Algorithm Library

magic::team.getname()

South China Normal University

August 29, 2024

Contents

头文件

DEBUG 头

```cpp
#include <bits/stdc++.h>
using namespace std;
using i64=long long;
using i128=__int128;

namespace DBG
{
    template <class T>
    void _dbg(const char *f,T t) { cerr<<f<<'='<<t<<'\n'; }

    template <class A,class... B>
    void _dbg(const char *f,A a,B... b)
    {
        while (*f!=',') cerr<<*f++;
        cerr<<'='<<a<<",";
        _dbg(f+1,b...);
    }

    template <class T>
    ostream& operator << (ostream& os,const vector<T> &v)
    {
        os<<"[ ";
        for (const auto &x:v) os<<x<<", ";
        os<<"]";
        return os;
    }

    #define dbg(...) _dbg(#__VA_ARGS__, __VA_ARGS__)
}

using namespace DBG;
```

__int128 输出流

```cpp
ostream &operator << (ostream &os,i128 n)
{
    string s;
    bool neg=n<0;
    if (neg) n=-n;
    while (n)
    {
        s+='0'+n%10;
        n/=10;
    }
    if (neg) s+='-';
    reverse(s.begin(),s.end());
    if (s.empty()) s+='0';
    return os<<s;
}
```

常用数学函数

```cpp
i64 ceilDiv(i64 n,i64 m)
{
    if (n>=0) return (n+m-1)/m;
    else return n/m;
}

i64 floorDiv(i64 n,i64 m)
{
    if (n>=0) return n/m;
    else return (n-m+1)/m;
}

i128 gcd(i128 a,i128 b)
{
```

```
    return b?gcd(b,a%b):a;
}
```

数学

欧拉筛

```
vector<int> minp,primes;

void sieve(int n)
{
    minp.assign(n+1,0);
    primes.clear();
    for (int i=2;i<=n;i++)
    {
        if (!minp[i])
        {
            minp[i]=i;
            primes.push_back(i);
        }
        for (auto p:primes)
        {
            if (i*p>n) break;
            minp[i*p]=p;
            if (p==minp[i]) break;
        }
    }
}
```

取模类（MInt）

```
template <class T>
constexpr T power(T a,i64 b)
{
    T res=1;
    for (;b;b>>=1,a*=a)
        if (b&1) res*=a;
    return res;
}

template <int P>
struct MInt
{
    int x;
    constexpr MInt():x{} {}
    constexpr MInt(i64 x):x{norm(x%getMod())} {}

    static int Mod;
    constexpr static int getMod()
    {
        if (P>0) return P;
        else return Mod;
    }

    constexpr static void setMod(int Mod_) { Mod=Mod_; }

    constexpr int norm(int x) const
    {
        if (x<0) x+=getMod();
        if (x>=getMod()) x-=getMod();
        return x;
    }

    constexpr int val() const { return x; }

    explicit constexpr operator int () const { return x; }

    constexpr MInt operator - () const
    {
```

```
        MInt res;
        res.x=norm(getMod()-x);
        return res;
    }

    constexpr MInt inv() const
    {
        assert(x!=0);
        return power(*this,getMod()-2);
    }

    constexpr MInt &operator *= (MInt rhs) &
    {
        x=1ll*x*rhs.x%getMod();
        return *this;
    }

    constexpr MInt &operator += (MInt rhs) &
    {
        x=norm(x+rhs.x);
        return *this;
    }

    constexpr MInt &operator -= (MInt rhs) &
    {
        x=norm(x-rhs.x);
        return *this;
    }

    constexpr MInt &operator /= (MInt rhs) &
    {
        return *this*=rhs.inv();
    }

    friend constexpr MInt operator * (MInt lhs,MInt rhs)
    {
        MInt res=lhs;
        res*=rhs;
        return res;
    }

    friend constexpr MInt operator + (MInt lhs,MInt rhs)
    {
        MInt res=lhs;
        res+=rhs;
        return res;
    }

    friend constexpr MInt operator - (MInt lhs,MInt rhs)
    {
        MInt res=lhs;
        res-=rhs;
        return res;
    }

    friend constexpr MInt operator / (MInt lhs,MInt rhs)
    {
        MInt res=lhs;
        res/=rhs;
        return res;
    }

    friend constexpr istream &operator >> (istream &is,MInt &a)
    {
        i64 v;
        is>>v;
        a=MInt(v);
        return is;
    }

    friend constexpr ostream &operator << (ostream &os,const MInt &a) { return os<<a.val(); }
```

```
    friend constexpr bool operator == (MInt lhs,MInt rhs) { return lhs.val()==rhs.val(); }

    friend constexpr bool operator != (MInt lhs,MInt rhs) { return lhs.val()!=rhs.val(); }
};

template<>
int MInt<0>::Mod=1;

template<int V,int P>
constexpr MInt<P> CInv=MInt<P>(V).inv();
```

