Template of Shanghai University

Wand Maker

May 14, 2016

Contents

1 DS

1.1 Intervals

```
class Intervals {
  TreeMap<Integer, Integer> map = new TreeMap<>();
  Intervals() {
    map.put(Integer.MIN_VALUE, -1);
    map.put(Integer.MAX_VALUE, -1);
  }
  void paint(int s, int t, int c) {
    int p = get(t);
    map.subMap(s, t).clear();
    if (get(s) != c) map.put(s, c);
    if (get(t) != p) map.put(t, p);
    if (p == c) map.remove(t);
  }
  int get(int k) {
    return map.floorEntry(k).getValue();
  }
}
```

1.2 Seg Lite

```
class SegLite {
  final int init;
  final IntBinaryOperator op;
  final int N;
  int[] is;

  SegLite(int[] vs, int init, IntBinaryOperator op) {
    this.init = init;
    this.op = op;
    N = Integer.highestOneBit(vs.length) << 1;
    is = new int[N * 2];
    System.arraycopy(vs, 0, is, N, vs.length);
    for (int i = N - 1; i > 0; i--) {
      is[i] = op.applyAsInt(is[i << 1], is[i << 1 | 1]);
    }
  }
  void update(int k, int v) {
    k += N;
    is[k] = v;
    for (k >>= 1; k > 0; k >>= 1) {
      is[k] = op.applyAsInt(is[k << 1], is[k << 1 | 1]);
    }
  }
  int query(int s, int t) {
    int left = init;
    int right = init;
    while (0 < s && s + (s & -s) <= t) {
      int i = (N + s) / (s & -s);
      left = op.applyAsInt(left, is[i]);
      s += s & -s;
    }
    while (s < t) {
      int i = (N + t) / (t & -t) - 1;
      right = op.applyAsInt(is[i], right);
      t -= t & -t;
    }
    return op.applyAsInt(left, right);
  }
}
```

1.3 Seg Lazy

```
class Seg {
  int N;
  int[] sum;
  int[] mul;
  int[] add;

  Seg(int[] is) {
    int n = is.length;
    N = Integer.highestOneBit(n) << 1;
    sum = new int[N * 2];
    mul = new int[N * 2];
    add = new int[N * 2];
    Arrays.fill(mul, 1);
    for (int i = 0; i < n; i++) {
      sum[i + N] = is[i];
    }
    for (int i = N - 1; i > 0; i--) {
      sum[i] = sum[i << 1] + sum[i << 1 | 1];
    }
  }
  void update(int o, int L, int R, int l, int r, int m,
  int a) {
    if (l <= L && R <= r) {
      push(o, L, R, m, a);
    } else {
      int M = (L + R) >> 1;
      push(o, L, M, R);
      if (l < M) update(o << 1, L, M, l, r, m, a);
      if (r > M) update(o << 1 | 1, M, R, l, r, m, a);
      sum[o] = sum[o << 1] + sum[o << 1 | 1];
    }
  }
  void push(int o, int L, int M, int R) {
    if (mul[o] != 1 || add[o] != 0) {
      push(o << 1, L, M, mul[o], add[o]);
      push(o << 1 | 1, M, R, mul[o], add[o]);
      mul[o] = 1;
      add[o] = 0;
    }
  }
  void push(int o, int L, int R, int m, int a) {
    sum[o] = sum[o] * m + a * (R - L);
    mul[o] = mul[o] * m;
    add[o] = add[o] * m + a;
  }
  int query(int o, int L, int R, int l, int r) {
    if (l <= L && R <= r) {
      return sum[o];
    } else {
      int M = (L + R) >> 1;
      push(o, L, M, R);
      int res = 0;
      if (l < M) res += query(o << 1, L, M, l, r);
      if (r > M) res += query(o << 1 | 1, M, R, l, r);
      sum[o] = sum[o << 1] + sum[o << 1 | 1];
      return res;
    }
  }
}
```

1.4 Treap

```
class T {
  int key;
  int size;
  double p;
  T left;
  T right;

  // TODO generate constructor
  T(int key) {
    this(key, 1, Math.random(), NULL, NULL);
  }
  T change(T left, T right) {
    size = left.size + right.size + 1;
    this.left = left;
    this.right = right;
    return this;
  }
  T push() {
    if (this != NULL) {

    }
    return this;
  }
}
static final T NULL = new T(0, 0, 0, null, null);

T[] splitSize(T t, int size) {
  T[] res;
  if (size <= 0) {
    res = new T[] { NULL, t };
  } else if (size <= t.push().left.size) {
    res = splitSize(t.left, size);
    res[1] = t.change(res[1], t.right);
  } else {
    res = splitSize(t.right, size - t.left.size - 1);
    res[0] = t.change(t.left, res[0]);
  }
```

```
  return res;
}
T[] splitKey(T t, int key) {
  T[] res;
  if (t == NULL) {
    res = new T[] { NULL, NULL };
  } else if (key < t.push().key) {
    res = splitKey(t.left, key);
    res[1] = t.change(res[1], t.right);
  } else {
    res = splitKey(t.right, key);
    res[0] = t.change(t.left, res[0]);
  }
  return res;
}
T merge(T t1, T t2) {
  if (t1 == NULL) return t2;
  if (t2 == NULL) return t1;
  if (t1.p < t2.p) return t1.push().change(t1.left, merge(t1.right, t2));
  return t2.push().change(merge(t1, t2.left), t2.right);
}
```

1.5 LCT

```
class T {
  int id;
  boolean rev;
  double p;
  T pre;
  T left;
  T right;

  // TODO generate constructor
  T(int id) {
    this(id, false, Math.random(), NULL, NULL, NULL);
  }
  T change(T left, T right) {
    this.left = left; left.pre = this;
    this.right = right; right.pre = this;
    return this;
  }
  T setRev() {
    if (this == NULL) return NULL;
    rev ^= true;
    T t = left; left = right; right = t;
    return this;
  }
  T push() {
    if (rev) {
      left.setRev();
      right.setRev();
      rev ^= true;
    }
    return this;
  }
}
static final T NULL = new T(0);

T merge(T t1, T t2) {
  if (t1 == NULL) return t2;
  if (t2 == NULL) return t1;
  if (t1.p < t2.p) return t1.push().change(t1.left, merge(t1.right, t2));
  return t2.push().change(merge(t1, t2.left), t2.right);
}
T[] split(T t) {
  pushDownAllMark(t);
  T[] res = new T[2];
  res[1] = t.right;
  res[0] = t.change(t.left, NULL);
  T tcp = t;
  for (;;) {
    if (t.pre.left == t) {
      t = t.pre;
      res[1] = t.change(res[1], t.right);
    } else if (t.pre.right == t) {
      t = t.pre;
      res[0] = t.change(t.left, res[0]);
    } else {
      res[0].pre = t.pre;
      res[1].pre = tcp;
      return res;
    }
  }
}
T access(T t) {
  T last = NULL;
  while (t != NULL) {
    T[] ss = split(t);
    t = ss[0].pre;
    last = merge(ss[0], last);
  }
  last.pre = NULL;
  return last;
}
// ——— Top Tree ———
T access(T t) {
  T last = NULL;
  while (t != NULL) {
    T[] ss = split(t);
    if (ss[1] != NULL) {
      inc(ss[1].pre.set, ss[1].top); // top is maintain-info
    }
    if (last != NULL) {
      dec(ss[1].pre.set, last.top);
    }
    update(ss[1].pre);
    t = ss[0].pre;
    last = merge(ss[0], last);
  }
  last.pre = NULL;
  return last;
}
void update(T t) {
  t.change(t.left, t.right);
  if (t.pre.right == t || t.pre.left == t) update(t.pre);
}
// ——— End ———
T makeRoot(T t) {
  return access(t).setRev();
}
T getRoot(T t) {
  t = access(t);
  while (t.push().left != NULL) t = t.left;
  return t;
}
void link(T x, T y) {
  makeRoot(x).pre = y;
}
// make y to be root and cut the edge x to its father.
void cut(T x, T y) {
  makeRoot(y);
  access(y);
  while (x.pre.left == x || x.pre.right == x) x = x.pre;
  x.pre = NULL;
}
void pushDownAllMark(T t) {
  if (t.pre.left == t || t.pre.right == t) pushDownAllMark(t.pre);
  t.push();
}
```

1.6 LCA

```
class LCA {
  List<Integer>[] vs;
  int root;

  int[] depth;
  int[][] pre;

  LCA(List<Integer>[] vs, int root) {
    this.vs = vs;
    this.root = root;
    int n = vs.length;
    depth = new int[n];
    pre = new int[Algo.log2(n) + 1][n];
    dfs(root, -1, 0);
    for (int k = 0; k + 1 < pre.length; k++) {
      for (int v = 0; v < n; v++) {
        if (pre[k][v] < 0) pre[k + 1][v] = -1;
        else pre[k + 1][v] = pre[k][pre[k][v]];
      }
    }
```

```
  }
  void dfs(int v, int p, int d) {
    pre[0][v] = p;
    depth[v] = d;
    for (int u : vs[v]) if (u != p) {
      dfs(u, v, d + 1);
    }
  }
  int lca(int u, int v) {
    if (depth[u] > depth[v]) return lca(v, u);
    v = climb(v, depth[v] - depth[u]);
    if (u == v) return u;
    for (int k = pre.length - 1; k >= 0; k--) {
      if (pre[k][u] != pre[k][v]) {
        u = pre[k][u];
        v = pre[k][v];
      }
    }
    return pre[0][u];
  }
  int climb(int v, int d) {
    for (int k = 0; k < pre.length; k++) {
      if ((d >> k & 1) != 0) v = pre[k][v];
    }
    return v;
  }
}
```

