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

Кафедра ІПІ

Звіт

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

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

Виконав(ла) *ІП-15, Закірова Олександра Володимирівна* _____
(шифр, прізвище, ім'я, по батькові)

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

Київ 2022

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
№1

3.1 Аналіз алгоритму на відповідність властивостям

Аналіз алгоритму ***сортування бульбашкою*** на відповідність властивостям наведено в таблиці 3.1.

Таблиця 3.1 – Аналіз алгоритму 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)	Алгоритм не потребує додаткової пам’яті
Необхідність в знаннях про структури даних	Для використання алгоритму потрібно мати базові знання про структури даних

3.2 Псевдокод алгоритму

Підпрограма BubbleSort(arr):

Початок

flag = true

Повторити для i від 0 до len(arr):

Повторити для j від 0 до len(arr) – i:

Якщо (arr[j] > arr[j+1]):

temp = arr[j]

arr[j] = arr[j+1]

arr[j+1] = temp

flag = false

Все повторити

Якщо flag = true:

break

Все повторити

Кінець

3.3 Аналіз часової складності

Найкращій випадок — O(n)

Найгірший випадок — O(n*log(n))

Середній випадок — Ω(n*log(n))

3.4 Програмна реалізація алгоритму

3.4.1 Вихідний код

```
import random


def bubbleSort(arr):
    flag = True
    compares = 0
    swaps = 0
    for i in range(len(arr)):
        for j in range(len(arr) - i - 1):
            compares += 1
            if arr[j] > arr[j + 1]:
                temp = arr[j]
                arr[j] = arr[j + 1]
                arr[j + 1] = temp
                swaps += 1
                flag = False
        if flag:
            print('Array is already sorted')
            break
    print("Compares: ", compares)
    print("Swaps: ", swaps)
    return arr


n = 50000

randArr = random.sample(range(1, n + 1), n)
print('Randomized array: ', randArr)
sortArr = bubbleSort(randArr)
print('Sorted array: ', sortArr)
```

3.4.2 Приклад роботи

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

Рисунок 3.1 – Сортування масиву на 100 елементів

```
C:\Users\Legion\PycharmProjects\bubbleSort\venv\Scripts\python.exe C:/Users/Legion/PycharmProjects/bubbleSort/main.py
Randomized array:  [33, 40, 16, 90, 93, 57, 46, 78, 98, 13, 6, 80, 18, 60, 39, 97, 52, 37, 66, 58, 69, 87, 35, 24, 50, 23, 7, 10, 15, 56, 8, 75, 88, 44, 84, 74, 86, 95, 38, 81, 94, 91, 2, 71, 14, 30,
 77, 26, 11, 100, 27, 61, 63, 32, 73, 99, 34, 83, 28, 59, 5, 65, 54, 67, 76, 17, 96, 92, 55, 4, 19, 51, 68, 72, 79, 70, 25, 42, 49, 1, 64, 89, 53, 43, 12, 47, 21, 22, 48, 85, 62, 20, 45, 29, 31, 9,
 82, 41, 3, 36]
Compares:  4950
Swaps:  2688
Sorted array:  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48,
 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98,
 99, 100]
```

