# Лекция 11. Двоичные (бинарное) деревья

**Казахский национальный университет имени аль-Фараби**

Кафедра компьютерных наук

Алгоритмы и структуры данных

Лектор: старший преподаватель, PhD, Кенжебек Ержан Ғалымжанұлы

# Цель и план лекции

**Цель лекции**

Изучить древовидные структуры данных, терминологию и виды бинарных деревьев, освоить способы обхода дерева и вставку узла.

**План лекции**

1. Дерево как структура данных, терминология
2. Представление бинарного дерева
3. Типы бинарных деревьев
4. Обходы: inorder, preorder, postorder
5. Свойства бинарного дерева, вставка узла

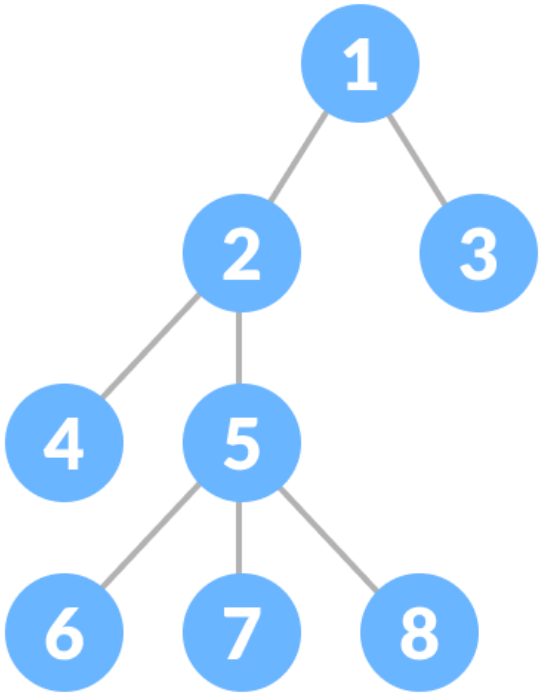
# Что такое бинарное дерево?

- Бинарное дерево — это древовидная структура данных, в которой **каждый узел может иметь не более двух дочерних элементов**, которые называются левым дочерним элементом и правым дочерним элементом.
- Самый верхний узел двоичного дерева называется корнем, а самые нижние узлы — листьями. Бинарное дерево можно представить как иерархическую структуру с корнем вверху и листьями внизу.

- Двоичные деревья имеют множество применений в информатике, включая хранение и извлечение данных, оценку выражений, сетевую маршрутизацию и игровой искусственный интеллект. Их также можно использовать для реализации различных алгоритмов, таких как алгоритмы поиска, сортировки и графов.

# Основные понятия: древовидная структура данных

- Дерево — это нелинейная иерархическая структура данных, состоящая из узлов, соединенных ребрами.



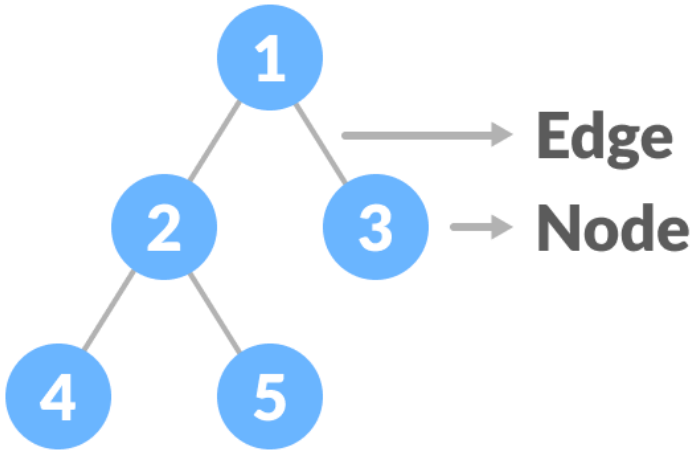
# Почему древовидная структура данных?

- Другие структуры данных, такие как массивы, связанный список, стек и очередь, представляют собой линейные структуры данных, в которых данные хранятся последовательно. Для выполнения какой-либо операции с линейной структурой данных временная сложность увеличивается с увеличением размера данных. Но это неприемлемо в современном вычислительном мире.

- Различные древовидные структуры данных обеспечивают более быстрый и простой доступ к данным, поскольку это нелинейная структура данных.

