

Лекция 8. Сортировки

Опять же классическая задача обработки данных.

А любой набор данных можно отсортировать?

Нужно определиться с операцией сравнения.

Для простоты ставим для начала задачу сортировки вещественного массива по возрастанию.

```
void Sort (int n, double *a);
```

Традиционные методы:

- восходящая пузырьковая сортировка
- нисходящая пузырьковая сортировка (просеивание)
- перестановка максимума (сортировка обменом)
- сортировка Шелла
- быстрая сортировка
- сортировка слиянием
- сортировка кучей (пирамидальная, турнирная)
- radix сортировки
- сортировки подсчетом

Оценка производительности и эффективности сортировки:

- трудоемкость
 - — гарантированная (для любой перестановки гарантируется)
 - — средняя (на некотором множестве перестановок)
 - — “естественная наилучшая”
- дополнительная память
- $O(1)$
- $O(\log N)$
- $O(\sqrt{N})$
- $O(N)$

Теперь подробнее обо всех этих сортировках.

Все простые сортировки надо уметь писать сходу и в любом состоянии !!!

```
void swap(double *a, double *b)      swap(&a[i], &a[j]);
{                                     swap(a+i, a+j);
    double c = *a;
    *a = *b;
    *b = c;
}

void BubbleSort (int n, double *a)
{
    int i, k;
    bool br; // признак прерывания при досрочной сортировке
    for (k = n; k > 1; k--) {
        br = true;
        for (i = 0; i < k-1; i++) {
            if (a[i] > a[i+1]) {
                swap(a+i, a+i+1);
                br = false;
            }
        }
    }
}
```

```

        }
        if (br) return;
    }
}

void SiftSort (int n, double *a)
{
    int i, k;
    for (i = 1, i < n; i++) {
        for (k = i; k > 0 && a[k] < a[k-1]; k--) {
            swap(a+k, a+k-1);
        }
    }
}

void SwapMaxSort (int n, double *a)
{
    int i, k, imax;
    double amax, c;
    for (k = n-1; k > 1; k--) {
        amax = a[0];
        imax = 0;
        for (i = 1; i <= k; i++) {
            if (amax > a[i]) { amax = a[i]; imax = i; }
        }
        if (imax != k) {
            swap(a+k, a+imax);
        }
    }
}

void ShellSort(int n, double *a)
{
    int i, j, k;
    for (int k = n / 2; k > 0; k /= 2) {
        for (int i = k; i < n; i++) {
            for (int j = i - k; j >= 0 && a[j] > a[j+k]; j -= k) {
                swap(a+j, a+j+k);
            }
        }
    }
}

void QuickSort (int n, double *a)

int QuickSortSplitting (int n, double *a)
{
    int left = -1, right = n;

```

```
double v = 0.5*(a[0] + a[n-1]);
while (true) {
    while (a[++left] < v);
    while (a[--right] > v);
    // yTB: a[left] >= v && a[right] <= v
    if (left >= right) {
        return (right < n-1) ? right+1 : right;
    } else {
        swap(a+left, a+right);
    }
}

void QuickSort(int n, double *a)
{
    int k;
    if (n<2) return;
    k = QuickSortSplitting(n,a);
    QuickSort(k,a);
    QuickSort(n-k,a+k);
}

void QuickSort(int n, double *a)
{
    int k;
    while ( n>1 )
    {
        k = QuickSortSplitting(n,a);
        if ( k < n-k ) {
            QuickSort(k, a);
            a = a+k;
            n = n-k;
        } else {
            QuickSort(n-k, a+k);
            n = k;
        }
    }
}
```