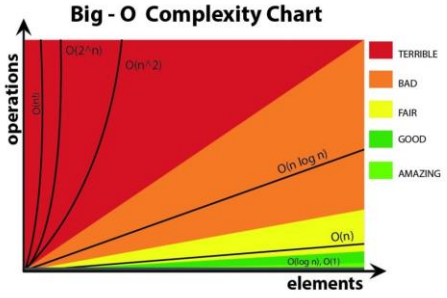
Big O: сложность основных операций и оценка алгоритмов

интерфейс
класс

Розовый – справедливо, оправдано: O(n) или линейное время (n - это количество эл. коллекции)
Жёлтый – хорошо O(log n) - деревья
Зелёный – превосходно! O(1) или константное время

		сруктура данных, особенность	**Временная сложность BigO**							
Iterable										
Collection			Среднее				Худшее			
List	упорядоченный **список** с дубликатами		Индекс	Поиск	Вставка	Удаление	Индекс	Поиск	Вставка	Удаление
	ArrayList	динамический массив объектов	O(1)	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(n)
FIFO|LIFO	LinkedList	дву- одно- связанный список	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)
	Vector	синхронизированный, устарел	O(1)	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(n)
LIFO	Stack									
Set	НЕ упорядоченное **множество** уникальных									
	HashSet	на основе **хэш-таблицы**	n/a	O(1)	O(1)	O(1)	n/a	O(n)	O(n)	O(n)
	LinkedHashSet	на основе **связанного списка**	n/a	O(1)	O(1)	O(1)	n/a	O(n)	O(n)	O(n)
	SortedSet	упорядоченный по возрастанию								
	NavigableSet	добавлена навигация								
	TreeSet	**дерево**	n/a	O(log (n))	O(log (n))	O(log (n))	n/a	O(log(n))	O(log(n))	O(log(n))
Queue	FIFO **очередь**	вход - хвост, выход - голова								
FIFO	PriorityQueue	Очередь, выход по приоритету								
FIFO|LIFO	Deque	LIFO/FIFO очередь двусторонняя								
	ArrayDeque	имеет реализацию кольцевой буфер								
ОТДЕЛЬНАЯ ВЕТВЬ, не наследуется от Collection										
Map<K, V>	**карта** пар ключ-значение (еще говорят "словарь")									
	SortedMap									
	NavigableMap									
			Среднее				Худшее			
	AbstractMap class		Индекс	Поиск	Вставка	Удаление	Индекс	Поиск	Вставка	Удаление
	HashMap	массив бакетов + LinkedList (1св) при перестроении КЧ дерево	n/a	O(1)	O(1)	O(1)	n/a	O(n)	O(n)	O(n)
	LinkedHashMap	массив бакетов + LinkedList (2св) **поддержка двусвязного списка**	n/a	O(1)	O(1)	O(1)	n/a	O(n)	O(n)	O(n)
	WeekHashMap	Слабые ключи, элемент удаляется GC								
	TreeMap	дерево с возможностью навигации	n/a	O(log (n))	O(log (n))	O(log (n))	n/a	O(log(n))	O(log(n))	O(log(n))



Вот обычный порядок возрастания времени:

- O(1) — постоянное время
- O(log n) — логарифмическое время
- O(n) — линейное время
- O(n²) — квадратичное время
- O(2^n) — экспоненциальное время
- O(n!) — факториальное время

Big O – это мера эффективности «в худшем случае», т.е. верхняя граница того, сколько времени потребуется **для выполнения задачи,** или сколько памяти для этого необходимо.

Оценка алгоритмов

Временная сложность обычно оценивается путём подсчёта числа элементарных операций, осуществляемых алгоритмом.

BIG-O	Сложность	Пример
O (1)	Константная	Поиск по ключу в хэш-таблице; арифметическая операция с числом
O (log2 (n))	Логарифмическая	Бинарный поиск, сложность, вставка сбалансированное бинарное дерево
O (n)	Линейная	Поиск перебором; среднеквадратическое отклонение
O (n*log2(n))	Квазилинейное	Самые быстрые алгоритмы сортировки
O (n2)	Квадратичная	Простые алгоритмы сортировки; перемножение n-значных чисел «столбиком»
O(nx)	Полиномиальная	LU-разложение матрицы; мощность графа, базовые матем операции
O (cn)	Экспоненциальная	Задача коммивояжёра (динамическое программирование)
O (n!)	Факториальная	Задача коммивояжёра перебором

Алгоритм — конечная совокупность точно заданных правил решения **типовых задач** или набор инструкций, описывающих порядок действий исполнителя для решения задачи.