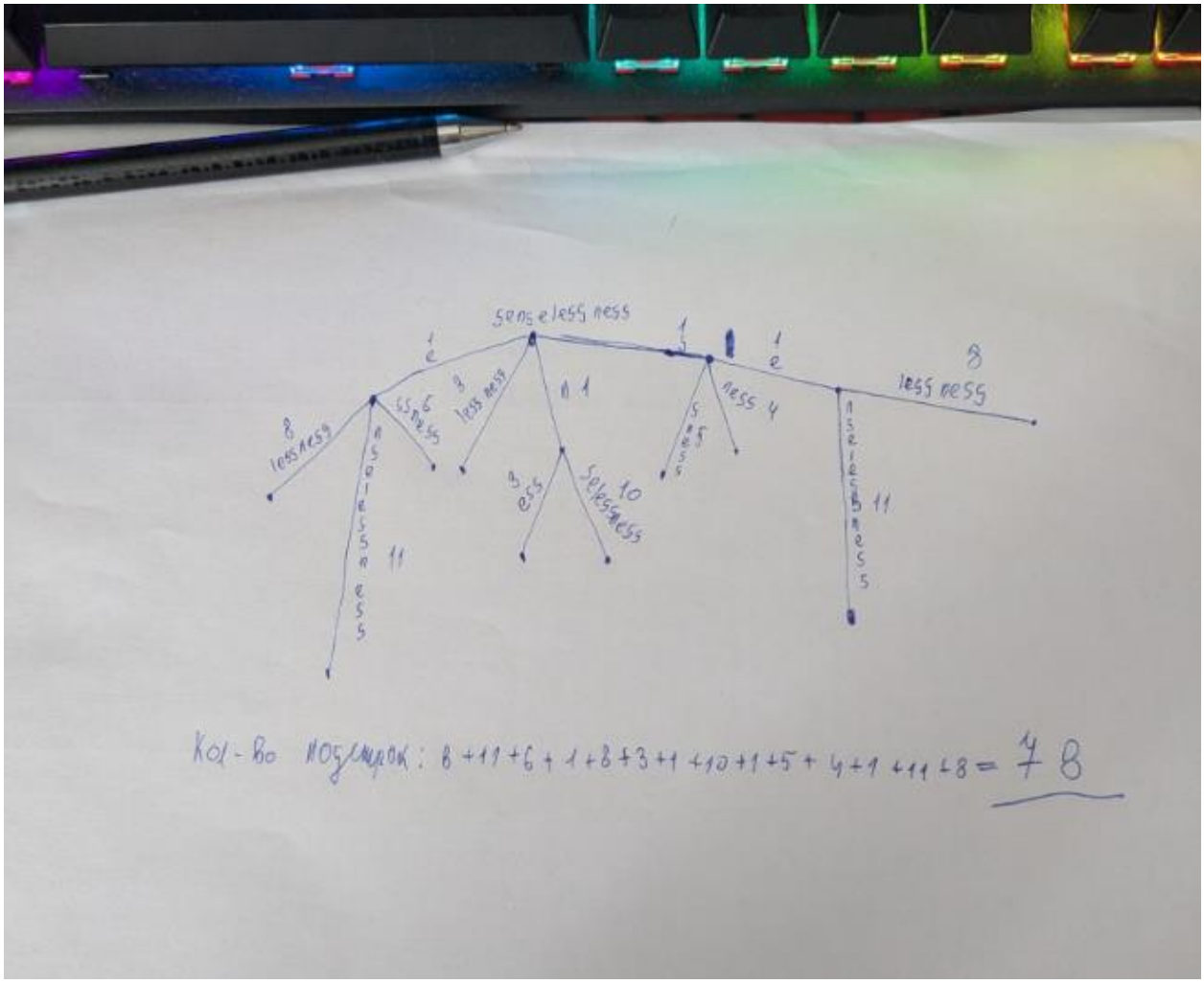
) Постройте сжатое суффиксное дерево для строки "senselessness" и найдите количество различных подстрок в этой строке. Объясните способ подсчета с использованием суффиксного дерева.

Способ подсчета: будем проходить по каждому ребру и к результату прибавлять длину строки в этом ребре



18) Модифицируйте алгоритм Дейкстры для решения задачи:
В городе есть NN площадей, соединенных MM дорогами. Известна длина каждой дороги и номера площадей aa_i, bb_i ($1 \le a_i, b_i \le N$), соединенных этой дорогой. Посчитайте количество способов добраться с площади AA до площади BB так, чтобы пройденный путь был минимален.

```
#include <iostream>

#include <vector>

#include <set>


using namespace std;

typedef long long ll;


vector<vector<pair<ll, ll>>> graph;

vector<pair<ll, ll>> visited;


int main()

{

    int n, m;

    cin >> n >> m;

    graph.resize(n + 1);

    visited.resize(n + 1, {LONG_LONG_MAX, 0});

    int a, b, c;

    for (int i = 0; i < m; i++)

    {

        cin >> a >> b >> c;

        graph[a].push_back({b, c});

        graph[b].push_back({a, c});

    }
```

```
cin >> a >> b;
multiset<pair<ll, ll>> st;
st.insert({0, a});
while (st.size())
{
    auto it = st.begin();
    ll curTime = it->first;
    ll curVert = it->second;
    st.erase(it);

    if (visited[curVert].first < curTime)
    {
        st.erase(*it);
        continue;
    }
    else if (visited[curVert].first == curTime)
        visited[curVert].second++;
    else
        visited[curVert] = {curTime, 1};

    if (curVert == b)
        continue;

    for (int i = 0; i < graph[curVert].size(); i++)
    {
        ll vert = graph[curVert][i].first;
```

```
            ll time = graph[curVert][i].second;


            if (visited[vert].first >= curTime + time)
                st.insert({curTime + time, vert});
        }
    }

    cout << visited[b].second;
}
```

19) Напишите функцию для получения KK-го в порядке возрастания числа из двоичного файла, содержащего NN (N>10^9N>109) 64-битных беззнаковых целых чисел. Функции передается имя файла с числами и KK. Можно считывать файл несколько раз. В памяти можно хранить не более 66000 64-битных чисел. Оцените эффективность вашего алгоритма.

```
#include <iostream>

#include <fstream>

#include <vector>

#include <algorithm>

using namespace std;

unsigned long long Knumber(const string& filename, int K) {

  ifstream file(filename, ios::binary);

  vector<unsigned long long> numbers;

  unsigned long long number;

  while (file.read(reinterpret_cast<char*>(&number), sizeof(unsigned long long))) {

    numbers.push_back(number);

    if (numbers.size() > 66000) {

      partial_sort(numbers.begin(), numbers.begin() + K, numbers.end());

      numbers.resize(K);

    }

  }


  if (K <= numbers.size()) {

    nth_element(numbers.begin(), numbers.begin() + K - 1, numbers.end());

    return numbers[K - 1];

  } else {

    return 0; // если К больше, чем чисел в файле

  }
```

```
}
```

Эффективность алгоритма: В данной реализации используется частичная сортировка вектора, которая выполняется за O(NlogK).