

神秘模板库

Toy ASM Truck

Huazhong University of Science and Technology

November 10, 2021

Contents

一切的开始	3
随机数生成	3
哈希相关	3
Ordered set	3
没什么用的卡常	3
数据结构	4
ST 表	4
树状数组区间修改区间查询	4
树剖 & LCA	4
无旋 Treap	5
替罪羊树	6
Splay	8
类欧几里得	10
Pollard-Rho	11
ex-gcd	12
crt	12
ex-crt	12
Meissel-Lehmer	12
Cipolla 二次剩余	13
BSGS	14
exBSGS	14
exLucas	15
min_25 筛	15
超现实数	17
图论	20
LCA	20
同余最短路	20
计算几何	21
二维几何：点与向量	21
完整的板板 by zcs	21
字符串	25
kmp	25
exkmp	25
manacher	25
AC 自动机	26
后缀自动机	26
后缀自动机求第 k 大子串	26
最小表示法	27
后缀数组	28
Lyndon 分解	28
Lyndon 分解求所有前缀最小字典序的后缀	29
多项式	30
NTT 模数	30
FFT	30
NTT	30
完整的板板 by hls	31
容斥与反演	34
莫比乌斯反演	34
快速莫比乌斯变换（反演）与子集卷积	36
莫比乌斯变换（反演）	36
子集卷积	37

二项式反演	38
内容	38
证明	38
应用举例	38
另一形式	39
斯特林反演	40
第一类斯特林数	40
第二类斯特林数	40
反演公式	40
最值反演 (min-max 容斥)	41
公式	41
证明	41
拉格朗日插值法	41
简介	41
求解	42
自然数的幂的前缀和	42
问题提出	42
问题解决	42
代码实现	43

一切的开始

随机数生成

```
1 mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());//init
2 //神秘的高级版本
3 seed_seq seq{
4     (uint64_t)
5     ↪ chrono::duration_cast<chrono::nanoseconds>(chrono::high_resolution_clock::now().time_since_epoch()).count(),
6     (uint64_t) __builtin_ia32_rdtsc(),
7     (uint64_t) (uintptr_t) make_unique<char>().get()
8 };
9 mt19937 rng(seq);
10 //64bit
11 mt19937_64 rng(seq);
12 //usage
13 shuffle(permutation.begin(), permutation.end(), rng);//using shuffle
14 uniform_int_distribution<int>(0, i)(rng);//get random number [0,i]
```

哈希相关

● 哈希表

```
1 #include <ext/pb_ds/assoc_container.hpp>
2 using namespace __gnu_pbds;
3 gp_hash_table<int, int> table;
```

● 哈希函数

```
1 struct custom_hash {
2     static uint64_t splitmix64(uint64_t x) {
3         // http://xorshift.di.unimi.it/splitmix64.c
4         x += 0x9e3779b97f4a7c15;
5         x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
6         x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
7         return x ^ (x >> 31);
8     }
9     size_t operator()(uint64_t x) const {
10         static const uint64_t FIXED_RANDOM = chrono::steady_clock::now().time_since_epoch().count();
11         return splitmix64(x + FIXED_RANDOM);
12     }
13 };
14 //inverse of splitmix
15 uint64_t splittable64_r(uint64_t x){
16     x ^= x >> 31 ^ x >> 62;
17     x *= 0x319642b2d24d8ec3U;
18     x ^= x >> 27 ^ x >> 54;
19     x *= 0x96de1b173f119089U;
20     x ^= x >> 30 ^ x >> 60;
21     return x;
22 }
23 //better hash map
24 unordered_map<long long, int, custom_hash> safe_map;
25 gp_hash_table<long long, int, custom_hash> safe_hash_table;
```

Ordered set

```
1 #include <ext/pb_ds/assoc_container.hpp> // Common file
2 #include <ext/pb_ds/tree_policy.hpp> // Including tree_order_statistics_node_update
3 typedef __gnu_pbds::tree<int, null_type, less<int>, rb_tree_tag, tree_order_statistics_node_update> ordered_set;
4 cout<<*X.find_by_order(1)<<endl;
5 cout<<X.order_of_key(-5)<<endl;
```

没什么用的卡常

```
1 #pragma GCC optimize("Ofast,no-stack-protector,unroll-loops,fast-math")
2 #pragma GCC target("sse,sse2,sse3,ssse3,sse4.1,sse4.2,avx,avx2,popcnt,tune=native")
3 #include <immintrin.h>//封装好的指令集函数
4 #include <emmintrin.h>
5 __m256i _mm256_set_epi32 (int e7, int e6, int e5, int e4, int e3, int e2, int e1, int e0);//将 8 个 int 扔进 256 里面
```

```

6  __m256i _mm256_set_epi64x (__int64 e3, __int64 e2, __int64 e1, __int64 e0); //int64 的版本
7  __m256i _mm256_set1_epi32 (int a); //放 8 个 a
8  __m256i _mm256_add_epi32 (__m256i a, __m256i b); //加法
9  __m256i _mm256_cmpeq_epi32 (__m256i a, __m256i b); //相等, 对应位置为 0xffffffff
10 __m256i _mm256_cmpgt_epi32 (__m256i a, __m256i b); //大于
11 __m256i _mm256_and_si256 (__m256i a, __m256i b); //按位与

```

数据结构

ST 表

- 二维

```

1  int f[maxn][maxn][10][10];
2  inline int highbit(int x) { return 31 - __builtin_clz(x); }
3  inline int calc(int x, int y, int xx, int yy, int p, int q) {
4      return max(
5          max(f[x][y][p][q], f[xx - (1 << p) + 1][yy - (1 << q) + 1][p][q]),
6          max(f[xx - (1 << p) + 1][y][p][q], f[x][yy - (1 << q) + 1][p][q])
7      );
8  }
9  void init() {
10     FOR (x, 0, highbit(n) + 1)
11     FOR (y, 0, highbit(m) + 1)
12     FOR (i, 0, n - (1 << x) + 1)
13     FOR (j, 0, m - (1 << y) + 1) {
14         if (!x && !y) { f[i][j][x][y] = a[i][j]; continue; }
15         f[i][j][x][y] = calc(
16             i, j,
17             i + (1 << x) - 1, j + (1 << y) - 1,
18             max(x - 1, 0), max(y - 1, 0)
19         );
20     }
21 }
22 inline int get_max(int x, int y, int xx, int yy) {
23     return calc(x, y, xx, yy, highbit(xx - x + 1), highbit(yy - y + 1));
24 }

```

树状数组区间修改区间查询

```

1  struct seg{
2      int n;
3      vector<ll> c1,c2;
4      seg(int n):n(n),c1(n+1),c2(n+1){}
5      void add(int x,ll y){for(int i=x;i<=n;i+=i&(-i)) c1[i]+=y,c2[i]+=x*y;}
6      void add(int l,int r,ll x){add(l,x);add(r+1,-x);}
7      ll sum(int x){
8          long long ans(0);
9          for(int i=x;i>=1;i-=i&(-i)) ans+=(x+1)*c1[i]-c2[i];
10         return ans;
11     }
12     ll query(int l,int r){return sum(r)-sum(l-1);}
13 };

```

树剖 & LCA

```

1  vector<int> fa(n+1),dfn(n+1),top(n+1),sz(n+1),son(n+1),ans(n+1),dep(n+1);
2  function<void(int,int)> dfs1=[&](int x,int f){
3      fa[x]=f;sz[x]=1;
4      for(auto v:G[x]){
5          if(v==f) continue;
6          dep[v]=dep[x]+1;
7          dfs1(v,x);
8          sz[x]+=sz[v];
9          if(sz[v]>sz[son[x]]) son[x]=v;
10     }
11 };
12 function<void(int,int)> dfs2=[&](int x,int tp){

```

```

13     top[x]=tp;
14     if(son[x]) dfs2(son[x],tp);
15     for(auto v:G[x]){
16         if(v==son[x] || v==fa[x]) continue;
17         dfs2(v,v);
18     }
19 };
20 function<int(int x,int y)> lca=[&](int u,int v){
21     for(;top[u]^top[v];u=fa[top[u]])
22         if(dep[top[u]]<dep[top[v]]) swap(u,v);
23     return dep[u]<dep[v] ? u:v;
24 };

```

无旋 Treap

```

1  struct Treap{
2      Treap *l,*r;
3      int fix,key,size;
4      Treap(int _key){fix=rand();key=_key;size=1;l=r=NULL;}
5      inline void update(){
6          size=(l? l->size:0)+(r? r->size:0)+1;
7      }
8  }*root;
9  typedef pair<Treap*,Treap*> Droot;
10 inline int size(Treap *x){return x?x->size:0;}
11 inline int key(Treap *x){return x?x->key:inf;}
12 Treap* merge(Treap *a,Treap *b){
13     if(!a) return b;
14     if(!b) return a;
15     if(a->fix<b->fix){
16         a->r=merge(a->r,b);
17         a->update();
18         return a;
19     }else{
20         b->l=merge(a,b->l);
21         b->update();
22         return b;
23     }
24 }
25 Droot split(Treap *x,int k){
26     if(!x) return Droot(NULL,NULL);
27     Droot y;
28     if(size(x->l)>=k){
29         y=split(x->l,k);
30         x->l=y.sc;
31         x->update();
32         y.sc=x;
33     }else{
34         y=split(x->r,k-size(x->l)-1);
35         x->r=y.fi;
36         x->update();
37         y.fi=x;
38     }
39     return y;
40 }
41 Treap* build(int *a,int n){
42     static Treap *stack[N],*x,*last;
43     int p=0;
44     rep(i,1,n){
45         x=new Treap(a[i]);
46         while(p>0&&x->fix<stack[p]->fix){
47             stack[p]->update();
48             last=stack[p];
49             stack[p--]=NULL;
50         }
51         if(p) stack[p]->r=x;
52         x->l=last;
53         stack[++p]=x;
54     }
55     while(p) stack[p--]->update();
56     return stack[1];

```

```

57 }
58 int findkth(int k){
59     Droot x=split(root,k-1);
60     Droot y=split(x.sc,1);
61     Treap *ans=y.fi;
62     root=merge(merge(x.fi,ans),y.sc);
63     return key(ans);
64 }
65 int getkth(Treap *x,int v){
66     if(!x) return 0;
67     return v<=x->key ? getkth(x->l,v):getkth(x->r,v)+size(x->l)+1;
68 }
69 void insert(int v){
70     int k=getkth(root,v);
71     Droot x=split(root,k);
72     Treap *tmp=new Treap(v);
73     root=merge(merge(x.fi,tmp),x.sc);
74 }
75 void del(int k){
76     Droot x=split(root,k);
77     Droot y=split(x.sc,1);
78     root=merge(x.fi,y.sc);
79 }
80 int n;
81 int main(){
82     n=read();
83     while(n --> 0){
84         int opt=read();
85         if(opt==1){
86             insert(read());//insert x
87         }else if(opt==2){
88             del(getkth(root,read()));//delete x
89         }else if(opt==3){
90             printf("%d\n",getkth(root,read()+1));//query rank
91         }else if(opt==4){
92             printf("%d\n",findkth(read()));//query kth
93         }else if(opt==5){
94             printf("%d\n",findkth(getkth(root,read())));//pre
95         }else if(opt==6){
96             printf("%d\n",findkth(getkth(root,read()+1)+1));//suc
97         }
98     }
99     return 0;
100 }

```

替罪羊树

```

1  const double al=0.73;
2  int fa[N],c[N][2],size[N],cnt[N],val[N],rt,sz;
3  int tmp[N],ind;
4  inline int get(int x){return c[fa[x]][1]==x;}
5  inline void rec(int x){
6      if(c[x][0]) rec(c[x][0]);
7      tmp[++ind]=x;
8      if(c[x][1]) rec(c[x][1]);
9  }
10 inline int build(int l,int r){
11     if(l>r) return 0;
12     int mid=(l+r)>>1,cur=tmp[mid];
13     c[cur][0]=build(l,mid-1);fa[c[cur][0]]=cur;
14     c[cur][1]=build(mid+1,r);fa[c[cur][1]]=cur;
15     size[cur]=size[c[cur][0]]+size[c[cur][1]]+cnt[cur];
16     return cur;
17 }
18 inline void rebuild(int x){
19     ind=0;int f=fa[x],d=get(x);
20     rec(x);int cur=build(1,ind);
21     fa[cur]=f;c[f][d]=cur;
22     if(x==rt) rt=cur;
23 }
24 inline int check(int x){

```

```

25     double now=size[x]*al;
26     return now>=size[c[x][0]]&&now>=size[c[x][1]];
27 }
28 inline void insert(int x){
29     int cur=rt;
30     if(!cur){cur=rt++;sz[cnt[cur]=size[cur]=1;val[cur]=x;return;}
31     while(cur){
32         ++size[cur];
33         if(val[cur]==x){cnt[cur]++;break;}
34         int &v=c[cur][val[cur]<x];
35         if(!v){v=++sz;fa[v]=cur;val[v]=x;cnt[v]=size[v]=1;cur=v;break;}
36         cur=v;
37     }
38     int id=0;
39     for(int i=cur;i=fa[i]) if(!check(i)) id=i;
40     if(id) rebuild(id);
41 }
42 inline int find(int x){
43     int cur=rt;
44     while(cur){
45         if(val[cur]==x) return cur;
46         cur=c[cur][x>val[cur]];
47     }
48     return -1;
49 }
50 inline void del(int x){
51     int cur=find(x);int k;assert(cur!=-1);
52     if(cnt[cur]>1){--size[cur];--cnt[cur];for(int i=fa[cur];i=fa[i]) --size[i];return;}
53     int pp=0;
54     if(c[cur][0]&&c[cur][1]){
55         k=c[cur][0];
56         while(c[k][1]) k=c[k][1];//pre
57         val[cur]=val[k];cnt[cur]=cnt[k];pp=cur;cur=k;
58     }
59     k=c[cur][0] ? c[cur][0]:c[cur][1];
60     c[fa[cur]][get(cur)]=k;fa[k]=fa[cur];
61     if(pp){for(int i=fa[cur];i!=pp;i=fa[i]) size[i]--cnt[cur];for(int i=pp;i=fa[i]) --size[i];}
62     else for(int i=fa[cur];i=fa[i]) --size[i];
63     if(cur==rt) rt=k;
64 }
65 inline int findrk(int x){
66     int cur=rt,ret=0;
67     while(cur){
68         int lsize=c[cur][0] ? size[c[cur][0]]:0;
69         if(val[cur]==x){ret+=lsize+1;break;}
70         if(val[cur]<x){ret+=lsize+cnt[cur];cur=c[cur][1];}
71         else cur=c[cur][0];
72     }
73     return ret;
74 }
75 inline int findkth(int k){
76     int cur=rt,ret=0;
77     while(cur){
78         int lsize=c[cur][0] ? size[c[cur][0]]:0;
79         if(k<=lsize) cur=c[cur][0];
80         else if(k<=lsize+cnt[cur]){ret=cur;break;}
81         else k-=lsize+cnt[cur],cur=c[cur][1];
82     }
83     return ret;
84 }
85 inline int getpre(int x){
86     int cur=rt,ret=0;
87     while(cur){
88         if(val[cur]>=x) cur=c[cur][0];
89         else{ret=cur;cur=c[cur][1];}
90     }
91     return ret;
92 }
93 inline int getsuc(int x){
94     int cur=rt,ret=0;
95     while(cur){

