JakeMate14

Contents

1 Busqueda Binaria

1.1 LB y UP

```
\\Devuelve el maximo indice tal que v[i]<=x
int closestToTheLeft(vector<int>& v, int x) {
    int l = -1, r = v.size(),m;
    while (r > l + 1) {
        m = l+(r - l) / 2;
        if (v[m] <= x) l = m;
        else r = m;
    }
    return l;
}

\\Devuelve el indice del primer elemento v[i]>=x
int closestToTheRight(vector<int>& v, int x){
    int l=-1,r=n,m;
    while(r>l+1){
        m = l+(r-l)/2;
        if(a[m]<x)  l=m;
        else    r=m;
    }
    return r;
}
```

1.2 Busqueda con numeros reales

```
//Es asegurado que con 100 funcionara
forn(i,100){
    m = l+(r-l)/2;
    if(ok(m))   r = m;
    else        l = m;
}
cout << setprecision(20) << l << "\n";
```

2 Estructuras de Datos

2.1 Segment tree

```
struct segtree {
    int size;
    vector<ll> vv;
```

```

    void build(vector<int> &nums) {
        size = 1;
        while (size < nums.size()) size *= 2;
        vv.assign(2 * size, 0);
        build(nums, 0, 0, size);
    }

    void build(vector<int> &nums, int x, int lx, int rx) {
        if (rx - lx == 1) {
            if (lx < nums.size()) {
                vv[x] = nums[lx];
            }
        } else {
            int m = (lx + rx) / 2;
            build(nums, 2 * x + 1, lx, m);
            build(nums, 2 * x + 2, m, rx);
            vv[x] = vv[2 * x + 1] + vv[2 * x + 2];
        }
    }

    void set(int i, int v) {
        set(i, v, 0, 0, size);
    }

    void set(int i, int v, int x, int lx, int rx) {
        if (rx - lx == 1) {
            vv[x] = v;
        } else {
            int m = (lx + rx) / 2;
            if (i < m) {
                set(i, v, 2*x+1, lx, m);
            } else {
                set(i, v, 2*x+2, m, rx);
            }
            vv[x] = vv[2*x+1] + vv[2*x+2];
        }
    }

    ll sum(int l, int r) {
        return sum(l, r, 0, 0, size);
    }

    ll sum(int l, int r, int x, int lx, int rx) {
        if (r <= lx) return 0;
        if (l >= rx) return 0;
        if (lx >= l && rx <= r) return vv[x];
        int m = (lx + rx) / 2;
        ll s1 = sum(l, r, 2*x+1, lx, m);
        ll s2 = sum(l, r, 2*x+2, m, rx);
        return s1 + s2;
    }
};
```

2.2 Suffix Array

```
vector<int> sort_cyclic_shifts(string const& s) {
    int n = s.size();
    const int alphabet = 256;

    vector<int> p(n), c(n), cnt(max(alphabet, n), 0);
    for (int i = 0; i < n; i++)
        cnt[s[i]]++;
    for (int i = 1; i < alphabet; i++)
        cnt[i] += cnt[i-1];
    for (int i = 0; i < n; i++)
        p[--cnt[s[i]]] = i;
    c[p[0]] = 0;
    int classes = 1;
    for (int i = 1; i < n; i++) {
        if (s[p[i]] != s[p[i-1]])
            classes++;
        c[p[i]] = classes - 1;
    }

    vector<int> pn(n), cn(n);
    for (int h = 0; (1 << h) < n; ++h) {
        for (int i = 0; i < n; i++) {
            pn[i] = p[i] - (1 << h);
            if (pn[i] < 0)
                pn[i] += n;
        }
        fill(cnt.begin(), cnt.begin() + classes, 0);
        for (int i = 0; i < n; i++)
            cnt[c[pn[i]]]++;
        for (int i = 1; i < classes; i++)
```

```
            cnt[i] += cnt[i-1];
        for (int i = n-1; i >= 0; i--)
            p[--cnt[c[pn[i]]]] = pn[i];
        cn[p[0]] = 0;
        classes = 1;
        for (int i = 1; i < n; i++) {
            pair<int, int> cur = {c[p[i]], c[(p[i] + (1 << h)) % n]};
            pair<int, int> prev = {c[p[i-1]], c[(p[i-1] + (1 << h)) % n
                ]};
            if (cur != prev)
                ++classes;
            cn[p[i]] = classes - 1;
        }
        c.swap(cn);
    }
    return p;
}

vector<int> suffix_array_construction(string s) {
    s += "$";
    vector<int> sorted_shifts = sort_cyclic_shifts(s);
    sorted_shifts.erase(sorted_shifts.begin());
    return sorted_shifts;
}

//Para imprimer el arreglo
forn(i,s.size())
    cout << s.substr(res[i],s.size()-res[i]) << endl;
```

3 Grafos

3.1 DFS en Grids

```
void dfs(int x, int y){
    if(x<0 || y<0 || x>=n || y>=m || vis[x][y] || grafo[x][y] == 0)
        return;

    vis[x][y] = true;

    dfs(x+1,y);
    dfs(x,y+1);
    dfs(x-1,y);
    dfs(x,y-1);

    return;
}
```

4 Arboles

4.1 Distancia de cada nodo desde la raiz

```
vector<vector<int>> arbol;
vi distInd(n+1);

void dfs(int nodo, int padre, vi &a){
    for(auto u: arbol[nodo]){
        if(u==padre)    continue;
        a[u] = a[nodo]+1;
        dfs(u,nodo,a);
    }
}

dfs(1,0,distInd);
```

