Сложность алгоритмов

Сложность алгоритмов

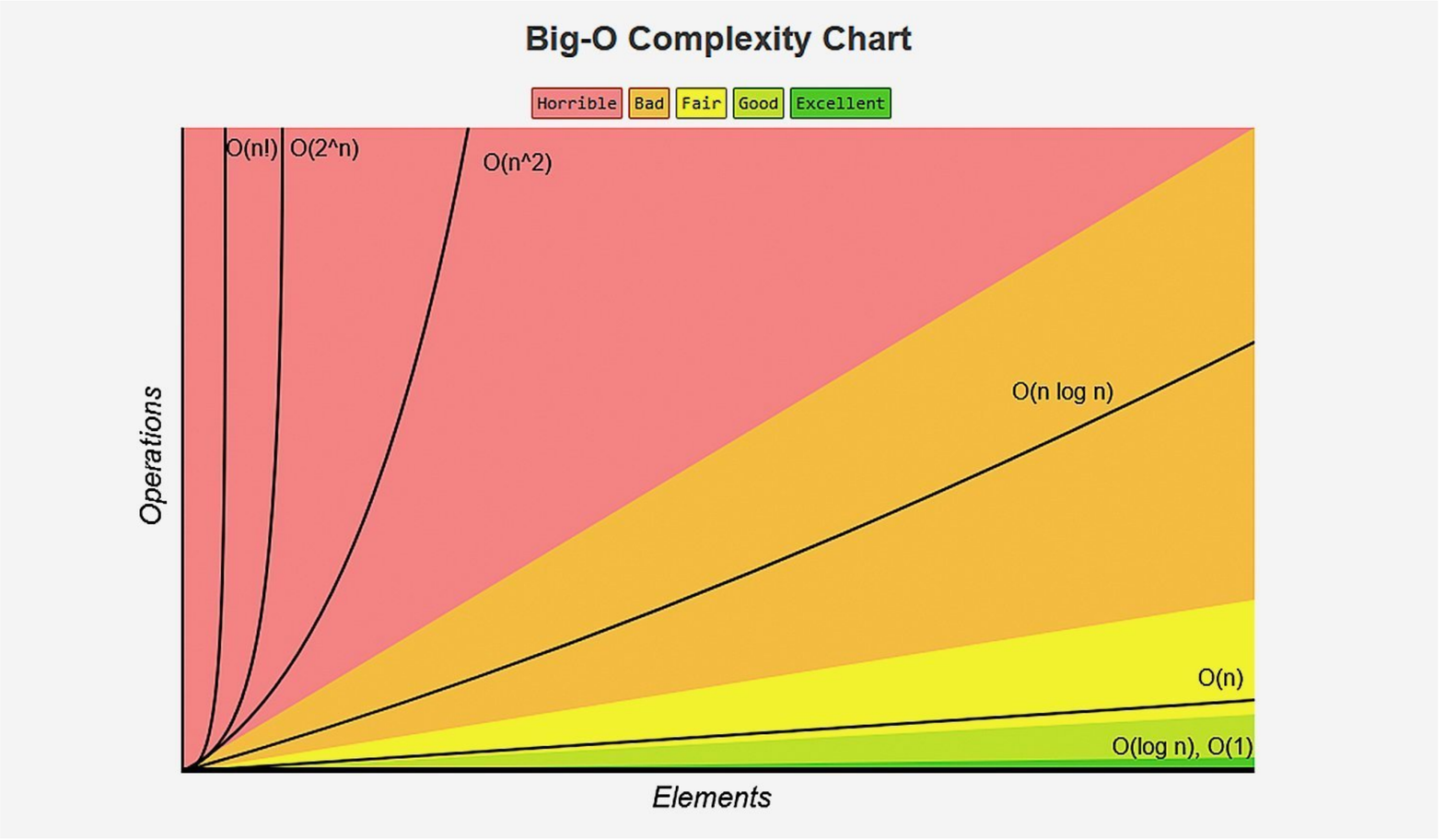
Основная характеристика алгоритма - временная сложность.
Также учитывается выделение памяти при работе алгоритма.
Измеряются в нотации O(n)

Сложность алгоритмов

Нотация Big O:

- **O(1): Константная сложность**. Время выполнения алгоритма не зависит от размера входных данных.
- **O(log n)**: **Логарифмическая сложность**. Время выполнения алгоритма растет медленно с увеличением размера входных данных.
- **O(n)**: **Линейная сложность**. Время выполнения алгоритма пропорционально размеру входных данных.
- **O(n log n)**: **Линейно-логарифмическая сложность**. Время выполнения алгоритма растет быстрее, чем линейно, но медленнее, чем квадратично.
- **O(n^2)**: **Квадратичная сложность**. Время выполнения алгоритма зависит от квадрата размера входных данных.
- **O(n^3): Кубическая сложность**. Время выполнения алгоритма зависит от размера входных данных в кубе.
- **O(n!): Факториальная сложность**. Это самая высокая степень роста времени выполнения алгоритма. Время выполнения алгоритма растет факториально размера входных данных.

Сложность алгоритмов

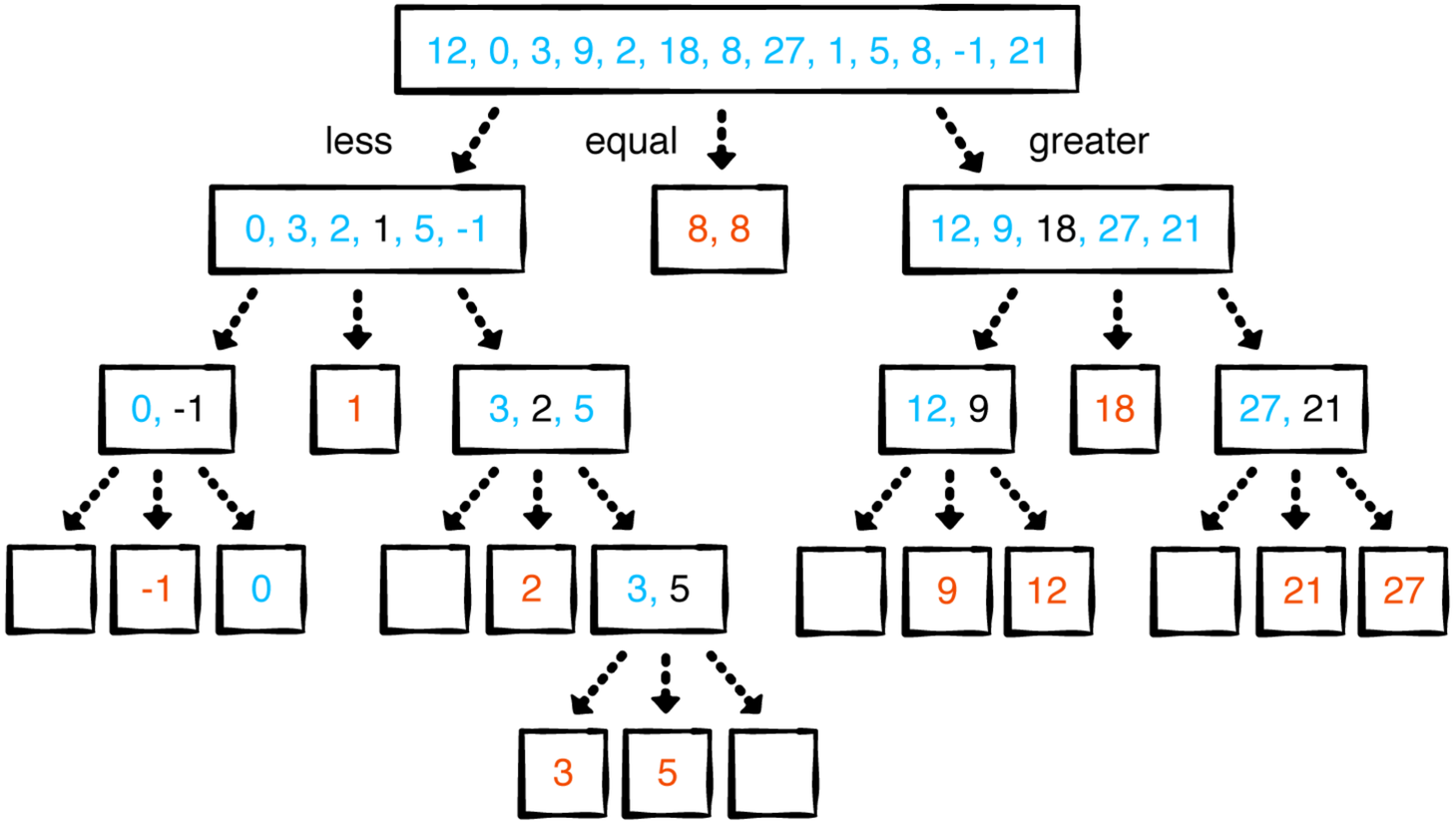


