Сортировки

Гусев Илья, Булгаков Илья

Московский физико-технический институт

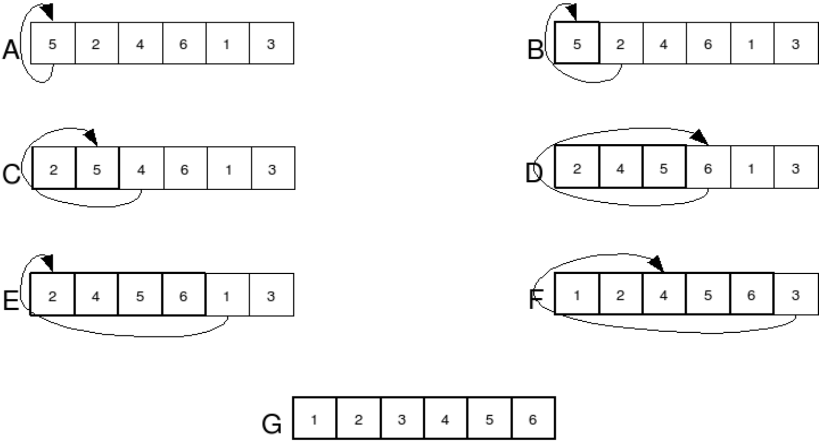
Москва, 2018

Содержание

Задача сортировки

Пусть требуется упорядочить N элементов: $R_1, R_2, \ldots, R_n$.
K - K - ключ сортировки, $\forall j \in 1 \ldots n, K_j \in R_j$
$\forall a, b, c \in K \rightarrow (a < b) \vee (b > a) \vee (a = b)$
$\forall a, b, c \in K \rightarrow (a < b) \wedge (b < c) \Rightarrow (a < c)$
Найти $p(1)p(2)\ldots p(n)$ т.ч. $K_{p(1)} \leq K_{p(2)} \leq \cdots \leq K_{p(n)}$
Устойчивая (стабильная) перестановка: $\forall i < j, K_{p(i)} = K_{p(j)} \rightarrow p(i) < p(j)$

Сортировка вставками



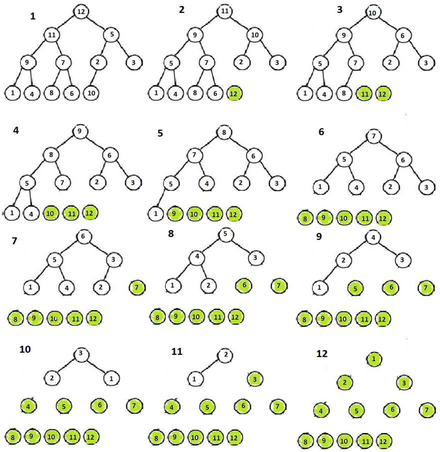
Сложность? Устойчивость? Доп. память? Сложность на уже сортированных массивах?

Сортировка вставками

Код

```
template <class T>
void insertion_sort(std::vector<T>& collection) {
    for (size_t i = 1; i < collection.size(); i++) {
        T key = collection[i];
        size_t j = i - 1;
        while (j >= 0 && collection[j] > key) {
            collection[j + 1] = collection[j];
            j -= 1;
        }
        A[j+1] = key;
    }
}
```

Пирамидальная сортировка (HeapSort)

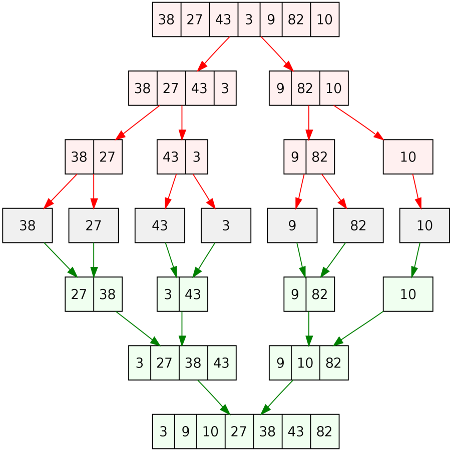


1. Строим над коллекцией кучу
2. Делаем ExtractMin n раз (минимум перемещаем в конец)
3. ...
4. PROFIT!

Сложность? Устойчивость? Доп. память? Сложность на уже сортированных массивах?

Сортировка слиянием (MergeSort)

Recursive

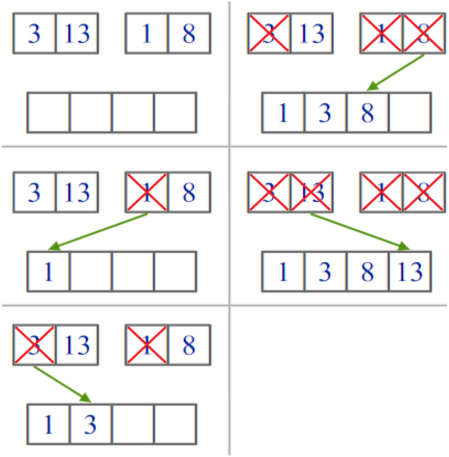


Рекурсивный вариант

1. Сортируемый массив разбивается на две части примерно одинакового размера
2. Каждая из получившихся частей сортируется отдельно, например — тем же самым алгоритмом
3. Два упорядоченных массива половинного размера соединяются в один

