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

ЗВІТ
З ПРАКТИЧНОЇ РОБОТИ №5

дисципліна: «Алгоритми та структур и даних»

Виконала: студентка групи КС-22
Узенкова Дар’я
Перевірив: Олешко Олег

Харків
2024

Завдання. Bubble Sort

1. Реалізуємо алгоритм Bubble Sort.
2. Підбираємо вхідні дані для визначення часу роботи:
    a. найгіршого випадка
    b. найкращого випадка
    c. середнього випадка
2. Вимірюємо час сортування на практикці.
3. Проводимо підрахунок кількості операцій порівняння та перестановки, таким чином визначаємо теоретичну складність алгоритма (для найгіршого/найкращого/середнього випадка).
4. Будуємо графіки витраченого часу.
5. Будуємо табличку оцінки складності.

Результати виконання завдання 1 наведено:

1. У лістингу 1 – вихідний код програми.
2. У малюнках 1, 2, 3 – результати виконання програми.
3. У графіку 1, 2, 3 – графіки витраченого часу для різних випадків.
4. У таблиці 1 – таблиця оцінки складності для різних випадків.

```
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>  // Для QueryPerformanceCounter

void swap(int* xp, int* yp) {
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

void bubbleSort(int arr[], int n, int* comparisons, int* swaps) {
    int i, j;
    bool swapped;
    *comparisons = 0;  // Лічильник порівнянь
    *swaps = 0;        // Лічильник перестановок

    for (i = 0; i < n - 1; i++) {
        swapped = false;
        for (j = 0; j < n - i - 1; j++) {
            (*comparisons)++;  // Збільшуємо лічильник порівнянь
            if (arr[j] > arr[j + 1]) {
                swap(&arr[j], &arr[j + 1]);
                (*swaps)++;     // Збільшуємо лічильник перестановок
                swapped = true;
            }
        }
        if (!swapped) // Якщо не було перестановок, масив
відсортовано
```

```
            break;
    }
}

void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

int main() {
    int comparisons, swaps;

    int* arr = NULL;
    int num;
    int count = 0;

    printf("Введіть числа для масиву (введіть 0 для завершення):\n");
    while (1) {
        scanf("%d", &num);

        // Якщо ввели 0, завершуємо введення
        if (num == 0)
            break;

        // Збільшуємо розмір масиву на 1 елемент
        arr = (int*)realloc(arr, (count + 1) * sizeof(int));
        if (arr == NULL) {
            perror("Не вдалося виділити пам'ять");
            exit(EXIT_FAILURE);
        }

        // Додаємо новий елемент у масив
        arr[count] = num;
        count++;
    }

    LARGE_INTEGER start, end, frequency;

    QueryPerformanceFrequency(&frequency); // Отримуємо частоту
таймера
    QueryPerformanceCounter(&start);       // Початок вимірювання
часу

    bubbleSort(arr, count, &comparisons, &swaps);

    QueryPerformanceCounter(&end);         // Кінець вимірювання часу

    double time_us = (double)(end.QuadPart - start.QuadPart) *
1000000.0 / frequency.QuadPart; // Час у мікросекундах
```

```
    printf("\nВідсортований масив: \n");
    printArray(arr, count);

    printf("\nЧас сортування: %f мкс\n", time_us);
    printf("Кількість порівнянь: %d\n", comparisons);
    printf("Кількість перестановок: %d\n", swaps);

    // Звільняємо пам'ять
    free(arr);

    return 0;
}
```

Лістинг 1 – вихідний код програми

Теоретична складність:

- Найгірший випадок: O(n^2) порівнянь і перестановок — коли масив упорядкований у зворотному порядку.

```
Введіть числа для масиву (введіть 0 для завершення):
1000 999 998 997 996 995 994 993 992 991 990 989 988 987 986 985
984 983 982 981 980 979 978 977 976 975 974 973 972 971 970 969
968 967 966 965 964 963 962 961 960 959 958 957 956 955 954 953
952 951 950 ... 0

Відсортований масив:
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 ... 1000

Час сортування: 3759.600000 мкс
Кількість порівнянь: 499500
Кількість перестановок: 499500
```

