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. Дзюбенко Даниїл Дмитрович*
(шифр, прізвище, ім'я, по батькові)

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

Київ 2022

ЗМІСТ

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.А Алгоритм сортування бульбашкою

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 цього алгоритму потрібно мати базові знання про структури даних

3.2 Псевдокод алгоритму

Підпрограма sort_bubble(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]

То

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

Найкращий випадок: O(n)

Найгірший випадок: $O(n^2)$

Середній випадок: $O(n^2)$

3.4 Програмна реалізація алгоритму

3.4.1 Вихідний код

```
import random

def sort_bubble(arr):
    flag = True
    numCompare = 0
    numSwap = 0
    for i in range(len(arr)):
        for j in range(len(arr) - i -1):
            numCompare += 1
            if arr[j] > arr[j+1]:
                temp = arr[j]
                arr[j] = arr[j+1]
                arr[j+1] = temp
                flag = False
                numSwap += 1
        if flag:
            print("Array is already sorted!")
            break
    print("Number of compares:", numCompare)
    print("Number of swaps:", numSwap)
    return arr

n = 10

random_arr = random.sample(range(1, n+1), n)
print('Basic array:', random_arr)

sorted_arr = sort_bubble(random_arr)
print('New array:', sorted_arr)
```

3.4.2 Приклад роботи

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

```
F:\Danil\Study\Алгоритмы\sem2\lab1\venv\Scripts\python.exe F:\Danil\Study\Алгоритмы\sem2\lab1\venv\bubbleSort.py
Basic array: [13, 98, 93, 23, 63, 75, 79, 91, 87, 29, 31, 12, 26, 66, 11, 8, 49, 19, 67, 37, 90, 65, 41, 78, 92, 22, 50, 47, 33, 58, 81, 10, 76, 4, 80, 34, 21, 24, 27, 42, 30, 71, 20, 40, 39, 77, 69, 51, 2, 83, 48, 99, 7,
 88, 70, 100, 56, 16, 95, 5, 25, 54, 73, 44, 84, 97, 14, 43, 96, 9, 52, 94, 61, 36, 82, 89, 57, 55, 64, 68, 15, 3, 85, 53, 46, 17, 35, 1, 59, 74, 28, 18, 62, 6, 72, 38, 60, 86, 32, 45]
New 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]

Process finished with exit code 0
```

Рисунок 3.1 – Сортування масиву на 100 елементів

```
F:\Danil\Study\Алгоритмы\sem2\lab1\venv\Scripts\python.exe F:\Danil\Study\Алгоритмы\sem2\lab1\venv\bubbleSort.py
Basic array: [402, 966, 383, 49, 696, 267, 306, 803, 254, 91, 218, 332, 643, 947, 938, 924, 972, 351, 772, 992, 855, 510, 501, 392, 752, 592, 831, 454, 297, 406, 113, 783, 764, 333, 149, 270, 57, 748, 487, 358, 844, 261,
 393, 152, 224, 115, 89, 385, 878, 412, 704, 679, 759, 13, 616, 17, 882, 397, 169, 542, 996, 409, 260, 512, 652, 94, 773, 273, 322, 48, 661, 248, 66, 585, 632, 589, 735, 867, 808, 605, 673, 747, 776, 816, 784, 959, 484, 80,
  530, 883, 998, 324, 36, 662, 599, 866, 415, 443, 405, 295, 435, 492, 644, 919, 767, 979, 523, 323, 567, 873, 790, 629, 318, 612, 173, 536, 79, 182, 161, 521, 914, 862, 962, 249, 865, 117, 101, 985, 34, 806, 796, 631, 317,
  370, 166, 709, 349, 686, 478, 517, 981, 623, 626, 572, 54, 421, 851, 651, 970, 713, 464, 893, 532, 287, 954, 167, 477, 336, 473, 426, 83, 762, 835, 698, 50, 833, 356, 485, 253, 562, 577, 327, 650, 432, 195, 65, 714, 371,
 871, 785, 897, 407, 569, 178, 502, 198, 257, 427, 362, 394, 769, 514, 856, 190, 904, 467, 469, 905, 930, 727, 390, 810, 128, 876, 46, 649, 787, 664, 237, 681, 840, 388, 107, 489, 326, 899, 42, 925, 125, 389, 886, 449, 483,
  148, 252, 410, 376, 359, 931, 1, 987, 838, 901, 743, 566, 82, 795, 734, 44, 559, 729, 460, 602, 957, 609, 750, 138, 374, 201, 923, 212, 215, 633, 531, 210, 124, 895, 104, 158, 229, 654, 663, 69, 155, 320, 903, 408, 861,
 28, 864, 830, 537, 21, 90, 872, 705, 965, 418, 2, 177, 461, 500, 387, 453, 365, 913, 368, 296, 525, 859, 920, 381, 918, 439, 766, 377, 216, 153, 422, 292, 286, 939, 869, 202, 159, 204, 587, 77, 311, 822, 165, 307, 303,
 236, 824, 122, 710, 820, 739, 786, 811, 345, 325, 598, 701, 272, 5, 545, 319, 264, 244, 32, 330, 31, 657, 220, 38, 191, 765, 70, 234, 96, 975, 112, 429, 538, 928, 37, 640, 353, 850, 185, 219, 707, 425, 563, 993, 479, 95,
 76, 64, 942, 829, 67, 289, 888, 106, 551, 916, 507, 258, 476, 900, 363, 885, 986, 222, 671, 950, 877, 441, 935, 308, 139, 594, 977, 40, 41, 749, 603, 269, 768, 187, 92, 284, 736, 293, 288, 760, 604, 23, 526, 533, 591, 135,
  984, 952, 809, 68, 961, 807, 309, 100, 9, 881, 223, 278, 908, 189, 458, 114, 299, 622, 800, 4, 554, 741, 437, 697, 693, 225, 164, 160, 378, 615, 994, 51, 250, 669, 706, 624, 108, 848, 98, 209, 103, 573, 659, 130, 534,
 586, 630, 863, 12, 666, 316, 922, 890, 583, 413, 493, 97, 294, 140, 228, 744, 879, 715, 61, 794, 120, 20, 482, 146, 558, 724, 597, 33, 369, 619, 367, 880, 193, 184, 731, 259, 801, 911, 263, 262, 123, 613, 141, 581, 143,
 199, 335, 568, 396, 126, 88, 277, 438, 456, 45, 300, 470, 722, 775, 611, 1000, 18, 416, 917, 6, 109, 593, 450, 645, 451, 59, 251, 617, 788, 670, 676, 842, 841, 230, 14, 357, 85, 81, 504, 634, 58, 874, 499, 700, 540, 571,
 804, 399, 181, 132, 955, 144, 647, 208, 417, 197, 400, 902, 118, 245, 35, 474, 133, 172, 355, 574, 403, 621, 99, 395, 552, 27, 348, 194, 188, 610, 39, 543, 638, 382, 10, 497, 695, 313, 774, 200, 528, 22, 596, 754, 751,
 440, 546, 78, 847, 828, 834, 678, 560, 180, 459, 341, 728, 282, 147, 8, 231, 620, 689, 601, 205, 781, 15, 857, 372, 738, 266, 496, 242, 433, 475, 444, 756, 719, 852, 854, 580, 940, 547, 956, 969, 805, 990, 758, 971, 291,
 718, 298, 508, 642, 802, 812, 310, 590, 912, 680, 25, 660, 906, 404, 991, 174, 777, 448, 56, 285, 936, 929, 217, 16, 606, 726, 424, 468, 281, 723, 960, 826, 364, 951, 328, 887, 818, 243, 315, 690, 150, 780, 674, 555, 896,
 793, 770, 518, 206, 203, 687, 875, 338, 668, 271, 846, 711, 136, 672, 471, 745, 595, 516, 175, 491, 283, 910, 352, 819, 84, 490, 725, 494, 235, 716, 430, 312, 274, 556, 958, 60, 614, 884, 921, 600, 655, 733, 268, 527, 789,
  428, 233, 561, 265, 111, 241, 379, 509, 692, 436, 142, 607, 186, 973, 825, 799, 933, 279, 868, 684, 162, 522, 255, 576, 314, 703, 411, 813, 742, 635, 791, 550, 354, 137, 637, 455, 682, 646, 827, 541, 503, 419, 93, 944,
 520, 588, 964, 163, 339, 401, 915, 53, 214, 457, 691, 836, 384, 892, 446, 71, 62, 814, 853, 74, 721, 712, 688, 196, 832, 472, 685, 648, 239, 102, 891, 849, 505, 839, 350, 380, 608, 845, 740, 442, 982, 565, 967, 179, 240,
 110, 549, 553, 717, 375, 889, 211, 909, 276, 506, 949, 761, 495, 515, 771, 360, 423, 452, 129, 980, 837, 373, 420, 151, 927, 434, 342, 753, 134, 934, 823, 524, 329, 978, 694, 997, 974, 119, 226, 304, 667, 999, 677, 131,
 730, 968, 481, 302, 564, 797, 232, 463, 675, 24, 321, 557, 337, 870, 683, 465, 168, 778, 183, 618, 582, 343, 539, 943, 987, 720, 665, 221, 926, 511, 361, 290, 344, 948, 575, 953, 480, 821, 398, 658, 305, 334, 579, 55, 858,
  3, 653, 280, 798, 488, 445, 498, 347, 746, 529, 976, 779, 641, 815, 29, 301, 52, 937, 170, 331, 156, 157, 86, 47, 87, 628, 176, 627, 73, 988, 43, 246, 702, 247, 548, 7, 171, 898, 207, 989, 213, 11, 26, 817, 121, 256, 30,
 544, 792, 932, 995, 519, 391, 639, 192, 860, 366, 941, 535, 238, 275, 513, 346, 708, 699, 946, 431, 227, 755, 625, 466, 414, 570, 757, 963, 584, 127, 75, 447, 656, 116, 19, 945, 763, 578, 782, 386, 843, 462, 894, 732, 105,
  145, 636, 737, 154, 72, 486, 63, 340, 983]

New 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]

Process finished with exit code 0
```