```



```

96         if(val[cur]<=x) cur=c[cur][1];
97         else{ret=cur;cur=c[cur][0];}
98     }
99     return ret;
100 }
101 int main(){
102     int n=read();
103     rep(i,1,n){
104         int op=read(),x=read();
105         if(op==1) insert(x);
106         else if(op==2) del(x);
107         else if(op==3) printf("%d\n",findrk(x));
108         else if(op==4) printf("%d\n",val[findkth(x)]);
109         else if(op==5) printf("%d\n",val[getpre(x)]);
110         else printf("%d\n",val[getsuc(x)]);
111     }
112     return 0;
113 }

```

Splay

```

1  struct Splay{
2      int c[N][2],fa[N],size[N],cnt[N],val[N],rt,sz;
3      inline int get(int x){return x==c[fa[x]][1];}
4      inline void upd(int x){
5          size[x]=cnt[x];
6          size[x]+=size[c[x][0]];size[x]+=size[c[x][1]];
7      }
8      inline void rotate(int x,int &k){
9          int f=fa[x],p=fa[f],d=get(x);
10         if(f==k) k=x;else c[p][get(f)]=x;
11         fa[x]=p;c[f][d]=c[x][d^1];if(c[x][d^1]) fa[c[x][d^1]]=f;
12         fa[f]=x;c[x][d^1]=f;upd(f);upd(x);
13     }
14     inline void splay(int x,int &k){
15         while(x^k){
16             int f=fa[x];
17             if(f!=k) rotate(get(x)==get(f) ? f:x,k);
18             rotate(x,k);
19         }
20     }
21     inline void insert(int cur,int x){
22         if(!cur){cur=rt=++sz;val[cur]=x;size[cur]=cnt[cur]=1;return;}
23         while(1){
24             ++size[cur];
25             if(val[cur]==x){++cnt[cur];break;}
26             int &v=c[cur][x>val[cur]];
27             if(!v){v=++sz;val[v]=x;size[v]=cnt[v]=1;fa[v]=cur;cur=v;break;}
28             cur=v;
29         }
30         splay(cur,rt);
31     }
32     inline void find(int cur,int x){
33         while(cur){
34             if(val[cur]==x){splay(cur,rt);return;}
35             cur=c[cur][x>val[cur]];
36         }
37     }
38     inline int findrk(int cur,int x){
39         int ret=0;
40         while(x){
41             if(val[cur]==x){ret+=size[c[cur][0]]+1;splay(cur,rt);break;}
42             if(val[cur]>x) cur=c[cur][0];
43             else ret+=size[c[cur][0]]+cnt[cur],cur=c[cur][1];
44         }
45         return ret;
46     }
47     inline int findkth(int cur,int k){
48         int ret=0;
49         while(cur){
50             int lsize=size[c[cur][0]];

```

```

51         if(k<=lsize) cur=c[cur][0];
52         else if(k<=lsize+cnt[cur]){ret=val[cur];splay(cur,rt);break;}
53         else k-=lsize+cnt[cur],cur=c[cur][1];
54     }
55     return ret;
56 }
57 inline int getpre(int cur,int x){
58     int ret=0;
59     while(cur){
60         if(val[cur]>=x) cur=c[cur][0];
61         else cur=c[ret=cur][1];
62     }
63     if(ret) return splay(ret,rt),val[ret];
64     else return -1;
65 }
66 inline int getsuc(int cur,int x){
67     int ret=0;
68     while(cur){
69         if(val[cur]<=x) cur=c[cur][1];
70         else cur=c[ret=cur][0];
71     }
72     if(ret) return splay(ret,rt),val[ret];
73     else return -1;
74 }
75 inline int pre(){
76     int now=c[rt][0];
77     while(c[now][1]) now=c[now][1];splay(now,rt);
78     return now;
79 }
80 inline int suc(){
81     int now=c[rt][1];
82     while(c[now][0]) now=c[now][0];splay(now,rt);
83     return now;
84 }
85 inline void del(int x){
86     find(rt,x);int cur=rt,k=c[cur][0]*c[cur][1];
87     if(cnt[cur]>1){--cnt[cur];--size[cur];return;}
88     if(!k){rt=c[cur][0]+c[cur][1];return;}
89     else{
90         int p=pre();splay(p,rt);
91         c[p][1]=c[cur][1];
92         fa[c[cur][1]]=c[cur][1] ? p:0;
93         upd(p);
94     }
95 }
96 }T;
97 int main(){
98     int n=read();
99     rep(i,1,n){
100         int op=read(),x=read();
101         if(op==1){
102             T.insert(T.rt,x);
103         }else if(op==2){
104             T.del(x);
105         }else if(op==3){
106             printf("%d\n",T.findrk(T.rt,x));
107         }else if(op==4){
108             printf("%d\n",T.findkth(T.rt,x));
109         }else if(op==5){
110             printf("%d\n",T.getpre(T.rt,x));
111         }else{
112             printf("%d\n",T.getsuc(T.rt,x));
113         }
114     }
115     return 0;
116 }
117
118 # 数学
119
120 ## 竞赛图相关结论
121

```

122 强连通是竞赛图有哈密顿回路的充分必要条件。

123

124 n 个点的竞赛图共有 $(n-1)! \cdot 2^{C(n,2)-n}$ 条哈密顿回路。

125

126 考虑枚举最小拓扑排序的强连通分量, n 个点的强连通竞赛图的个数为 $f[n] = 2^{C(n,2)} - \sum_{i=1}^{n-1} \{f[i] \cdot C(n,i) \cdot 2^{C(n-i,2)}\}$, 可以用分治FFT 或者多项式求逆来解决。

127

128 ## 模整数类

129 除法为整除, 请乘逆元。

130 ```cpp

131 template<int P>

132 struct moint {

133 int x;

134 moint():x(0){}

135 moint(int n){x=n<0?n%P+P:n%P;}

136 moint(ll n) {x=n<0?n%P+P:n%P;}

137 int get()const{return (int)x;}

138 moint &operator+=(moint b){x+=b.x;if(x>=P)x-=P;return *this;}

139 moint &operator-=(moint b){x-=b.x;if(x<0)x+=P;return *this;}

140 moint &operator*=(moint b){x=1ll*x*b.x%P;return *this;}

141 moint &operator/=(moint b){x=x/b.x;return *this;}

142 moint &operator%=(moint b){x=x%b.x;return *this;}

143 moint operator+(moint b)const{return moint(*this)+=b;}

144 moint operator-(moint b)const{return moint(*this)-=b;}

145 moint operator*(moint b)const{return moint(*this)*=b;}

146 moint operator/(moint b)const{return moint(*this)/=b;}

147 moint operator%(moint b)const{return moint(*this)%=b;}

148 moint operator+(int b)const{return moint(*this)+=moint(b);}

149 moint operator-(int b)const{return moint(*this)-=moint(b);}

150 moint operator*(int b)const{return moint(*this)*=moint(b);}

151 moint operator/(int b)const{return moint(*this)/=moint(b);}

152 moint operator%(int b)const{return moint(*this)%=moint(b);}

153 bool operator==(moint b)const{return x==b.x;}

154 bool operator>=(moint b)const{return x>=b.x;}

155 bool operator!=(moint b)const{return x!=b.x;}

156 };

157 typedef moint<998244353> mint;

类欧几里得

- $m = \lfloor \frac{a+b}{c} \rfloor$.
- $f(a, b, c, n) = \sum_{i=0}^n \lfloor \frac{ai+b}{c} \rfloor$: 当 $a \geq c$ 或 $b \geq c$ 时, $f(a, b, c, n) = (\frac{a}{c})n(n+1)/2 + (\frac{b}{c})(n+1) + f(a \bmod c, b \bmod c, c, n)$; 否则 $f(a, b, c, n) = nm - f(c, c-b-1, a, m-1)$ 。
- $g(a, b, c, n) = \sum_{i=0}^n i \lfloor \frac{ai+b}{c} \rfloor$: 当 $a \geq c$ 或 $b \geq c$ 时, $g(a, b, c, n) = (\frac{a}{c})n(n+1)(2n+1)/6 + (\frac{b}{c})n(n+1)/2 + g(a \bmod c, b \bmod c, c, n)$; 否则 $g(a, b, c, n) = \frac{1}{2}(n(n+1)m - f(c, c-b-1, a, m-1) - h(c, c-b-1, a, m-1))$ 。
- $h(a, b, c, n) = \sum_{i=0}^n \lfloor \frac{ai+b}{c} \rfloor^2$: 当 $a \geq c$ 或 $b \geq c$ 时, $h(a, b, c, n) = (\frac{a}{c})^2 n(n+1)(2n+1)/6 + (\frac{b}{c})^2 (n+1) + (\frac{a}{c})(\frac{b}{c})n(n+1) + h(a \bmod c, b \bmod c, c, n) + 2(\frac{a}{c})g(a \bmod c, b \bmod c, c, n) + 2(\frac{b}{c})f(a \bmod c, b \bmod c, c, n)$; 否则 $h(a, b, c, n) = nm(m+1) - 2g(c, c-b-1, a, m-1) - 2f(c, c-b-1, a, m-1) - f(a, b, c, n)$ 。

```

1  struct ans{mint f,g,h;};
2  static ans calc(int a,int b,int c,int n){
3      ans ret;
4      if(!a){
5          ret.f=mint(b/c)*(n+1);
6          ret.g=mint(b/c)*n*(n+1)*iv2;
7          ret.h=mint(b/c)*(b/c)*(n+1);
8          return ret;
9      }
10     if(a>=c||b>=c){
11         ans to=calc(a%c,b%c,c,n);
12         ret.f=mint(a/c)*n*(n+1)*iv2+mint(b/c)*(n+1)+to.f;
13         ret.g=mint(a/c)*n*(n+1)*(n*2+1)*iv6+mint(b/c)*n*(n+1)*iv2+to.g;
14         ret.h=mint(a/c)*(a/c)*n*(n+1)*(n*2+1)*iv6+mint(b/c)*(b/c)*(n+1)+\
15             mint(a/c)*(b/c)*n*(n+1)+to.h+mint(a/c)*2*to.g+mint(b/c)*2*to.f;
16         return ret;
17     }else{
18         ll m=(1ll*a*n+b)/c;
19         ans to=calc(c,c-b-1,a,m-1);

```

```

20     ret.f=min(n*m%P)-to.f;
21     ret.g=(mint(m*n%P*(n+1)%P)-to.f-to.h)*iv2;
22     ret.h=min(m*n%P*(m+1)%P)-to.g*2-to.f*2-ret.f;
23     return ret;
24 }
25 }

```

Pollard-Rho

```

1  template<const int test_case>// set 8 usually
2  struct Pollard_Rho {
3      vector<long long> fac;
4      long long quick_pow(long long a, long long b, long long mod) {
5          long long ans = 1;
6          while (b) {
7              if (b&1) ans = (__int128)ans*(__int128)a%mod;
8              b >>= 1, a = (__int128)a*(__int128)a%mod;
9          }
10         return ans;
11     }
12     bool Miller_Rabin(long long n) {// return if n is a prime
13         if (n < 3) return n == 2;
14         long long a = n-1, b = 0;
15         while (a%2 == 0) a /= 2, ++b;
16         for (int i = 0, j; i < test_case; i++) {
17             long long x = rand()%(n-2)+2, v = quick_pow(x, a, n);
18             if (v == 1 || v == n-1) continue;
19             for (j = 0; j < b; j++) {
20                 v = (__int128)v*(__int128)v%n;
21                 if (v == n-1) break;
22             }
23             if (j >= b) return false;
24         }
25         return true;
26     }
27     long long f(long long x, long long c, long long n) { return ((__int128)x * x + c) % n; }
28     long long rho(long long x) {
29         long long s = 0, t = 0;
30         long long c = (__int128)rand() % (x - 1) + 1;
31         int step = 0, goal = 1;
32         long long val = 1;
33         for (goal = 1;; goal <= 1, s = t, val = 1) {
34             for (step = 1; step <= goal; ++step) {
35                 t = f(t, c, x);
36                 val = (__int128)val * abs(t - s) % x;
37                 if ((step % 127) == 0) {
38                     long long d = __gcd(val, x);
39                     if (d > 1) return d;
40                 }
41             }
42             long long d = __gcd(val, x);
43             if (d > 1) return d;
44         }
45     }
46     void find(long long x) {
47         if (x == 1) return;
48         if (Miller_Rabin(x)) {
49             fac.push_back(x);
50             return;
51         }
52         long long p = x;
53         while (p >= x) p = rho(x);
54         //while ((x % p) == 0) x /= p;
55         find(x/p), find(p);
56     }
57     vector<long long> factor(long long n) {// return the factors of n
58         srand((unsigned)time(NULL));
59         fac.clear();
60         find(n);
61         sort(fac.begin(), fac.end());
62         return fac;

```

```

63     }
64 };

```

ex-gcd

```

1  template<typename T>
2  struct ex_gcd {
3      T gcd(const T a, const T b, T &x, T &y) { // x'=x_0+b/gcd, y'=y_0-a/gcd
4          if (b == 0) {x = 1, y = 0; return a; }
5          T d = gcd(b, a%b, x, y);
6          T t = x;
7          x = y;
8          y = t - a/b*y;
9          return d;
10     }
11     T inv(const T a, const T m) { // return -1 if inv is not exist
12         if (a == 0 || m <= 1) return -1;
13         T x, y, d = gcd(a, m, x, y);
14         if (d != 1) return -1;
15         return (x%m+m)%m;
16     }
17 };

```

crt

```

1  template<typename T>
2  struct crt {
3      ex_gcd<T> *exgcd = new ex_gcd<T>();
4      T cal(const T *a, const T *m, const int n) { // a[1..n], m[1..n], gcd(m_i) = 1
5          T M = 1, ans = 0;
6          for (int i = 1; i <= n; i++) M *= m[i];
7          for (int i = 1; i <= n; i++)
8              (ans += (__int128)a[i]*(M/m[i])%M*exgcd->inv(M/m[i], m[i])%M) %= M;
9          return ans;
10     }
11 };

```

ex-crt

```

1  template<typename T>
2  struct ex_crt {
3      ex_gcd<T> *exgcd = new ex_gcd<T>();
4      T cal(T *a, T *m, const int n) { // a[1..n], m[1..n], return -1 if no ans
5          T x, y, gcd, lcm;
6          for (int i = 2; i <= n; i++) {
7              gcd = exgcd->gcd(m[1], m[i], x, y);
8              if ((a[i]-a[1])%gcd) return -1;
9              lcm = (__int128)m[1]*m[i]/gcd;
10             x = (__int128)x*(a[i]-a[1])/gcd%lcm;
11             gcd = m[i]/gcd;
12             x = (x%gcd+gcd)%gcd;
13             a[1] = ((__int128)m[1]*x%lcm+a[1])%lcm, m[1] = lcm;
14         }
15         return a[1];
16     }
17 };