Малюнок 1 – Найгірший випадок

- Найкращий випадок: O(n) порівнянь і O(1) перестановок — коли масив уже відсортований.

```
Введіть числа для масиву (введіть 0 для завершення):
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 ... 1000 0

Відсортований масив:
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 ... 1000

Час сортування: 2.800000 мкс
Кількість порівнянь: 999
Кількість перестановок:0
```

Малюнок 2 – Накращий випадок

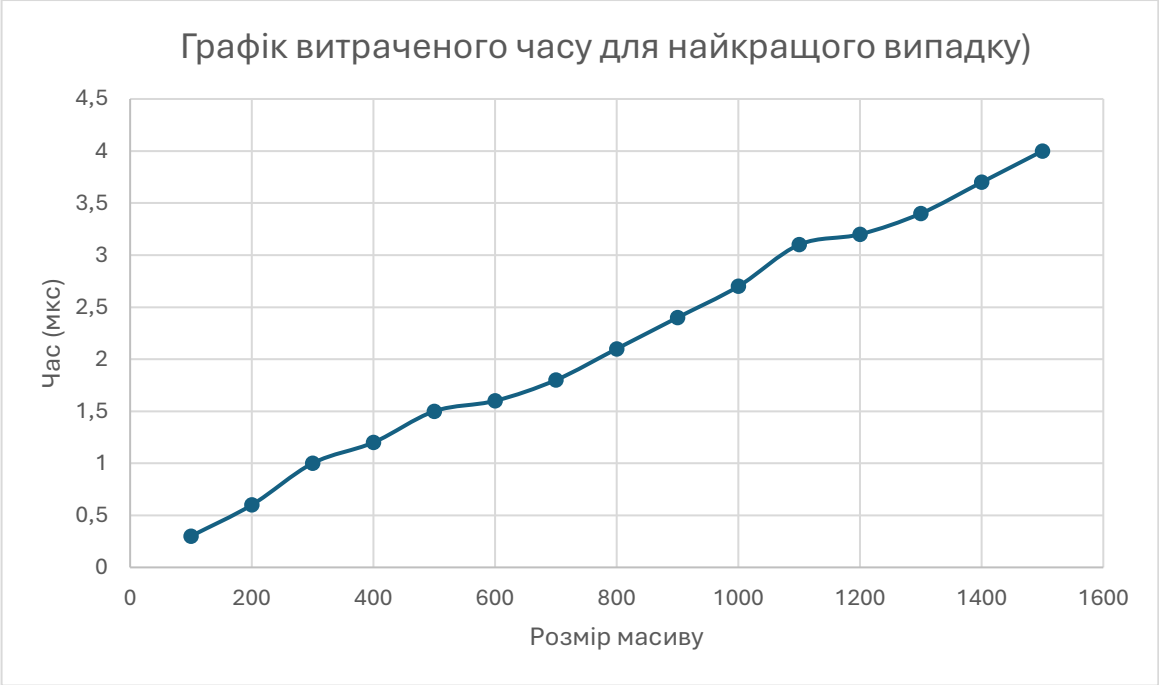
- Середній випадок: O(n^2) порівнянь і перестановок — при випадкових вхідних даних.

```
Введіть числа для масиву (введіть 0 для завершення):
853 427 912 578 284 51 492 111 236 600 408 935 798 17 849 732 5 416
295 364 205 263 978 646 121 89 71 284 503 998 43 222 786 706 987 132
718 660 576 860 88 637 427 509 394 721 834 777 852 379 118 765 230
689 402 611 486 970 663 304 188 524 298 689 96 883 174 542 564 19 693
193 236 827 226 267 314 963 276 303 475 834 218 300 29 975 756 153
413 35 447 184 276 540 992 239 512 546 913 669 554 670 501 745 645
614 830 615 828 614 872 905 481 267 724 293 161 315 222 494 146 27 9
230 783 451 551 611 816 273 493 319 225 942 762 926 300 167 99 887
978 189 970 94 218 90 636 746 370 3 327 415 392 217 206 949 796 268
713 471 234 298 268 35 687 299 81 793 610 576 15 930 782 43 812 568
454 146 936 75 496 174 478 856 754 293 32 333 321 163 733 1 588 522
88 63 877 885 568 29 119 86 153 526 489 635 222 178 66 563 129 270
250 467 651 740 226 556 473 481 43 204 39 76 934 370 292 258 917 337
383 923 777 32 0

Відсортований масив:
1 3 5 9 15 17 19 27 29 29 32 32 35 35 39 43 43 43 51 63 66 71 75 76
81 86 88 88 89 90 94 96 99 111 118 119 121 129 132 146 146 153 153
161 163 167 174 174 178 184 188 189 193 204 205 206 217 218 218 222
222 222 225 226 226 230 230 234 236 236 239 250 258 263 267 267 268
268 270 273 276 276 284 284 292 293 293 295 298 298 299 300 300 303
304 314 315 319 321 327 333 337 364 370 370 379 383 392 394 402 408
413 415 416 427 427 447 451 454 467 471 473 475 478 481 481 486 489
492 493 494 496 501 503 509 512 522 524 526 540 542 546 551 554 556
563 564 568 568 576 576 578 588 600 610 611 611 614 614 615 635 636
637 645 646 651 660 663 669 670 687 689 689 693 706 713 718 721 724
732 733 740 745 746 754 756 762 765 777 777 782 783 786 793 796 798
812 816 827 828 830 834 834 849 852 853 856 860 872 877 883 885 887
905 912 913 917 923 926 930 934 935 936 942 949 963 970 970 975 978
978 987 992 998

Час сортування: 174.300000 мкс
Кількість порівнянь: 27206
Кількість перестановок: 14647
```