Рисунок 3.2 – Сортування масиву на 1000 елементів

```
Randomized array:  [513, 915, 83, 723, 638, 460, 818, 928, 35, 590, 274, 244, 324, 962, 620, 911, 919, 94, 588, 427, 779, 927, 15, 650, 794, 843, 39, 958, 865, 755, 277, 188, 93, 840, 995, 245, 606,
 758, 903, 416, 549, 330, 320, 197, 317, 383, 194, 468, 133, 993, 534, 467, 179, 959, 499, 500, 436, 451, 321, 286, 264, 661, 72, 391, 252, 141, 772, 285, 684, 374, 343, 316, 522, 608, 591, 81, 281,
 242, 223, 730, 607, 918, 263, 662, 257, 583, 624, 986, 28, 143, 657, 931, 126, 936, 217, 107, 461, 323, 566, 686, 671, 891, 511, 211, 405, 546, 578, 589, 565, 279, 55, 754, 741, 895, 648, 844, 344,
 79, 426, 955, 708, 712, 994, 470, 503, 384, 804, 394, 599, 307, 908, 185, 89, 355, 116, 327, 834, 492, 22, 284, 204, 596, 142, 984, 399, 293, 196, 150, 415, 545, 96, 568, 218, 108, 764, 226, 572,
 392, 960, 398, 258, 907, 665, 560, 386, 670, 759, 952, 319, 781, 329, 977, 199, 433, 154, 575, 664, 9, 633, 824, 785, 595, 573, 890, 514, 939, 232, 303, 892, 189, 135, 149, 666, 564, 828, 166, 757,
 332, 88, 300, 651, 180, 618, 314, 376, 816, 711, 440, 269, 707, 645, 95, 390, 736, 847, 5, 876, 389, 105, 396, 571, 106, 265, 737, 632, 512, 823, 429, 672, 831, 697, 32, 203, 886, 769, 635, 146, 505,
 487, 272, 617, 579, 400, 137, 153, 1, 30, 743, 775, 267, 867, 621, 974, 674, 601, 732, 112, 972, 372, 305, 864, 418, 128, 92, 616, 57, 114, 365, 408, 200, 275, 782, 479, 961, 774, 820, 495, 102,
 412, 875, 637, 437, 90, 482, 558, 367, 239, 253, 849, 926, 656, 110, 975, 388, 145, 934, 721, 631, 756, 125, 535, 821, 956, 659, 42, 113, 819, 262, 37, 313, 526, 249, 987, 945, 942, 158, 366, 647,
 478, 56, 626, 13, 788, 883, 982, 315, 247, 229, 905, 830, 639, 700, 491, 308, 464, 815, 170, 69, 424, 900, 296, 216, 521, 44, 404, 763, 291, 861, 718, 248, 874, 893, 11, 629, 692, 976, 909, 636, 103,
 168, 368, 848, 897, 480, 234, 301, 326, 373, 518, 922, 379, 295, 488, 375, 940, 164, 341, 983, 748, 790, 46, 710, 222, 733, 857, 690, 930, 609, 24, 519, 238, 888, 193, 120, 727, 641, 803, 586, 445,
 598, 471, 669, 627, 906, 181, 192, 122, 1000, 349, 544, 98, 932, 428, 507, 441, 350, 884, 100, 360, 825, 127, 162, 462, 944, 999, 380, 563, 119, 287, 872, 377, 456, 225, 541, 346, 836, 369, 537, 347,
 351, 14, 348, 969, 23, 678, 25, 361, 954, 309, 40, 6, 981, 920, 139, 345, 948, 450, 420, 251, 542, 138, 442, 612, 101, 856, 709, 529, 681, 515, 334, 811, 290, 965, 683, 434, 453, 973, 51, 963, 688,
 469, 882, 528, 60, 868, 311, 4, 812, 256, 747, 752, 423, 799, 419, 559, 220, 183, 649, 310, 753, 714, 839, 496, 276, 77, 951, 538, 370, 602, 340, 744, 801, 109, 746, 502, 910, 477, 66, 771, 336, 144,
 809, 465, 751, 685, 75, 805, 622, 510, 605, 929, 567, 889, 2, 597, 10, 207, 623, 780, 463, 837, 184, 603, 985, 8, 682, 422, 827, 989, 520, 322, 498, 859, 604, 878, 74, 45, 634, 31, 97, 7, 902, 677,
 235, 531, 561, 335, 230, 630, 829, 807, 855, 642, 458, 749, 660, 797, 877, 289, 943, 123, 810, 261, 768, 506, 569, 845, 530, 806, 585, 937, 455, 255, 67, 425, 833, 508, 328, 213, 691, 271, 968, 713,
 675, 354, 997, 773, 614, 570, 68, 970, 140, 551, 163, 270, 454, 475, 439, 722, 550, 41, 52, 18, 739, 338, 353, 63, 352, 587, 121, 19, 971, 50, 795, 254, 667, 846, 540, 791, 176, 735, 417, 161, 913,
 615, 157, 210, 421, 761, 209, 212, 738, 533, 950, 793, 362, 904, 817, 879, 333, 118, 838, 640, 789, 701, 552, 646, 85, 381, 625, 294, 494, 104, 132, 484, 946, 236, 205, 726, 160, 48, 862, 776, 497,
 490, 967, 679, 202, 190, 124, 796, 27, 745, 201, 850, 443, 778, 580, 680, 59, 34, 435, 134, 49, 43, 858, 312, 863, 182, 808, 155, 173, 237, 231, 325, 165, 393, 880, 762, 73, 553, 481, 448, 842, 978,
 517, 652, 556, 658, 826, 17, 26, 953, 221, 720, 174, 860, 147, 53, 912, 594, 668, 873, 111, 687, 784, 854, 177, 663, 705, 644, 278, 318, 869, 750, 191, 306, 414, 87, 557, 923, 719, 509, 555, 896,
 689, 527, 466, 78, 554, 957, 382, 214, 734, 171, 131, 240, 702, 870, 331, 156, 871, 431, 243, 20, 787, 582, 742, 167, 525, 493, 613, 653, 457, 358, 980, 894, 728, 938, 152, 654, 16, 947, 403, 47,
 402, 268, 813, 65, 172, 357, 485, 206, 655, 887, 933, 38, 99, 802, 250, 280, 786, 29, 914, 82, 387, 696, 703, 698, 395, 304, 504, 302, 766, 584, 342, 292, 966, 581, 447, 577, 337, 814, 991, 676, 169,
 925, 371, 547, 725, 798, 283, 729, 898, 704, 259, 12, 501, 543, 260, 178, 851, 770, 524, 611, 233, 136, 841, 438, 835, 297, 717, 409, 881, 619, 130, 117, 472, 159, 411, 866, 964, 86, 486, 175, 921,
 724, 298, 80, 198, 64, 407, 339, 219, 885, 996, 489, 282, 363, 832, 3, 228, 523, 715, 536, 924, 215, 476, 792, 401, 151, 783, 852, 699, 70, 548, 84, 446, 731, 364, 378, 432, 33, 61, 992, 643, 916,
 54, 299, 195, 473, 288, 935, 406, 800, 452, 208, 91, 246, 58, 71, 483, 413, 767, 539, 760, 695, 224, 148, 592, 241, 777, 998, 532, 186, 694, 359, 356, 610, 693, 227, 822, 474, 21, 600, 516, 385, 941,
 574, 129, 444, 273, 459, 187, 716, 115, 266, 76, 62, 593, 901, 449, 410, 979, 430, 36, 740, 397, 628, 706, 988, 576, 562, 765, 899, 949, 853, 990, 673, 917]
Compares:  499500
Swaps:  249990
Sorted array:  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48,
 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98,
 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138,
 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177,
 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217,
 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256,
 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296,
 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335,
 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375,
 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414,
 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454,
 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493,
 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533,
 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572,
 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612,
 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651,
 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691,
 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730,
 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770,
 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809,
 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849,
 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888,
 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928,
 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967,
 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000]
```

