, 606, 56, 206, 753, 455, 645,
923, 762, 553, 964, 629, 459, 795, 348, 996, 35, 120, 892, 732, 480, 714, 353, 105, 854, 255, 407, 637, 614, 949, 311, 193, 869, 236,
246, 449, 833, 680, 333, 216, 135, 237, 621, 469, 15, 275, 130, 970, 932, 8, 371, 282, 543, 692, 640, 625, 579, 827, 784, 446, 601,
830, 58, 816, 274, 127, 155, 756, 346, 359, 499, 381, 176, 479, 393, 305, 302, 816, 696, 112, 268, 808, 249, 810, 614, 997, 453, 89,
730, 36, 973, 926, 43, 946, 786, 476, 836, 545, 477, 540, 837, 54, 927, 846, 632, 113, 312, 231, 832, 72, 665, 414, 892, 257, 31,
401, 454, 59, 689, 744, 105, 116, 746, 789, 451, 563, 879, 804, 513, 305, 957, 331, 532, 557, 746, 446, 422, 389, 962, 607, 248, 931,
454, 682, 331, 891, 415, 329, 788, 555, 120, 323, 594, 990, 851, 611, 521, 221, 331, 859, 645, 331, 238, 973, 461, 977, 62, 656, 31,
591, 887, 321, 999, 653, 460, 834, 757, 720, 264, 370, 538, 613, 153, 407, 687, 17, 544, 218, 930, 8, 53, 547, 31, 456, 958, 66,
813, 79, 832, 60, 151, 905, 250, 297, 254, 282, 501, 430, 180, 164, 823, 376, 310, 279, 823, 20, 246, 990, 372, 318, 387, 82, 443,
286, 262, 481, 88, 670, 543, 871, 984, 88, 568, 534, 740, 248, 87, 286, 505, 473, 912, 232, 191, 96, 127, 230, 49, 490, 521, 368,
202, 387, 932, 448, 192, 919, 291, 479, 536, 243, 238, 435, 802, 987, 310, 73, 724, 695, 888, 462, 445, 438, 260, 952, 42, 249, 417,
733, 135, 539, 416, 784, 508, 801, 477, 980, 354, 878, 272, 353, 981, 844, 225, 24, 546, 47, 35, 124, 663, 215, 43, 81, 532, 571,
492, 49, 589, 843, 244, 854, 518, 650, 2, 774, 899, 219, 613, 698, 655, 606, 635, 638, 976, 573, 930, 391, 332, 89, 674, 160, 762,
959, 857, 292, 651, 422, 31, 886, 810, 746, 801, 928, 772, 657, 292, 779, 903, 220, 649, 290, 60, 212, 413, 560, 291, 272, 491, 376,
253, 181, 320, 649, 880, 404, 721, 767, 49, 643, 918, 629, 914, 512, 136, 265, 855, 606, 248, 386, 999, 184, 917, 889, 586, 396, 825,
341, 505, 472, 617, 749, 760, 619, 373, 419, 686, 366, 542, 604, 276, 19, 505, 691, 61, 445, 145, 113, 614, 854, 54, 821, 900, 427,
315, 93, 361, 604, 101, 753, 854, 585, 698, 291, 284, 156, 932, 278, 70, 257, 412, 474, 370, 766, 423, 406, 336, 927, 803, 873, 536,
421, 631, 689, 84, 584, 109, 248, 943, 453, 569, 926, 323, 351, 165, 133, 116, 484, 430, 882, 797, 594, 529, 21, 299, 64, 773, 963,
963, 984, 651, 552, 746, 596, 403, 974, 580, 243, 977, 962, 881, 388, 972, 399, 992, 540, 104, 344, 528, 950, 536, 849, 411, 758, 375,
515, 647, 855, 589, 582, 776, 111, 530, 488, 259, 785, 314, 273, 992, 831, 306, 883, 529, 603, 56, 434, 631, 891, 110, 767, 155, 544,
286, 133, 400, 2, 136, 743, 552, 873, 538, 951, 941, 342, 571, 302, 484, 615, 94, 302, 788, 742, 749, 730, 108, 201, 399, 564, 545,
310, 402, 103, 614, 121, 483, 641, 598, 712, 373, 167, 898, 945, 876, 900, 926, 571, 422, 112, 969, 290, 777, 625, 833, 729, 9, 557,
23, 613, 485, 401, 668, 323, 184, 255, 387, 882, 333, 97, 843, 913, 950, 319, 750, 521, 648, 338, 716, 123, 742, 999, 406, 189, 466,
656, 164, 531, 900, 286, 422, 259, 36, 694, 906, 762, 251, 908, 941, 98, 341, 165, 884, 29, 199, 453, 43, 474, 148, 430, 289, 577,
930, 711, 884, 721, 350, 900, 33, 759, 634, 477, 13, 471, 595, 631, 736, 717, 855, 367, 108, 90, 449, 814, 539, 893, 352, 439, 525,
131, 621, 27, 331, 835, 113, 636, 385, 425, 974, 445, 179, 31, 376, 351, 126, 646, 345, 397, 47, 733, 807, 650, 711, 456, 429, 939,
103, 2, 208, 921, 937, 567, 215, 348, 208, 444, 384, 1, 539, 326, 748, 709, 365, 354, 40, 69, 25, 672, 199, 979, 236, 254, 848,
817, 434, 549, 524, 224, 5, 33, 56, 707, 456, 146, 186, 913, 184, 169, 281, 347, 622, 538, 720, 427, 160, 760, 336, 254, 477, 586,
