

Деревья

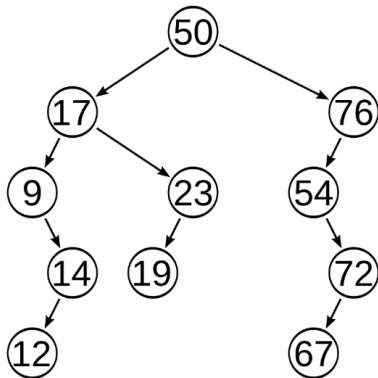
Юрий Литвинов
y.litvinov@spbu.ru

25.10.2023

Дерево

Ещё один абстрактный тип данных, используемый в программировании повсеместно

- ▶ Файловая система
- ▶ Абстрактное синтаксическое дерево
 - ▶ Дерево разбора арифметического выражения
- ▶ Двоичное дерево поиска
 - ▶ Основа для реализации множеств
- ▶ Дерево контролов (или виджетов) в пользовательском интерфейсе
- ▶ ...



Определения

- ▶ Дерево — совокупность элементов, называемых узлами (один из которых — корень), и отношений, образующих иерархическую структуру узлов
 - ▶ Узел является деревом, он же — корень дерева
 - ▶ Есть узел n и деревья T_1, T_2, \dots, T_k — деревья с корнями n_1, n_2, \dots, n_k соответственно. Тогда можно построить новое дерево, с корнем n и поддеревьями T_1, T_2, \dots, T_k . Узлы n_1, n_2, \dots, n_k называются сыновьями узла n
- ▶ Нулевое дерево — дерево без узлов
- ▶ Дерево — связный ациклический граф
- ▶ Несвязный ациклический граф — лес

Ещё определения

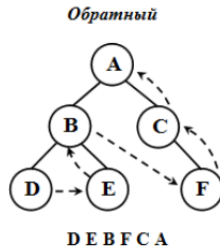
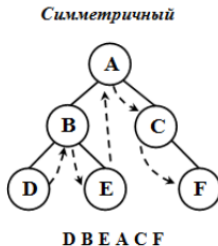
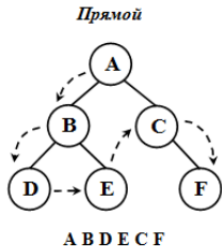
- ▶ Путь из n_1 в n_k — последовательность узлов n_1, \dots, n_k , в которой каждый узел является родителем следующего
- ▶ Длина пути — число, на единицу меньшее количества узлов, составляющих путь
- ▶ Путь нулевой длины — путь из узла к самому себе
- ▶ Узел a называется предком узла b , если существует путь из a в b
 - ▶ b в этом случае — потомок a
 - ▶ Каждый узел — предок и потомок самого себя
- ▶ Потомок, не являющийся самим узлом, называется истинным потомком, с предком аналогично
- ▶ Узел, не имеющий истинных потомков, называется листом
- ▶ Поддерево какого-либо дерева — узел вместе со всеми потомками

И ещё определения

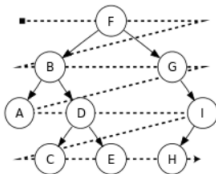
- ▶ Высота узла — длина самого длинного пути из узла до какого-либо листа
- ▶ Глубина узла — длина пути от узла до корня
- ▶ Высота дерева — высота корня
- ▶ Деревья бывают упорядоченными и неупорядоченными
 - ▶ Можно упорядочить узлы дерева, не связанные отношением предок-потомок (слева-справа)
- ▶ Деревья бывают помеченными (каждой вершине сопоставлено значение)

Обходы

► В глубину



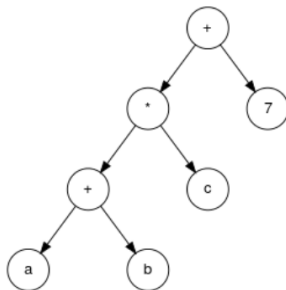
► В ширину



Деревья выражений

$$(a + b) * c + 7$$

- ▶ Прямой порядок — префиксная запись
 - ▶ $+ * + a b c 7$
- ▶ Обратный порядок — постфиксная запись
 - ▶ $a b + c * 7 +$
- ▶ Симметричный порядок — инфиксная запись
 - ▶ $a + b * c + 7$