2 DS G

2.1 Binary Indexed Tree

```
void add(int i, ll x) {
  while(i<=sz) {
    bit[i]+=x;
    i+=i&-i;
  }
}
ll query(int i) {
  ll sum=0;
  while(i>0) {
    sum+=bit[i];
    i-=i&-i;
  }
  return sum;
}
int kth(int k) { // k >= 1
  // init: H=1; while(H <= sz) H<<=1;
  int r = 0, h = H;
  while(h>>=1) {
    r += h;
    if(r > sz || bit[r] >= k)
      r -= h;
    else
      k -= bit[r];
  }
  return r+1;
}
```

2.2 Heavy-Light Decomposition

```
VI Map[MAXN]; //处理后Map[0]中存放重儿子
int w[MAXN], fa[MAXN], dep[MAXN], sz[MAXN], top[MAXN];
int e[MAXN]; //对操作子树操作需要e[]
int depth = 0, tot = 0; //记得初始化Map[]和tot
void dfs(int u, int f=-1) {
  dep[u]=depth++;
  sz[u]=1;
  fa[u]=f;
  top[u]=u;
  PII mx=MP(0,0);
  REP(i,Map[u].size()) {
    int v=Map[u][i];
    if(v!=f) {
      dfs(v,u);
      sz[u]+=sz[v];
      mx=max(mx, MP(sz[v], i));
    }
  }
  if(mx.Y) swap(Map[u][0], Map[u][mx.Y]);
  depth--;
}
void dfs2(int u, int f=-1) {
  w[u]=tot++;
  REP(i,Map[u].size()) {
    int v=Map[u][i];
    if(v!=f) {
      if(i==0) top[v]=top[u];
      dfs2(v,u);
    }
  }
  e[u]=tot-1;
}
void operation(int u, int v) {
  while(top[u]!=top[v]) {
    if(dep[top[u]]<dep[top[v]]) swap(u,v);
    update(1,0,N-1,w[top[u]],w[u]); //在相应线段树上操作
    u=fa[top[u]];
  }
  if(dep[u]>dep[v]) swap(u,v);
  update(1,0,N-1,w[u],w[v]); //操作点权， 若为边权则操作w[u]+1, w[v]
}
void operationOnSubtree(int u) {
  update(1,0,N-1,w[u],e[u]);
}
```

2.3 Persistent SegTree

静态区间第 K 大

```
#define MAXN 30000*20
int ls[MAXN], rs[MAXN], c[MAXN];
int root[30001];
int p;
int a[30000];
int b[30001];
int cnt;
int id(int x) {
  return lower_bound(b, b+cnt, x) - b;
}
void ins(int &i, int j, int l, int r, int pos) {
  ls[p]=ls[j]; rs[p]=rs[j], c[p]=c[j]+1; i=p++;
  if(l==r) return;
  int m=(l+r)>>1;
  if(pos<=m) {
    ins(ls[i],ls[j],l,m,pos);
  }else{
    ins(rs[i],rs[j],m+1,r,pos);
  }
}
int query(int i, int j, int l, int r, int k) {
  if(l==r) return c[i]-c[j];
  int m=(l+r)>>1;
  return (k<=m?c[rs[i]]-c[rs[j]]+query(ls[i],ls[j],l,m,k):
           query(rs[i],rs[j],m+1,r,k));
}
int main() {
  int n;
  n=Scan();
  REP(i,n) {
    a[i]=Scan();
    b[i]=a[i];
  }
  sort(b,b+n);
  cnt = unique(b,b+n)-b;
  ls[0]=rs[0]=c[0]=0;
  root[0]=0;
  p=1;
  REP(i,n) {
    a[i] = id(a[i]);
    ins(root[i+1], root[i], 0, cnt-1, a[i]);
  }
  int q=Scan();
  int l,r,x;
  REP(i,q) {
    l=Scan();r=Scan();x=Scan();
    int d=id(x+1);
    if(d>=cnt) {
      putchar('0');
    }else{
      Out(query(root[r], root[l-1], 0, cnt-1, d));
    }
    putchar('\n');
  }
```

```
  return 0;
}
```

2.4 Tree Division by Point

```
int head[MAXN];
int Next[MAXN];
int e[MAXN];
int D[MAXN];
int vis[MAXN];
int del[MAXN];
int n_edge;
void addEdge(int u, int v, int d) {
  Next[n_edge] = head[u];
  e[n_edge] = v;
  D[n_edge] = d;
  head[u] = n_edge++;
}
int fpoint;
int midpoint;
int mx;
int color;
void dfs(int u, int dep = 0) {
  vis[u] = color;
  if(dep > mx) {
    mx = dep;
    fpoint = u;
    midpoint = -1;
  }
  for(int i = head[u]; ~i; i = Next[i]) {
    int v = e[i];
    if(!del[v] && vis[v] != color) dfs(v, dep+1);
  }
  if(dep == (mx+1) / 2 && midpoint == -1) {
    midpoint = u;
  }
}
int getp(int u) {
  color++; mx = -1;
  dfs(u);
  color++; mx = -1;
  dfs(fpoint);
  return midpoint;
}
void solve(int u) {
  u = getp(u);
  del[u] = 1;
  for(int i = head[u]; ~i; i = Next[i]) {
    int v = e[i];
    if(!del[v]) {
      solve(v);
      getdis(v);
      // calculate
    }
  }
  del[u] = 0;
}
```

2.5 K-d Tree

维护二维单点最小值

```
#define MAXN 100010
#define D 2

int d[MAXN][D];
int sp[MAXN];
int id[MAXN];
int mnx[MAXN], mxx[MAXN], mny[MAXN], mxy[MAXN];
int ls[MAXN], rs[MAXN];

int dim;
int build(int L, int R) {
  if(L > R) return -1;
  int M = (L + R) >> 1;
  dim = 0;
  double mx = 0;
  REP(i,D) {
    double avg = 0;
    REP2(j,L,R)
      avg += d[id[j]][i];
    avg /= R-L+1;
    double m = 0;
    REP2(j,L,R)
      m += sqr(d[id[j]][i] - avg);
    if(m >= mx) {
      mx = m;
      dim = i;
    }
  }
  sp[M] = dim;
  nth_element(id+L, id+M, id+R+1, [](int x, int y){return
  d[x][dim] < d[y][dim];});

  ls[M] = build(L, M-1);
  rs[M] = build(M+1, R);

  mnx[M] = mxx[M] = d[id[M]][0];
  mny[M] = mxy[M] = d[id[M]][1];

  if(ls[M] != -1) {
    mnx[M] = min(mnx[M], mnx[ls[M]]);
    mny[M] = min(mny[M], mny[ls[M]]);
    mxx[M] = max(mxx[M], mxx[ls[M]]);
    mxy[M] = max(mxy[M], mxy[ls[M]]);
  }
  if(rs[M] != -1) {
    mnx[M] = min(mnx[M], mnx[rs[M]]);
    mny[M] = min(mny[M], mny[rs[M]]);
    mxx[M] = max(mxx[M], mxx[rs[M]]);
    mxy[M] = max(mxy[M], mxy[rs[M]]);
  }

  return M;
}

ll add[MAXN];
ll hmin[MAXN];
ll padd[MAXN];
ll pmin[MAXN];

void pushdown(int t) {
  if(~ls[t]) hmin[ls[t]] = min(hmin[ls[t]], add[ls[t]] +
  hmin[t]);
  if(~rs[t]) hmin[rs[t]] = min(hmin[rs[t]], add[rs[t]] +
  hmin[t]);
  pmin[t] = min(pmin[t], padd[t] + hmin[t]);
  hmin[t] = 0;

  if(~ls[t]) add[ls[t]] += add[t];
  if(~rs[t]) add[rs[t]] += add[t];
  padd[t] += add[t];
  add[t] = 0;
}

void update(int L, int R, int P, int x) {
  if(L > R) return;
  int M = (L + R) >> 1;
  if(P < mnx[M] || P > mxy[M]) return ;
  if(P >= mxx[M] && P <= mny[M]) {
    hmin[M] = min(hmin[M], add[M] += x);
    return ;
  }
  pushdown(M);
    if(P >= d[id[M]][0] && P <= d[id[M]][1]) {
      pmin[M] = min(pmin[M], padd[M] += x);
    }
  update(L, M-1, P, x);
  update(M+1, R, P, x);
}

ll query(int L, int R, int p) {
  int M = (L + R) >> 1;
  pushdown(M);
  if(p == M) {
    return pmin[M];
  }
  if(p < M) return query(L, M-1, p);
  else return query(M+1, R, p);
}
//REP(i,cnt) idx[id[i]] = i;
//query(0, cnt-1, idx[i])


//维护最近点
template<typename T> T sqr(T a) {return a*a;}
```

```

int d[MAXN][D];
int sp[MAXN];
int id[MAXN];

int dim;
void build(int L, int R) {
  if(L > R) return;
  int M = (L + R) >> 1;
  dim = 0;
  double mx = 0;
  REP(i,D) {
    double avg = 0;
    REP2(j,L,R)
      avg += d[id[j]][i];
    avg /= R-L+1;
    double m = 0;
    REP2(j,L,R)
      m += sqr(d[id[j]][i] - avg);
    if(m >= mx) {
      mx = m;
      dim = i;
    }
  }
  sp[M] = dim;
  nth_element(id+L, id+M, id+R+1, [](int x, int y){return
  d[x][dim] < d[y][dim];});

  build(L, M-1);
  build(M+1, R);
}

ll mindis;
int minid;
int use[MAXN]; //表示暂时删除该点
void query(int Q[D], int L, int R) {
  if(L > R) return;
  int M = (L + R) >> 1;

  ll dis = 0;
  REP(i,D)
    dis += sqr<ll>(d[id[M]][i] - Q[i]);

  if(!use[id[M]]) {
    if(mindis > dis) {
      mindis = dis;
      minid = id[M];
    }
  }

  ll rad = sqr<ll>(d[id[M]][sp[M]] - Q[sp[M]]);

  if(d[id[M]][sp[M]] > Q[sp[M]]) {
    query(Q, L, M-1);
    if(rad < mindis) query(Q, M+1, R);
  }else{
    query(Q, M+1, R);
    if(rad < mindis) query(Q, L, M-1);
  }
}
```

