

Харківський національний університет імені В. Н. Каразіна  
Факультет комп'ютерних наук  
Кафедра штучного інтелекту та програмного забезпечення

ЗВІТ  
З ПРАКТИЧНОЇ РОБОТИ №8  
дисципліна: «Алгоритми та структур и даних»

Виконала: студентка групи КС-22

Узенкова Дар'я

Перевірив: Олешко Олег

Харків

2024

Завдання. Аналіз ефективності сортування Jsort (J-sort) by Jason Morrison

1. Реалізувати сортування
2. Обґрунтувати теоретичну складність для кращого та гіршого випадків
3. Виміряти практичну швидкодію алгоритму для кращого та гіршого випадків
4. Зробити висновки на підставі отриманих результатів

### Методика тестування ефективності алгоритмів

1. Реалізувати функцію сортування. Переконайтеся, що вона коректно працює для невеликого масиву (10-15 елементів, щоб можна було перевірити "вручну").
2. Додати вимірювання часу сортування. Використовуйте функції, що забезпечують точність вимірювання до мікросекунд (або наносекунд).
3. Додати функцію читання масиву із файлу (бінарного або текстового). Програма тепер повинна зчитувати файл довольного розміру в динамічний масив і відсортувати цей масив. Відсортований масив записується в інший файл, щоб можна було перевірити коректність результату.
4. Написати окрему програму, яка буде генерувати файли з вхідними даними для сортування. Щоб можна було згенерувати файл необхідного розміру та з певними властивостями (вже відсортований, відсортований у зворотному порядку, випадковий і т.ін.)
5. Провести вимірювання часу сортування для файлів різного розміру та з різними властивостями.  
Рекомендується виконувати по декілька вимірювань для одного й того ж файлу та усереднювати отримані результати.  
Також рекомендується при тестуванні закрити всі інші програми, щоб мінімізувати їх вплив на результати вимірювань.  
- Дані з файлу зчитуємо в буфер (масив) і на ньому вже проводимо вимірювання.
6. Оформлюємо результати в вигляді фінальної таблиці (та/або графіків) і робимо висновки на основі отриманих експериментальних результатів.

Результати виконання завдання 1 наведено:

1. У лістингу 1 – вихідний код програми.
2. У малюнках 1, 2, 3, 4 – результати виконання програми.
3. У графіку порівняння трьох випадків вхідних даних
4. У таблицях 1-5.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>

// Функція для сортування методом Дженсена
void j_sort(int *arr, int n, int *comparisons, int *swaps) {
    int j = n / 2; // Початкова відстань для порівняння
```

```

// Завжди виконуємо сортування для всіх значень j
while (j > 0) {
    for (int i = 0; i + j < n; i++) {
        (*comparisons)++; // Ураховуємо кожне порівняння

        if (arr[i] > arr[i + j]) {
            int temp = arr[i];
            arr[i] = arr[i + j];
            arr[i + j] = temp;
            (*swaps)++; // Ураховуємо перестановку
        }
    }
    j /= 2; // Зменшуємо відстань j вдвічі
}

// Функція для зчитування масиву з файлу
int *read_array_from_file(const char *filename, int *n) {
    FILE *file = fopen(filename, "r");
    if (!file) {
        perror("Помилка при відкриванні файлу.");
        return NULL;
    }

    // Визначаємо максимальний розмір масиву
    int size = 10; // Початковий розмір масиву
    int *arr = (int *)malloc(size * sizeof(int));
    if (!arr) {
        perror("Помилка виділення пам'яті");
        fclose(file);
        return NULL;
    }

    *n = 0; // Ініціалізуємо кількість елементів
    while (fscanf(file, "%d", &arr[*n]) == 1) { // Читаємо числа до кінця файлу
        (*n)++;
        if (*n >= size) { // Збільшуємо розмір масиву за необхідності
            size *= 2; // Подвоюємо розмір
            arr = (int *)realloc(arr, size * sizeof(int));
            if (!arr) {
                perror("Помилка при перевиділенні пам'яті");
                fclose(file);
                return NULL;
            }
        }
    }
    fclose(file);

    return arr;
}

```

```

// Функція для запису масиву у файл
void write_array_to_file(const char *filename, int *arr, int n) {
    FILE *file = fopen(filename, "w");
    if (!file) {
        perror("Помилка при відкриванні файлу.");
        return;
    }

    for (int i = 0; i < n; i++) {
        fprintf(file, "%d ", arr[i]);
    }
    fclose(file);
}

