Сортировки

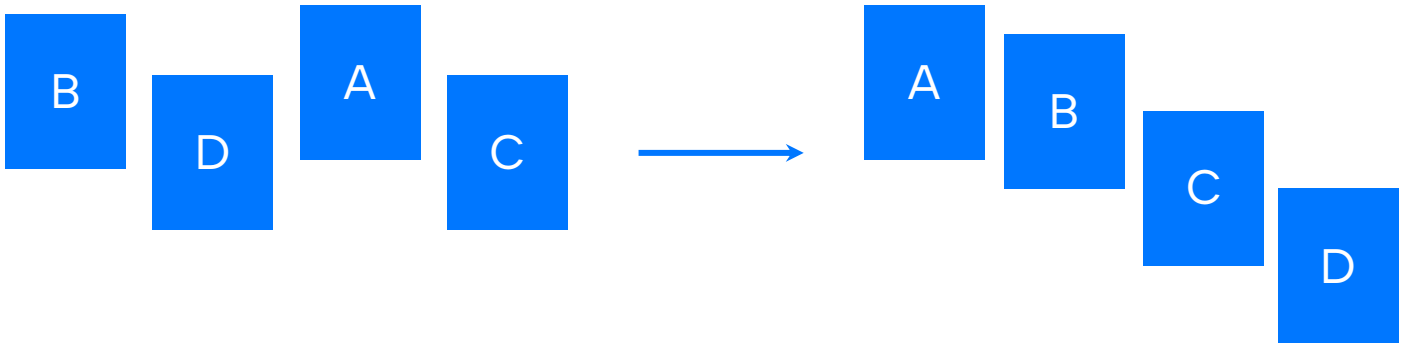
education

Концепция сортировки

education

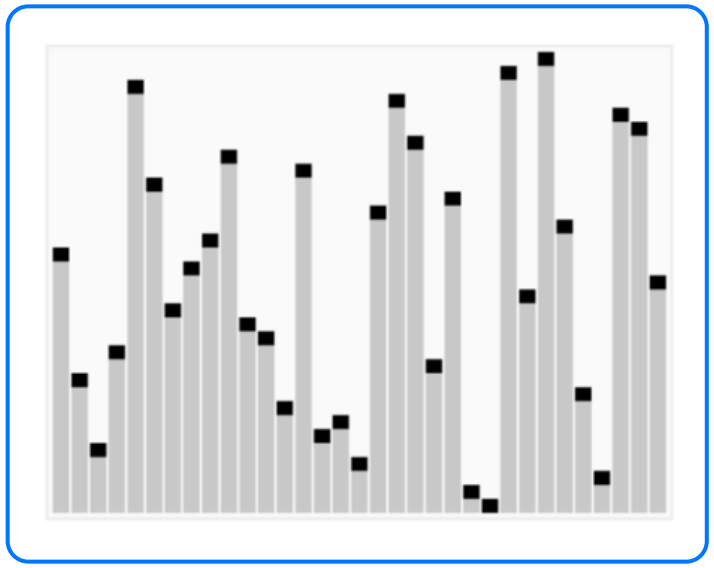
Сортировка

Сортировка — это процесс упорядочивания набора данных по некоторому ключу.



education

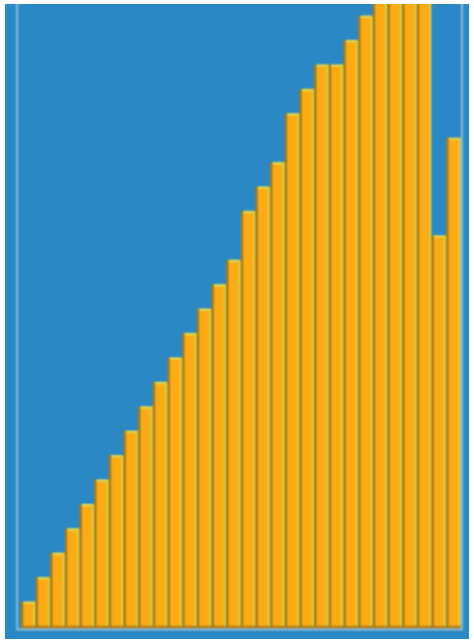
Быстрая сортировка (Quick Sort)



Временная сложность O(n log n) и O(n^2)

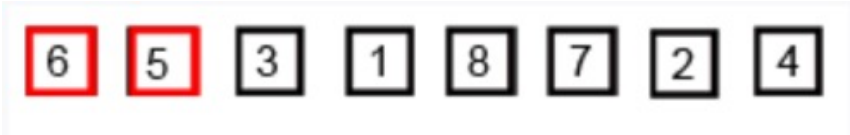
education

Сортировка вставками (Insertion Sort)



education

Сортировка слиянием (Merge Sort)



education

Bubble Sort

Сортировка пузырьком

```
функция bubbleSort(массив)
  повторять
    флаг_обмена = false
    для i от 0 до длина(массив) - 1
      если массив[i] > массив[i + 1]
        поменять местами массив[i] и массив[i + 1]
        флаг_обмена = true
  пока флаг_обмена = true
```

```
void BubbleSort(std::vector<int>& arr) {
    bool iSswapped;
    do {
        iSswapped = false;
        for (size_t i = 0; i < arr.size() - 1; i++) {
            if (arr[i] > arr[i + 1]) {
                std::swap(arr[i], arr[i + 1]);
                iSswapped = true;
            }
        }
    } while (iSswapped);
}
```