```

Meissel-Lehmer

求解 $1e11$ 内的质数个数, 约为 $O(n^{2/3})$ 。

```

1  namespace pcf{
2      #define chkbit(ar, i) (((ar[(i) >> 6]) & (1 << (((i) >> 1) & 31))))
3      #define setbit(ar, i) (((ar[(i) >> 6]) |= (1 << (((i) >> 1) & 31))))
4      #define isprime(x) (( (x) && ((x)&1) && (!chkbit(ar, (x)))) || ((x) == 2))
5      const int MAXN=100;
6      const int MAXM=100001;
7      const int MAXP=400000;
8      const int MAX=4000000;

```

```

9   long long dp[MAXN][MAXM];
10  unsigned int ar[(MAX >> 6) + 5] = {0};
11  int len = 0, primes[MAXP], counter[MAX];
12  void Sieve(){
13      setbit(ar, 0), setbit(ar, 1);
14      for (int i = 3; (i * i) < MAX; i++, i++){
15          if (!chkbit(ar, i)){
16              int k = i << 1;
17              for (int j = (i * i); j < MAX; j += k) setbit(ar, j);
18          }
19      }
20      for (int i = 1; i < MAX; i++){
21          counter[i] = counter[i - 1];
22          if (isprime(i)) primes[len++] = i, counter[i]++;
23      }
24  }
25  void init(){
26      Sieve();
27      for (int n = 0; n < MAXN; n++){
28          for (int m = 0; m < MAXM; m++){
29              if (!n) dp[n][m] = m;
30              else dp[n][m] = dp[n - 1][m] - dp[n - 1][m / primes[n - 1]];
31          }
32      }
33  }
34  long long phi(long long m, int n){
35      if (n == 0) return m;
36      if (primes[n - 1] >= m) return 1;
37      if (m < MAXM && n < MAXN) return dp[n][m];
38      return phi(m, n - 1) - phi(m / primes[n - 1], n - 1);
39  }
40  long long Lehmer(long long m){
41      if (m < MAX) return counter[m];
42      long long w, res = 0;
43      int i, a, s, c, x, y;
44      s = sqrt(0.9 + m), y = c = cbrt(0.9 + m);
45      a = counter[y], res = phi(m, a) + a - 1;
46      for (i = a; primes[i] <= s; i++) res = res - Lehmer(m / primes[i]) + Lehmer(primes[i]) - 1;
47      return res;
48  }
49  }
50  int main(){
51      pcf::init();
52      long long n;
53      while (scanf("%lld", &n) != EOF){
54          printf("%lld\n", pcf::Lehmer(n));
55      }
56      return 0;
57  }

```

Cipolla 二次剩余

- $x^2 \equiv n \pmod{P}$
- 仅有两个解，返回小的那个，另一个是相反数。
- 大概 1s 能跑 $1e5$ 个数

```

1  inline int qpow(int a, int b){
2      int q=1; while(b){if(b&1)q=1ll*q*a%P;a=1ll*a*a%P;b>>=1;}return q;
3  }
4  namespace Cipolla{
5      ll w,a;
6      struct node{
7          ll x,y;
8          node friend operator *(node x,node y){
9              node z;
10             z.x=(x.x*y.x%P+x.y*y.y%P*w%P)%P;
11             z.y=(x.x*y.y%P+x.y*y.x%P)%P;
12             return z;
13         }
14     }u,v;

```

```

15 inline node Cqpow(node a,ll b){
16     node q;q.x=1;q.y=0;
17     while(b){if(b&1) q=q*a;a=a*a;b>>=1;}
18     return q;
19 }
20 inline ll cipolla(int n){
21     n%=P;rand(0x20010412);
22     if(P==2) return n;
23     if(!n) return n;
24     if(qpow(n,(P-1)/2)==P-1) return -1;
25     while(1){
26         a=rand()%P;
27         w=(a*a-n+P)%P;
28         if(qpow(w%P,(P-1)/2)==P-1) break;
29     }
30     u.x=a,u.y=1;
31     u=Cqpow(u,(P+1)/2);
32     ll fir=u.x,sec=P-u.x;
33     if(fir>sec)swap(fir,sec);
34     return fir;
35 }
36 }

```

BSGS

```

1 template<typename T>
2 struct BSGS {
3     T cal(T a, T b, T c) { // return  $a^x = b \pmod c$ ,  $\gcd(a, c) = 1$ 
4         mp.clear();
5         T tim = ceil(sqrt(c)), tmp = b%c;
6         for (int i = 0; i <= tim; i++) {
7             mp[tmp] = i; tmp = (__int128)tmp*a%c;
8         }
9         T t = tmp = quick_pow(a, tim, c);
10        for (int i = 1; i <= tim; i++) {
11            if (mp.count(tmp)) return tim*i-mp[tmp];
12            tmp = (__int128)tmp*t%c;
13        }
14        return -1;
15    }
16 } ;

```

exBSGS

```

1 template<typename T>
2 struct exBSGS {
3     T cal(T a, T b, T c) { // return  $a^x = b \pmod c$ 
4         if (b == 1) return 0;
5         T cnt = 0, d = 1, t;
6         while ((t = __gcd(a, c)) != 1) {
7             if (b%t) return -1;
8             ++cnt, b /= t, c /= t, d = (__int128)d*(a/t)%c;
9             if (d == b) return cnt;
10        }
11        mp.clear();
12        T tim = ceil(sqrt(c)), tmp = b%c;
13        for (int i = 0; i <= tim; i++) {
14            mp[tmp] = i; tmp = (__int128)tmp*a%c;
15        }
16        t = tmp = quick_pow(a, tim, c); tmp = (__int128)tmp*d%c;
17        for (int i = 1; i <= tim; i++) {
18            if (mp.count(tmp)) return tim*i-mp[tmp]+cnt;
19            tmp = (__int128)tmp*t%c;
20        }
21        return -1;
22    }
23 } ;

```

exLucas

```
1  template<typename T>
2  struct exLucas {
3      T quick_pow(T a, T b, T p) {
4          T ans = 1;
5          while (b) {
6              if (b&1) ans = (__int128)ans*a%p;
7              b >>= 1, a = (__int128)a*a%p;
8          }
9          return ans;
10     }
11     void ex_gcd(T a, T b, T &x, T &y) {
12         if (b == 0) {x = 1, y = 0; return; }
13         ex_gcd(b, a%b, x, y);
14         T t = x; x = y, y = t-a/b*y;
15     }
16     T inv(T a, T p) {
17         T x, y; ex_gcd(a, p, x, y);
18         return (x%p+p)%p;
19     }
20     T mul(T n, T pi, T pk) {
21         if (!n) return 1;
22         T ans = 1;
23         for (int i = 2; i <= pk; i++) if (i%pi != 0) ans = (__int128)ans*i%pk;
24         ans = quick_pow(ans, n/pk, pk);
25         for (int i = 2; i <= n%pk; i++) if (i%pi != 0) ans = (__int128)ans*i%pk;
26         return (__int128)ans*mul(n/pi, pi, pk)%pk;
27     }
28     T C(T n, T m, T pi, T pk, T p) {
29         T a = mul(n, pi, pk), b = mul(m, pi, pk), c = mul(n-m, pi, pk);
30         T k = 0;
31         for (T i = n; i; i /= pi) k += i/pi;
32         for (T i = m; i; i /= pi) k -= i/pi;
33         for (T i = n-m; i; i /= pi) k -= i/pi;
34         return (__int128)a*inv(b, pk)%pk*inv(c, pk)%pk*quick_pow(pi, k, pk)%pk;
35     }
36     T ex_lucas(T n, T m, T p) {
37         T ans = 0;
38         for (T i = 2, x = p; i <= x; i++)
39             if (x%i == 0) {
40                 T k = 1; while (x%i == 0) k *= i, x /= i;
41                 (ans += (__int128)C(n, m, i, k, p)*(p/k)%p*inv(p/k, k)%p) %= p;
42             }
43         return ans;
44     }
45 } ;
```

min_25 筛

```
1  /* 「LOJ #6053」简单的函数 */
2  #include <algorithm>
3  #include <cmath>
4  #include <cstdio>
5
6  using i64 = long long;
7
8  constexpr int maxs = 200000; // 2sqrt(n)
9  constexpr int mod = 1000000007;
10
11 template <typename x_t, typename y_t>
12 inline void inc(x_t &x, const y_t &y) {
13     x += y;
14     (mod <= x) && (x -= mod);
15 }
16 template <typename x_t, typename y_t>
17 inline void dec(x_t &x, const y_t &y) {
18     x -= y;
19     (x < 0) && (x += mod);
20 }
21 template <typename x_t, typename y_t>
```



```

22 inline int sum(const x_t &x, const y_t &y) {
23     return x + y < mod ? x + y : (x + y - mod);
24 }
25 template <typename x_t, typename y_t>
26 inline int sub(const x_t &x, const y_t &y) {
27     return x < y ? x - y + mod : (x - y);
28 }
29 template <typename _Tp>
30 inline int div2(const _Tp &x) {
31     return ((x & 1) ? x + mod : x) >> 1;
32 }
33 template <typename _Tp>
34 inline i64 sqrll(const _Tp &x) {
35     return (i64)x * x;
36 }
37
38 int pri[maxs / 7], lpf[maxs + 1], spri[maxs + 1], pcnt;
39
40 inline void sieve(const int &n) {
41     for (int i = 2; i <= n; ++i) {
42         if (lpf[i] == 0)
43             pri[lpf[i] = ++pcnt] = i, spri[pcnt] = sum(spri[pcnt - 1], i);
44         for (int j = 1, v; j <= lpf[i] && (v = i * pri[j]) <= n; ++j) lpf[v] = j;
45     }
46 }
47
48 i64 global_n;
49 int lim;
50 int le[maxs + 1], // x \le \sqrt{n}
51     ge[maxs + 1]; // x > \sqrt{n}
52 #define idx(v) (v <= lim ? le[v] : ge[global_n / v])
53
54 int G[maxs + 1][2], Fprime[maxs + 1];
55 i64 lis[maxs + 1];
56 int cnt;
57
58 inline void init(const i64 &n) {
59     for (i64 i = 1, j, v; i <= n; i = n / j + 1) {
60         j = n / i;
61         v = j % mod;
62         lis[++cnt] = j;
63         idx(j) = cnt;
64         G[cnt][0] = sub(v, 1ll);
65         G[cnt][1] = div2((i64)(v + 2ll) * (v - 1ll) % mod);
66     }
67 }
68
69 inline void calcFprime() {
70     for (int k = 1; k <= pcnt; ++k) {
71         const int p = pri[k];
72         const i64 sqrp = sqrll(p);
73         for (int i = 1; lis[i] >= sqrp; ++i) {
74             const i64 v = lis[i] / p;
75             const int id = idx(v);
76             dec(G[i][0], sub(G[id][0], k - 1));
77             dec(G[i][1], (i64)p * sub(G[id][1], spri[k - 1] % mod));
78         }
79     }
80     /* F_prime = G_1 - G_0 */
81     for (int i = 1; i <= cnt; ++i) Fprime[i] = sub(G[i][1], G[i][0]);
82 }
83
84 inline int f_p(const int &p, const int &c) {
85     /* f(p^{c}) = p xor c */
86     return p xor c;
87 }
88
89 int F(const int &k, const i64 &n) {
90     if (n < pri[k] || n <= 1) return 0;
91     const int id = idx(n);
92     i64 ans = Fprime[id] - (spri[k - 1] - (k - 1));

```

```

93     if (k == 1) ans += 2;
94     for (int i = k; i <= pcnt && sqrrll(pri[i]) <= n; ++i) {
95         i64 pw = pri[i], pw2 = sqrrll(pw);
96         for (int c = 1; pw2 <= n; ++c, pw = pw2, pw2 *= pri[i])
97             ans +=
98                 ((i64)f_p(pri[i], c) * F(i + 1, n / pw) + f_p(pri[i], c + 1)) % mod;
99     }
100     return ans % mod;
101 }
102
103 int main() {
104     scanf("%lld", &global_n);
105     lim = sqrt(global_n);
106
107     sieve(lim + 1000);
108     init(global_n);
109     calcFprime();
110     printf("%lld\n", (F(1, global_n) + 1ll + mod) % mod);
111
112     return 0;
113 }

```

超现实数

```

1  #include <bits/stdc++.h>
2  const int inf=1000;
3  const int maxn=3e5;
4  using namespace std;
5  struct fs{                                     //分数结构体
6      int fz, fm;
7      fs(int _fz=0, int _fm=1): fz(_fz), fm(_fm){}
8      bool operator==(const fs &oth) const{
9          return fz*oth.fm==fm*oth.fz;
10     }
11     friend int ceil(const fs &x){
12         return (int)ceil(1.0*x.fz/x.fm);
13     }
14     friend int floor(const fs &x){
15         return (int)floor(1.0*x.fz/x.fm);
16     }
17     friend fs abs(const fs &x){
18         return fs(abs(x.fz), abs(x.fm));
19     }
20     bool operator<(const fs &oth) const{
21         return (double)fz/fm < (double)oth.fz/oth.fm;
22         //return fz*oth.fm < fm*oth.fz;    //不能这么写啊    负数的时候不成立，直接用 double 就完事了
23     }
24     fs operator+(const fs &oth) const{
25         int n_fm=fm*oth.fm;
26         int n_fz=fz*oth.fm+fm*oth.fz;
27         int yf=__gcd(n_fz, n_fm);
28         return fs(n_fz/yf, n_fm/yf);
29     }
30     friend void print(const fs &x){
31         printf("%d/%d\n", x.fz, x.fm);
32     }
33 };
34 int pd(int dex, int n){                         //判断二进制状压后的第 n 位上的棋子类型
35     if(dex < (1 << (2*n-2))) return 2;         //二进制的右二位、最右位表示第一枚棋子，以此类推
36     int flag1, flag2;
37     dex >>= (2*n)-2;
38     flag1=dex&1;
39     if(flag1) return 0;                         //11 空格
40     dex >>= 1;
41     flag2=dex&1;
42     if(flag2) return 1;                         // 10 黑色
43     else return 2;                             // 00 白色
44 }
45 void solve(int x){                             //逆向解出棋盘状态
46     char s[7][7];
47     for(int i=1; i<=3; i++){

