Деревья

Для вопросов по курсу:

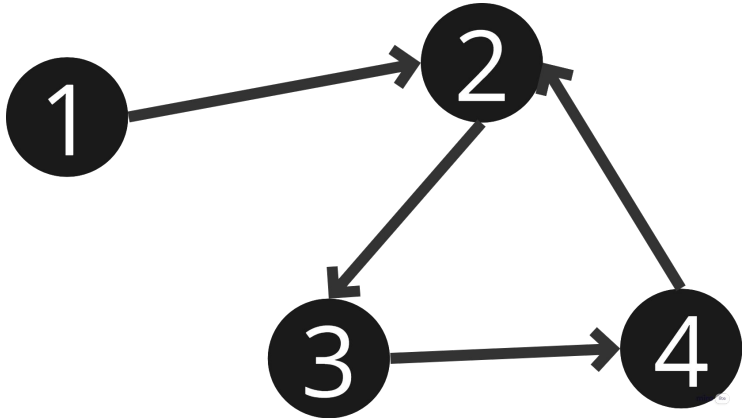
- Почта дисциплины - alg@moevm.pro

Префикс в теме письма [ALG_53XX]

Граф

G = (V, E) - граф, где V - набор вершин, E - набор ребер, которые соединяют эти вершины.

Один из способов представления графа - это матрица смежности



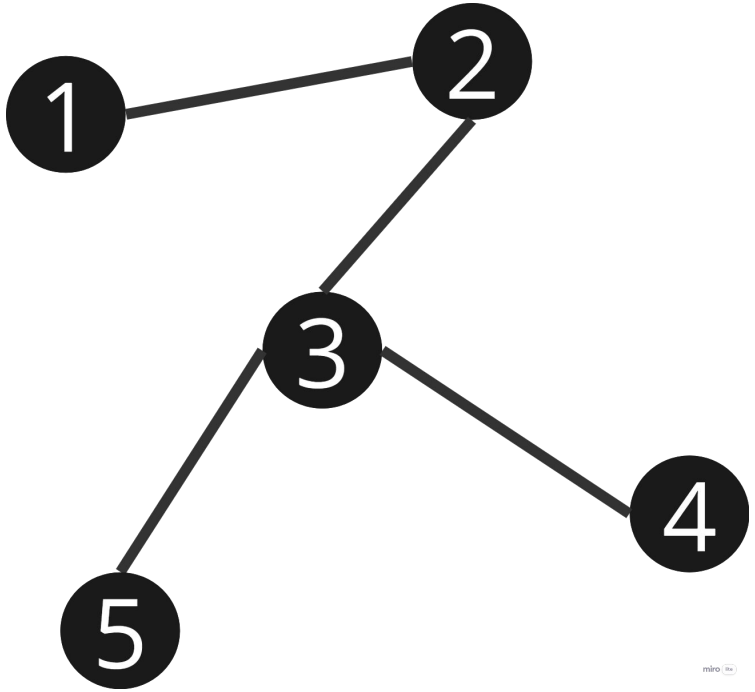
	1	2	3	4
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
4	0	1	0	0

Деревья

- Определение

  Дерево - это связный граф без циклов.

- Свойства
  - n вершин
  - n-1 ребер
  - 1 путь для 2-х любых вершин



Деревья

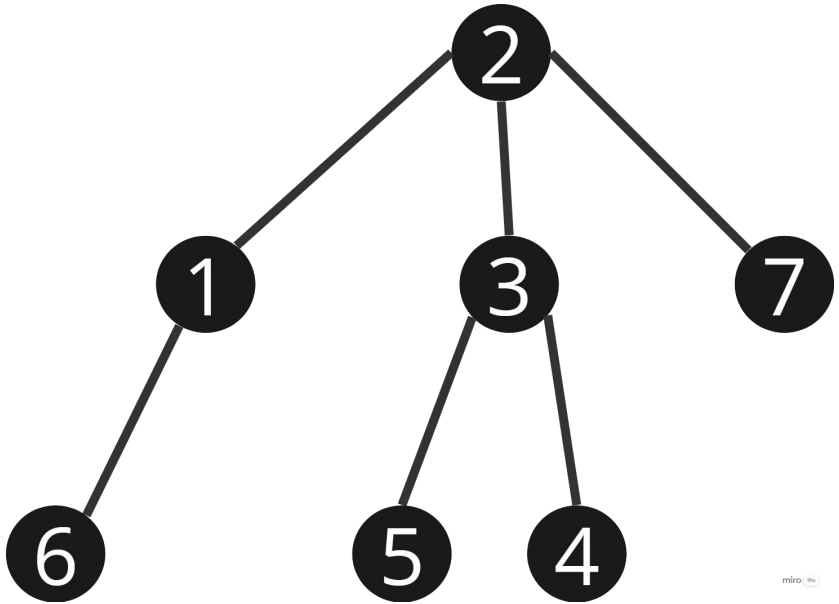
Корневое дерево - дерево с отмеченной вершиной - корнем.

