上海理工大学 ICPC 校队模板

HiedanoAkyuu、Oneman233、KR12138

2019 年 10 月 16 日

目录

1 字符串

1.1 KMP

1.2 EX-KMP

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

/*
    ex[i]: s1[i..l1-1]与s2的最大公共前缀长度
    exnext[i]: s2[i..l2-1]与s2的最大公共前缀长度
    get_exnext(s2) 求exnext[]
    exkmp(s1,s2) 求ex[]
*/

const int N=50005;
char s1[N],s2[N];
int ex[N],exnext[N];

void get_exnext(char s[N])
{
    int n=strlen(s),p=1,j,i;
    exnext[0]=n;
    for(i=0;i<n-1&&s[i]==s[i+1];i++);
    exnext[1]=i;
    for(i=2;i<n;i++)
        if(exnext[i-p]+i<p+exnext[p])
            exnext[i]=exnext[i-p];
        else
        {
            j=exnext[p]+p-i;
            if(j<0) j=0;
            while(i+j<n&&s[j]==s[i+j]) j++;
            exnext[i]=j;
            p=i;
        }
}

void exkmp(char s1[N],char s2[N])
{
    int l1=strlen(s1),l2=strlen(s2),p=0,i,j;
    get_exnext(s2);
    for(i=0;i<l1&&i<l2&&s1[i]==s2[i];i++);
    ex[0]=i;
    for(int i=1;i<l1;i++)
    {
        if(exnext[i-p]+i<p+ex[p])
            ex[i]=exnext[i-p];
        else
        {
            j=ex[p]+p-i;
            if(j<0) j=0;
            while(i+j<l1&&s1[i+j]==s2[j]) j++;
            ex[i]=j;
            p=i;
        }
    }
}
```

1.3 Manacher

1.4 串的最小表示

1.5 后缀数组

1.5.1 倍增 SA

1.5.2 DC3

1.6 回文自动机

1.7 AC 自动机

1.7.1 多模匹配

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

/*
    trie静态开点+trie图优化
*/

int sz,hd=1,nxt[1000005][26],fail[1000005],id
    [1000005],n;
char s[1000005];

void trie_clean()
{
    sz=1;
    memset(nxt,0,sizeof(nxt));
    memset(fail,0,sizeof(fail));
    memset(id,0,sizeof(id));
}

void trie_insert(int head,char s[],int len,int idx)
{
    int p=head;
    for(int i=0;i<len;i++)
    {
        int c=s[i]-'a';
        if(!nxt[p][c]) nxt[p][c]=++sz;
        p=nxt[p][c];
    }
    id[p]+=idx;
}

void acatm_build(int head)
{
    int p,tp;
    queue<int> q;
    q.push(head);
    fail[head]=0;
    while(!q.empty())
    {
        p=q.front();
        q.pop();
        for(int i=0;i<26;i++)
            if(nxt[p][i])
            {
                fail[nxt[p][i]]=p==head?head:nxt[fail[p
                    ]][i];
                q.push(nxt[p][i]);
            }
            else
                nxt[p][i]=p==head?head:nxt[fail[p]][i];
    }
```

```
}

int acatm_match(int head,char s[],int len)
{
    int p=head,ret=0;
    for(int i=0;i<len;i++)
    {
        int c=(int)s[i]-'a';
        p=nxt[p][c];
        for(int tp=p;tp;tp=fail[tp])
            if(id[tp]) ret++;
    }
    return ret;
}
```

1.7.2 自动机上 DP

1.8 后缀自动机

2 计算几何

2.1 二维几何

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

#define db double
const db EPS=1e-9;
inline int sign(db a){return a<-EPS?-1:a>EPS;}
inline int cmp(db a,db b){return sign(a-b);}
struct P
{
    db x,y;
    P(){}
    P(db x,db y):x(x),y(y){}
    P operator+(P p){return {x+p.x,y+p.y};}
    P operator-(P p){return {x-p.x,y-p.y};}
    P operator*(db d){return {x*d,y*d};}
    P operator/(db d){return {x/d,y/d};}
    bool operator<(P p) const
    {
        int c=cmp(x,p.x);
        if(c) return c==-1;
        return cmp(y,p.y)==-1;
    }
    bool operator==(P o) const
    {
        return cmp(x,o.x)==0&&cmp(y,o.y)==0;
    }
    db distTo(P p){return (*this-p).abs();}
    db alpha(){return atan2(y,x);}
    void read(){scanf("%lf%lf",&x,&y);}
    void write(){printf("(%.10f,%.10f)\n",x,y);}
    db abs(){return sqrt(abs2());}
    db abs2(){return x*x+y*y;}
    P rot90(){return P(-y,x);}
    P unit(){return *this/abs();}
    int quad() const {return sign(y)==1||(sign(y)==0&&
        sign(x)>=0);}
    db dot(P p){return x*p.x+y*p.y;}
    db det(P p){return x*p.y-y*p.x;}
    P rot(db an){return {x*cos(an)-y*sin(an),x*sin(an)
        +y*cos(an)};}
};

