Standard Code Library

Your TeamName

Your School

April 2, 2022

Contents

Data Structure

Xor MST

```
struct Trie{
    int son[2][200000*30+10],tot;
    void Insert(int a){
        int now=0,id;
        for(int i=30;i>=0;i--){
            id=(a>>i)&1;
            if(!son[id][now])son[id][now]=++tot;
            now=son[id][now];
        }
    }
    int Find(int r1,int r2,int b){
        if(b<0) return 0;
        int a1=-1,a2=-1;
        if(son[0][r1]&&son[0][r2]) a1=Find(son[0][r1],son[0][r2],b-1);
        if(son[1][r1]&&son[1][r2]) a2=Find(son[1][r1],son[1][r2],b-1);
        if(~a1&&~a2) return min(a1,a2);
        if(~a1) return a1;if(~a2) return a2;
        if(son[1][r1]&&son[0][r2]) a1=Find(son[1][r1],son[0][r2],b-1)+(1<<b);
        if(son[0][r1]&&son[1][r2]) a2=Find(son[0][r1],son[1][r2],b-1)+(1<<b);
        if(~a1&&~a2) return min(a1,a2);
        if(~a1) return a1;if(~a2) return a2;
    }
}T;
long long ans;
void dfs(int a,int b){
    if(b<0) return;
    if(T.son[0][a]&&T.son[1][a]) ans+=1ll*T.Find(T.son[0][a],T.son[1][a],b-1)+(1ll<<b);
    if(T.son[0][a]) dfs(T.son[0][a],b-1);
    if(T.son[1][a]) dfs(T.son[1][a],b-1);
}
int n,v;
int main() {
    n=read();
    for(int i=1;i<=n;i++)T.Insert(read());
    dfs(0,30);
    printf("%I64d\n",ans);
}
```

First element >= x and index >= l

```
#define maxn 200010
#define ll long long
#define lowbit(i) ((i) & (-i))

int n, m, c[maxn], w[maxn];

ll Bit[maxn];
void add(int i, int v) { while (i <= n) Bit[i] += v, i += lowbit(i); }

ll get_sum(int i) {
    ll s = 0;
    while (i) s += Bit[i], i -= lowbit(i);
    return s;
}

int pre[maxn];
set<int> S[maxn];

#define lc i << 1
#define rc i << 1 | 1
int T[maxn * 4];
inline void maintain(int i) { T[i] = max(T[lc], T[rc]); }

void build(int i, int l, int r) {
    if (l == r) return T[i] = pre[l], void();
    int m = l + r >> 1;
```

```
    build(lc, l, m); build(rc, m + 1, r);
    maintain(i);
}

void update(int i, int l, int r, int k, int v) {
    if (l == r) return T[i] = v, void();
    int m = l + r >> 1;
    if (k <= m) update(lc, l, m, k, v);
    else update(rc, m + 1, r, k, v);
    maintain(i);
}

int query(int i, int l, int r, int L, int R, int k) {
    if (l > R || r < L || T[i] < k) return 0;
    if (l == r) return T[i] >= k ? l : 0;
    int m = l + r >> 1, v = query(lc, l, m, L, R, k);
    if (v) return v;
    else return query(rc, m + 1, r, L, R, k);
}

inline void solve_1() {
    int x, y, z; cin >> x >> y >> z;
    add(x, z - w[x]); w[x] = z;
    auto l = S[c[x]].lower_bound(x), r = S[c[x]].upper_bound(x); --l;
    if (*r != n + 1) pre[*r] = *l, update(1, 1, n, *r, *l);
    S[c[x]].erase(x); c[x] = y; S[c[x]].insert(x);
    l = S[c[x]].lower_bound(x), r = S[c[x]].upper_bound(x); --l;
    if (*r != n + 1) pre[*r] = x, update(1, 1, n, *r, x);
    pre[x] = *l; update(1, 1, n, x, *l);
}

int tmp[maxn];
inline void solve_2() {
    int x, y; cin >> x >> y;
    vector<int> vec; ll ans = 0; int p = x;
    while (p <= n) {
        int t = query(1, 1, n, p, n, x);
        if (!t) { ans += get_sum(n) - get_sum(p - 1); break; }
        ans += get_sum(t - 1) - get_sum(p - 1);
        if (!y) break;
        if (!tmp[c[t]]) tmp[c[t]] = w[pre[t]], vec.push_back(c[t]);
        if (tmp[c[t]] < w[t]) ans += w[t] - tmp[c[t]], tmp[c[t]] = w[t];
        p = t + 1; --y;
    } cout << ans << "\n";
    for (auto t : vec) tmp[t] = 0;
}

int main() {
    cin >> n >> m;
    for (int i = 1; i <= n; ++i) cin >> c[i] >> w[i], S[c[i]].insert(i);
    for (int i = 1; i <= n; ++i) S[i].insert(0), S[i].insert(n + 1);
    for (int i = 1, last = 0; i <= n; ++i, last = 0)
        for (auto t : S[i])
            if (1 <= t && t <= n) pre[t] = last, last = t;
    build(1, 1, n);
    for (int i = 1; i <= n; ++i) add(i, w[i]);
    for (int i = 1; i <= m; ++i) {
        int opt; cin >> opt;
        if (opt == 1) solve_1();
        else solve_2();
    }
}
```

CDQ

```
#define lowbit(x) ((x)&(-(x)))
const int maxn=100000+10;
int n,m,c[maxn<<1],ans[maxn],cnt;

struct Element{
    int a,b,c,w,f;
```

```
}e[maxn],t[maxn];

bool cmp(Element x,Element y){
    if(x.a!=y.a) return x.a<y.a;
    if(x.b!=y.b) return x.b<y.b;
    return x.c<y.c;
}

void update(int x,int y){
    for(;x<=m;x+=lowbit(x)) c[x]+=y;
}
int sum(int x){
    int ans=0;
    for(;x;x-=lowbit(x)) ans+=c[x];
    return ans;
}

void CDQ(int l,int r){
    int mid=(l+r)>>1;
    if(l==r) return ;
    CDQ(l,mid);CDQ(mid+1,r);
    int p=l,q=mid+1,tot=l;
    while(p<=mid&&q<=r){
        if(e[p].b<=e[q].b) update(e[p].c,e[p].w),t[tot++]=e[p++];
        else e[q].f+=sum(e[q].c),t[tot++]=e[q++];
    }
    while(p<=mid) update(e[p].c,e[p].w),t[tot++]=e[p++];
    while(q<=r) e[q].f+=sum(e[q].c),t[tot++]=e[q++];
    for(int i=l;i<=mid;i++) update(e[i].c,-e[i].w);
    for(int i=l;i<=r;i++) e[i]=t[i];
}

int main() {
    n=read();m=read();
    for(int i=1;i<=n;i++)
        e[i].a=read(),e[i].b=read(),e[i].c=read(),e[i].w=1;
    sort(e+1,e+n+1,cmp);
    cnt=1;
    for(int i=2;i<=n;i++){
        if(e[i].a==e[cnt].a&&e[i].b==e[cnt].b&&e[i].c==e[cnt].c) e[cnt].w++;
        else e[++cnt]=e[i];
    }
    CDQ(1,cnt);
    for(int i=1;i<=cnt;i++) ans[e[i].f+e[i].w-1]+=e[i].w;
    for(int i=0;i<n;i++) printf("%d\n",ans[i]);
}
```