```

```

48     if(x&1)s[i][0]='#';
49     else if((x>>1)&1)s[i][0]='X';
50     else s[i][0]='0';
51     s[i][1]='|';
52     x>>=2;
53     if(x&1)s[i][2]='#';
54     else if((x>>1)&1)s[i][2]='X';
55     else s[i][2]='0';
56     s[i][3]='|';
57     x>>=2;
58     if(x&1)s[i][4]='#';
59     else if((x>>1)&1)s[i][4]='X';
60     else s[i][4]='0';
61     s[i][5]=0;
62     x>>=2;
63 }
64 for(int i=1;i<=3;i++)
65     printf("%s\n",s[i]);
66 }
67 int sw(int dex,int n,int typ){ // 换第 i 个石头变为空格
68     int tmp=dex;
69     tmp|=3<<(2*n-2);
70     if(typ==1 || typ==3){ //变换左右
71         if(n%3!=0){
72             tmp|=3<<(2*(n+1)-2);
73         }
74         if(n%3!=1){
75             tmp|=3<<(2*(n-1)-2);
76         }
77     }
78     if(typ==2 || typ==3){ //变换上下
79         if((n+2)/3!=1){
80             tmp|=3<<(2*(n-3)-2);
81         }
82         if((n+2)/3!=3){
83             tmp|=3<<(2*(n+3)-2);
84         }
85     }
86     return tmp;
87 }
88 fs a[maxn];
89 void init(){ //初始化, 将数组中每一个值都赋 inf+2
90     for(int i=0;i<maxn;i++){
91         a[i].fm=1;
92         a[i].fz=inf+2;
93     }
94 }
95 fs getl(int dex); //获取左集合的 S
96 fs getr(int dex); //获取右集合的 S
97 fs getsr(int dex); //获取 { | } 整体的 S
98 fs calc(fs l,fs r){ //get 辅助函数 Surreal Number 计算规则
99     if (r<l) swap(l,r);
100     int x=ceil(l);
101     if (l==x) ++x;
102     if (fs(x)<r){
103         if (fs(0)<l || l==0 || l<0&&fs(0)<r) return x;
104         int y=floor(r);
105         if (fs(y)==r) --y;
106         if (r<0 || r==0) return y;
107         return y;
108     }
109     for (int y=1;;y*=2){
110         for (x=1;;++x){
111             fs tmp1=fs(x,y);
112             fs tmp2=fs(-x,y);
113             if (l<tmp1&&tmp1<r) return tmp1;
114             if (l<tmp2&&tmp2<r) return tmp2;
115             if (fs(0)<r&&r<tmp1) break;
116             if (l<0&&tmp2<l) break;
117         }
118     }

```

```

119     return fs(0);
120 }
121 fs get(fs l,fs r){ //计算 S
122     fs res=fs(0);
123     if(l.fz==-inf-1 && r.fz==inf+1) return 0; //当左集合和右集合都是空, 输出 0
124     else if(l.fz==-inf-1){ //当左集合为空, 输出 S (右集合) -1
125         res=res+r+fs(-1,1);
126     }
127     else if(r.fz==inf+1){ //当右集合为空, 输出 S (左集合) +1
128         res=res+l+fs(1,1);
129     }
130     else{
131         res=calc(l,r); //调用 calc 计算
132     }
133     return res;
134 }
135 fs getl(int dex){ //左集合最大值
136     fs res=-inf-1; //若左集合为空, 输出 -inf-1
137     for(int i=1;i<=9;i++){
138         if(pd(dex,i)==2){
139             res=max(res,getsr(sw(dex,i,1))); //Alice 三种选择, 取最大值
140             res=max(res,getsr(sw(dex,i,2)));
141             res=max(res,getsr(sw(dex,i,3)));
142         }
143     }
144     return res;
145 }
146 fs getr(int dex){ //右集合最小值
147     fs res=inf+1; //若右集合为空, 输出 inf+1
148     for(int i=1;i<=9;i++){
149         if(pd(dex,i)==1){
150             res=min(res,getsr(sw(dex,i,0)));
151         }
152     }
153     return res;
154 }
155 fs getsr(int dex){ //获得某个状态的 sunr
156     if(a[dex].fz!=inf+2){ //如果已经保存过, 则直接取用, 如果没有则计算。
157         return a[dex];
158     }
159     else{
160         fs m=getl(dex),n=getr(dex);
161         a[dex]=get(m,n);
162         //printf("getl,getr=\n");
163         //print(m);
164         //print(n);
165         //print(a[dex]);
166         return a[dex];
167     }
168 }
169 int make(){ //构造映射, 将棋盘状态转化为二进制数
170     int res=0;
171     char s[10];
172     int dex=1;
173     for(int i=1;i<=3;i++){
174         scanf("%s",s);
175         for(int j=1;j<=3;j++){
176             int tmp=0;
177             if(s[2*j-2]=='X') tmp=2;
178             else if(s[2*j-2]=='.' || s[2*j-2]=='#') tmp=3;
179             res+=(tmp<<(2*dex-2));
180             dex++;
181         }
182     }
183     return res;
184 }
185 int main(){
186     init();
187     int ca;
188     scanf("%d",&ca);
189     while(ca--){

```

```

190     int n;
191     fs res=fs(0);
192     scanf("%d",&n);
193     for(int i=0;i<n;i++){
194         int dex=make();
195         res=res+getsr(dex);
196     }
197     if(fs(0)<res)printf("Alice\n");
198     else if(res<fs(0))printf("Bob\n");
199     else printf("Second\n");
200 }
201 return 0;
202 }

```

图论

LCA

- 倍增

```

1 void dfs(int u, int fa) {
2     pa[u][0] = fa; dep[u] = dep[fa] + 1;
3     FOR (i, 1, SP) pa[u][i] = pa[pa[u][i - 1]][i - 1];
4     for (int& v: G[u]) {
5         if (v == fa) continue;
6         dfs(v, u);
7     }
8 }
9
10
11 int lca(int u, int v) {
12     if (dep[u] < dep[v]) swap(u, v);
13     int t = dep[u] - dep[v];
14     FOR (i, 0, SP) if (t & (1 << i)) u = pa[u][i];
15     FORD (i, SP - 1, -1) {
16         int uu = pa[u][i], vv = pa[v][i];
17         if (uu != vv) { u = uu; v = vv; }
18     }
19     return u == v ? u : pa[u][0];
20 }

```

同余最短路

```

1 //d[i] = k[i]*a[1]+i
2 //smallest d[i] % a[1] ==i
3 //O(n*a)
4 sort(a+1,a+n+1,greater<int>());
5 rep(i,0,a[1]-1) k[i]=inf;
6 k[0]=0;
7 deque<int> q;
8 q.emplace_front(0);
9 while(!q.empty()){
10     int x=q.front();q.pop_front();
11     if(vis[x]) continue;
12     vis[x]=1;
13     rep(i,2,n){
14         if(a[i]+x<a[1]){
15             chmin(k[x+a[i]],k[x]);
16             q.emplace_front(x+a[i]);
17         }else{
18             int to=(x+a[i])%a[1];
19             chmin(k[to],k[x]+1);
20             q.emplace_back(to);
21         }
22     }
23 }

```

计算几何

二维几何：点与向量

```
1  #define y1 yy1
2  #define nxt(i) ((i + 1) % s.size())
3  typedef double LD;
4  const LD PI = 3.14159265358979323846;
5  const LD eps = 1E-10;
6  int sgn(LD x) { return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1); }
7  struct L;
8  struct P;
9  typedef P V;
10 struct P {
11     LD x, y;
12     explicit P(LD x = 0, LD y = 0): x(x), y(y) {}
13     explicit P(const L& l);
14 };
15 struct L {
16     P s, t;
17     L() {}
18     L(P s, P t): s(s), t(t) {}
19 };
20
21 P operator + (const P& a, const P& b) { return P(a.x + b.x, a.y + b.y); }
22 P operator - (const P& a, const P& b) { return P(a.x - b.x, a.y - b.y); }
23 P operator * (const P& a, LD k) { return P(a.x * k, a.y * k); }
24 P operator / (const P& a, LD k) { return P(a.x / k, a.y / k); }
25 inline bool operator < (const P& a, const P& b) {
26     return sgn(a.x - b.x) < 0 || (sgn(a.x - b.x) == 0 && sgn(a.y - b.y) < 0);
27 }
28 bool operator == (const P& a, const P& b) { return !sgn(a.x - b.x) && !sgn(a.y - b.y); }
29 P::P(const L& l) { *this = l.t - l.s; }
30 ostream &operator << (ostream &os, const P &p) {
31     return (os << "(" << p.x << ", " << p.y << ")");
32 }
33 istream &operator >> (istream &is, P &p) {
34     return (is >> p.x >> p.y);
35 }
36
37 LD dist(const P& p) { return sqrt(p.x * p.x + p.y * p.y); }
38 LD dot(const V& a, const V& b) { return a.x * b.x + a.y * b.y; }
39 LD det(const V& a, const V& b) { return a.x * b.y - a.y * b.x; }
40 LD cross(const P& s, const P& t, const P& o = P()) { return det(s - o, t - o); }
41 // -----
```

完整的板板 by zcs

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  const double pi=acos(-1.0); //高精度圆周率
4  const double eps=1e-8; //偏差值
5  const int maxp=200005; //点的数量
6  int sgn(double x) //判断 x 是否为 0
7  {
8      if(fabs(x)<eps) return 0;
9      else return x<0?-1:1; //小于 0 返回-1, 大于 0 返回 1
10 }
11 int Dcmp(double x,double y) //比较浮点数大小
12 {
13     if(fabs(x-y)<eps) return 0;
14     else return x<y?-1:1;
15 }
16 //-----平面几何: 点和线-----
17 struct Point
18 {
19     double x,y;
20     Point(){}
21     Point(double x,double y):x(x),y(y){}
22     Point operator + (Point B){return Point(x+B.x,y+B.y);}
```

```

23 Point operator - (Point B){return Point(x-B.x,y-B.y);}
24 Point operator * (double k){return Point(x*k,y*k);} //长度扩大 k 倍
25 Point operator / (double k){return Point(x/k,y/k);} //长度缩小 k 倍
26 bool operator == (Point B){return sgn(x-B.x)==0 && sgn(y-B.y)==0;}
27 };
28 typedef Point Vector;
29 double Dot(Vector A,Vector B){return A.x*B.x+A.y*B.y;} //向量点乘
30 double Len(Vector A){return sqrt(Dot(A,A));} //向量取模
31 double Angle(Vector A,Vector B){return acos(Dot(A,B)/Len(A)/Len(B));} //向量 A 和 B 夹角
32 double Cross(Vector A,Vector B){return A.x*B.y-A.y*B.x;} //向量叉乘
33 //double Area(Point A,Point B,Point C){return Cross(B-A,C-A);} //三角形面积 2 倍
34 double Distance(Point A,Point B){return hypot(A.x-B.x,A.y-B.y);} //两点距离
35 Vector Normal(Vector A){return Vector(-A.y/Len(A),A.x/Len(A));} //向量 A 的单位 * 法 * 向量
36 bool Parallel(Vector A,Vector B){return sgn(Cross(A,B))==0;} //平行
37 Vector Rotate(Vector A,double rad) //向量旋转
38 {return Vector(A.x*cos(rad)-A.y*sin(rad),A.x*sin(rad)+A.y*cos(rad));}
39 struct Line
40 {
41     Point p1,p2;
42     Line(){}
43     Line(Point p1,Point p2):p1(p1),p2(p2){} //两点确定直线
44     Line(Point p,double angle) //点 + 倾斜角
45     {
46         p1=p;
47         if(sgn(angle-pi/2)==0){p2=(p1+Point(0,1));}
48         else {p2=(p1+Point(1,tan(angle)));}
49     }
50     Line(double a,double b,double c) //ax+by+c=0;
51     {
52         if(sgn(a)==0) {p1=Point(0,-c/b);p2=Point(1,-c/b);}
53         else if(sgn(b)==0) {p1=Point(-c/a,0);p2=Point(-c/a,1);}
54         else {p1=Point(0,-c/b);p2=Point(1,(-c-a)/b);}
55     }
56 };
57 typedef Line Segment;
58 //直线倾斜角, 返回值 [0,pi);
59 double Line_angle(Line v)
60 {
61     double k=atan2(v.p2.y-v.p1.y,v.p2.x-v.p1.x);
62     if(sgn(k)<0) k+=pi;
63     if(sgn(k-pi)==0) k-=pi;
64     return k;
65 }
66 //点和直线关系
67 int Point_line_relation(Point p,Line v)
68 {
69     int c=sgn(Cross(p-v.p1,v.p2-v.p1));
70     if(c<0) return 1; //1:p 在 v 左侧
71     if(c>0) return 2; //2:p 在 v 右侧
72     return 0; //0:p 在 v 上
73 }
74 //点和线段关系: 0 为 p 不在线段 v 上; 1 为 p 在线段 v 上
75 bool Point_on_seg(Point p,Segment v)
76 {
77     return sgn(Cross(p-v.p1,v.p2-v.p1))==0 && sgn(Dot(p-v.p1,p-v.p2))<=0;
78 }
79 //两直线的关系: 0 为平行, 1 为重合, 2 为相交
80 int Line_relation(Line v1,Line v2)
81 {
82     if(sgn(Cross(v1.p2-v1.p1,v2.p2-v2.p1))==0)
83     {
84         if(Point_line_relation(v1.p1,v2)==0) return 1;
85         else return 0;
86     }
87     return 2;
88 }
89 //点到直线距离
90 double Dis_point_line(Point p,Line v)
91 {
92     return fabs(Cross(p-v.p1,v.p2-v.p1))/Distance(v.p1,v.p2);
93 }

```

```

94 //点在直线上的投影
95 Point Point_line_proj(Point p,Line v)
96 {
97     double k=Dot(v.p2-v.p1,p-v.p1)/Dot(v.p2-v.p1,v.p2-v.p1);
98     return v.p1+(v.p2-v.p1)*k;
99 }
100 //点 p 对直线 v 的对称点
101 Point Point_line_symmetry(Point p,Line v)
102 {
103     Point q=Point_line_proj(p,v);
104     return Point(2*q.x-p.x,2*q.y-p.y);
105 }
106 //点到线段的距离
107 double Dis_point_seg(Point p,Segment v)
108 {
109     if(sgn(Dot(p-v.p1,v.p2-v.p1))<0 || sgn(Dot(p-v.p2,v.p1-v.p2))<0)
110         return min(Distance(p,v.p1),Distance(p,v.p2)); //点的投影不在线段上
111     return Dis_point_line(p,v); //点的投影在线段上
112 }
113 //求两直线 ab 和 cd 的交点, 在调用前要保证两直线不平行或重合
114 Point Cross_point(Point a,Point b,Point c,Point d)
115 {
116     double s1=Cross(b-a,c-a);
117     double s2=Cross(b-a,d-a);
118     return Point(c.x*s2-d.x*s1,c.y*s2-d.y*s1)/(s2-s1);
119 }
120 //线段 ab 和 cd 是否相交
121 bool Cross_segment(Point a,Point b,Point c,Point d)
122 {
123     double c1=Cross(b-a,c-a),c2=Cross(b-a,d-a);
124     double d1=Cross(d-c,a-c),d2=Cross(d-c,b-c);
125     return sgn(c1)*sgn(c2)<=0 && sgn(d1)*sgn(d2)<=0;
126 }
127 //-----平面几何: 多边形-----
128 struct Polygon
129 {
130     int n;
131     Point p[maxp]; //从 0 开始
132     Line v[maxp];
133 };
134 //极角排序
135 bool Polar_angle_cmp(Point a,Point b)
136 {
137     if(Cross(a,b)==0) return a.x<b.x;
138     else return Cross(a,b)>0;
139 }
140 //按照 x 大小排序 (计算凸包使用)
141 bool Hull_cmp(Point A,Point B)
142 {
143     return sgn(A.x-B.x)<0 || (sgn(A.x-B.x)==0 && sgn(A.y-B.y)<0);
144 }
145 //判断点和任意多边形的关系: 3 为点上; 2 为边上; 1 为内部; 0 为外部
146 int Point_in_polygon(Point pt,Point *p,int n) //点 pt, 多边形 *p
147 {
148     for(int i=0;i<n;i++)
149         if(p[i]==pt) return 3;
150     for(int i=0;i<n;i++)
151     {
152         Line v=Line(p[i],p[(i+1)%n]);
153         if(Point_on_seg(pt,v)) return 2;
154     }
155     int num=0;
156     for(int i=0;i<n;i++)
157     {
158         int j=(i+1)%n;
159         int c=sgn(Cross(pt-p[j],p[i]-p[j]));
160         int u=sgn(p[i].y-pt.y);
161         int v=sgn(p[j].y-pt.y);
162         if(c>0 && u<0 && v>=0) num++;
163         if(c<0 && u>=0 && v<0) num--;
164     }

```