数据结构

并查集（启发式合并 + 带撤销）

```
struct DSU
{
    int n=0;
    vector<int> fa,siz;
    stack<int> s;

    DSU(int n) { init(n); }

    void init(int n)
    {
        fa.resize(n);
        iota(fa.begin(),fa.end(),0);
        siz.assign(n,1);
        while (!s.empty()) s.pop();
    }

    int get(int x) { return fa[x]==x?x:get(fa[x]); }

    void merge(int x,int y)
    {
        x=get(x),y=get(y);
        if (x==y) return;
        if (siz[x]<siz[y]) swap(x,y);
        s.push(y),fa[y]=x,siz[x]+=siz[y];
    }

    void undo()
    {
        if (s.empty()) return;
        int y=s.top();
        s.pop();
        siz[fa[y]]-=siz[y];
        fa[y]=y;
    }

    void back(int t=0) { while (s.size()>t) undo(); }
};
```

状压 RMQ

```
template <class T,class Cmp=less<T>>
struct RMQ
{
    const Cmp cmp=Cmp();
    static constexpr unsigned B=64;
    using u64=unsigned long long;
    int n;
    vector<vector<T>> a;
    vector<T> pre,suf,ini;
    vector<u64> stk;

    RMQ() {}
    RMQ(const vector<T> &v) { init(v); }
```

```
    void init(const vector<T> &v)
    {
        n=v.size();
        pre=suf=ini=v;
        stk.resize(n);
        if (!n) return;
        const int M=(n-1)/B+1;
        const int lg=__lg(M);
        a.assign(lg+1,vector<T>(M));
        for (int i=0;i<M;i++)
        {
            a[0][i]=v[i*B];
            for (int j=1;j<B&&i*B+j<n;j++)
                a[0][i]=min(a[0][i],v[i*B+j],cmp);
        }
        for (int i=1;i<n;i++)
            if (i%B) pre[i]=min(pre[i],pre[i-1],cmp);
        for (int i=n-2;i>=0;i--)
            if (i%B!=B-1) suf[i]=min(suf[i],suf[i+1],cmp);
        for (int j=0;j<lg;j++)
            for (int i=0;i+(2<<j)<=M;i++)
                a[j+1][i]=min(a[j][i],a[j][i+(1<<j)],cmp);
        for (int i=0;i<M;i++)
        {
            const int l=i*B;
            const int r=min(1U*n,l+B);
            u64 s=0;
            for (int j=l;j<r;j++)
            {
                while (s&&cmp(v[j],v[__lg(s)+l])) s^=1ULL<<__lg(s);
                s|=1ULL<<(j-l);
                stk[j]=s;
            }
        }
    }

    //查询区间 [l,r) 的 RMQ
    T operator()(int l,int r)
    {
        if (l/B!=(r-1)/B)
        {
            T ans=min(suf[l],pre[r-1],cmp);
            l=l/B+1,r=r/B;
            if (l<r)
            {
                int k=__lg(r-l);
                ans=min({ans,a[k][l],a[k][r-(1<<k)]},cmp);
            }
            return ans;
        }
        else
        {
            int x=B*(l/B);
            return ini[__builtin_ctzll(stk[r-1]>>(l-x))+l];
        }
    }
};
```

树状数组

```
template <class T>
struct BIT
{
    int n;
    vector<T> a;

    BIT(int n_=0) { init(n_); }

    void init(int n_)
    {
```

```
        n=n_;
        a.assign(n,T{});
    }

    void add(int x,const T &v)
    {
        for (int i=x+1;i<=n;i+=i&-i)
            a[i-1]=a[i-1]+v;
    }

    //查询区间 [0,x)
    T sum(int x)
    {
        T ans{};
        for (int i=x;i>0;i-=i&-i)
            ans=ans+a[i-1];
        return ans;
    }

    //查询区间 [l,r)
    T rangeSum(int l,int r) { return sum(r)-sum(l); }

    int select(const T &k)
    {
        int x=0;
        T cur{};
        for (int i=1<<__lg(n);i;i>>=1)
        {
            if (x+i<=n&&cur+a[x+i-1]<=k)
            {
                x+=i;
                cur=cur+a[x-1];
            }
        }
        return x;
    }
};
```