Parallel Binary Search

```
#define lowbit(x) ((x)&(-(x)))
const int maxn=200000+10;
const int inf=1e9;
int n,m,a[maxn],c[maxn],ans[maxn],cnt,tot;

struct Query{
    int l,r,k,id,op;
}q[maxn*3],q1[maxn*3],q2[maxn*3];

void add(int x,int y){
    for(;x<=n;x+=lowbit(x)) c[x]+=y;
}
int sum(int x){
    int ans=0;
    for(;x;x-=lowbit(x)) ans+=c[x];
    return ans;
}

void solve(int l,int r,int L,int R){
    if(L > R) return ;
    if(l == r){
        for(int i=L;i<=R;i++)
```

```
            if(q[i].op==2) ans[q[i].id]=l;
        return ;
    }
    int mid=(l+r)>>1,cnt1=0,cnt2=0,x;
    for(int i=L;i<=R;i++){
        if(q[i].op==1){
            if(q[i].l <= mid) q1[++cnt1]=q[i],add(q[i].id,q[i].r);
            else q2[++cnt2]=q[i];
        }
        else {
            x=sum(q[i].r)-sum(q[i].l-1);
            if(q[i].k <= x) q1[++cnt1]=q[i];
            else q[i].k-=x,q2[++cnt2]=q[i];
        }
    }
    for(int i=1;i<=cnt1;i++)
        if(q1[i].op==1) add(q1[i].id,-q1[i].r);
    for(int i=1;i<=cnt1;i++) q[L+i-1]=q1[i];
    for(int i=1;i<=cnt2;i++) q[L+i+cnt1-1]=q2[i];
    solve(l,mid,L,L+cnt1-1);
    solve(mid+1,r,L+cnt1,R);
}

int main() {
    n=read(),m=read();
    int l,r,k;char op;
    for(int i=1;i<=n;i++) a[i]=read(),q[++cnt]=(Query){a[i],1,0,i,1};
    for(int i=1;i<=m;i++){
        op=getchar();
        while(!isalpha(op)) op=getchar();
        if(op=='Q') l=read(),r=read(),k=read(),q[++cnt]=(Query){l,r,k,++tot,2};
        else l=read(),r=read(),q[++cnt]=(Query){a[l],-1,0,l,1},q[++cnt]=(Query){a[l]=r,1,0,l,1};
    }
    solve(-inf,inf,1,cnt);
    for(int i=1;i<=tot;i++) printf("%d\n",ans[i]);
    return 0;
}
```

Segment Tree D & C

```
const int N = 1e5 + 7, M = 2e5 + 7;
int n, m, k, u[M], v[M], f[N<<1], d[N<<1];
struct T {
    int l, r;
    vi e;
} t[N<<2];
stack< pi > s;

void build(int p, int l, int r) {
    t[p].l = l, t[p].r = r;
    if (l == r) return;
    build(ls, l, md), build(rs, md + 1, r);
}

void ins(int p, int l, int r, int x) {
    if (t[p].l >= l && t[p].r <= r) return t[p].e.pb(x), void();
    if (l <= md) ins(ls, l, r, x);
    if (r > md) ins(rs, l, r, x);
}

inline int get(int x) {
    while (x ^ f[x]) x = f[x];
    return x;
}

inline void merge(int x, int y) {
    if (x == y) return;
    if (d[x] > d[y]) swap(x, y);
    s.push(mp(x, d[x] == d[y])), f[x] = y, d[y] += d[x] == d[y];
}

```

```cpp
void dfs(int p, int l, int r) {
    bool ok = 1;
    ui o = s.size();
    for (ui i = 0; i < t[p].e.size(); i++) {
        int x = t[p].e[i], u = get(::u[x]), v = get(::v[x]);
        if (u == v) {
            for (int j = l; j <= r; j++) prints("No");
            ok = 0;
            break;
        }
        merge(get(::u[x] + N), v), merge(get(::v[x] + N), u);
    }
    if (ok) {
        if (l == r) prints("Yes");
        else dfs(ls, l, md), dfs(rs, md + 1, r);
    }
    while (s.size() > o) d[f[s.top().fi]] -= s.top().se, f[s.top().fi] = s.top().fi, s.pop();
}

int main() {
    rd(n), rd(m), rd(k), build(1, 1, k);
    for (int i = 1, l, r; i <= m; i++) {
        rd(u[i]), rd(v[i]), rd(l), rd(r);
        if (l ^ r) ins(1, l + 1, r, i);
    }
    for (int i = 1; i <= n; i++) f[i] = i, f[i+N] = i + N;
    dfs(1, 1, k);
    return 0;
}
```

Mo’s Algorithm

```cpp
int main() {
  int n;
  cin >> n;
  vector<int> a(n);
  for (auto& o : a) {
    cin >> o;
    --o;
  }
  int q;
  cin >> q;
  vector queries(q, tuple(0, 0, 0));
  for (int i = 0; i < q; ++i) {
    int l, r;
    cin >> l >> r;
    queries[i] = {l - 1, r, i};
  }
  const int BLOCK_SIZE = int(n / sqrt(q)) > 0 ? n / sqrt(q) : sqrt(n);
  sort(queries.begin(), queries.end(), [&](const auto& x, const auto& y) {
    auto [xl, xr, _i] = x;
    auto [yl, yr, _j] = y;
    if (xl / BLOCK_SIZE != yl / BLOCK_SIZE) {
      return xl / BLOCK_SIZE < yl / BLOCK_SIZE;
    }
    return (xl / BLOCK_SIZE) & 1 ? xr > yr : xr < yr;
  });
  vector<int> ans(q), cnt(n);
  int cur = 0;
  auto add = [&](int i, int v) {
    cnt[a[i]] += v;
    cur += v * ((cnt[a[i]] & 1) == (v < 0));
  };
  // [l, r)
  for (int l = 0, r = 0; auto [ql, qr, i] : queries) {
    while (l > ql) add(--l, 1);
    while (r < qr) add(r++, 1);
    while (l < ql) add(l++, -1);
    while (r > qr) add(--r, -1);
    ans[i] = cur;
  }
```

```
  for (auto o : ans) {
    cout << o << '\n';
  }
}
```