Алгоритмы сортировки массива

Алгоритмы сортировки массива

Существует множество алгоритмов сортировки: https://en.wikipedia.org/wiki/Sorting_algorithm.
Самый быстрый алгоритм среди устойчивых (сохраняет порядок элементов с разными ключами) - Merge sort.
В Go применяется разновидность алгоритма Quick sort.

Quick sort


12, 0, 3, 9, 2, 18, 8, 27, 1, 5, 8, -1, 21
less
equal
greater
0, 3, 2, 1, 5, -1
8, 8
12, 9, 18, 27, 21
0, -1
1
3, 2, 5
12, 9
18
27, 21
-1
0
2
3, 5
9
12
21
27
3
5

Алгоритмы поиска в массиве

Алгоритмы поиска элемента в массиве

Основные алгоритмы:

- Линейный поиск - для неотсортированного массива. Временная сложность - O(n). Затраты памяти O(1).
- Бинарный поиск - для отсортированного массива. Временная сложность - O(log n). Затраты памяти O(1).

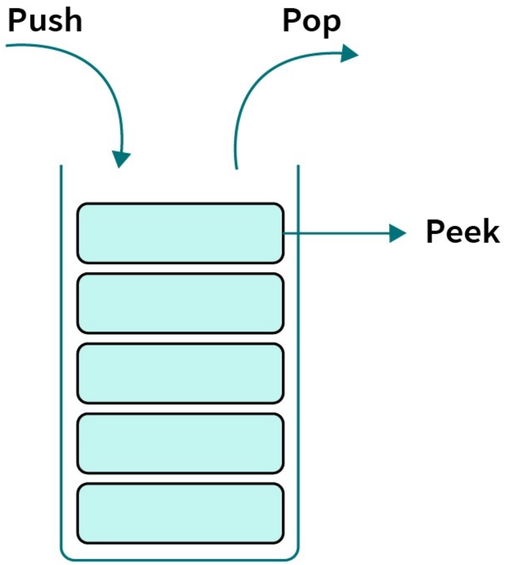
Массивы

Массивы


Элемент
(с индексом 8)
Первый индекс
0
1
2
3
4
5
6
7
8
9
Индексы
Длина массива = 10