// Головна функція програми
int main() {
    char input_filename[100];
    int n;

    printf("Введіть назву файлу для зчитування: ");
    scanf("%99s", input_filename);

    int *arr = read_array_from_file(input_filename, &n);
    if (arr == NULL) {
        return 1;
    }

    int comparisons = 0; // Змінна для підрахунку кількості порівнянь
    int swaps = 0; // Змінна для підрахунку кількості перестановок

    clock_t start, end;
    start = clock();

    j_sort(arr, n, &comparisons, &swaps); // Передаємо адреси змінних

    end = clock();

    // Вимірювання часу у мікросекундах
    double time_taken = ((double)(end - start) / CLOCKS_PER_SEC) * 1000000;
    printf("\nЧас сортування: %.2f мкс\n", time_taken);
    printf("Кількість порівнянь: %d\n", comparisons);
    printf("Кількість перестановок: %d\n", swaps);

    // Запис результатів у файл output.txt
    write_array_to_file("output.txt", arr, n);
    printf("Відсортований масив записано до файлу 'output.txt'\n");
    free(arr);
    return 0;
}

```

Лістинг 1 – вихідний код програми сортування J-Sort

```

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void generate_sorted_array(const char *filename, int n) {
    FILE *file = fopen(filename, "w");
    if (!file) {
        perror("Помилка при створенні файлу.");
        return;
    }

    for (int i = 0; i < n; i++) {
        fprintf(file, "%d ", i + 1);
    }
    fclose(file);
}

void generate_reverse_sorted_array(const char *filename, int n) {
    FILE *file = fopen(filename, "w");
    if (!file) {
        perror("Помилка при створенні файлу.");
        return;
    }

    for (int i = n; i > 0; i--) {
        fprintf(file, "%d ", i);
    }
    fclose(file);
}

void generate_random_array(const char *filename, int n) {
    FILE *file = fopen(filename, "w");
    if (!file) {
        perror("Помилка при створенні файлу.");
        return;
    }

    srand(time(NULL));

    for (int i = 0; i < n; i++) {
        fprintf(file, "%d ", rand() % n);
    }
    fclose(file);
}

int main() {
    int n;
    printf("Введіть кількість елементів у масиві: ");
    scanf("%d", &n);

    generate_sorted_array("sorted_input.txt", n);
}

```

```

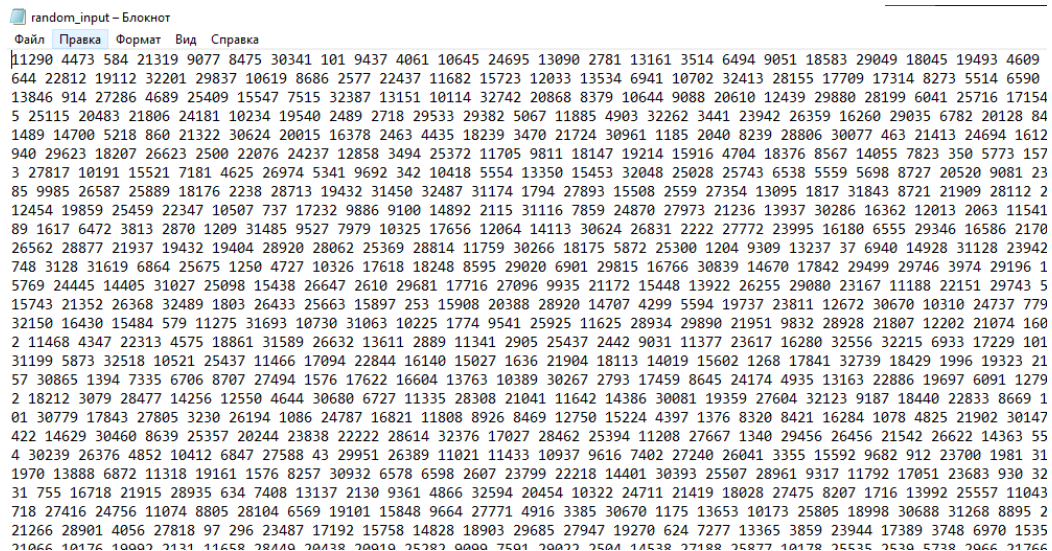
generate_reverse_sorted_array("reverse_sorted_input.txt", n);
generate_random_array("random_input.txt", n);