Рисунок 3.2 – Сортування масиву на 1000 елементів

3.5 Тестування алгоритму

```python
import random

def sort_bubble(arr):
    numCompare = 0
    numSwap = 0
    for i in range(len(arr)):
        flag = True
        for j in range(len(arr) - i -1):
            numCompare += 1
            if arr[j] > arr[j+1]:
                temp = arr[j]
                arr[j] = arr[j+1]
                arr[j+1] = temp
                flag = False
                numSwap += 1
        if flag:
            # print("Array is already sorted!")
            break
    print("Number of compares:", numCompare)
    print("Number of swaps:", numSwap)
    return arr

n = 20

random_arr = random.sample(range(1, n+1), n)
print('Basic array:', random_arr)

sorted_arr = sort_bubble(random_arr)
print('New array:', sorted_arr)
```

```
F:\Danil\Study\Алгоритмы\sem2\lab1\venv\Scripts\python.exe F:\Danil\Study\Алгоритмы\sem2\lab1\venv\bubbleSort.py
Basic array: [2, 13, 18, 5, 3, 14, 20, 10, 9, 17, 12, 19, 6, 1, 15, 11, 4, 16, 8, 7]
Number of compares: 175
Number of swaps: 99
New 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
```

```python
def sort_bubble(arr):
    numCompare = 0
    numSwap = 0
    for i in range(len(arr)):
        flag = True
        for j in range(len(arr) - i -1):
            numCompare += 1
            if arr[j] > arr[j+1]:
                temp = arr[j]
                arr[j] = arr[j+1]
                arr[j+1] = temp