//For segment
#define cross(p1,p2,p3) ((p2.x-p1.x)*(p3.y-p1.y)-(p3.
    x-p1.x)*(p2.y-p1.y))
#define crossOp(p1,p2,p3) sign(cross(p1,p2,p3))

bool chkLL(P p1,P p2,P q1,P q2) //0:parallel
{
    db a1=cross(q1,q2,p1),a2=-cross(q1,q2,p2);
    return sign(a1+a2)!=0;
}

P isLL(P p1,P p2,P q1,P q2) //crossover point if
    chkLL()
{
    db a1=cross(q1,q2,p1),a2=-cross(q1,q2,p2);
    return (p1*a2+p2*a1)/(a1+a2);
}

bool intersect(db l1,db r1,db l2,db r2)
{
    if(l1>r1) swap(l1,r1);if(l2>r2) swap(l2,r2);
    return !(cmp(r1,l2)==-1||cmp(r2,l1)==-1);
}

bool isSS(P p1,P p2,P q1,P q2)
{
    return intersect(p1.x,p2.x,q1.x,q2.x)&&intersect(
        p1.y,p2.y,q1.y,q2.y)&&
    crossOp(p1,p2,q1)*crossOp(p1,p2,q2)<=0&&crossOp(q1
        ,q2,p1)*crossOp(q1,q2,p2)<=0;
}

bool isSS_strict(P p1,P p2,P q1,P q2)
{
    return crossOp(p1,p2,q1)*crossOp(p1,p2,q2)<0
    &&crossOp(q1,q2,p1)*crossOp(q1,q2,p2)<0;
}

bool isMiddle(db a,db m,db b)
{
    return sign(a-m)==0||sign(b-m)==0||(a<m!=b<m);
}

bool isMiddle(P a,P m,P b)
{
    return isMiddle(a.x,m.x,b.x)&&isMiddle(a.y,m.y,b.y
        );
}

bool onSeg(P p1,P p2,P q)
{
    return crossOp(p1,p2,q)==0&&isMiddle(p1,q,p2);
}

bool onSeg_strict(P p1,P p2,P q)
{
    return crossOp(p1,p2,q)==0&&sign((q-p1).dot(p1-p2)
        )*sign((q-p2).dot(p1-p2))<0;
}

P proj(P p1,P p2,P q)
{
    P dir=p2-p1;
    return p1+dir*(dir.dot(q-p1)/dir.abs2());
```

```
}

P reflect(P p1,P p2,P q)
{
    return proj(p1,p2,q)*2-q;
}

db nearest(P p1,P p2,P q)
{
    P h=proj(p1,p2,q);
    if(isMiddle(p1,h,p2))
        return q.distTo(h);
    return min(p1.distTo(q),p2.distTo(q));
}

db disSS(P p1,P p2,P q1,P q2) //dist of 2 segments
{
    if(isSS(p1,p2,q1,q2)) return 0;
    return min(min(nearest(p1,p2,q1),nearest(p1,p2,q2)
        ),min(nearest(q1,q2,p1),nearest(q1,q2,p2)));
}

db rad(P p1,P p2)
{
    return atan2l(p1.det(p2),p1.dot(p2));
}

db area(vector<P> ps)
{
    db ret=0;
    for(int i=0;i<ps.size();i++)
        ret+=ps[i].det(ps[(i+1)%ps.size()]);
    return ret/2;
}

int contain(vector<P> ps,P p) //2:inside,1:on_seg,0:
    outside
{
    int n=ps.size(),ret=0;
    for(int i=0;i<n;i++)
    {
        P u=ps[i],v=ps[(i+1)%n];
        if(onSeg(u,v,p)) return 1;
        if(cmp(u.y,v.y)<=0) swap(u,v);
        if(cmp(p.y,u.y)>0||cmp(p.y,v.y)<=0) continue;
        ret^=crossOp(p,u,v)>0;
    }
    return ret*2;
}

vector<P> convexHull(vector<P> ps)
{
    int n=ps.size();if(n<=1) return ps;
    sort(ps.begin(),ps.end());
    vector<P> qs(n*2);int k=0;
    for(int i=0;i<n;qs[k++]=ps[i++])
        while(k>1&&crossOp(qs[k-2],qs[k-1],ps[i])<=0)
            --k;
    for(int i=n-2,t=k;i>=0;qs[k++]=ps[i--])
        while(k>t&&crossOp(qs[k-2],qs[k-1],ps[i])<=0)
            --k;
    qs.resize(k-1);
    return qs;
}