АТД “Дерево”

- ▶ `parent(n, t)`
- ▶ `leftmostChild(n, t)`
- ▶ `rightSibling(n, t)`
- ▶ `label(n, t)`
- ▶ `create(n, t1, ..., ti)`
- ▶ `root(t)`
- ▶ `makenull(t)`

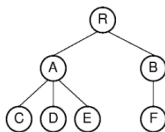
```
void preorder(Node *n)
{
    printf("%s\n", label(n));
    Node *child = leftmostChild(n);
    while (child != NULL)
    {
        preorder(child);
        child = rightSibling(child);
    }
}
```


Нерекурсивный обход в прямом порядке

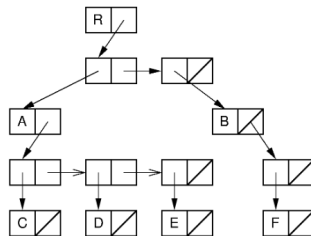
```
void nonRecursivePreorder(Node *root) {  
    Stack* stack = createStack();  
    Node *current = root;  
    while (true) {  
        if (current != NULL) {  
            printf("%s\n", label(current));  
            push(stack, current);  
            current = leftmostChild(current);  
        } else {  
            if (isEmpty(stack))  
                return;  
            current = rightSibling(top(stack));  
            pop(stack);  
        }  
    }  
    deleteStack(stack);  
}
```

Реализация списком сыновей

```
struct Node
{
    ElementType value;
    struct Node *sibling;
    struct Node *child;
};
```



(a)



(b)

Двоичные деревья

Деревья, у которых есть левый и правый сын, и это разные вещи

```
struct Node
```

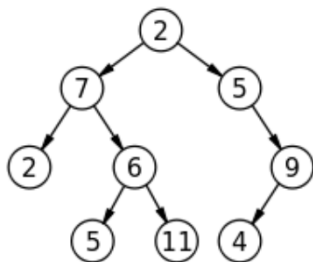
```
{
```

```
    ElementType value;
```

```
    struct Node *leftChild;
```

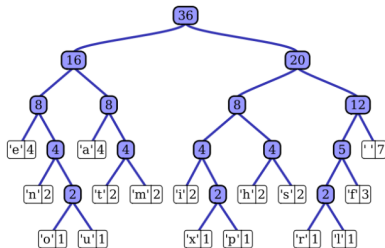
```
    struct Node *rightChild;
```

```
};
```



Пример: алгоритм Хаффмана

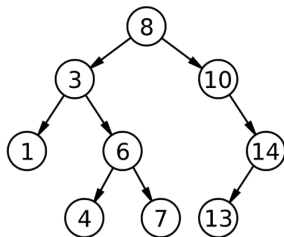
- ▶ Алгоритм сжатия, вычисляющий кратчайшую кодовую последовательность для символа
 - ▶ Если в тексте одни буквы “А”, нет смысла кодировать А 16-ю битами
- ▶ Префиксные коды
- ▶ Дерево частот символов



Пример: “this is an example of a huffman tree”

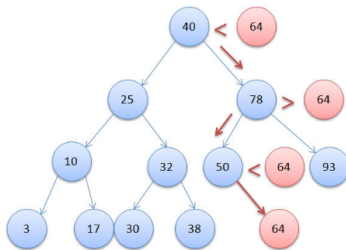
Двоичное дерево поиска

- ▶ Двоичное дерево, у которого для каждого узла в левом поддереве элементы, меньшие значения в узле, в правом — элементы, большие значения в узле
- ▶ Используется для представления множеств и ассоциативных массивов
 - ▶ Если дерево сбалансировано (т.е. высота примерно логарифм количества вершин), операции вставки, удаления и поиска выполняются за $\log(n)$

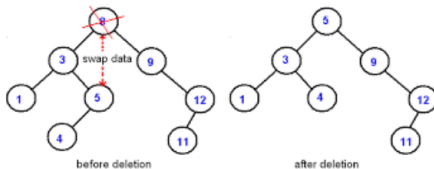


Операции

► Вставка



► Удаление



Проблема

- ▶ При неудачном порядке вставки дерево может вырождаться в список
- ▶ Трудоёмкости всех операций сразу станут линейными