                flag = False
                numSwap += 1
        if flag:
            # print("Array is already sorted!")
            break
    print("Number of compares:", numCompare)
    print("Number of swaps:", numSwap)
    return arr

n = 15

random_arr = random.sample(range(1, n+1), n)
print('Basic array:', random_arr)

sorted_arr = sort_bubble(random_arr)
print('New array:', sorted_arr)
```

```
Basic array: [2, 14, 10, 4, 1, 6, 5, 13, 7, 15, 9, 11, 3, 12, 8]
Number of compares: 99
Number of swaps: 43
New array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

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	4 950	4 950
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.

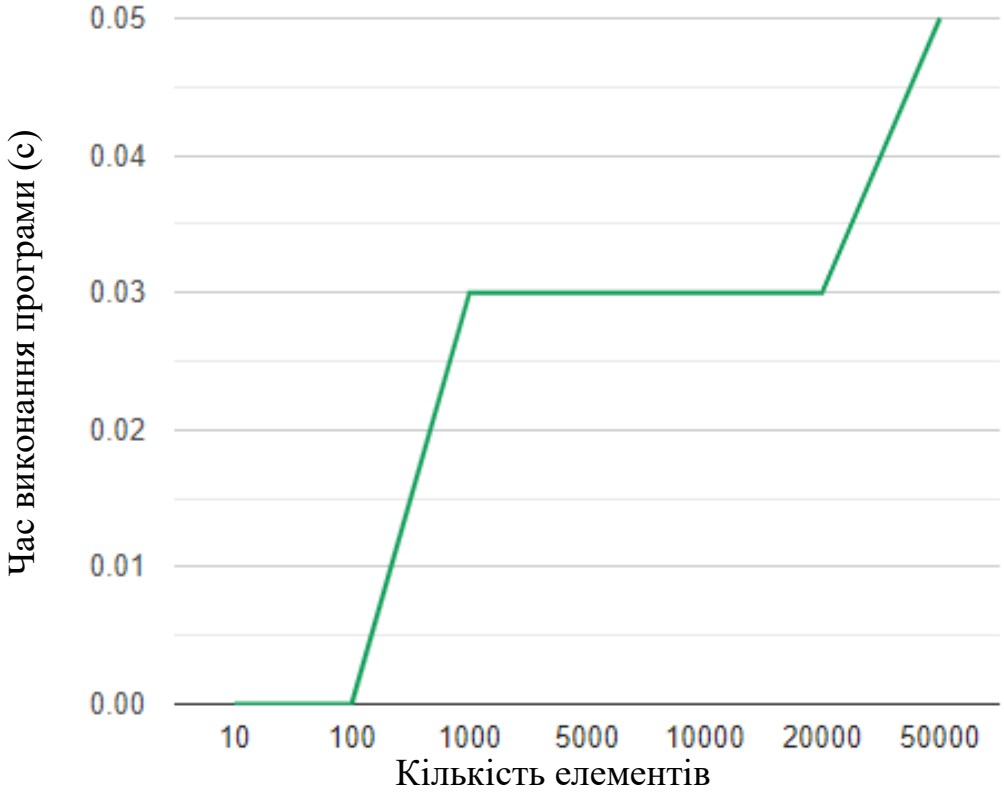
Розмірність масиву	Число порівнянь	Число перестановок
10	35	15
100	4 544	2 141
1000	497 609	254 313
5000	12 488 455	6 287 843
10000	49 993 725	24 915 720
20000	199 975 465	99 872 304
50000	1 249 970 905	623 960 244

3.5.2 Графіки залежності часових характеристик оцінювання від розмірності масиву

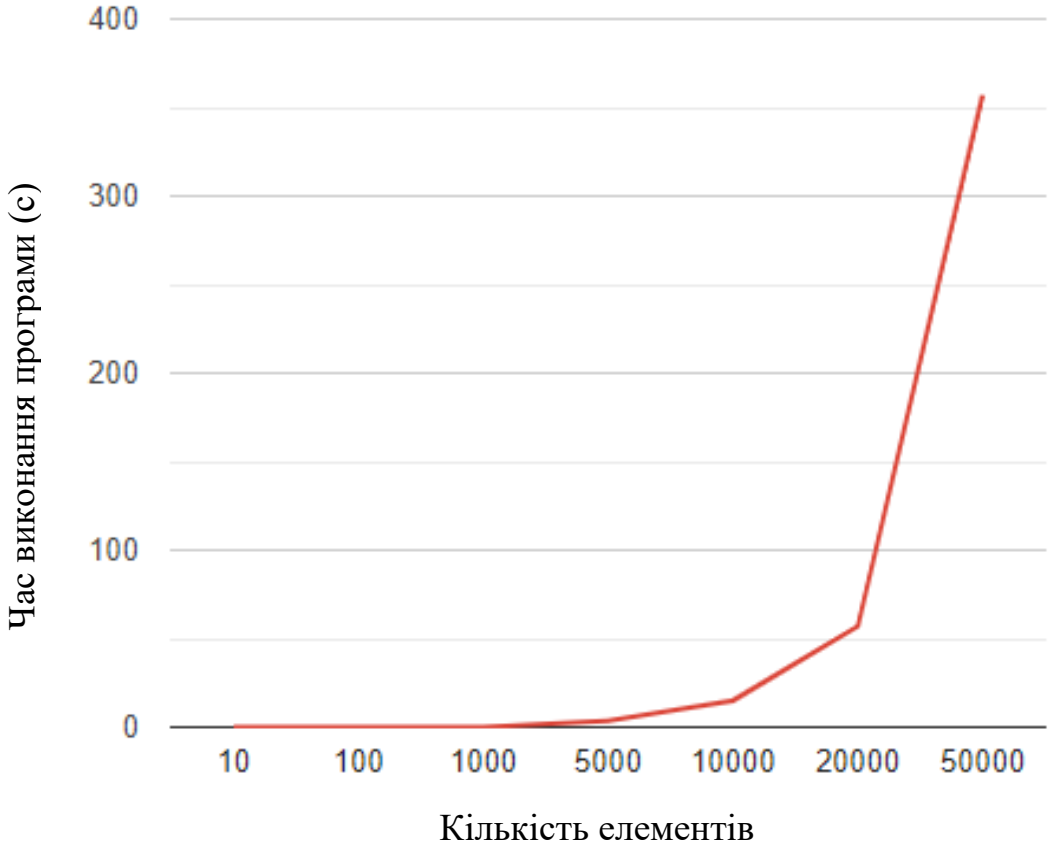
На рисунку 3.3 показані графіки залежності часових характеристик оцінювання від розмірності масиву для випадків, коли масиви містять упорядковану послідовність елементів (зелений графік), коли масиви містять зворотно упорядковану послідовність елементів (червоний графік), коли масиви містять випадкову 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