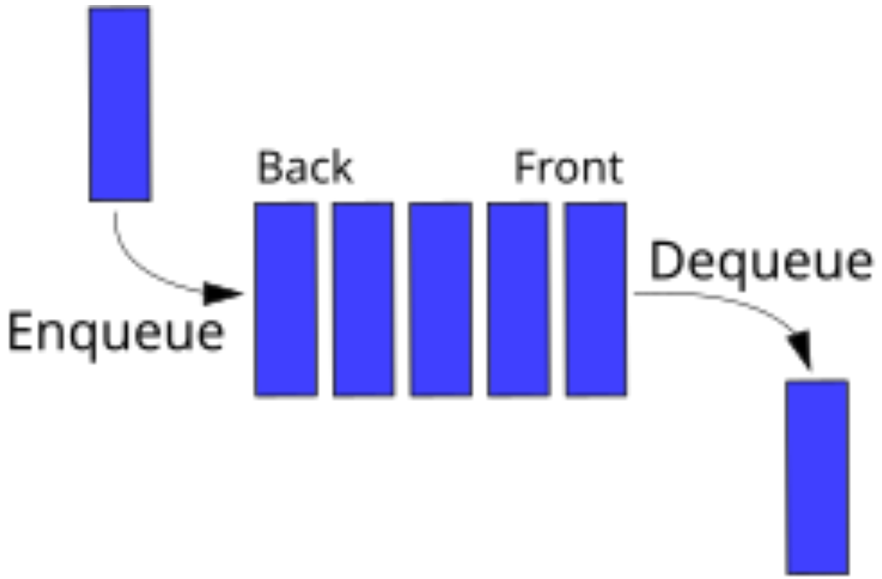
Стек

Стэк



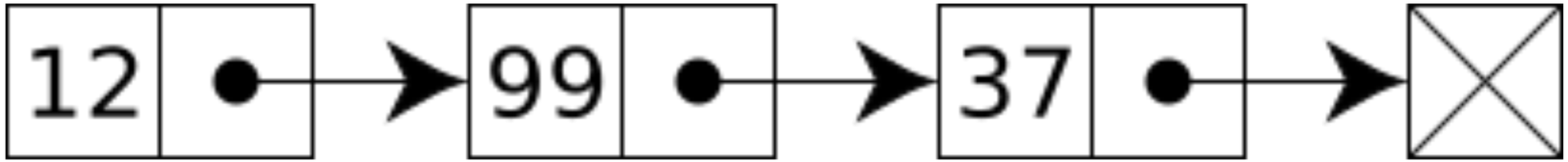
Очередь

Очередь



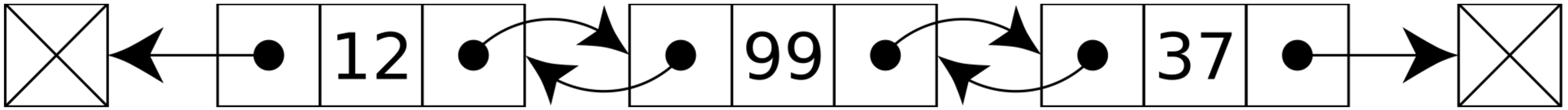
Односвязные списки

Односвязные списки



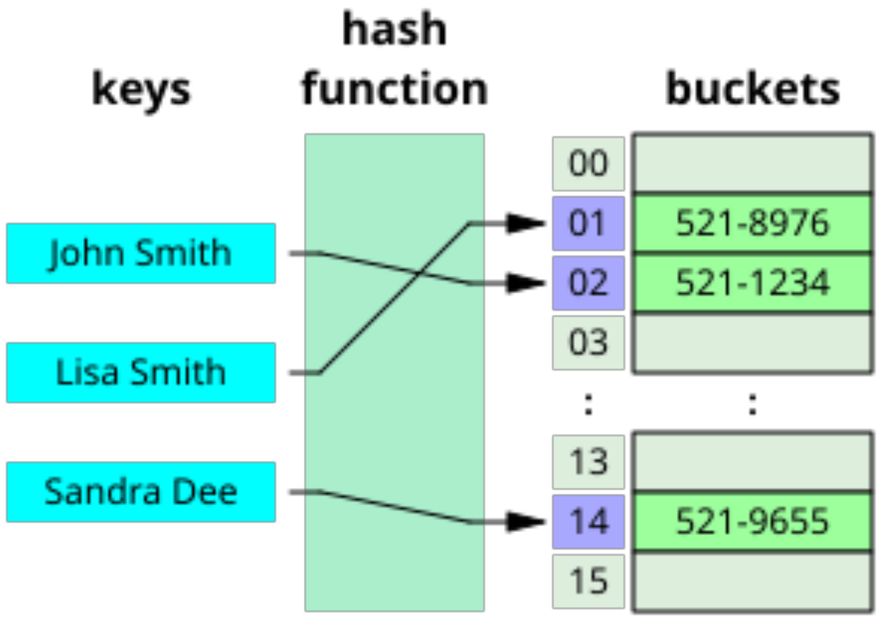
Двусвязные списки

Двусвязные списки



Хеш-таблица

Хеш-таблицы


keys
hash function
buckets
John Smith
Lisa Smith
Sandra Dee
00
01
02
03
13
14
15
521-8976
521-1234
521-9655

Хеш-таблицы

Hash table

Type	Unordered associative array
Invented	1953

Time complexity in big O notation

Algorithm	Average	Worst case
Space	O(*n*)[1]	O(*n*)
Search	O(1)	O(*n*)
Insert	O(1)	O(*n*)
Delete	O(1)	O(*n*)

Сложность операций с map

Множество (набор)

Множество

Множество - абстрактная структура данных, хранящая набор уникальных элементов.
Обычно реализовывается на основе хеш-таблицы, сложность операций при этом сохраняется.

Сравнение сложности операций

Data Structure	Time Complexity							
	Average				Worst			
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion
Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)	O(1)	O(n)	O(n)	O(n)
Stack	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)
Queue	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)
Singly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)
Doubly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)
Hash Table	N/A	Θ(1)	Θ(1)	Θ(1)	N/A	O(n)	O(n)	O(n)
Binary Search Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)