3.5 Тестування алгоритму

```python
import random


def bubbleSort(arr):
    flag = True
    compares = 0
    swaps = 0
    for i in range(len(arr)):
        for j in range(len(arr) - i - 1):
            compares += 1
            if arr[j] > arr[j + 1]:
                temp = arr[j]
                arr[j] = arr[j + 1]
                arr[j + 1] = temp
                flag = False
                swaps += 1
        if flag:
            break
    print("Compares: ", compares)
    print("Swaps: ", swaps)
    return arr


n = 12
randArr = random.sample(range(1, n + 1), n)
print('Randomized array: ', randArr)
sortArr = bubbleSort(randArr)
print('Sorted array: ', sortArr)
```

```
Randomized array:  [8, 11, 5, 1, 12, 4, 9, 6, 7, 10, 3, 2]
Compares:  66
Swaps:  40
Sorted array:  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

Process finished with exit code 0
```

```python
import random


def bubbleSort(arr):
    flag = True
    compares = 0
    swaps = 0
    for i in range(len(arr)):
        for j in range(len(arr) - i - 1):
            compares += 1
            if arr[j] > arr[j + 1]:
                temp = arr[j]
                arr[j] = arr[j + 1]
                arr[j + 1] = temp
                flag = False
                swaps += 1
        if flag:
            break
    print("Compares: ", compares)
    print("Swaps: ", swaps)
    return arr


n = 20
randArr = random.sample(range(1, n + 1), n)
print('Randomized array: ', randArr)
sortArr = bubbleSort(randArr)
print('Sorted array: ', sortArr)
```

```
Randomized array:  [18, 9, 12, 17, 6, 16, 15, 3, 5, 11, 2, 1, 20, 7, 19, 4, 14, 10, 13, 8]
Compares:  190
Swaps:  107
Sorted array:  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

Process finished with exit code 0
```

3.5.1 Часові характеристики оцінювання

В таблиці 3.2 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування бульбашки для масивів різної розмірності, коли масив містить упорядковану послідовність елементів.