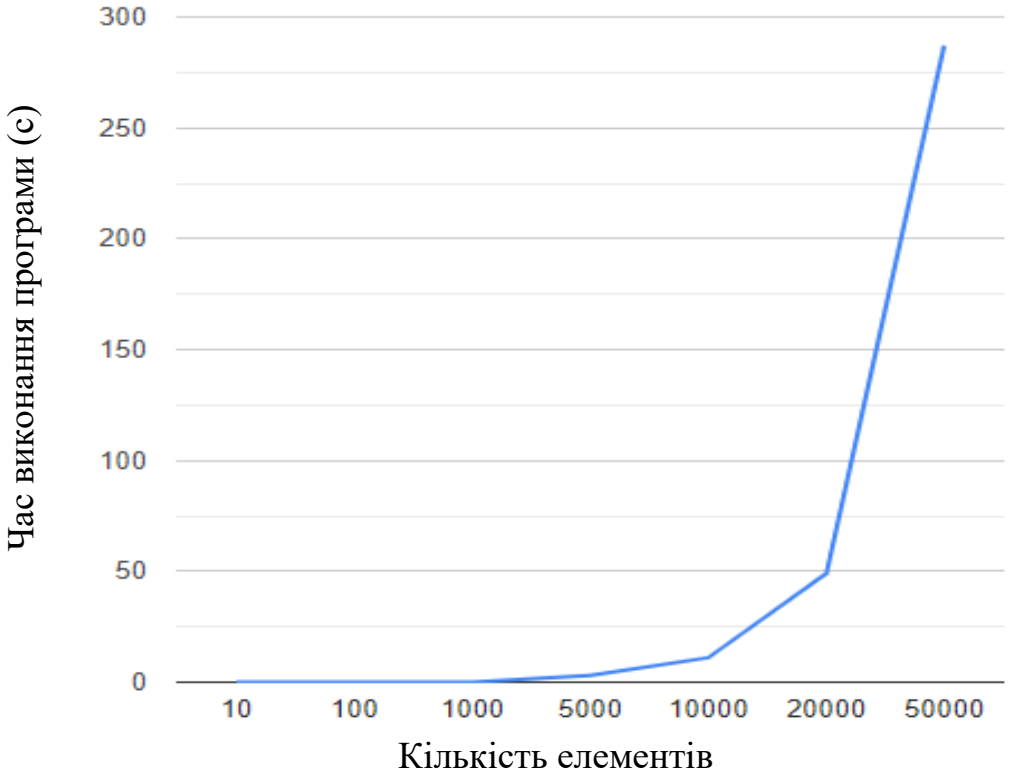
Графік залежності часових характеристик оцінювання від кількості упорядкованої послідовності елементів в масиві



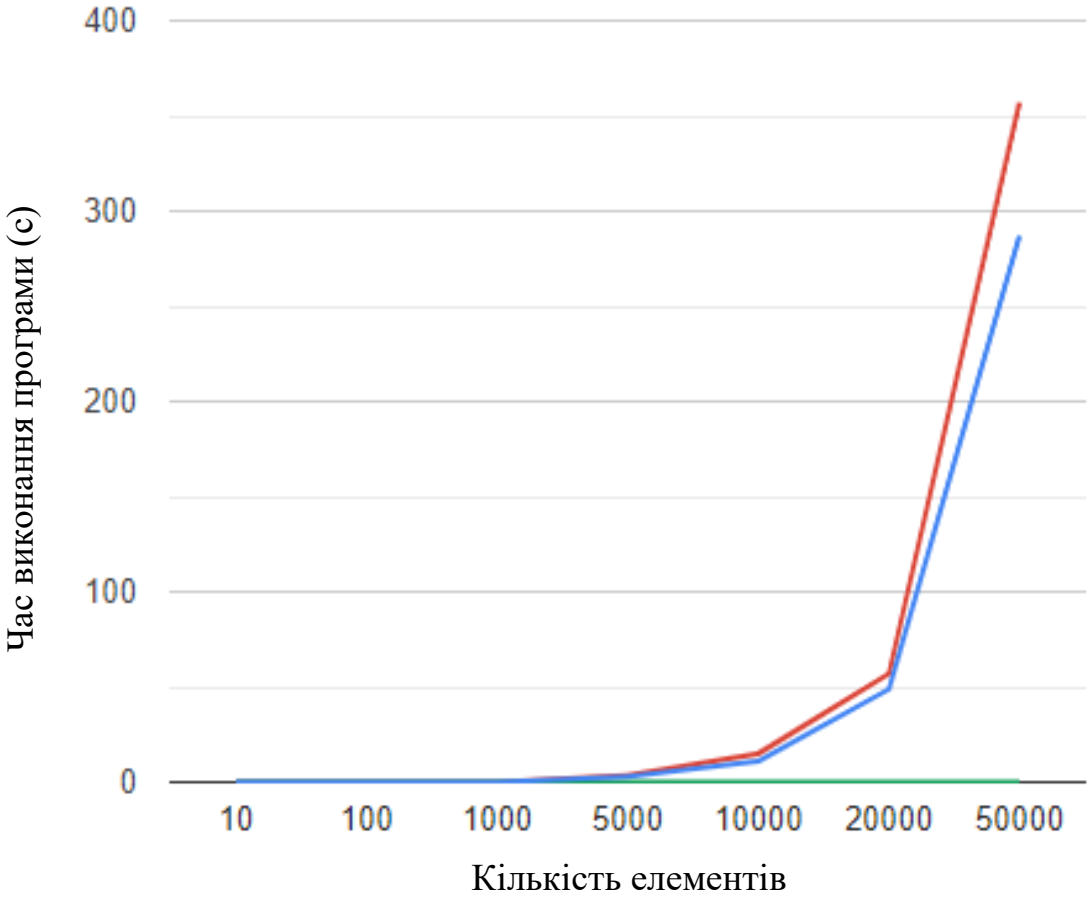
Графік залежності часових характеристик оцінювання від кількості зворотно упорядкованої послідовності елементів в масиві

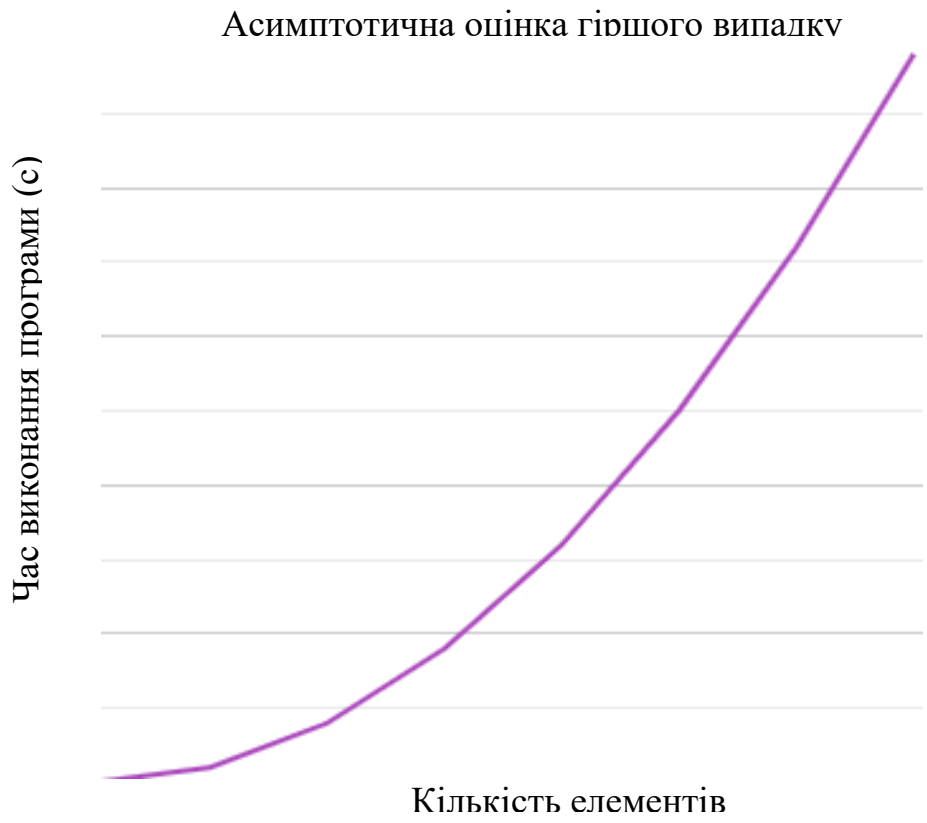


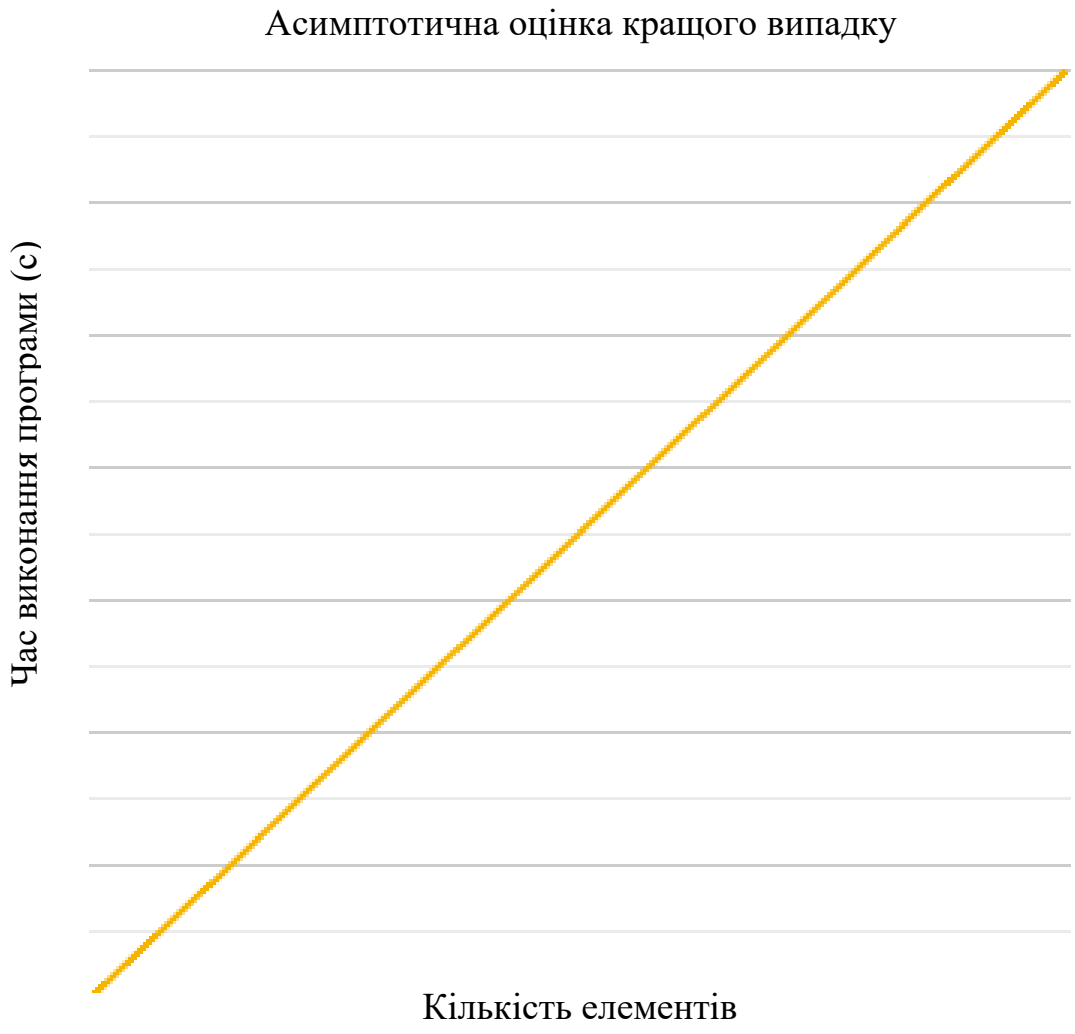
Графік залежності часових характеристик оцінювання від кількості випадкової послідовності елементів в масиві



Усі графіки разом




Асимптотична оцінка гіршого випадку
Час виконання програми (с)
Кількість елементів


Асимптотична оцінка кращого випадку
Час виконання програми (с)
Кількість елементів

3.Б 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 гребінцем**
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 цього алгоритму потрібно мати базові знання про структури даних

3.7 Псевдокод алгоритму

Підпрограма sort_bubble(arr):
Початок
  step = len(arr) - 1
  Поки step >= 1 повторити
    Повторити для i від 0 до len(arr) – step
      Якщо arr[i] > arr[i+step] **То**
        temp = arr[i]
        arr[i] = arr[i+step]
        arr[i+step] = temp
    Все повторити
  Все повторити
Кінець

3.8 Аналіз часової складності

Найкращий випадок: O(n*log(n))

Найгірший випадок: $O(n^2)$

Середній випадок: $\Omega(n^2/2^p)$

3.9 Програмна реалізація алгоритму

3.9.1 Вихідний код

