Сортировки и поиск

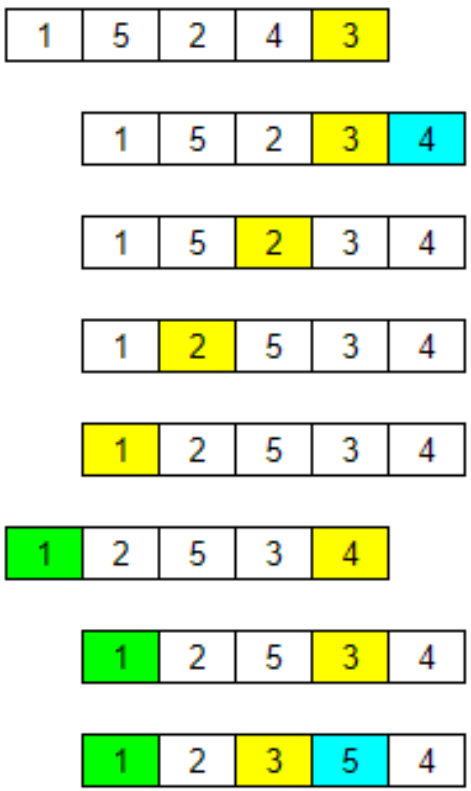
Юрий Литвинов
yurii.litvinov@gmail.com

02.03.2018г

Свойства алгоритмов сортировки

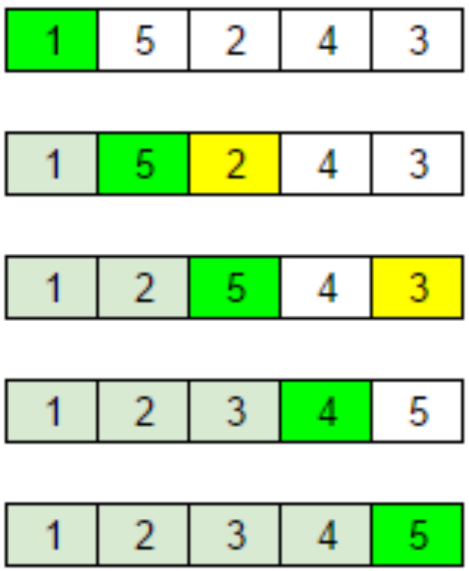
- Работают над любыми контейнерами данных
- Есть понятие “ключ”
- Устойчивость — сохраняется ли взаимное расположение элементов с одинаковым ключом
- Естественность — учёт степени отсортированности исходных данных
- Внутренняя сортировка — работает над данными, целиком помещающимися в память
- Внешняя сортировка работает над данными на устройствах с последовательным доступом, которые медленнее, чем память

Сортировка пузырьком (bubble sort)



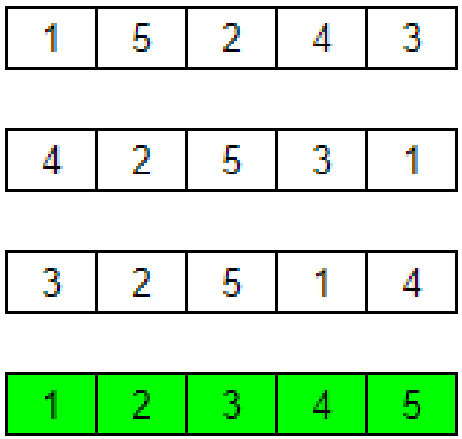
- $O(n^2)$
- Устойчива
- Естественная (на отсортированном массиве за линейное время)
- https://cathyatseneca.github.io/DSAnim/web/bubble.html

Сортировка выбором (selection sort)



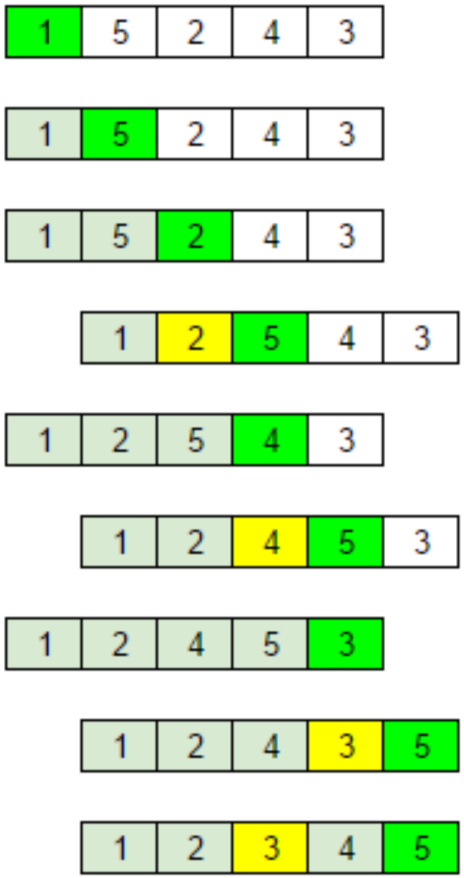
- $O(n^2)$
- Обычно неустойчива ($[2_a, 2_b, 1_a] -> [1_a, 2_b, 2_a]$)
- Отсортированность массива ничего не даёт
- Меньше всего операций обмена (меньше операций записи, что иногда позитивно)
- https://cathyatseneca.github.io/DSAnim/web/selection.html