3 Graph

3.1 Min Cost Flow

```

final int INF = Integer.MAX_VALUE / 4;

int minCostFlow(V[] vs, V s, V t, int flow) {
  int res = 0;
  while (flow > 0) {
    for (V v : vs) v.min = INF;
    PriorityQueue<E> que = new PriorityQueue<E>();
    s.min = 0;
    que.offer(new E(s, 0, 0));
    while (!que.isEmpty()) {
      E crt = que.poll();
      if (crt.cost == crt.to.min) {
        for (E e : crt.to.es) {
          int tmp = crt.cost + e.cost + crt.to.h - e.to.h;
          if (e.cap > 0 && e.to.min > tmp) {
            e.to.min = tmp;
            e.to.prev = e;
            que.offer(new E(e.to, 0, e.to.min));
          }
        }
      }
    }
    if (t.min == INF) return INF;
    int d = flow;
    for (E e = t.prev; e != null; e = e.rev.to.prev) {
      d = Math.min(d, e.cap);
    }
    for (E e = t.prev; e != null; e = e.rev.to.prev) {
      res += d * e.cost;
      e.cap -= d;
      e.rev.cap += d;
    }
    flow -= d;
    for (V v : vs) v.h += v.min;
  }
  return res;
}
class V {
  ArrayList<E> es = new ArrayList<E>();
  E prev;
  int min;
  int h;

  void add(V to, int cap, int cost) {
    E e = new E(to, cap, cost), rev = new E(this, 0, -cost
    );
    e.rev = rev;
    rev.rev = e;
    es.add(e);
    to.es.add(rev);
  }
}
class E implements Comparable<E> {
  V to;
  E rev;
  int cap;
  int cost;

  E(V to, int cap, int cost) {
    this.to = to;
    this.cap = cap;
    this.cost = cost;
  }
  int compareTo(E o) {
    return cost - o.cost;
  }
}
```

3.2 Global Min Cut

$O(n^3)$

```

final int INF = Integer.MAX_VALUE / 4;
int minCut(int[][] c) {
  int n = c.length, cut = INF;
  int[] id = new int[n], b = new int[n];
  for (int i = 0; i < n; i++) id[i] = i;
  for (; n > 1; n--) {
    Arrays.fill(b, 0);
    for (int i = 0; i + 1 < n; i++) {
      int p = i + 1;
      for (int j = i + 1; j < n; j++) {
        b[id[j]] += c[id[i]][id[j]];
        if (b[id[p]] < b[id[j]]) p = j;
      }
      swap(id, i + 1, p);
    }
    cut = Math.min(cut, b[id[n - 1]]);
    for (int i = 0; i < n - 2; i++) {
      c[id[i]][id[n - 2]] += c[id[i]][id[n - 1]];
      c[id[n - 2]][id[i]] += c[id[n - 1]][id[i]];
    }
  }
  return cut;
}
```

3.3 Dinic

```

final int INF = Integer.MAX_VALUE / 4;
int p = 0;
```

```
int dinic(V s, V t) {
  int flow = 0;
  for (p++; ; p++) {
    Queue<V> que = new LinkedList<V>();
    s.level = 0;
    s.p = p;
    que.offer(s);
    while (!que.isEmpty()) {
      V v = que.poll();
      v.iter = v.es.size() - 1;
      for (E e : v.es)
        if (e.to.p < p && e.cap > 0) {
          e.to.level = v.level + 1;
          e.to.p = p;
          que.offer(e.to);
        }
    }
    if (t.p < p) return flow;
    for (int f; (f = dfs(s, t, INF)) > 0; ) flow += f;
  }
}
int dfs(V v, V t, int f) {
  if (v == t) return f;
  for (; v.iter >= 0; v.iter--) {
    E e = v.es.get(v.iter);
    if (v.level < e.to.level && e.cap > 0) {
      int d = dfs(e.to, t, Math.min(f, e.cap));
      if (d > 0) {
        e.cap -= d;
        e.rev.cap += d;
        return d;
      }
    }
  }
  return 0;
}
class V {
  ArrayList<E> es = new ArrayList<E>();
  int level;
  int p;
  int iter;
  void add(V to, int cap) {
    E e = new E(to, cap), rev = new E(this, 0);
    e.rev = rev;
    rev.rev = e;
    es.add(e);
    to.es.add(rev);
  }
}
class E {
  V to;
  E rev;
  int cap;
}
```

3.4 SCC

```
int n;
V[] us;
int scc(V[] vs) {
  n = vs.length;
  us = new V[n];
  for (V v : vs) if (!v.visit) dfs(v);
  for (V v : vs) v.visit = false;
  for (V u : us) if (!u.visit) dfsRev(u, n++);
  return n;
}
void dfs(V v) {
  v.visit = true;
  for (V u : v.fs) if (!u.visit) dfs(u);
  us[--n] = v;
}
void dfsRev(V v, int k) {
  v.visit = true;
  for (V u : v.rs) if (!u.visit) dfsRev(u, k);
  v.comp = k;
}
class V {
  boolean visit;
  int comp;
  List<V> fs = new LinkedList<V>();
  List<V> rs = new LinkedList<V>();
  void add(V u) {
    fs.add(u);
    u.rs.add(this);
  }
}
```

3.5 Bipartite Matching

```
int bipartiteMatching(V[] vs) {
  int match = 0;
  for (V v : vs)
    if (v.pair == null) {
      for (V u : vs) u.used = false;
      if (dfs(v)) match++;
    }
  return match;
}
boolean dfs(V v) {
  v.used = true;
  for (V u : v.vs) {
    u.used = true;
    V w = u.pair;
    if (w == null || !w.used && dfs(w)) {
      v.pair = u;
      u.pair = v;
      return true;
    }
  }
  return false;
}
class V {
  List<V> vs = new ArrayList<V>();
  V pair;
  boolean used;
  void connect(V v) {
    vs.add(v);
    v.vs.add(this);
  }
}
```

3.6 Hopcroft Karp

```
int hopcroftKarp(V[] vs) {
  for (int match = 0; ; ) {
    Queue<V> que = new LinkedList<V>();
    for (V v : vs) v.level = -1;
    for (V v : vs)
      if (v.pair == null) {
        v.level = 0;
        que.offer(v);
      }
    while (!que.isEmpty()) {
      V v = que.poll();
      for (V u : v.vs) {
        V w = u.pair;
        if (w != null && w.level < 0) {
          w.level = v.level + 1;
          que.offer(w);
        }
      }
    }
    for (V v : vs) v.used = false;
    int d = 0;
    for (V v : vs) if (v.pair == null && dfs(v)) d++;
    if (d == 0) return match;
    match += d;
  }
}
boolean dfs(V v) {
  v.used = true;
  for (V u : v.vs) {
    V w = u.pair;
    if (w == null || !w.used && v.level < w.level && dfs(w
    )) {
      v.pair = u;
      u.pair = v;
      return true;
    }
  }
  return false;
}
class V {
  List<V> vs = new ArrayList<V>();
```

```
  V pair;
  boolean used;
  int level;
  void connect(V v) {
    vs.add(v);
    v.vs.add(this);
  }
}
```

3.7 General Max Matching

```
typedef unsigned int uint;
uint xrand() {
  static uint x = 314159265, y = 358979323, z = 846264338, w = 327950288;
  uint t = x ^ x << 11; x = y; y = z; z = w;
  return w = w ^ w >> 19 ^ t ^ t >> 8;
}
namespace MM { // O(N^3)
#define MAXN 222
#define adj(i, j) _adj[tr[i]][tr[j]]
#define et(i, j) _et[tr[i]][tr[j]]
  const double EPS = 1e-10;
  inline bool zero(double r) { return (-EPS <= r && r <= EPS); }
  int n, rank, tr[MAXN];
  bool _adj[MAXN][MAXN];
  double _et[MAXN][MAXN];
  inline void init(int _n) {
    int i, j;
    n = _n;
    for (i = 0; i < n; ++i) tr[i] = i;
    for (i = 0; i < n; ++i) for (j = 0; j < n; ++j) adj(i, j) = 0;
  }
  inline void ae(int i, int j) {
    adj(j, i) = adj(i, j) = 1;
  }
  int calc() { // return max matching
    int h, i, j, im;
    double r, s;
    for (i = 0; i < n; ++i) et(i, i) = 0;
    for (i = 0; i < n; ++i) for (j = i + 1; j < n; ++j) {
      et(j, i) = -(et(i, j) = adj(i, j) ?
          (1.0 + xrand()) / 4294967296.0 : 0);
    }
    for (rank = n, h = 0; h < rank; ++h) {
      if (zero(et(h, h))) {
        for (im = h, i = h + 1; i < rank; ++i)
          if (abs(et(im, h)) < abs(et(i, h))) im = i;
        if (zero(et(im, h))) {
          swap(tr[h--], tr[--rank]);
          continue;
        }
        for (j = h; j < rank; ++j) et(h, j) += et(im, j);
      }
      r = 1 / et(h, h);
      for (j = h; j < rank; ++j) et(h, j) *= r;
      for (i = h + 1; i < rank; ++i) {
        s = et(i, h);
        for (j = h; j < rank; ++j) et(i, j) -= s * et(h, j);
      }
    }
    return rank;
  }
}
```

