Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра системного проектування сервісів

# ЗВІТ

про виконання комп'ютерного практикуму № 3
з дисципліни «Алгоритми і структури даних»

Виконав:
Студент І курсу
Групи ДА-72
Хоменко Є.С
Варіант № 27

Перевірив:
Караюз І.В

Київ – 2017

**1. Варіант завдання : 27**

**2. Завдання:** Написати програму що реалізує такі методи сортування:

1) обмін, базовий
2) пірамідальний
3) підрахунком

**3. Лістинг програми:**

```
#include <stdio.h>
#include <stdlib.h>
#define N 1000 // 10000, 100000
#define R 100
#define RAND(R) rand()%R
#define CNT_PRN printf("\n cnt_cmp = %lld cnt_cpy =
%lld\n\n",cnt_cmp,cnt_cpy);

long cnt_cmp, cnt_cpy;

int printf_array(int a[],int n);
int best_array(int a[],int n);
int worst_array(int a[],int n);
int avg_array(int a[],int n,int k);
int counting_sort(int a[], int n, int max);
int bubble_sort(int a[],int n);
int heap_sort(int a[], int n);
int build_max_heap(int a[],int n);
int max_heapify(int a[], int i,int n);



int main() {

    int array[N];

    best_array(array,N);

    avg_array(array, N, 3);

    worst_array(array,N);


    return 0;
}
int printf_array(int a[], int n){
    int i;
    for(i = 0; i < n; i++) printf("%d\t",a[i]);
    printf("\n\n");
    return i;
}

int best_array(int a[],int n){
    int i, max;
    max = a[0];
    for(i = 0; i < n; i++){
        a[i]=i;
```

```
    //printf_array(a, n);
    //bubble_sort(a,N);
    // counting_sort(a, N, max);              //call sorting method here!
    //heap_sort(a, N);

    return i;
}

int worst_array(int a[],int n){
    int i, max;
    max = a[0];
    for(i = 0; i < n; i++){
       a[i]=n - 1 - i;
        if(max < a[i]) max = a[i];
    }
    printf("Worst array: \n");
    //printf_array(a, n);
    //bubble_sort(a,N);
    //counting_sort(a, N, max);            // call sorting method here!
    //heap_sort(a, N);
    return i;
}

int avg_array(int a[],int n, int k){
    int i,j, max;
    max = a[0];
    for(j = 1; j <= k; j++){
        srand(j);
        for(i = 0; i < n; i++){
            a[i] = RAND(R);
            if(max < a[i]) max = a[i];
        }
        printf("Average array: \n");
        //printf_array(a, n);
        //bubble_sort(a,N);
        //counting_sort(a, N, max);          // call sorting method here!
        //heap_sort(a, N);
        }

    return j;
}

int bubble_sort(int a[], int n){
    int temp,i,j;

    for( i = 0; i < n - 1; i++){
        for(j = 0; j < n - 1 - i ; j++){
            cnt_cmp++;
                if(a[j] > a[j+1]){
                temp = a[j+1];
                a[j+1]= a[j];
                a[j] = temp;
                    cnt_cpy += 3;
            }
        }

    }
    printf("Bubble sort:\n");
   // printf_array(a, n);
    CNT_PRN;
    cnt_cmp = 0;
```

```
    return 0;
}

int counting_sort(int a[], int n,int max){
    int  count[max + 1];
    int res[n];
    int j;
    for(j = 0; j <= max ; j++){
        count[j] = 0;

    }
    for(j = 0;j < n; j++){
        count[a[j]]=count[a[j]]+1;

    }
    for(j = 1; j <= max; j++) {
        count[j] = count[j - 1] + count[j];

    }
    for(j = 0; j < n; j++){
        res[count[a[j]] - 1] = a[j];
        cnt_cpy++;
        count[a[j]] = count[a[j]] - 1;

    }
    printf("Counting sort:\n");
    //printf_array(res, N);
    CNT_PRN;
    cnt_cmp = 0;
    cnt_cpy = 0;
    printf("\n\n");
    return 0;
}

int heap_sort(int a[],int n){
    build_max_heap(a,n);
    int heapsize,i, temp;
    heapsize = n;
    for(i = n - 1;i > 0; i--){
        temp = a[0];
        a[0] = a[i];
        a[i] = temp;
        cnt_cpy += 3;
        heapsize--;
        max_heapify(a, 0, heapsize);
    }
    printf("Heap sort:\n");
    //printf_array(a, N);
    CNT_PRN;
    cnt_cmp = 0;
    cnt_cpy = 0;
    return 0;
}

int build_max_heap(int a[],int n){
    int i;
    for(i = n/2; i >= 0; i--){
        max_heapify(a, i, n);
    }
    return 0;
```

```
int max_heapify(int a[], int i, int n){
    int left, right;
    int largest, temp;

    left = 2 * (i + 1) - 1;
    right = 2 * (i + 1);

    if((left < n) && (a[left] > a[i])){
        largest = left;
        cnt_cmp++;
    }else {
        largest = i;
    }

    if((right < n) && (a[right] > a[largest])){
        largest = right;
        cnt_cmp++;
    }
    if(largest != i){
        temp = a[i];
        a[i] = a[largest];
        a[largest] = temp;
        cnt_cpy += 3;
        max_heapify(a, largest, n);
    }
    return 0;
}
```