Link/Cut Tree

```
#include<bits/stdc++.h>
#define R register int
#define I inline void
#define G if(++ip==ie)if(fread(ip=buf,1,SZ,stdin))
#define lc c[x][0]
#define rc c[x][1]
using namespace std;
const int SZ=1<<19,N=3e5+9;
char buf[SZ],*ie=buf+SZ,*ip=ie-1;
inline int in(){
    G;while(*ip<'-')G;
    R x=*ip&15;G;
    while(*ip>'-'){x*=10;x+=*ip&15;G;}
    return x;
}
int f[N],c[N][2],v[N],s[N],st[N];
bool r[N];
inline bool nroot(R x){//判断节点是否为一个 Splay 的根（与普通 Splay 的区别 1）
    return c[f[x]][0]==x||c[f[x]][1]==x;
}//原理很简单，如果连的是轻边，他的父亲的儿子里没有它
I pushup(R x){//上传信息
    s[x]=s[lc]^s[rc]^v[x];
}
I pushr(R x){R t=lc;lc=rc;rc=t;r[x]^=1;}//翻转操作
I pushdown(R x){//判断并释放懒标记
    if(r[x]){
        if(lc)pushr(lc);
        if(rc)pushr(rc);
        r[x]=0;
    }
}
I rotate(R x){//一次旋转
    R y=f[x],z=f[y],k=c[y][1]==x,w=c[x][!k];
    if(nroot(y))c[z][c[z][1]==y]=x;c[x][!k]=y;c[y][k]=w;//额外注意 if(nroot(y)) 语句，此处不判断会引起致命错误（与普通 Splay 的
  ↪ 区别 2）
    if(w)f[w]=y;f[y]=x;f[x]=z;
    pushup(y);
}
I splay(R x){//只传了一个参数，因为所有操作的目标都是该 Splay 的根（与普通 Splay 的区别 3）
    R y=x,z=0;
    st[++z]=y;//st 为栈，暂存当前点到根的整条路径，pushdown 时一定要从上往下放标记（与普通 Splay 的区别 4）
    while(nroot(y))st[++z]=y=f[y];
    while(z)pushdown(st[z--]);
    while(nroot(x)){
        y=f[x];z=f[y];
        if(nroot(y))
            rotate((c[y][0]==x)^(c[z][0]==y)?x:y);
        rotate(x);
    }
    pushup(x);
}
/* 当然了，其实利用函数堆栈也很方便，代替上面的手工栈，就像这样
I pushall(R x){
    if(nroot(x))pushall(f[x]);
    pushdown(x);
}*/
I access(R x){//访问
    for(R y=0;x;x=f[y=x])
        splay(x),rc=y,pushup(x);
}
I makeroot(R x){//换根
    access(x);splay(x);
    pushr(x);
}
```

```
int findroot(R x){//找根（在真实的树中的）
    access(x);splay(x);
    while(lc)pushdown(x),x=lc;
    splay(x);
    return x;
}
I split(R x,R y){//提取路径
    makeroot(x);
    access(y);splay(y);
}
I link(R x,R y){//连边
    makeroot(x);
    if(findroot(y)!=x)f[x]=y;
}
I cut(R x,R y){//断边
    makeroot(x);
    if(findroot(y)==x&&f[y]==x&&!c[y][0]){
        f[y]=c[x][1]=0;
        pushup(x);
    }
}
int main() {
    R n=in(),m=in();
    for(R i=1;i<=n;++i)v[i]=in();
    while(m--){
        R type=in(),x=in(),y=in();
        switch(type){
        case 0:split(x,y);printf("%d\n",s[y]);break;
        case 1:link(x,y);break;
        case 2:cut(x,y);break;
        case 3:splay(x);v[x]=y;//先把 x 转上去再改，不然会影响 Splay 信息的正确性
        }
    }
    return 0;
}
```

Filler

```
const int N = 205;
ll n, m, Q;
ll t1[N][N], t2[N][N], t3[N][N], t4[N][N];

void add(ll x, ll y, ll z){
  for(int X = x; X <= n; X += X & -X)
    for(int Y = y; Y <= m; Y += Y & -Y){
      t1[X][Y] += z;
      t2[X][Y] += z * x;
      t3[X][Y] += z * y;
      t4[X][Y] += z * x * y;
    }
}

void range_add(ll xa, ll ya, ll xb, ll yb, ll z){ //(xa, ya) 到 (xb, yb) 的矩形
  add(xa, ya, z);
  add(xa, yb + 1, -z);
  add(xb + 1, ya, -z);
  add(xb + 1, yb + 1, z);
}

ll ask(ll x, ll y){
  ll res = 0;
  for(int i = x; i; i -= i & -i)
  for(int j = y; j; j -= j & -j)
    res += (x + 1) * (y + 1) * t1[i][j]
         - (y + 1) * t2[i][j]
         - (x + 1) * t3[i][j]
         + t4[i][j];
  return res;
}

ll range_ask(ll xa, ll ya, ll xb, ll yb){
```

```
  return ask(xb, yb) - ask(xb, ya - 1) - ask(xa - 1, yb) + ask(xa - 1, ya - 1);
}
```

Two pointers without deletion

```
typedef long long ll;
const int MAXN = 200005;
int T, N;
ll A[MAXN], resl[MAXN], resr;
int main() {
    for (scanf("%d", &T); T; T--) {
        scanf("%d", &N);
        for (int i=1; i<=N; i++) scanf("%lld", &A[i]);
        if (N==1) { puts("1"); continue; }
        for (int i=2; i<=N; i++) A[i-1] -= A[i], A[i-1] = abs(A[i-1]);
        //for (int i=1; i< N; i++) printf("%lld ", A[i]); puts("");
        int l = 1, r = 1, mid = 1, ans = A[1] > 1; // [l, mid], (mid, r];
        resl[1] = A[1]; if (A[1]==1) l = mid+1;
        while (r< N-1) {
            ++r, resr = r==mid+1 ? A[r] : gcd(resr, A[r]);
            while (l<=mid && gcd(resl[l], resr)==1) l++;
            if (l> mid) {
                mid = r, l = r+1, resl[l] = A[l-1];
                while (l> 1 && (resl[l-1]=gcd(resl[l], A[l-1]))> 1) l--;
            }
            ans = max(ans, r-l+1);
            //printf("[%d, %d]\n", l, r);
        }
        printf("%d\n", ans+1);
    }
}
```

Maths

Linear Sieve

```
void init() {
  phi[1] = 1;
  for (int i = 2; i < MAXN; ++i) {
    if (!vis[i]) {
      pri[cnt++] = i;
    }
    for (int j = 0; j < cnt; ++j) {
      if (1ll * i * pri[j] >= MAXN) break;
      vis[i * pri[j]] = 1;
      if (i % pri[j] == 0) break;
    }
  }
}
```

Euler Phi Function Sieve

```
void pre() {
  memset(is_prime, 1, sizeof(is_prime));
  int cnt = 0;
  is_prime[1] = 0;
  phi[1] = 1;
  for (int i = 2; i <= 5000000; i++) {
    if (is_prime[i]) {
      prime[++cnt] = i;
      phi[i] = i - 1;
    }
    for (int j = 1; j <= cnt && i * prime[j] <= 5000000; j++) {
      is_prime[i * prime[j]] = 0;
      if (i % prime[j])
        phi[i * prime[j]] = phi[i] * phi[prime[j]];
      else {
        phi[i * prime[j]] = phi[i] * prime[j];
```

```
          break;
        }
      }
    }
  }
```

Euler Phi Function

```
auto calc = [&] (int n) {
  int res = n;
  for (int p = 2; p * p <= n; ++p) {
    if (n % p == 0) {
      while (n % p == 0) n /= p;
      res -= res / p;
    }
  }
  if (n > 1) res -= res / n;
  return res;
};
```