Глупая сортировка (Bogosort)



- $O(n * n!)$
- Неустойчива
- Неестественная (на отсортированном массиве за линейное время, но на частично отсортированном массиве за n*n!)
- http://www.algostructure.com/sorting/bogosort.php

Сортировка вставкой (insertion sort)

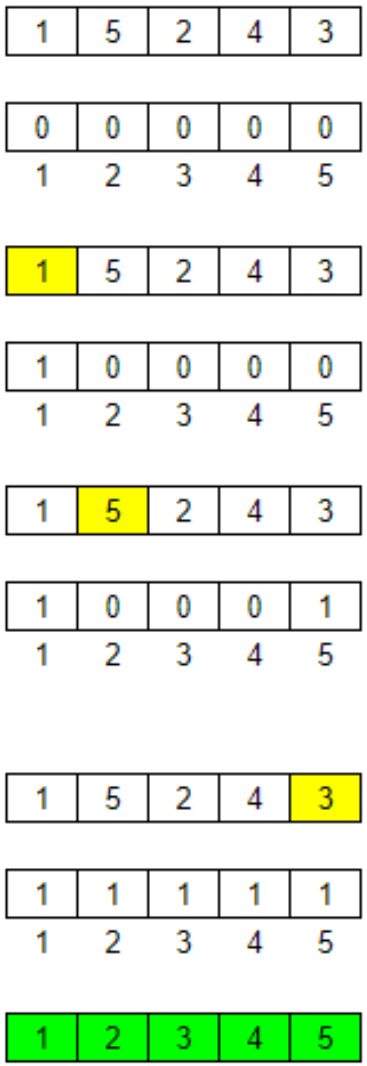


- $O(n^2)$
- Устойчива
- Естественная ($O(n)$ на отсортированном массиве)
- Данные могут приходить постепенно
- Позволяет выбрать наибольшие (или наименьшие) k чисел из входного потока
- https://cathyatseneca.github.io/DSAnim/web/insertion.html

Сортировка Шелла (Shell sort)

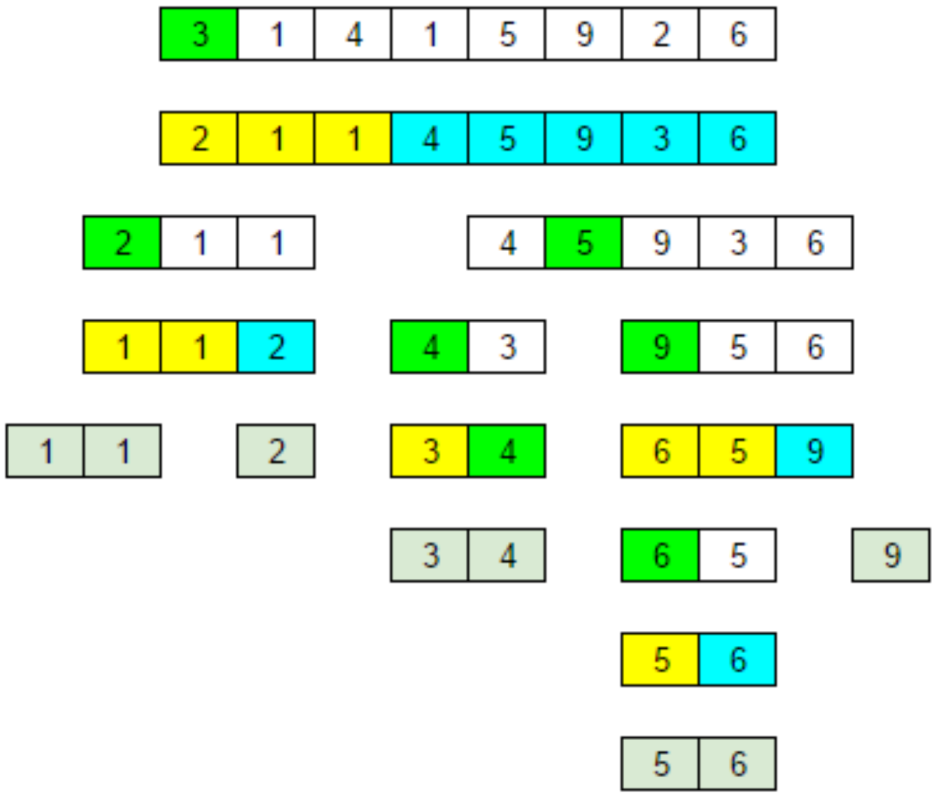
- Сортировка вставкой подпоследовательностей в массиве с постепенно убывающим шагом
- Элементы “быстрее” встают на свои места
  - Сортировка вставкой на каждом шаге уменьшает количество инверсий максимум на 1
- $O(n * log(n)^2)$ при правильном выборе h
- Неустойчива
- Легко пишется и довольно быстра
  - Не вырождается до квадратичной
- http://www.programming-algorithms.net/article/41653/Shell-sort