printf("Файли згенеровані: sorted_input.txt, reverse_sorted_input.txt,
random_input.txt\n");

return 0;
}

```

## Лістинг 2 – код програми для генерації різних випадків вхідних даних

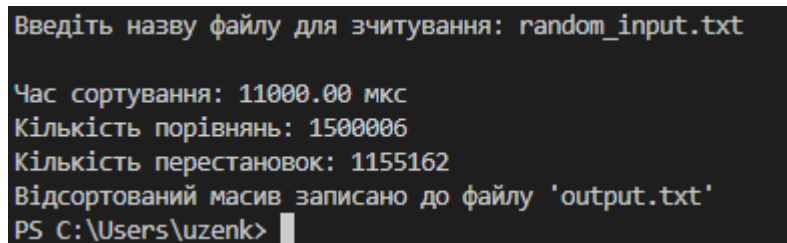


random\_input – Блокнот

Файл Правка Формат Вид Справка

11290 4473 584 21319 9077 8475 30341 101 9437 4061 10645 24695 13090 2781 13161 3514 6494 9051 18583 29049 18045 19493 4609  
644 22812 19112 32201 29837 10619 8686 2577 22437 11682 15723 12033 13534 6941 10702 32413 28155 17709 17314 8273 5514 6590  
13846 914 27286 4689 25409 15547 7515 32387 13151 10114 32742 20868 8379 10644 9088 20610 12439 29880 28199 6041 25716 17154  
5 25115 20483 21806 24181 10234 19540 2489 2718 29533 29382 5067 11885 4903 32262 3441 23942 26359 16260 29035 6782 20128 84  
1489 14700 5218 860 21322 30624 20015 16378 2463 4435 18239 3470 21724 30961 1185 2040 8239 28806 30077 463 21413 24694 1612  
940 29623 18207 26623 2500 22076 24237 12858 3494 25372 11705 9811 18147 19214 15916 4704 18376 8567 14055 7823 350 5773 157  
3 27817 10191 15521 7181 4625 26974 5341 9692 342 10418 5554 13350 15453 32048 25028 25743 6538 5559 5698 8727 20520 9081 23  
85 9985 26587 25889 18176 2238 28713 19432 31450 32487 31174 1794 27893 15508 2559 27354 13095 1817 31843 8721 21909 28112 2  
12454 19859 25459 22347 10507 737 17232 9886 9100 14892 2115 31116 7859 24870 27973 21236 13937 30286 16362 12013 2063 11541  
89 1617 6472 3813 2870 1209 31485 9527 7979 10325 17656 12064 14113 30624 26831 2222 27772 23995 16180 6555 29346 16586 2170  
26562 28877 21937 19432 19404 28920 28062 25369 28814 11759 30266 18175 5872 25300 1204 9309 13237 37 6940 14928 31128 23942  
748 3128 31619 6864 25675 1250 4727 10326 17618 18248 8595 29020 6901 29815 16766 30839 14670 17842 29499 29746 3974 29196 1  
5769 24445 14405 31027 25098 15438 26647 2610 29681 17716 27096 9935 21172 15448 13922 26255 29080 23167 11188 22151 29743 5  