4.2 Distancia de la raiz a cualquier nodo

```
//Calculo de la distancia mas larga iniciando desde cualquier nodo
vector<vector<int>> arbol;
vi a(n+1),b(n+1),distInd(n+1);

void dfs(int nodo, int padre, vi &a){
    for(auto u: arbol[nodo]){
        if(u==padre)    continue;
        a[u] = a[nodo]+1;
        dfs(u,nodo,a);
    }
}

void distMaximas(){
    dfs(1,0,distInd);
    int n1 = max_element(all(distInd))-distInd.begin();

    dfs(n1,0,a);
    int n2 = max_element(all(a))-a.begin();

    dfs(n2,0,b);
```

```
21
22      //La respuesta de cada nodo i es
23      max(a[i],b[i]);
24  }
```

4.3 Diametro de un arbol

```
1   //Es la distancia mas larga entre dos nodos en el arbol
2   vector<vector<int>> arbol;
3
4   void dfs(int nodo, int padre, vi &a){
5       for(auto u: arbol[nodo]){
6           if(u==padre)    continue;
7           a[u] = a[nodo]+1;
8           dfs(u,nodo,a);
9       }
10  }
11
12  int diametro(){
13      vi a(n+1),b(n+1);
14      dfs(1,0,a);
15      int d1 = max_element(all(a))-a.begin();
16
17      dfs(d1,0,b);
18      int d2 = max_element(all(b))-b.begin();
19
20      return max(a[d1],b[d2]);
21  }
```

4.4 Tamaño del subarbol del nodo x

```
1   //Se calcula la cantidad de nodos del subarbol del nodo x
2   const int maxV = 2*1e5+1;
3   vector<vector<int>> arbol;
4   int cantNodos[maxV];
5
6   void dfs(int nodo, int padre){
7       cantNodos[nodo] = 1;
8       for(int u: arbol[nodo]){
9           if(u==padre) continue;
10          dfs(u,nodo);
11          cantNodos[nodo]+=cantNodos[u];
12      }
13  }
14
15  dfs(1,0);
```

5 Math

6 DP

6.1 Coin Problem

```
1   int dp[ESTADO_MAX+1];
2   vector<int>coins(ESTADO_MAX+1,INF);
3
4   dp[0] = 0;
5   memset(dp,INF,sizeof(dp));
6   for (int x = 1; x <= n; x++) {
7       value[x] = INF;
8       for (auto c : coins) {
9           if (x-c >= 0) {
10              dp[x] = min(dp[x], dp[x-c]+1);
11          }
12      }
13  }
```

6.2 Digitos

```
1   res = solve(b) - solve(a-1);
2   vector<int>num;
3   int dp[20][20][2];
4
5   int solve(lli b){
6       num.clear();
7       while(b>0){
8           num.push_back(b%10);
9           b/=10;
10      }
11      reverse(num.begin(), num.end());
12
13      memset(dp, -1, sizeof(dp));
14      lli res = mem(0, 0, 0);
15      return res;
16  }
17
18  //Numeros con a los mas 3 digitos distintos de cero
```

```
//4, 200000, 10203
int mem(int pos, int cant, int goodAll){
    if(cant>3)  return 0;
    if(pos==num.size()){
        if(cant<=3) return 1;
        return 0;
    }

    int &a = dp[pos][cant][goodAll];
    if(a!=-1)   return a;
    a = 0;

    int limite = goodAll==0?num[pos]:9;
    fore(dig,0,limite){
        int nG = goodAll;
        int nCant = cant;
        if(goodAll==0 && dig<limite)    nG=1;
        if(dig!=0)  nCant++;
        if(nCant<=3)    a+=mem(pos+1,nCant,nG);
    }

    return a;
}

//Numeros donde el digito d ocurre exactamente k veces
int call(int pos, int cnt, int f){
    if(cnt > k) return 0;

    if(pos == num.size()){
        if(cnt == k)    return 1;
        return 0;
    }

    if(DP[pos][cnt][f] != -1)   return DP[pos][cnt][f];
    int res = 0;

    int LMT;

    if(f == 0)  LMT = num[pos];
    else        LMT = 9;

    for(int dgt = 0; dgt<=LMT; dgt++){
        int nf = f;
        int ncnt = cnt;
        if(f == 0 && dgt < LMT) nf = 1;
        if(dgt == d)    ncnt++;
        if(ncnt <= k)   res += call(pos+1, ncnt, nf);
    }

    DP[pos][cnt][f] = res;
    return DP[pos][cnt][f];
}
```

7 Geometry

8 Strings

9 Flow

10 Other

10.1 Template

```
#include<bits/stdc++.h>
using namespace std;

typedef long long int lli;
typedef vector<int> vi;
#define IO  ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
#define forn(i,n)   for(lli (i)=0; i<n; i++)
#define forr(i,a,n) for(lli i=(a); i<n; i++)
#define fore(i,a,n) for(lli i=(a); i<=n; i++)
#define all(v)      v.begin(),v.end()
#define borra(s)    s.erase(unique(all(s)),s.end())
#define YES         cout << "YES\n"
#define NO          cout << "NO\n"
#define debug(a)    cout << a << "\n"

void sol(){
}

int main(){IO
    int t=1;//cin>>t;
    while(t--)sol();
}
```