3.8 Bridge

```
List<E> bridge;
List<E> connection(V[] vs) {
  bridge = new ArrayList<E>();
  for (V v : vs) if (v.num < 0) {
    dfs(v, 0);
    if (v.count > 0) v.count--;
  }
  return bridge;
}
int dfs(V v, int c) {
  v.num = c;
  int low = c;
  boolean rev = false;
  for (V u : v.vs) {
    if (u.num < 0) {
      int t = dfs(u, c + 1);
      low = Math.min(low, t);
      if (v.num <= t) v.count++;
      if (v.num < t) bridge.add(new E(v, u));
    } else if (u.num != v.num - 1 || rev)
      low = Math.min(low, u.num);
    else rev = true;
  }
  return low;
}
class V {
  List<V> vs = new ArrayList<V>();
  int num = -1;
  int count;
  void connect(V u) {
    vs.add(u);
    u.vs.add(this);
  }
}
class E {
  V u;
  V v;
}
```

3.9 SPFA

```
void spfa(V s) {
  Queue<V> que = new LinkedList<V>();
  s.min = 0;
  que.offer(s);
  while (!que.isEmpty()) {
    V crt = que.poll();
    crt.inQue = false;
    for (E e : crt.es) {
      if (crt.min + e.cost < e.to.min) {
        e.to.min = crt.min + e.cost;
        if (!e.to.inQue) {
          e.to.inQue = true;
          que.offer(e.to);
        }
      }
    }
  }
}
final int INF = 1 << 29;
class V {
  ArrayList<E> es = new ArrayList<E>();
  int min = INF;
  boolean inQue = false;
  void add(V to, int cost) {
    es.add(new E(to, cost));
  }
}
class E {
  V to;
  int cost;
}
```

3.10 Dijkstra

```
void dijkstra(V s) {
  PriorityQueue<E> que = new PriorityQueue<E>();
  s.min = 0;
  que.offer(new E(s, 0));
  while (!que.isEmpty()) {
    E crt = que.poll();
    if (crt.cost > crt.to.min) continue;
    for (E e : crt.to.es) {
      if (crt.cost + e.cost < e.to.min) {
        e.to.min = crt.cost + e.cost;
        que.offer(new E(e.to, e.to.min));
      }
    }
  }
}
final int INF = 1 << 29;
class V {
  ArrayList<E> es = new ArrayList<E>();
  int min = INF;
```

```
  void add(V to, int cost) {
    es.add(new E(to, cost));
  }
}
class E implements Comparable<E> {
  V to;
  int cost;
  int compareTo(E o) {
    return cost - o.cost;
  }
}
```

3.11 Stenier

$O(3^t n + 2^t n^2 + n^3)$

```
int steiner(int[][] g, int[] ts) {
  int n = g.length, m = ts.length;
  if (m < 2) return 0;
  int[][] dp = new int[1 << m][n];
  for (int k = 0; k < n; k++) {
    for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
        g[i][j] = min(g[i][j], g[i][k] + g[k][j]);
      }
    }
  }
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
      dp[1 << i][j] = g[ts[i]][j];
    }
  }
  for (int i = 1; i < 1 << m; i++) if (((i - 1) & i) != 0)
   {
    for (int j = 0; j < n; j++) {
      dp[i][j] = INF;
      for (int k = (i - 1) & i; k > 0; k = (k - 1) & i) {
        dp[i][j] = min(dp[i][j], dp[k][j] + dp[i ^ k][j]);
      }
    }
    for (int j = 0; j < n; j++) {
      for (int k = 0; k < n; k++) {
        dp[i][j] = min(dp[i][j], dp[i][k] + g[k][j]);
      }
    }
  }
  return dp[(1 << m) - 1][ts[0]];
}
```

3.12 Direct MST

$O(VE)$

```
int arborescence(V[] vs, V r) {
  int res = 0;
  for (V v : vs) for (E e : v.es) e.to.min = min(e.to.min,
   e.cost);
  for (V v : vs) if (v != r) {
    if (v.min == INF) return INF;
    res += v.min;
  }
  for (V v : vs) for (E e : v.es) if (e.to != r) {
    e.cost -= e.to.min;
    if (e.cost == 0) {
      v.fs.add(e.to);
      e.to.rs.add(v);
    }
  }
  int m = scc(vs);
  if (m == vs.length) return res;
  V[] us = new V[m];
  for (int i = 0; i < m; i++) us[i] = new V();
  for (V v : vs) for (E e : v.es) {
    if (v.comp != e.to.comp) us[v.comp].add(us[e.to.comp],
     e.cost);
  }
  return min(INF, res + arborescence(us, us[r.comp]));
}
```

4 Graph G

4.1 ISAP

```
#define MAXN 100010
#define MAXE 200010
struct Edge{
  int u, v;
  ll c, f;
  Edge(){};
  Edge(int _u, int _v, ll _c, ll _f) {
    u=_u; v=_v; c=_c; f=_f;
  }
}edge[MAXE*2];
int n_edge;
VI Map[MAXN];
int last[MAXN];
void init() {
  REP(i,MAXN) Map[i].clear();
  n_edge = 0;
}
int d[MAXN];
int gap[MAXN];
int bfs(int s, int t) {
  int n = 0;
  queue<int> q;
  REP(i,MAXN) d[i] = INF;
  REP(i,MAXN) gap[i] = 0;
  d[t] = 0;
  q.push(t);
  while(!q.empty()) {
    int u = q.front();
    q.pop();
    gap[d[u]]++;
    n++;
    REP(i,Map[u].size()) {
      int e=Map[u][i];
      int v=edge[e].v;
      if(edge[e^1].f < edge[e^1].c && d[v] == INF) {
        d[v] = d[u] + 1;
        q.push(v);
      }
    }
  }
  return n;
}
ll augument(int t) {
  int e = last[t];
  ll fl = INF;
  while(e != -1) {
    int v = edge[e].u;
    fl = min(fl, edge[e].c - edge[e].f);
    e = last[v];
  }
  e = last[t];
  while(e != -1) {
    int v = edge[e].u;
    edge[e].f += fl;
    edge[e^1].f -= fl;
    e = last[v];
  }
  return fl;
}
ll ISAP(int s, int t) {
  ll res = 0;
  int n = bfs(s, t);
  last[s] = -1;
  int u = s;
  while(d[s] < n) {
    if(u == t) {
      res += augument(t);
      u = s;
    }
    int f = 0;
    REP(i, Map[u].size()) {
      int e = Map[u][i];
      int v = edge[e].v;
      if(edge[e].f < edge[e].c && d[u] == d[v] + 1) { // advance
        u = v;
        last[v] = e;
        f = 1;
        break;
      }
    }
    if(!f) { // retreat
```

```
      int _min = n;
      REP(i,Map[u].size()) {
        int e = Map[u][i];
        int v = edge[e].v;
        if(edge[e].f < edge[e].c) _min = min(_min, d[v]);
      }
      if(--gap[d[u]]==0) break;
      d[u] = _min+1;
      gap[d[u]]++;
      if(u != s) u = edge[last[u]].u;
    }
  }
  return res;
}

void addEdge(int u, int v, ll c) {
  edge[n_edge] = Edge(u,v,c,0);
  Map[u].PB(n_edge++);
  edge[n_edge] = Edge(v,u,0,0);
  Map[v].PB(n_edge++);
}
```

4.2 Kosaraju

```
VI Map[MAXN], rMap[MAXN], vex;
int sgn[MAXN], scc;
void dfs(int u) {
  sgn[u] = 1;
  REP(i,Map[u].size()) {
    int v = Map[u][i];
    if(!sgn[v]) dfs(v);
  }
  vex.PB(v);
}
void rdfs(int u) {
  sgn[u] = scc;
  REP(i,rMap[u].size()) {
    int v = rMap[u][i];
    if(!sgn[v]) rdfs(v);
  }
}
void kosaraju() {
  CLR(sgn, 0);
  vex.clear();
  REP(i,n) if(!sgn[i]) dfs(i);
  CLR(sgn, 0);
  reverse(vex.begin(), vex.end());
  REP(i,vex.size()) if(!sgn[vex[i]]) {
    scc++;
    rdfs(vex[i]);
  }
}
```

4.3 Tarjan

```
vector<pair<int> > Map[MAXN];
vector<pair<PII> > bri;
int low[MAXN], dfn[MAXN], cut[MAXN];
int dep;
void tarjan(int u, int fa=-1)
{
  low[u] = dfn[u] = dep++;
  for(auto it = Map[u].begin(); it != Map[u].end(); it++)
  {
    int v = it->X;
    if(v==fa) continue;
    int son = 0;
    if(dfn[v] == -1)
    {
      tarjan(v, u);
      son++;
      low[u] = min(low[u], low[v]);
      //cut point
      if((fa<0 && son>1) || (fa >=0&&low[v]>=dfn[v])) cut[
      u]=1;
      //bridge
      if(low[v] > dfn[u])
        bri.push_back(make_pair(u,v));
    }else{
      low[u] = min(low[u], dfn[v]);
    }
  }
}
```