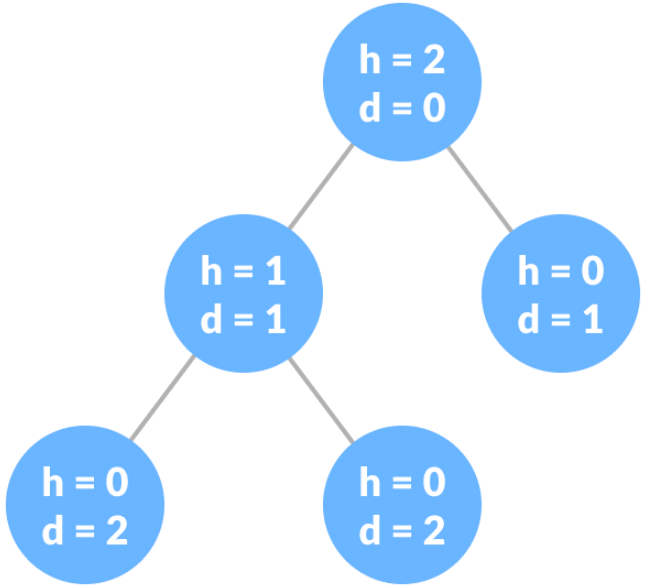
# Древовидная терминология

- **Узел (Node)**
  - Узел — это объект, который содержит ключ или значение и указатели на его дочерние узлы.
  - Последние узлы каждого пути называются **листовыми узлами или внешними узлами** , которые не содержат ссылки/указателя на дочерние узлы.
  - Узел, имеющий хотя бы дочерний узел, называется **внутренним узлом**
- **Ребра (edge)** - это связь между любыми двумя узлами.



# Древовидная терминология #2

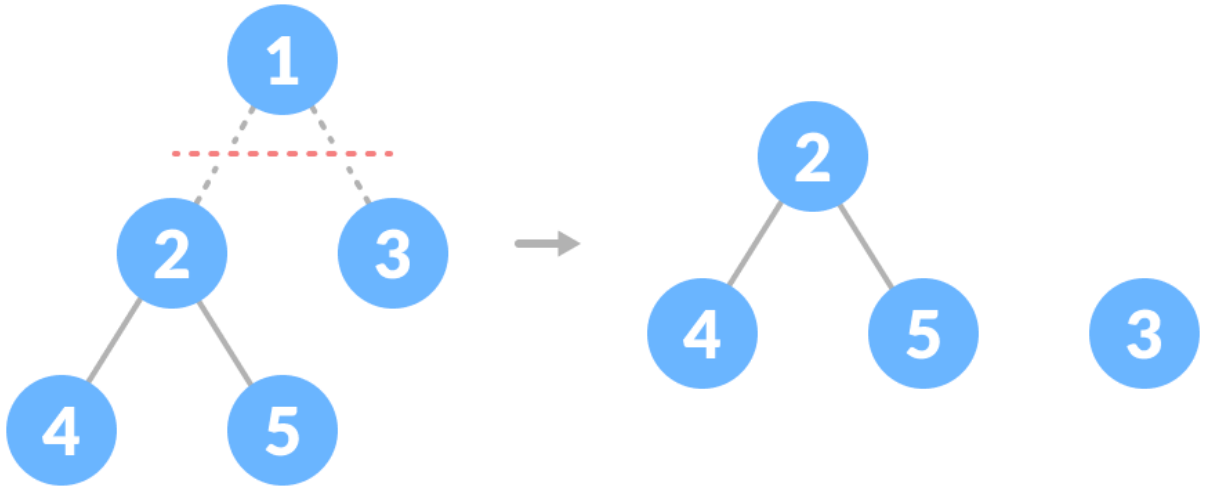
- **Корень**
  - Это самый верхний узел дерева.
- **Высота узла**
  - Высота узла — это количество ребер от узла до самого глубокого листа (т. е. самый длинный путь от узла до листового узла).
- **Глубина узла**
  - Глубина узла — это количество ребер от корня до узла.
- **Высота дерева**
  - Высота дерева — это высота корневого узла или глубина самого глубокого узла.




**Высота и глубина каждого узла в дереве**

# Древовидная терминология #3

- **Лес**
  - Совокупность непересекающихся деревьев называется лесом.
- Вы можете создать лес, срезав корень дерева.