15743 21352 26368 32489 1803 26433 25663 15897 253 15908 20388 28920 14707 4299 5594 19737 23811 12672 30670 10310 24737 779  
32150 16430 15484 579 11275 31693 10730 31063 10225 1774 9541 25925 11625 28934 29890 21951 9832 28928 21807 12202 21074 160  
2 11468 4347 22313 4575 18861 31589 26632 13611 2889 11341 2905 25437 2442 9031 11377 23617 16280 32556 32215 6933 17229 101  
31199 5873 32518 10521 25437 11466 17094 22844 16140 15027 1636 21904 18113 14019 15602 1268 17841 32739 18429 1996 19323 21  
57 30865 1394 7335 6706 8707 27494 1576 17622 16604 13763 10389 30267 2793 17459 8645 24174 4935 13163 22886 19697 6091 1279  
2 18212 3079 28477 14256 12550 4644 30680 6727 11335 28308 21041 11642 14386 30081 19359 27604 32123 9187 18440 22833 8669 1  
01 30779 17843 27805 3230 26194 1086 24787 16821 11808 8926 8469 12750 15224 4397 1376 8320 8421 16284 1078 4825 21902 30147  
422 14629 30460 8639 25357 20244 23838 22222 28614 32376 17027 28462 25394 11208 27667 1340 29456 26456 21542 26622 14363 55  
4 30239 26376 4852 10412 6847 27588 43 29951 26389 11021 11433 10937 9616 7402 27240 26041 3355 15592 9682 912 23700 1981 31  
1970 13888 6872 11318 19161 1576 8257 30932 6578 6598 2607 23799 22218 14401 30393 25507 28961 9317 11792 17051 23683 930 32  
31 755 16718 21915 28935 634 7408 13137 2130 9361 4866 32594 20454 10322 24711 21419 18028 27475 8207 1716 13992 25557 11043  
718 27416 24756 11074 8805 28104 6569 19101 15848 9664 27771 4916 3385 30670 1175 13653 10173 25805 18998 30688 31268 8895 2  
21266 28901 4056 27818 97 296 23487 17192 15758 14828 18903 29685 27947 19270 624 7277 13365 3859 23944 17389 3748 6970 1535  
21066 10176 10002 2131 11658 20410 20438 20010 25782 0000 7501 20002 2504 14538 27108 25877 10178 25535 2530 5738 2066 21766

## Текстовий файл з 100 000 рандомно розташованими елементами

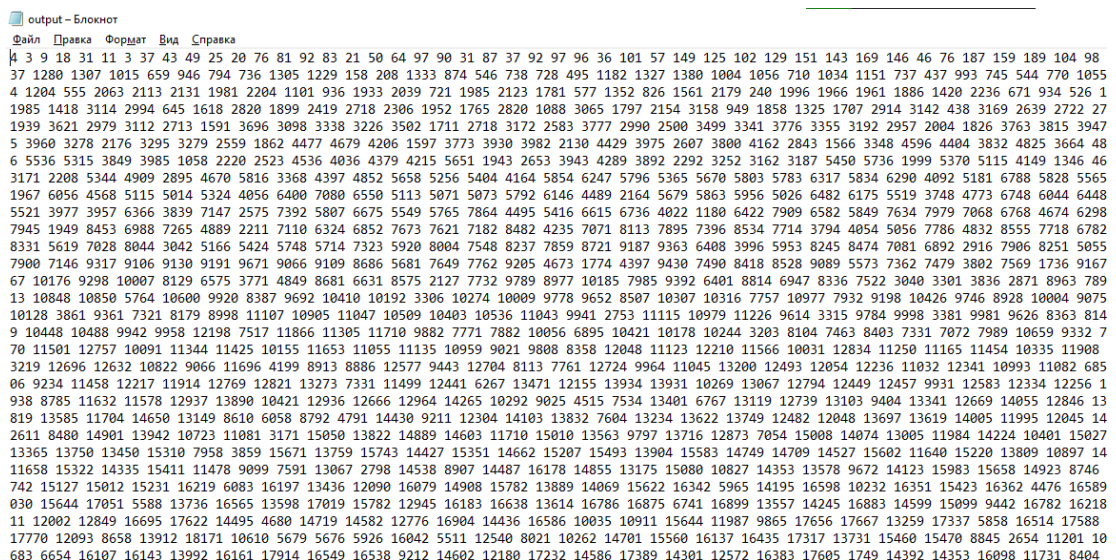