Таблиця 3.2 – Характеристики оцінювання алгоритму сортування бульбашки для упорядкованої послідовності елементів у масиві

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

Розмірність масиву	Число порівнянь	Число перестановок
10	45	45
100	4950	4950
1000	499 500	499 500
5000	12 497 500	12 497 500
10000	49 995 000	49 995 000
20000	199 990 000	199 990 000
50000	1 249 975 000	1 249 975 000

У таблиці 3.4 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування бульбашки для масивів різної розмірності, масиви містять випадкову послідовність елементів.

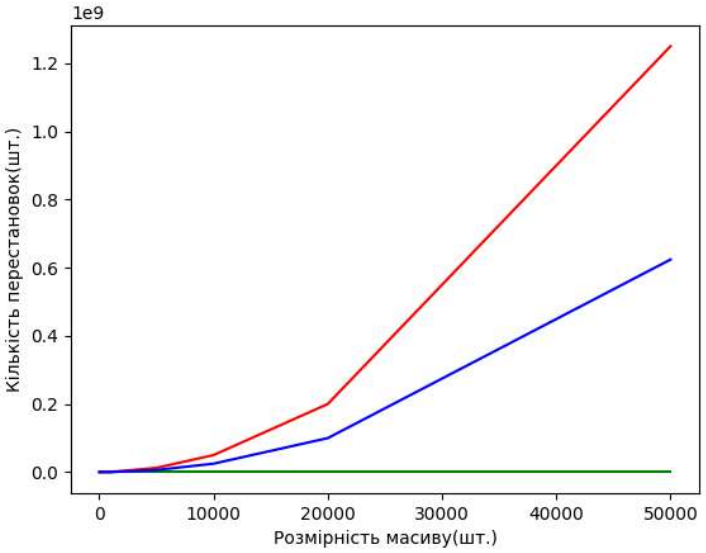
Таблиця 3.4 – Характеристика оцінювання алгоритму сортування бульбашки для випадкової послідовності елементів у масиві.

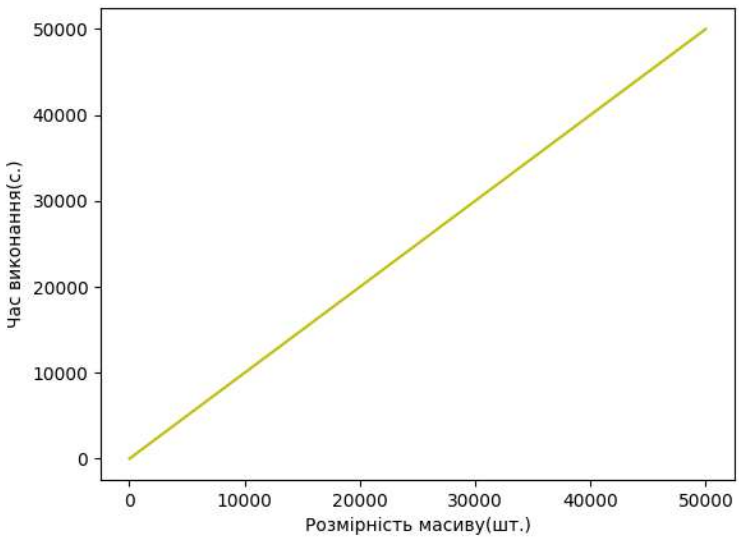
Розмірність масиву	Число порівнянь	Число перестановок
10	45	22
100	4940	2 648
1000	499 500	254 089
5000	12 488 455	6 287 843
10000	49 995 000	24 796 720
20000	188 975 465	99 872 304
50000	1 249 970 905	623 960 500

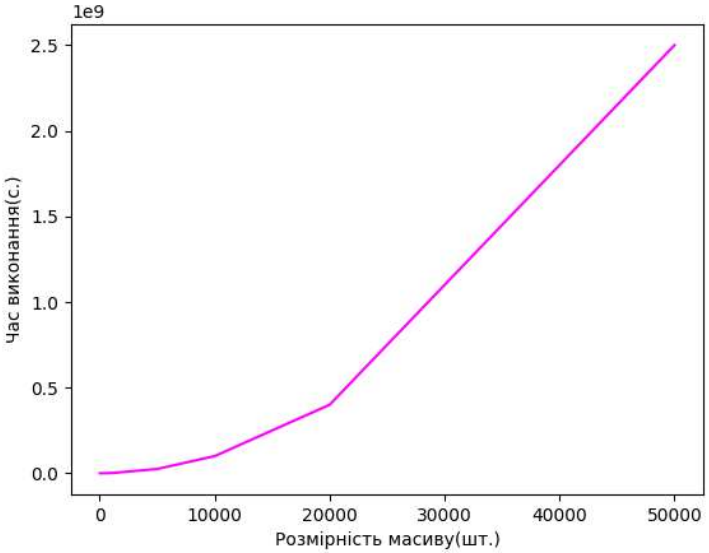
3.5.2 Графіки залежності часових характеристик оцінювання від розмірності масиву

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.