2 - корень

1, 3, 7 - дети 2

3 - родитель 4, 5

6, 5, 4, 7 - листы



Деревья

➢ Высота дерева

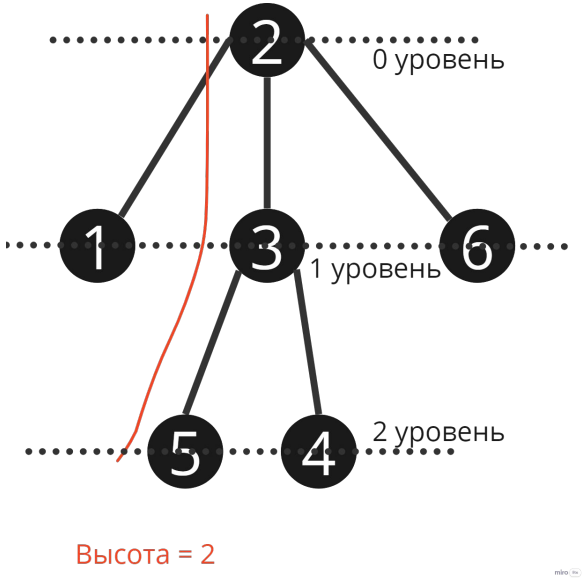
длина самого дальнего пути от корня к листу

➢ n-й уровень в дереве

множество узлов, находящихся в дереве на высоте n

➢ Рекуррентное определение

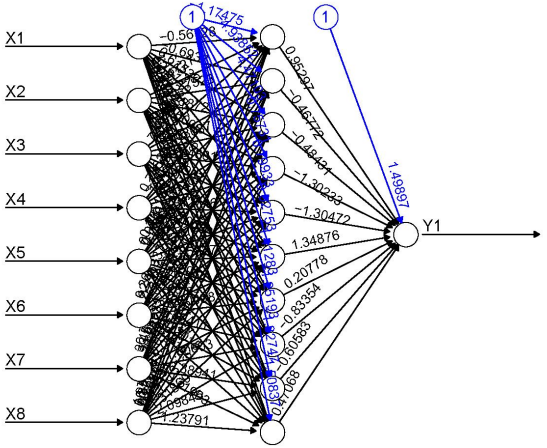
дерево - это корень и несколько деревьев, которые являются детьми этого корня.