Linear Inverse

```
vl fac, inv, numinv; // ncr and fac

inline ll ncr(int n, int r){
    if (n < 0 || r < 0 || r > n)
        return 0;
    return fac[n] * inv[r] % mod * inv[n - r] % mod;
}
inline void calfacinv(int n){
    fac.reserve(n + 1);
    fac[0] = fac[1] = 1;
    for (int i = 2; i <= n; i++){
        fac[i] = fac[i - 1] * i % mod;
    }
    numinv.reserve(n + 1);
    numinv[0] = numinv[1] = 1;
    for (int i = 2; i <= n; i++){
        numinv[i] = numinv[mod % i] * (mod - mod / i) % mod;
    }
    inv.reserve(n + 1);
    inv[0] = inv[1] = 1;
    for (int i = 2; i <= n; i++){
        inv[i] = numinv[i] * inv[i - 1] % mod;
    }
    return;
}
```

FFT

- n 需补成 2 的幂（n 必须超过 a 和 b 的最高指数之和）

```
typedef double LD;
const LD PI = acos(-1);
struct C {
    LD r, i;
    C(LD r = 0, LD i = 0): r(r), i(i) {}
};
C operator + (const C& a, const C& b) {
    return C(a.r + b.r, a.i + b.i);
}
C operator - (const C& a, const C& b) {
    return C(a.r - b.r, a.i - b.i);
}
C operator * (const C& a, const C& b) {
    return C(a.r * b.r - a.i * b.i, a.r * b.i + a.i * b.r);
}

void FFT(C x[], int n, int p) {
    for (int i = 0, t = 0; i < n; ++i) {
```

```
        if (i > t) swap(x[i], x[t]);
        for (int j = n >> 1; (t ^= j) < j; j >>= 1);
    }
    for (int h = 2; h <= n; h <<= 1) {
        C wn(cos(p * 2 * PI / h), sin(p * 2 * PI / h));
        for (int i = 0; i < n; i += h) {
            C w(1, 0), u;
            for (int j = i, k = h >> 1; j < i + k; ++j) {
                u = x[j + k] * w;
                x[j + k] = x[j] - u;
                x[j] = x[j] + u;
                w = w * wn;
            }
        }
    }
    if (p == -1)
        FOR (i, 0, n)
            x[i].r /= n;
}

void conv(C a[], C b[], int n) {
    FFT(a, n, 1);
    FFT(b, n, 1);
    FOR (i, 0, n)
        a[i] = a[i] * b[i];
    FFT(a, n, -1);
}
```

FWT

- $C_k = \sum_{i \oplus j = k} A_i B_j$
- FWT 完后需要先模一遍

```
template<typename T>
void fwt(LL a[], int n, T f) {
    for (int d = 1; d < n; d *= 2)
        for (int i = 0, t = d * 2; i < n; i += t)
            FOR (j, 0, d)
                f(a[i + j], a[i + j + d]);
}

void AND(LL& a, LL& b) { a += b; }
void OR(LL& a, LL& b) { b += a; }
void XOR (LL& a, LL& b) {
    LL x = a, y = b;
    a = (x + y) % MOD;
    b = (x - y + MOD) % MOD;
}
void rAND(LL& a, LL& b) { a -= b; }
void rOR(LL& a, LL& b) { b -= a; }
void rXOR(LL& a, LL& b) {
    static LL INV2 = (MOD + 1) / 2;
    LL x = a, y = b;
    a = (x + y) * INV2 % MOD;
    b = (x - y + MOD) * INV2 % MOD;
}
```

- FWT 子集卷积

```
a[popcount(x)][x] = A[x]
b[popcount(x)][x] = B[x]
fwt(a[i]) fwt(b[i])
c[i + j][x] += a[i][x] * b[j][x]
rfwt(c[i])
ans[x] = c[popcount(x)][x]
```

simpson 自适应积分

```
LD simpson(LD l, LD r) {
    LD c = (l + r) / 2;
```

```
    return (f(l) + 4 * f(c) + f(r)) * (r - l) / 6;
}

LD asr(LD l, LD r, LD eps, LD S) {
    LD m = (l + r) / 2;
    LD L = simpson(l, m), R = simpson(m, r);
    if (fabs(L + R - S) < 15 * eps) return L + R + (L + R - S) / 15;
    return asr(l, m, eps / 2, L) + asr(m, r, eps / 2, R);
}

LD asr(LD l, LD r, LD eps) { return asr(l, r, eps, simpson(l, r)); }
```

公式

一些数论公式

- 当 $x \geq \phi(p)$ 时有 $a^x \equiv a^{x \ mod \ \phi(p)+\phi(p)} \pmod p$
- $\mu^2(n) = \sum_{d^2|n} \mu(d)$
- $\sum_{d|n} \varphi(d) = n$
- $\sum_{d|n} 2^{\omega(d)} = \sigma_0(n^2)$，其中 ω 是不同素因子个数
- $\sum_{d|n} \mu^2(d) = 2^{\omega(d)}$

一些数论函数求和的例子

- $\sum_{i=1}^n i[gcd(i,n)=1] = \frac{n\varphi(n)+[n=1]}{2}$
- $\sum_{i=1}^n \sum_{j=1}^m [gcd(i,j)=x] = \sum_d \mu(d) \lfloor \frac{n}{dx} \rfloor \lfloor \frac{m}{dx} \rfloor$
- $\sum_{i=1}^n \sum_{j=1}^m gcd(i,j) = \sum_{i=1}^n \sum_{j=1}^m \sum_{d|gcd(i,j)} \varphi(d) = \sum_d \varphi(d) \lfloor \frac{n}{d} \rfloor \lfloor \frac{m}{d} \rfloor$
- $S(n) = \sum_{i=1}^n \mu(i) = 1 - \sum_{i=1}^n \sum_{d|i,d<i} \mu(d) \overset{t=\frac{i}{d}}{=} 1 - \sum_{t=2}^n S(\lfloor \frac{n}{t} \rfloor)$
  - 利用 $[n=1] = \sum_{d|n} \mu(d)$
- $S(n) = \sum_{i=1}^n \varphi(i) = \sum_{i=1}^n i - \sum_{i=1}^n \sum_{d|i,d<i} \varphi(i) \overset{t=\frac{i}{d}}{=} \frac{i(i+1)}{2} - \sum_{t=2}^n S(\frac{n}{t})$
  - 利用 $n = \sum_{d|n} \varphi(d)$
- $\sum_{i=1}^n \mu^2(i) = \sum_{i=1}^n \sum_{d^2|n} \mu(d) = \sum_{d=1}^{\lfloor \sqrt{n} \rfloor} \mu(d) \lfloor \frac{n}{d^2} \rfloor$
- $\sum_{i=1}^n \sum_{j=1}^n gcd^2(i,j) = \sum_d d^2 \sum_t \mu(t) \lfloor \frac{n}{dt} \rfloor^2$
  $\overset{x=dt}{=} \sum_x \lfloor \frac{n}{x} \rfloor^2 \sum_{d|x} d^2 \mu(\frac{x}{d})$
- $\sum_{i=1}^n \varphi(i) = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n [i \perp j] - 1 = \frac{1}{2} \sum_{i=1}^n \mu(i) \cdot \lfloor \frac{n}{i} \rfloor^2 - 1$