```
import random

def sort_comb(arr):
    numCompare = 0
    numSwap = 0
    step = len(arr) - 1
    while step >= 1:
        for i in range(len(arr) - step):
            numCompare += 1
            if arr[i] > arr[i + step]:
                b = arr[i]
                arr[i] = arr[i+step]
                arr[i+step] = b
                numSwap += 1
        step = int(step / 1.247)
    print("Number of compares:", numCompare)
    print("Number of swaps:", numSwap)
    return arr

n = 10

random_arr = random.sample(range(1, n+1), n)
print('Basic array:', random_arr)

sorted_arr = sort_comb(random_arr)
print('New array:', sorted_arr)
```

3.9.2 Приклад роботи

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

```
F:\Danil\Study\Алгоритмы\sem2\lab1\venv\Scripts\python.exe F:/Danil/Study/Алгоритмы/sem2/lab1/venv/combSort.py
Basic array: [59, 46, 21, 51, 32, 27, 34, 6, 86, 2, 41, 83, 56, 9, 63, 73, 17, 72, 74, 3, 25, 95, 47, 23, 1, 55, 4, 38,
 10, 60, 16, 64, 92, 33, 43, 99, 97, 78, 82, 19, 28, 70, 67, 12, 61, 20, 94, 8, 54, 77, 36, 68, 42, 98, 39, 91, 79, 76,
 58, 87, 44, 22, 30, 50, 13, 85, 75, 100, 81, 24, 29, 96, 49, 65, 14, 80, 52, 37, 89, 26, 57, 40, 18, 15, 31, 84, 5, 45,
  66, 93, 90, 62, 35, 53, 88, 69, 11, 48, 71, 7]
Number of compares: 1233
Number of swaps: 222
New 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]

Process finished with exit code 0
```

Рисунок 3.1 – Сортування масиву на 100 елементів

```
F:\Danil\Study\Алгоритмы\sem2\lab1\venv\Scripts\python.exe F:/Danil/Study/Алгоритмы/sem2/lab1/venv/combSort.py
Basic array: [622, 377, 40, 878, 60, 750, 349, 109, 127, 782, 94, 643, 827, 296, 521, 863, 802, 302, 275, 799, 156, 781, 174, 626, 141, 986, 608, 734, 338, 753, 539, 910, 8, 810, 426, 63, 654, 216, 627, 451, 232, 30, 562,
 595, 831, 278, 95, 118, 39, 476, 766, 458, 247, 769, 982, 834, 279, 319, 495, 217, 475, 804, 465, 66, 433, 945, 814, 990, 381, 740, 712, 629, 615, 970, 709, 676, 254, 147, 395, 983, 652, 191, 739, 168, 290, 527, 573, 100,
 151, 625, 158, 533, 711, 698, 669, 345, 478, 75, 874, 772, 673, 901, 190, 801, 23, 968, 316, 954, 998, 76, 592, 166, 466, 576, 773, 43, 630, 24, 136, 817, 920, 373, 70, 110, 647, 516, 257, 526, 431, 843, 396, 705, 795,
 713, 642, 537, 468, 84, 803, 746, 529, 439, 721, 443, 660, 71, 566, 348, 926, 383, 832, 262, 481, 202, 444, 16, 357, 860, 452, 848, 507, 830, 124, 492, 664, 500, 423, 541, 749, 758, 114, 938, 167, 412, 273, 303, 922, 20,
 708, 361, 107, 419, 581, 281, 224, 64, 977, 742, 881, 374, 379, 157, 866, 482, 251, 336, 995, 992, 227, 600, 780, 535, 610, 984, 856, 790, 682, 14, 212, 207, 858, 840, 670, 89, 899, 56, 978, 463, 161, 173, 668, 545, 415,
 650, 508, 154, 45, 163, 735, 988, 277, 297, 453, 646, 549, 583, 177, 38, 128, 185, 718, 577, 499, 105, 528, 960, 966, 967, 366, 553, 683, 714, 430, 376, 601, 244, 509, 693, 946, 28, 842, 590, 903, 852, 811, 142, 588, 850,
 555, 565, 747, 548, 420, 53, 505, 820, 42, 981, 884, 963, 394, 334, 77, 311, 794, 74, 241, 137, 702, 352, 386, 293, 896, 400, 950, 808, 710, 226, 432, 371, 21, 65, 328, 853, 805, 471, 36, 731, 934, 359, 919, 560, 380, 342,
  104, 947, 414, 477, 140, 591, 602, 286, 578, 949, 462, 441, 3, 835, 90, 604, 851, 636, 192, 671, 547, 329, 25, 18, 929, 219, 638, 786, 640, 942, 88, 474, 250, 760, 54, 398, 606, 822, 885, 996, 484, 748, 964, 370, 955,
 665, 659, 307, 890, 888, 97, 324, 609, 384, 179, 15, 445, 417, 372, 390, 271, 952, 180, 876, 633, 745, 92, 450, 725, 35, 321, 666, 248, 641, 298, 12, 575, 534, 101, 34, 943, 969, 833, 621, 238, 123, 552, 730, 152, 215,
 350, 318, 839, 355, 882, 696, 198, 428, 353, 959, 44, 689, 520, 436, 891, 614, 741, 895, 497, 210, 409, 2, 976, 312, 27, 582, 411, 346, 272, 260, 138, 261, 249, 470, 326, 736, 33, 153, 294, 719, 675, 589, 222, 486, 344,
 517, 425, 584, 939, 112, 871, 569, 354, 367, 764, 716, 518, 524, 169, 446, 332, 125, 148, 442, 68, 677, 701, 558, 672, 61, 972, 214, 697, 269, 838, 403, 464, 134, 655, 913, 11, 546, 187, 678, 596, 729, 333, 143, 695, 544,
 598, 648, 67, 620, 408, 707, 684, 434, 209, 645, 572, 531, 937, 948, 873, 323, 658, 985, 872, 170, 530, 999, 305, 130, 798, 775, 87, 930, 846, 887, 847, 632, 160, 26, 413, 314, 706, 809, 111, 287, 965, 206, 456, 789, 880,
 402, 703, 78, 617, 239, 818, 927, 339, 757, 900, 6, 200, 994, 768, 106, 754, 485, 181, 974, 870, 320, 784, 680, 911, 674, 593, 704, 213, 826, 195, 894, 473, 267, 542, 504, 220, 807, 435, 365, 131, 586, 624, 550, 340, 175,
 935, 845, 252, 1, 875, 690, 313, 129, 243, 916, 791, 506, 467, 914, 472, 17, 86, 245, 651, 358, 579, 905, 49, 283, 457, 317, 228, 686, 908, 440, 263, 276, 382, 119, 268, 270, 62, 282, 388, 397, 991, 93, 236, 331, 893, 536,
  259, 427, 656, 862, 857, 58, 806, 561, 828, 993, 115, 306, 140, 438, 812, 763, 483, 188, 777, 10, 774, 694, 155, 825, 956, 193, 761, 879, 982, 47, 308, 327, 378, 980, 912, 944, 315, 493, 416, 783, 551, 126, 915, 375, 841,
 574, 700, 728, 618, 46, 564, 494, 291, 337, 150, 360, 607, 649, 639, 407, 460, 662, 918, 557, 859, 637, 989, 691, 149, 513, 233, 496, 726, 459, 778, 231, 723, 623, 266, 196, 619, 823, 184, 165, 765, 79, 292, 139, 133,
 387, 32, 72, 923, 925, 454, 563, 182, 37, 567, 208, 556, 515, 230, 19, 864, 511, 422, 767, 883, 295, 330, 635, 836, 599, 519, 258, 363, 135, 792, 532, 727, 958, 737, 611, 559, 685, 634, 223, 255, 59, 391, 310, 815, 762,
 55, 7, 405, 733, 180, 751, 722, 48, 800, 849, 570, 325, 176, 5, 280, 218, 237, 144, 50, 663, 351, 347, 235, 525, 437, 159, 776, 83, 940, 22, 274, 936, 189, 837, 98, 103, 309, 797, 364, 171, 362, 264, 113, 661, 953, 892,
 653, 514, 52, 502, 928, 1000, 108, 720, 821, 122, 819, 941, 341, 299, 657, 543, 816, 587, 41, 788, 356, 957, 933, 580, 898, 399, 571, 120, 447, 931, 253, 932, 211, 203, 692, 813, 855, 448, 979, 785, 501, 225, 57, 201, 404,
  868, 393, 756, 613, 869, 424, 603, 82, 4, 889, 406, 117, 793, 96, 491, 523, 538, 104, 865, 240, 644, 585, 699, 13, 961, 861, 779, 300, 205, 628, 490, 99, 904, 429, 510, 755, 29, 421, 877, 616, 172, 924, 343, 897, 503,
 987, 787, 304, 962, 102, 162, 681, 284, 829, 724, 796, 971, 242, 194, 183, 973, 289, 132, 759, 487, 81, 844, 51, 824, 687, 229, 145, 368, 335, 594, 554, 284, 246, 744, 116, 85, 69, 886, 540, 909, 732, 392, 867, 385, 743,
 906, 322, 285, 688, 771, 80, 752, 854, 221, 917, 679, 31, 418, 199, 73, 479, 715, 605, 907, 597, 197, 997, 631, 488, 121, 951, 9, 770, 738, 522, 667, 369, 717, 234, 612, 91, 288, 449, 568, 256, 921, 498, 512, 455, 401,
 389, 178, 461, 489, 410, 265, 975, 301, 469, 480]
Number of compares: 22023
Number of swaps: 4323
```

