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

Кафедра автоматизованих систем обробки інформації і управління

Звіт

з лабораторної роботи № 6 з дисципліни
«Алгоритми та структури даних 2. Структури даних»

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

Виконав(ла) *ІП-02 Гущін Д.О.* _____
(шифр, прізвище, ім'я, по батькові)

Перевірив *Головченко М.М.* _____
(прізвище, ім'я, по батькові)

Київ 2021

ЗМІСТ

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

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

2 ЗАВДАННЯ

Згідно варіанту (таблиця 2.1), виконати аналіз алгоритму внутрішнього сортування на відповідність наступним властивостям:

- 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 узагальнений висновок з лабораторної роботи.

Таблиця 2.1 – Варіанти алгоритмів

№	**Алгоритм сортування**
1	Сортування перемішуванням
2	Сортування гребінцем («розчіскою»)
3	Сортування вибором
4	Сортування Шелла (класичне)

5	Сортування Шелла (*d* = послідовність Хіббарда)
6	Сортування Шелла (*d* = послідовність Пратта)
7	Сортування Шелла (*d* = послідовність Кнута)
8	Сортування Шелла (*d* = послідовність Седжвіка)
9	Сортування Шелла (*d* = послідовність Фібоначі)
10	Сортування Шелла ($d = (3^j - 1) \leq n, j \in N$)
11	Сортування Шелла (*d* = послідовність Гоннета и Бєза-Єтс)
12	Сортування Шелла (*d* = послідовність Токуда)
13	Сортування Шелла (*d* = послідовність Марцина Циура)
14	Швидке сортування (розбиття Ломуто)
15	Швидке сортування (розбиття Хоара)
16	Швидке сортування (медіана із трьох)
17	Швидке сортування (елементи, що повторюються)
18	Пірамідальне сортування
19	Плавне сортування
20	Інтроспективне сортування
21	Stdsort
22	Сортування за допомогою двійкового дерева
23	Швидке сортування (медіана із трьох)
24	Сортування Шелла (*d* = послідовність Хіббарда)
25	Сортування Шелла (*d* = послідовність Пратта)
26	Сортування Шелла (*d* = послідовність Кнута)
27	Сортування Шелла (*d* = послідовність Седжвіка)
28	Сортування Шелла (*d* = послідовність Фібоначі)
29	Пірамідальне сортування
30	Плавне сортування
31	Інтроспективне сортування
32	Stdsort
33	Швидке сортування (розбиття Ломуто)

34	Швидке сортування (розбиття Хоара)
35	Швидке сортування (елементи, що повторюються)

3 ВИКОНАННЯ

3.1 Аналіз алгоритму на відповідність властивостям

Аналіз алгоритму швидкого сортування на відповідність властивостям наведено в таблиці 3.1.

Таблиця 3.1 – Аналіз алгоритму на відповідність властивостям

Властивість	Швидке сортування
Стійкість	-
«Природність» поведінки (Adaptability)	-
Базуються на порівняннях	+
Необхідність в додатковій пам'яті (об'єм)	+
Необхідність в знаннях про структури даних	-

3.2 Псевдокод алгоритму

```
algorithm quicksort(A, lo, hi) is
   if lo < hi then
       p:= partition(A, lo, hi)
       quicksort(A, lo, p)
       quicksort(A, p + 1, hi)
algorithm partition(A, low, high) is
   pivot:= A[(low + high) / 2]
   i:= low
   j:= high
   loop forever

       while A[i] < pivot
              i:= i + 1
       while A[j] > pivot
              j:= j - 1
       if i >= j then
           return j
       swap A[i++] with A[j--]
```

3.3 Аналіз часової складності

Кращий випадок

Найбільш сприятливий для швидкого сортування випадок має місце, коли на кожній стадії розбивки файл ділиться на дві рівні частини. Ця обставина призводить до того, що кількість операцій порівняння, які виконуються в процесі швидкого сортування, задовольняє рекурентному співвідношенню типу "розділяй і володарюй" $C_n = 2C_{n/2} + n$, що дає загальну складність алгоритму $O(n\log_2 n)$.

