

Практична робота № 5

Тема: Реалізація алгоритмів Quick_sort

Мета роботи: закріплення навичок роботи з масивами, засвоєння алгоритмів сортування вкладенням.

Теоретичні відомості

Швидке сортування (Quick Sort) — алгоритм сортування, розроблений Тоні Гоаром, який не потребує додаткової пам'яті і виконує у середньому $O(n \log n)$ операцій. Алгоритм використовує дуже прості цикли і операції, він працює швидше за інші алгоритми, що мають таку ж асимптотичну оцінку складності.

Ідея алгоритму полягає в перестановленні елементів масиву таким чином, щоб його можна було розділити на дві частини і кожний елемент з першої частини був не більший за будь-який елемент з другої. Впорядкування кожної з частин відбувається рекурсивно. Алгоритм швидкого сортування може бути реалізований як у масиві, так і в двозв'язному списку.

Алгоритм складається з трьох кроків:

1. Вибрати елемент з масиву. Назвемо його опорним.
2. Розбиття : перерозподіл елементів в масиві таким чином, що елементи менше опорного поміщаються перед ним, а більше або рівні після.
3. Рекурсивно застосувати перші два кроки до двох підмасивів зліва і праворуч від опорного елемента. Рекурсія не застосовується до масиву, в якому тільки один елемент або відсутні елементи.

У найбільш загальному вигляді алгоритм на псевдокоді (де A - сортований масив, а low і $high$ - відповідно, нижня і верхня межі сортується ділянки цього масиву) виглядає так:

```
algorithm quicksort (A, low, high) is
    if low < high then
        p: = partition (A, low, high)
        quicksort (A, low, p - 1)
        quicksort (A, p + 1, high)
```

Тут передбачається, що масив A передається за посиланням, тобто сортування відбувається «на тому ж місці», а неописана функція `partition` повертає індекс опорного елемента.

Для вибору опорного елемента і операції розбиття існують різні підходи, що впливають на продуктивність алгоритму.

Можлива також наступна реалізація швидкого сортування:

```
algorithm quicksort (A) is
    if A is empty
        return A
    pivot: = A.pop ()
    //витягти останній або перший елемент з масиву
    lA: = A.filter (where e < pivot)
    //створити масив з елементами менше опорного
    rA: = A.filter (where e > pivot)
    //створити масив з елементами більше опорного
    return quicksort (lA) + [pivot] + quicksort (rA)
//повернути масив складається з відсортованої лівій частині,
//опорного й відсортованої правій частині.
```

Приклад реалізації програми швидкого сортування на мові C++

```
#include <iostream>

using namespace std;

void quick_sort(int[],int,int);
int partition(int[],int,int);

int main()
{
    int a[50],n,i;
    cout<<"How many elements?";
    cin>>n;
    cout<<"\nEnter array elements:";

    for(i=0;i<n;i++)
        cin>>a[i];

    quick_sort(a,0,n-1);
    cout<<"\nArray after sorting:";

    for(i=0;i<n;i++)
        cout<<a[i]<<" ";

    return 0;
}

void quick_sort(int a[],int l,int u)
{
    int j;
    if(l<u)
    {
        j=partition(a,l,u);
        quick_sort(a,l,j-1);
```

```

        quick_sort(a, j+1, u);
    }
}

int partition(int a[], int l, int u)
{
    int v, i, j, temp;
    v=a[l];
    i=l;
    j=u+1;

    do
    {
        do
            i++;
        while (a[i]<v&& i<=u);

        do
            j--;
        while (v<a[j]);

        if (i<j)
        {
            temp=a[i];
            a[i]=a[j];
            a[j]=temp;
        }
    }while (i<j);

    a[l]=a[j];
    a[j]=v;

    return(j);
}

```

Завдання

I. Написати програму, в якій:

1. Згенерувати одновимірний масив цілих чисел розмірністю згідно варіанту.
2. Елементи масиву задати випадковим чином в діапазоні 0 ... 1000.
3. Запам'ятати цей масив. Виконати друк цього масиву на екран.
4. Виконати обробку масиву відповідно до варіанту. При написанні програми використати метод Quick_sort або сортування вкладенням.
5. Виконати друк перетвореного масиву на екран.
6. Здійснити пошук вказаного (з клавіатури) елемента у масиві, використовуючи лінійний пошук).
7. Здійснити пошук вказаного (з клавіатури) елемента у масиві, використовуючи бінарний пошук).

II. Оформити звіт.

Контрольні питання

1. Для чого потрібні алгоритми сортування?
2. З яких основних частин складається будь-який алгоритм сортування?
3. Які основні параметри алгоритмів сортування ви знаєте?
4. Поясніть принцип роботи сортування вибором.
5. Поясніть принцип роботи сортування методом простого обміну

Варіанти індивідуальних завдань

№	Завдання
1	Є одинірний масив довжиною $N = 33$. Упорядкувати масив за зростанням.
2	Відсортувати одновимірний масив довжиною $N = 67$ за спаданням.
3	Є одинірний масив довжиною $N = 45$. Відсортувати за спаданням елементи масиву, які є парними числами.
4	Є одинірний масив довжиною $N = 38$. Упорядкувати масив таким чином, щоб половина чисел масиву розташовувалися за зростанням, а друга половина - за спаданням.
5	Є одинірний масив довжиною $N = 32$. Відсортувати за зростанням ті елементи масиву, які розташовуються на непарних позиціях.