```

165     return num!=0;
166 }
167 //多边形面积
168 double Polygon_area(Point *p,int n)
169 {
170     double area=0;
171     for(int i=0;i<n;i++)
172         area+=Cross(p[i],p[(i+1)%n]);
173     return area/2;
174 }
175 //求多边形重心
176 Point Polygon_center(Point *p,int n)
177 {
178     Point ans(0,0);
179     if(Polygon_area(p,n)==0) return ans;
180     for(int i=0;i<n;i++)
181         ans=ans+(p[i]+p[(i+1)%n])*Cross(p[i],p[(i+1)%n]);
182     return ans/Polygon_area(p,n)/6;
183 }
184 //Convex_hull() 求凸包, 凸包顶点放在 ch 中, 返回值是凸包的顶点数
185 int Convex_hull(Point *p,int n,Point *ch)
186 {
187     sort(p,p+n,Hull_cmp);
188     n=unique(p,p+n)-p;
189     int v=0;
190     //求下凸包, 如果 p[i] 是右拐的, 则不在凸包上, 往回退
191     for(int i=0;i<n;i++)
192     {
193         while(v>1 && sgn(Cross(ch[v-1]-ch[v-2],p[i]-ch[v-2]))<=0)
194             v--;
195         ch[v++]=p[i];
196     }
197     int j=v;
198     //求上凸包
199     for(int i=n-2;i>=0;i--)
200     {
201         while(v>j && sgn(Cross(ch[v-1]-ch[v-2],p[i]-ch[v-2]))<=0)
202             v--;
203         ch[v++]=p[i];
204     }
205     if(n>1) v--;
206     return v;
207 }
208 //-----平面几何: 圆-----
209 struct Circle
210 {
211     Point c; //圆心
212     double r;
213     Circle(){}
214     Circle(Point c,double r):c(c),r(r){}
215     Circle(double x,double y,double _r){c=Point(x,y);r=_r;}
216 };
217 //点和圆的关系: 0 为圆内, 1 为圆上, 2 为圆外
218 int Point_circle_relation(Point p,Circle C)
219 {
220     double dst=Distance(p,C.c);
221     if(sgn(dst-C.r)<0) return 0;
222     if(sgn(dst-C.r)==0) return 1;
223     return 2;
224 }
225 //直线和圆的关系: 0 为直线和圆相交, 1 为直线和圆相切, 2 为直线和圆相离
226 int Line_circle_relation(Line v,Circle C)
227 {
228     double dst=Dis_point_line(C.c,v);
229     if(sgn(dst-C.r)<0) return 0;
230     if(sgn(dst-C.r)==0) return 1;
231     return 2;
232 }
233 //线段和圆的关系: 0 为线段和圆相交, 1 为线段和圆相切, 2 为线段和圆相离
234 int Seg_circle_relation(Segment v,Circle C)
235 {

```

```

236     double dst=Dis_point_seg(C.c,v);
237     if(sgn(dst-C.r)<0) return 0;
238     if(sgn(dst-C.r)==0) return 1;
239     return 2;
240 }
241 //直线和圆的交点, pa,pb 是交点, 返回值是交点个数
242 int Line_cross_circle(Line v,Circle C,Point &pa,Point &pb)
243 {
244     if(Line_circle_relation(v,C)==2) return 0;           //无交点
245     Point q=Point_line_proj(C.c,v);
246     double d=Dis_point_line(C.c,v);
247     double k=sqrt(C.r*C.r-d*d);
248     if(sgn(k)==0)
249     {
250         pa=q;pb=q;return 1;
251     }
252     Point n=(v.p2-v.p1)/Len(v.p2-v.p1);                //直线的单位向量
253     pa=q+n*k;
254     pb=q-n*k;
255     return 2;
256 }

```

字符串

kmp

```

1  inline void kmp(char *s,int *f){
2      //enum from 1
3      //every i : s[i-f[i]+1...i]=s[1...f[i]]
4      int j=0,n=strlen(s+1);
5      f[1]=0;
6      for(int i=2;i<=n;++i){
7          while(j&&s[j+1]!=s[i]) j=f[j];
8          j+=(s[j+1]==s[i]);
9          f[i]=j;
10     }
11 }

```

exkmp

```

1  inline void ex_kmp(char *s,int *nxt,int n){
2      //ENUM FROM 1
3      //s[1..next[i]]=s[i...i+next[i]-1]
4      int a=0,l=0,p=0;
5      nxt[1]=n;
6      for(int i=2;i<=n;++i){
7          l=max(min(nxt[i-a+1],p-i+1),0);
8          while(i+l<=n&&s[i+l]==s[i+l]) ++l;
9          nxt[i]=l;
10         if(i+l-1>p) a=i,p=i+l-1;
11     }
12 }

```

manacher

```

1  namespace manacher{
2      //ENUM FROM 0
3      const int N=1.1e7+1000;// CHANGE IT!!! DONT CHANGE +1000
4      char ch[N<<1],s[N];
5      int f[N<<1],id,mx,n,len;
6      // center i == f[i*2] center(i,i+1) == f[i*2+1]
7      // len of palindrome = f[i]-1
8      void init(){
9          n=strlen(s);ch[0]='$';ch[1]='#';
10         for(int i=1;i<=n;++i){
11             ch[i*2]=s[i-1];
12             ch[i*2+1]='#';
13         }

```

```

14         id=0;mx=0;ch[n*2+2]='#';
15         for(int i=0;i<=2*n+10;++i) f[i]=0;
16         for(int i=1;i<=2*n+2;++i){
17             if(i>mx) f[i]=1;else f[i]=min(f[id*2-i],mx-i);
18             while(ch[i-f[i]]==ch[i+f[i]]) ++f[i];
19             if(i+f[i]>mx){mx=i+f[i];id=i;}
20         }
21     }
22 }

```

AC 自动机

```

1  const int N=1e6+10;
2  int c[N][26],val[N],f[N],sz;
3  inline void ins(char *s){
4      int n=(int)strlen(s);int now=0;
5      rep(i,0,n-1){
6          int v=s[i]-'a';
7          if(!c[now][v]) c[now][v]=++sz;
8          now=c[now][v];
9      }
10     ++val[now];
11 }
12 inline void build(){
13     queue<int> q;rep(i,0,25) if(c[0][i]){f[c[0][i]]=0;q.push(c[0][i]);}
14     while(!q.empty()){
15         int x=q.front();q.pop();
16         rep(i,0,25){
17             int &v=c[x][i];
18             if(!v){v=c[f[x]][i];continue;}
19             f[v]=c[f[x]][i];
20             q.push(v);
21         }
22     }
23 }

```

后缀自动机

```

1  const int N=1e6+10;
2  struct SAM{
3      int c[N<<1][26],fa[N<<1],len[N<<1],val[N<<1],last,sz;
4      SAM(){last=sz=1;}
5      void append(int x){
6          int cur=++sz,p=last;last=cur;len[cur]=len[p]+1;
7          for(;p&&!c[p][x];p=fa[p]) c[p][x]=cur;
8          if(!p) fa[cur]=1;
9          else{
10             int q=c[p][x];
11             if(len[q]==len[p]+1) fa[cur]=q;
12             else{
13                 int nq=++sz;len[nq]=len[p]+1;
14                 memcpy(c[nq],c[q],sizeof(c[q]));
15                 fa[nq]=fa[q];fa[q]=fa[cur]=nq;
16                 for(;p&& c[p][x]==q;p=fa[p]) c[p][x]=nq;
17             }
18         }
19         val[cur]=1;
20     }
21 }sam;

```

后缀自动机求第 k 大子串

```

1  const int N=5e5+10;
2  struct SAM{
3      int c[N<<1][26],fa[N<<1],len[N<<1],val[N<<1],last,sz;
4      //dp : routes start from i
5      ll dp[N<<1];
6      SAM(){last=sz=1;}
7      void append(int x){

```

```

8     int cur=++sz,p=last;last=cur;len[cur]=len[p]+1;
9     for(;p&&!c[p][x];p=fa[p]) c[p][x]=cur;
10    if(!p) fa[cur]=1;
11    else{
12        int q=c[p][x];
13        if(len[q]==len[p]+1) fa[cur]=q;
14        else{
15            int nq=++sz;len[nq]=len[p]+1;
16            memcpy(c[nq],c[q],sizeof(c[q]));
17            fa[nq]=fa[q];fa[q]=fa[cur]=nq;
18            for(;p&&c[p][x]==q;p=fa[p]) c[p][x]=nq;
19        }
20    }
21    val[cur]=1;
22    }
23    int bkt[N<<1],id[N<<1];
24    void init(int t){
25        rep(i,1,sz) ++bkt[len[i]];
26        rep(i,1,sz) bkt[i]+=bkt[i-1];
27        rpe(i,sz,1) id[bkt[len[i]]--]=i;
28        if(t){
29            rpe(i,sz,1){
30                int cur=id[i];
31                val[fa[cur]]+=val[cur];//count of occurence
32            }
33        }else{
34            rep(i,1,sz) val[i]=1;
35        }
36        val[1]=0;
37        rpe(i,sz,1){
38            int cur=id[i];
39            dp[cur]=val[cur];
40            rep(j,0,25) dp[cur]+=dp[c[cur][j]];
41        }
42    }
43    void dfs(int cur,int k){
44        if(k<=val[cur]) return;
45        k-=val[cur];
46        rep(i,0,25){
47            int t=c[cur][i];
48            if(t){
49                if(k<=dp[t]){
50                    putchar(i+'a');
51                    dfs(t,k);
52                    return;
53                }
54                k-=dp[t];
55            }
56        }
57    }
58    }sam;
59    char s[N];
60    int main(){
61        scanf("%s",s+1);
62        int t=read(),k=read();
63        int n=(int)strlen(s+1);
64        rep(i,1,n) sam.append(s[i]-'a');
65        sam.init(t);
66        if(sam.dp[1]>=k){
67            sam.dfs(1,k);
68        }else{
69            puts("-1");
70        }
71        return 0;
72    }

```

最小表示法

```

1 //寻找字典序最小的循环表示，下标从 0 开始，字符串扩展两倍
2 //rev true 返回最后一个循环环节的起始点，否则返回第一个
3 int lex_find(int s[],int n,bool rev){

```

```

4     int a=0,b=1,l;
5     while(a<n&&b<n){
6         for(l=0;l<n;++l)
7             if(s[a+l]!=s[b+l]) break;
8         if(l<n){
9             if(s[a+l]<s[b+l]) b=b+l+1;//更改大于号变为最大表示
10            else a=a+l+1;
11            if(a==b) ++b;
12        }else{
13            if(a>b) swap(a,b);
14            if(rev) return n-(b-a)+a;
15            else return a;
16        }
17    }
18    return min(a,b);
19 }

```

后缀数组

```

1  const int N=1e5+10;
2  char s[N];int sa[N],hei[N],rk[N],t1[N],t2[N],c[N];
3  inline void buildsa(char *a,int n,int m){
4      int *x=t1,*y=t2,p=0;
5      rep(i,1,n) c[x[i]=a[i]]++;
6      rep(i,1,m) c[i]+=c[i-1];
7      rpe(i,n,1) sa[c[x[i]]--]=i;
8      for(int k=1;k<=n&&p<=n;k<=1){
9          p=0;
10         rep(i,n-k+1,n) y[++p]=i;
11         rep(i,1,n) if(sa[i]>k) y[++p]=sa[i]-k;
12         rep(i,1,m) c[i]=0;
13         rep(i,1,n) c[x[y[i]]]++;
14         rep(i,1,m) c[i]+=c[i-1];
15         rpe(i,n,1) sa[c[x[y[i]]]--]=y[i];
16         swap(x,y);x[sa[1]]=1;p=2;
17         rep(i,2,n) x[sa[i]]=y[sa[i]]==y[sa[i-1]]&&y[sa[i]+k]==y[sa[i-1]+k]? p-1:p++;
18     }
19     m=p;
20     rep(i,1,n) rk[sa[i]]=i;
21     int k=0;
22     rep(i,1,n){
23         if(rk[i]==1) continue;if(k) --k;
24         while(a[i+k]==a[sa[rk[i]-1]+k]) ++k;
25         hei[rk[i]]=k;
26     }
27 }
28 int main(){
29     scanf("%s",s+1);int n=(int)strlen(s+1);
30     buildsa(s,n,233);
31     rep(i,1,n) printf("%d ",sa[i]);pts;
32     rep(i,2,n) printf("%d ",hei[i]);pts;
33     return 0;
34 }

```

Lyndon 分解

将字符串分解成若干个 LyndonWord，他们的字典序单调减，每个 Word 都是它所有后缀中最小的。

```

1  namespace lyndon{
2      vector<int> work(char *s,int n){
3          int i=1;vector<int> res;res.clear();
4          while(i<=n){
5              int j=i;
6              int k=i+1;
7              while(k<=n&&s[j]<=s[k]){
8                  if(s[j]<s[k]) j=i;
9                  else ++j;
10                 ++k;
11             }
12             while(i<=j){

```

```

13         res.emplace_back(i);
14         i+=k-j;
15     }
16 }
17 return res;
18 }
19 }

```

Lyndon 分解求所有前缀最小字典序的后缀