Середнє

На жаль, ймовірність вибрати навмання точно середній елемент досить низька. Але припустимо, що роздільник є досить хорошим, якщо він знаходиться в центральній половині масиву, тобто в діапазоні елементів для сортування від n/4 до 3n/4. Таких досить хороших розділових елементів є досить багато, тому що половина всіх елементів розташована ближче до центру масиву, ніж до його краях. Таким чином, ймовірність вибору досить хорошого розділового елемента при кожному виборі дорівнює 1/2.

При виборі найгіршого можливого досить хорошого роздільник велика частина розділеного масиву містить 3/4 елемента. Якою буде висота h_g дерева алгоритму швидкого сортування, створеного в результаті послідовного вибору найгірших досить хороших роздільників? Найдовший шлях по цьому дереву проходить через частини розміром по n, (3/4) * n, $(3/4)^2$ * n , і так далі до 1 елемента. Скільки разів можна помножити n на 3/4, поки ми не дійдемо до 1? Так як $(3/4)^{h_g} n = 1$, то $h_g = \log_{4/3} n$.

Але тільки половина обираних довільно роздільників є досить хорошими, а іншу половину ми назвемо поганими. Найгірші з поганих роздільників по суті не зменшують розміру розділу уздовж найглибшого шляху. Найглибший шлях

від кореня вниз по типовому дереву розділів швидкого сортування, створеного за допомогою вибору довільних роздільників, проходить через приблизно однакову кількість досить хороших і поганих роздільників. Так як очікувана кількість досить хороших і поганих роздільників однаково, то погані роздільники можуть збільшити висоту дерева, щонайбільше, вдвічі, тому середня складність складе $O(n \log n)$.

Найгірший випадок

Базова програма швидкого сортування володіє певним недоліком, який полягає в тому, що вона виключно неефективна на деяких простих файлах, які можуть зустрітися на практиці. Наприклад, якщо вона застосовується для сортування файлу розміром N, який вже відсортований, то програма викличе сама себе N раз, переміщаючи за кожен виклик всього лише один елемент. В силу тільки що наведеного аргументу, число операцій порівняння, виконаних при сортуванні вже відсортованого файлу, виражається як N + (N - 1) + … + 2 + 1 = (N + 1) * N / 2, тому O(n) = n^2.

Подібна поведінка означає те, що простір пам'яті, необхідний для виконання цієї рекурсії, приблизно пропорційний N, що в разі великих файлів неприпустимо. Для великих значень n найгірший випадок може привести до вичерпання пам'яті (переповнення стека) під час роботи програми.

3.4 Програмна реалізація алгоритму

3.4.1 Вихідний код

```
def quick_sort(sequence, depth=None, lo_index=None, hi_index=None):
    global recursive
    recursive = depth
    if lo_index is None:
        lo_index = 0
    if hi_index is None:
        hi_index = len(sequence) - 1
    if lo_index >= hi_index:
        return None
    h = partition(sequence, lo_index, hi_index)
    quick_sort(sequence, depth + 1, lo_index, h - 1)
    quick_sort(sequence, depth + 1, h + 1, hi_index)


def partition(sequence, lo_index, hi_index):
    support_element = sequence[lo_index]
    i = lo_index + 1
```

```
    j = hi_index
    while True:
        while i < hi_index and sequence[i] < support_element:
            i += 1
        while sequence[j] > support_element:
            j -= 1
        if i >= j:
            break
        sequence[i], sequence[j] = sequence[j], sequence[i]
        i += 1
        j -= 1
    sequence[lo_index], sequence[j] = sequence[j], sequence[lo_index]
    return j
```

