

ЛЕКЦИЯ 24

Кодирование

1. Равномерное и неравномерное кодирование

Есть два типа кодирования: равномерное кодирование и неравномерное кодирование.

Обозначим алфавит допустимых символов \mathbb{A} .

В неравномерном кодировании код символов разной длины (например, кодировка UTF-8).

Закодируем буквы «А», «Б», «В», «Г» таким образом:

$$\begin{aligned}A &= 0 \\ B &= 1 \\ V &= 10 \\ \Gamma &= 111\end{aligned}$$

Тогда запись «ГАГА» можно закодировать так:

$$\text{ГАГА} = 11101110 = \text{БББГА},$$

т.е. декодирование неоднозначно, такое кодирование плохое.

Условие Фано: ни один код не является началом другого. - достаточное условие. Тогда $A=0$
 $B=110$ $V=10$ $\Gamma=111$

Возьмем

$$\begin{aligned}A &= 1 \quad 1 \\ B &= 10 \quad 01 \\ V &= 100 \quad 001 \\ \Gamma &= 000 \quad 000\end{aligned}$$

БАГАВА=10100011001 - однозначность есть, хотя и декодировать очень сложно. Обратное условие Фано: ни один код не является концом (суффиксом) другого.

Кодировка в равномерном кодировании UTF-16. Можно закодировать 2^n различных символов (мощность алфавита).

2. Поиск подстроки в строке

2.1. Наивный поиск подстроки в строке

Программа №2.1. Примитивный поиск подстроки в строке

```
1 s = "abbbbabbbaabbababababb"
2 subs = "bbbaba"
3 def find(s, sub):
4     for pos in range(0, len(s)-len(sub)+1):
5         for i in range(len(sub)):
6             if sub[i] != s[pos+i]:
7                 break
8         else:
9             return pos
10    return -1
```

Сложность алгоритма $O(N \cdot M)$. В итоге алгоритм получается неэффективным. Смотрим на каждый символ только по одному разу! Методика хранения автомата: орграф. Если конечный автомат уже построен, то время поиска $O(N)$, N — длина строки.

Программа №2.2.

```
1 state = 0
2 for c in s:
3     if state == 0:
4         if c == "1":
5             state = 1:
6     elif state == 1:
7         if c == "1":
8             state = 0
```

Программа №2.3. 1

```
1 state = 0
2 for c in s:
3     if state == 0:
4         if c == "a":
5             state = 1
6     elif state == 1:
7         if c == 'b':
8             state = 2
9         elif c == 'a':
10            state = 1
11        else:
12            state = 0
13    elif state == 2:
14        if c == 'c':
15            state = 3
16        elif c == 'b':
17            state = 2
18        elif c == 'a':
19            state = 1
20        else:
21            state = 0
22    elif state == 3:
23        if c == 'c':
24            state = 3
25        elif c == 'b':
26            state = 2
27        elif c == 'a':
28            state = 1
29        elif c == 'd':
30            state = 4
31        else:
32            state = 0
```

3. Расстояние Левенштейна

Введем расстройство лев. между подстроками а и b. $a[:i] \ b[:j] \ F_{ij} = L(a[:i], b[:j])$

$$F_{ij} = \begin{cases} \text{Последние буквы совпадают, то } F_{(i-1)(j-1)} & F_{oj} = j \\ 1 + \min(F_{(i-1)(j-1)}, F_{(i-1)j}, F_{i(j-1)}) & \end{cases}$$

$$F_{io} = i$$