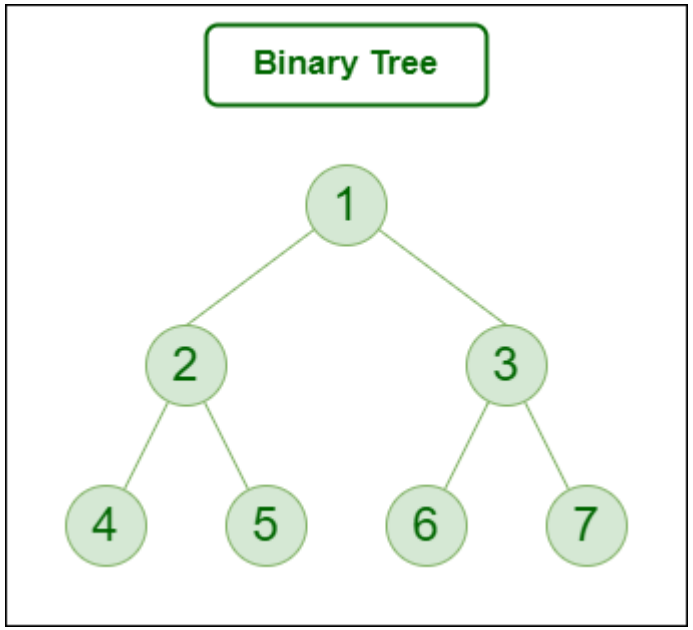


# Виды деревьев

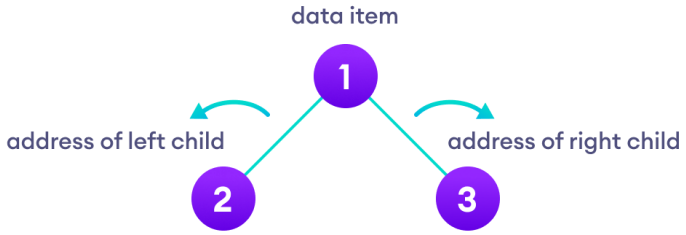
- **Двоичное дерево (Binary Tree)**
- Двоичное дерево поиска (Binary Search Tree)
- Сбалансированные двоичные деревья (AVL, Red-Black Tree)

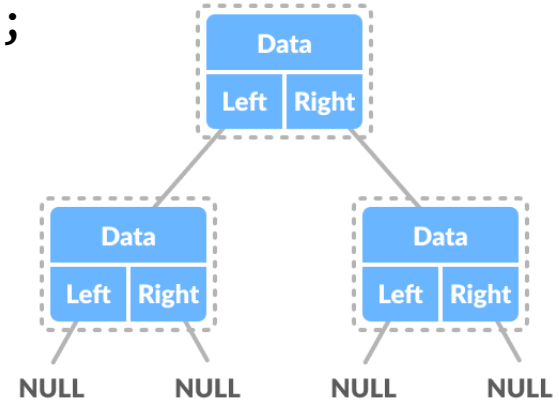
# Представление двоичного дерева

- Каждый узел дерева содержит следующее:
  - Данные
  - Указатель на левого дочернего элемента
  - Указатель на нужного дочернего элемента




```
class Node {
public:
    int data;
    Node* left;
    Node* right;
};
```



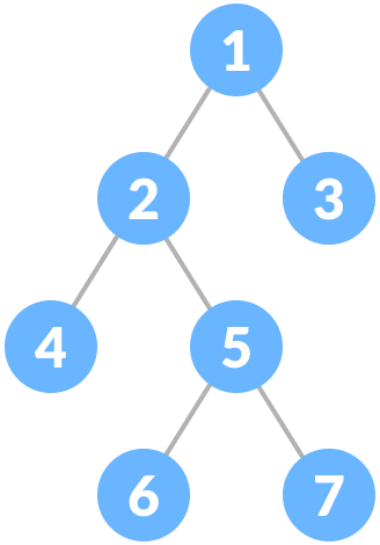



# Типы двоичного дерева

1. Полное двоичное дерево (**Full Binary Tree**)
2. Идеальное двоичное дерево (**Perfect Binary Tree**)
3. Complete binary tree
4. Вырожденное или патологическое дерево. (**Degenerate or Pathological Tree**)
5. Искаженное двоичное дерево (**Skewed Binary Tree**)
6. Сбалансированное двоичное дерево (**Balanced Binary Tree**)

