

Алгоритмы и структуры данных на Python. Интерактивный  
курс

## Урок 9



# Деревья. Хеш-функция

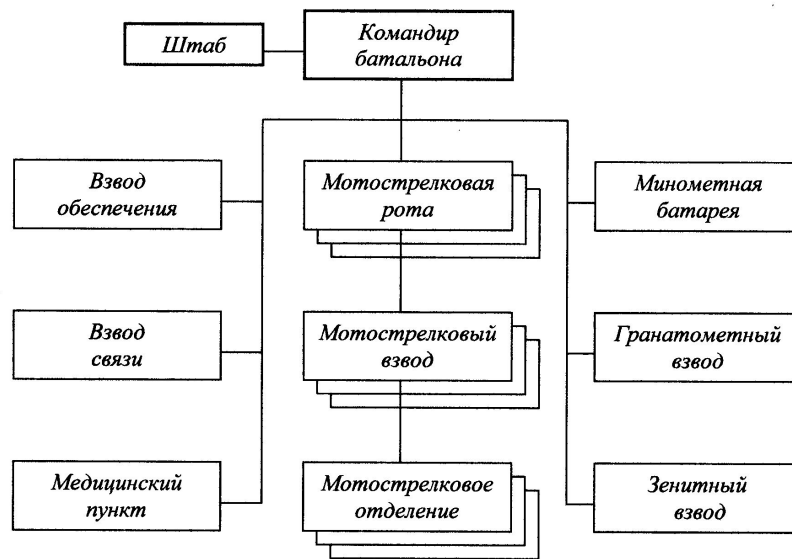
Двоичные деревья поиска.  
Проход по дереву. Хеш-функция.

# План

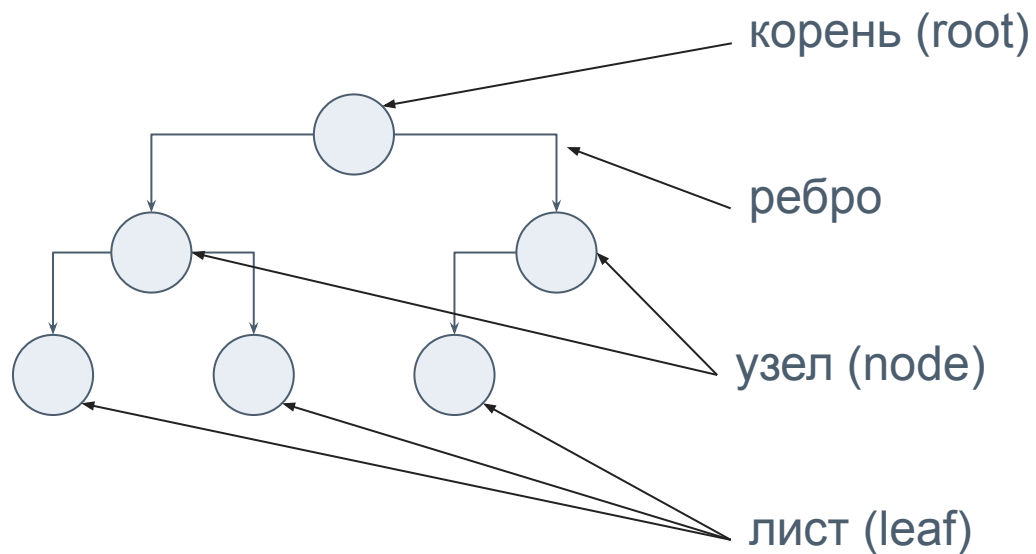
- Что такое дерево?
- Классификация деревьев
- Создание деревьев в Python