3.4.2 Приклад роботи

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

```
Initial list:  [41, 53, 9, 69, 9, 85, 68, 59, 66, 26, 73, 18, 6, 7, 62, 36, 32, 93, 86, 8, 34, 32, 61, 31, 84, 28, 7, 18, 54, 85, 17, 77, 69, 66, 29, 92, 47, 72, 15, 98, 66, 13, 76, 35, 91, 32, 5, 23, 49, 47, 21, 9, 62, 76, 96, 19, 85, 80, 31, 62, 54, 56, 49, 5, 27, 0, 57, 20, 72, 36, 53, 24, 23, 85, 30, 21, 21, 1, 44, 1, 45, 72, 86, 74, 72, 54, 91, 34, 19, 85, 80, 13, 90, 16, 82, 51, 15, 32, 36, 55]
Result list:  [0, 1, 1, 5, 5, 6, 7, 7, 8, 9, 9, 9, 13, 13, 15, 15, 16, 17, 18, 18, 19, 19, 20, 21, 21, 21, 23, 23, 24, 26, 27, 28, 29, 30, 31, 31, 32, 32, 32, 32, 34, 34, 35, 36, 36, 36, 41, 44, 45, 47, 47, 49, 49, 51, 53, 53, 54, 54, 54, 55, 56, 57, 59, 61, 62, 62, 62, 66, 66, 66, 68, 69, 69, 72, 72, 72, 72, 73, 74, 76, 76, 77, 80, 80, 82, 84, 85, 85, 85, 85, 85, 86, 86, 90, 91, 91, 92, 93, 96, 98]
Recursive depth:  5
Process finished with exit code 0
```

Рисунок 3.1 – Сортування масиву на 100 елементів

```
Initial list:  [538, 284, 433, 560, 790, 324, 893, 392, 685, 413, 454, 483, 537, 290, 725, 62, 370, 782, 140, 646, 402, 532, 642, 788, 44, 436, 753, 746, 430, 281, 472, 488, 2, 656, 909, 530, 212, 268, 901, 356, 353, 258, 744, 868, 808, 751, 671, 536, 296, 661, 888, 449, 439, 831, 706, 65, 118, 202, 741, 780, 268, 105, 734, 120, 450, 221, 124, 362, 317, 556, 802, 940, 397, 699, 551, 51, 662, 259, 403, 624, 50, 691, 346, 285, 621, 395, 943, 405, 416, 88, 378, 408, 639, 974, 613, 963, 947, 838, 756, 209, 662, 56, 445, 538, 353, 802, 865, 906, 76, 368, 770, 897, 459, 736, 943,
```

409, 749, 382, 317, 651, 839, 822, 185, 386, 680, 837, 699, 843, 794, 537, 750, 6, 872, 338, 409, 865, 968, 746, 599, 292, 117, 792, 637, 215, 282, 350, 608, 605, 161, 171, 523, 674, 456, 443, 721, 749, 783, 148, 222, 925, 178, 725, 588, 940, 26, 152, 714, 525, 184, 656, 189, 735, 862, 378, 687, 256, 284, 872, 197, 93, 276, 696, 729, 765, 894, 807, 705, 996, 773, 801, 8, 696, 857, 360, 572, 967, 273, 759, 735, 210, 265, 918, 183, 870, 463, 8, 505, 198, 94, 627, 812, 63, 764, 873, 243, 677, 732, 212, 104, 370, 130, 501, 791, 271, 379, 510, 236, 828, 991, 771, 263, 251, 367, 234, 981, 40, 616, 69, 624, 148, 119, 656, 33, 921, 746, 377, 819, 599, 936, 920, 8, 574, 113, 498, 214, 671, 106, 555, 818, 259, 308, 545, 833, 728, 145, 670, 472, 747, 653, 428, 287, 15, 993, 673, 331, 631, 473, 177, 608, 757, 881, 501, 938, 303, 770, 79, 469, 788, 874, 863, 758, 71, 26, 43, 23, 560, 185, 323, 814, 702, 577, 632, 96, 420, 168, 270, 194, 140, 716, 188, 20, 677, 124, 557, 822, 2, 784, 283, 874, 121, 90, 560, 197, 806, 757, 571, 912, 646, 605, 29, 284, 343, 930, 415, 638, 719, 848, 718, 600, 266, 829, 122, 206, 182, 129, 526, 311, 294, 905, 510, 46, 496, 536, 939, 543, 94, 111, 701, 438, 380, 874, 106, 629, 316, 511, 13, 795, 643, 953, 280, 159, 838, 310, 378, 270, 396, 759, 193, 90, 763, 392, 807, 659, 100, 922, 967, 29, 493, 357, 582, 490, 144, 962, 584, 34, 690, 802, 364, 428, 594, 152, 757, 913, 537, 869, 442, 435, 991, 451, 134, 609, 33, 359, 287, 332, 742, 871, 171, 853, 401, 379, 841, 998, 513, 657, 194, 235, 1, 847, 958, 592, 155, 442, 630, 793, 172, 243, 996, 205, 28, 182, 509, 609, 410, 631, 529, 121, 568, 517, 714, 453, 647, 924, 481, 958, 223, 482, 320, 47, 948, 77, 484, 825, 955, 732, 482, 644, 678, 124, 515, 669, 977, 824, 193, 428, 945, 968, 420, 992, 189, 867, 953, 369, 865, 176, 233, 931, 22, 930, 281, 268, 711, 86, 537, 51, 983, 215, 977, 712, 391, 674, 774, 893, 719, 607, 4, 202, 841, 739, 497, 945, 528, 699, 589, 434, 431, 264, 364, 932, 229, 412, 105, 286, 615, 413, 49, 684, 803, 409, 302, 620, 358, 419, 368, 104, 214, 978, 100, 138, 277, 686, 628, 642, 515, 929, 367, 422, 959, 410, 282, 697, 311, 44, 183, 342, 710, 724, 141, 800, 713, 886, 480, 58, 53, 905, 842, 611, 820, 151, 427, 383, 553, 10, 76, 605, 700, 600, 924, 399, 897, 575, 643, 686, 455, 934, 481, 170, 248, 263, 911, 286, 513, 190, 640, 975, 105, 48, 674, 116, 288, 866, 425, 790, 340, 442, 190, 55, 447, 492, 856, 58, 350, 7, 322, 953, 589, 741, 592, 399, 80, 409, 503, 69, 453, 959, 867, 269, 310, 261, 448, 596, 489, 631,