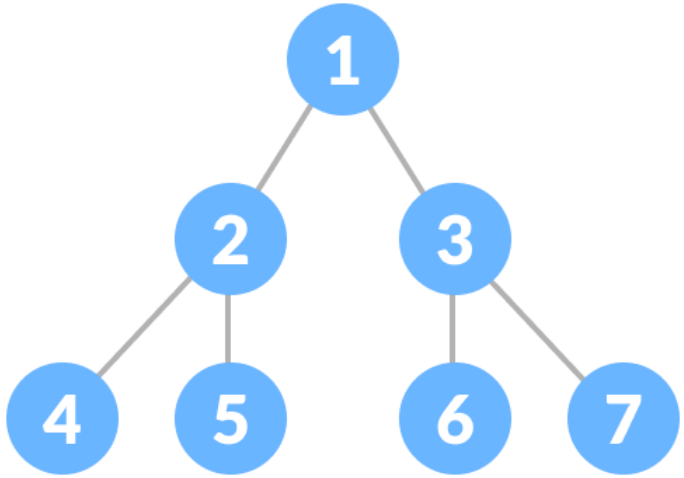
# 1. Полное двоичное дерево

- 1. **Полное двоичное дерево (Full Binary Tree)** — это особый тип двоичного дерева, в котором каждый родительский узел/внутренний узел имеет либо два дочерних узла, либо не имеет их ни одного.



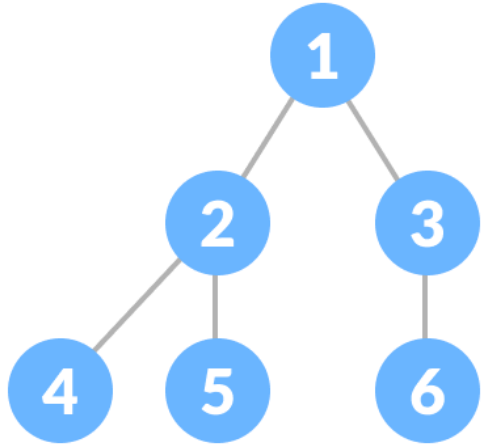
# 2. Идеальное двоичное дерево

- **Идеальное двоичное дерево (Perfect Binary Tree)** — это тип двоичного дерева, в котором каждый внутренний узел имеет ровно два дочерних узла, а все листовые узлы находятся на одном уровне.



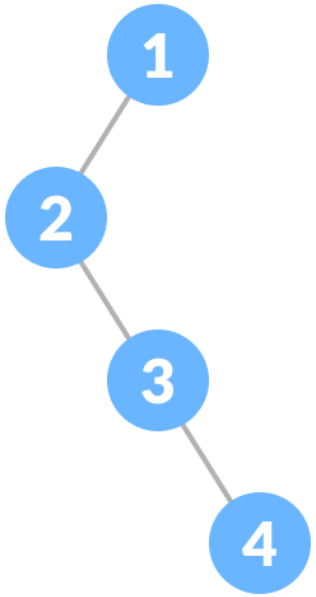
# 3. Complete (Полное) Binary Tree

- **Complete двоичное дерево** похоже на полное двоичное дерево, но с двумя основными отличиями.
  - Каждый уровень должен быть полностью заполнен
  - Все элементы листа должны наклоняться влево.
  - Последний листовой элемент может не иметь правого родственного элемента, т. е. Complete binary tree не обязательно должно быть полным двоичным деревом.



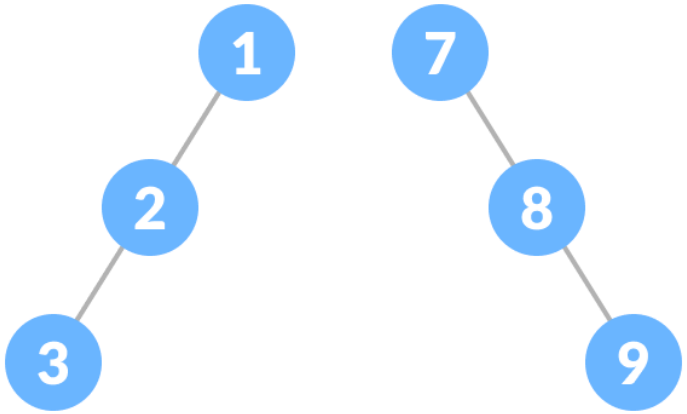
# 4. Вырожденное или патологическое дерево. (**Degenerate or Pathological Tree**)

- Вырожденное или патологическое дерево — это дерево, имеющее единственного дочернего элемента слева или справа.