**4. Результати експерименту :**

| N | Обмін, базовий | | | | | |
|---|---|---|---|---|---|---|
| | Кількість порівнянь | | | Кількість обмінів | | |
| | Теор. | Експерим. | Відношення | Теор. | Експерим. | Відношення |
| | Найкращий випадок | | | | | |
| 1000 | 499500 | 499500 | 1 | 0 | 0 | 1 |
| 10000 | 49995000 | 49995000 | 1 | 0 | 0 | 1 |
| 100000 | 4999950000 | 4999950000 | 1 | 0 | 0 | 1 |
| | Середній випадок | | | | | |
| 1000 | 499500 | 499500 | 1 | 749250 | 738828 | 1,04 |
| 10000 | 49995000 | 49995000 | 1 | 74992500 | 73968276 | 1,01 |
| 100000 | 4999950000 | 4999950000 | 1 | 7499925000 | 7429042533 | 1,01 |
| | Найгірший випадок | | | | | |
| 1000 | 499500 | 499500 | 1 | 1498500 | 1498500 | 1 |
| 10000 | 49995000 | 49995000 | 1 | 149985000 | 149985000 | 1 |
| 100000 | 4999950000 | 4999950000 | 1 | 14999850000 | 14999850000 | 1 |

Для пірамідального сортування вирахуємо теоретичні данні ( кількість порівнянь і кількість обмінів) для найгіршого, середнього і найкращого випадків як O(n*log(n)) у зв'язку з відсутністю конретних данних для кількісті порівнянь і кількості обмінів . Виникає суттєва похибка вимірювань6.

| N | Пірамідальний | | | | | |
|---|---|---|---|---|---|---|
| | Кількість порівнянь | | | Кількість обмінів | | |
| | Теор. | Експерим. | Відношення | Теор. | Експерим. | Відношення |
| | Найкращий випадок | | | | | |
| 1000 | 9965 | 12963 | 0,77 | 9965 | 29124 | 0,342 |
| 10000 | 132877 | 180584 | 0,74 | 132877 | 395868 | 0,336 |
| 100000 | 1660964 | 23003177 | 0,072 | 1660964 | 4952562 | 0,335 |
| | Середній випадок | | | | | |
| 1000 | 9965 | 11512 | 0,9 | 9965 | 26979 | 0,37 |
| 10000 | 132877 | 164169 | 0,81 | 132877 | 369786 | 0,36 |
| 100000 | 1660964 | 2132174 | 0,78 | 1660964 | 4685649 | 0,354 |
| | Найгірший випадок | | | | | |
| 1000 | 9965 | 10340 | 0,96 | 9965 | 24948 | 0,4 |
| 10000 | 132877 | 153619 | 0,87 | 132877 | 350088 | 0,4 |
| 100000 | 1660964 | 2024810 | 0,82 | 1660964 | 4492302 | 0,37 |

| N | Обмін, базовий | | | | | |
|---|---|---|---|---|---|---|
| | Кількість порівнянь | | | Кількість обмінів | | |
| | Теор. | Експерим. | Відношення | Теор. | Експерим. | Відношення |
| | Найкращий випадок | | | | | |
| 1000 | 0 | 0 | 1 | 1000 | 1000 | 1 |
| 10000 | 0 | 0 | 1 | 10000 | 10000 | 1 |
| 100000 | 0 | 0 | 1 | 100000 | 100000 | 1 |
| | Середній випадок | | | | | |
| 1000 | 0 | 0 | 1 | 1000 | 1000 | 1 |
| 10000 | 0 | 0 | 1 | 10000 | 10000 | 1 |
| 100000 | 0 | 0 | 1 | 100000 | 10000 | 1 |
| | Найгірший випадок | | | | | |
| 1000 | 0 | 0 | 1 | 1000 | 1000 | 1 |
| 10000 | 0 | 0 | 1 | 10000 | 10000 | 1 |
| 100000 | 0 | 0 | 1 | 100000 | 100000 | 1 |

**Висновки:** У ході виконання комп’ютерного практикуму мовою С був написаний код, що дозволяє експериментально порівняти 3 види сортування: базовий обмін, пірамідальний та сортування підрахунком. Аналізуючи результати, можна дійти висновку , що у даному випадку найшвидше відпрацьовує сортування підрахунком. Хоча, метод сортування підрахунком ефективний тільки для масивів з натуральними числами невеликого діапазону (max -min); найскладдніша реалізація у методу пірамідального сортування, але для масиву з довільними елементами він є найбільш ефективним. Найменш ефективним виявився метотод базового обміну, він є найпростішим для реалізації.