

ACM-ICPC TEAM REFERENCE DOCUMENT

HSE-NN 2

Содержание

1	Шаблон	2
2	Алгоритмы на строки	2
2.1	Префикс-функция	2
2.2	Z-функция	2
3	Алгоритмы на графах	2
3.1	Алгоритм Дейкстры $O(n^2)$	2
3.2	Алгоритм Дейкстры $O(\log(n) \cdot m)$. .	2
4	Простые алгоритмы	2
4.1	Решето Эратосфена $O(n)$	2
4.2	Решето Эратосфена $O(n \cdot \log(\log(n)))$	3
5	Структуры данных	3
5.1	Дерево отрезков	3

1 Шаблон

```
#define _USE_MATH_DEFINES
#include <iostream>
#include <algorithm>
#include <vector>
#include <string>
#include <set>
#include <queue>
#include <utility>
#include <iomanip>
#include <cstdio>
#include <cstdlib>
#include <numeric>
#include <cmath>
#include <stack>
#include <map>
#include <deque>
#include <sstream>
using namespace std;
#define int long long
typedef vector<int> vi;
typedef vector<pair<int, int>> vii;
typedef long long ll;
typedef long double ld;
// #define pi M_PI
#define all(x) (x).begin(), (x).end()
#define pb push_back
#define re return
#define fr(x) for(int i = 0; i < (x); i++)
const int inf = 1000000000 + 7;
signed main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
}
```

2 Алгоритмы на строки

2.1 Префикс-функция

```
vector<int> prefix_function (string s) {
    int n = (int) s.length();
    vector<int> pi (n);
    for (int i=1; i<n; ++i) {
        int j = pi[i-1];
        while (j > 0 && s[i] != s[j])
            j = pi[j-1];
        if (s[i] == s[j]) ++j;
        pi[i] = j;
    }
    return pi;
}
```

2.2 Z-функция

```
vector<int> z_function (string s) {
    int n = (int) s.length();
    vector<int> z (n);
    for (int i=1, l=0, r=0; i<n; ++i) {
        if (i <= r)
            z[i] = min (r-i+1, z[i-l]);
        while (i+z[i] < n && s[z[i]] == s[i+z[i]])
            ++z[i];
        if (i+z[i]-1 > r)
            l = i, r = i+z[i]-1;
    }
    return z;
}
```

3 Алгоритмы на графах

3.1 Алгоритм Дейкстры $O(n^2)$

was - брали вершину или нет

v - список смежности

d - массив расстояний для точки x

```
int d[2001];
int was[2001];
vector<pair<int, int>> v[2001];
int n;
void dijkstra(int x) {
    for (int i = 0; i < n; i++)
        d[i] = inf;
    d[x] = 0;
    for (int it = 0; it < n; it++)
    {
        int id = -1;
        for (int i = 0; i < n; i++)
            if (!was[i] && (id == -1 || d[id] > d[i]))
                id = i;
        was[id] = 1;
        for (auto p : v[id]) {
            int y = p.first;
            int t = p.second;
            d[y] = min(d[y], d[id] + t);
        }
    }
}
```

3.2 Алгоритм Дейкстры $O(\log(n) \cdot m)$

d - массив расстояний для точки x

```
int d[3001];
vector<pair<int, int>> v[3001];
bool f(int x, int y) {
    if (d[x] != d[y])
        return d[x] < d[y];
    return x < y;
}
set <int, bool(*) (int, int)> s(f);
void dijkstra(int x) {
    x--;
    for (int i = 0; i <= n; i++)
    {
        d[i] = inf;
    }
    d[x] = 0;
    s.insert(x);
    while (!s.empty()) {
        int x = *s.begin();
        s.erase(x);
        for (auto p : v[x]) {
            int y = p.first;
            int t = p.second;
            if (d[y] > d[x] + t) {
                s.erase(y);
                d[y] = d[x] + t;
                s.insert(y);
            }
        }
    }
}
```

4 Простые алгоритмы

4.1 Решето Эратосфена $O(n)$

pr - все простые числа до n

lp - минимальный простой делитель числа i

```
const int N = 10001000;
int lp[N + 1];
vector<int> pr;
void pcalc() {
    for (int i = 2; i <= N; ++i) {
        if (lp[i] == 0) {
            lp[i] = i;
            pr.push_back(i);
        }
        for (int j = 0; j < (int) pr.size() && pr[j] <= lp[i] && i * pr[j] <= N)
            lp[i * pr[j]] = pr[j];
    }
}
```

4.2 Решето Эратосфена

$$O(n \cdot \log(\log(n)))$$

$d[i] == 1$ если число i простое

```
long long d[10000000];
void calc_p(int n)
{
    d[0] = 1;
    d[1] = 1;
    for (int i = 2; i <= n; i++)
    {
        if(d[i]==0)
            for (int j = i + i; j <= n; j += i)
            {
                d[j] = 1;
            }
    }
}
```

5 Структуры данных

5.1 Дерево отрезков

```
ll t[4*100000];
void build(int v, int vl, int vr, vi& a){
    if(vl == vr){
        t[v] = a[vl];
        return;
    }
    int c = vl + (vr - vl)/2;
    build(2*v+1, vl, c, a);
    build(2*v+2, c+1, vr, a);
    t[v] = max(t[2*v+1], t[2*v+2]);
}
ll sum(int v, int vl, int vr, int l, int r){
    if(l > vr || r < vl){
        return -inf - 1;
    }
    if(l <= vl && vr <= r)
        return t[v];
    int c = vl + (vr - vl)/2;
    ll q1 = sum(2*v+1, vl, c, l, r);
    ll q2 = sum(2*v+2, c+1, vr, l, r);
    return max(q1, q2);
}
void modify(int v, int vl, int vr, int pos, int x){
    if(vl == vr){
        t[v] = x;
        return;
    }
    int c = vl + (vr - vl)/2;
    if(c >= pos)
        modify(2*v + 1, vl, c, pos, x);
    else
        modify(2*v + 2, c + 1, vr, pos, x);
    t[v] = max(t[2*v+1], t[2*v+2]);
}
```

Прибавление на отрезке

```
void update (int v, int vl, int vr, int l, int r, int add) {
    if (l > r)
        return;
    if (l == vl && vr == r)
        t[v] += add;
    else {
        int c = vl + (vr - vl)/2;
        update (v*2+1, vl, c, l, min(r,c), add);
        update (v*2+2, c+1, vr, max(l,c+1), r, add);
    }
}

int get (int v, int vl, int vr, int pos) {
    if (vl == vr)
        return t[v];
    int c = vl + (vr - vl)/2;
    if (pos <= c)
        return t[v] + get (v*2+1, vl, c, pos);
    else
        return t[v] + get (v*2+2, c+1, vr, pos);
}
```

Присвоение на отрезке

```
void push (int v) {
    if (t[v] != -1) {
        t[v*2+1] = t[v*2+2] = t[v];
        t[v] = -1;
    }
}

void update (int v, int vl, int vr, int l, int r, int color) {
    if (l > r)
        return;
    if (l == vl && vr == r)
        t[v] = color;
    else {
        push (v);
        int c = vl + (vr - vl)/2;
        update (v*2+1, vl, c, l, min(r,c), color);
        update (v*2+2, c+1, vr, max(l,c+1), r, color);
    }
}

int get (int v, int vl, int vr, int pos) {
    if (vl == vr)
        return t[v];
    push (v);
    int c = vl + (vr - vl)/2;
    if (pos <= c)
        return get (v*2+1, vl, c, pos);
    else
        return get (v*2+2, c+1, vr, pos);
}
```

TODO: Присвоение на отрезке с получением суммы