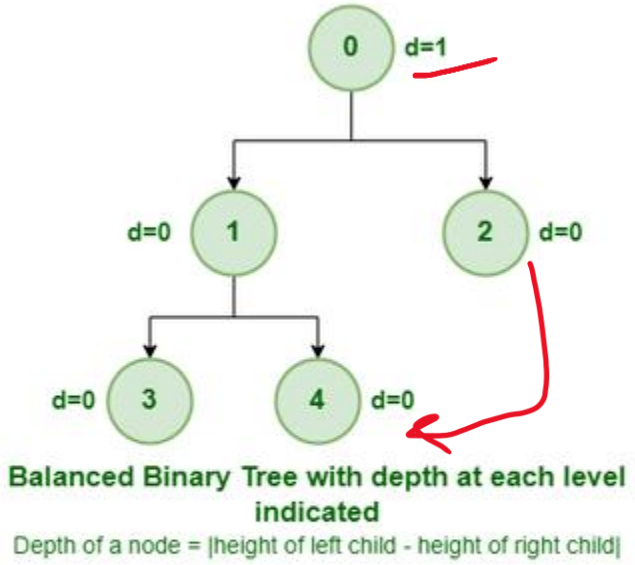
# 5. Искаженное двоичное дерево (**Skewed Binary Tree**)

- Искаженное двоичное дерево — это патологическое/вырожденное дерево, в котором в дереве доминируют либо левые, либо правые узлы. Таким образом, существует два типа перекошенного двоичного дерева: двоичное дерево с перекосом влево и двоичное дерево с перекосом вправо .



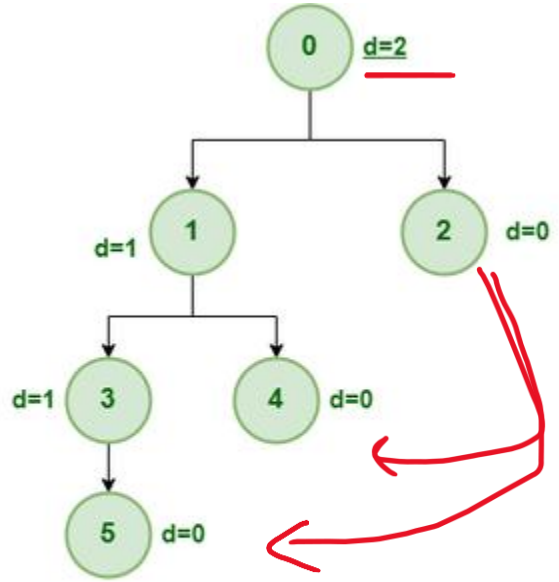
# 6. Сбалансированное двоичное дерево (**Balanced Binary Tree**)

- Это тип двоичного дерева, в котором разница между высотой левого и правого поддерева для каждого узла равна 0 или 1.




**Balanced Binary Tree with depth at each level indicated**

Depth of a node = |height of left child - height of right child|




**Unbalanced Binary Tree with depth at each level indicated**

Depth of a node = |height of left child - height of right child|

# Обход дерева

- Чтобы выполнить какую-либо операцию с деревом, вам необходимо добраться до определенного узла. Алгоритм обхода дерева помогает посетить необходимый узел дерева.

- Обход дерева означает посещение каждого узла в дереве. **Например, вы можете захотеть сложить все значения в дереве или найти наибольшее из них.** Для всех этих операций вам нужно будет посетить каждый узел дерева.

- Линейные структуры данных, такие как массивы, стеки , очереди и связанные списки, имеют только один способ чтения данных. Но иерархическую структуру данных, такую как дерево, можно просматривать по-разному.

# Виды обходов деревья

1. Inorder traversal
2. Preorder traversal
3. Postorder traversal

# Inorder traversal

1.Сначала посетите все узлы левого поддерева.

2.После, корневой узел

3.Посетите все узлы в правом поддереве

```
inorder(root->left)
display(root->data)
inorder(root->right)
```

# Preorder traversal

1.Посетите корневой узел

2.Посетите все узлы в левом поддереве

3.Посетите все узлы в правом поддереве

```
display(root->data)
preorder(root->left)
preorder(root->right)
```

# Postorder traversal

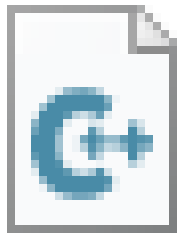
1.Посетите все узлы в левом поддереве

2.Посетите все узлы в правом поддереве

3.Посетите корневой узел

```
postorder(root->left)
postorder(root->right)
display(root->data)
```

# Пример кода



binary_tree.cpp

# Свойства двоичного дерева

**2. Максимальное количество узлов в бинарном дереве высоты h составляет $2^h - 1$:**

**Примечание.** Здесь высота дерева — это максимальное количество узлов на пути от корня к листу. Высота дерева с одним узлом считается равной 1.