208, 295, 14, 890, 190, 674, 881, 804, 529, 998, 610, 819, 8, 104, 333, 821, 556, 278, 932, 585, 277, 979, 733, 177, 656, 635, 904, 947, 32, 813, 494, 418, 925, 605, 470, 866, 912, 630, 672, 495, 969, 652, 336, 309, 180, 753, 561, 596, 453, 342, 793, 358, 708, 778, 509, 174, 225, 512, 11, 138, 326, 254, 296, 189, 343, 221, 184, 28, 188, 629, 802, 940, 186, 41, 707, 886, 617, 894, 183, 884, 925, 737, 513, 330, 626, 474, 875, 856, 467, 228, 707, 344, 667, 215, 89, 190, 200, 664, 556, 297, 543, 734, 273, 963, 159, 440, 597, 965, 170, 381, 539, 914, 932, 354, 637, 860, 159, 413, 577, 291, 416, 593, 238, 66, 435, 838, 296, 1, 820, 915, 677, 594, 77, 734, 46, 698, 814, 291, 186, 137, 474, 913, 864, 683, 462, 219, 260, 804, 919, 173, 943, 277, 503, 988, 707, 143, 771, 644, 31, 68, 380, 768, 788, 911, 983, 241, 526, 519, 457, 458, 823, 127, 629, 36, 936, 128, 367, 828, 313, 976, 669, 656, 902, 250, 36, 115, 168, 653, 752, 640, 836, 527, 848, 116, 539, 755, 379, 730, 165, 795, 242, 955, 711, 496, 643, 426, 496, 656, 74, 221, 411, 672, 999, 96, 599, 441, 113, 887, 712, 189, 469, 713, 9, 918, 984, 150, 722, 549, 956, 249, 661, 159, 16, 225, 535, 848, 562, 24, 327, 282, 31, 989, 38, 732, 691, 111, 653, 140, 339, 65, 304, 398, 324, 461, 579, 754, 990, 458, 734, 30, 69, 12, 72, 51, 802, 585, 779, 574, 204, 742, 168, 551, 809, 584, 779, 273, 905, 213, 652, 505, 127, 126, 119, 80, 167, 883, 447, 415, 108, 257, 507, 220, 524, 828, 732, 369, 169, 804, 513, 380, 368, 716, 820, 389, 6, 571, 791, 245, 470, 753, 72, 425, 254, 880, 747, 8, 827, 425, 539, 405, 970, 599, 875, 126, 892, 649, 689, 309, 910, 459, 495, 354, 133, 511, 816, 233, 826, 114, 678, 906, 851, 633, 616, 902, 762, 337, 143, 531, 364, 3, 240, 956, 328, 125, 812, 191, 216, 973, 512, 607, 51, 724, 185, 299, 300, 10, 901]

