

# SKILLFACTORY

## Fullstack разработчик на Python

## Модуль: Алгоритмы и структуры данных

### Сложность алгоритма

<b>"O"</b> большое	верхняя граница сложности
<b>Θ (Тета)</b>	точная оценка сложности
<b>Ω (Омега)</b>	нижняя граница сложности

### Сложность алгоритма

<b>O(1)</b>	константная
<b>O(log(n))</b>	логарифмическая
<b>O(n)</b>	линейная
<b>O(n log(n))</b>	линейно-логарифмическая
<b>O(n^k)</b>	полиномиальная
<b>O(n!)</b>	факториальная

### Операции с массивами

получение размера	<b>O(1)</b>
элемент по индексу	<b>O(1)</b>
вставка в конец/ удаление с конца	<b>O(1)</b>
вставка в конец (с расширением)	<b>O(n)</b>
вставка/удаление (с произвольного места)	<b>O(n)</b>

### Хэш-таблицы

вставка пары	<b>O(1)</b>
удаление пары	<b>O(1)</b>
получение значения по ключу	<b>O(1)</b>
получение размера	<b>O(n)</b>

### Односвязный список

вставка в конец/начало	<b>O(1)</b>
вставка на произвольное место	<b>O(1)</b>
удаление из начала	<b>O(1)</b>
удаление с произвольного места	<b>O(n)</b>
получение размера списка	<b>O(n)</b>

### Стек

вставка наверх (push)	<b>O(1)</b>
удаление сверху (pop)	<b>O(1)</b>
верхний элемент (top)	<b>O(1)</b>
размер (size)	<b>O(1)</b>

### Очередь на циклическом массиве

вставка в конец	<b>O(1)</b>
удаление из начала	<b>O(1)</b>
получение первого элемента	<b>O(1)</b>
общий размер	<b>O(1)</b>

### Графы

наличие ребра	<b>O( E )</b>	<b>O(1)</b>
степень вершины	<b>O(1)</b>	<b>O( V )</b>
память	<b>O( V + E )</b>	<b>O( V ^2)</b>
вставка/ удаление	<b>O(1)</b>	<b>O(d)</b>
обход графа	<b>O( V + E )</b>	<b>O( V ^2)</b>
алгоритм Дейкстры (время)	<b>O( V ^2+ E )</b>	
алгоритм Дейкстры (доп.память)	<b>O( V )</b>	

|V| - количество вершин

|E| - количество ребер

d - степень вершины

# SKILLFACTORY

## Fullstack разработчик на Python

## Модуль: Алгоритмы и структуры данных

### Алгоритм Дейкстры

<b>Цель: поиск минимального пути в графе</b>
1. Создание словаря расстояний D
2. Создание словаря просмотренных вершин U
3. Выбор вершины с наименьшим d
4. Во все непросмотренные вершины записать расстояния до них
5. Повтор п.3-4 $ V $ раз

### Бинарные деревья

<b>Бинарное дерево - это граф, обладающий свойствами:</b>
1. связный (нет вершин без ребер)
2. не имеет циклов
3. неориентированный
4. невзвешенный
5. имеет не более 2 потомков

### Виды обхода деревьев

обход в глубину (DFS)	префиксный ( <b>pre-order</b> )
	инфиксный ( <b>in-order</b> )
	постфиксный ( <b>post-order</b> )
обход в ширину (BFS)	

### Алгоритмы поиска

линейный поиск	$O(n)$
двоичный поиск	$O(\log(n))$
поиск в графе	$O(n)$
двоичное дерево поиска	$O(\log(n))$
поиск в хэш-таблице	$O(1)$

### Сортировка пузырьком

<b>"всплытие" максимума вправо</b>	
Лучший случай	$O(n)$
Средняя оценка	$O(n^2)$
Худший случай	$O(n^2)$

### Сортировка вставками

<b>поиск положения в отсортированной части</b>	
Лучший случай	$O(n)$
Средняя оценка	$O(n^2)$
Худший случай	$O(n^2)$

### Сортировка слиянием

<b>"разделяй и властвуй" деление на части и слияние с сортировкой</b>	
Лучший случай	$O(n \log(n))$
Средняя оценка	$O(n \log(n))$
Худший случай	$O(n \log(n))$

### Быстрая сортировка

<b>"разделяй и властвуй" деление массива на части относительно опорного элемента</b>	
Лучший случай	$O(n \log(n))$
Средняя оценка	$O(n \log(n))$
Худший случай	$O(n^2)$