4.4 Union Find

```
int pa[MAXN], ra[MAXN];
void init(int n) {
  REP(i,n)  {
    pa[i] = i;
    ra[i] = 0;
  }
}
int find(int x) {
  if(pa[x]!=x) pa[x] = find(pa[x]);
  return pa[x];
}
// 0: already united;  1: successfully united;
int unite(int x, int y) {
  x = find(x);
  y = find(y);
  if(x==y) return 0;
  if(ra[x] < ra[y]) {
    pa[x] = y;
  }else{
    pa[y] = x;
    if(ra[x] == ra[y]) ra[x]++;
  }
  return 1;
}
```

5 String

5.1 Hash

```
class Hash {
  final long BASE = (long) (1e9 + 7);
  long[] ps;

  Hash(int n) {
    ps = new long[n + 1];
    for (int i = 0; i <= n; i++)
      ps[i] = (i == 0 ? 1 : ps[i - 1] * BASE);
  }
  long[] build(char[] cs) {
    int n = cs.length;
    long[] hs = new long[n + 1];
    for (int i = 0; i < n; i++)
      hs[i + 1] = hs[i] * BASE + cs[i];
    return hs;
  }
  long get(long[] hs, int b, int e) {
    return hs[e] - hs[b] * ps[e - b];
  }
}
```

5.2 Hash2

```
class Hash2 {
  Random RND = new Random(System.nanoTime());
  long BL = (long) (1e9 + RND.nextInt((int) 1e9));
  long BR = (long) (1e9 + RND.nextInt((int) 1e9));
  long ML = (long) (1e9 + RND.nextInt((int) 1e9));
  long MR = (long) (1e9 + RND.nextInt((int) 1e9));
  long[] psl;
  long[] psr;

  Hash2(int n) {
    psl = new long[n + 1];
    psr = new long[n + 1];
    for (int i = 0; i <= n; i++)
      psl[i] = (i == 0 ? 1 : psl[i - 1] * BL) % ML;
    for (int i = 0; i <= n; i++)
      psr[i] = (i == 0 ? 1 : psr[i - 1] * BR) % MR;
  }
  long[] build(char[] cs) {
    int n = cs.length;
    long[] hs = new long[n + 1];
    long l = 0, r = 0;
    for (int i = 0; i < n; i++) {
      l = (l * BL + cs[i]) % ML;
      r = (r * BR + cs[i]) % MR;
      if (l < 0) l += ML;
```

```
      if (r < 0) r += MR;
      hs[i + 1] = (l << 32) | r;
    }
    return hs;
  }
  long get(long[] hs, int b, int e) {
    long el = hs[e] >>> 32;
    long er = hs[e] & 0xffffffffL;
    long bl = hs[b] >>> 32;
    long br = hs[b] & 0xffffffffL;
    long l = el - bl * psl[e - b] % ML;
    long r = er - br * psr[e - b] % MR;
    if (l < 0) l += ML;
    if (r < 0) r += MR;
    return (l << 32) | r;
  }
}
```

5.3 Manacher

```
[a   c   c   a   b   a] ->
[1 0 1 4 1 0 1 0 3 0 1 0]
int[] manacher(char[] cs) {
  int n = cs.length;
  int[] len = new int[n * 2];
  for (int i = 0, j = 0, k;
      i < n * 2;
      i += k, j = Math.max(j - k, 0)) {
    while (i - j >= 0 && i + j + 1 < n * 2
        && cs[(i - j) / 2] == cs[(i + j + 1) / 2]) j++;
    len[i] = j;
    for (k = 1;
        i - k >= 0 && j - k >= 0 && len[i - k] != j - k;
        k++) {
      len[i + k] = Math.min(len[i - k], j - k);
    }
  }
  return len;
}
```

5.4 Suffix Tree

```
String ALPHABET = "$abcdefghijklmnopqrstuvwxyz\1\2";
class Node {
  int begin;
  int end;
  int depth; // distance in characters from tree root to
   this node
  Node parent;
  Node[] children;
  Node suffixLink; // null means link to root

  Node(int begin, int end, int depth, Node parent) {
    children = new Node[ALPHABET.length()];
    this.begin = begin;
    this.end = end;
    this.parent = parent;
    this.depth = depth;
  }
  boolean contains(int d) {
    return depth <= d && d < depth + (end - begin);
  }
}
Node buildSuffixTree(CharSequence s) {
  int n = s.length();
  byte[] a = new byte[n];
  for (int i = 0; i < n; i++)
    a[i] = (byte) ALPHABET.indexOf(s.charAt(i));
  Node root = new Node(0, 0, 0, null);
  Node cn = root;
  // root.suffixLink must be null, but that way it gets
   more convenient processing
  root.suffixLink = root;
  Node needsSuffixLink = null;
  int lastRule = 0;
  int j = 0;
  for (int i = -1; i < n - 1; i++) {// strings s[j..i] are
   already in tree, add s[i+1] to it
    int cur = a[i + 1]; // last char of current string
    for (; j <= i + 1; j++) {
      int curDepth = i + 1 - j;
      if (lastRule != 3) {
        cn = cn.suffixLink != null ? cn.suffixLink : cn.
        parent.suffixLink;
        int k = j + cn.depth;
        while (curDepth > 0 && !cn.contains(curDepth - 1))
         {
          k += cn.end - cn.begin;
          cn = cn.children[a[k]];
        }
      }
      if (!cn.contains(curDepth)) { // explicit node
        if (needsSuffixLink != null) {
          needsSuffixLink.suffixLink = cn;
          needsSuffixLink = null;
        }
        if (cn.children[cur] == null) {
          // no extension - add leaf
          cn.children[cur] = new Node(i + 1, n, curDepth,
          cn);
          lastRule = 2;
        } else {
          cn = cn.children[cur];
          lastRule = 3; // already exists
          break;
        }
      } else { // implicit node
        int end = cn.begin + curDepth - cn.depth;
        if (a[end] != cur) { // split implicit node here
          Node newn = new Node(cn.begin, end, cn.depth, cn
          .parent);
          newn.children[cur] = new Node(i + 1, n, curDepth
          , newn);
          newn.children[a[end]] = cn;
          cn.parent.children[a[cn.begin]] = newn;
          if (needsSuffixLink != null) {
            needsSuffixLink.suffixLink = newn;
          }
          cn.begin = end;
          cn.depth = curDepth;
          cn.parent = newn;
          cn = needsSuffixLink = newn;
          lastRule = 2;
        } else if (cn.end != n || cn.begin - cn.depth < j)
         {
          lastRule = 3;
          break;
        } else {
          lastRule = 1;
        }
      }
    }
  }
  root.suffixLink = null;
  return root;
}
```

5.5 Aho Corasick

```
class AhoCorasick {
  int m;
  Node root;
  AhoCorasick(char[][] ps) {
    m = ps.length;
    root = new Node();
    for (int i = 0; i < m; i++) {
      Node t = root;
      for (char c : ps[i]) {
        if (!t.cs.containsKey(c))
          t.cs.put(c, new Node());
        t = t.cs.get(c);
      }
      t.accept.add(i);
    }
    Queue<Node> que = new LinkedList<>();
    que.offer(root);
    while (!que.isEmpty()) {
      Node t = que.poll();
      for (Map.Entry<Character, Node> e : t.cs.entrySet())
       {
        char c = e.getKey();
        Node u = e.getValue();
        que.offer(u);
        Node r = t.fail;
        while (r != null && !r.cs.containsKey(c))
```

```
        r = r.fail;
      if (r == null) u.fail = root;
      else u.fail = r.cs.get(c);
      u.accept.addAll(u.fail.accept);
    }
  }
}
Map<Node, Integer> getNodeIndex(List<Node> ns) {
  Map<Node, Integer> index = new HashMap<>();
  Queue<Node> que = new LinkedList<>();
  que.add(root);
  int crt = 0;
  while (!que.isEmpty()) {
    Node t = que.poll();
    ns.add(t);
    index.put(t, crt++);
    que.addAll(t.cs.values());
  }
  return index;
}
int[] searchFrom(char[] t) {
  int n = t.length;
  int[] count = new int[m];
  Node u = root;
  for (int i = 0; i < n; i++) {
    while (u != null && !u.cs.containsKey(t[i]))
      u = u.fail;
    if (u == null) u = root;
    else u = u.cs.get(t[i]);
    for (int j : u.accept) count[j]++;
  }
  return count;
}
static class Node {
  Map<Character, Node> cs = new TreeMap<>();
  List<Integer> accept = new ArrayList<>();
  Node fail;
}
}
```

5.6 KMP

```
class KMP {
  int m;
  char[] p;
  int[] fail;
  KMP(char[] p) {
    m = p.length;
    this.p = p;
    fail = new int[m + 1];
    int crt = fail[0] = -1;
    for (int i = 1; i <= m; i++) {
      while (crt >= 0 && p[crt] != p[i - 1])
        crt = fail[crt];
      fail[i] = ++crt;
    }
  }
  int searchFrom(char[] t) {
    int n = t.length, count = 0;
    for (int i = 0, j = 0; i < n; i++) {
      while (j >= 0 && t[i] != p[j])
        j = fail[j];
      if (++j == m) {
        count++;
        j = fail[j];
      }
    }
    return count;
  }
}
```