Рисунок 3.3 – Графіки залежності часових характеристик оцінювання







№2

Алгоритм сортування гребінцем

3.6 Аналіз алгоритму на відповідність властивостям

Аналіз алгоритму ***сортування гребінцем*** на відповідність властивостям наведено в таблиці 3.1.

Таблиця 3.1 – Аналіз алгоритму 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)	Алгоритм не потребує додаткової пам’яті
Необхідність в знаннях про структури даних	Для використання алгоритму потрібно мати базові знання про структури даних

Таблиця 3.1

3.7 Псевдокод алгоритму

Підпрограма **sort_bubble(arr):**

Початок

gap = len(arr) – 1

Поки gap >= 1 **повторити:**

Повторити для і від 0 до len(arr) – step:

Якщо (arr[i] > arr[i+gap]):

temp = arr[i]

arr[i] = arr[i+gap]

arr[i+step] = gap

gap = gap/1.247

Все повторити

Все повторити

Кінець

3.8 Аналіз часової складності

Найкращий випадок: O(n*log(n))

Найгірший випадок: O(n 2)

Середній випадок: Ω(n 2 /2p)

3.9 Програмна реалізація алгоритму

3.9.1 Вихідний код

```
import random


def grebenSort(arr):
    compares = 0
    swaps = 0
    gap = len(arr) - 1
    factor = 1.247  # оптимальное число для вычисления шага сравнения
    while gap >= 1:
        for i in range(len(arr) - gap):
            compares += 1
            if arr[i] > arr[i + gap]:
                temp = arr[i]
                arr[i] = arr[i + gap]
                arr[i + gap] = temp
                swaps += 1
        gap = int(gap/factor)
    print("Compares: ", compares)
    print("Swaps: ", swaps)
    return arr


n = 10
randArr = random.sample(range(1, n + 1), n)
print('Randomized array: ', randArr)
sortArr = grebenSort(randArr)
print('Sorted array: ', sortArr)
```

3.9.2 Приклад роботи

На рисунках 3.9.2.1 і 3.9.2.2 показані приклади роботи програми сортування масивів на 100 і 1000 елементів відповідно.

```
Randomized array:  [58, 39, 53, 18, 33, 49, 27, 10, 88, 68, 21, 19, 71, 25, 74, 62, 78, 40, 73, 3, 81, 86, 22, 11, 61, 70, 66, 41, 29, 79, 95, 55, 38, 36, 52, 4, 93, 90, 91, 44, 30, 24, 64, 34, 6, 87,
  100, 75, 65, 13, 47, 56, 51, 17, 7, 72, 28, 46, 50, 59, 15, 37, 96, 16, 94, 26, 1, 83, 63, 2, 84, 98, 12, 23, 9, 8, 35, 42, 67, 5, 97, 76, 99, 45, 89, 92, 48, 60, 57, 80, 77, 31, 20, 43, 32, 82, 54,
  85, 14, 69]
Compares:  1233
Swaps:  260
Sorted array:  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48,
 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98,
  99, 100]
```