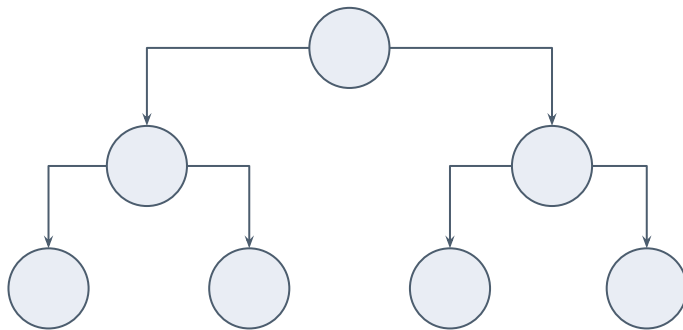
# Иерархия позволяет управлять



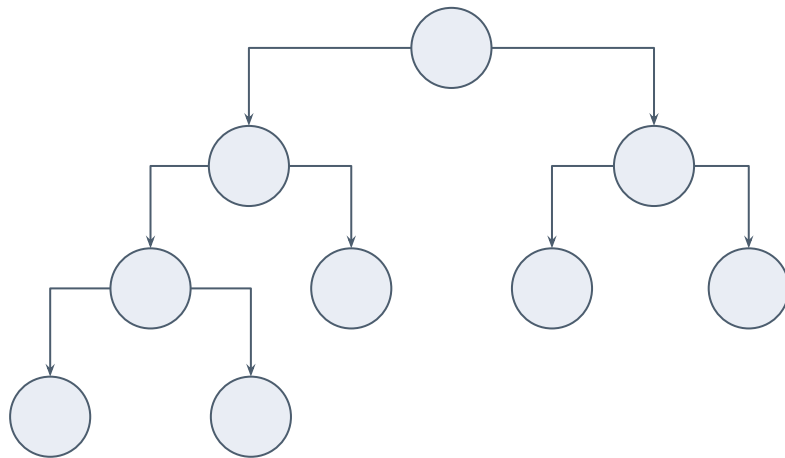
# Бинарное дерево



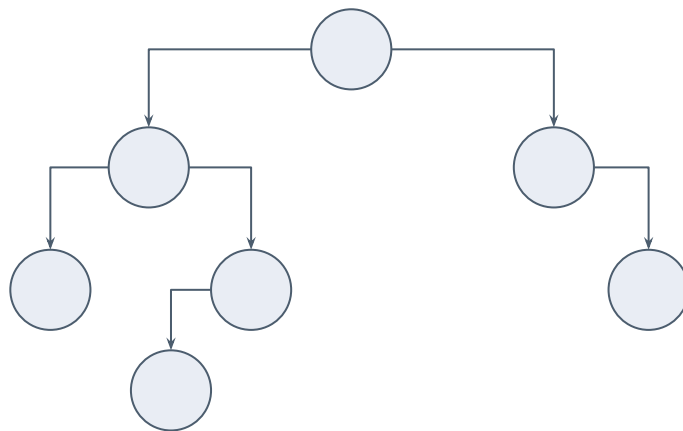
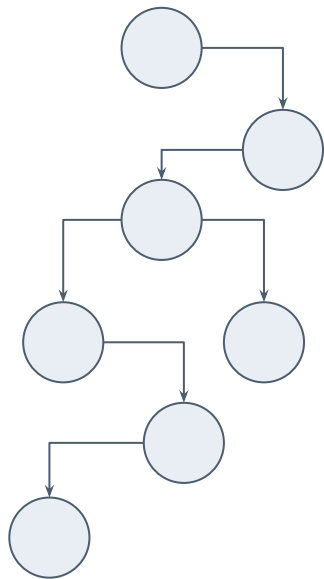
# Разновидности бинарных деревьев: полное (расширенное), идеальное