Result list: [1, 1, 2, 2, 3, 4, 6, 6, 7, 8, 8, 8, 8, 8, 9, 10, 10, 11, 12, 13, 14, 15, 16, 20, 22, 23, 24, 26, 26, 28, 28, 29, 29, 30, 31, 31, 32, 33, 33, 34, 36, 36, 38, 40, 41, 43, 44, 44, 46, 46, 47, 48, 49, 50, 51, 51, 51, 51, 53, 55, 56, 58, 58, 62, 63, 65, 65, 66, 68, 69, 69, 69, 71, 72, 72, 74, 76, 76, 77, 77, 79, 80, 80, 86, 88, 89, 90, 90, 93, 94, 94, 96, 96, 100, 100, 104, 104, 104, 105, 105, 105, 106, 106, 108, 111, 111, 113, 113, 114, 115, 116, 116, 117, 118, 119, 119, 120, 121, 121, 122, 124, 124, 124, 125, 126, 126, 127, 127, 128, 129, 130, 133, 134, 137, 138, 138, 140, 140, 140, 141, 143, 143, 144, 145, 148, 148, 150, 151, 152, 152, 155, 159, 159, 159, 159, 161, 165, 167, 168,

168, 168, 169, 170, 170, 171, 171, 172, 173, 174, 176, 177, 177, 178, 180, 182, 182, 183, 183, 183, 184, 184, 185, 185, 185, 186, 186, 188, 188, 189, 189, 189, 189, 190, 190, 190, 190, 191, 193, 193, 194, 194, 197, 197, 198, 200, 202, 202, 204, 205, 206, 208, 209, 210, 212, 212, 213, 214, 214, 215, 215, 215, 216, 219, 220, 221, 221, 221, 222, 223, 225, 225, 228, 229, 233, 233, 234, 235, 236, 238, 240, 241, 242, 243, 243, 245, 248, 249, 250, 251, 254, 254, 256, 257, 258, 259, 259, 260, 261, 263, 263, 264, 265, 266, 268, 268, 268, 269, 270, 270, 271, 273, 273, 273, 276, 277, 277, 277, 278, 280, 281, 281, 282, 282, 282, 283, 284, 284, 284, 285, 286, 286, 287, 287, 288, 290, 291, 291, 292, 294, 295, 296, 296, 296, 297, 299, 300, 302, 303, 304, 308, 309, 309, 310, 310, 311, 311, 313, 316, 317, 317, 320, 322, 323, 324, 324, 326, 327, 328, 330, 331, 332, 333, 336, 337, 338, 339, 340, 342, 342, 343, 343, 344, 346, 350, 350, 353, 353, 354, 354, 356, 357, 358, 358, 359, 360, 362, 364, 364, 364, 367, 367, 367, 368, 368, 368, 369, 369, 370, 370, 377, 378, 378, 378, 379, 379, 379, 380, 380, 380, 381, 382, 383, 386, 389, 391, 392, 392, 395, 396, 397, 398, 399, 399, 401, 402, 403, 405, 405, 408, 409, 409, 409, 409, 410, 410, 411, 412, 413, 413, 413, 415, 415, 416, 416, 418, 419, 420, 420, 422, 425, 425, 425, 426, 427, 428, 428, 428, 430, 431, 433, 434, 435, 435, 436, 438, 439, 440, 441, 442, 442, 442, 443, 445, 447, 447, 448, 449, 450, 451, 453, 453, 453, 454, 455, 456, 457, 458, 458, 459, 459, 461, 462, 463, 467, 469, 469, 470, 470, 472, 472, 473, 474, 474, 480, 481, 481, 482, 482, 483, 484, 488, 489, 490, 492, 493, 494, 495, 495, 496, 496, 496, 497, 498, 501, 501, 503, 503, 505, 505, 507, 509, 509, 510, 510, 511, 511, 512, 512, 513, 513, 513, 513, 515, 515, 517, 519, 523, 524, 525, 526, 526, 527, 528, 529, 529, 530, 531, 532, 535, 536, 536, 537, 537, 537, 537, 538, 538, 539, 539, 539, 543, 543, 545, 549, 551, 551, 553, 555, 556, 556, 556, 557, 560, 560, 560, 561, 562, 568, 571, 571, 572, 574, 574, 575, 577, 577, 579, 582, 584, 584, 