Рисунок 3.9.2.1 – Сортування масиву на 100 елементів

```
Randomized array:  [33, 21, 847, 427, 115, 803, 595, 12, 253, 34, 152, 861, 92, 424, 596, 810, 730, 63, 264, 113, 141, 904, 443, 964, 280, 837, 214, 741, 202, 630, 912, 64, 459, 586, 842, 997, 785,
 998, 422, 518, 222, 993, 65, 242, 960, 30, 876, 986, 751, 472, 198, 271, 593, 186, 251, 49, 869, 117, 128, 136, 67, 171, 341, 983, 772, 149, 494, 793, 36, 84, 760, 474, 261, 75, 924, 933, 765, 605,
 901, 852, 80, 14, 484, 786, 336, 971, 981, 293, 259, 486, 195, 213, 7, 544, 375, 808, 936, 413, 804, 976, 710, 771, 781, 532, 969, 669, 887, 62, 458, 859, 820, 666, 550, 310, 19, 863, 838, 491, 217,
 230, 385, 369, 927, 388, 297, 565, 729, 883, 561, 648, 420, 519, 778, 17, 816, 409, 139, 558, 68, 223, 184, 987, 182, 510, 634, 791, 567, 888, 244, 269, 559, 684, 307, 853, 291, 664, 3, 159, 932,
 966, 252, 827, 968, 614, 714, 109, 102, 289, 745, 738, 373, 304, 800, 54, 866, 756, 270, 875, 944, 590, 327, 490, 725, 95, 541, 775, 748, 324, 652, 76, 228, 58, 466, 726, 709, 91, 196, 746, 639, 704,
 653, 329, 620, 623, 560, 301, 602, 487, 1, 193, 185, 674, 611, 74, 638, 589, 880, 692, 659, 164, 702, 683, 632, 780, 832, 365, 376, 910, 60, 391, 870, 192, 150, 737, 828, 548, 881, 686, 407, 292,
 29, 637, 239, 399, 699, 606, 617, 679, 56, 209, 731, 414, 140, 941, 994, 318, 170, 783, 410, 73, 148, 784, 359, 428, 566, 931, 380, 438, 194, 98, 850, 509, 527, 24, 356, 499, 556, 221, 818, 885, 959,
 831, 531, 279, 948, 119, 703, 957, 432, 538, 97, 529, 657, 322, 650, 972, 274, 954, 232, 302, 799, 689, 416, 513, 929, 907, 504, 813, 757, 460, 502, 580, 89, 750, 951, 100, 525, 802, 764, 32, 124,
 481, 431, 395, 120, 296, 740, 31, 705, 168, 555, 587, 240, 114, 913, 199, 79, 249, 88, 840, 175, 471, 609, 526, 849, 131, 908, 501, 641, 44, 40, 367, 795, 190, 615, 371, 644, 51, 742, 96, 651, 415,
 8, 440, 278, 872, 231, 743, 13, 522, 435, 397, 482, 45, 899, 889, 187, 134, 207, 940, 744, 478, 465, 158, 629, 337, 169, 773, 325, 493, 982, 392, 568, 77, 921, 343, 716, 167, 386, 35, 854, 523, 281,
 93, 672, 658, 311, 210, 855, 446, 154, 860, 103, 250, 313, 254, 858, 554, 299, 333, 387, 788, 285, 85, 693, 732, 469, 445, 950, 890, 848, 81, 6, 135, 546, 377, 132, 846, 856, 508, 372, 18, 884, 579,
 898, 874, 206, 583, 917, 452, 701, 798, 151, 926, 179, 599, 557, 992, 233, 300, 454, 811, 685, 894, 727, 879, 812, 636, 178, 328, 915, 183, 607, 902, 27, 681, 378, 640, 708, 15, 189, 146, 654, 403,
 9, 903, 430, 978, 963, 503, 945, 577, 470, 728, 346, 949, 985, 419, 121, 511, 543, 584, 520, 349, 330, 829, 55, 66, 441, 298, 801, 631, 265, 530, 400, 69, 516, 483, 203, 763, 582, 108, 354, 600, 534,
 177, 208, 723, 973, 776, 839, 286, 655, 480, 485, 188, 934, 834, 942, 886, 361, 47, 463, 105, 572, 721, 938, 759, 99, 389, 355, 675, 277, 815, 122, 515, 437, 698, 290, 867, 408, 384, 749, 461, 610,
 809, 256, 344, 284, 537, 690, 601, 506, 718, 955, 396, 110, 918, 235, 468, 282, 382, 127, 10, 524, 930, 479, 360, 308, 314, 805, 581, 52, 668, 722, 977, 835, 895, 22, 4, 133, 591, 381, 406, 952, 633,
 61, 660, 989, 733, 643, 897, 303, 900, 267, 216, 755, 796, 720, 588, 553, 649, 505, 862, 321, 137, 174, 316, 836, 642, 806, 401, 970, 871, 906, 201, 450, 342, 825, 736, 331, 218, 447, 464, 71, 243,
 26, 50, 967, 83, 147, 495, 236, 245, 958, 678, 545, 676, 163, 975, 247, 824, 28, 807, 46, 429, 821, 646, 995, 990, 215, 673, 200, 444, 339, 735, 358, 496, 238, 700, 412, 645, 442, 779, 919, 946, 500,
 574, 939, 143, 234, 48, 988, 792, 517, 315, 717, 830, 305, 628, 706, 768, 107, 625, 106, 713, 719, 592, 417, 635, 540, 335, 670, 787, 423, 925, 142, 497, 165, 326, 922, 323, 851, 552, 112, 309, 598,
 348, 562, 844, 157, 263, 357, 682, 191, 162, 160, 739, 266, 436, 425, 551, 492, 319, 155, 597, 283, 662, 864, 118, 680, 220, 402, 340, 475, 822, 769, 845, 411, 125, 996, 656, 777, 16, 383, 916, 909,
 260, 153, 166, 23, 535, 696, 797, 473, 390, 241, 984, 288, 618, 695, 205, 130, 896, 5, 211, 613, 57, 332, 398, 39, 145, 173, 362, 72, 421, 393, 694, 612, 498, 923, 991, 37, 833, 762, 770, 892, 488,
 928, 25, 462, 873, 70, 366, 38, 767, 433, 549, 507, 782, 11, 180, 943, 111, 453, 539, 688, 608, 352, 219, 312, 489, 521, 129, 747, 43, 712, 667, 979, 42, 257, 262, 426, 754, 715, 434, 953, 172, 379,
 204, 878, 405, 370, 956, 573, 935, 248, 477, 404, 536, 306, 622, 893, 920, 766, 789, 619, 594, 156, 82, 819, 197, 665, 320, 697, 287, 138, 891, 229, 275, 334, 227, 711, 911, 865, 961, 41, 237, 181,
 53, 626, 176, 814, 575, 687, 59, 585, 101, 758, 455, 294, 826, 962, 627, 317, 774, 126, 794, 841, 843, 753, 448, 817, 439, 123, 86, 374, 457, 345, 790, 877, 161, 965, 616, 20, 671, 350, 364, 570,
 868, 268, 116, 947, 752, 663, 691, 338, 255, 647, 449, 512, 94, 937, 914, 78, 980, 394, 224, 104, 451, 905, 273, 226, 418, 677, 563, 661, 528, 514, 734, 353, 569, 882, 363, 90, 276, 144, 603, 351,
 225, 258, 533, 823, 2, 456, 272, 974, 295, 87, 368, 724, 707, 467, 576, 761, 542, 571, 604, 578, 212, 621, 999, 547, 624, 857, 1000, 476, 347, 564, 246]
Compares:  22023
Swaps:  4337
Sorted array:  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48,
 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98,
  99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138,
 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177,
 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217,
 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256,
 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296,
 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335,
 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375,
 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414,
 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454,
 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493,
 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533,
 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572,
 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612,
 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651,
 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691,
 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730,
 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770,
 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809,
 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849,
 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888,
 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928,
 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967,
968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000]
```