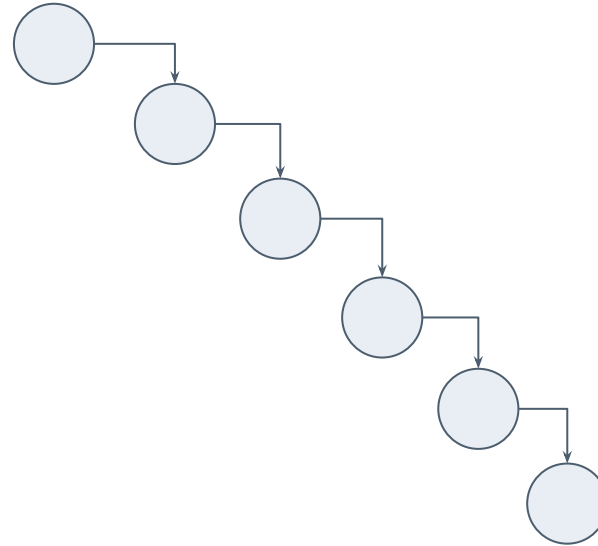
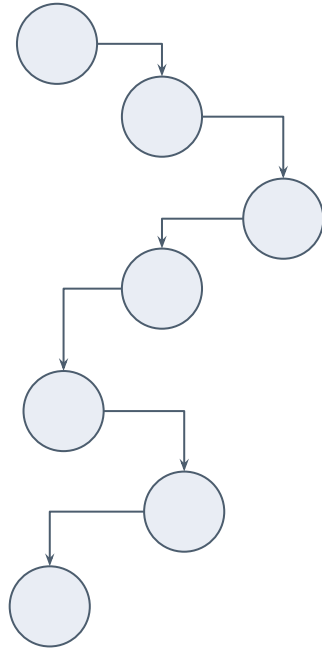
# Разновидности бинарных деревьев: сбалансированное