```

1  const int N=1e6+10;
2  namespace lyndon{
3      vector<int> work(char *s,int n){
4          int i=1;vector<int> res;res.clear();
5          while(i<=n){
6              int j=i;
7              int k=i+1;
8              while(k<=n&&s[j]<=s[k]){
9                  if(s[j]<s[k]) j=i;
10                 else ++j;
11                 ++k;
12             }
13             while(i<=j){
14                 res.emplace_back(i);
15                 i+=k-j;
16             }
17         }
18         return res;
19     }
20     vector<int> work_min_index(char *s,int n){
21         int i=1;vector<int> ans;
22         ans.resize(n+1);
23         while(i<=n){
24             ans[i]=i;
25             int j=i;
26             int k=i+1;
27             while(k<=n&&s[j]<=s[k]){
28                 if(s[j]<s[k]){
29                     ans[k]=i;
30                     j=i;
31                 }
32                 else{
33                     ans[k]=ans[j]+k-j;
34                     ++j;
35                 }
36                 ++k;
37             }
38             while(i<=j){
39                 //res.emplace_back(i);//get Lyndon
40                 i+=k-j;
41             }
42         }
43         return ans;
44     }
45 }
46 const int P=1e9+7;
47 const int base=1112;
48 char s[N];
49 inline void wk(){
50     scanf("%s",s+1);
51     int n=(int)strlen(s+1);
52     vector<int> ans=lyndon::work_min_index(s,n);
53     int ret=0;
54     //rep(i,1,n) cerr<<ans[i]<<" ";cerr<<endl;
55     rpe(i,n,1)
56         ret=(1ll*ret*base%P+ans[i])%P;
57     printf("%d\n",ret);
58 }

```

多项式

NTT 模数

```
1 NTTPrimes = {1053818881, 1051721729, 1045430273, 1012924417, 1007681537, 1004535809, 998244353, 985661441,  
  ↪ 976224257, 975175681};  
2 NTTPrimitiveRoots = {7, 6, 3, 5, 3, 3, 3, 3, 3, 17};
```

FFT

```
1 namespace FFT{  
2     const db pi=acos(-1);  
3     struct cp{  
4         db re,im;  
5         cp(db _re=0,db _im=0){re=_re;im=_im;}  
6         cp operator +(cp b){return cp(re+b.re,im+b.im);}  
7         cp operator -(cp b){return cp(re-b.re,im-b.im);}  
8         cp operator *(cp b){return cp(re*b.re-im*b.im,re*b.im+im*b.re);}  
9     };  
10    int r[N];cp c[N<<1];  
11    inline void fft(cp *a,int f,int n){  
12        rep(i,0,n-1) if(r[i]>i) swap(a[r[i]],a[i]);  
13        for(int i=1;i<n;i<=<1){  
14            cp wn(cos(pi/i),f*sin(pi/i));  
15            for(int j=0,p=(i<<1);j<n;j+=p){  
16                cp w(1,0);  
17                for(int k=0;k<i;++k,w=w*wn){  
18                    cp x=a[j+k],y=w*a[j+k+i];  
19                    a[j+k]=x+y;a[j+k+i]=x-y;  
20                }  
21            }  
22        }  
23        if(f==-1){rep(i,0,n-1) a[i].re/=n,a[i].im/=n;}  
24    }  
25    inline int mul(db *a,db *b,int n,int m){  
26        n+=m;rep(i,0,n) c[i]=cp(a[i],b[i]);  
27        int l=0;m=n;for(n=1;n<=m;n<=<1) ++l;  
28        rep(i,0,n-1) r[i]=(r[i>>1]>>1)|((i&1)<<(l-1));  
29        rep(i,m+1,n) c[i]=cp(0,0);  
30        fft(c,1,n);rep(i,0,n-1) c[i]=c[i]*c[i];  
31        fft(c,-1,n);  
32        rep(i,0,m) a[i]=c[i].im/2;  
33        return n;  
34    }  
35 }
```

NTT

```
1 namespace NTT{  
2     const int P=998244353,g=3,ig=332748118;  
3     inline int qpow(int a,int b){int q=1;while(b){if(b&1)q=1LL*q*a%P;a=1LL*a*a%P;b>>=1;}return q;}  
4     int r[N],ow[N],inv[N];  
5     inline void ntt(int *a,int f,int n){  
6         rep(i,0,n-1) if(r[i]>i) swap(a[i],a[r[i]]);  
7         for(int i=1;i<n;i<=<1){  
8             int wn=qpow(f,(P-1)/(i<<1));  
9             ow[0]=1;rep(k,1,i-1) ow[k]=1LL*ow[k-1]*wn%P;  
10            for(int j=0,p=(i<<1);j<n;j+=p){  
11                for(int k=0;k<i;++k){  
12                    int x=a[j+k],y=1LL*ow[k]*a[j+k+i]%P;  
13                    a[j+k]=(x+y)%P;a[j+k+i]=(x-P-y)%P;  
14                }  
15            }  
16        }  
17        if(f==ig){  
18            int iv=qpow(n,P-2);  
19            rep(i,0,n-1) a[i]=1LL*a[i]*iv%P;  
20        }  
21    }  
22    int tma[N],tmb[N];
```

```

23 inline int mul(int *a,int *b,int n,int m,int ci){
24     int _n=n,_m=m,l=0;m+=n;for(n=1;n<=m;n<=1) ++l;
25     rep(i,0,n-1) r[i]=(r[i]>>1)>>1|((i&1)<<(l-1));
26     rep(i,0,n-1) tma[i]=a[i];rep(i,0,n-1) tmb[i]=b[i];
27     rep(i,_n+1,n) tma[i]=0;rep(i,_m+1,n) tmb[i]=0;
28     ntt(tma,g,n);ntt(tmb,g,n);
29     while(ci){
30         if(ci&1) rep(i,0,n-1) tma[i]=1LL*tma[i]*tmb[i]%P;
31         rep(i,0,n-1) tmb[i]=1LL*tmb[i]*tmb[i]%P;
32         ci>>=1;
33     }
34     ntt(tma,ig,n);
35     rep(i,0,n-1) a[i]=tma[i];
36     return n;
37 }
38 }
39 inline void prepare(){
40     //NTT inv
41     using NTT::inv;using NTT::P;
42     inv[1]=1;rep(i,2,N-1) inv[i]=1LL*(P-P/i)*inv[P%i]%P;
43 }

```

完整的板板 by hls

```

1  /*
2     NTTPrimes = {1053818881, 1051721729, 1045430273, 1012924417, 1007681537, 1004535809, 998244353, 985661441,
   ↪ 976224257, 975175681};
3     NTTPrimitiveRoots = {7, 6, 3, 5, 3, 3, 3, 3, 17};
4  */
5  //poly start
6  namespace Poly{
7     const int N=6e5+10;
8     namespace NTT{
9         const int P=998244353,g=3,ig=332748118;
10        inline int qpow(int a,int b){int q=1;while(b){if(b&1)q=1LL*q*a%P;a=1LL*a*a%P;b>>=1;}return q;}
11        int r[N],ow[N],inv[N];
12        inline void ntt(int *a,int f,int n){
13            rep(i,0,n-1) if(r[i]>i) swap(a[i],a[r[i]]);
14            for(int i=1;i<n;i<=1){
15                int wn=qpow(f,(P-1)/(i<<1));
16                ow[0]=1;rep(k,1,i-1) ow[k]=1LL*ow[k-1]*wn%P;
17                for(int j=0,p=(i<<1);j<n;j+=p){
18                    for(int k=0;k<i;++k){
19                        int x=a[j+k],y=1LL*ow[k]*a[j+k+i]%P;
20                        a[j+k]=(x+y)%P;a[j+k+i]=(x-P-y)%P;
21                    }
22                }
23            }
24            if(f==ig){
25                int iv=qpow(n,P-2);
26                rep(i,0,n-1) a[i]=1LL*a[i]*iv%P;
27            }
28        }
29        int tma[N],tmb[N];
30        inline int mul(int *a,int *b,int n,int m,int ci){
31            int _n=n,_m=m,l=0;m+=n;for(n=1;n<=m;n<=1) ++l;
32            rep(i,0,n-1) r[i]=(r[i]>>1)>>1|((i&1)<<(l-1));
33            rep(i,0,n-1) tma[i]=a[i];rep(i,0,n-1) tmb[i]=b[i];
34            rep(i,_n+1,n) tma[i]=0;rep(i,_m+1,n) tmb[i]=0;
35            ntt(tma,g,n);ntt(tmb,g,n);
36            while(ci){
37                if(ci&1) rep(i,0,n-1) tma[i]=1LL*tma[i]*tmb[i]%P;
38                rep(i,0,n-1) tmb[i]=1LL*tmb[i]*tmb[i]%P;
39                ci>>=1;
40            }
41            ntt(tma,ig,n);
42            rep(i,0,n-1) a[i]=tma[i];
43            return n;
44        }
45    }
46    namespace FFT{

```



```

47     const db pi=acos(-1);
48     struct cp{
49         db re,im;
50         cp(db _re=0,db _im=0){re=_re;im=_im;}
51         cp operator +(cp b){return cp(re+b.re,im+b.im);}
52         cp operator -(cp b){return cp(re-b.re,im-b.im);}
53         cp operator *(cp b){return cp(re*b.re-im*b.im,re*b.im+im*b.re);}
54     };
55     int r[N];cp c[N<<1];
56     inline void fft(cp *a,int f,int n){
57         rep(i,0,n-1) if(r[i]>i) swap(a[r[i]],a[i]);
58         for(int i=1;i<n;i<=<1){
59             cp wn(cos(pi/i),f*sin(pi/i));
60             for(int j=0,p=(i<<1);j<n;j+=p){
61                 cp w(1,0);
62                 for(int k=0;k<i;++k,w=w*wn){
63                     cp x=a[j+k],y=w*a[j+k+i];
64                     a[j+k]=x+y;a[j+k+i]=x-y;
65                 }
66             }
67         }
68         if(f==1){rep(i,0,n-1) a[i].re/=n,a[i].im/=n;}
69     }
70     inline int mul(db *a,db *b,int n,int m){
71         n+=m;rep(i,0,n) c[i]=cp(a[i],b[i]);
72         int l=0;m=n;for(n=1;n<=m;n<=<1) ++l;
73         rep(i,0,n-1) r[i]=(r[i]>>1)>>1|((i&1)<<(l-1));
74         rep(i,m+1,n) c[i]=cp(0,0);
75         fft(c,1,n);rep(i,0,n-1) c[i]=c[i]*c[i];
76         fft(c,-1,n);
77         rep(i,0,m) a[i]=c[i].im/2;
78         return n;
79     }
80 }
81 using namespace NTT;
82 ll w,a;
83 struct node{
84     ll x,y;
85     node friend operator *(node x,node y){
86         node z;
87         z.x=(x.x*y.x%P+x.y*y.y%P*w%P)%P;
88         z.y=(x.x*y.y%P+x.y*y.x%P)%P;
89         return z;
90     }
91 }u,v;
92 inline node Cqpow(node a,ll b){
93     node q;q.x=1;q.y=0;
94     while(b){if(b&1) q=q*a;a=a*a;b>>=1;}
95     return q;
96 }
97 inline ll cipolla(int n,int P){
98     n%=P;rand(0x20010412);
99     if(P==2) return 1;
100    if(qpow(n,(P-1)/2)==P-1) return -1;
101    while(1){
102        a=rand()%P;
103        w=(a*a-n+P)%P;
104        if(qpow(w%P,(P-1)/2)==P-1) break;
105    }
106    u.x=a,u.y=1;
107    u=Cqpow(u,(P+1)/2);
108    ll fir=u.x,sec=P-u.x;
109    if(fir>sec)swap(fir,sec);
110    return fir;
111 }
112 inline void derivative(int *a,int *b,int n){
113     rpe(i,n-2,0) b[i]=1LL*a[i+1]*(i+1)%P;
114     b[n-1]=0;
115 }
116 inline void integral(int *a,int *b,int n){
117     rpe(i,n-1,1) b[i]=1LL*a[i-1]*inv[i]%P;

```

```

118     b[0]=0;
119 }
120 inline void differential(int *a,int *b,int n){
121     rep(i,1,n-1) b[i]=(a[i]+P-a[i-1])%P;
122     b[0]=0;
123 }
124 int tf[N],tg[N];
125 inline void inverse(int *f,int *g,int n){
126     if(n==1){
127         g[0]=qpow(f[0],P-2);return;
128     }
129     inverse(f,g,n>>1);
130     rep(i,0,n-1) tf[i]=f[i],tg[i]=g[i];
131     int tmp=mul(tf,tg,n-1,n-1,2);
132     rep(i,0,n-1) g[i]=((-tf[i]+2LL*tmp)%P+P)%P;
133     rep(i,0,tmp) tf[i]=tg[i]=0;
134 }
135 int ta[N],tb[N];
136 inline void sqrt(int *a,int *b,int n){
137     if(n==1){
138         b[0]=cipolla(a[0],P);//debug(b[0]);debug(1LL*b[0]*b[0]%P);
139         return;
140     }
141     sqrt(a,b,n>>1);rep(i,n,(n<<1)) b[i]=0;
142     inverse(b,tb,n);
143     rep(i,0,n-1) ta[i]=a[i];
144     mul(ta,tb,n-1,n-1,1);//debug(ta[0]);debug(b[0]);
145     rep(i,0,n-1) b[i]=1LL*(b[i]+ta[i])%P*inv[2]%P;
146     rep(i,0,n<<1) ta[i]=tb[i]=0;
147 }
148 inline void ln(int *a,int *b,int n){
149     inverse(a,ta,n);
150     derivative(a,b,n);
151     mul(b,ta,n,n,1);
152     integral(b,b,n);
153 }
154 inline void exp(int *a,int *b,int n){
155     if(n==1){
156         b[0]=1;return;
157     }
158     exp(a,b,n>>1);
159     ln(b,tb,n);rep(i,0,n-1) ta[i]=a[i];++ta[0];
160     rep(i,0,n-1) ta[i]-=tb[i];
161     mul(ta,b,n,n,1);rep(i,0,n-1) b[i]=ta[i];
162     rep(i,0,n<<1) ta[i]=tb[i]=0;
163 }
164 }
165 //
166 using namespace Poly;
167 inline void prepare(){
168     //NTT
169     using NTT::inv;using NTT::P;
170     inv[1]=1;rep(i,2,N-1) inv[i]=1LL*(P-P/i)*inv[P%i]%P;
171 }
172 int n,A[N],B[N];
173 int main(){
174     prepare();
175     n=read();rep(i,0,n) A[i]=read();
176     int len=1;for(;len<=n;len<<=1);
177     /*
178     sqrt(A,B,len);
179     rep(i,0,n) printf("%d ",B[i]);pts;
180     // debug(1LL*B[0]*B[0]%P);
181     NTT::mul(B,B,n,n,1);
182     rep(i,0,n) printf("%d ",B[i]);pts;
183     */ //sqrt 1e5 uoj 333ms
184     /*
185     ln(A,B,len);
186     rep(i,0,n) printf("%d ",B[i]);pts;
187     exp(B,A,len);
188     rep(i,0,n) printf("%d ",A[i]);pts;

```

