

上海理工大学 ICPC 校队模板

HiedanoAkyuu、Oneman233、KR12138

2019 年 10 月 31 日

目录

1 字符串	1		
1.1 KMP	1		
1.2 EX-KMP	1		
1.3 Manacher	1		
1.4 串的最小表示	1		
1.5 后缀数组	1		
1.5.1 倍增 SA	1		
1.5.2 DC3	1		
1.6 回文自动机	1		
1.7 AC 自动机	1		
1.7.1 多模匹配	1		
1.7.2 自动机上 DP	2		
1.8 后缀自动机	2		
2 计算几何	2		
2.1 二维几何	2		
2.2 三维几何	4		
3 图论	4		
3.1 最短路	4		
3.1.1 Dijkstra	4		
3.1.2 SPFA	4		
3.1.3 Floyd	4		
3.1.4 负环	4		
3.1.5 差分约束	4		
3.2 最小生成树	4		
3.2.1 Prim	4		
3.2.2 Kruskal	4		
3.2.3 最小生成树计数	4		
3.2.4 次小生成树	4		
3.2.5 最小乘积生成树	4		
3.3 树的直径	4		
3.4 LCA	4		
3.4.1 Tarjan 离线	4		
3.4.2 倍增 LCA	4		
3.5 无向图与有向图联通性	4		
3.5.1 割点	4		
3.5.2 桥	4		
3.5.3 e-DCC	4		
3.5.4 v-DCC	4		
3.5.5 SCC	4		
3.5.6 2-SAT	4		
3.5.7 支配树	4		
3.6 二分图	4		
3.6.1 最大匹配-匈牙利	4		
3.6.2 带权匹配-KM	4		
3.7 网络流	4		
3.7.1 最大流-Dinic	4		
3.7.2 最小费用最大流-Dij+Dinic	4		
3.7.3 上下界流	4		
3.8 欧拉路	4		
3.9 Prufer 序列	4		
4 数据结构	4		
4.1 树状数组	4		
4.2 线段树	4		
4.2.1 多操作线段树	4		
4.2.2 吉司机线段树	4		
4.2.3 扫描线	4		
4.3 RMQ	4		
4.3.1 一维	4		
4.3.2 二维	4		
4.4 树链剖分	4		
4.4.1 点剖分	4		
4.4.2 边剖分	4		
4.5 平衡树	4		
4.5.1 Treap	4		
4.5.2 Splay	4		
4.6 动态树	4		
4.7 主席树	4		
4.8 树套树	4		
4.8.1 线段树套 Treap	4		
4.8.2 线段树套树状数组	4		
4.9 K-D Tree	4		
4.10 分治	4		
4.10.1 CDQ	4		
4.10.2 点分治	4		
4.10.3 dsu on tree	4		
4.10.4 整体二分	4		
4.11 分块	4		
4.11.1 普通分块	4		
4.11.2 莫队	4		
4.12 线性基	4		
4.13 珂朵莉树	4		
4.14 跳舞链	4		
5 动态规划	4		
5.1 SOS	4		
5.2 动态 DP	4		
5.3 插头 DP	4		
6 数学	4		
6.1 矩阵类	4		
6.2 质数筛	4		
6.2.1 埃筛	4		
6.2.2 线筛	4		
6.3 质数判定	4		
6.3.1 Miller Rabin	4		
6.4 质因数分解	4		
6.4.1 Pollard-Rho	4		
6.5 逆元	4		
6.5.1 EX-GCD 求逆元	4		
6.5.2 线性筛逆元	4		
6.5.3 阶乘逆元	4		
6.6 欧拉函数	4		
6.6.1 欧拉线筛	4		
6.6.2 求单个数的欧拉函数	4		
6.6.3 欧拉降幂	4		
6.6.4 一般积性函数求法	4		
6.7 EX-GCD	4		
6.8 CRT	4		
6.9 N 次剩余	4		
6.10 数论分块	4		
6.11 高斯消元	4		
6.11.1 普通消元	4		
6.11.2 异或方程组消元	4		
6.12 莫比乌斯反演	4		
6.12.1 莫比乌斯函数	4		
6.12.2 杜教筛	4		
6.12.3 洲阁筛	4		
6.12.4 min25 筛	4		
6.13 BSGS	4		
6.14 FFT	4		
6.15 FWT	4		
6.16 NTT	4		
6.17 数值计算	4		
6.17.1 辛普森	4		
6.17.2 自适应辛普森	4		
6.18 康拓展开	4		
6.19 卢卡斯定理	4		