585, 585, 588, 589, 589, 592, 592, 593, 594, 594, 596, 596, 597, 599, 599, 599, 599, 600, 600, 605, 605, 605, 605, 607, 607, 608, 608, 609, 609, 610, 611, 613, 615, 616, 616, 617, 620, 621, 624, 624, 626, 627, 628, 629, 629, 629, 630, 630, 631, 631, 631, 632, 633, 635, 637, 637, 638, 639, 640, 640, 642, 642, 643, 643, 643, 644, 644, 646, 646, 647, 649, 651, 652, 652, 653, 653, 653, 656, 656, 656, 656, 656, 656, 657, 659, 661, 661, 662, 662, 664, 667, 669, 669, 670, 671, 671, 672, 672, 673,

```
674, 674, 674, 674, 677, 677, 677, 678, 678, 680, 683, 684, 685, 686, 686, 687, 689,
690, 691, 691, 696, 696, 697, 698, 699, 699, 699, 700, 701, 702, 705, 706, 707, 707,
707, 708, 710, 711, 711, 712, 712, 713, 713, 714, 714, 716, 716, 718, 719, 719, 721,
722, 724, 724, 725, 725, 728, 729, 730, 732, 732, 732, 732, 733, 734, 734, 734, 734,
735, 735, 736, 737, 739, 741, 741, 742, 742, 744, 746, 746, 746, 747, 747, 749, 749,
750, 751, 752, 753, 753, 753, 754, 755, 756, 757, 757, 757, 758, 759, 759, 762, 763,
764, 765, 768, 770, 770, 771, 771, 773, 774, 778, 779, 779, 780, 782, 783, 784, 788,
788, 788, 790, 790, 791, 791, 792, 793, 793, 794, 795, 795, 800, 801, 802, 802, 802,
802, 802, 803, 804, 804, 804, 806, 807, 807, 808, 809, 812, 812, 813, 814, 814, 816,
818, 819, 819, 820, 820, 820, 821, 822, 822, 823, 824, 825, 826, 827, 828, 828, 828,
829, 831, 833, 836, 837, 838, 838, 838, 839, 841, 841, 842, 843, 847, 848, 848, 848,
851, 853, 856, 856, 857, 860, 862, 863, 864, 865, 865, 865, 866, 866, 867, 867, 868,
869, 870, 871, 872, 872, 873, 874, 874, 874, 875, 875, 880, 881, 881, 883, 884, 886,
886, 887, 888, 890, 892, 893, 893, 894, 894, 897, 897, 901, 901, 902, 902, 904, 905,
905, 905, 906, 906, 909, 910, 911, 911, 912, 912, 913, 913, 914, 915, 918, 918, 919,
920, 921, 922, 924, 924, 925, 925, 925, 929, 930, 930, 931, 932, 932, 932, 934, 936,
936, 938, 939, 940, 940, 940, 943, 943, 943, 945, 945, 947, 947, 948, 953, 953, 953,
955, 955, 956, 956, 958, 958, 959, 959, 962, 963, 963, 965, 967, 967, 968, 968, 969,
970, 973, 974, 975, 976, 977, 977, 978, 979, 981, 983, 983, 984, 988, 989, 990, 991,
991, 992, 993, 996, 996, 998, 998, 999]

    Recursive depth:  6

    Process finished with exit code 0
```

Рисунок 3.2 – Сортування масиву на 1000 елементів

3.5 Тестування алгоритму

3.5.1 Часові характеристики оцінювання

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

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

Розмірність масиву	Глибина рекурсії
10	9
100	99
1000	infinity
5000	infinity
10000	infinity
20000	infinity
50000	infinity

В таблиці 3.3 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування вставками для масивів різної розмірності, коли масиви містять зворотно упорядковану послідовність елементів.