education

Bubble Sort

Сортировка пузырьком

```
функция bubbleSort(массив)
  повторять
    флаг_обмена = false
    для i от 0 до длина(массив) - 1
      если массив[i] > массив[i + 1]
        поменять местами массив[i] и массив[i + 1]
        флаг_обмена = true
  пока флаг_обмена = true
```

```
void BubbleSort(std::vector<int>& arr) {
    bool iSswapped;
    do {
        iSswapped = false;
        for (size_t i = 0; i < arr.size() - 1; i++) {
            if (arr[i] > arr[i + 1]) {
                std::swap(arr[i], arr[i + 1]);
                iSswapped = true;
            }
        }
    } while (iSswapped);
}
```

education

Bubble Sort

Сортировка пузырьком

```
if (arr[i] > arr[i + 1]) {
    std::swap(arr[i], arr[i + 1]);
    iSswapped = true;
```

education

Insertion Sort

Сортировка вставками

```
функция insertionSort(массив)
  для i от 1 до длина(массив)
    ключ = массив[i]
    j = i - 1
    пока j >= 0 и массив[j] > ключ
      массив[j + 1] = массив[j]
      j = j - 1
    массив[j + 1] = ключ
```

```
void InsertionSort(std::vector<int>& arr) {
    for (size_t i = 1; i < arr.size(); i++) {
                const int key = arr[i];
        int j = i - 1;

        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }

        arr[j + 1] = key;
    }
}
```

education

Insertion Sort

Сортировка вставками

```
функция insertionSort(массив)
  для i от 1 до длина(массив)
    ключ = массив[i]
    j = i - 1
    пока j >= 0 и массив[j] > ключ
      массив[j + 1] = массив[j]
      j = j - 1
    массив[j + 1] = ключ
```

```
void InsertionSort(std::vector<int>& arr) {
    for (size_t i = 1; i < arr.size(); i++) {
                const int key = arr[i];
        int j = i - 1;

        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }

        arr[j + 1] = key;
    }
}
```

education

Insertion Sort

Сортировка вставками

```
while (j >= 0 && arr[j] > key)
{
        arr[j + 1] = arr[j];
        j--;
}
```

education

Сортировка слиянием

```
void merge(std::vector<int>& arr, const int left, const int mid, const int right) {
                    const int n1 = mid - left + 1;
                    const int n2 = right - mid;
    std::vector<int> left_arr(n1), right_arr(n2);
    for (int i = 0; i < n1; i++)
        left_arr[i] = arr[left + i];
    for (int j = 0; j < n2; j++)
        right_arr[j] = arr[mid + 1 + j];
    int i = 0, j = 0, k = left;
    while (i < n1 && j < n2) {
        if (left_arr[i] <= right_arr[j])
            arr[k] = left_arr[i++];
        else
            arr[k] = right_arr[j++];
        k++;
    }

    while (i < n1)
        arr[k++] = left_arr[i++];
    while (j < n2)
        arr[k++] = right_arr[j++];
}
 void MergeSort(std::vector<int>& arr, const int left, const int right) {
    if (left >= right)
        return;

    const int mid = left + (right - left) / 2;

    MergeSort(arr, left, mid);
    MergeSort(arr, mid + 1, right);

    merge(arr, left, mid, right);
}
```

Быстрая сортировка

функция quicksort(массив)
  если длина(массив) ≤ 1
    возврат массива

  выбрать элемент из массива, назовем его «опорным»,
  создать два пустых массива: меньше и больше
  для каждого элемента в массиве, кроме опорного:
    если элемент ≤ опорному, добавить его в массив «меньше»;
    иначе — добавить его в массив «больше»;
  возврат quicksort(меньше) + опорный + quicksort(больше)

K-й порядковой статистикой набора элементов линейно упорядоченного множества называется такой его элемент, который является k.

Важно отметить, что эта порядковая статистика
и является нашим опорным элементом!

```
void QuickSort(std::vector<int>& arr, const int left, const int right) {
    int i = left, j = right;
    const int pivot = arr[(left + right) / 2];
    while (i <= j) {
        while (arr[i] < pivot)
            i++;
        while (arr[j] > pivot)
            j--;
        if (i <= j) {
            std::swap(arr[i], arr[j]);
            i++;
            j--;
        }
    }
    if (left < j)
        QuickSort(arr, left, j);
    if (i < right)
        QuickSort(arr, i, right);
}
```

Опорный элемент

```
while (i <= j) {
    while (arr[i] < pivot)
        i++;
    while (arr[j] > pivot)
        j--;
    if (i <= j) {
        std::swap(arr[i], arr[j]);
        i++;
        j--;
    }
}

if (left < j)
    QuickSort(arr, left, j);
if (i < right)
    QuickSort(arr, i, right);
```

education

Опорный элемент

```
int i = left, j = right;

    const int pivot = arr[(left + right) / 2];
```

education

Иллюстрация работы алгоритма

- Пусть у нас есть массив [3, 6, 8, 10, 1, 2, 1].
- Выбираем опорный элемент, например 6.
- Разделяем массив на две части: меньше [3, 1, 2, 1] и больше [8, 10].
- Применяем процесс рекурсивно для каждого подмассива.
  - Для [3, 1, 2, 1] выбираем 1 как опорный. Подмассивы: [] и [3, 2, 1].
  - Применяем быструю сортировку для [3, 2, 1], получаем [1, 2, 3].
  - Для [8, 10] выбираем 10 как опорный. Подмассивы: [8] и [].
- Объединяем результаты вместе: [1, 1, 2, 3] + [6] + [8, 10] = [1, 1, 2, 3, 6, 8, 10], что является отсортированным массивом.

Будем ВКонтакте!