斐波那契数列性质

- $F_{a+b} = F_{a-1} \cdot F_b + F_a \cdot F_{b+1}$
- $F_1 + F_3 + \dots + F_{2n-1} = F_{2n}, F_2 + F_4 + \dots + F_{2n} = F_{2n+1} - 1$
- $\sum_{i=1}^n F_i = F_{n+2} - 1$
- $\sum_{i=1}^n F_i^2 = F_n \cdot F_{n+1}$
- $F_n^2 = (-1)^{n-1} + F_{n-1} \cdot F_{n+1}$
- $gcd(F_a, F_b) = F_{gcd(a,b)}$
- 模 n 周期（皮萨诺周期）
  - $\pi(p^k) = p^{k-1}\pi(p)$
  - $\pi(nm) = lcm(\pi(n), \pi(m)), \forall n \perp m$
  - $\pi(2) = 3, \pi(5) = 20$
  - $\forall p \equiv \pm 1 \pmod{10}, \pi(p) | p-1$
  - $\forall p \equiv \pm 2 \pmod 5, \pi(p) | 2p+2$

常见生成函数

- $(1+ax)^n = \sum_{k=0}^n \binom{n}{k} a^k x^k$

- $\dfrac{1-x^{r+1}}{1-x} = \sum_{k=0}^{n} x^k$
- $\dfrac{1}{1-ax} = \sum_{k=0}^{\infty} a^k x^k$
- $\dfrac{1}{(1-x)^2} = \sum_{k=0}^{\infty} (k+1) x^k$
- $\dfrac{1}{(1-x)^n} = \sum_{k=0}^{\infty} \binom{n+k-1}{k} x^k$
- $e^x = \sum_{k=0}^{\infty} \dfrac{x^k}{k!}$
- $\ln(1+x) = \sum_{k=0}^{\infty} \dfrac{(-1)^{k+1}}{k} x^k$

佩尔方程

若一个丢番图方程具有以下的形式：$x^2 - ny^2 = 1$。且 n 为正整数，则称此二元二次不定方程为**佩尔方程**。

若 n 是完全平方数，则这个方程式只有平凡解 $(\pm 1, 0)$（实际上对任意的 n，$(\pm 1, 0)$ 都是解）。对于其余情况，拉格朗日证明了佩尔方程总有非平凡解。而这些解可由 $\sqrt{n}$ 的连分数求出。

$$x = [a_0; a_1, a_2, a_3] = x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\ddots}}}}$$

设 $\frac{p_i}{q_i}$ 是 $\sqrt{n}$ 的连分数表示: $[a_0; a_1, a_2, a_3, \ldots]$ 的渐近分数列, 由连分数理论知存在 i 使得 (p_i, q_i) 为佩尔方程的解。取其中最小的 i, 将对应的 (p_i, q_i) 称为佩尔方程的基本解, 或最小解, 记作 (x_1, y_1), 则所有的解 (x_i, y_i) 可表示成如下形式: $x_i + y_i\sqrt{n} = (x_1 + y_1\sqrt{n})^i$。或者由以下的递回关系式得到:

$x_{i+1} = x_1 x_i + n y_1 y_i, y_{i+1} = x_1 y_i + y_1 x_i$。

但是：佩尔方程千万不要去推（虽然推起来很有趣，但结果不一定好看，会是两个式子）。记住佩尔方程结果的形式通常是 $a_n = k a_{n-1} - a_{n-2}$（$a_{n-2}$ 前的系数通常是 -1）。暴力 / 凑出两个基础解之后加上一个 0，容易解出 k 并验证。

Burnside & Polya

- $|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|$

注：X^g 是 g 下的不动点数量，也就是说有多少种东西用 g 作用之后可以保持不变。

- $|Y^X/G| = \frac{1}{|G|} \sum_{g \in G} m^{c(g)}$

注：用 m 种颜色染色，然后对于某一种置换 g，有 $c(g)$ 个置换环，为了保证置换后颜色仍然相同，每个置换环必须染成同色。

皮克定理

$2S = 2a + b - 2$

- S 多边形面积
- a 多边形内部点数
- b 多边形边上点数

莫比乌斯反演

- $g(n) = \sum_{d|n} f(d) \Leftrightarrow f(n) = \sum_{d|n} \mu(d) g(\frac{n}{d})$
- $f(n) = \sum_{n|d} g(d) \Leftrightarrow g(n) = \sum_{n|d} \mu(\frac{d}{n}) f(d)$

低阶等幂求和

- $\sum_{i=1}^{n} i^1 = \frac{n(n+1)}{2} = \frac{1}{2}n^2 + \frac{1}{2}n$

- $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} = \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n$
- $\sum_{i=1}^{n} i^3 = \left[\frac{n(n+1)}{2}\right]^2 = \frac{1}{4}n^4 + \frac{1}{2}n^3 + \frac{1}{4}n^2$
- $\sum_{i=1}^{n} i^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30} = \frac{1}{5}n^5 + \frac{1}{2}n^4 + \frac{1}{3}n^3 - \frac{1}{30}n$
- $\sum_{i=1}^{n} i^5 = \frac{n^2(n+1)^2(2n^2+2n-1)}{12} = \frac{1}{6}n^6 + \frac{1}{2}n^5 + \frac{5}{12}n^4 - \frac{1}{12}n^2$

一些组合公式

- 错排公式：$D_1 = 0, D_2 = 1, D_n = (n-1)(D_{n-1} + D_{n-2}) = n!(\frac{1}{2!} - \frac{1}{3!} + \cdots + (-1)^n \frac{1}{n!}) = \lfloor \frac{n!}{e} + 0.5 \rfloor$
- 卡塔兰数（n 对括号合法方案数，n 个结点二叉树个数，$n \times n$ 方格中对角线下方的单调路径数，凸 $n+2$ 边形的三角形划分数，n 个元素的合法出栈序列数）：$C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!n!}$

二次剩余

URAL 1132

```
LL q1, q2, w;
struct P { // x + y * sqrt(w)
    LL x, y;
};

P pmul(const P& a, const P& b, LL p) {
    P res;
    res.x = (a.x * b.x + a.y * b.y % p * w) % p;
    res.y = (a.x * b.y + a.y * b.x) % p;
    return res;
}

P bin(P x, LL n, LL MOD) {
    P ret = {1, 0};
    for (; n; n >>= 1, x = pmul(x, x, MOD))
        if (n & 1) ret = pmul(ret, x, MOD);
    return ret;
}
LL Legendre(LL a, LL p) { return bin(a, (p - 1) >> 1, p); }

LL equation_solve(LL b, LL p) {
    if (p == 2) return 1;
    if ((Legendre(b, p) + 1) % p == 0)
        return -1;
    LL a;
    while (true) {
        a = rand() % p;
        w = ((a * a - b) % p + p) % p;
        if ((Legendre(w, p) + 1) % p == 0)
            break;
    }
    return bin({a, 1}, (p + 1) >> 1, p).x;
}

int main() {
    int T; cin >> T;
    while (T--) {
        LL a, p; cin >> a >> p;
        a = a % p;
        LL x = equation_solve(a, p);
        if (x == -1) {
            puts("No root");
        } else {
            LL y = p - x;
            if (x == y) cout << x << endl;
            else cout << min(x, y) << " " << max(x, y) << endl;
        }
    }
}
```