```

Введіть назву файлу для зчитування: random_input.txt

Час сортування: 11000.00 мкс
Кількість порівнянь: 1500006
Кількість перестановок: 1155162
Відсортований масив записано до файлу 'output.txt'
PS C:\Users\uzenk>

```



output – Блокнот

Файл Правка Формат Вид Справка

4 3 9 18 31 11 3 37 43 49 25 20 76 81 92 83 21 50 64 97 90 31 87 37 92 97 96 36 101 57 149 125 102 129 151 143 169 146 46 76 187 159 189 104 98  
37 1280 1307 1015 659 946 794 736 1305 1229 158 208 1333 874 546 738 728 495 1182 1327 1380 1004 1056 710 1034 1151 737 437 993 745 544 770 1055  
4 1204 555 2063 2113 2131 1981 2204 1101 936 1933 2039 721 1985 2123 1781 577 1352 826 1561 2179 240 1996 1966 1961 1886 1420 2236 671 934 526 1  
1985 1418 3114 2994 645 1618 2820 1899 2419 2718 2306 1952 1765 2820 1088 3065 1797 2154 3158 949 1858 1325 1707 2914 3142 438 3169 2639 2722 27  
1939 3621 2979 3112 2713 1591 3696 3098 3338 3226 3502 1711 2718 3172 2583 3777 2990 2500 3499 3341 3776 3355 3192 2957 2004 1826 3763 3815 3947  
5 3960 3278 2176 3295 3279 2559 1862 4477 4679 4206 1597 3773 3930 3982 2130 4429 3975 2607 3800 4162 2843 1566 3348 4596 4404 3832 4825 3664 48  
6 5536 5315 3849 3985 1058 2220 2523 4536 4036 4379 4215 5651 1943 2653 3943 4289 3892 2292 3252 3162 3187 5450 5736 1999 5370 5115 4149 1346 46  
3171 2208 5344 4909 2895 4670 5816 3368 4397 4852 5658 5256 5404 4164 5854 6247 5796 5365 5670 5803 5783 6317 5834 6290 4092 5181 6788 5828 5565  
1967 6056 4568 5115 5014 5324 4056 6400 7080 6550 5113 5071 5073 5792 6146 4489 2164 5679 5863 5956 5026 6482 6175 5519 3748 4773 6748 6044 6448  
5521 3977 3957 6366 3839 7147 2575 7392 5807 6675 5549 5765 7864 4495 5416 6615 6736 4022 1180 6422 7909 6582 5849 7634 7979 7068 6768 4674 6298  
7945 1949 8453 6988 7265 4889 2211 7110 6324 6852 7673 7621 7182 8482 4235 7071 8113 7895 7396 8534 7714 3794 4054 5056 7786 4832 8555 7718 6782  
8331 5619 7028 8044 3042 5166 5424 5748 5714 7323 5920 8004 7548 8237 7859 8721 9187 9363 6408 3996 5953 8245 8474 7081 6892 2916 7906 8251 5055  
7900 7146 9317 9106 9130 9191 9671 9066 9109 8686 5681 7649 7762 9205 4673 1774 4397 9430 7490 8418 8528 9089 5573 7362 7479 3802 7569 1736 9167  
67 10176 9298 10007 8129 6575 3771 4849 8681 6631 8575 2127 7732 9279 8977 10185 7985 9392 6401 8814 6947 8336 7522 3040 3301 3836 2871 8963 789  