线段树

```
template <class Info,class Tag>
struct SGT
{
    int n;
    vector<Info> info;
    vector<Tag> tag;

    SGT():n(0) {}
    SGT(int n_,Info v_=Info()) { init(n_,v_); }

    template <class T>
    SGT(vector<T> init_) { init(init_); }

    void init(int n_,Info v_=Info()) { init(vector(n_,v_)); }

    template <class T>
    void init(vector<T> init_)
    {
        n=init_.size();
        info.assign(4<<__lg(n),Info());
        tag.assign(4<<__lg(n),Tag());
        function<void(int,int,int)> build=[&](int p,int l,int r)
        {
            if (r-l==1)
            {
                info[p]=init_[l];
                return;
            }
            int m=(l+r)>>1;
            build(p<<1,l,m);
            build(p<<1|1,m,r);
```

```
            pushup(p);
        };
        build(1,0,n);
    }

    void pushup(int p) { info[p]=info[p<<1]+info[p<<1|1]; }

    void apply(int p,const Tag &v)
    {
        info[p].apply(v);
        tag[p].apply(v);
    }

    void pushdown(int p)
    {
        apply(p<<1,tag[p]);
        apply(p<<1|1,tag[p]);
        tag[p]=tag();
    }

    void modify(int p,int l,int r,int x,const Info &v)
    {
        if (r-l==1)
        {
            info[p]=v;
            return;
        }
        int m=(l+r)>>1;
        pushup(p);
        if (x<m) modify(p<<1,l,m,x,v);
        else modify(p<<1|1,m,r,x,v);
    }

    //O(log n) 单点修改
    void modify(int p,const Info &v) { modify(1,0,n,p,v); }

    Info rangeQuery(int p,int l,int r,int x,int y)
    {
        if (l>=y||r<=x) return Info();
        if (l>=x&&r<=y) return info[p];
        int m=(l+r)>>1;
        pushdown(p);
        return rangeQuery(p<<1,l,m,x,y)+rangeQuery(p<<1|1,m,r,x,y);
    }

    //O(log n) 区间查询 [l,r)
    Info rangeQuery(int l,int r) { rangeQuery(1,0,n,l,r); }

    void rangeApply(int p,int l,int r,int x,int y,const Tag &v)
    {
        if (l>=y||r<=x) return;
        if (l>=x&&r<=y)
        {
            apply(p,v);
            return;
        }
        int m=(l+r)>>1;
        pushdown(p);
        rangeApply(p<<1,l,m,x,y,v);
        rangeApply(p<<1|1,m,r,x,y,v);
        pushup(p);
    }

    //O(log n) 区间操作 [l,r)
    void rangeApply(int l,int r,const Tag &v) { rangeApply(1,0,n,l,r,v); }

    //O(log n) 区间 [l,r) 内查找第一个合法位置
    template <class F>
    int findFirst(int p,int l,int r,int x,int y,F pred)
    {
        if (l>=y||r<=x||!pred(info[p])) return -1;
```

```
        if (r-l==1) return l;
        int m=(l+r)>>1;
        pushdown(p);
        int res=findFirst(p<<1,l,m,x,y,pred);
        if (res==-1) res=findFirst(p<<1|1,m,r,x,y,pred);
        return res;
    }

    template <class F>
    int findFirst(int l,int r,F pred) { return findFirst(1,0,n,l,r,pred); }

    template <class F>
    int findLast(int p,int l,int r,int x,int y,F pred)
    {
        if (l>=y||r<=x||!pred(info[p])) return -1;
        if (r-l==1) return l;
        int m=(l+r)>>1;
        pushdown(p);
        int res=findFirst(p<<1|1,m,r,x,y,pred);
        if (res==-1) res=findFirst(p<<1,l,m,x,y,pred);
        return res;
    }

    template <class F>
    int findLast(int l,int r,F pred) { return findLast(1,0,n,l,r,pred); }
};

//这里默认乘法优先 (x*a+b)*c+d=x*(a*c)+(b*c+d)
struct Tag
{
    i64 a=1,b=0;
    void apply(Tag t)
    {
        a*=t.a;
        b=b*t.a+t.b;
    }
};

struct Info
{
    i64 x=0,l=0,r=0;
    void apply(Tag t)
    {
        int len=r-l+1;
        x=x*t.a+len*t.b;
    }
};

Info operator + (Info a,Info b)
{
    return {a.x+b.x,min(a.l,b.l),max(a.r,b.r)};
}
```