伯努利数和等幂求和

- 预处理逆元
- 预处理组合数
- $\sum_{i=0}^{n} i^k = \frac{1}{k+1} \sum_{i=0}^{k} \binom{k+1}{i} B_{k+1-i} (n+1)^i$.
- 也可以 $\sum_{i=0}^{n} i^k = \frac{1}{k+1} \sum_{i=0}^{k} \binom{k+1}{i} B^{+}_{k+1-i} n^i$。区别在于 $B^{+}_1 = 1/2$。(心态崩了)

```
namespace Bernoulli {
    const int M = 100;
    LL inv[M] = {-1, 1};
    void inv_init(LL n, LL p) {
        FOR (i, 2, n)
            inv[i] = (p - p / i) * inv[p % i] % p;
    }

    LL C[M][M];
    void init_C(int n) {
        FOR (i, 0, n) {
            C[i][0] = C[i][i] = 1;
            FOR (j, 1, i)
                C[i][j] = (C[i - 1][j] + C[i - 1][j - 1]) % MOD;
        }
    }

    LL B[M] = {1};
    void init() {
        inv_init(M, MOD);
        init_C(M);
        FOR (i, 1, M - 1) {
            LL& s = B[i] = 0;
            FOR (j, 0, i)
                s += C[i + 1][j] * B[j] % MOD;
            s = (s % MOD * -inv[i + 1] % MOD + MOD) % MOD;
        }
    }

    LL p[M] = {1};
    LL go(LL n, LL k) {
        n %= MOD;
        if (k == 0) return n;
        FOR (i, 1, k + 2)
            p[i] = p[i - 1] * (n + 1) % MOD;
        LL ret = 0;
        FOR (i, 1, k + 2)
            ret += C[k + 1][i] * B[k + 1 - i] % MOD * p[i] % MOD;
        ret = ret % MOD * inv[k + 1] % MOD;
        return ret;
    }
}
```

离散对数

BSGS

- 模数为素数

```
LL BSGS(LL a, LL b, LL p) { // a^x = b (mod p)
    a %= p;
    if (!a && !b) return 1;
    if (!a) return -1;
    static map<LL, LL> mp; mp.clear();
    LL m = sqrt(p + 1.5);
    LL v = 1;
    FOR (i, 1, m + 1) {
        v = v * a % p;
        mp[v * b % p] = i;
    }
    LL vv = v;
    FOR (i, 1, m + 1) {
        auto it = mp.find(vv);
```

```
        if (it != mp.end()) return i * m - it->second;
        vv = vv * v % p;
    }
    return -1;
}
```

exBSGS

- 模数可以非素数

```
LL exBSGS(LL a, LL b, LL p) { // a^x = b (mod p)
    a %= p; b %= p;
    if (a == 0) return b > 1 ? -1 : b == 0 && p != 1;
    LL c = 0, q = 1;
    while (1) {
        LL g = __gcd(a, p);
        if (g == 1) break;
        if (b == 1) return c;
        if (b % g) return -1;
        ++c; b /= g; p /= g; q = a / g * q % p;
    }
    static map<LL, LL> mp; mp.clear();
    LL m = sqrt(p + 1.5);
    LL v = 1;
    FOR (i, 1, m + 1) {
        v = v * a % p;
        mp[v * b % p] = i;
    }
    FOR (i, 1, m + 1) {
        q = q * v % p;
        auto it = mp.find(q);
        if (it != mp.end()) return i * m - it->second + c;
    }
    return -1;
}
```

数论分块

$f(i) = \lfloor \frac{n}{i} \rfloor = v$ 时 i 的取值范围是 $[l, r]$。

```
for (LL l = 1, v, r; l <= N; l = r + 1) {
    v = N / l; r = N / v;
}
```

博弈

- Nim 游戏：每轮从若干堆石子中的一堆取走若干颗。先手必胜条件为石子数量异或和非零。
- 阶梯 Nim 游戏：可以选择阶梯上某一堆中的若干颗向下推动一级，直到全部推下去。先手必胜条件是奇数阶梯的异或和非零（对于偶数阶梯的操作可以模仿）。
- Anti-SG：无法操作者胜。先手必胜的条件是：
  - SG 不为 0 且某个单一游戏的 SG 大于 1 。
  - SG 为 0 且没有单一游戏的 SG 大于 1。
- Every-SG：对所有单一游戏都要操作。先手必胜的条件是单一游戏中的最大 step 为奇数。
  - 对于终止状态 step 为 0
  - 对于 SG 为 0 的状态，step 是最大后继 step +1
  - 对于 SG 非 0 的状态，step 是最小后继 step +1
- 树上删边：叶子 SG 为 0，非叶子结点为所有子结点的 SG 值加 1 后的异或和。

尝试：

- 打表找规律
- 寻找一类必胜态（如对称局面）
- 直接博弈 dp # Graph

Euler Path/Cycle

```cpp
struct edge {
  int to;
  bool exists;
  int revref;

  bool operator<(const edge& b) const { return to < b.to; }
};

vector<edge> beg[505];
int cnt[505];

const int dn = 500;
stack<int> ans;

void Hierholzer(int x) {  // 关键函数
  for (int& i = cnt[x]; i < (int)beg[x].size();) {
    if (beg[x][i].exists) {
      edge e = beg[x][i];
      beg[x][i].exists = 0;
      beg[e.to][e.revref].exists = 0;
      ++i;
      Hierholzer(e.to);
    } else {
      ++i;
    }
  }
  ans.push(x);
}

int deg[505];
int reftop[505];

int main() {
  for (int i = 1; i <= dn; ++i) {
    beg[i].reserve(1050);  // vector 用 reserve 避免动态分配空间，加快速度
  }

  int m;
  scanf("%d", &m);
  for (int i = 1; i <= m; ++i) {
    int a, b;
    scanf("%d%d", &a, &b);
    beg[a].push_back((edge){b, 1, 0});
    beg[b].push_back((edge){a, 1, 0});
    ++deg[a];
    ++deg[b];
  }

  for (int i = 1; i <= dn; ++i) {
    if (!beg[i].empty()) {
      sort(beg[i].begin(), beg[i].end());  // 为了要按字典序贪心，必须排序
    }
  }

  for (int i = 1; i <= dn; ++i) {
    for (int j = 0; j < (int)beg[i].size(); ++j) {
      beg[i][j].revref = reftop[beg[i][j].to]++;
    }
  }

  int bv = 0;
  for (int i = 1; i <= dn; ++i) {
    if (!deg[bv] && deg[i]) {
      bv = i;
    } else if (!(deg[bv] & 1) && (deg[i] & 1)) {
      bv = i;
    }
  }

```

```
  Hierholzer(bv);

  while (!ans.empty()) {
    printf("%d\n", ans.top());
    ans.pop();
  }
}
```