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

Розмірність масиву	Глибина рекурсії
10	1
100	1
1000	infinity
5000	infinity
10000	infinity
20000	infinity

50000	infinity

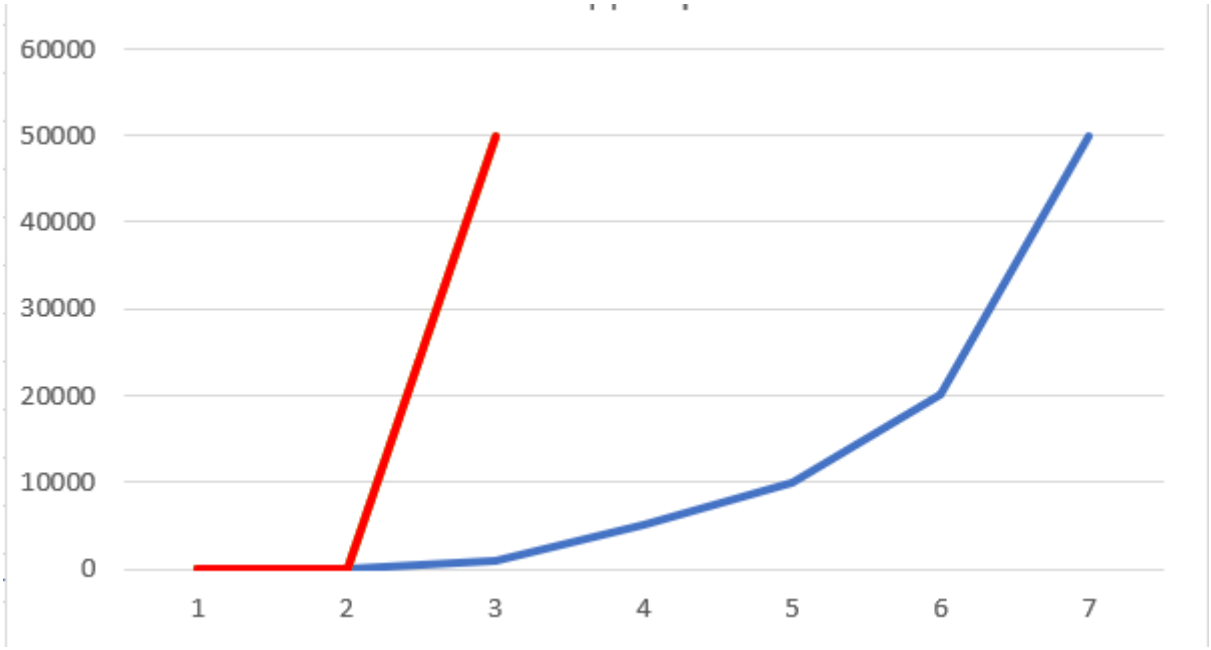
У таблиці 3.4 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування вставками для масивів різної розмірності, масиви містять випадкову послідовність елементів.