```
New 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, 317, 316, 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, 880, 879, 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 3.2 – Сортування масиву на 1000 елементів

3.10 Тестування алгоритму

3.10.1Часові 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	1 233	0
1000	22 023	0
5000	144 834	0
10000	329 606	0
20000	719 141	0
50000	1 997 682	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	5
100	1 233	106
1000	22 023	1 498
5000	144 834	9 050
10000	329 606	18 994
20000	719 141	40 668
50000	1 997 682	110 406

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.

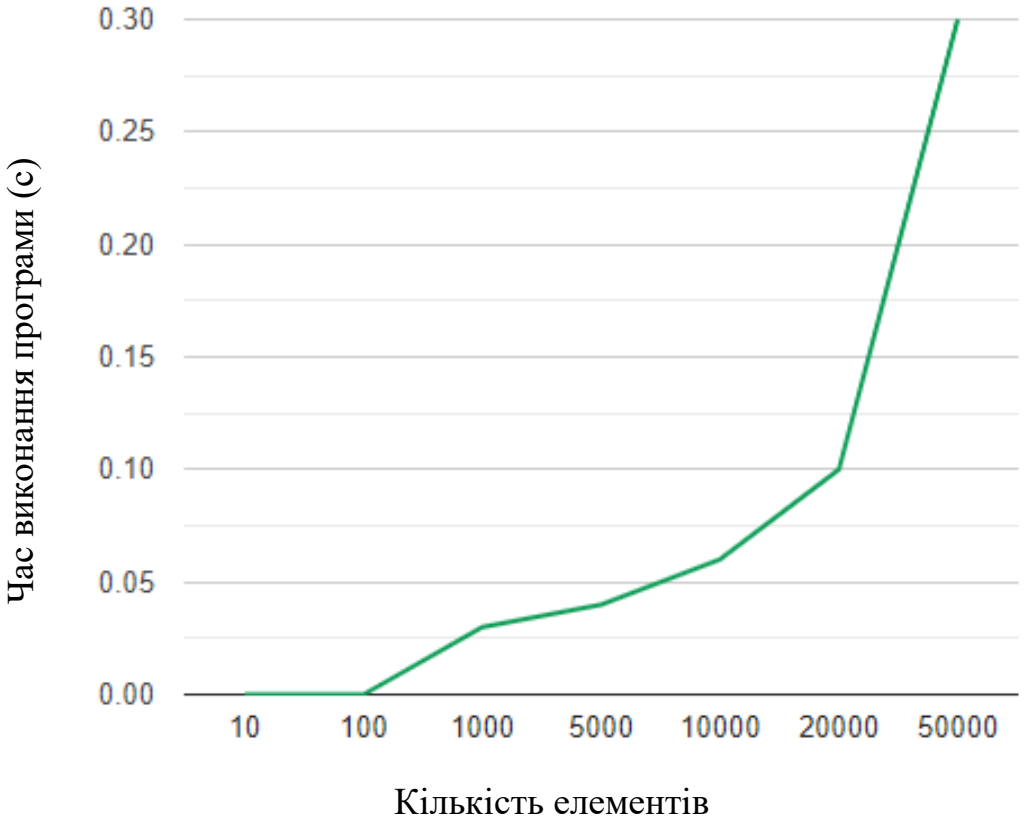
Розмірність масиву	Число порівнянь	Число перестановок
10	39	7
100	1 233	240
1000	22 023	4 335
5000	144 834	27 342
10000	329 606	60 314
20000	719 141	130 944
50000	1 997 682	367 813

3.10.2 Графіки залежності часових характеристик оцінювання від розмірності масиву