String

Generalized Suffix Automaton

```
namespace SA {
  constexpr int N = 4e5;
  int ch[N][26], vis[N][26];
  int last, tot, len[N], link[N];

  inline void init() {
    last = 0;
    tot = 1;
    link[0] = -1;
  }

  inline void extend(int c) {
    int u = last;
    if (ch[u][c]) {
      if (len[u] + 1 == len[ch[u][c]]) {
        last = ch[u][c];
        return;
      }
      int w = ch[u][c], clone = tot++;
      memcpy(ch[clone], ch[w], sizeof ch[w]);
      len[clone] = len[u] + 1;
      link[clone] = link[w];
      link[w] = clone;
      for (; u != -1 && ch[u][c] == w; u = link[u]) {
        ch[u][c] = clone;
      }
      last = clone;
      return;
    }
    int v = tot++;
    len[v] = len[u] + 1;
    for (; u != -1 && !ch[u][c]; u = link[u]) {
      ch[u][c] = v;
    }
    if (u == -1) {
      link[v] = 0;
    } else if (len[u] + 1 == len[ch[u][c]]) {
      link[v] = ch[u][c];
    } else {
      int w = ch[u][c], clone = tot++;
      memcpy(ch[clone], ch[w], sizeof ch[w]);
      len[clone] = len[u] + 1;
      link[clone] = link[w];
      link[w] = link[v] = clone;
      for (; u != -1 && ch[u][c] == w; u = link[u]) {
        ch[u][c] = clone;
      }
    }
    last = v;
  }
}

int main() {
  int n;
  cin >> n;
  vector<string> s(n);
  vector<int> ec(n);
  using namespace SA;
```

```cpp
  init();
  for (int i = 0; i < n; ++i) {
    cin >> s[i];
    last = 0;
    for (auto& c : s[i]) {
      extend(c - 'a');
    }
    ec[i] = last;
  }
  for (int i = 0; i < n; ++i) {
    int u = 0;
    for (auto c : s[i]) {
      c -= 'a';
      vis[u][c] = 1;
      u = ch[u][c];
    }
  }
  string t;
  cin >> t;
  int u = 0;
  vector<int> match(tot), cnt(tot), q(tot);
  for (auto c : t) {
    c -= 'a';
    for (; u != -1 && !(ch[u][c] && vis[u][c]); u = link[u]);
    if (u == -1) {
      u = 0;
      continue;
    }
    u = ch[u][c];
    ++match[u];
  }
  for (int i = 0; i < tot; ++i) {
    ++cnt[len[i]];
  }
  for (int i = 1; i < tot; ++i) {
    cnt[i] += cnt[i - 1];
  }
  for (int i = 0; i < tot; ++i) {
    q[--cnt[len[i]]] = i;
  }
  for (int i = tot - 1; i > 0; --i) {
    match[link[q[i]]] += match[q[i]];
  }
  for (auto& o : ec) {
    cout << match[o] << '\n';
  }
}
```

Generator

makefile

```makefile
% : %.cpp
	g++ -g -std=c++2a $< -o $@ -O2

# % : %.cpp
#	g++ -Wall -Wconversion -Wfatal-errors -Wshadow -g -std=c++17 -fsanitize=undefined,address $< -o $@ -O2
```

bash script

```bash
#!/bin/bash

cnt=1
cnt_max=12
wa=0
g++ -std=c++2a gen.cpp -o gen -O2
g++ -std=c++2a all.cpp -o all -O2
g++ -std=c++2a all2.cpp -o all2 -O2
# make gen
```

```
# make all
# make all2

# im=Impossible
# im=-1

# while [ $wa -eq 0 ] && [ $cnt -le $cnt_max ]
while [ $wa -eq 0 ]
do
    ./gen ${cnt} > all.in
    # echo Case# $cnt input:
    ./all < all.in > all.out
    # cat "all.in" > "all2.in"
    # cat "all.out" >> "all2.in"
    ./all2 < all.in > all2.out
    # python3 all.py < all.in > all2.out
    diff all.out all2.out
    wa=$?
    [ $wa == 1 ] && s="WA" || s="AC"
    echo Case# $cnt result: $s
    # line=$(head -n 1 all.out)
    # if [ "$line" != "$im" ]
    # then echo Case# $cnt answer:
        # cat all.out
    # then cp all.in ${cnt}.in
    # fi
    cnt=$(($cnt + 1))
done

# zip -m input.zip *[0-9].in
```

minified “testlib.h”

```
// ICPC Generator by gabrielliu2001 (Mar 31 2022)
#include "bits/stdc++.h"
using namespace std;

// https://github.com/MikeMirzayanov/testlib
// Adapted from "testlib.h" for stressing solutions in ICPC
struct icpc_gen {
  unsigned long long seed = 3905348978240129619LL;
  unsigned long long mul = 0x5DEECE66DLL;
  unsigned long long add = 0xBLL;
  unsigned long long mask = (1LL << 48) - 1;
  mt19937_64 rng;

  void setSeed(int argc, char *argv[], int fix_seed) {
    if (fix_seed) {
      for (int i = 1; i < argc; ++i) {
        size_t le = strlen(argv[i]);
        for (size_t j = 0; j < le; ++j) {
          seed = seed * mul + (unsigned int) (argv[i][j]) + add;
        }
        seed += mul / add;
      }
    } else {
      seed = chrono::steady_clock::now().time_since_epoch().count();
    }
    rng.seed(seed);
  }

  // Random value in range [l, r] for int, range [l, r) for real
  template<class T>
  T next(T l, T r) {
    assert(l <= r);
    if constexpr(is_floating_point_v<T>) {
      return uniform_real_distribution<T>(l, r)(rng);
    } else {
      return uniform_int_distribution<T>(l, r)(rng);
    }
  }
```

```
// Random permutation of the given size (values are between `f` and `f + n - 1`)
template<class T, class E = int>
vector<E> perm(T n, E f = 0) {
  assert(n > 0);
  vector<E> p(n);
  iota(p.begin(), p.end(), f);
  shuffle(p.begin(), p.end(), rng);
  return p;
}

// Return `size` unordered (unsorted) distinct numbers between `l` and `r`
template<class T>
vector<T> distinct(int size, T l, T r) {
  vector<T> v;
  assert(size > 0);
  assert(l <= r);
  T n = r - l + 1;
  assert(size <= n);
  double ev = 0.0;
  for (int i = 1; i <= size; ++i) {
    ev += double(n) / double(n - i + 1);
  }
  if (ev < double(n)) {
    set<T> s;
    while (int(s.size()) < size) {
      T x = T(next(l, r));
      if (s.emplace(x).second) {
        v.emplace_back(x);
      }
    }
  } else {
    assert(n <= int(1e9));
    vector<T> p(perm(int(n), l));
    v.insert(v.end(), p.begin(), p.begin() + size);
  }
  return v;
}

// Return random (unsorted) partition which is a representation of sum as a sum of integers not less than min_part
template<class T>
vector<T> partition(int size, T sum, T min_part = 1) {
  assert(size > 0);
  assert(min_part * size <= sum);
  T given_sum = sum, result_sum = 0;
  sum -= min_part * size;
  vector<T> septums(size), result(size);
  vector<T> d = distinct(size - 1, T(1), T(sum + size - 1));
  for (int i = 0; i + 1 < size; ++i) {
    septums[i + 1] = d[i];
  }
  sort(septums.begin(), septums.end());
  for (int i = 0; i + 1 < size; ++i) {
    result[i] = septums[i + 1] - septums[i] - 1;
  }
  result[size - 1] = sum + size - 1 - septums.back();
  for (auto& o : result) {
    o += min_part;
    result_sum += o;
  }
  assert(result_sum == given_sum);
  assert(*min_element(result.begin(), result.end()) >= min_part);
  assert(int(result.size()) == size && result.size() == (size_t) size);
  return result;
}

// Random interval [l, r] where l <= r
template<class T>
pair<T, T> interval(T l, T r) {
  assert(l <= r);
  T x = next(l - 1, r);
```

```cpp
    T y = next(l, r);
    return x == l - 1 ? pair(y, y) : pair(min(x, y), max(x, y));
  }
} rnd;

int main(int argc, char* argv[]) {
  ios_base::sync_with_stdio(0), cin.tie(0);
  rnd.setSeed(argc, argv, 1);
}
```