Сортировка подсчётом (counting sort)



- $O(n)$
- Сортирует только числа, так что устойчивость неприменима
- Отсортированность массива ничего не даёт
- Требует заранее знать диапазон чисел в массиве
- https://www.cs.usfca.edu/~galles/JavascriptVisual/CountingSort.html

Быстрая сортировка (qsort)



- $O(n * log(n))$, вырождается до $O(n^2)$
- Неустойчива
- Требует $O(n * log(n))$ дополнительной памяти
- Самый быстрый на практике алгоритм сортировки, используется в стандартных библиотеках
- Легко пишется (но тяжело отлаживается)
- https://cathyatseneca.github.io/DSAnim/web/quick.html

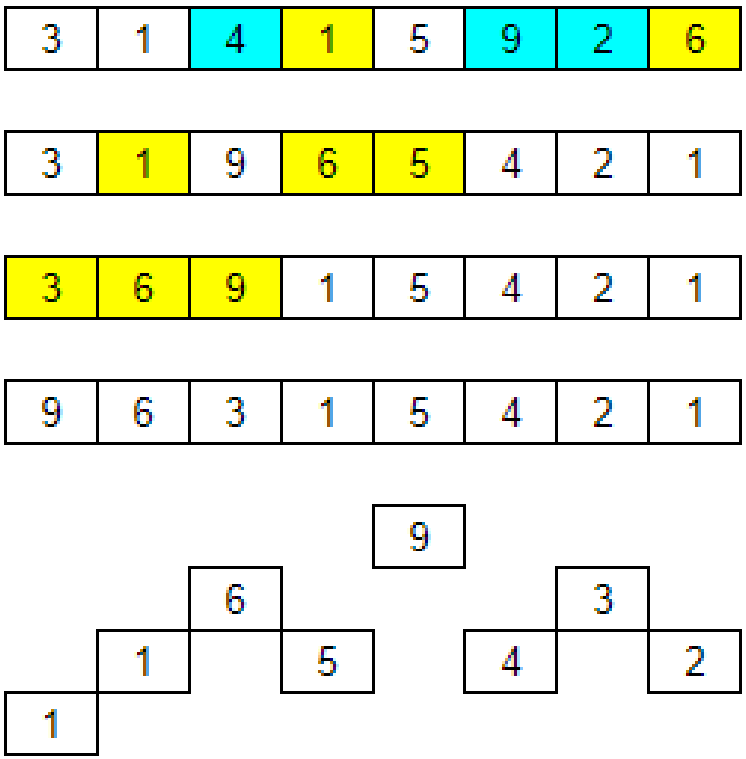
Псевдокод

```
algorithm quicksort(A, lo, hi) is
    if lo < hi then
        p := partition(A, lo, hi)
        quicksort(A, lo, p – 1)
        quicksort(A, p + 1, hi)
```

```
algorithm partition(A, lo, hi) is
    pivot := A[hi]
    i := lo
    for j := lo to hi – 1 do
        if A[j] ≤ pivot then
            swap A[i] with A[j]
            i := i + 1
    swap A[i] with A[hi]
    return i
```