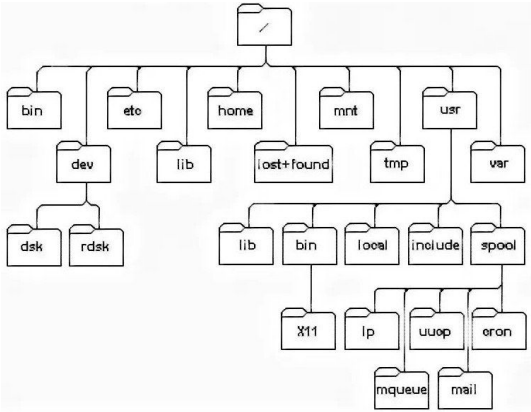


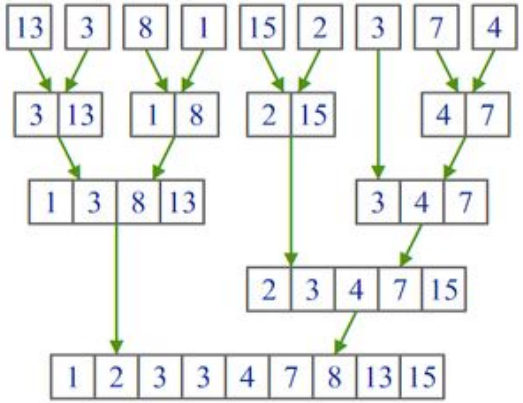
Примечание: дерево может быть пустым

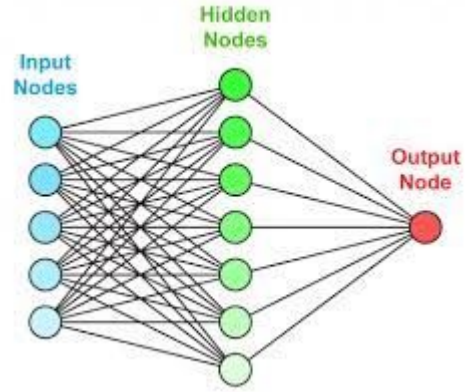
Использование деревьев

- Искусственный интеллект
- Операционные системы
- Сложные алгоритмы (в том числе некоторые сортировки) и способы хранения данных







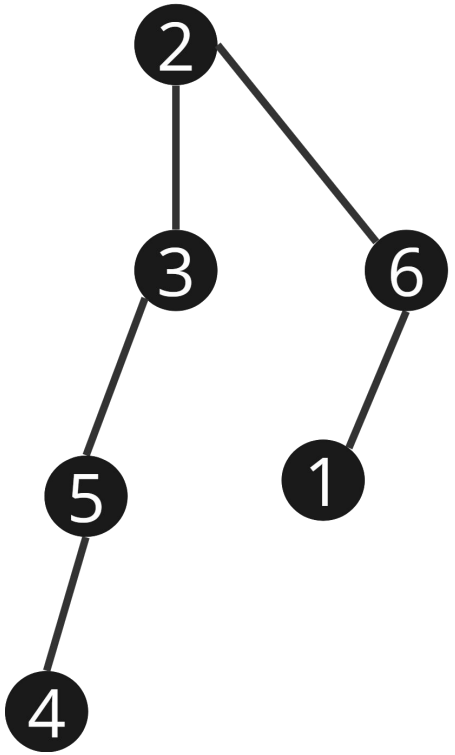


Способы представления деревьев

6	2	2	5	3	2

- Массив родителей

в ячейку с текущим номером записывается родителей соответствующей вершины, причем для корневой вершины записывается сам корень