13 10848 10850 5764 10600 9920 8387 9692 10410 10192 3306 10274 10009 9778 9652 8507 10307 10316 7757 10977 7932 9198 10426 9746 8928 10004 9075  
10128 3861 9361 7321 8179 8998 11107 10905 11047 10509 10403 10536 11043 9941 2753 11115 10979 11226 9614 3315 9784 9998 3381 9981 9626 8363 814  
9 10448 10488 9942 9958 12198 7517 11866 11305 11710 9882 7771 7882 10056 6895 10421 10178 10244 3203 8104 7463 8403 7331 7072 7989 10659 9332 7  
70 11501 12757 10091 11344 11425 10155 11653 11055 11135 10959 9021 9008 8358 12048 11123 12210 11566 10031 12034 11250 11165 11454 10335 11908  
3219 12696 12632 10822 9066 11696 4199 8913 8886 12577 9443 12704 8113 7761 12724 9964 11045 13200 12493 12854 12236 11032 12341 10993 11082 685  
96 9234 11458 12217 11914 12769 12821 13273 7331 11499 12441 6267 13471 12155 13934 13931 10269 13067 12794 12449 12457 9931 12583 12334 12256 1  
938 8785 11632 11578 12937 13890 10421 12936 12666 12964 14265 10292 9045 4515 7534 13401 6767 13119 12739 13103 9404 13341 12669 14055 12846 13  
819 13585 11704 14650 13149 8610 6058 8792 4791 14430 9211 12304 14103 13832 7604 13234 13622 13749 12482 12048 13697 13619 14005 11995 12045 14  
2611 8480 14901 13942 10723 11081 3171 15050 13822 14889 14603 11710 15010 13563 9797 13716 12873 7054 15008 14074 13003 11984 14224 10401 15027  
13365 13750 13450 15310 7958 3859 15671 13759 15743 14427 15351 14662 15207 15493 13904 15583 14749 14709 14527 15602 11640 15220 13809 10897 14  
11658 15322 14335 15411 11478 9099 7591 13067 2798 14538 8907 14487 16178 14855 13175 15080 10827 14353 13578 9672 14123 15983 15658 14923 8746  
742 15127 15012 15231 16219 6083 16197 13436 12090 16079 14908 15782 13889 14069 15622 16342 5965 14195 16598 10232 16351 15423 16362 4476 16589  
030 15644 17051 5588 13736 16565 13598 17019 15782 12945 16183 16638 13614 16786 16875 6741 16899 13557 14245 16883 14599 15099 9442 16782 16218  
11 12082 12849 16695 17622 14495 4680 14719 14582 12776 16904 14436 16586 10035 10911 15644 11987 9865 17656 17667 13259 17337 5585 16514 17588  
17770 12093 8658 13912 18171 10610 5679 5676 5926 16042 5511 12540 8021 10622 14701 15560 16137 16435 17317 13731 15460 15470 8845 2654 11201 10  
683 6654 16107 16143 13992 16161 17914 16549 16538 9212 14602 12180 17232 14586 17389 14301 12572 16383 17605 1749 14392 14353 16098 11731 8404

## 'output.txt' – Файл з відсортованим масивом

Таблиця 1 - Дані для найкращого випадку