Рисунок 3.9.2.2 – Сортування масиву на 1000 елементів

3.10 Тестування алгоритму

3.10.1 Часові характеристики оцінювання

В таблиці 3.2 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування гребінцем для масивів різної розмірності, коли масив містить упорядковану послідовність елементів.

Таблиця 3.2 – Характеристики оцінювання алгоритму сортування гребінцем для упорядкованої послідовності елементів у масиві

Розмірність масиву	Число порівнянь	Число перестановок
10	39	0
100	1 243	0
1000	22 500	0
5000	144 200	0
10000	350 343	0
20000	720 000	0
50000	1 999 922	0

В таблиці 3.3 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 – Характеристики оцінювання алгоритму сортування гребінцем для зворотно упорядкованої послідовності елементів у масиві.

Розмірність масиву	Число порівнянь	Число перестановок
10	39	7
100	1 243	100
1000	22 500	1 500
5000	144 200	9 023
10000	350 343	19 001
20000	720 000	39 564
50000	1 999 922	101 223

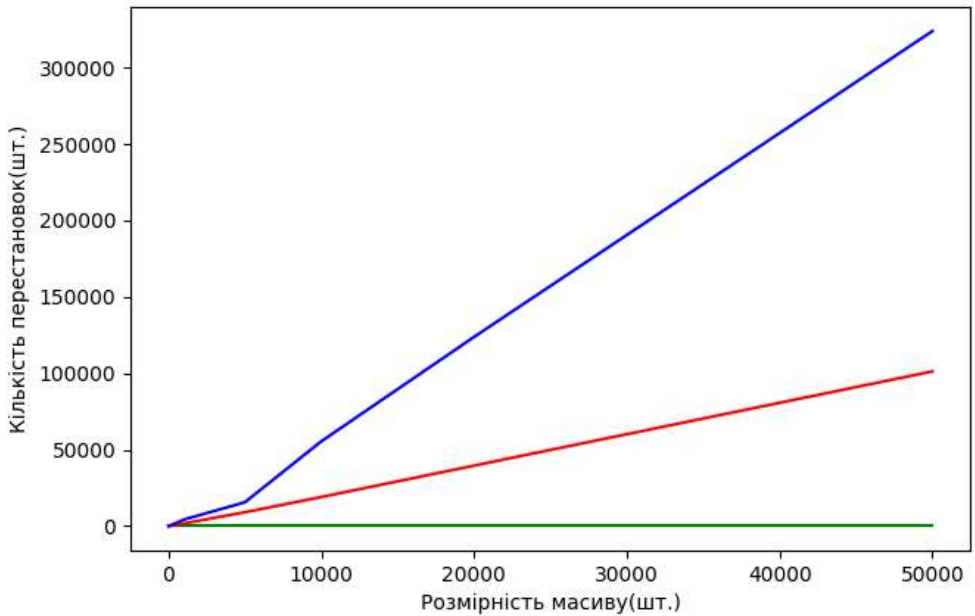
У таблиці 3.4 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування гребінцем для масивів різної розмірності, масиви містять випадкову послідовність елементів.