*Этот результат можно получить из пункта 2 выше. Дерево имеет максимальное количество узлов, если все уровни имеют максимальное количество узлов. Таким образом, максимальное количество узлов в бинарном дереве высоты h равно $1 + 2 + 4 + .. + 2^{h-1}$ . Это простая геометрическая прогрессия с h членами, сумма которой равна $2^h - 1$.*

*В некоторых книгах высота корня считается равной 0. В этом соглашении приведенная выше формула принимает вид $2^{h+1} - 1$.*

**3. В двоичном дереве с N узлами минимально возможная высота или минимальное количество уровней равно $\text{Log}_2 (N+1)$:**

На каждом уровне должен быть хотя бы один элемент, поэтому высота не может быть больше N. Бинарное дерево высоты «h» может иметь максимум $2^h - 1$ узлов (предыдущее свойство). Таким образом, количество узлов будет меньше или равно этому максимальному значению.

$N <= 2^h - 1$
$2^h >= N+1$
$\log_2(2^h) >= \log_2(N+1)$ (Taking log both sides)
$h\log_2 2 >= \log_2(N+1)$ (h is an integer)
$h >= \lfloor \log_2(N+1) \rfloor$

So the minimum height possible is $\lfloor \log_2(N+1) \rfloor$

# Задание. Вставка в структуру данных двоичного дерева

- Для вставки узла в бинарное дерево вам необходимо проверить следующие условия:
  - Если у узла в двоичном дереве нет левого дочернего элемента, то вставьте данный узел (тот, который нам нужно вставить) в качестве его левого дочернего элемента.
  - Если у узла в двоичном дереве нет правого дочернего элемента, вставьте данный узел в качестве его правого дочернего элемента.
  - Если приведенные выше условия не применимы, найдите узел, у которого вообще нет дочернего элемента, и вставьте туда данный узел.
  - Мы можем использовать очередь для реализации этого алгоритма, так как таким образом легко хранить и извлекать узлы двоичного дерева, поскольку очередь подчиняется правилу FIFO. Кроме того, большинство древовидных алгоритмов реализованы с использованием очереди.

# Выводы

1. Дерево — нелинейная иерархическая структура из узлов и рёбер; в бинарном дереве у узла не более двух потомков.
2. Различают полные, идеальные, вырожденные, искажённые и сбалансированные деревья; форма дерева влияет на эффективность операций.
3. Обходы inorder, preorder и postorder посещают каждый узел один раз и требуют O(n) времени.
4. Для вставки узла в первое свободное место удобно использовать очередь.
5. Бинарные деревья применяют для хранения и поиска данных, вычисления выражений, маршрутизации и игрового ИИ.

# Контрольные вопросы

1. Что такое бинарное дерево? Что такое корень, лист и высота дерева?
2. Почему для некоторых задач используют деревья, а не линейные структуры данных?
3. Перечислите типы бинарных деревьев. Чем полное дерево отличается от идеального?
4. В каком порядке посещаются узлы при inorder, preorder и postorder обходах?
5. Выполните три вида обхода для заданного бинарного дерева.
6. Как выполнить вставку узла в бинарное дерево с помощью очереди?

# Список литературы

## Основная литература


1. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ. 3-е изд. — М.: Вильямс, 2013.
2. Скиена С. Алгоритмы. Руководство по разработке. 2-е изд. — СПб.: БХВ-Петербург, 2011.
3. Седжвик Р., Уэйн К. Алгоритмы на Java. 4-е изд. — М.: Вильямс, 2013.


## Дополнительная литература


4. Вирт Н. Алгоритмы и структуры данных. — М.: ДМК Пресс, 2010.
5. Aho A.V., Hopcroft J.E., Ullman J.D. Data Structures and Algorithms. — Addison-Wesley, 1983.
6. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms. 4th ed. — MIT Press, 2022.


## Электронные ресурсы


7. VisuAlgo — визуализация алгоритмов и структур данных: visualgo.net
8. Справочник по C++ и STL: en.cppreference.com