7 其他5

7.1 快读快写5

7.2 约瑟夫环5

7.3 悬线法5

7.4 蔡勒公式5

7.5 三角公式5

7.6 海伦公式5

7.7 匹克定理5

7.8 组合计数5

7.8.1 计数原理5

7.8.2 卡特兰数5

7.8.3 Polya5

7.8.4 二项式反演公式5

7.8.5 斯特林反演公式5

7.8.6 组合数恒等式5

1 字符串

1.1 KMP

1.2 EX-KMP

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 /*
5  ex[i]: s1[i..l1-1]与s2的最大公共前缀长度
6  exnext[i]: s2[i..l2-1]与s2的最大公共前缀长度
7  get_exnext(s2) 求exnext[]
8  exkmp(s1,s2) 求ex[]
9 */
10
11 const int N=50005;
12 char s1[N],s2[N];
13 int ex[N],exnext[N];
14
15 void get_exnext(char s[N])
16 {
17     int n=strlen(s),p=1,j,i;
18     exnext[0]=n;
19     for(i=0;i<n-1&&s[i]==s[i+1];i++);
20     exnext[1]=i;
21     for(i=2;i<n;i++)
22         if(exnext[i-p]+i<p+exnext[p])
23             exnext[i]=exnext[i-p];
24         else
25         {
26             j=exnext[p]+p-i;
27             if(j<0) j=0;
28             while(i+j<n&&s[j]==s[i+j]) j++;
29             exnext[i]=j;
30             p=i;
31         }
32 }
33
34 void exkmp(char s1[N],char s2[N])
35 {
36     int l1=strlen(s1),l2=strlen(s2),p=0,i,j;
37     get_exnext(s2);
38     for(i=0;i<l1&&i<l2&&s1[i]==s2[i];i++);
39     ex[0]=i;
40     for(int i=1;i<l1;i++)
41     {
42         if(exnext[i-p]+i<p+ex[p])
43             ex[i]=exnext[i-p];
44         else
45         {
46             j=ex[p]+p-i;
47             if(j<0) j=0;
48             while(i+j<l1&&s1[i+j]==s2[j]) j++;
49             ex[i]=j;
50             p=i;
51         }
52     }
53 }

```

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

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 /*
5  trie静态开点+trie图优化
6 */
7
8 int sz,hd=1,nxt[1000005][26],fail[1000005],id
   [1000005],n;
9 char s[1000005];
10
11 void trie_clean()
12 {
13     sz=1;
14     memset(nxt,0,sizeof(nxt));
15     memset(fail,0,sizeof(fail));
16     memset(id,0,sizeof(id));
17 }
18
19 void trie_insert(int head,char s[],int len,int idx)
20 {
21     int p=head;
22     for(int i=0;i<len;i++)
23     {
24         int c=s[i]-'a';
25         if(!nxt[p][c]) nxt[p][c]=++sz;
26         p=nxt[p][c];
27     }
28     id[p]+=idx;
29 }
30
31 void acatm_build(int head)
32 {
33     int p,tp;
34     queue<int> q;
35     q.push(head);
36     fail[head]=0;
37     while(!q.empty())
38     {
39         p=q.front();
40         q.pop();
41         for(int i=0;i<26;i++)
42             if(nxt[p][i])
43             {
44                 fail[nxt[p][i]]=p==head?head:nxt[fail[p]
   ][i];
45                 q.push(nxt[p][i]);
46             }
47         else
48             nxt[p][i]=p==head?head:nxt[fail[p]][i];
49     }
50 }