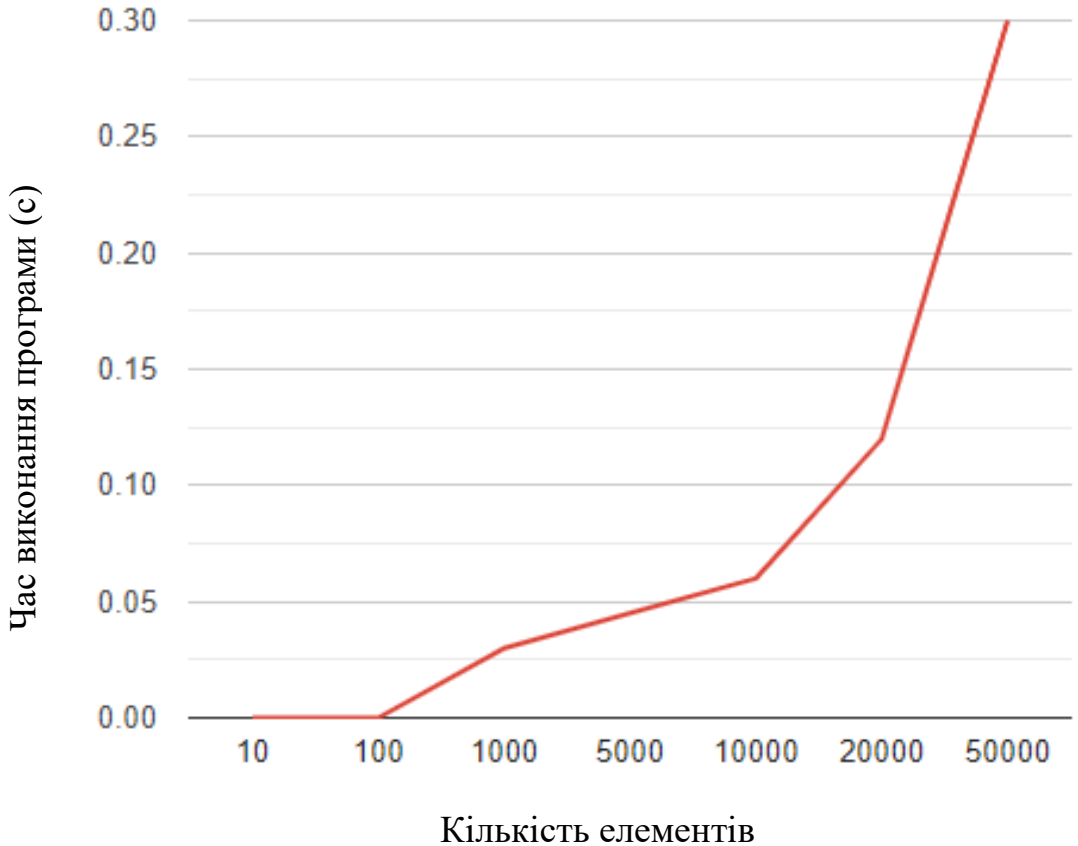
На рисунку 3.3 показані графіки залежності часових характеристик оцінювання від розмірності масиву для випадків, коли масиви містять упорядковану послідовність елементів (зелений графік), коли масиви містять зворотно упорядковану послідовність елементів (червоний графік), коли масиви містять випадкову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

Рисунок 3.3 – Графіки залежності часових характеристик оцінювання

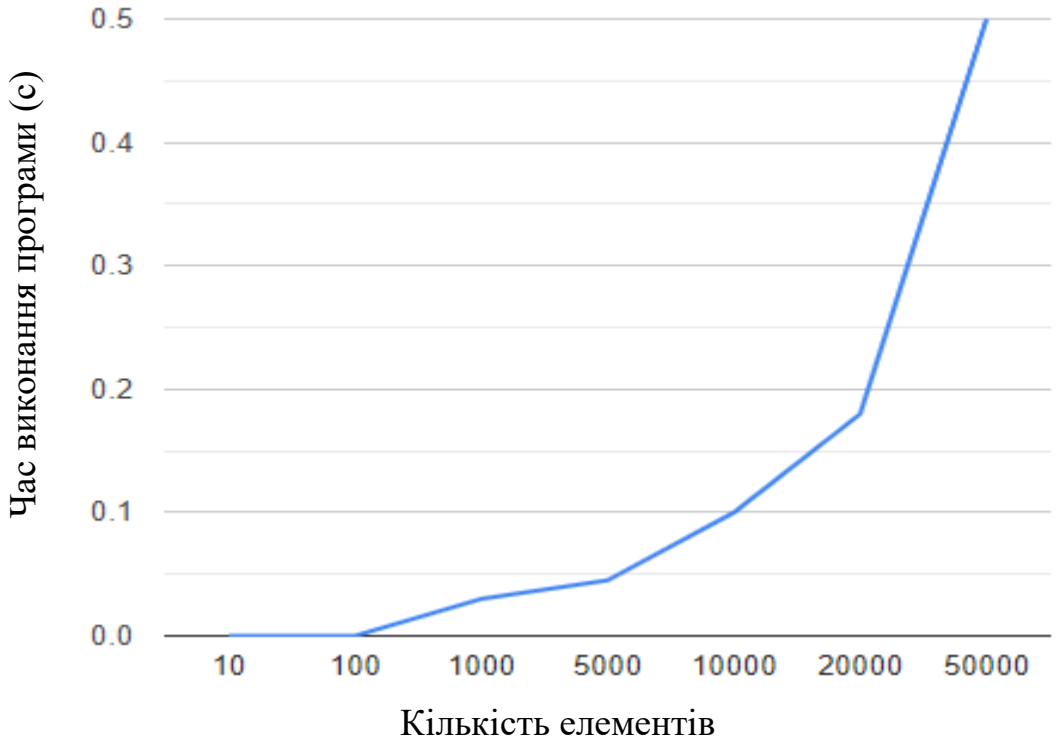
Графік залежності часових характеристик оцінювання від кількості упорядкованої послідовності елементів в масиві



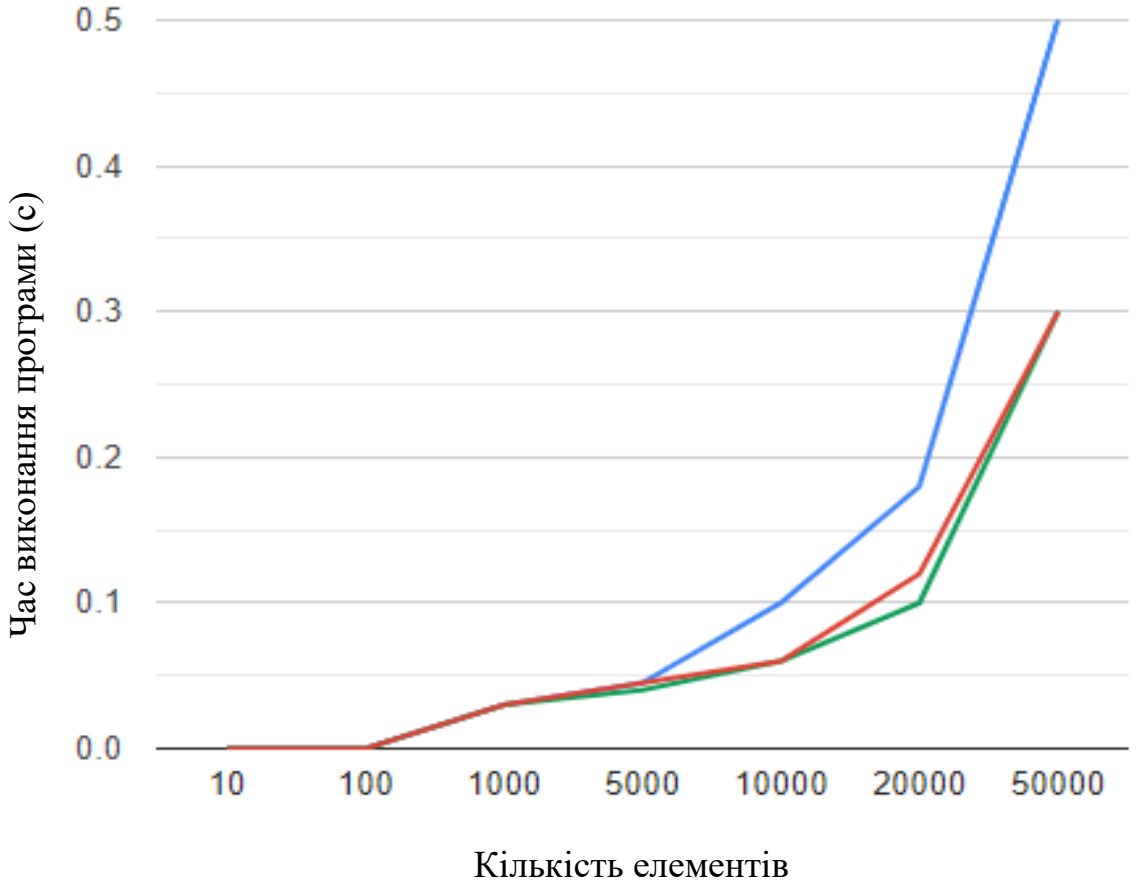
Графік залежності часових характеристик оцінювання від кількості зворотно упорядкованої послідовності елементів в масиві