```

189 */ //ln&exp 1e5 uoj 766ms
190    //ln 222ms
191    //exp 568ms
192    return 0;
193 }

```

容斥与反演

莫比乌斯反演

2.2.1 莫比乌斯反演定理若 $F(n)$ 和 $f(n)$ 是定义在非负整数集合上的两个函数，并且满足条件 $F(n) = \sum_{d|n} f(d)$ 。那么我们得到结论：

$$f(n) = \sum_{d|n} \mu(d) F\left(\frac{n}{d}\right)$$

证明：

$$\sum_{d|n} \mu(d) F\left(\frac{n}{d}\right) = \sum_{d|n} \mu(d) \sum_{k|\frac{n}{d}} f(k) = \sum_{k|n} f(k) \sum_{d|\frac{n}{k}} \mu(d) = f(n)$$

由这个定理我们可以得到莫比乌斯函数的另外一个性质：对于 $\forall n \in \mathbb{N}_+$

$$\sum_{d|n} \frac{\mu(d)}{d} = \frac{\varphi(n)}{n}$$

证明：

$$\text{引理：} n = \sum_{d|n} \varphi(d)$$

证明：

对于 n 的每个因数 d ，我们取出 $[1, d]$ 内的 $\varphi(d)$ 个与 n 互素的数记做集合 $A = \{a_1, a_2, \dots, a_{\varphi(d)}\}$ ，将集合 A 内的元素对应到集合 $B = \{a_1 \frac{n}{d}, a_2 \frac{n}{d}, \dots, a_{\varphi(d)} \frac{n}{d}\}$ 。显然 $\gcd(a_i \frac{n}{d}, n) = \frac{n}{d}$ 。由于 d 枚举了所有 n 的因数，所以 $\frac{n}{d}$ 也是。则集合 B 内是 $[1, n]$ 内所有的数。故原命题成立。

有了这个引理，我们将莫比乌斯反演定理中的 $F(n) = n, f(n) = \varphi(n)$ 。

$$\varphi(n) = \sum_{d|n} \frac{n}{d} \mu(d)$$

$$\frac{\varphi(n)}{n} = \sum_{d|n} \frac{\mu(d)}{d}$$

第二形式：

$$F(n) = \sum_{n|d} f(d) \Rightarrow f(n) = \sum_{n|d} \mu\left(\frac{d}{n}\right) F(d)$$

证明：令 $k = \frac{d}{n}$ ，那么

$$\sum_{k=1}^{+\infty} \mu(k) F(nk) = \sum_{k=1}^{+\infty} \mu(k) \sum_{nk|t} f(t) = \sum_{n|t} f(t) \sum_{k|\frac{t}{n}} \mu(k) = f(n)$$

2.2.2 杜教筛（普适）若 $f(n)$ 是一个积性函数，求 $f(n)$ 的前缀 $S(n)$ 。即 $S(n) = \sum_{i=1}^n f(i)$ 。

狄利克雷卷积

对于数论函数 $g(n), f(n)$ ，其狄利克雷卷积 $h(n)$ 也是一个数论函数

$$h(n) = \sum_{d|n} g(d) f\left(\frac{n}{d}\right)$$

我们找到另一个积性函数 $g(n)$ ，让 $f(n)$ 和 $g(n)$ 做一个卷积

$$(g * f)(n) = \sum_{d|n} g(d) f\left(\frac{n}{d}\right)$$

求卷积的前缀

$$\sum_{i=1}^n (g * f)(i) = \sum_{i=1}^n \sum_{d|i} g(d) f\left(\frac{i}{d}\right)$$

提出右式的 d

$$\begin{aligned} \Rightarrow \sum_{i=1}^n (g * f)(i) &= \sum_{d=1}^n g(d) \sum_{d|i} f\left(\frac{i}{d}\right) \\ &= \sum_{d=1}^n g(d) \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} f(i) \\ &= \sum_{d=1}^n g(d) S\left(\left\lfloor \frac{n}{d} \right\rfloor\right) \end{aligned}$$

容易得到这个式子

$$g(1)S(n) = \sum_{i=1}^n g(i)S\left(\left\lfloor \frac{n}{i} \right\rfloor\right) - \sum_{i=2}^n g(i)S\left(\left\lfloor \frac{n}{i} \right\rfloor\right)$$

其实就是

$$g(1)S(n) = \sum_{i=1}^n (g * f)(i) - \sum_{i=2}^n g(i)S\left(\left\lfloor \frac{n}{i} \right\rfloor\right)$$

我们发现如果狄利克雷卷积前缀很好算的话，积性函数的前缀也可以分块递归来算了。

举几个例子：

$$1. \text{ 求 } S(n) = \sum_{i=1}^n \mu(n)$$

上述式子

$$g(1)S(n) = \sum_{i=1}^n (g * \mu)(i) - \sum_{i=2}^n g(i)S\left(\left\lfloor \frac{n}{i} \right\rfloor\right)$$

考虑到 $\sum_{d|n} \mu(d) = [n=1]$ ，又由于 $(g * \mu)(n) = \sum_{d|n} g(d) \mu\left(\frac{n}{d}\right)$ 。我们考虑让 $g(n) = 1(n)$ ，那么 $(1 * \mu)(n) = \sum_{d|n} 1 \cdot \mu(d) = [n=1]$

。显然这个卷积的前缀为 $\sum_{i=1}^n (g * \mu)(i) = 1(n)$ 。

故对于 μ

$$S(n) = 1 - \sum_{i=2}^n S\left(\left\lfloor \frac{n}{i} \right\rfloor\right)$$

$$2. \text{ 求 } S(n) = \sum_{i=1}^n \varphi(n)$$

上述式子

$$g(1)S(n) = \sum_{i=1}^n (g * \varphi)(i) - \sum_{i=2}^n g(i)S\left(\left\lfloor \frac{n}{i} \right\rfloor\right)$$

考虑到 $\sum_{d|n} \varphi(d) = n$ ，又由于 $(g * \varphi)(n) = \sum_{d|n} g(d) \varphi\left(\frac{n}{d}\right)$ 。我们考虑让 $g(n) = 1(n)$ ，那么 $(1 * \varphi)(n) = \sum_{d|n} 1 \cdot \varphi(d) = n$ 。显

然这个卷积的前缀为 $\sum_{i=1}^n (g * \varphi)(i) = \frac{n(n+1)}{2}$ 。

故对于 φ

$$S(n) = \frac{n(n+1)}{2} - \sum_{i=2}^n S\left(\left\lfloor \frac{n}{i} \right\rfloor\right)$$

```

1  LL mu[N+5], phi[N+5], n;
2  struct num {
3      LL ans1, ans2;
4      num() {}
5      num(LL _ans1, LL _ans2) {ans1 = _ans1, ans2 = _ans2; }
6  }ans;
7  map<LL, num>mp;
8
9  int prime[N+5], isprime[N+5], tot;
10 void get_pre() {
11     memset(isprime, 1, sizeof(isprime)); isprime[1] = 0, mu[1] = phi[1] = 1;
12     for (int i = 2; i <= N; i++) {
13         if (isprime[i]) prime[++tot] = i, mu[i] = -1, phi[i] = i-1;
14         for (int j = 1; j <= tot && i*prime[j] <= N; j++) {
15             isprime[i*prime[j]] = 0;
16             if (i%prime[j]) mu[i*prime[j]] = -mu[i], phi[i*prime[j]] = phi[i]*(prime[j]-1);
17             else {mu[i*prime[j]] = 0, phi[i*prime[j]] = phi[i]*prime[j]; break; }
18         }
19         mu[i] += mu[i-1], phi[i] += phi[i-1];
20     }
21 }
22 num Less (const num &a, const num &b) {num ans; ans.ans1 = a.ans1 - b.ans1, ans.ans2 = a.ans2-b.ans2; return ans; }
23 num Times (const num &a, const LL &x) {num ans; ans.ans1 = a.ans1*x , ans.ans2 = a.ans2*x; return ans; }
24 num cal(LL x) {
25     if (x <= N) return num(phi[x], mu[x]);
26     if (mp.count(x)) return mp[x];
27     num ans = num(x*(x+1)/2, 1);
28     for (LL i = 2, last; i <= x; i = last+1) {
29         last = x/(x/i); ans = Less(ans, Times(cal(x/i), (last-i+1)));
30     }
31     return mp[x] = ans;
32 }
33 void work() {
34     read(n); ans = cal(n);
35     write(ans.ans1), putchar(' ');
36     if (ans.ans2 < 0) putchar('-'); writeln(-ans.ans2);
37     else writeln(ans.ans2);
38 }

```

快速莫比乌斯变换（反演）与子集卷积

莫比乌斯变换（反演）

问题提出

若 h, f, g 为下标为集合的函数，我们定义

$$h = f * g$$

表示

$$h(S) = \sum_{L \subseteq S} \sum_{R \subseteq S} [L \cup R = S] f(L) \times g(R)$$

容易发现，对于这个问题，我们可以用 $O((2^n)^2)$ 的枚举 L, R 来计算。

然而这样复杂度较高，我们考虑类比多项式卷积的过程，可以求出 f, g 的点值，直接相乘得到 h 的点值然后再插回去。

值得注意的是为了便于表述以及规范表达，快速莫比乌斯变换就相当于点值，快速莫比乌斯反演就相当于插值。

算法原理

- 我们定义 f 的莫比乌斯变换为 F ，其中 $F(S) = \sum_{X \subseteq S} f(X)$ ；由这个定义，我们可以推出 F 莫比乌斯反演 f 为 $f(S) = \sum_{X \subseteq S} (-1)^{|S|-|X|} F(X)$ 。对于莫比乌斯反演的证明，可以带入莫比乌斯变换的式子或容斥来证。
- 我们对于一个函数 f, g, h ，记它的点值式为 F, G, H 。我们将问题提出中的卷积式两边同时做莫比乌斯变换，得到

$$\begin{aligned}
 H(S) &= \sum_{L \subseteq S} \sum_{R \subseteq S} [L \cup R \subseteq S] f(L) \times g(R) \\
 &= \sum_{L \subseteq S} \sum_{R \subseteq S} f(L) \times g(R) \\
 &= \left(\sum_{L \subseteq S} f(L) \right) \times \left(\sum_{R \subseteq S} g(R) \right) \\
 &= F(S) \times G(S)
 \end{aligned}$$

至此算法原理及过程已经完全结束。似乎我们可以用 $O(3^n)$ 枚举子集来变换和反演，实际上我们可以让复杂度更优。

算法实现

- 设 $\hat{f}_S^{(i)}$ 表示 $\sum_{T \subseteq S} [(S-T) \subseteq \{1, 2, \dots, i\}] f_T$
- 易得初始状态: $\hat{f}_S^{(0)} = f_S$
- 对于每一个不包含 $\{i\}$ 的集合 S ，可知 $\hat{f}_S^{(i)} = \hat{f}_S^{(i-1)}$ (因为 S 并没有 i 这位)， $\hat{f}_{S \cup \{i\}}^{(i)} = \hat{f}_S^{(i-1)} + \hat{f}_{S \cup \{i\}}^{(i-1)}$ (前者的 T 没有包含 $\{i\}$ ，而后者的 T 必须包含了 $\{i\}$)。
- 显然，递推了 n 轮之后， \hat{f}_S^n 就是所求的变换了。

用高维前缀和可以做到 $O(n \times 2^n)$ 的递推，求出点值和插值。

```

1 void FMT(int *A, int o) { // o 为识别因子
2     for (int i = 1; i < ST; i <= 1) // ST-1 表示全集
3         for (int j = 0; j < ST; j++)
4             if (i&j) (A[j] += A[j^i]*o) %= mod;
5 }

```

例题 - [HAOI 2015] 按位或

子集卷积

FWT：“你刚才说的那个玩意我也能做啊，要你何用？”

FMT：“.....”

问题提出

若 h, f, g 为下标为集合的函数，我们定义

$$h = f * g$$

表示

$$h(S) = \sum_{X \subseteq S} f(X) \times g(S - X)$$

算法实现

回顾刚刚的集合并卷积，子集卷积的条件比集合并卷积更苛刻，即 L 和 R 的集合应该不相交。

我们可以在卷积时多加一维，维护集合的大小，如 $f_{i,S}$ 表示集合中有 i 个元素，集合表示为 S 。显然，当 i 和 S 的真实元素个数符合时才是对的。记数组 $\text{cnt}[S]$ 表示集合 S 的模。初始时，我们只把 $f_{\text{cnt}[S], S}$ 的值赋成原来的 $f(S)$ (g 同理)，然后每一维做一遍 FMT，点值相乘时这么写: $h_{i,S} = \sum_{j=0}^i f_{j,S} \times g_{i-j,S}$ 。最后扫一遍把不符合实际情况的状态赋成 0 即可。