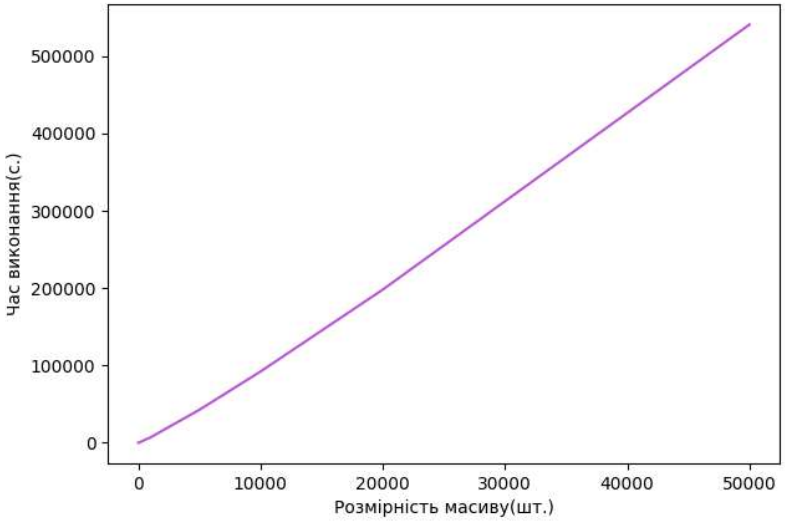
Таблиця 3.4 – Характеристика оцінювання алгоритму сортування гребінцем для випадкової послідовності елементів у масиві.

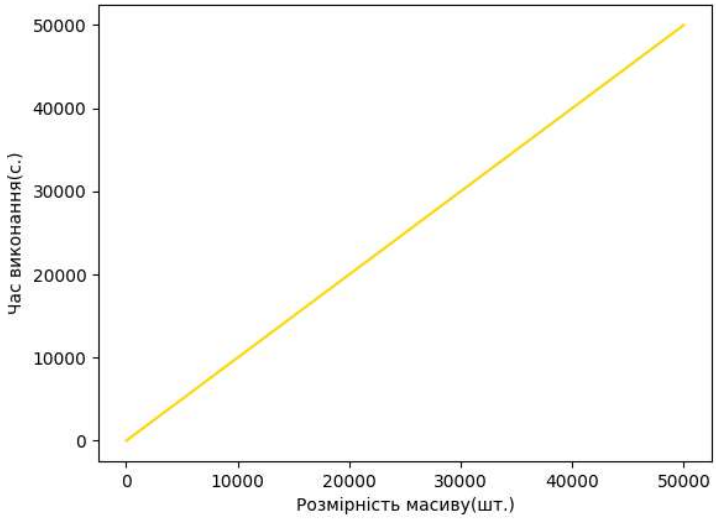
Розмірність масиву	Число порівнянь	Число перестановок
10	39	9
100	1 243	250
1000	22 500	4 222
5000	144 200	15 564
10000	350 343	55 453
20000	720 000	123 788
50000	1 999 922	324 000

3.10.2 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.







ВИСНОВОК

При виконанні даної лабораторної роботи я вивчив основні методи аналізу обчислювальної складності алгоритмів внутрішнього сортування і оцінив поріг їх ефективності.

В результаті отримав два алгоритми сортування а саме сортування бульбашкою та сортування гребінцем. Дослідив властивості кожного з цих алгоритмів. Алгоритм сортування гребінцем є значно швидшим за алгоритм сортування бульбашкою.