Приложения на хеширането в състезателните задачи

Соня Милева

C. Watto and Mechanism (http://codeforces.com/contest/514/problem/C)

Резюме: имаме *n* думи в паметта на машината, *m* заявки за дума *s* дали в паметта се съдържа дума със същата дължина и се различва от *s* само с 1 буква. Всяка дума съдържа само *'a', 'b', 'c'* . ($0 \le n \le 3 \cdot 10^5$, $0 \le m \le 3 \cdot 10^5$)

Анализ: *n* и *m* са твърде големи за да търсим за всяка дума, променяйки всяка нейна буква, дали присъства в паметта. Следователно трябва да хешираме думите в паметта и заявките, като всяка промяна във всяка заявка ще променя хеша на заявката и ще търси дали съществува същия хеш в паметта.

Знаем че всяка дума *s* с дължина *m*, в *B*-ична бройна система, може да бъде представена като число пресметнато по следния начин:
$H = s[0] * B^{(m - 1)} + s[1] * B^{(m - 2)} + \ldots + s[m - 2] * B^1 + s[m - 1] * B^0$, което ще използваме за да изчислим хеша на различните думи.

Когато използваме метода на хеширането трябва да имаме предвид, че могат да се получат колизии (две или повече различни думи да се преобразуват в една и съща стойност). Колизиите могат да бъдат намалени до минимум, ако изберем базата да е просто число, по-голямо от азбуката ни; а модулът по който смятаме – да е голямо и просто число. За да сме сигурни, че решението ни е вярно можем да използваме две различни хеширания или когато намерим съвпадение на две хеш-стойности да сравняваме и стринговете за които те отговарят, но тези проверки обикновенно не е нужно да се правят в състезателните задачи, тъй като отнемат много време.

Решение:

```
#include <iostream>
#include <cstdlib>
#include <cstdio>
#include <string>
#include <cstring>
#include <cmath>
#include <cctype>
#include <algorithm>
#include <sstream>

#include <vector>
#include <map>
#include <set>
#include <queue>
#include <stack>
#include <bitset>
#include <iterator>

using namespace std;

typedef pair<int, int> ii;
typedef vector<int> vi;
typedef vector<ii> vii;

#define INF 1e9
#define ll long long
#define ull unsigned long long

const ll mod=100000000000000003;
const int B = 7;
const int MAXN = 600001;
int n, m;

int main()
{
    //freopen("C:\\Users\\Sonia\\Documents\\CodeForces\\2015.02.14(291)\\a.txt", "r", stdin);
    int n, m;
    string s;
    set<ll> h;
    vector<ll> pows(MAXN+1);
    scanf("%d %d", &n, &m);
    pows[0] = 1;
    for(int i = 1; i < MAXN; i++)
    {
        pows[i] = pows[i-1]*B%mod;
    }


    for(int t= 0; t < n; t++)
    {
        cin >> s;
        int len = s.length();
        ll code = 0;
        for(int i = 0; i < len; i++)
        {
            code = (code*B + s[i] - 'a')%mod;
        }
        h.insert(code);
    }
```

```
    for(int t= 0; t < m; t++)
    {
        cin >> s;
        int len = s.length();
        ll code = 0;
        bool found = false;
        for(int i = 0; i < len; i++)
        {
            code = (code*B + s[i] - 'a')%mod;
        }
        for(int i = 0; i < len; i++)
        {
            ll cur = (code - (s[len-1-i] - 'a') * pows[i])%mod;
            if(cur < 0)
                cur += mod;
            for(int j = 0; j < 3; j++)
            {
                if(s[len-1-i] - 'a' != j)
                {
                    if(h.find((cur+j*pows[i])%mod)!=h.end())
                    {
                        found = true;
                    }
                }
            }
            if(found)
                break;
        }
        if(found)
            printf("YES\n");
        else
            printf("NO\n");
    }

    return 0;
}
```

Използвана литература:

- „Introduction to Algorithms“, 3rd Edition, T. Cormen, C. Leiserson, R. Rivest
- http://codeforces.com/blog/entry/16398