```

```

50 }
51
52 int acatm_match(int head,char s[],int len)
53 {
54     int p=head,ret=0;
55     for(int i=0;i<len;i++)
56     {
57         int c=(int)s[i]-'a';
58         p=nxt[p][c];
59         for(int tp=p;tp;tp=fail[tp])
60             if(id[tp]) ret++;
61     }
62     return ret;
63 }

```

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

```

1 #include<bits/stdc++.h>
2 using namespace std;
3
4 #define db double
5 const db EPS=1e-9;
6 inline int sign(db a){return a<-EPS?-1:a>EPS;}
7 inline int cmp(db a,db b){return sign(a-b);}
8 struct P
9 {
10     db x,y;
11     P(){}
12     P(db x,db y):x(x),y(y){}
13     P operator+(P p){return {x+p.x,y+p.y};}
14     P operator-(P p){return {x-p.x,y-p.y};}
15     P operator*(db d){return {x*d,y*d};}
16     P operator/(db d){return {x/d,y/d};}
17     bool operator<(P p) const
18     {
19         int c=cmp(x,p.x);
20         if(c) return c==1;
21         return cmp(y,p.y)==1;
22     }
23     bool operator==(P o) const
24     {
25         return cmp(x,o.x)==0&&cmp(y,o.y)==0;
26     }
27     db distTo(P p){return (*this-p).abs();}
28     db alpha(){return atan2(y,x);}
29     void read(){scanf("%lf%lf",&x,&y);}
30     void write(){printf("%.10f,%.10f\n",x,y);}
31     db abs(){return sqrt(abs2());}
32     db abs2(){return x*x+y*y;}
33     P rot90(){return P(-y,x);}
34     P unit(){return *this/abs();}
35     int quad() const {return sign(y)==1||((sign(y)==0&&
36         sign(x)>=0);}
37     db dot(P p){return x*p.x+y*p.y;}
38     db det(P p){return x*p.y-y*p.x;}
39     P rot(db an){return {x*cos(an)-y*sin(an),x*sin(an)
40         +y*cos(an)}};

```

```

40 //For segment
41 #define cross(p1,p2,p3) ((p2.x-p1.x)*(p3.y-p1.y)-(p3.
42     x-p1.x)*(p2.y-p1.y))
43 #define crossOp(p1,p2,p3) sign(cross(p1,p2,p3))
44
45 bool chkLL(P p1,P p2,P q1,P q2) //0:parallel
46 {
47     db a1=cross(q1,q2,p1),a2=-cross(q1,q2,p2);
48     return sign(a1+a2)!=0;
49 }
50
51 P isLL(P p1,P p2,P q1,P q2) //crossover point if
52     chkLL()
53 {
54     db a1=cross(q1,q2,p1),a2=-cross(q1,q2,p2);
55     return (p1*a2+p2*a1)/(a1+a2);
56 }
57
58 bool intersect(db l1,db r1,db l2,db r2)
59 {
60     if(l1>r1) swap(l1,r1);if(l2>r2) swap(l2,r2);
61     return !(cmp(r1,l2)==-1||cmp(r2,l1)==-1);
62 }
63
64 bool isSS(P p1,P p2,P q1,P q2)
65 {
66     return intersect(p1.x,p2.x,q1.x,q2.x)&&intersect(
67         p1.y,p2.y,q1.y,q2.y)&&
68         crossOp(p1,p2,q1)*crossOp(p1,p2,q2)<=0&&crossOp(q1
69             ,q2,p1)*crossOp(q1,q2,p2)<=0;
70 }
71
72 bool isSS_strict(P p1,P p2,P q1,P q2)
73 {
74     return crossOp(p1,p2,q1)*crossOp(p1,p2,q2)<0
75         &&crossOp(q1,q2,p1)*crossOp(q1,q2,p2)<0;
76 }
77
78 bool isMiddle(db a,db m,db b)
79 {
80     return sign(a-m)==0||sign(b-m)==0||((a<m!=b<m);
81 }
82
83 bool isMiddle(P a,P m,P b)
84 {
85     return isMiddle(a.x,m.x,b.x)&&isMiddle(a.y,m.y,b.y
86         );
87 }
88
89 bool onSeg(P p1,P p2,P q)
90 {
91     return crossOp(p1,p2,q)==0&&isMiddle(p1,q,p2);
92 }
93
94 bool onSeg_strict(P p1,P p2,P q)
95 {
96     return crossOp(p1,p2,q)==0&&sign((q-p1).dot(p1-p2)
97         )*sign((q-p2).dot(p1-p2))<0;
98 }
99
100 P proj(P p1,P p2,P q)
101 {
102     P dir=p2-p1;
103     return p1+dir*(dir.dot(q-p1)/dir.abs2());