6 String G

6.1 Palindromic Tree

```
const int MAXN = 100005;
const int N = 26;
struct Palindromic_Tree {
  int next[MAXN][N]; next指针，next指针和字典树类似，指向的串为当前串两端加上同一个字符构成
  int fail[MAXN]; fail指针，失配后跳转到fail指针指向的节点
  int cnt[MAXN]; 节点i表示的本质不同的串的个数(count()后)
  int num[MAXN]; 以节点i表示的最长回文串的最右端点为回文串结尾的回文串个数。
  int len[MAXN]; len[i]表示节点i表示的回文串的长度
  int S[MAXN]; 存放添加的字符
  int last; 指向上一个字符所在的节点，方便下一次add
  int n; 字符数组指针
  int p; 节点指针
  int newnode(int l) {
    for (int i = 0 ; i < N ; ++i) next[p][i] = 0;
    cnt[p] = 0;
    num[p] = 0;
    len[p] = l;
    return p ++;
  }
  void init() {
    p = 0;
    newnode(0);
    newnode(-1);
    last = 0;
    n = 0;
    S[n] = -1;
    fail[0] = 1;
  }
  int get_fail(int x) {
    while(S[n - len[x] - 1] != S[n] ) x = fail[x];
    return x;
  }
  void add (int c ) {
    c -= 'a';
    S[++ n] = c;
    int cur = get_fail(last);
    if (!next[cur][c]) {
      int now = newnode(len[cur] + 2);
      fail[now] = next[get_fail(fail[cur])][c];
      next[cur][c] = now;
      num[now] = num[fail[now]] + 1;
    }
    last = next[cur][c];
    cnt[last]++;
  }
  void count() {
    for (int i = p - 1; i >= 0; --i) cnt[fail[i]] += cnt[i];
  }
};
```

6.2 Suffix Automaton

```
struct State {
  State *par, *go[26];
  int val;
  State(){};
  State(int _val) : par(NULL), val(_val) {
    memset(go,0,sizeof(go));
  }
}*root, *last, buffer[200000];
int p;
State* newState(int val) {
  return &(buffer[p++] = State(val));
}
void init() {
  p = 0;
  root = last = newState(0);
}
void extend(int w) {
  State *p = last;
  State *np = newState(p->val + 1);
  while(p && p->go[w] == 0)
    p->go[w] = np, p = p->par;
  if(p == 0)
    np->par = root;
  else {
    State *q = p->go[w];
    if(p->val + 1 == q->val) {
      np->par = q;
    } else {
      State *nq = newState(p->val + 1);
      memcpy(nq->go, q->go, sizeof q->go);
      nq->par = q->par;
      q->par = nq;
      np->par = nq;
      while(p && p->go[w] == q)
```

```
      p->go[w] = nq, p = p->par;
    }
  }
  last = np;
}
```

6.3 Suffix Array

```
/* Usage
  str[n] = 0;
  dc3(str,sa,n+1,200);
  calheight(str,sa,n);
  //sa[]: 第i大的字符串的起始位置
  //ra[]: 起始位置为i的字符串的rank
  //height[]: lcp(sa[i], sa[i+1])
*/
#define F(x) ((x)/3+((x)%3==1?0:tb))
#define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
int wa[MAXN],wb[MAXN],wv[MAXN],ww[MAXN];
int c0(int *r, int a, int b) {
  return r[a]==r[b]&&r[a+1]==r[b+1]&&r[a+2]==r[b+2];
}
int c12(int k, int *r, int a, int b) {
  if(k==2) return r[a]<r[b]||r[a]==r[b]&&c12(1,r,a+1,b+1);
  else return r[a]<r[b]||r[a]==r[b]&&wv[a+1]<wv[b+1];
}
void rsort(int *r, int *a, int *b, int n, int m) {
  REP(i,n) wv[i]=r[a[i]];
  REP(i,m) ww[i]=0;
  REP(i,n) ww[wv[i]]++;
  REP(i,m-1) ww[i+1]+=ww[i];
  DEP(i,n-1,0) b[--ww[wv[i]]]=a[i];
}
void dc3(int *r, int *sa, int n, int m) {
  int j,*rn=r+n,*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;
  r[n]=r[n+1]=0;
  REP(i,n) if(i%3!=0) wa[tbc++]=i;
  rsort(r+2,wa,wb,tbc,m);
  rsort(r+1,wb,wa,tbc,m);
  rsort(r,wa,wb,tbc,m);
  for(p=1,rn[F(wb[0])]=0,j=1;j<tbc;j++)
    rn[F(wb[j])]=c0(r,wb[j-1],wb[j])?p-1:p++;
  if(p<tbc) dc3(rn,san,tbc,p);
  else REP(i,tbc) san[rn[i]]=i;
  REP(i,tbc) if(san[i]<tb) wb[ta++]=san[i]*3;
  if(n%3==1) wb[ta++]=n-1;
  rsort(r,wb,wa,ta,m);
  REP(i,tbc) wv[wb[i]=G(san[i])]=i;
  int i;
  for(i=j=p=0;i<ta&&j<tbc;p++)
    sa[p]=c12(wb[j]%3,r,wa[i],wb[j])?wa[i++]:wb[j++];
  for(;i<ta;p++) sa[p]=wa[i++];
  for(;j<tbc;p++) sa[p]=wb[j++];
}
int ra[MAXN], height[MAXN];
void calheight(int *r,int *sa,int n) {
  int i,j,k=0;
  for(i=1;i<=n;i++) ra[sa[i]]=i;
  for(i=0;i<n;height[ra[i++]]=k)
    for(k?k--:0,j=sa[ra[i]-1];r[i+k]==r[j+k];k++);
}
```

6.4 KMP

```
vector<int> pre_kmp(string & s) {
  vector<int> f;
  f.push_back(-1);
  int k = -1;
  for(int i = 0; i < s.size(); i++) {
    if(k == -1 || s[i] == s[k]) {
      k++;
      f.push_back(k);
    }else{
      k = f[k];
      i--;
    }
  }
  return f;
}

int kmp(string& text, string& pattern, vector<int>& f) {
  int i = 0, j = 0, c = 0;
  while(i < text.size()) {
    if(j == -1 || text[i] == pattern[j]) {
      i++; j++;
      if(j == pattern.size()) c++, j = f[j];
    }else{
      while(j != -1 && text[i] != pattern[j])
        j = f[j];
    }
  }
  return c;
}
```

6.5 exKMP

```
//exKMP(s, s1, strlen(s), strlen(s1), _next, lcp);
//find the lcp of s[0..] & s1[i..]

void exKMP(char* a, char *b, int M, int N, int *Next, int
*ret) {
  int i,j,k;
  for(j=0;1+j<M&&a[j]==a[1+j];j++);
  Next[1]=j;
  k=1;
  for(i=2;i<M;i++) {
    int Len=k+Next[k], L=Next[i-k];
    if(L<Len-i) Next[i]=L;
    else{
      for(j=max(0,Len-i);i+j<M&&a[j]==a[i+j];j++);
      Next[i]=j;
      k=i;
    }
  }
  for(j=0;j<N&&j<M&&a[j]==b[j];j++);
  ret[0]=j;
  k=0;
  for(int i=1;i<N;i++){
    int Len=k+ret[k],L=Next[i-k];
    if(L<Len-i){
      ret[i]=L;
    }else{
      for(j=max(0,Len-i);j<M&&i+j<N&&a[j]==b[i+j];j++);
      ret[i]=j;
      k=i;
    }
  }
}
```

6.6 AC Automaton

```
int Next[MAXN][26];
int fail[MAXN];
int val[MAXN];
int n_node;
int root;

int newnode() {
  val[n_node] = 0;
  REP(i,26) Next[n_node][i] = -1;
  return n_node++;
}
void init() {
  n_node = 0;
  int p = newnode();
  REP(i,26) Next[p][i] = 1;
  root = newnode();
}
void insert(char s[]) {
  int now = root;
  for(int i = 0; s[i]; i++) {
    if(Next[now][s[i]-'a']==-1)
      Next[now][s[i]-'a'] = newnode();
    now = Next[now][s[i]-'a'];
  }
  val[now]++;
}
void build() {
  queue<int> q;
  fail[root] = 0;
  q.push(root);
  while(!q.empty()) {
    int now = q.front();  q.pop();
    REP(i,26) {
```

```
      if(Next[now][i] == -1) {
        Next[now][i] = Next[fail[now]][i];
      }else{
        fail[Next[now][i]] = Next[fail[now]][i];
        q.push(Next[now][i]);
      }
    }
  }
}
int query(char s[]) {
  int res = 0;
  int now = root;
  for(int i = 0; s[i]; i++) {
    now = Next[now][s[i]-'a'];
    int p = now;
    while(p != root) {
      res += val[p];
      val[p] = 0;
      p = fail[p];
    }
  }
  return res;
}
```

7 Math

7.1 Rational Number

```
class Rational implements Comparable<Rational> {
  static final Rational R0 = new Rational(Big.ZERO, Big.
  ONE);

  final Big num;
  final Big den;

  Rational(long num, long den) {
    this(Big.valueOf(num), Big.valueOf(den));
  }
  Rational(Big num, Big den) {
    Big gcd = num.gcd(den);
    if (gcd.signum() != 0) {
      num = num.div(gcd);
      den = den.div(gcd);
    }
    if (den.signum() < 0) {
      num = num.negate();
      den = den.negate();
    }
    this.num = num;
    this.den = den;
  }
  Rational add(Rational r) {
    return new Rational(
      num.mul(r.den).add(r.num.mul(den)),
      den.mul(r.den)
    );
  }
  Rational sub(Rational r) {
    return new Rational(
      num.mul(r.den).sub(r.num.mul(den)),
      den.mul(r.den)
    );
  }
  Rational mul(Rational r) {
    return new Rational(
      num.mul(r.num),
      den.mul(r.den)
    );
  }
  Rational div(Rational r) {
    return new Rational(
      num.mul(r.den),
      den.mul(r.num)
    );
  }
  int signum() {
    return num.signum();
  }
  Rational pow(int b) {
    Big n = Big.ONE, d = Big.ONE, an = num, ad = den;
    while (b > 0) {
      if ((b & 1) == 1) {
        n = n.mul(an);
        d = d.mul(ad);
      }
      an = an.mul(an);
      ad = ad.mul(ad);
      b >>>= 1;
    }
    return new Rational(n, d);
  }
  public int compareTo(Rational o) {
    return num.mul(o.den)
        .compareTo(o.num.mul(den));
  }
}
```