Малюнок 3 – Середній випадок

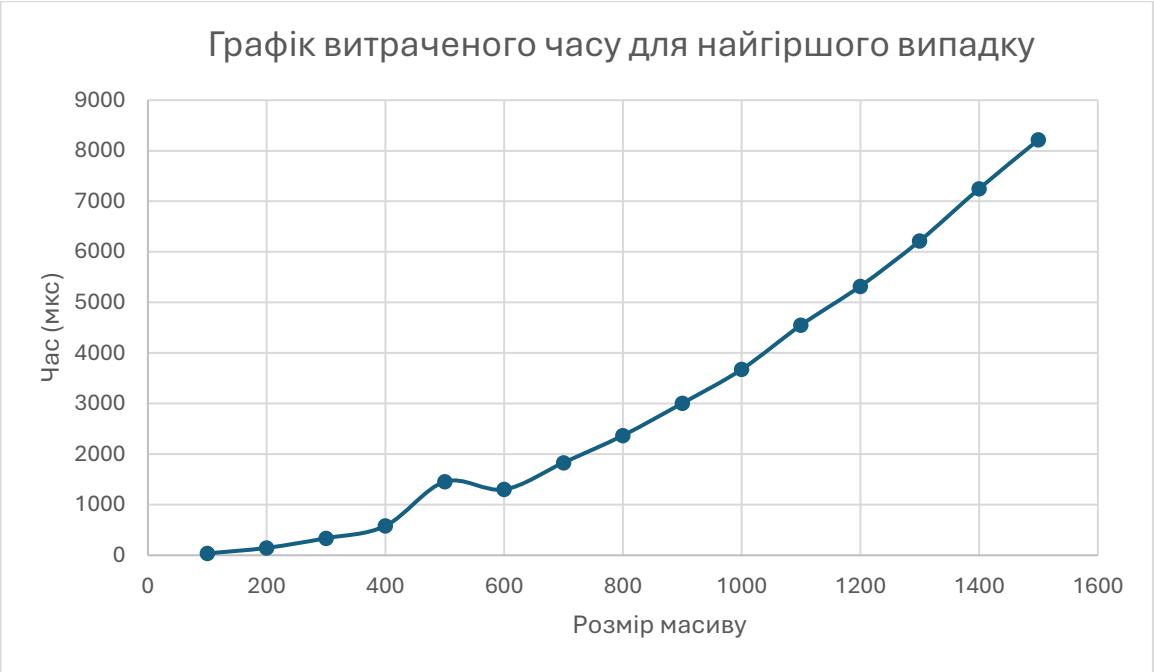
Графіки витраченого часу

Найкращий випадок (відсортований масив)



Середній випадок (масив з випадковою послідовністю)

Найгірший випадок (відсортований за спаданням масив)



Таблиця оцінки складності

Випадок	**Порівняння**	**Перестановки**	**Час (мкс)**
Найкращий випадок	999	0	2.80 мкс
Середній випадок	27206	14647	174.30 мкс
Найгірший випадок	499500	499500	3759.60 мкс