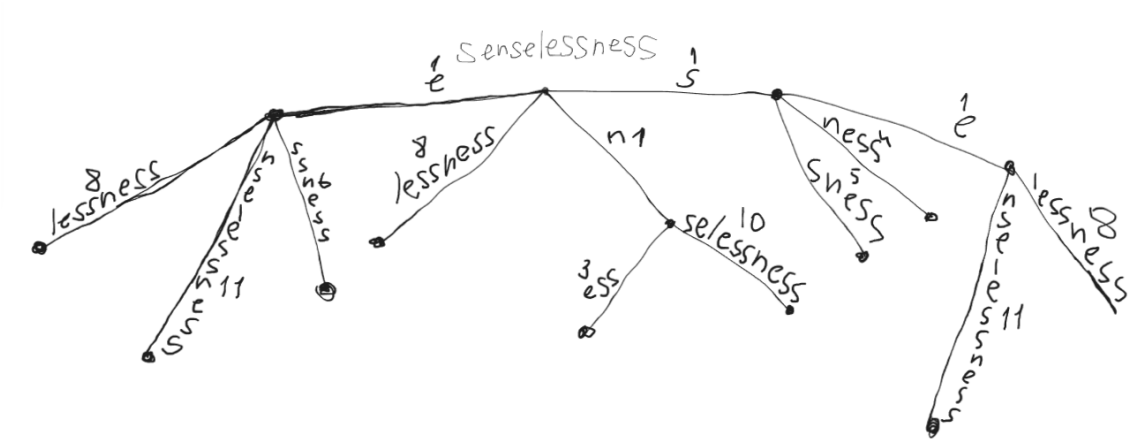
22. Постройте сжатое суффиксное дерево для строки "senselessness" и найдите количество различных подстрок в этой строке. Объясните способ подсчета с использованием суффиксного дерева.



Количество подстрок: 78

Как считаем: идём по каждому ребру и к результату прибавляем длину строки в этом ребре.

9. Сравните время работы set и unordered_set из STL для операций добавления N элементов, где N=100,10000,10^6,10^7. Ключами являются строки из случайных букв от a до z длиной ровно 16. Результат оформить в виде таблицы, время в ns. Привести код, использованный для измерения времени для одного значения N.

n	100	1e4	1e6	1e7
Set,ns	56800	5477400	908249200	14450391900
unordered_set,ns	392400	3924000	505049400	6350879200

```
количество: 100
set: 56800 наносекунд
unordered_set: 49800 наносекунд
>Код завершения: 0 Время выполнения: 90 ms
Для продолжения нажмите любую клавишу . . .
```

```
количество: 10000
set: 5477400 наносекунд
unordered_set: 3924000 наносекунд
>Код завершения: 0 Время выполнения: 90 ms
Для продолжения нажмите любую клавишу . . .
```

```
>Запуск программы...
количество: 1000000
set: 908249200 наносекунд
unordered_set: 505049400 наносекунд
>Код завершения: 0 Время выполнения: 1840 ms
Для продолжения нажмите любую клавишу . . .
```

```
>Запуск программы...
количество: 10000000
set: 14450391900 наносекунд
unordered_set: 6350879200 наносекунд
>Код завершения: 0 Время выполнения: 25550 ms
Для продолжения нажмите любую клавишу . . .
```

```
#include <iostream>
#include <set>
#include <unordered_set>
#include <chrono>
#include <random>
using namespace std;
using namespace chrono;
```

```
string generateStr() {
    string str;
    for (int i = 0; i < 16; ++i) {
        str.push_back('a' + rand() % 26);
    }
    return str;
}
long long setT(int N) {
    set<string> s;
    auto start = high_resolution_clock::now();
    for (int i = 0; i < N; ++i) {
        s.insert(generateStr());
    }
    auto stop = high_resolution_clock::now();
    return duration_cast<nanoseconds>(stop -
start).count();
}
long long uSetT(int N) {
    unordered_set<string> us;
    auto start = high_resolution_clock::now();
    for (int i = 0; i < N; ++i) {
        us.insert(generateStr());
    }
    auto stop = high_resolution_clock::now();
    return duration_cast<nanoseconds>(stop -
start).count();
}

int main()
{
```

```
    set<string> st;
    unordered_set<string> u_st;

    vector<string> tests;
    int n = (int)1e7;

    cout <<  "количество: " << n << endl;
    cout << "unordered_set: " <<  uSetT(n) << " наносекунд"<< endl;
   cout << "set: " << setT(n) << " наносекунд" << endl;

    st.clear();
    u_st.clear();
    return 0;
}
```

24. Напишите функцию для проверки принадлежности точки невыпуклому многоугольнику. В качестве параметров функции передаются координаты точки и вектор координат вершин многоугольника против часовой стрелки. Для точки использовать класс из лекций и его методы.

```
#include <iostream>
#include <vector>
#include <cmath>
using namespace std;
struct Point
{
double x, y;
double len() const // длина вектора
{
return hypot(x, y);
}
Point operator-(Point p) const //опреатор вычитания
{
return {x - p.x, y - p.y};
}
double operator^(Point p) const //оператор векьрного произведения
{
return x * p.y - y * p.x;
}
};
bool inside(Point point, const vector<Point>& top)
{
int n = top.size();
bool inside = false;
for (int i = 0, j = n - 1; i < n; j = i++)
{
if (((top[i].y > point.y) != (top[j].y > point.y)) &&
(point.x < (top[j].x - top[i].x) * (point.y - top[i].y) / (top[j].y - top[i].y) +
top[i].x)) {
inside = !inside;
}
}
return inside;
}
int main()
{
vector<Point> top = {{0, 0}, {4, 0}, {4, 4}, {2, 6}, {0, 4}};
Point point = {2, 3};
```

```
if (inside(point, top))
{
cout << "Точка находится внутри многоугольника" << endl;
}
else
{
cout << "Точка находится снаружи многоугольника" << endl;
}
return 0;
}
```