字符串

字符串哈希（随机模数）

```
bool isPrime(int n)
{
    if (n<=1) return 0;
    for (int i=2;i*i<=n;i++)
        if (n%i==0) return 0;
    return 1;
}

int findPrime(int n)
{
    while (!isPrime(n)) n++;
    return n;
}
```

```cpp

mt19937 rng(time(0));
const int P=findPrime(rng()%900000000+100000000);
struct StrHash
{
    int n;
    vector<int> h,p;

    StrHash(const string &s){ init(s); }

    void init(const string &s)
    {
        n=s.size();
        h.resize(n+1);
        p.resize(n+1);
        p[0]=1;
        for (int i=0;i<n;i++) h[i+1]=(10ll*h[i]+s[i]-'a')%P;
        for (int i=0;i<n;i++) p[i+1]=10ll*p[i]%P;
    }

    //查询 [l,r) 的区间哈希
    int get(int l,int r) { return (h[r]+1ll*(P-h[l])*p[r-l])%P; }
};
```

KMP

```cpp
vector<int> KMP(const string &s)
{
    int now=0;
    vector<int> pre(s.size(),0);
    for (int i=1;i<s.size();i++)
    {
        while (now&&s[i]!=s[now]) now=pre[now-1];
        if (s[i]==s[now]) now++;
        pre[i]=now;
    }
    return pre;
}
```

Z 函数

```cpp
vector<int> zFunction(string s)
{
    int n=s.size();
    vector<int> z(n);
    z[0]=n;
    for (int i=1,j=1;i<n;i++)
    {
        z[i]=max(0,min(j+z[j]-i,z[i-j]));
        while (i+z[i]<n&&s[z[i]]==s[i+z[i]]) z[i]++;
        if (i+z[i]>j+z[j]) j=i;
    }
    return z;
}
```

AC 自动机

```cpp
struct ACAM
{
    static constexpr int ALPHABET=26;
    struct Node
    {
        int len;
        int link;
        array<int,ALPHABET> next;
        Node():len{0},link{0},next{}{}
    };

    vector<Node> t;
```

```
    ACAM() { init(); }

    void init()
    {
        t.assign(2,Node());
        t[0].next.fill(1);
        t[0].len=-1;
    }

    int newNode()
    {
        t.emplace_back();
        return t.size()-1;
    }

    int add(const string &a)
    {
        int p=1;
        for (auto c:a)
        {
            int x=c-'a';
            if (t[p].next[x]==0)
            {
                t[p].next[x]=newNode();
                t[t[p].next[x]].len=t[p].len+1;
            }
            p=t[p].next[x];
        }
        return p;
    }

    void work()
    {
        queue<int> q;
        q.push(1);
        while (!q.empty())
        {
            int x=q.front();
            q.pop();
            for (int i=0;i<ALPHABET;i++)
            {
                if (t[x].next[i]==0) t[x].next[i]=t[t[x].link].next[i];
                else
                {
                    t[t[x].next[i]].link=t[t[x].link].next[i];
                    q.push(t[x].next[i]);
                }
            }
        }
    }

    int next(int p,int x) { return t[p].next[x]; }

    int link(int p) { return t[p].link; }

    int size() { return t.size(); }
};
```

后缀数组

```
struct SA
{
    int n;
    vector<int> sa,rk,lc;
    SA(const string &s)
    {
        n=s.length();
        sa.resize(n);
        rk.resize(n);
        lc.resize(n-1);
```

```
        iota(sa.begin(),sa.end(),0);
        sort(sa.begin(),sa.end(),[&](int a,int b){ return s[a]<s[b]; });
        rk[sa[0]]=0;
        for (int i=1;i<n;i++) rk[sa[i]]=rk[sa[i-1]]+(s[sa[i]]!=s[sa[i-1]]);
        int k=1;
        vector<int> tmp,cnt(n);
        tmp.reserve(n);
        while (rk[sa[n-1]]<n-1)
        {
            tmp.clear();
            for (int i=0;i<k;i++) tmp.push_back(n-k+i);
            for (auto i:sa)
                if (i>=k) tmp.push_back(i-k);
            fill(cnt.begin(),cnt.end(),0);
            for (int i=0;i<n;i++) cnt[rk[i]]++;
            for (int i=1;i<n;i++) cnt[i]+=cnt[i-1];
            for (int i=n-1;i>=0;i--) sa[--cnt[rk[tmp[i]]]]=tmp[i];
            swap(rk,tmp);
            rk[sa[0]]=0;
            for (int i=1;i<n;i++)
                rk[sa[i]]=rk[sa[i-1]]+(tmp[sa[i-1]]<tmp[sa[i]]||sa[i-1]+k==n||tmp[sa[i-1]+k]<tmp[sa[i]+k]);
            k<<=1;
        }
        for (int i=0,j=0;i<n;i++)
        {
            if (rk[i]==0) j=0;
            else
            {
                for (j-=j>0;i+j<n&&sa[rk[i]-1]+j<n&&s[i+j]==s[sa[rk[i]-1]+j];) j++;
                lc[rk[i]-1]=j;
            }//lc[i]:lcp(sa[i],sa[i+1]),lcp(sa[i],sa[j])=min{lc[i...j-1]}
        }
    }
};
```