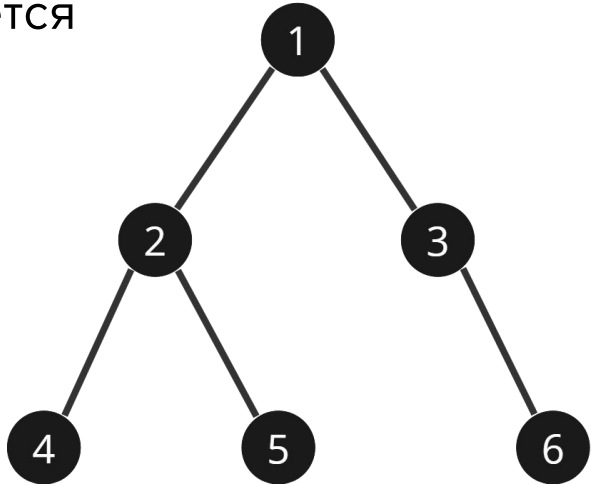


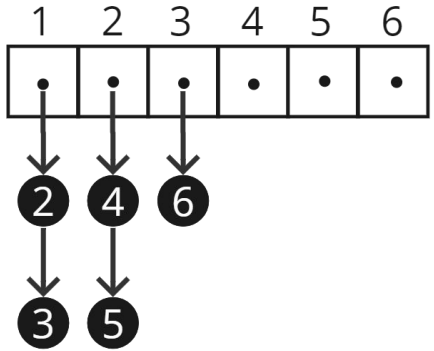
Способы представления деревьев

➢ Списки детей

в элемент массива записывается

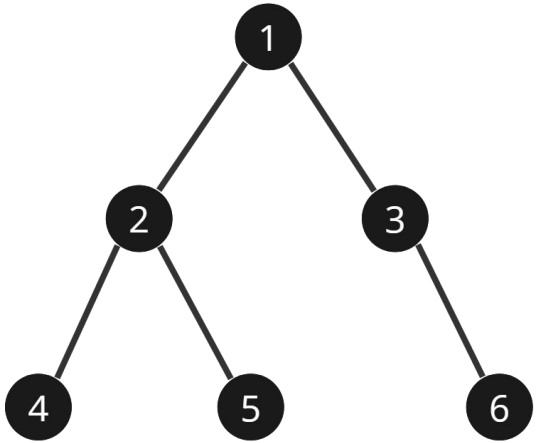
массив детей текущего узла

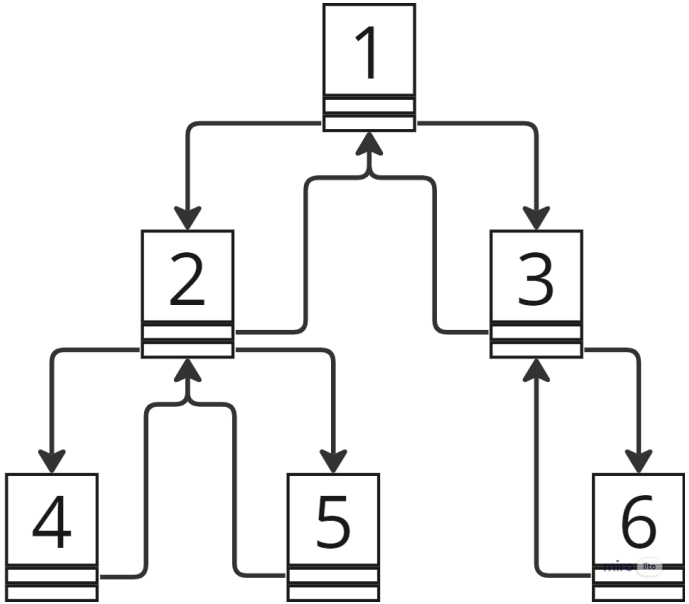




Способы представления деревьев

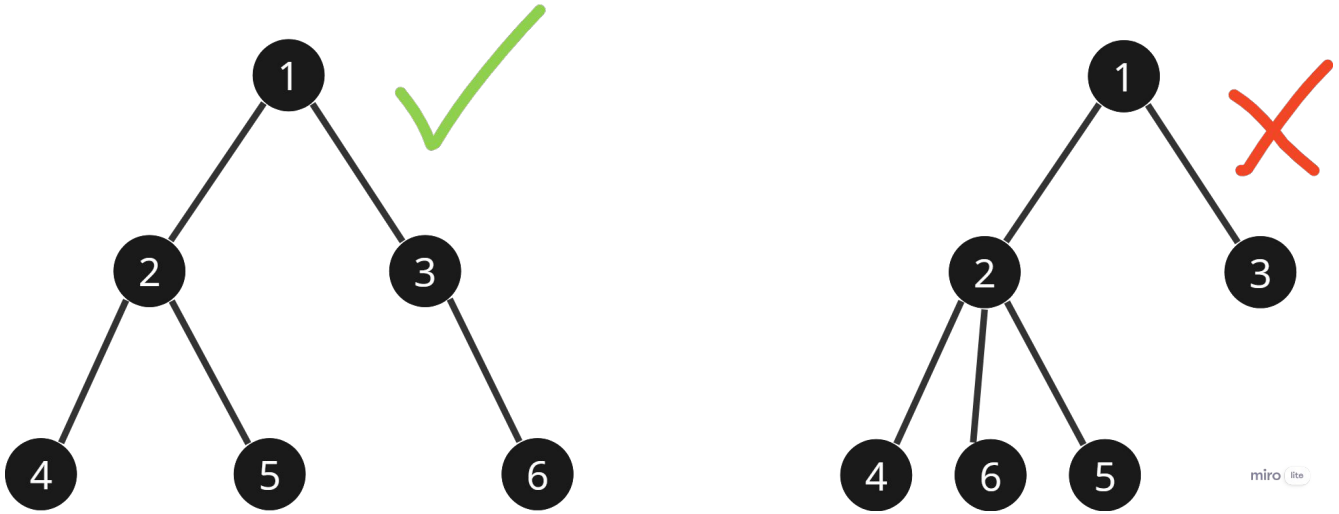
- Связный список с ссылками на детей и родителя





Бинарное дерево

- Бинарное дерево - дерево, каждый узел которого имеет максимум 2 потомка.