122, 221, 554, 321, 250, 559, 958, 202, 341, 270, 713, 593, 851, 427, 388, 310, 355, 718, 895, 814, 528, 229, 808, 109, 592, 900, 972,
185, 66, 765, 557, 494, 714, 133, 698, 515, 684, 41, 328, 711, 994, 951, 891, 233, 633, 425, 99, 618, 807, 671, 61, 287, 15, 187,
671, 135, 985, 483, 325, 612, 680, 721, 719, 732, 36, 441, 657, 157, 324, 980, 584, 53, 488, 39, 812, 708, 203, 917, 207, 36, 849,
271, 465, 934, 43, 954, 935, 101, 954, 334, 401, 441, 60, 778, 105, 282, 693, 397, 809, 756, 211, 987, 635, 377, 371, 74, 8, 479,
45, 378, 21, 482, 212, 537, 164, 96, 796, 915, 476, 495, 758, 404, 813, 138, 583, 651, 112, 441, 156, 835, 261, 696, 862, 153, 384]
```

## Основи програмування – 2. Алгоритми та структури даних

Sorted array:

```
[1, 2, 2, 2, 5, 8, 8, 8, 9, 9, 9, 13, 13, 14, 14, 15, 15, 15, 17, 19, 20, 21, 21, 23, 24, 25, 27, 29,
31, 31, 31, 31, 31, 33, 33, 35, 35, 36, 36, 36, 36, 38, 39, 40, 40, 41, 42, 43, 43, 43, 43, 45, 47, 47, 48,
49, 49, 49, 49, 53, 53, 54, 54, 56, 56, 56, 58, 59, 60, 60, 60, 61, 61, 62, 64, 66, 66, 68, 69, 70, 72, 73,
74, 76, 78, 79, 81, 81, 82, 84, 87, 87, 88, 88, 89, 89, 89, 90, 93, 94, 96, 96, 97, 98, 99, 101, 101, 103, 103,
104, 105, 105, 105, 108, 108, 108, 109, 109, 110, 111, 111, 112, 112, 112, 113, 113, 113, 113, 116, 116, 118, 120, 120, 121, 122, 123,
124, 126, 127, 127, 130, 131, 133, 133, 133, 135, 135, 135, 136, 136, 138, 138, 139, 145, 146, 148, 151, 153, 153, 155, 155, 156, 156,
157, 160, 160, 164, 164, 164, 164, 165, 165, 167, 169, 176, 179, 180, 180, 181, 183, 184, 184, 184, 185, 186, 187, 189, 191, 192, 192, 193,
193, 194, 199, 199, 201, 202, 202, 203, 206, 207, 208, 208, 209, 210, 210, 211, 212, 212, 213, 215, 215, 216, 218, 219, 220, 221, 221,
224, 225, 226, 228, 229, 230, 231, 232, 233, 236, 236, 237, 238, 238, 243, 243, 244, 246, 246, 247, 248, 248, 248, 248, 249, 249, 250,
250, 251, 253, 254, 254, 255, 255, 257, 257, 258, 259, 260, 261, 262, 264, 265, 268, 270, 271, 272, 272, 273, 274, 275, 276,
278, 279, 281, 282, 282, 282, 284, 285, 286, 286, 286, 287, 289, 290, 290, 290, 291, 291, 291, 292, 292, 294, 297, 299, 302,
302, 302, 303, 305, 305, 305, 306, 308, 310, 310, 310, 310, 311, 312, 314, 315, 316, 318, 319, 320, 321, 321, 323, 323, 323, 324, 325,
326, 328, 329, 329, 331, 331, 331, 331, 331, 332, 332, 333, 333, 334, 336, 336, 338, 338, 341, 341, 341, 342, 343, 344, 345, 346, 347,
347, 347, 348, 348, 350, 351, 351, 351, 352, 352, 353, 353, 354, 354, 355, 359, 361, 362, 365, 365, 366, 367, 368, 370, 370, 371, 371,
372, 373, 373, 375, 376, 376, 376, 377, 378, 381, 384, 384, 386, 385, 387, 387, 387, 388, 388, 389, 390, 391, 393, 393, 396, 397, 397,
399, 399, 400, 401, 401, 401, 402, 403, 404, 404, 406, 406, 407, 407, 410, 411, 412, 413, 413, 413, 414, 415, 416, 417, 419, 419, 421,
422, 422, 422, 422, 423, 425, 425, 427, 427, 428, 429, 430, 430, 431, 434, 434, 435, 438, 439, 441, 441, 441, 443, 444, 445,
445, 445, 446, 446, 448, 448, 449, 449, 451, 453, 453, 453, 454, 454, 455, 456, 456, 456, 456, 459, 460, 461, 462, 465, 466, 466, 469, 471,
471, 472, 473, 474, 474, 476, 476, 477, 477, 477, 477, 479, 479, 479, 480, 481, 482, 483, 483, 483, 484, 484, 485, 488, 488, 490, 491,
492, 494, 495, 495, 499, 501, 504, 504, 505, 505, 505, 506, 508, 512, 513, 515, 515, 516, 518, 520, 521, 521, 521, 524, 525, 528, 528,
528, 529, 529, 530, 531, 532, 532, 534, 536, 536, 536, 537, 537, 538, 538, 538, 538, 539, 539, 539, 540, 540, 542, 543, 543, 544, 544,
545, 545, 546, 547, 549, 552, 552, 553, 554, 555, 557, 557, 557, 559, 560, 562, 563, 564, 567, 568, 569, 571, 571, 571, 573, 573, 577,