```

```

99 }
100
101 P reflect(P p1,P p2,P q)
102 {
103     return proj(p1,p2,q)*2-q;
104 }
105
106 db nearest(P p1,P p2,P q)
107 {
108     P h=proj(p1,p2,q);
109     if(isMiddle(p1,h,p2))
110         return q.distTo(h);
111     return min(p1.distTo(q),p2.distTo(q));
112 }
113
114 db disSS(P p1,P p2,P q1,P q2) //dist of 2 segments
115 {
116     if(isSS(p1,p2,q1,q2)) return 0;
117     return min(min(nearest(p1,p2,q1),nearest(p1,p2,q2)),
118                min(nearest(q1,q2,p1),nearest(q1,q2,p2)));
119 }
120
121 db rad(P p1,P p2)
122 {
123     return atan2l(p1.det(p2),p1.dot(p2));
124 }
125
126 db area(vector<P> ps)
127 {
128     db ret=0;
129     for(int i=0;i<ps.size();i++)
130         ret+=ps[i].det(ps[(i+1)%ps.size()]);
131     return ret/2;
132 }
133
134 int contain(vector<P> ps,P p) //2:inside,1:on_seg,0:
135     outside
136 {
137     int n=ps.size(),ret=0;
138     for(int i=0;i<n;i++)
139     {
140         P u=ps[i],v=ps[(i+1)%n];
141         if(onSeg(u,v,p)) return 1;
142         if(cmp(u.y,v.y)<=0) swap(u,v);
143         if(cmp(p.y,u.y)>0||cmp(p.y,v.y)<=0) continue;
144         ret^=crossOp(p,u,v)>0;
145     }
146     return ret*2;
147 }
148
149 vector<P> convexHull(vector<P> ps)
150 {
151     int n=ps.size();if(n<=1) return ps;
152     sort(ps.begin(),ps.end());
153     vector<P> qs(n*2);int k=0;
154     for(int i=0;i<n;qs[k++]=ps[i++])
155         while(k>1&&crossOp(qs[k-2],qs[k-1],ps[i])<=0)
156             --k;
157     for(int i=n-2,t=k;i>=0;qs[k++]=ps[i--])
158         while(k>t&&crossOp(qs[k-2],qs[k-1],ps[i])<=0)
159             --k;
160     qs.resize(k-1);
161     return qs;
162 }

```

```

160 db convexDiameter(vector<P> ps)
161 {
162     int n=ps.size();if(n<=1) return 0;
163     int is=0,js=0;
164     for(int k=1;k<n;k++) is=ps[k]<ps[is]?k:is,js=ps[js]
165         <ps[k]?js:k;
166     int i=is,j=js;
167     db ret=ps[i].distTo(ps[j]);
168     do{
169         if((ps[(i+1)%n]-ps[i]).det(ps[(j+1)%n]-ps[j])
170             >=0) (++j)%=n;
171         else (++i)%=n;
172         ret=max(ret,ps[i].distTo(ps[j]));
173     }while(i!=is||j!=js);
174     return ret;
175 }