Обход дерева

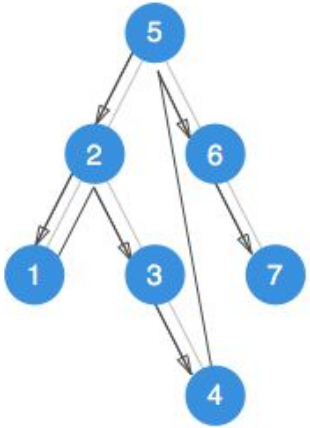
- Обходы в глубину (Depth-first traversal)

Центрированный обход (In-order traversal)

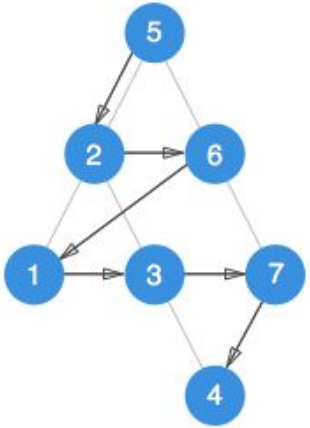
Прямой обход (Pre-order traversal)

Обратный обход (Post-order traversal)

- Поиск в ширину (Breadth-first search)



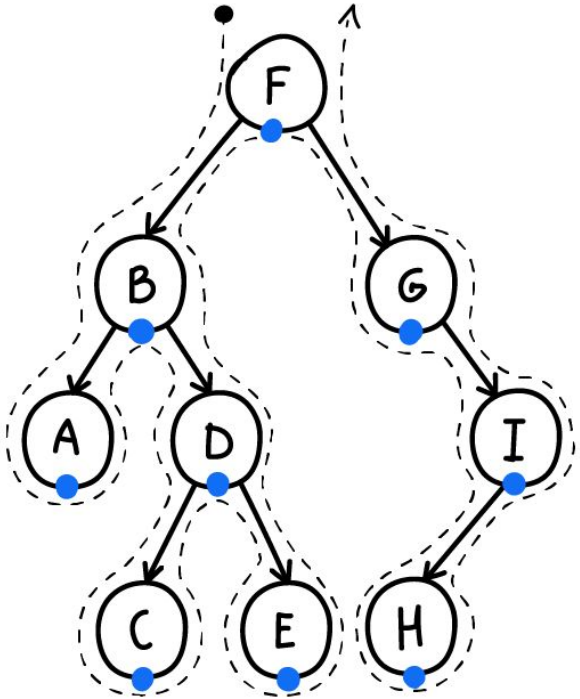
Depth-first traversal



Breadth-first traversal

Обход в глубину (Центрированный)

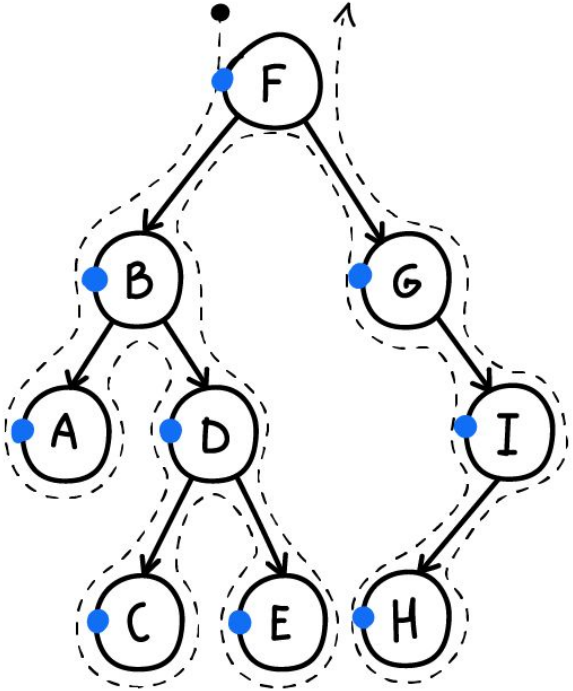
- Сначала посещается левое поддерево
- Затем корень
- В конце правое поддерево



Порядок посещения на примере: ABCDEFGHI

Обход в глубину (Прямой)