# Разновидности бинарных деревьев: неполное

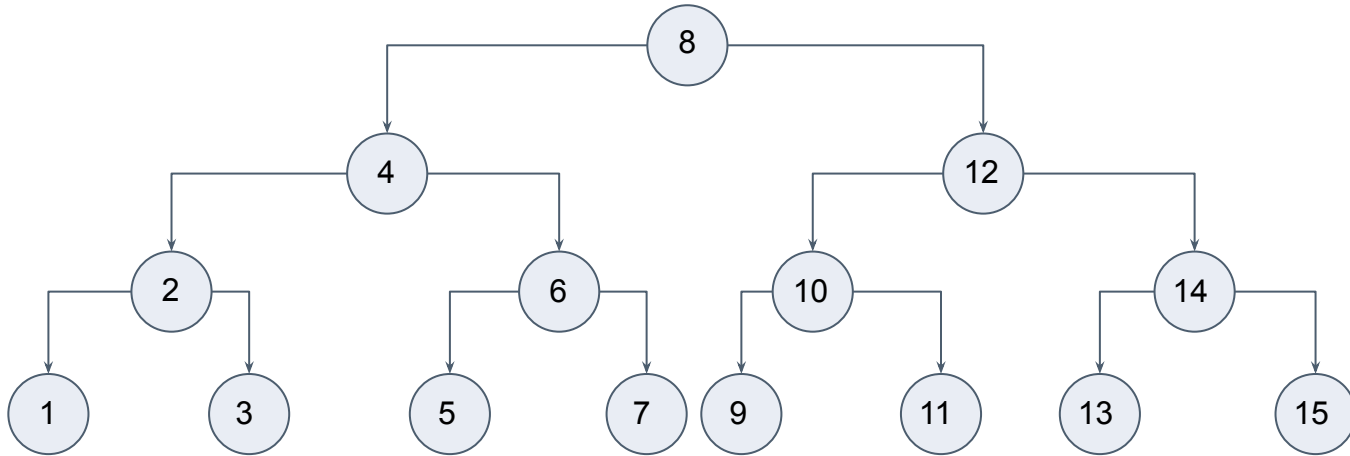


# Разновидности бинарных деревьев: вырожденное





# Разновидности бинарных деревьев: бинарное поисковое дерево (BST)



# Итоги:

## Теория

- Что такое дерево
- Классификация деревьев

## Практика

- Создание деревьев в Python

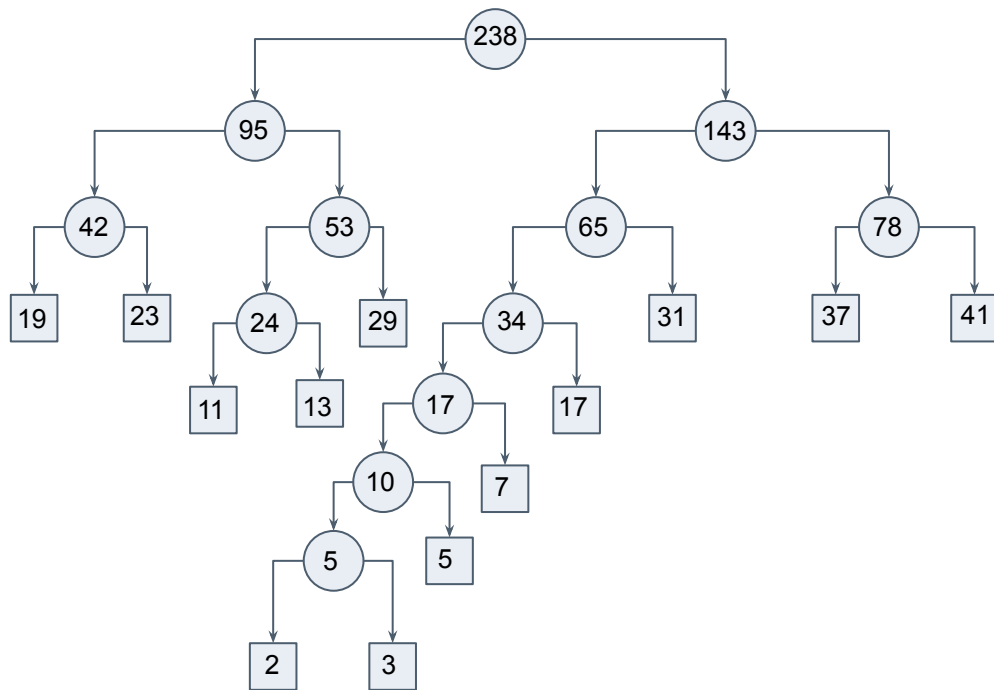


# План

- Бинарный поиск
- Алгоритм Хаффмана



# Алгоритм Хаффмана



```
2 3 5 7 11 13 17 19 23 29 31 37 41
5 5 7 11 13 17 19 23 29 31 37 41
10 7 11 13 17 19 23 29 31 37 41
17 24 17 19 23 29 31 37 41
24 34 19 23 29 31 37 41
24 34 42 29 31 37 41
42 53 65 37 41
42 53 65 78
95 65 78
95 143
238
```



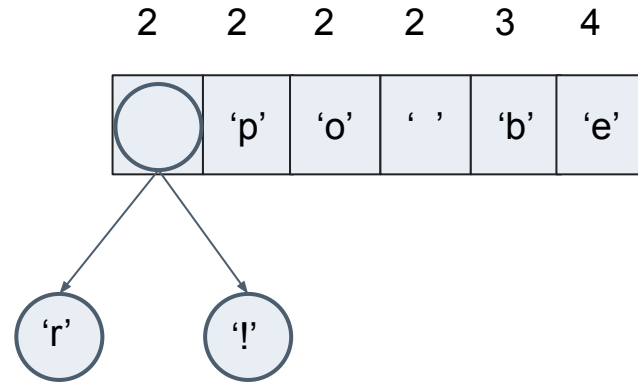
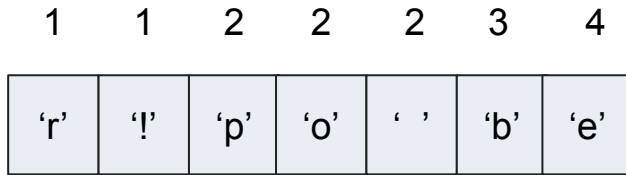
# Кодирование по методу Хаффмана