Tree Generator

```cpp
// Tree Generator by gabrielliu2001 (Mar 31 2022)
#include "testlib.h"
#include "bits/stdc++.h"
using namespace std;

struct tree_generator {
  // https://cp-algorithms.com/graph/pruefer_code.html#restoring-the-tree-using-the-prufer-code-in-linear-time
  // Random tree by decoding random generated pruefer code
  vector<pair<int, int>> gen_random_tree(int n) {
    if (n == 1) {
      return vector<pair<int, int>>();
    }
    vector<int> code(n - 2);
    for (int& v : code) v = rnd.next(n);

    vector<int> degree(n, 1);
    for (int i : code) degree[i]++;

    int ptr = 0;
    while (degree[ptr] != 1) ptr++;
    int leaf = ptr;

    vector<pair<int, int>> edges;
    for (int v : code) {
      edges.emplace_back(leaf, v);
      if (--degree[v] == 1 && v < ptr) {
        leaf = v;
      } else {
        ptr++;
        while (degree[ptr] != 1) ptr++;
        leaf = ptr;
      }
    }
    edges.emplace_back(leaf, n - 1);
    return edges;
  }

  // i <= chain_sz --> chain, i > chain_sz --> center
  // is_caterpillar --> random center
  // strict_caterpillar --> single node connect to chain node only
  vector<pair<int, int>> gen_chain_related(int n, int chain_sz, bool is_caterpillar = false, bool strict_caterpillar = false) {
    vector<pair<int, int>> edges;
    vector<int> f = rnd.perm(n);
    for (int i = 1; i < n; ++i) {
      int p = i <= chain_sz ? i - 1 : (is_caterpillar ? (strict_caterpillar ? rnd.next(0, chain_sz) : rnd.next(i)) : 0);
      edges.emplace_back(f[p], f[i]);
    }
    return edges;
  }

  // sz = number of children a parent has
  // node_sz = actual number of vertices within a node
  vector<pair<int, int>> gen_binary_tree_related(int n, int sz, int node_sz = 1) {
    vector<pair<int, int>> edges;
    vector<vector<int>> nodes((n - 1) / node_sz + 1);
    vector<int> f = rnd.perm(n);
    for (int i = 0; i < n; ++i) {
```

```cpp
      nodes[i / node_sz].emplace_back(i);
    }
    for (auto v : nodes) {
      for (int i = 1; i < v.size(); ++i) {
        edges.emplace_back(f[v[i - 1]], f[v[i]]);
      }
    }
    for (int i = 1; i < nodes.size(); ++i) {
      int u = nodes[(i - 1) / sz].back(), v = nodes[i].front();
      edges.emplace_back(f[u], f[v]);
    }
    return edges;
  }

  // Binary chain
  vector<pair<int, int>> gen_binary_chain(int n) {
    vector<pair<int, int>> edges;
    vector<int> f = rnd.perm(n);
    vector<int> tri;
    for (int acc = 1, sum = 1; sum <= n; ++acc, sum += acc) {
      tri.emplace_back(sum);
    }
    for (int i = 1; i < tri.size(); ++i) {
      for (int j = tri[i - 1]; j < tri[i]; ++j) {
        int p = j - i - (j + 1 == tri[i]);
        edges.emplace_back(f[p], f[j]);
      }
    }
    for (int i = tri.back(); i < n; ++i) {
      int p = rnd.next(i);
      edges.emplace_back(f[p], f[i]);
    }
    return edges;
  }

  // https://oi-wiki.org/contest/problemsetting/#_26
  // type 0: random tree, 1: chain, 2: star, 3: (n / 2) chain + (n / 2) star
  // {4, 5, 6}: caterpillar with {[5, 10], sqrt(n), (n / 2)} single nodes
  // 7: binary tree, 8: binary sqrt(n) chain, 9: sqrtary tree, 10: sqrtray sqrt(n) chain
  // 11: binary chain of depth d --> chain of depth d and binary chain of depth (d - 1)
  vector<pair<int, int>> gen_common_tree(int n, int type = 0, int base = 0) {
    vector<pair<int, int>> edges;
    switch (type) {
      case 0: edges = gen_random_tree(n); break;
      case 1: edges = gen_chain_related(n, n); break;
      case 2: edges = gen_chain_related(n, 0); break;
      case 3: edges = gen_chain_related(n, n / 2); break;
      case 4: edges = gen_chain_related(n, n - rnd.next(5, 10), true); break;
      case 5: edges = gen_chain_related(n, n - int(sqrt(n)), true); break;
      case 6: edges = gen_chain_related(n, n / 2, true); break;
      case 7: edges = gen_binary_tree_related(n, 2); break;
      case 8: edges = gen_binary_tree_related(n, 2, sqrt(n)); break;
      case 9: edges = gen_binary_tree_related(n, sqrt(n)); break;
      case 10: edges = gen_binary_tree_related(n, sqrt(n), sqrt(n)); break;
      case 11: edges = gen_binary_chain(n); break;
      default: edges = gen_random_tree(n);
    }
    for (auto& [u, v] : edges) {
      u += base, v += base;
      if (rnd.next(2)) {
        swap(u, v);
      }
    }
    shuffle(edges.begin(), edges.end());
    return edges;
  }
} tg;

int main(int argc, char* argv[]) {
  registerGen(argc, argv, 1);
  int n = opt<int>("n");
```

```
  int type = opt<int>("type");
  int base = opt<int>("base");

  vector<pair<int, int>> edges = tg.gen_common_tree(n, type, base);

  println(n);
  for (auto [u, v] : edges) {
    println(u, v);
  }
}
```

Prime Related Generator

```
# ICPC Generator (Prime Related) by gabrielliu2001 (Mar 31 2022)

# Maximize number of duplicate prime divisor: power of 2
# Maximize number of prime divisor: 2 * 3 * 5 * 7 * ...
# Highly composite numbers:
def divisor_count(n):
  i = 2
  cnt = 0
  while i ** 2 <= n:
    if n % i == 0:
      cnt += 2
      if n // i == i:
        cnt -= 1
    i += 1
  return cnt

A002182_list, r = [], 0
i = 1
while i <= 10 ** 5:
  d = divisor_count(i)
  if d > r:
    r = d
    A002182_list.append(i)
  i += 1

print(A002182_list)
```