7.2 Mod Comb

```
long combination(int n, int m, long mod) {
  if (m < 0 || m > n) return 0;
  if (2 * m > n) m = n - m;
  long res = 1;
  for (int i = n - m + 1; i <= n; i++)
    res = res * i % mod;
  return res * Big.vOf(factorial(m, mod))
      .modInv(Big.vOf(mod)).long() % mod;
}
long[] combinationRowTable(int n, long mod) {
  long[] res = invTable(n, mod);
  res[0] = 1;
  for (int i = 1; i <= n; i++) {
    res[i] = res[i - 1] * (n - i + 1) % mod * res[i] % mod
    ;
  }
  return res;
}
long[][] combinationTable(int n, long mod) {
  long[][] res = new long[n + 1][n + 1];
  for (int i = 0; i <= n; i++) {
    res[i][0] = 1;
    for (int j = 1; j <= i; j++) {
      res[i][j] = res[i - 1][j - 1] + res[i - 1][j];
      if (res[i][j] >= mod)
        res[i][j] -= mod;
    }
  }
  return res;
}
// C(n, k) % p, p should be small
int modComb(int n, int k, int p) {
  if (n < 0 || k < 0 || n < k) return 0;
  int[] a1 = modFact(n, p), a2 = modFact(k, p), a3 =
  modFact(n - k, p);
  if (a1[1] > a2[1] + a3[1]) return 0;
  return a1[0] * (int) inv(a2[0] * a3[0] % p, p) % p;
}
```

7.3 Mod Facterial

$n = r[0]\ p^{r[1]}$

```
Map<Integer, int[]> fact;
int e;
int[] modFact(int n, int p) {
  return new int[] { modFactRec(n, p), e };
}
int modFactRec(int n, int p) {
  e = 0;
  if (n == 0) return 1;
  int res = modFactRec(n / p, p);
  e += n / p;
  if (n / p % 2 != 0)
    return res * (p - fact(n % p, p)) % p;
  return res * fact(n % p, p) % p;
}
int fact(int n, int p) {
  if (fact == null) fact = new HashMap<>();
  if (!fact.containsKey(p)) {
    int[] f = new int[p];
    f[0] = 1;
    for (int i = 1; i < p; i++)
      f[i] = (int) ((long) f[i - 1] * i % p);
    fact.put(p, f);
  }
```

```
  return fact.get(p)[n];
}
```

7.4 Mod Inv

```
long invS(long a, long mod) {
  if (a == 1) return 1;
  return invS(mod % a, mod) * (mod − mod / a) % mod;
}
long inv(long a, long mod) {
  return Big.vOf(a).modInv(Big.vOf(mod)).long();
}
long[] invTable(int n, long mod) {
  long[] res = new long[n + 1];
  if (n >= 1) res[1] = 1;
  for (int i = 2; i <= n; i++)
    res[i] = (mod − mod / i * res[(int) (mod % i)] % mod)
    % mod;
  return res;
}
```

7.5 Mod Log

$a^x = b(mod\ m)$ return $min(x)$; return -1 if no solution

```
long modLog(long a, long b, long m) {
  if (b % gcd(a, m) != 0) return −1;
  if (m == 0) return 0;
  long n = (long) Math.sqrt(m) + 1;
  Map<Long, Long> map = new HashMap<>();
  long an = 1;
  for (long j = 0; j < n; j++) {
    if (!map.containsKey(an)) map.put(an, j);
    an = an * a % m;
  }
  long ain = 1, res = Long.MAX_VALUE;
  for (long i = 0; i < n; i++) {
    long[] is = congruence(ain, b, m);
    for (long aj = is[0]; aj < m; aj += is[1]) {
      if (map.containsKey(aj)) {
        long j = map.get(aj);
        res = Math.min(res, i * n + j);
      }
    }
    if (res < Long.MAX_VALUE) return res;
    ain = ain * an % m;
  }
  return −1;
}
```

7.6 Factorize

```
void factorize(Big n, Map<Big, Int> factors) {
  if (isPrime(n)) {
    Num.inc(factors, n);
  } else {
    for (Int prime : primes) {
      Big p = Big.vOf(prime);
      while (n.mod(p).equals(Big.ZERO)) {
        Num.inc(factors, p);
        n = n.divide(p);
      }
    }
    if (!n.equals(Big.ONE)) {
      if (isPrime(n)) {
        Num.inc(factors, n);
      } else {
        Big d = pollardRho(n, Big.ONE);
        factorize(d, factors);
        factorize(n.divide(d), factors);
      }
    }
  }
}
Big pollardRho(Big n, Big c) {
  Big x = Big.vOf(2);
  Big y = Big.vOf(2);
  Big d = Big.ONE;
  while (d.equals(Big.ONE)) {
    x = x.mul(x).mod(n).add(c);
    y = y.mul(y).mod(n).add(c);
    y = y.mul(y).mod(n).add(c);
    d = x.subtract(y).abs().gcd(n);
  }
  if (d.equals(n)) return pollardRho(n, c.add(Big.ONE));
  return d;
}
```

7.7 Prime Table

```
boolean[] primeTable(int n, List<Integer> primes) {
  Num.primes = primes;
  isPrime = new boolean[n + 1];
  Arrays.fill(isPrime, true);
  isPrime[0] = isPrime[1] = false;
  /* for (int i = 2; i <= n; i++) {
    if (isPrime[i]) primes.add(i);
    for (int p : primes) {
      if (i > n / p) break;
      isPrime[i * p] = false;
      if (i % p == 0) break;
    }
  } */
  for (int i = 2; i <= n; i++) {
    if (isPrime[i]) {
      primes.add(i);
      for (int j = i + i; j <= n; j += i) {
        isPrime[j] = false;
      }
    }
  }
  return isPrime;
}
```

7.8 Eular Phi

```
long phi(long n) {
  long ans = n;
  for (long i : primes) {
    if (i * i > n) break;
    if (n % i == 0) {
      ans = ans / i * (i − 1);
      while (n % i == 0) n /= i;
    }
  }
  if (n > 1) ans = ans / n * (n − 1);
  return ans;
}
int[] phiTable(int n) {
  int[] phi = new int[n + 1];
  phi[1] = 1;
  for (int i = 2; i <= n; i++)
    if (phi[i] == 0) {
      for (int j = i; j <= n; j += i) {
        if (phi[j] == 0) phi[j] = j;
        phi[j] = phi[j] / i * (i − 1);
      }
    }
  return phi;
}
```

7.9 Extend GCD

求解 $a, b, c = gcd(x, y)$
其中，$ax + by = gcd(x, y)$，(a, b) 一般解 $(a + t\ y/c, b−t\ x/c)$

```
int[] exGcd(int x, int y) {
  int a0 = 1, a1 = 0, b0 = 0, b1 = 1, t;
  while (y != 0) {
    t = a0 − x / y * a1; a0 = a1; a1 = t;
    t = b0 − x / y * b1; b0 = b1; b1 = t;
    t = x % y; x = y; y = t;
  }
  if (x < 0) { a0 = −a0; b0 = −b0; x = −x; }
  return new int[]{a0, b0, x};
}
```

7.10 Congruence

$Ax = B(mod\ M)$

```
long[] congruence(long[] A, long[] B, long[] M) {
  long x = 0, m = 1;
  for (int i = 0; i < A.length; i++) {
    long a = A[i] * m, b = B[i] − A[i] * x, d = gcd(a, M[i
    ]);
```

```
    if (b % d != 0) return null;
    x += m * (b / d * inv(a / d, M[i] / d) % (M[i] / d));
    m *= M[i] / d;
  }
  return new long[] { (x % m + m) % m, m };
}
```

7.11 Mobius

```
把 n 的约数的莫比乌斯值用 map 形式的返回。O(\sqrt n)
Map<Long, Integer> moebius(long n) {
  Map<Long, Integer> res = new TreeMap<>();
  List<Long> primes = primeFactors(n);
  int m = primes.size();
  for (int i = 0; i < (1 << m); i++) {
    int mu = 1;
    long d = 1;
    for (int j = 0; j < m; j++) {
      if ((i & (1 << j)) != 0) {
        mu *= -1;
        d *= primes.get(j);
      }
    }
    res.put(d, mu);
  }
  return res;
}
int[] moebiusTable(int n) {
  boolean[] check = new boolean[n + 1];
  List<Integer> primes = new ArrayList<>(n / 10);
  int[] mu = new int[n + 1];
  mu[1] = 1;
  for (int i = 2; i <= n; i++) {
    if (!check[i]) {
      primes.add(i);
      mu[i] = -1;
    }
    for (int p : primes) {
      if (i * p > n) break;
      check[i * p] = true;
      if (i % p == 0) {
        mu[i * p] = 0;
        break;
      } else {
        mu[i * p] = -mu[i];
      }
    }
  }
  return mu;
}
```