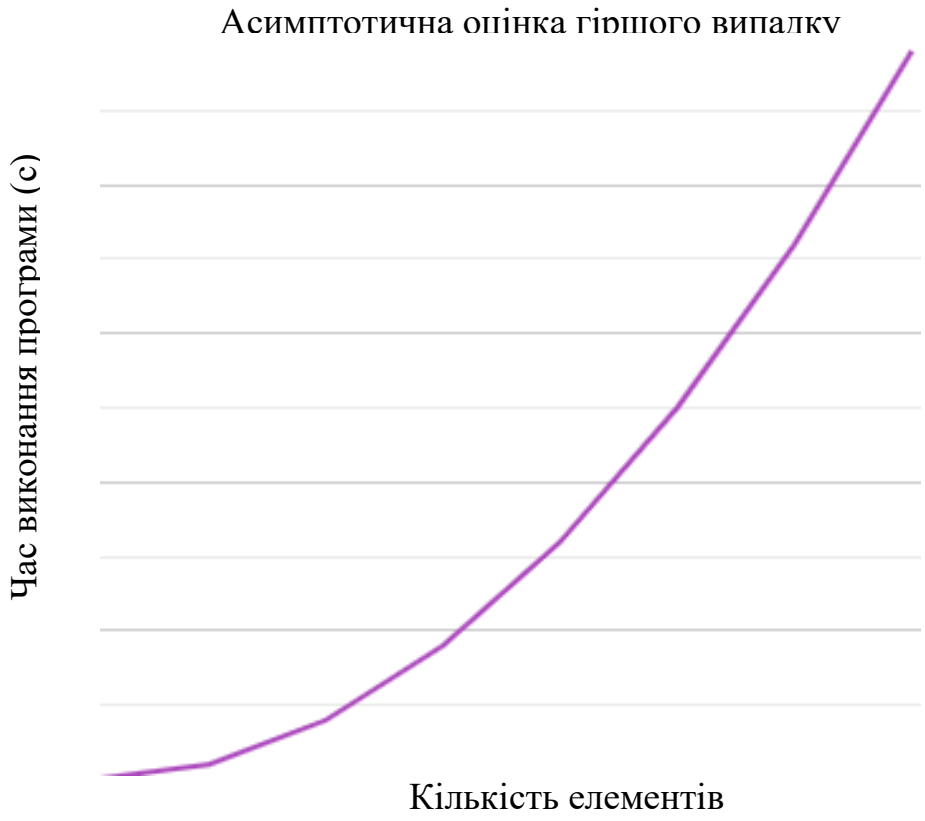


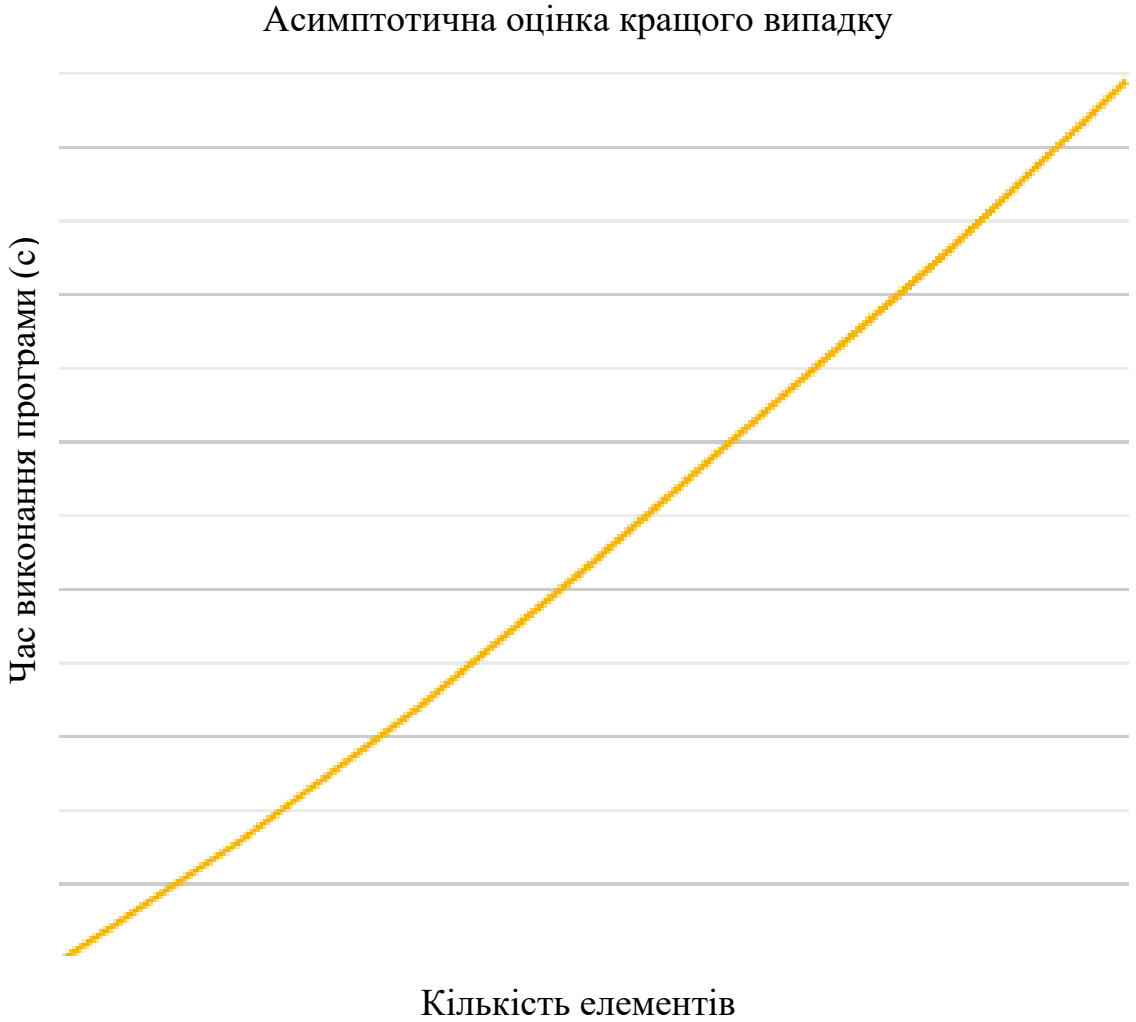
Графік залежності часових характеристик оцінювання від кількості випадкової послідовності елементів в масиві



Усі графіки разом




Асимптотична оцінка гіршого випадку
Час виконання програми (с)
Кількість елементів


Асимптотична оцінка кращого випадку
Час виконання програми (с)
Кількість елементів

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.