579, 580, 581, 582, 583, 584, 584, 585, 586, 586, 589, 589, 591, 592, 592, 593, 594, 594, 595, 596, 598, 601, 603, 604, 604, 606, 606,
606, 607, 611, 611, 612, 613, 613, 613, 614, 614, 614, 614, 615, 617, 618, 619, 619, 621, 621, 622, 624, 625, 625, 629, 629, 631, 631,
631, 632, 633, 633, 634, 635, 635, 636, 637, 637, 638, 640, 640, 640, 641, 641, 643, 645, 645, 646, 647, 648, 649, 649, 650, 650, 651,
651, 651, 653, 655, 656, 656, 657, 657, 663, 665, 667, 668, 670, 671, 671, 672, 674, 675, 680, 680, 680, 682, 682, 684, 684, 686, 687,
689, 689, 690, 691, 692, 693, 694, 694, 695, 696, 696, 698, 698, 698, 702, 705, 706, 707, 708, 709, 709, 709, 711, 711, 711, 712, 713,
714, 714, 716, 716, 717, 718, 719, 720, 720, 721, 721, 721, 724, 729, 730, 730, 732, 732, 733, 733, 736, 737, 740, 742, 742, 743, 744,
746, 746, 746, 746, 748, 749, 749, 750, 753, 753, 756, 756, 757, 758, 758, 759, 759, 760, 760, 762, 762, 762, 762, 765, 766, 767, 767,
771, 772, 772, 773, 773, 774, 776, 777, 778, 779, 781, 782, 783, 784, 784, 785, 786, 786, 788, 788, 789, 795, 796, 797, 801, 801, 802,
803, 804, 807, 807, 808, 808, 809, 810, 810, 812, 812, 813, 813, 814, 814, 816, 816, 817, 821, 821, 823, 823, 824, 825, 827, 830, 831,
832, 832, 833, 833, 834, 835, 835, 836, 837, 839, 843, 843, 843, 844, 846, 848, 849, 849, 851, 851, 853, 854, 854, 854, 854, 855,
855, 855, 857, 859, 862, 866, 869, 870, 871, 872, 873, 873, 875, 876, 878, 879, 880, 881, 882, 882, 883, 884, 884, 886, 887, 888, 889,
891, 891, 891, 892, 892, 893, 895, 898, 899, 900, 900, 900, 900, 900, 900, 903, 905, 906, 908, 912, 912, 913, 913, 914, 915, 917, 917,
918, 919, 921, 923, 926, 926, 926, 927, 927, 928, 929, 930, 930, 930, 931, 932, 932, 932, 934, 935, 937, 939, 941, 941, 941, 943, 945,
946, 948, 949, 950, 950, 951, 951, 951, 952, 954, 954, 957, 958, 958, 959, 962, 962, 963, 963, 963, 964, 964, 969, 970, 971, 972, 972,
973, 973, 974, 974, 975, 976, 977, 977, 979, 980, 980, 981, 984, 984, 985, 987, 987, 990, 990, 992, 992, 994, 996, 997, 999, 999, 999]
```

Comparisons = 22022; Swaps = 4002

## 2.7 Часові характеристики оцінювання

В таблиці 2.7.1 наведені характеристики оцінювання алгоритму сортування бульбашки для впорядкованої послідовності елементів у масиві.

Таблиця 2.7.1

| Розмірність масиву | Число порівнянь | Число перестановок |
|--------------------|-----------------|--------------------|
| 10                 | 36              | 0                  |
| 100                | 1229            | 0                  |
| 1000               | 22022           | 0                  |
| 5000               | 144862          | 0                  |
| 10000              | 329644          | 0                  |
| 20000              | 719241          | 0                  |
| 50000              | 1997958         | 0                  |

В таблиці 2.7.2 наведені характеристики оцінювання алгоритму сортування бульбашки для зворотно впорядкованої послідовності елементів у масиві.

Таблиця 2.7.2

| Розмірність масиву | Число порівнянь | Число перестановок |
|--------------------|-----------------|--------------------|
| 10                 | 36              | 9                  |
| 100                | 1229            | 110                |
| 1000               | 22022           | 1512               |
| 5000               | 144862          | 9016               |
| 10000              | 329644          | 19132              |
| 20000              | 719241          | 40852              |
| 50000              | 1997958         | 109958             |

В таблиці 2.7.3 наведені характеристики оцінювання алгоритму сортування бульбашки для випадкової послідовності елементів у масиві.