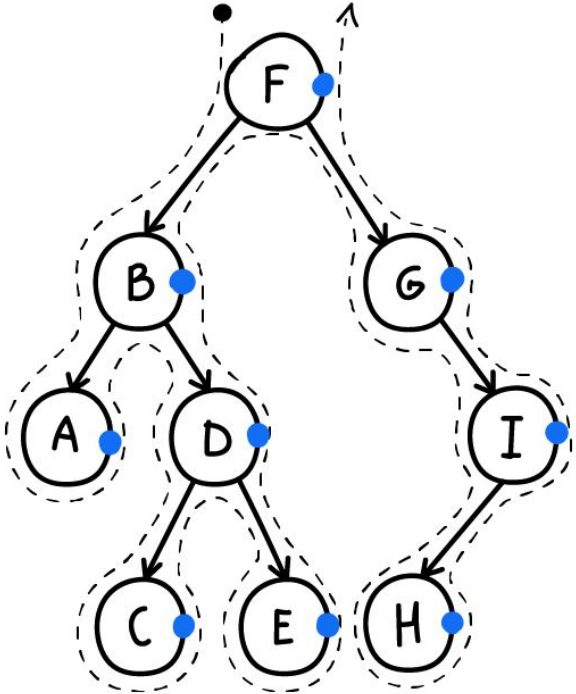
- Сначала посещается корень
- Затем левое поддерево
- В конце правое поддерево



Порядок посещения на примере: FBADCEGIH

Обход в глубину (Обратный)

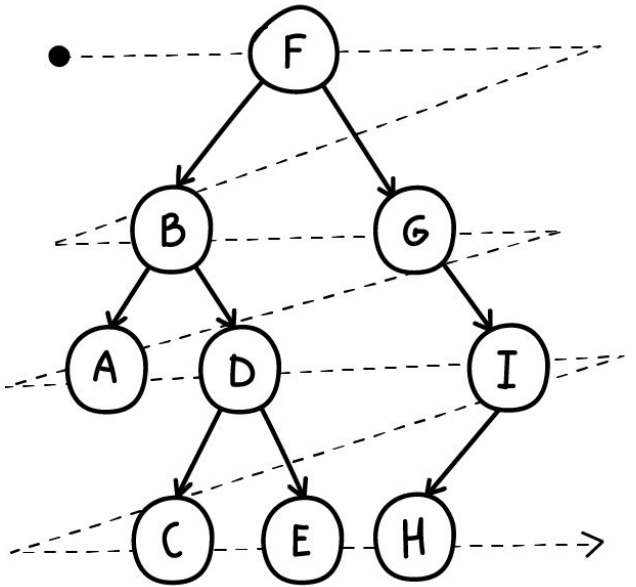
- Сначала посещается левое поддерево
- Затем правое поддерево
- В конце - корень



Порядок посещения на примере: ACEDBHIGF

Обход в ширину

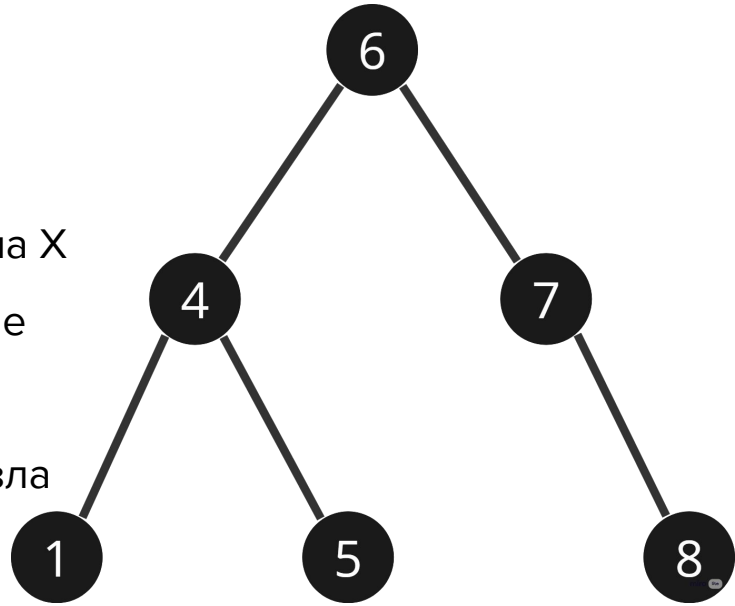
1. Используется очередь
2. В очередь добавляется корень
3. Корень берется из очереди и обрабатывается
4. В очередь добавляются все дети корня
5. Пункты 3 и 4 повторяются, пока очередь не станет пустой



Порядок посещения на примере: FBGADICEH

Бинарное дерево поиска

- Бинарное дерево, для которого выполняются следующие условия:
  - оба поддерева — левое и правое — являются бинарными деревьями поиска;
  - у всех узлов левого поддерева произвольного узла X значения ключей данных меньше, нежели значение ключа данных самого узла X;
  - у всех узлов правого поддерева произвольного узла X значения ключей данных больше либо равны, нежели значение ключа данных самого узла X.