Сортировка слиянием (MergeSort)

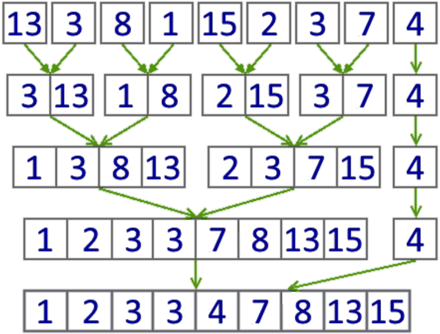
Merge



- Время работы процедуры: $\Theta(m)$, где m - суммарное количество входных данных
- Суммарно для всех вызовов на одном уровне: $\Theta(n)$, где n - количество элементов коллекции

Сортировка слиянием (MergeSort)

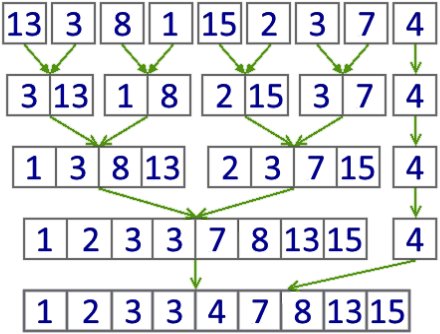
Iterative



- Альтернатива: итеративный алгоритм
- Сложность? Устойчивость? Доп. память? Сложность на уже сортированных массивах?

Сортировка слиянием (MergeSort)

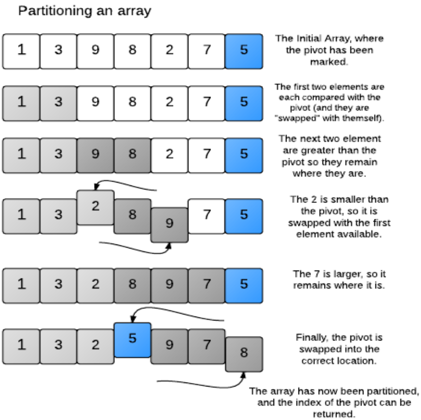
Iterative



- Альтернатива: итеративный алгоритм
- Сложность? Устойчивость? Доп. память? Сложность на уже сортированных массивах?

Быстрая сортировка (QuickSort)

Partition



- Сделать Partition коллекции
- Partition - берём опорный элемент, а остальные элементы делим на две части: меньше опорного и больше или равные опорному. $\Theta(n)$
- Для обоих частей рекурсивно выполняем Partition
- Тип Partition: Хоара или Ломуто

Быстрая сортировка (QuickSort)

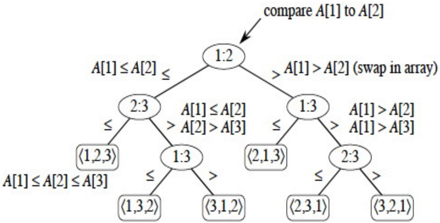
Сложность? Устойчивость? Доп. память? Сложность на уже сортированных массивах?
Модификации:

- Устойчивость через второй ключ-индекс
- Выбор опорного элемента: первый, последний, средний, медианный из 3, случайный
- Разбиение на 3 части

Сравнение сортировок

Алгоритм	Худшее	Лучшее	В среднем	Sorted	Уст.	+ память
Insertion	$\Theta(n^2)$	$\Theta(n)$	$\mathcal{O}(n^2)$	$\Theta(n)$	Да	$\Theta(1)$
Heap	$\Theta(nlog(n))$	$\Theta(nlog(n))$	$\Theta(nlog(n))$	$\Theta(nlog(n))$	Нет	$\Theta(1)$
Merge	$\Theta(nlog(n))$	$\Theta(nlog(n))$	$\Theta(nlog(n))$	$\Theta(nlog(n))$	Да	$\Theta(n)$
Quick	$\Theta(n^2)$	$\Theta(nlog(n))$	$\Theta(nlog(n))$	$\Theta(n^2)$	Нет	$\Theta(1)$

Доказательство $\Omega(nlog(n))$ для сортировок сравнениями



- $count(leaves) = l \geq n!$
- $l \leq 2^h$
- $n! \leq l \leq 2^h \Rightarrow log_2(n!) \leq h$
- $n! > (\frac{n}{e})^n$
  - $1 > \frac{1}{e}$
  - $(\frac{n+1}{e})^{n+1} = (\frac{n}{e})^n \frac{(n+1)^{n+1}}{n^n e} = (\frac{n}{e})^n (1 + \frac{1}{n})^n \frac{n+1}{e} > n!(n+1) = (n+1)!$
- $h \geq log_2(\frac{n}{e})^n = n \cdot log_2(\frac{n}{e}) = \Omega(n \cdot log(n))$

Полезные ссылки I

- ICS 311 #10: Theoretical Limits of Sorting, O(n) Sorts
  https://www2.hawaii.edu/ñodari/teaching/s17/Notes/Topic-10.html
- Wiki - Sorting algorithm
  https://en.wikipedia.org/wiki/Sorting_algorithm
- Викиконспекты: Сортировка слиянием
  https://neerc.ifmo.ru/wiki/index.php?title=Сортировка