Символ	Частота
'b'	3
'e'	4
'p'	2
' '	2
'o'	2
'r'	1
'!'	1

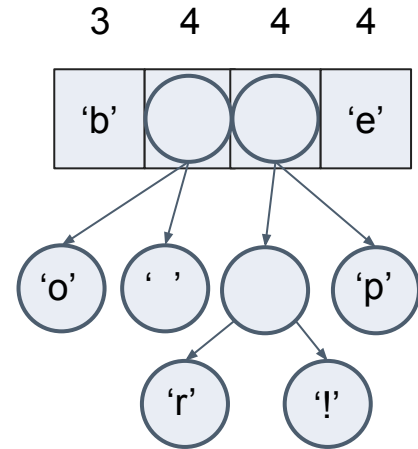
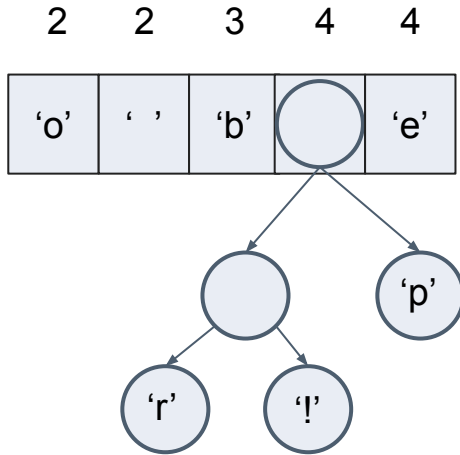
Закодируем строку «beer boor beer!»



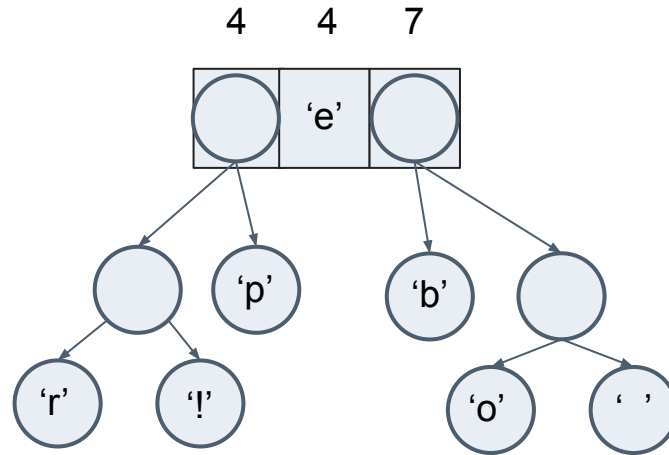
# Кодирование по методу Хаффмана



# Кодирование по методу Хаффмана

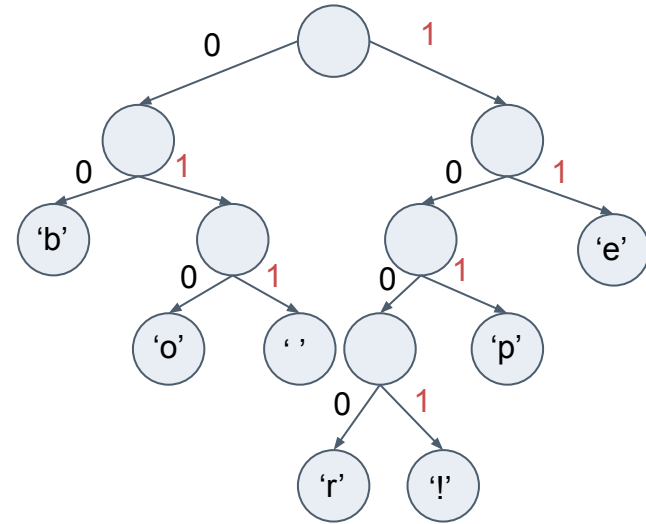
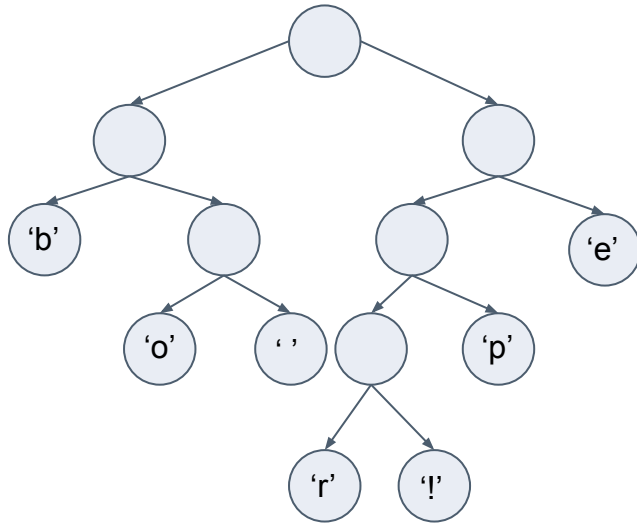