Основные операции

➢ Поиск

Рекурсивный обход по дереву, сравнивая значение элемента с текущим, чтобы определить, в какую сторону дальше продвинуться.

```
1  Node search(x : Node, k : T):
2      if x == null or k == x.key
3          return x
4      if k < x.key
5          return search(x.left, k)
6      else
7          return search(x.right, k)
```

Основные операции

- Вставка

Сначала производится поиск места, в которое данный узел будет корректно поместить, а затем уже создается новый узел.

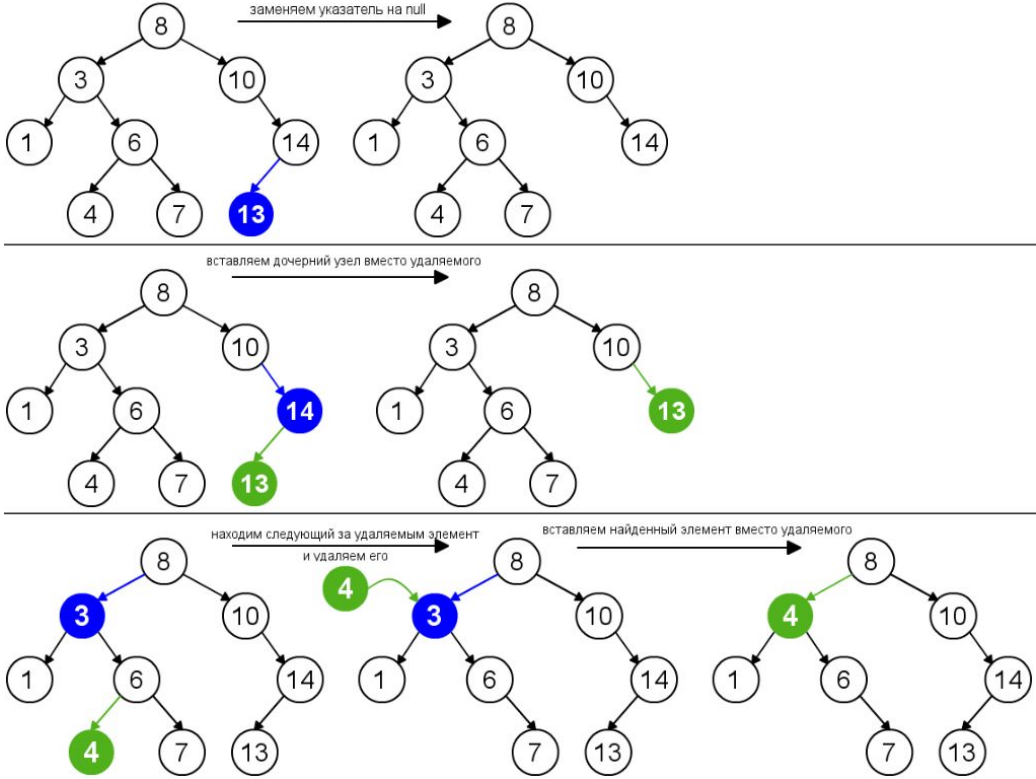
```
1  Node insert(x : Node, z : T): // x – корень поддерева, z – вставляемый ключ
2      if x == null
3          return Node(z)  // Создание нового узла
4      else if z < x.key
5          x.left = insert(x.left, z)
6      else
7          x.right = insert(x.right, z)
```

Основные операции

➢ Удаление

При удалении стоит обратить внимание на 3 допустимые ситуации:

- удаление листа
- удаление узла с 1 ребенком
- удаление узла с 2 детьми



Основные операции

- Удаление

```
Node delete(root : Node, z : T): // корень поддерева, удаляемый ключ
  if root == null
    return root
  if z < root.key
    root.left = delete(root.left, z)
  else if z > root.key
    root.right = delete(root.right, z)
  else if root.left != null and root.right != null
    root.key = minimum(root.right).key
    root.right = delete(root.right, root.key)
  else
    if root.left != null
      root = root.left
    else if root.right != null
      root = root.right
    else
      root = null
  return root
```

Подумайте: Какая сложность будет у операций поиска минимального и максимального элементов в лучшем, среднем и худшем случаях?

Вопросы по курсу можно задавать:

Почта дисциплины - alg@moevm.pro
Иванов Дмитрий Владимирович - dmitry.ivanov@moevm.pro