```

2.2 三维几何

3 图论

3.1 最短路

3.1.1 Dijkstra

3.1.2 SPFA

3.1.3 Floyd

3.1.4 负环

3.1.5 差分约束

3.2 最小生成树

3.2.1 Prim

3.2.2 Kruskal

3.2.3 最小生成树计数

3.2.4 次小生成树

3.2.5 最小乘积生成树

3.3 树的直径

3.4 LCA

3.4.1 Tarjan 离线

3.4.2 倍增 LCA

3.5 无向图与有向图联通性

3.5.1 割点

3.5.2 桥

3.5.3 e-DCC

3.5.4 v-DCC

3.5.5 SCC

3.5.6 2-SAT

3.5.7 支配树

3.6 二分图

3.6.1 最大匹配-匈牙利

3.6.2 带权匹配-KM

3.7 网络流

3.7.1 最大流-Dinic

3.7.2 最小费用最大流-Dij+Dinic

3.7.3 上下界流

3.8 欧拉路

3.9 Prufer 序列

4 数据结构

4.1 树状数组

4.2 线段树

4.2.1 多操作线段树

$$\binom{N}{m} \equiv \binom{N \bmod p}{m \bmod p} \cdot \binom{N/p}{m/p} \pmod{p}$$

可理解为将 N 和 m 表示为 p 进制数，对每一位的 N_i 和 m_i 分别求组合数，再累乘

```

1 //洛谷P3807
2 #include <bits/stdc++.h>
3 using namespace std;
4 typedef long long ll;
5 const int MN = 5e6 + 5;
6
7 inline ll qpow(ll a,ll b,int P){ //a^b%P
8     ll ans=1;
9     for(;b>>=1;a=a*a%P)
10         if(b&1) ans=ans*a%P;
11 return ans;}
12
13 ll fct[MN],fi[MN]; //阶乘及其逆元
14 inline void init(int k,int P){ //打表模P的[1,k]阶乘及其逆元
15     fct[0]=1;
16     for(int i=1; i<=k; ++i) fct[i]=fct[i-1]*i%P;
17     if(k<P){
18         fi[k]=qpow(fct[k],P-2,P);
19         for(int i=k; i>=1; --i) fi[i-1]=fi[i]*i%P;
20     }else{ //k阶乘为0, 会把所有逆元都变成0, 应从P-1开始
21         fi[P-1]=qpow(fct[P-1],P-2,P);
22         for(int i=P-1; i>=1; --i) fi[i-1]=fi[i]*i%P;
23     }
24 }
25
26 inline int C(int N,int m,int P){ //C_N^m % P
27     if(m>N) return 0;
28     return fct[N]*fi[m]%P*fi[N-m]%P;
29 }
30
31 //ll lucas(int N,int m,int P){ //递归求C_N^m % P
32 // if(!m) return 1;
33 // return C(N%P,m%P,P)*lucas(N/P,m/P,P)%P;
34 //}
35
36 int lucas(int N,int m,int P){ //循环求C_N^m % P
37     ll rt=1;
38     while(N&& m)
39         (rt*=C(N%P,m%P,P))%=P,
40         N/=P, m/=P;
41     return rt;
42 }
43
44 void solve(){
45     int n,m,p; scanf("%d%d%d",&n,&m,&p);
46     init(n+m,p);
47     printf("%lld\n",lucas(n+m,m,p));
48 }
49
50 int main(int argc, char** argv){
51     int _; scanf("%d",&_); while(--)
52         solve();
53     return 0;
54 }

```

7 其他

7.1 快读快写

7.2 约瑟夫环

7.3 悬线法

7.4 蔡勒公式

7.5 三角公式

7.6 海伦公式

7.7 匹克定理

7.8 组合计数

7.8.1 计数原理

7.8.2 卡特兰数

7.8.3 Polya

7.8.4 二项式反演公式

7.8.5 斯特林反演公式

7.8.6 组合数恒等式