Таблиця 2.7.3

| Розмірність масиву | Число порівнянь | Число перестановок |
|--------------------|-----------------|--------------------|
| 10                 | 36              | 4                  |
| 100                | 1229            | 211                |
| 1000               | 22022           | 4030               |
| 5000               | 144862          | 25591              |
| 10000              | 329644          | 57112              |
| 20000              | 719241          | 124466             |
| 50000              | 1997958         | 353199             |

## 2.8 Графіки залежності часових характеристик оцінювання від розмірності масиву

На рисунку 2.8.1 та 2.8.2 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 на рисунках 2.8.3 та 2.8.4 показані асимптотичні оцінки гіршого (фіолетовий графік) і кращого (жовтий графік) випадків для порівняння.

Рисунок 2.8.3

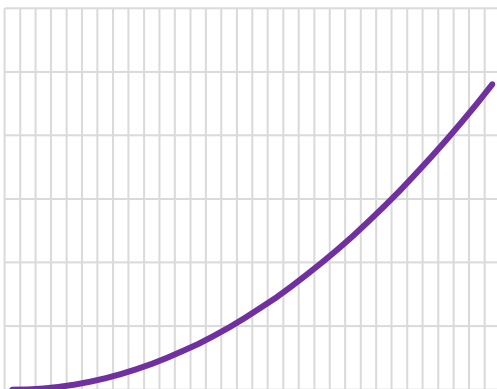


Рисунок 2.8.4

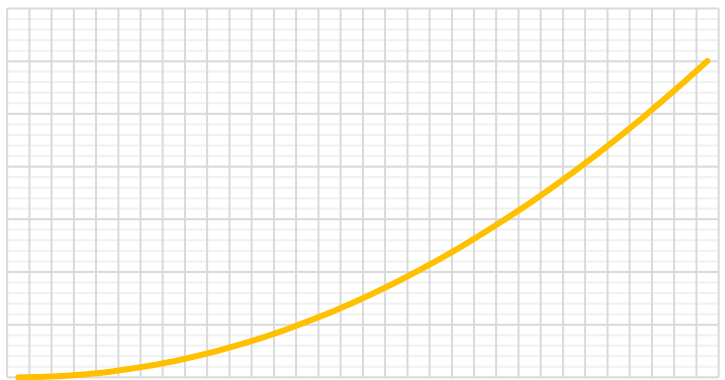




Рисунок 2.8.1

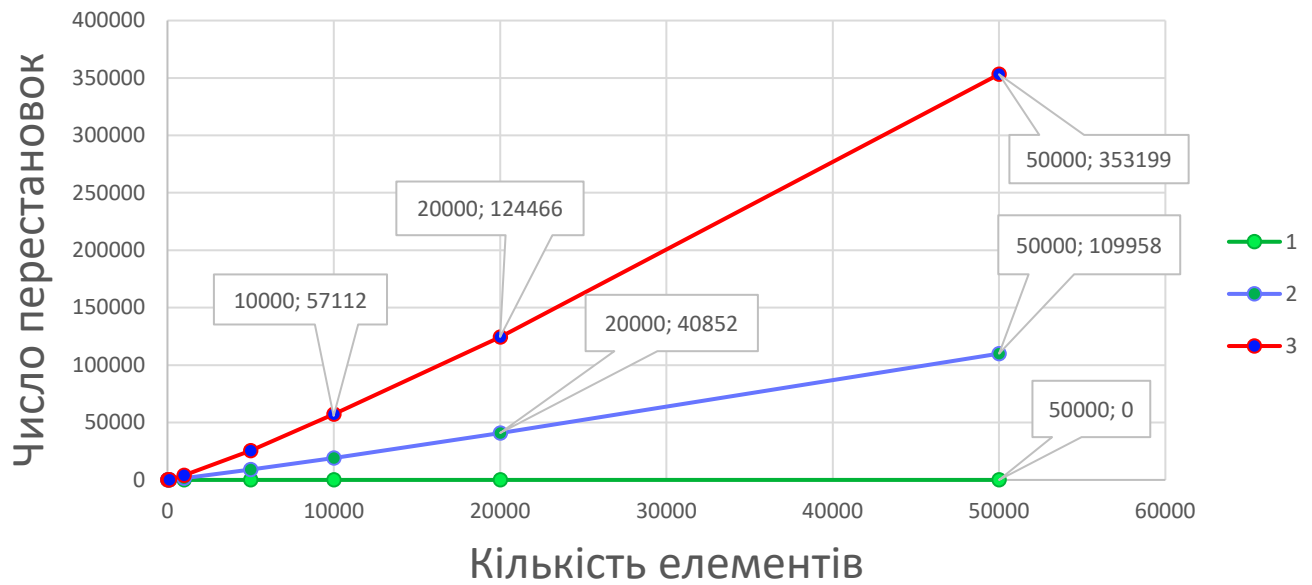
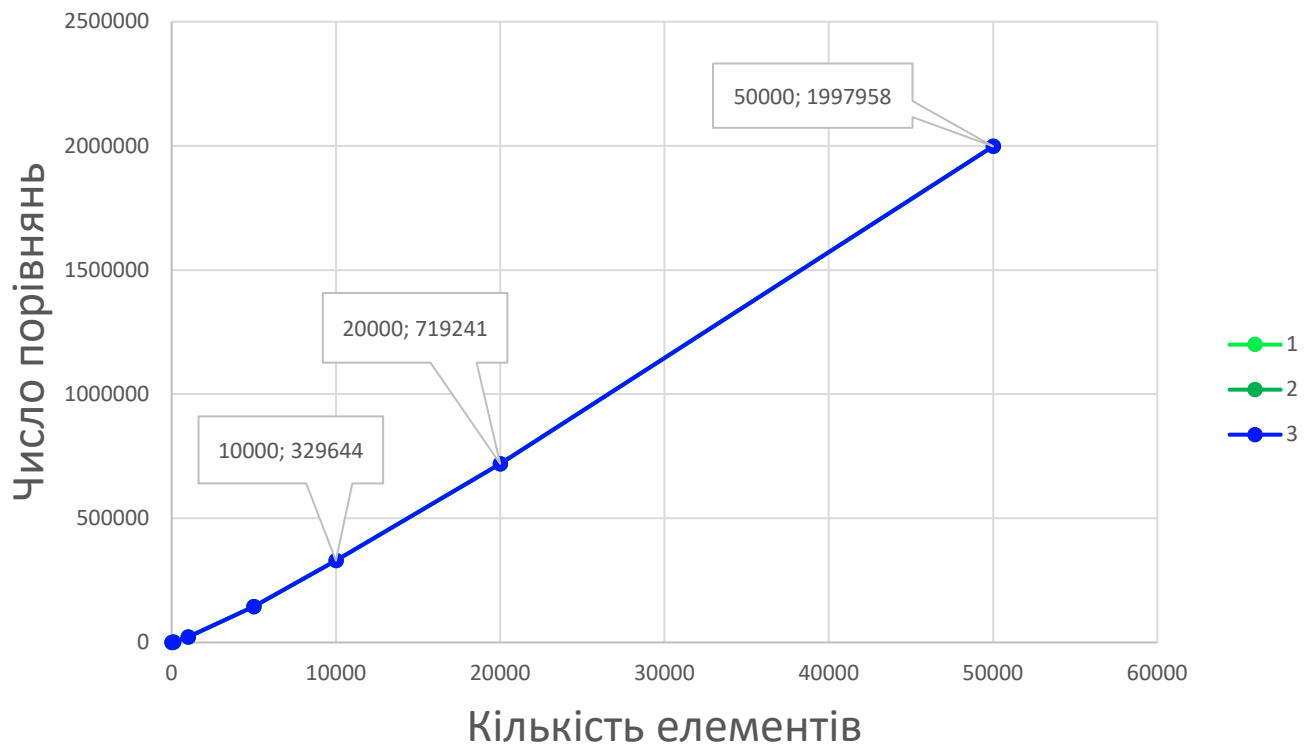


Рисунок 2.8.2

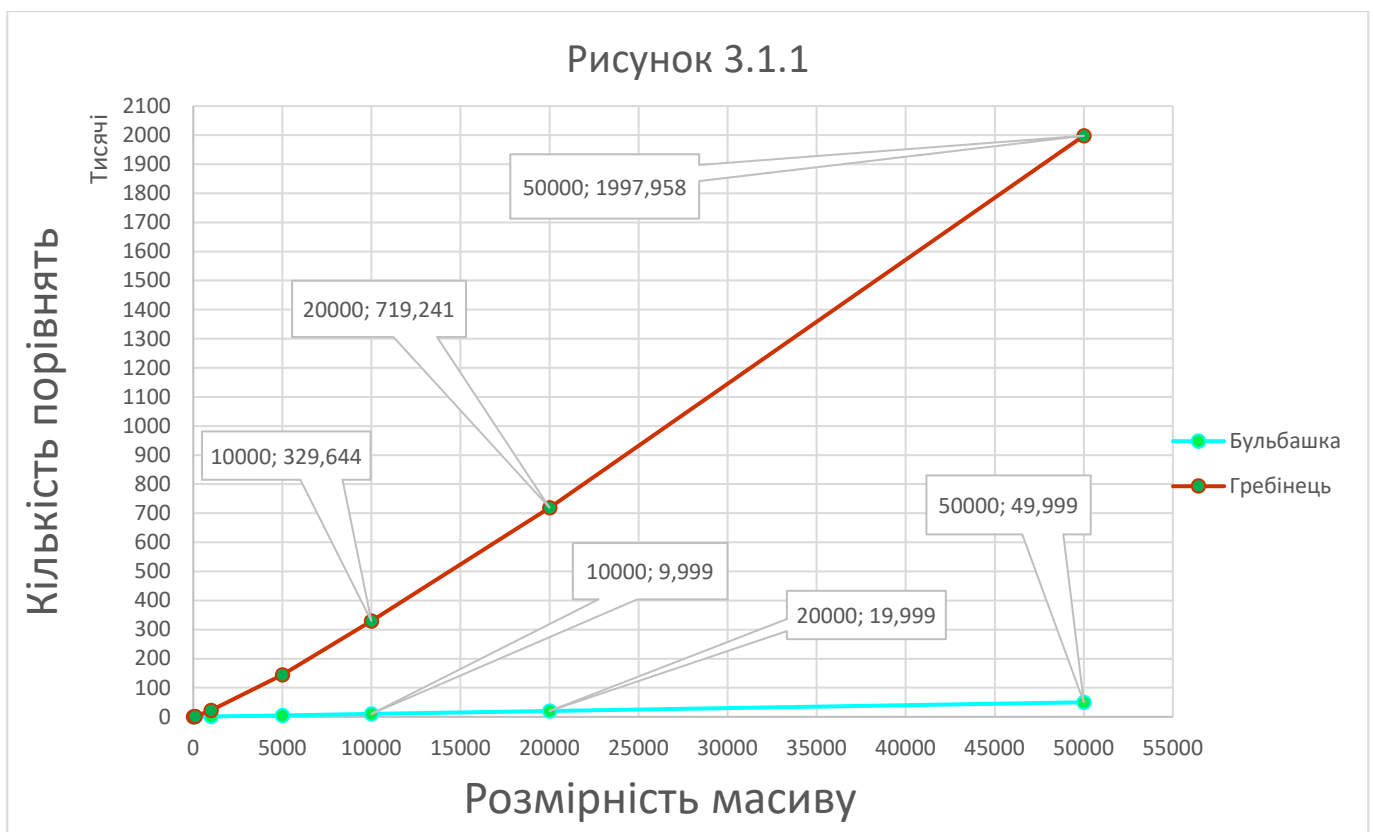


### 3.1 Зробимо порівняльний аналіз алгоритмів сортування

Спираючись на властивості алгоритмів, можна сказати, що сортування бульбашкою є стійким, на відміну від сортування гребінцем, що також пришвидшує роботу алгоритму.