db convexDiameter(vector<P> ps)
{
    int n=ps.size();if(n<=1) return 0;
    int is=0,js=0;
    for(int k=1;k<n;k++) is=ps[k]<ps[is]?k:is,js=ps[js
        ]<ps[k]?js:k;
    int i=is,j=js;
    db ret=ps[i].distTo(ps[j]);
    do{
        if((ps[(i+1)%n]-ps[i]).det(ps[(j+1)%n]-ps[j])
            >=0) (++j)%=n;
        else (++i)%=n;
        ret=max(ret,ps[i].distTo(ps[j]));
    }while(i!=is||j!=js);
    return ret;
}
```

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

/*
    s,t 超级源、超级汇
    cur[] 当前弧优化
    时间复杂度 O(n^2*m)
*/

const int MAXN=10005;
const ll inf=0x3f3f3f3f3f3f3f3fll;
int n,m,s,t,tot,dis[MAXN],cur[MAXN];

struct edge
{
    int to,cap,rev;
    edge(){}
    edge(int to,int cap,int rev):to(to),cap(cap),rev(rev){}
};
vector<edge> E[MAXN];

inline void add_edge(int x,int y,int f)
{
    E[x].emplace_back(y,f,E[y].size());
    E[y].emplace_back(x,0,E[x].size()-1);
}

int bfs()
{
    for(int i=1;i<=n;i++) dis[i]=0x3f3f3f3f;
    dis[s]=0;
    queue<int> q;
    q.push(s);
    while(!q.empty())
    {
        int now=q.front();q.pop();
        for(int i=0;i<E[now].size();i++)
        {
            edge &e=E[now][i];
            if(dis[e.to]>dis[now]+1&&e.cap)
            {
                dis[e.to]=dis[now]+1;
                if(e.to==t) return 1;
                q.push(e.to);
            }
        }
    }
    return 0;
}

ll dfs(int now,ll flow)
{
    if(now==t) return flow;
    ll rest=flow,k;
    for(int i=cur[now];i<E[now].size();i++)
    {
        edge &e=E[now][i];
        if(e.cap&&dis[e.to]==dis[now]+1)
        {
            cur[now]=i;
            k=dfs(e.to,min(rest,(long long)e.cap));
            e.cap-=k;
            E[e.to][e.rev].cap+=k;
            rest-=k;
        }
    }
    return flow-rest;
}

ll dinic()
{
    ll ret=0,delta;
    while(bfs())
    {
        for(int i=1;i<=n;i++) cur[i]=0;
        while(delta=dfs(s,inf)) ret+=delta;
    }
```

```
    return ret;
}
```

3.7.2 最小费用最大流-Dij+Dinic

```
#include <bits/stdc++.h>
using namespace std;
typedef pair<int,int> pii;

/*
    第一遍跑的spfa,然后是加上势函数的dij,玄学
    h[] 势函数
    cur[] 当前弧优化
    msmf 最大流时的最小费用
    s,t 超级源、超级汇
    时间复杂度 O(n^2*m)
*/

const int MAXN=2005;
const int inf=0x3f3f3f3f;
int msmf,s,t,cur[MAXN],dis[MAXN],vis[MAXN],h[MAXN];
struct edge
{
    int to,val,cap,rev;
    edge(){}
    edge(int to,int cap,int val,int rev):to(to),cap(
        cap),val(val),rev(rev){}
};
vector<edge> E[MAXN];

inline void add_edge(int x,int y,int f,int cost)
{
    E[x].emplace_back(y,f,cost,E[y].size());
    E[y].emplace_back(x,0,-cost,E[x].size()-1);
}

int dij()
{
    fill(dis,dis+t+1,inf);
    priority_queue<pii,vector<pii>,greater<pii>> q;
    q.emplace(0,s);dis[s]=0;
    while(!q.empty())
    {
        pii p=q.top();q.pop();
        int now=p.second;
        if(dis[now]<p.first) continue;
        for(int i=0;i<E[now].size();i++)
        {
            edge &e=E[now][i];
            if(e.cap>0&&dis[e.to]>p.first+e.val+h[now]-
                h[e.to])
            {
                dis[e.to]=p.first+e.val+h[now]-h[e.to];
                q.emplace(dis[e.to],e.to);
            }
        }
    }
    return dis[t]!=inf;
}

int dfs(int now,int flow)
{
    if(now==t) return flow;
    int rest=flow,k;
    vis[now]=1;
    for(int i=cur[now];i<E[now].size();i++)
    {
        edge &e=E[now][i];
        if(e.cap&&dis[now]+e.val+h[now]-h[e.to]==dis[e
            .to]&&!vis[e.to])
        {
            cur[now]=i;
            k=dfs(e.to,min(e.cap,rest));
            e.cap-=k;
            E[e.to][e.rev].cap+=k;
            rest-=k;
            msmf+=k*e.val;
        }
    }
    vis[now]=0;
    return flow-rest;
}

int dinic()
{
    int ret=0,delta;
    while(dij())
    {
        for(int i=s;i<=t;i++) cur[i]=0;
        while(delta=dfs(s,inf)) ret+=delta;
        for(int i=s;i<=t;i++) h[i]+=(dis[i]==inf)?0:
            dis[i];
    }
    return ret;
}
```