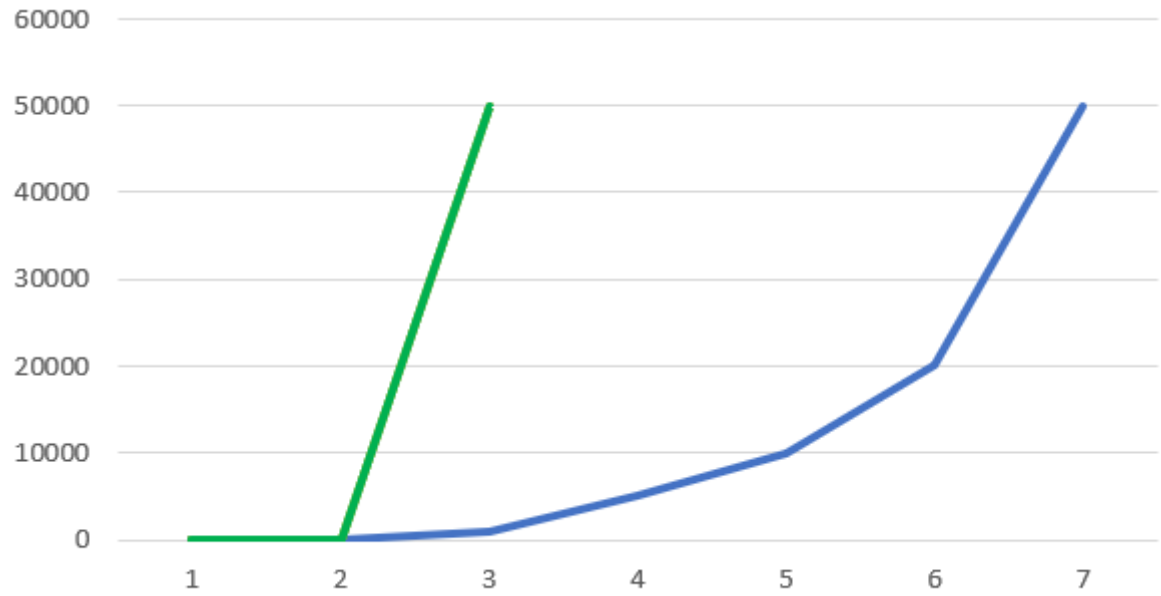
Таблиця 3.4 – Характеристика оцінювання алгоритму швидкого сортування для випадкової послідовності елементів у масиві.

Розмірність масиву	Глибина рекурсії
10	2
100	4
1000	6
5000	7
10000	10
20000	12
50000	15

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.


60000
50000
40000
30000
20000
10000
0
1
2
3
4
5
6
7


60000
50000
40000
30000
20000
10000
0
1
2
3
4
5
6
7

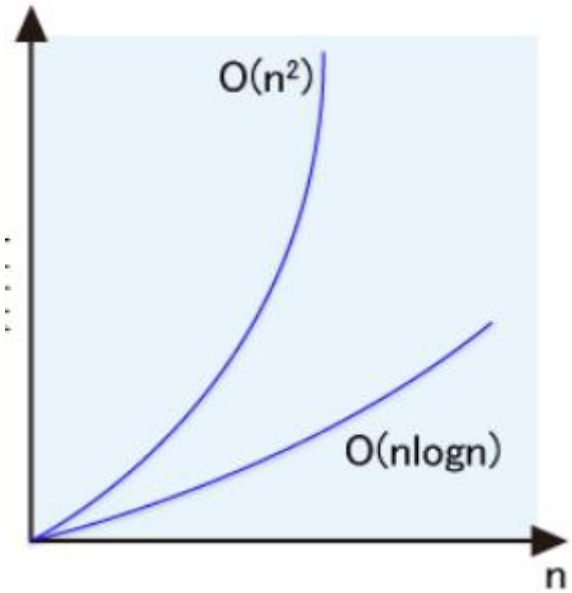

O(n²)
O(nlogn)
n

Рисунок 3.3 – Графіки залежності часових характеристик оцінювання

ВИСНОВОК

При виконанні даної лабораторної роботи ми вивчили основні методи аналізу обчислювальної складності алгоритмів внутрішнього сортування навчилися оцінювати поріг їх ефективності, розробили алгоритм розв’язання задачі відповідно до варіанту, виконали аналіз алгоритму внутрішнього сортування, записали алгоритм за допомогою псевдокоду, а також виконали програмну реалізацію задачі з фіксацією часових характеристик оцінювання, провели аналіз часової складності в гіршому, кращому і середньому випадках та записали часову складність в асимптотичних оцінках, провели 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 побудували графіки залежності часових характеристик оцінювання від розмірності масиву.

.

КРИТЕРІЇ ОЦІНЮВАННЯ

У випадку здачі лабораторної роботи до 18.05.2020 включно максимальний бал дорівнює – 5. Після 18.05.2020 максимальний бал дорівнює – 1.

Критерії оцінювання у відсотках від максимального балу:

- аналіз алгоритму на відповідність властивостям – 10%;
- псевдокод алгоритму – 15%;
- аналіз часової складності – 25%;
- програмна реалізація алгоритму – 25%;
- тестування алгоритму – 20%;
- висновок – 5%.