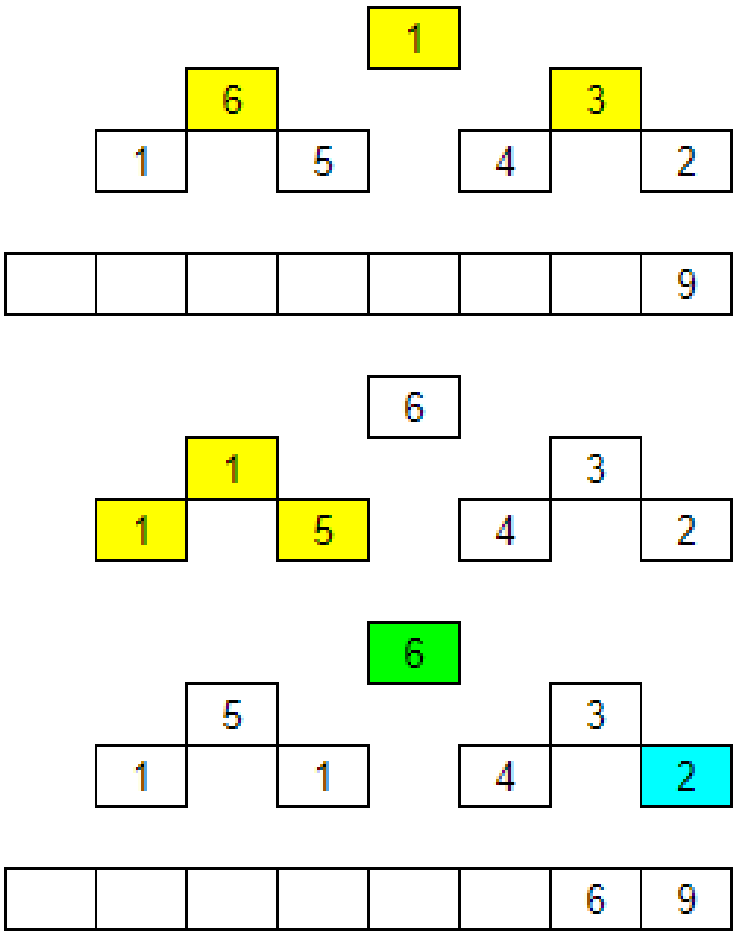
Нерекурсивная реализация — через стек, в котором хранятся границы сортируемых кусков массива

Сортировка кучей (пирамидальная, heapsort)



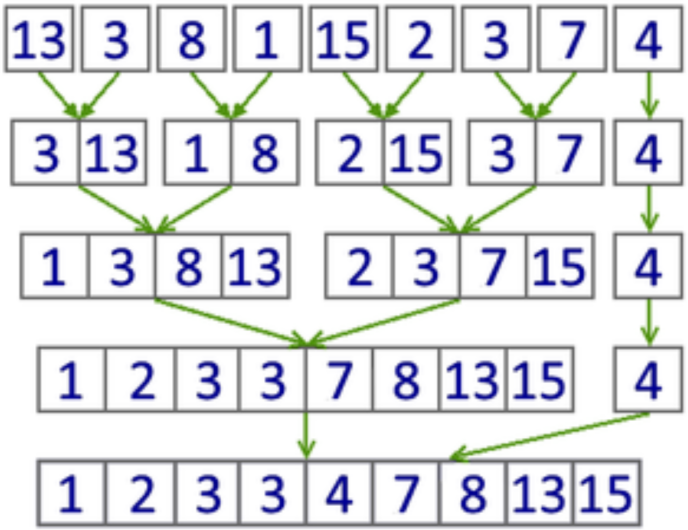
- $O(n * log(n))$, не вырождается
- Не требует дополнительной памяти
- Неустойчива
- Требует произвольного доступа к памяти
- Относительно сложна в реализации

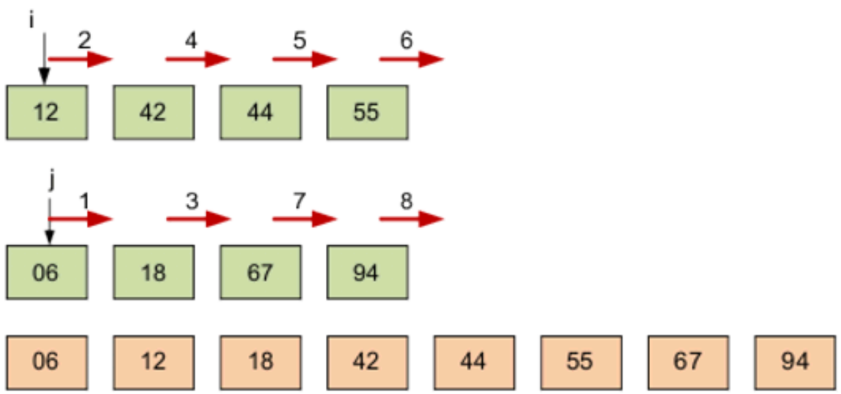
Сортировка кучей, сама сортировка



- Меняем первый элемент (корень кучи) с последним
- Проталкиваем новый корень вглубь кучи, восстанавливая её структуру
- https://www.ee.ryerson.ca/~courses/coe428/sorting/heapsort.html

Сортировка слиянием (mergesort)





- $O(n * log(n))$, не вырождается
- Устойчива
- Внешняя (подходит для больших данных, не помещающихся в память)
- https://cathyatseneca.github.io/DSAnim/web/merge.html

Двоичный поиск

- Находит элемент в массиве за $O(log(n))$
- Легко напутать с индексами и уйти в бесконечный цикл