```

1  for (int i = 0; i <= n; i++) FMT(g[i], 1);
2  for (int i = 0; i <= n; i++) FMT(f[i], 1);
3  for (int i = 1; i <= n; i++) {
4      for (int j = 0; j <= i; j++)
5          for (int k = 0; k < ST; k++)
6              (h[i][k] += 1ll*f[j][k]*g[i-j][k]%mod) %= mod;
7      FMT(h[i], -1);
8      for (int k = 0; k < ST; k++) if (cnt[k] != i) h[i][k] = 0;
9      if (i != n) FMT(h[i], 1);
10 }

```

例题 - [WC 2018] 州区划分

二项式反演

内容

对于函数 f, g , $\forall p \in \mathbb{N}$ 若 $\forall n \geq p$, 满足

$$f(n) = \sum_{k=p}^n \binom{n}{k} g(k)$$

那么 $\forall n \geq p$

$$g(n) = \sum_{k=p}^n (-1)^{n-k} \binom{n}{k} f(k)$$

证明

为了方便表达, 我们取 $p = 0$, 实质和取 $p \in \mathbb{N}$ 的证明方法是一样的。

$$\begin{aligned}
 g(n) &= \sum_{k=0}^n (-1)^{n-k} \binom{n}{k} f(k) \\
 &= \sum_{k=0}^n (-1)^{n-k} \binom{n}{k} \sum_{i=0}^k \binom{k}{i} g(i) \\
 &= \sum_{k=0}^n \sum_{i=0}^k (-1)^{n-k} \binom{n}{k} \binom{k}{i} g(i) \\
 &= \sum_{i=0}^n \left(\sum_{k=i}^n (-1)^{n-k} \binom{n}{k} \binom{k}{i} \right) g(i) \\
 &= \sum_{i=0}^n \left(\sum_{k=i}^n (-1)^{n-k} \binom{n}{i} \binom{n-i}{k-i} \right) g(i) \\
 &= \sum_{i=0}^n \left(\binom{n}{i} \sum_{k=i}^n (-1)^{n-k} \binom{n-i}{n-k} \right) g(i) \\
 &= \sum_{i=0}^n \left(\binom{n}{i} (1-1)^{n-i} \right) g(i) \\
 &= g(n)
 \end{aligned}$$

故成立。

应用举例

推导错排公式

我们记 $f(n)$ 为 n 个数字任意放的方案数, $g(n)$ 为 n 个数没有一个放在自己位置上的方案数。

枚举不在自己位置上的个数，容易得到

$$f(n) = \sum_{i=0}^n \binom{n}{i} g(i)$$

那么

$$\begin{aligned} g(n) &= \sum_{i=0}^n (-1)^{n-i} \binom{n}{i} f(i) \\ &= \sum_{i=0}^n (-1)^i \binom{n}{i} f(n-i) \end{aligned}$$

注意到 $f(x) = x!$ ，那么

$$\begin{aligned} g(n) &= \sum_{i=0}^n (-1)^i \frac{n!}{i!(n-i)!} (n-i)! \\ &= n! \sum_{i=0}^n (-1)^i \frac{1}{i!} \end{aligned}$$

棋盘染色

有个 $1 \times n$ 的格子， m 种颜色 ($m \geq 2$)，要求相邻格子的颜色不相同且每种颜色都要用到，求染色方案数。

我们记 $f(n)$ 为至多用到 n 种颜色的方案数， $g(n)$ 为恰用到 n 种颜色的方案数。

那么

$$\begin{aligned} f(m) &= \sum_{i=2}^m \binom{m}{i} g(i) \\ \Rightarrow g(m) &= \sum_{i=2}^m (-1)^i \binom{m}{i} f(m-i) \end{aligned}$$

注意到 $f(x) = x \times (x-1)^{n-1}$ 。那么就可以带入直接算了。

另一形式

$$a_k = \sum_{i=k}^n \binom{i}{k} b_i \Rightarrow b_k = \sum_{i=k}^n (-1)^{i-k} \binom{i}{k} a_i$$

证明：

$$\begin{aligned}
& \sum_{i=k}^n (-1)^{i-k} \binom{i}{k} a_i \\
&= \sum_{i=k}^n (-1)^{i-k} \binom{i}{k} \sum_{j=k}^n \binom{j}{i} b_i \\
&= \sum_{i=k}^n \sum_{j=k}^n (-1)^{i-k} \binom{i}{k} \binom{j}{i} b_i \\
&= \sum_{i=k}^n \sum_{j=k}^n (-1)^{i-k} \binom{j}{i} \binom{i}{k} b_i \\
&= \sum_{j=k}^n \sum_{i=k}^j (-1)^{i-k} \binom{j}{k} \binom{j-k}{i-k} b_i \\
&= \sum_{j=k}^n \binom{j}{k} \sum_{i=k}^j (-1)^{i-k} \binom{j-k}{i-k} b_i \\
&= \sum_{j=k}^n \binom{j}{k} (1-1)^{j-k} b_i \\
&= b_k
\end{aligned}$$

例题 - [BZOJ 2839] 集合计数 - [BZOJ 3622] 已经没有什么好害怕的了

斯特林反演

第一类斯特林数

$\left[\begin{smallmatrix} n \\ m \end{smallmatrix} \right]$ 表示将 n 个元素排成 m 个轮换的方法数。

$$\text{递推公式: } \left[\begin{smallmatrix} n \\ m \end{smallmatrix} \right] = \left[\begin{smallmatrix} n-1 \\ m-1 \end{smallmatrix} \right] + (n-1) \left[\begin{smallmatrix} n-1 \\ m \end{smallmatrix} \right]$$

含义是考虑第 n 个元素的放法：要么新开一个轮换，要么就放在前 $n-1$ 个元素的左边。

第二类斯特林数

$\left\{ \begin{smallmatrix} n \\ m \end{smallmatrix} \right\}$ 表示将 n 个元素划分成 m 个非空子集的方法数。

$$\text{递推公式: } \left\{ \begin{smallmatrix} n \\ m \end{smallmatrix} \right\} = \left\{ \begin{smallmatrix} n-1 \\ m-1 \end{smallmatrix} \right\} + m \left\{ \begin{smallmatrix} n-1 \\ m \end{smallmatrix} \right\}$$

含义是考虑第 n 个元素的放法：要么新开一个组，要么就放在前 m 组内。

$$\text{通项公式 (容斥式): } \left\{ \begin{smallmatrix} n \\ m \end{smallmatrix} \right\} = \frac{1}{m!} \sum_{k=0}^m (-1)^k \binom{m}{k} (m-k)^n$$

有关通项公式的证明及运用可以参考多项式类数学相关这篇文章。

例题 - [Codeforces 932E] Team Work - [Codeforces 961G] Partitions - [TJOI 2016&HEOI 2016] 求和

反演公式

$$f(x) = \sum_{i=0}^x \left\{ \begin{smallmatrix} x \\ i \end{smallmatrix} \right\} g(i) \Leftrightarrow g(x) = \sum_{i=0}^x (-1)^{x-i} \left[\begin{smallmatrix} x \\ i \end{smallmatrix} \right] f(i)$$

$$f(x) = \sum_{i=0}^x \left[\begin{smallmatrix} x \\ i \end{smallmatrix} \right] g(i) \Leftrightarrow g(x) = \sum_{i=0}^x (-1)^{x-i} \left\{ \begin{smallmatrix} x \\ i \end{smallmatrix} \right\} f(i)$$

例题 - 给出 n 个点的一张简单图，问有多少个边的子集，满足保留子集中的边后，该图连通。（摘自Sdchr） - 大概就是枚举连通块的个数，然后块内随便连，然后容斥就好。 - 考虑如何求容斥系数 $f(i)$ 。设实际上是 x 个连通块的方案，它应该被计算 $[x = 1]$ 次，实际上在所有更仔细的分块中被统计，所以 -

$$[x = 1] = \sum_{i=1}^x \left\{ \begin{matrix} x \\ i \end{matrix} \right\} f(i)$$

- 由斯特林反演 -

$$\begin{aligned} f(x) &= \sum_{i=1}^x (-1)^{x-i} \left[\begin{matrix} x \\ i \end{matrix} \right] [i = 1] \\ &= (-1)^{x-1} \left[\begin{matrix} x \\ 1 \end{matrix} \right] \\ &= (-1)^{x-1} (x-1)! \end{aligned}$$

- [BZOJ 4671] 异或图

最值反演（min-max 容斥）

公式

记 $\max(S)$ 为集合 S 中的最大值， $\min(S)$ 为集合 S 中的最小值， $|S|$ 为集合 S 的元素数量，那么以下两个等式成立

$$\max(S) = \sum_{T \subseteq S} (-1)^{|T|+1} \min(T)$$

$$\min(S) = \sum_{T \subseteq S} (-1)^{|T|+1} \max(T)$$

证明

这里只证明第一个等式好了，后边的可以自行推出。

其实只需要证明一件事，就是除了 $\min(T) = \max(S)$ 的那个值，其他的 \min 值都被消掉了就可以了（这里说明一下，我们假定集合中的元素两两相异）

先来说明 $\max(S)$ 的系数为什么是 1，假设中 S 最大的元素是 a ，那么我们会发现只有 $\min(\{a\}) = \max(S)$ 所以 $\max(S)$ 的系数必须是 1。

然后再说明为什么别的 \min 都被消掉了，假设某个元素 b 的排名是 k ，那么 $\min(T) = b$ 当且仅当我们选出的集合是后 $n - k$ 个的元素构成的集合的子集然后并上 $\{b\}$ 得到的，我们会发现显然这样的集合有 2^{n-k} 种，而显然这其中恰有 2^{n-k-1} 中是有奇数个元素的，恰有 2^{n-k-1} 种是有偶数个元素的，两两相消自然就成了 0 了，当然上述等式在 $k = n$ 的时候不成立，但是此时剩下的刚好是最大值，所以证明完毕。

拉格朗日插值法

简介

给定 $n + 1$ 个横坐标不相同的点，可以唯一确定一个 n 次的多项式。最直观的求多项式的做法就是列方程求解。但是这样需要 $O(n^3)$ 的时间来计算。而拉格朗日插值法则通过构造的方法，得到了一个经过 $n + 1$ 个点的 n 次多项式。具体的过程是这样的，假设现在我们得到了 $n + 1$ 个点：

$$(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$$

设拉格朗日基本多项式为

$$\ell_j(x) = \prod_{i=0, i \neq j}^n \frac{x - x_i}{x_j - x_i}$$

这个基本多项式构造十分巧妙，因为注意到 $\ell_j(x_j) = 1$ ，并且 $\ell_j(x_i) = 0, \forall i \neq j$ 。那么，接着构造出这个 n 次多项式

$$P(x) = \sum_{i=0}^n y_i \ell_i(x)$$

根据基本多项式的性质，我们可以知道 $P(x_i) = y_i$ ，也就是经过了这 $n+1$ 个点。通过简单的多项式乘法和多项式除法就可以在 $O(n^2)$ 的时间求出这个多项式的系数表达。

求解

已知 n 次多项式 $f(n)$ 上的 $n+1$ 个点 $(x_i, y_i), i \in [0, n]$ ，求 $f(x_i)$

```
1 int lagrange(int n, int *x, int *y, int xi) {
2     int ans = 0;
3     for (int i = 0; i <= n; i++) {
4         int s1 = 1, s2 = 1;
5         for (int j = 0; j <= n; j++)
6             if (i != j) {
7                 s1 = 1ll*s1*(xi-x[j])%mod;
8                 s2 = 1ll*s2*(x[i]-x[j])%mod;
9             }
10        ans = (1ll*ans+1ll*y[i]*s1%mod*quick_pow(s2, mod-2)%mod)%mod;
11    }
12    return (ans+mod)%mod;
13 }
```

如果 x 的取值是连续一段的话，我们可以做到 $O(n)$ 求解。假设 $\forall i < j, x_i < x_j$ （具体公式推导的话，如果你有兴趣可以参看之后的内容。因为比较显然，这里不再讲解。）

```
1 int lagrange(int n, int *x, int *y, int xi) {
2     int ans = 0;
3     s1[0] = (xi-x[0])%mod, s2[n+1] = 1;
4     for (int i = 1; i <= n; i++) s1[i] = 1ll*s1[i-1]*(xi-x[i])%mod;
5     for (int i = n; i >= 0; i--) s2[i] = 1ll*s2[i+1]*(xi-x[i])%mod;
6     ifac[0] = ifac[1] = 1;
7     for (int i = 2; i <= n; i++) ifac[i] = -1ll*mod/i*ifac[mod%i]%mod;
8     for (int i = 2; i <= n; i++) ifac[i] = 1ll*ifac[i]*ifac[i-1]%mod;
9     for (int i = 0; i <= n; i++)
10        (ans += 1ll*y[i]*(i == 0 ? 1 : s1[i-1])%mod*s2[i+1]%mod
11         *ifac[i]%mod*((n-i)&1 ? -1 : 1)*ifac[n-i]%mod) %= mod;
12    return (ans+mod)%mod;
13 }
```

例题 - [BZOJ 2655]calc

自然数的幂的前缀和

问题提出

给定的 n 和 k ，求

$$\sum_{i=1}^n i^k$$

通常 n 比较大，而 k 只有几千或者几万。

问题解决

我们可以知道，对于上述式子，推导公式一定是 $k+1$ 次多项式。对于证明的话，我们可以参考 [riteme](#) 的介绍。

考虑使用拉格朗日插值法来获得答案多项式。

首先如果我们得知了 $n = 0, 1, \dots, k+1$ 处的答案 $f(n)$ ，那么给定的 n 处的答案可以写成

$$\begin{aligned}
f(n) &= \sum_{i=0}^{k+1} f(i) \frac{(n-0)(n-1) \cdots [n-(i-1)][(n-(i+1)) \cdots [n-(k+1)]}{(i-0)(i-1) \cdots [i-(i-1)][(i-(i+1)) \cdots [i-(k+1)]} \\
&= \sum_{i=0}^{k+1} f(i) \frac{\prod_{j=0}^{i-1} (n-j) \prod_{j=i+1}^{k+1} (n-j)}{i!(-1)^{k-i+1}(k+1-i)!} \\
&= \sum_{i=0}^{k+1} (-1)^{k-i+1} f(i) \frac{\prod_{j=0}^{i-1} (n-j) \prod_{j=i+1}^{k+1} (n-j)}{i!(k+1-i)!}
\end{aligned}$$

注意到后面的分式中，分子是一个前缀积乘以一个后缀积，而分母是两个阶乘。这些都可以在 $O(k)$ 的时间内求出。现在剩下的问题就是如何求出 $f(0), f(1), \dots, f(k+1)$ 了。由于 $g(x) = x^k$ 是个完全积性函数，所以可以通过欧拉筛法求出 g 函数前面的一些值。具体的就是对于质数采取直接快速幂，合数则拆出任意一个因子来算，通常是欧拉筛法中可以顺便求得的最小质因子。根据素数定理，素数大约有 $O(\frac{k}{\ln k})$ 个。每次快速幂需要花费 $O(\log k)$ 的时间，因此总的时间复杂度可以估计为 $O(k)$ ，是一个非常优秀的算法。上面的方法具有通用性，只要我们可以快速的求出某个 k 次多项式的前 $k+1$ 个值，那么剩下的部分可以使用拉格朗日插值法在 $O(k)$ 的时间内完成计算。

代码实现

```

1  int lagrange(int k, int *f, int xi) { // k+2 个点 (i, f[i]), 0 <= i <= k+1
2      int ans = 0; ++k;
3      s1[0] = xi, s2[k+1] = 1;
4      for (int i = 1; i <= k; i++) s1[i] = 1ll*s1[i-1]*(xi-i)%mod;
5      for (int i = k; i >= 0; i--) s2[i] = 1ll*s2[i+1]*(xi-i)%mod;
6      for (int i = 0; i <= k; i++)
7          (ans += 1ll*f[i]*(i == 0 ? 1 : s1[i-1])%mod*s2[i+1]%mod
8              *ifac[i]%mod*((k-i)&1 ? -1 : 1)*ifac[k-i]%mod) %= mod;
9      return (ans+mod)%mod;
10 }
11 void pre() { // 预处理出阶乘逆元、插值的 k+2 个点
12     f[1] = ifac[0] = ifac[1] = 1;
13     for (int i = 2; i <= k+1; i++) ifac[i] = -1ll*mod/i*ifac[mod%i]%mod;
14     for (int i = 2; i <= k+1; i++) ifac[i] = 1ll*ifac[i-1]*ifac[i]%mod;
15     memset(isprime, 1, sizeof(isprime));
16     for (int i = 2; i <= k+1; i++) {
17         if (isprime[i]) prime[++tot] = i, f[i] = quick_pow(i, k);
18         for (int j = 1; j <= tot && prime[j]*i <= k+1; j++) {
19             isprime[i*prime[j]] = 0;
20             f[i*prime[j]] = 1ll*f[i]*f[prime[j]]%mod;
21             if (i%prime[j] == 0) break;
22         }
23     }
24     for (int i = 1; i <= k+1; i++) f[i] = (f[i]+f[i-1])%mod;
25 }
26 void work() {
27     scanf("%d", &k); pre();
28     while (~scanf("%d", &n)) {
29         if (n <= k+1) printf("%d\n", f[n]);
30         else printf("%d\n", lagrange(k, f, n));
31     }
32 }

```

例题 - [JLOI 2016] 成绩比较