（广义）后缀自动机

```
struct SAM
{
    static constexpr int ALPHABET=26;
    struct Node
    {
        int len;
        int link;
        array<int,ALPHABET> next;
        Node():len{},link{},next{} {}
    };

    vector<Node> t;

    SAM() { init(); }

    void init()
    {
        t.assign(2,Node());
        t[0].next.fill(1);
        t[0].len=-1;
    }

    int newNode()
    {
        t.emplace_back();
        return t.size()-1;
    }

    int extend(int lst,int c)
    {
        if (t[lst].next[c]&&t[t[lst].next[c]].len==t[lst].len+1)
            return t[lst].next[c];
        int p=lst,np=newNode(),flag=0;
        t[np].len=t[p].len+1;
```

```
        while (!t[p].next[c])
        {
            t[p].next[c]=np;
            p=t[p].link;
        }
        if (!p)
        {
            t[np].link=1;
            return np;
        }
        int q=t[p].next[c];
        if (t[q].len==t[p].len+1)
        {
            t[np].link=q;
            return np;
        }
        if (p==lst) flag=1,np=0,t.pop_back();
        int nq=newNode();
        t[nq].link=t[q].link;
        t[nq].next=t[q].next;
        t[nq].len=t[p].len+1;
        t[q].link=t[np].link=nq;
        while (p&&t[p].next[c]==q)
        {
            t[p].next[c]=nq;
            p=t[p].link;
        }
        return flag?nq:np;
    }

    int add(const string &a)
    {
        int p=1;
        for (auto c:a) p=extend(p,c-'a');
        return p;
    }

    int next(int p,int x) { return t[p].next[x]; }

    int link(int p) { return t[p].link; }

    int len(int p) { return t[p].len; }

    int size() { return t.size(); }
};
```

Manacher

```
vector<int> manacher(vector<int> s)
{
    vector<int> t{0};
    for (auto c:s)
    {
        t.push_back(c);
        t.push_back(0);
    }
    int n=t.size();
    vector<int> r(n);
    for (int i=0,j=0;i<n;i++)
    {
        if (j*2-i>=0&&j+r[j]>i) r[i]=min(r[j*2-i],j+r[j]-i);
        while (i-r[i]>=0&&i+r[i]<n&&t[i-r[i]]==t[i+r[i]]) r[i]++;
        if (i+r[i]>j+r[j]) j=i;
    }
    return r;
}
```

回文自动机

```
struct PAM
{
    static constexpr int ALPHABET_SIZE=28;
    struct Node
    {
        int len,link,cnt;
        array<int,ALPHABET_SIZE> next;
        Node():len{},link{},cnt{},next{}{}
    };
    vector<Node> t;
    int suff;
    string s;

    PAM() { init(); }

    void init()
    {
        t.assign(2,Node());
        t[0].len=-1;
        suff=1;
        s.clear();
    }

    int newNode()
    {
        t.emplace_back();
        return t.size()-1;
    }

    bool add(char c,char offset='a')
    {
        int pos=s.size();
        s+=c;
        int let=c-offset;
        int cur=suff,curlen=0;
        while (1)
        {
            curlen=t[cur].len;
            if (pos-curlen-1>=0&&s[pos-curlen-1]==s[pos]) break;
            cur=t[cur].link;
        }
        if (t[cur].next[let])
        {
            suff=t[cur].next[let];
            return 0;
        }
        int num=newNode();
        suff=num;
        t[num].len=t[cur].len+2;
        t[cur].next[let]=num;
        if (t[num].len==1)
        {
            t[num].link=t[num].cnt=1;
            return 1;
        }
        while (1)
        {
            cur=t[cur].link;
            curlen=t[cur].len;
            if (pos-curlen-1>=0&&s[pos-curlen-1]==s[pos])
            {
                t[num].link=t[cur].next[let];
                break;
            }
        }
        t[num].cnt=t[t[num].link].cnt+1;
        return 1;
    }
};
```