# Кодирование по методу Хаффмана





# Кодирование по методу Хаффмана



# Кодирование по методу Хаффмана

Символ	Код
'b'	00
'e'	11
'p'	101
' '	011
'o'	010
'r'	1000
'l'	1001



# Кодирование по методу Хаффмана

**Входная строка:** «beer boor beer!»

**Входная строка в бинарном виде:** «01100010 01100101 01100101  
01110000 00100000 01100010 01101111 01101111 01110000 00100000  
01100010 01100101 01100101 01110010 00100000»

**Закодированная строка:** «00 11 11 101 011 00 010 010 101 011 00 11  
11 1000 1001»



# Итоги:

## Теория

- Алгоритм Хаффмана

## Практика

- Бинарный поиск



# План

- Что такое хэш-функция
- Хэш-подпись сообщения



# Хеширование

**Хеш-функции** предназначены для «сжатия» произвольного сообщения или набора данных, записанных в двоичном алфавите, в битовую комбинацию фиксированной длины – свертку.



# Сферы применения хэш-функций

- Словарь (`dict`)
- Множество (`set`)
- Построение систем контроля целостности данных при передаче или хранении.
- Аутентификация источника данных.



# Итоги:

## Теория и Практика

- Хэш-функция
- Хэш-подпись сообщения





# План

- Алгоритм Secure Hash Algorithm 1 (SHA-1)



# Алгоритм SHA-1

Алгоритм получает на входе сообщение максимальной длины  $2^{64} - 1$  бит и создает в качестве выхода хэш-строку длиной 160 бит.



# Алгоритм SHA-1

Псевдокод + Python Style

==

Понятный пример + Новые знания



# Итоги:

## Теория и Практика

- Алгоритм SHA-1 в виде Python Style псевдокода



# План

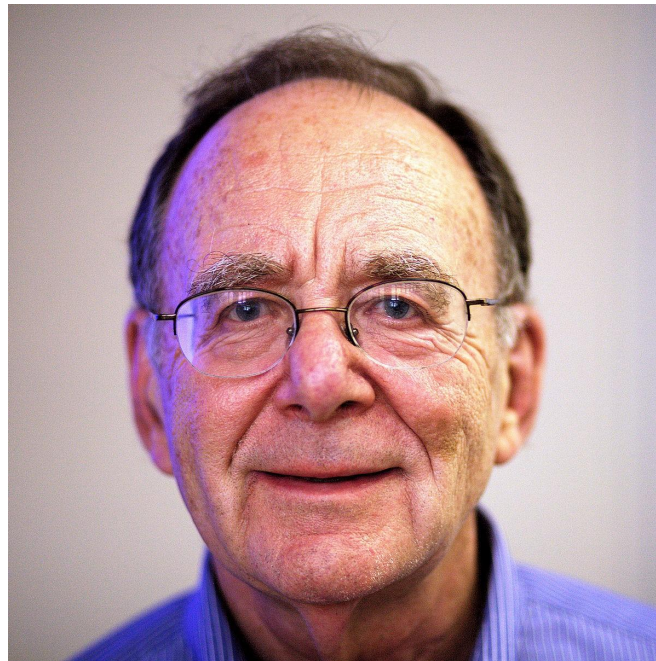
- Задача 1. Сравнение строк с помощью хеширования
- Задача 2. Поиск подстроки в строке.



# Поиск подстроки в строке



Михаэль Ошер Рабин



Ричард Мэннинг Карп



# Итоги:

## Практика

- Сравнение строк с помощью хеширования.
- Поиск подстроки в строке.



# Домашнее задание

1. Определить количество различных подстрок с использованием хеш-функции. Дана строка  $S$  длиной  $N$ , состоящая только из строчных латинских букв.  
Требуется найти количество подстрок в этой строке.





# Домашнее задание

2. Закодировать любую строку из трех слов по алгоритму Хаффмана.



# План

- Разбор домашнего задания