Час сортування (мкс)	0	0	0	0	1000
Розмір масиву	1000	10 000	30 000	100 000	500 000

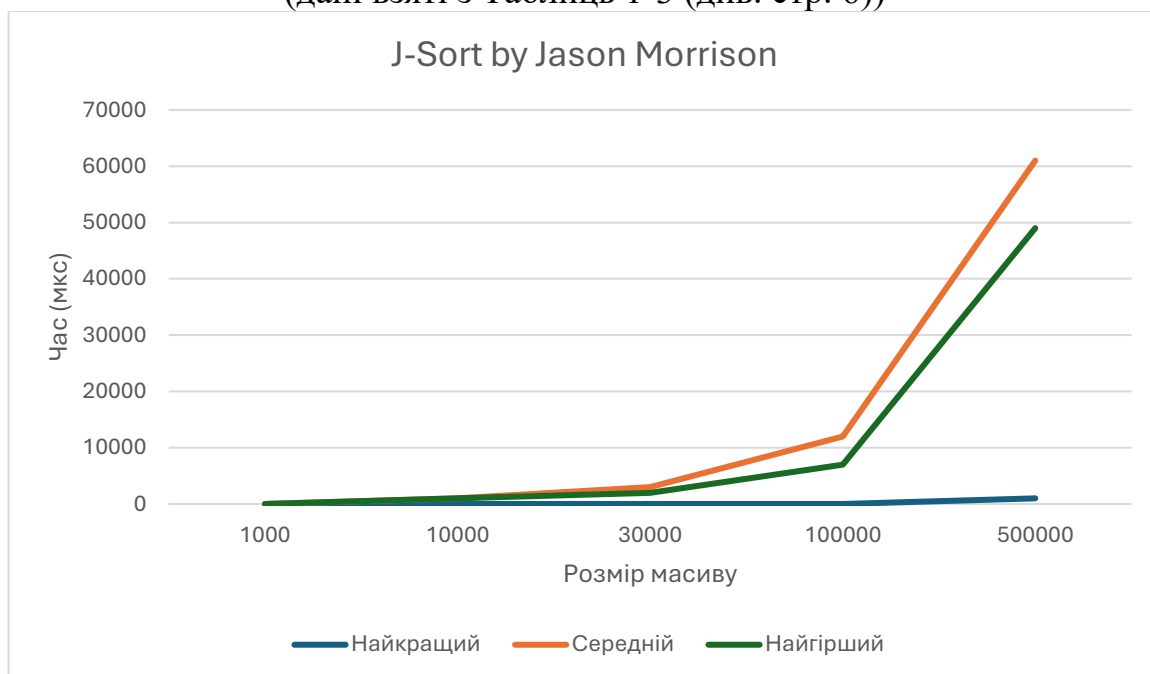
Таблиця 2 - Дані для середнього випадку

Час сортування (мкс)	0	1000	3000	12000	61000
Розмір масиву	1000	10 000	30 000	100 000	500 000

Таблиця 3 - Дані для найгіршого випадку

Час сортування (мкс)	0	1000	2000	7000	49000
Розмір масиву	1000	10 000	30 000	100 000	500 000

Графік порівняння трьох випадків (найкращий, середній, найгірший)  
(дані взяті з Таблиць 1-3 (див. стр. 6))



Таблиця 4 - Операції при сортуванні (масив розміром 500 000)

Випадок	Порівняння	Перестановки	Час
Найкращий	250000	0	1000
Середній	8500007	6779127	61000
Найгірший	8500007	4526576	49000

Таблиця 5 – Оцінка ефективності алгоритму

Алгоритм	Структура даних	Найкращий випадок	Середній випадок	Найгірший випадок
Сортування	Масив	$O(n)$	$O(n \log n)$	$O(n \log n)$

Судячи з таблиці, ефективність алгоритму в різних випадках наступна:

1. Найкращий випадок –  $O(n)$ :

У цьому випадку масив вже відсортований, тому алгоритм завершує роботу мінімальною кількістю порівнянь (250 000) та без жодної перестановки. Це відповідає лінійній складності  $O(n)$ . Час виконання в цьому випадку становить 1 000 мс, що є дуже оптимальним.

2. Середній випадок –  $O(n \log n)$ :

У середньому випадку (для випадкових даних) алгоритм демонструє значно більшу кількість порівнянь (8 500 007) та перестановок (6 779 127), із часом виконання 61 000 мс. Це свідчить про те, що у середньому випадку він працює повільніше, ніж у найгіршому, ймовірно через розподіл елементів, який потребує великої кількості переміщень, навіть якщо не так багато елементів безпосередньо упорядковані. Але загальна поведінка все ж наближається до  $O(n \log n)$ , хоча час доволі значний.

3. Найгірший випадок –  $O(n \log n)$ :

У найгіршому випадку (зворотний порядок елементів) алгоритм здійснює стільки ж порівнянь, скільки і в середньому випадку (8 500 007), але кількість перестановок (4 526 576) нижча, ніж у середньому випадку. Час виконання становить 49 000 мс, що швидше, ніж у середньому випадку.

## Висновок

Тут алгоритм демонструє цікавий результат: середній випадок виявився повільнішим, ніж найгірший. Це може свідчити про те, що при певних випадкових розподілах елементів алгоритм витрачає більше операцій перестановок, ніж коли масив повністю зворотно впорядкований.