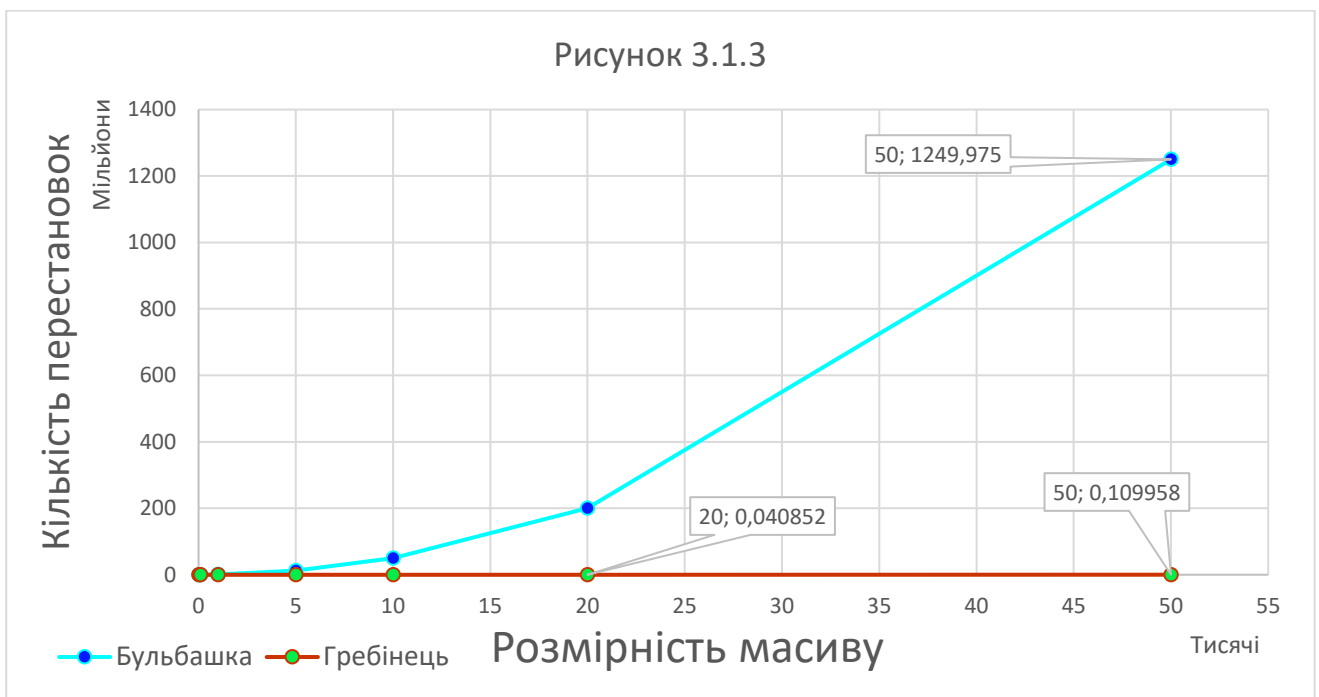
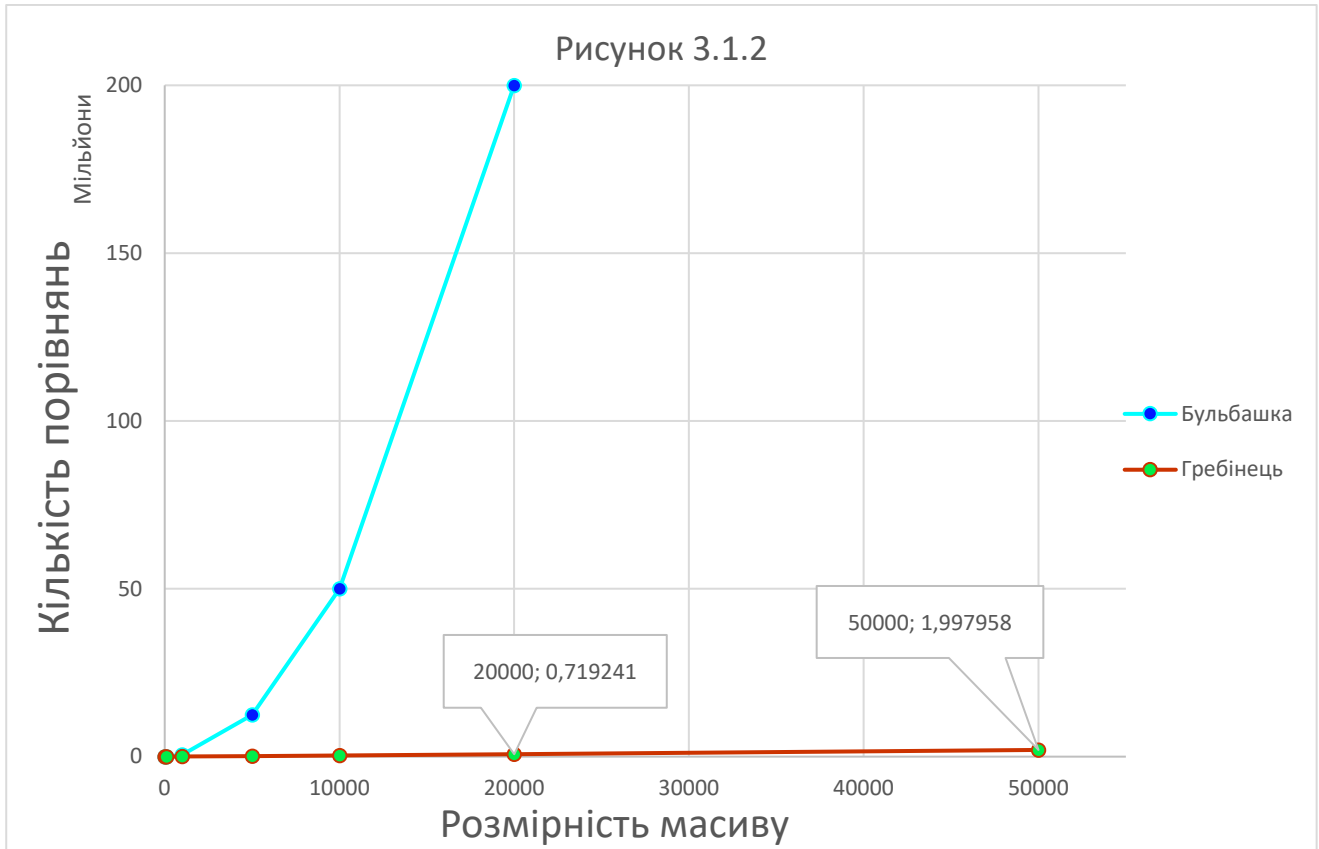
Порівняємо графіки залежності часових характеристик оцінювання від розмірності масиву.

Так як обидва алгоритми є природними, то порівнювати графіки залежності числа перестановок від розмірності масиву для вже впорядкованих масивів немає сенсу, але кількість порівнянь в них різна, що показано на рисунку 3.1.1.



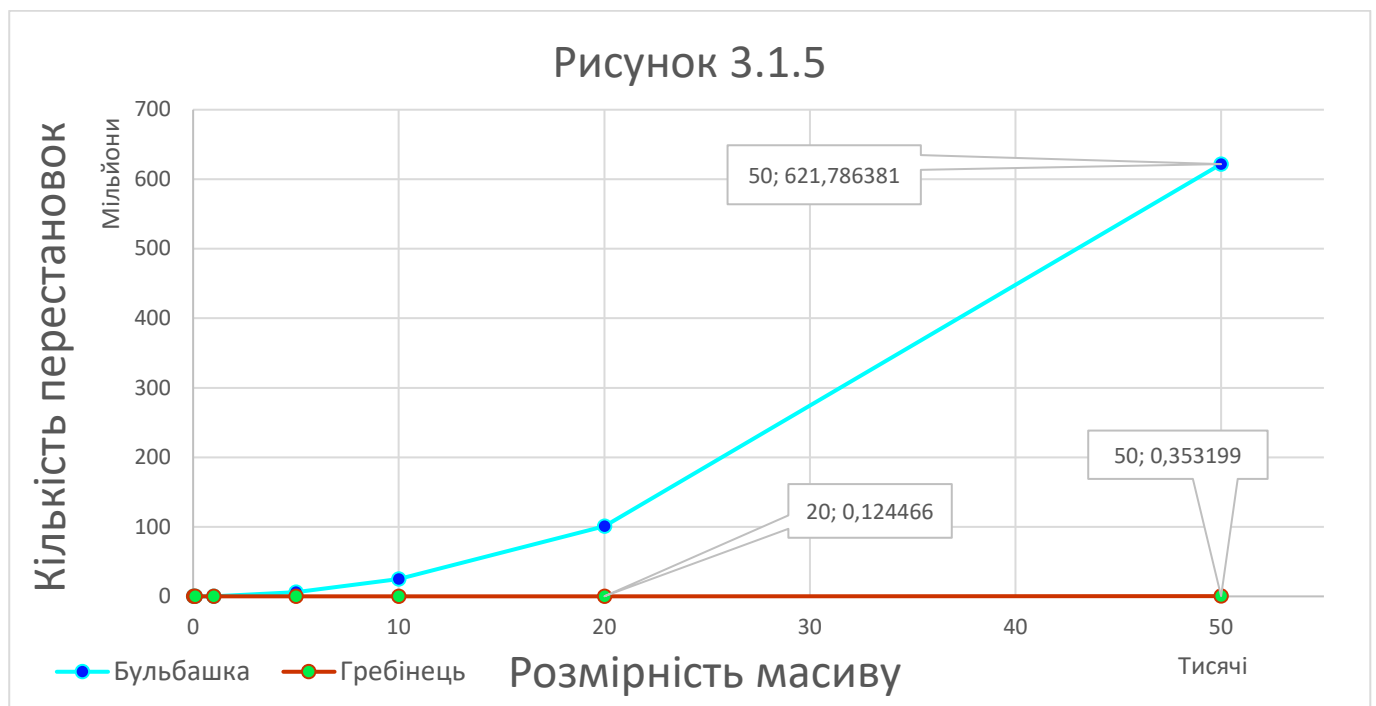
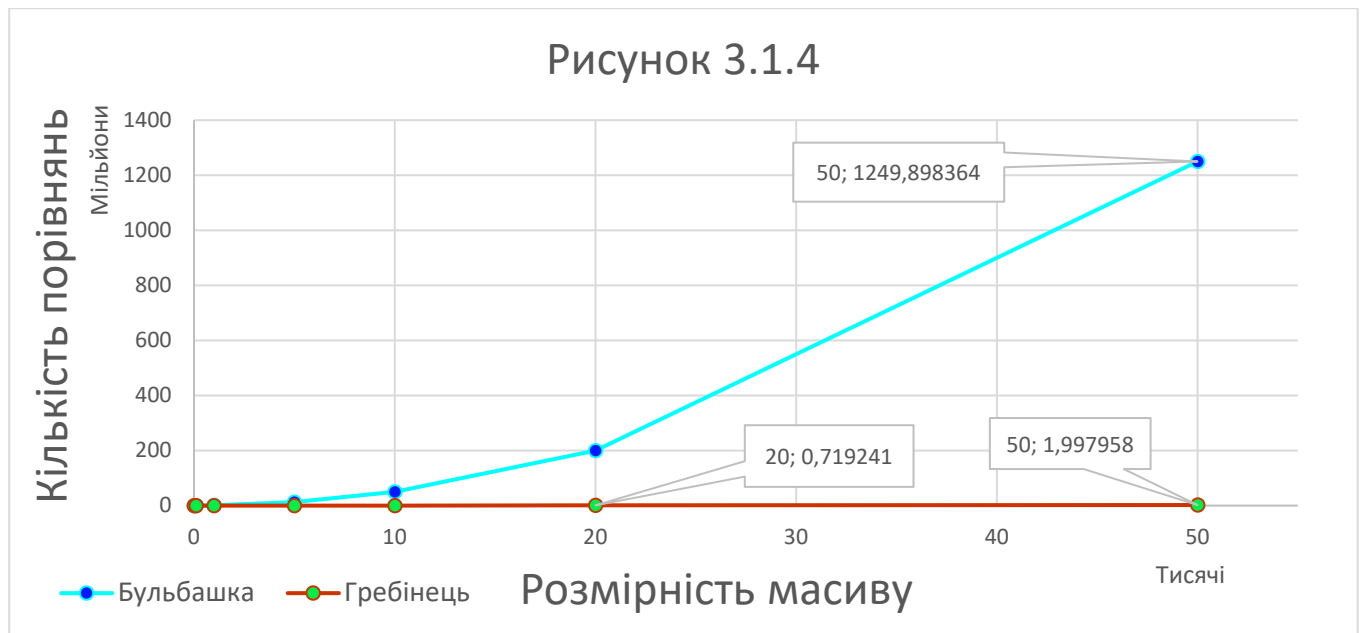
Можемо побачити, що на вже відсортованому масиві бульбашка працює набагато швидше та ефективніше.

На рисунках 3.1.2 та 3.1.3 розглянемо графіки кількості порівнянь та перестановок до кількості елементів у масиві відповідно.



З графіків видно, що для зворотно впорядкованої послідовності елементів у масиві, сортування гребінцем набагато ефективніше та швидкіше за сортування бульбашкою.

Тепер порівняємо алгоритми на випадковій послідовності елементів на рисунках 3.1.4 та 3.1.5.



## ВИСНОВОК

При виконанні даної лабораторної роботи вивчили основні методи аналізу обчислювальної складності алгоритмів внутрішнього сортування і оцінили поріг їх ефективності.

Розглянули алгоритми сортування бульбашкою та гребінцем. Дослідили їх властивості, провели аналіз часової складності, провели ряд випробувань на різних наборах вхідних даних. За допомогою графіків порівняли часові характеристики оцінювання.

Отже, можемо прийти до висновку, що сортування бульбашкою є ефективним, якщо масив повністю або частково відсортований, для всіх інших випадків сортування гребінцем набагато краще, бо робить значно менше порівнянь та перестановок.