7.12 Sqrt Int

```
Big sqrt(String theNumber) {
  int length = theNumber.length(), i;
  Big res = Big.ZERO;
  Big B20 = Big.vOf(20);
  Big t, x = Big.ZERO, v, few = Big.ZERO;
  Big hg = Big.vOf(100);
  int pos = 2 - length % 2;
  String tmp = theNumber.substring(0, pos);
  while (true) {
    v = few.mul(hg).add(Big.vOf(Int.parse(tmp)));
    if (res.compareTo(Big.ZERO) == 0) i = 9;
    else i = v.divide(res.mul(B20)).intValue();
    for (; i >= 0; i--) {
      t = res.mul(B20).add(Big.vOf(i)).mul(Big.vOf(i));
      if (t.compareTo(v) <= 0) {
        x = t;
        break;
      }
    }
    res = res.mul(Big.TEN).add(Big.vOf(i));
    few = v.subtract(x);
    pos++;
    if (pos > length) break;
    tmp = theNumber.substring(pos - 1, ++pos);
  }
  return res;
}
```

7.13 Int Partition

整数拆分方案数 $O(n^{1.5})$

```
int partition(int n) {
  int[] dp = new int[n + 1];
  dp[0] = 1;
  for (int i = 1; i <= n; i++) {
    for (int j = 1, r = 1;
        i - (3 * j * j - j) / 2 >= 0;
        j++, r *= -1) {
      dp[i] += dp[i - (3 * j * j - j) / 2] * r;
      if (i - (3 * j * j + j) / 2 >= 0) {
        dp[i] += dp[i - (3 * j * j + j) / 2] * r;
      }
    }
  }
  return dp[n];
}
```

8 Math G

8.1 Eular

```
#define MAXN 1000000
int vis[MAXN]={1,1}, phi[MAXN], mu[MAXN];
int prime[MAXN], n_prime;
void Eular() {
  n_prime=0;
  phi[1]=1;
  mu[1]=1;
  for(int i = 2; i < MAXN; i++) {
    if(!vis[i]) {
      prime[n_prime++] = i;
      phi[i] = i-1;
      mu[i] = -1;
    }
    for(int j = 0; j < n_prime && i * prime[j] < MAXN; j
    ++) {
      vis[i * prime[j]] = 1;
      if(i % prime[j] == 0) {
        phi[i * prime[j]] = phi[i] * prime[j];
        mu[i * prime[j]] = 0;
        break;
      }else{
        phi[i * prime[j]] = phi[i] * phi[prime[j]];
        mu[i * prime[j]] = -mu[i];
      }
    }
  }
}
```

8.2 Mod Inverse n CRT

```
ll ex_gcd(ll a,ll b,ll &x,ll &y){
  if (a==0&&b==0) return -1;
  if (b==0){x=1;y=0;return a;}
  ll d=ex_gcd(b,a%b,y,x);
  y-=a/b*x;
  return d;
}
ll inv(ll a,ll n){ // a != 0 && n is prime
  ll x,y;
  ll d = ex_gcd(a,n,x,y);
  return (x%n+n)%n;
}
ll CRT(ll a[], ll m[], int n) {
  ll M=1;
  REP(i,n) M*=m[i];
  ll ret = 0;
  REP(i,n) {
    ll x,y;
    ll tm=M/m[i];
    exgcd(tm,m[i],x,y);
    ret=(ret+tm*x*a[i])%M;
  }
  return (ret+M)%M;
}
//对2^64的逆元
ull inv64(ull a) {
  ull b = a;
  b *= (2 - a*b);
```

```
  b *= (2 - a*b);
  b *= (2 - a*b);
  b *= (2 - a*b);
  b *= (2 - a*b);
  return b;
}
//O(n)预处理逆元
inv[1] = ..;
for(i = 2; i < LIM; i++) inv[i] = MOD - MOD / i * inv[MOD % i] % MOD;
```

8.3 FXT

```
---------- F F T ----------
const double PI=acos(-1.0);
//typedef complex<double> Complex;
struct Complex {
  double r,i;
  Complex(double _r = 0.0,double _i = 0.0) {
    r = _r; i = _i;
  }
  Complex operator +(const Complex &b) {
    return Complex(r+b.r,i+b.i);
  }
  Complex operator -(const Complex &b) {
    return Complex(r-b.r,i-b.i);
  }
  Complex operator *(const Complex &b) {
    return Complex(r*b.r-i*b.i,r*b.i+i*b.r);
  }
};
void change(Complex y[],int len) {
  int i,j,k;
  for(i = 1, j = len/2;i < len-1; i++) {
    if(i < j)swap(y[i],y[j]);
    k = len/2;
    while( j >= k) {
      j -= k;
      k /= 2;
    }
    if(j < k) j += k;
  }
}
void FFT(Complex y[],int len,int on) {
  change(y,len);
  for(int h = 2; h <= len; h <<= 1) {
    Complex wn(cos(-on*2*PI/h),sin(-on*2*PI/h));
    for(int j = 0;j < len;j+=h) {
      Complex w(1,0);
      for(int k = j;k < j+h/2;k++) {
        Complex u = y[k];
        Complex t = w*y[k+h/2];
        y[k] = u+t;
        y[k+h/2] = u-t;
        w = w*wn;
      }
    }
  }
  if(on == -1)
    for(int i = 0;i < len;i++)
      y[i].r /= len;
}
---------- N T T ----------
/*
P      G
998244353  3
1004535809 3
786433   10
880803841  26
*/
const int N = 1 << 18;
const int P = (479 << 21) + 1;
const int G = 3;
ll wn[31];

ll pow_mod(ll a, ll b, ll m) {
  ll res = 1, t = a%m;
  while(b) {
    if(b&1) res = res*t%m;
    b>>=1;
    t = t*t%m;
  }
  return res;
}
void GetWn() {
  REP(i, 31) {
    int t = 1 << i;
    wn[i] = pow_mod(G, (P - 1) / t, P);
  }
}
void Rader(ll a[], int len) {
  int j = len >> 1;
  for(int i=1; i<len-1; i++) {
    if(i < j) swap(a[i], a[j]);
    int k = len >> 1;
    while(j >= k) {
      j -= k;
      k >>= 1;
    }
    if(j < k) j += k;
  }
}
void NTT(ll a[], int len, int on) {
  Rader(a, len);
  for(int h = 2, id = 0; h <= len; h <<= 1) {
    id++;
    for(int j = 0; j < len; j += h) {
      ll w = 1;
      for(int k = j; k < j + h / 2; k++) {
        ll u = a[k] % P;
        ll t = w * (a[k + h / 2] % P) % P;
        a[k] = (u + t) % P;
        a[k + h / 2] = ((u - t) % P + P) % P;
        w = w * wn[id] % P;
      }
    }
  }
  if(on == -1) {
    for(int i = 1; i < len / 2; i++)
      swap(a[i], a[len - i]);
    ll Inv = pow_mod(len, P - 2, P);
    REP(i,len)
      a[i] = a[i] % P * Inv % P;
  }
}
---------- F W T ----------
/*
P      G
998244353  3
1004535809 3
786433   10
880803841  26
*/
const int N = 1 << 18;
const int P = (479 << 21) + 1;
const int G = 3;
ll wn[31];

ll pow_mod(ll a, ll b, ll m) {
  ll res = 1, t = a%m;
  while(b) {
    if(b&1) res = res*t%m;
    b>>=1;
    t = t*t%m;
  }
  return res;
}
void GetWn() {
  REP(i, 31) {
    int t = 1 << i;
    wn[i] = pow_mod(G, (P - 1) / t, P);
  }
}
void Rader(ll a[], int len) {
  int j = len >> 1;
  for(int i=1; i<len-1; i++) {
    if(i < j) swap(a[i], a[j]);
    int k = len >> 1;
    while(j >= k) {
      j -= k;
      k >>= 1;
    }
    if(j < k) j += k;
  }
}
void NTT(ll a[], int len, int on) {
```

```
  Rader(a, len);
  for(int h = 2, id = 0; h <= len; h <<= 1) {
    id++;
    for(int j = 0; j < len; j += h) {
      ll w = 1;
      for(int k = j; k < j + h / 2; k++) {
        ll u = a[k] % P;
        ll t = w * (a[k + h / 2] % P) % P;
        a[k] = (u + t) % P;
        a[k + h / 2] = ((u - t) % P + P) % P;
        w = w * wn[id] % P;
      }
    }
  }
  if(on == -1) {
    for(int i = 1; i < len / 2; i++)
      swap(a[i], a[len - i]);
    ll Inv = pow_mod(len, P - 2, P);
    REP(i,len)
      a[i] = a[i] % P * Inv % P;
  }
}
---------- F M T ----------
#define MOD 1000000007
void Mod(int& a) {
  if(a >= MOD) a -= MOD;
  if(a < 0) a += MOD;
}
void fmt(int* a, int n) {
  int m = __builtin_ctz(n)-1;
  REP(i,m) {
    REP(j,n) if(!(j>>i&1)) {
      Mod(a[j|(1<<i)] += a[j]);
    }
  }
}
void ifmt(int* a, int n) {
  int m = __builtin_ctz(n)-1;
  REP(i,m) {
    REP(j,n) if(!(j>>i&1)) {
      Mod(a[j|(1<<i)] -= a[j]);
    }
  }
}
```

8.4 Matrix Tree

基尔霍夫矩阵 = 度数矩阵 - 邻接矩阵

```
ll M[MAXN][MAXN];
ll det(ll a[][MAXN], int n, ll mod) {
  REP(i,n)
  REP(j,n)
    a[i][j]%=mod;
  ll ret=1;
  REP(i,n) if(i){
    REP2(j,i+1,n-1) {
      while(a[j][i]) {
        int t=a[i][i]/a[j][i];
        REP2(k,i,n-1)
           a[i][k] = (a[i][k]-a[j][k]*t)%mod;
        REP2(k,i,n-1)
          swap(a[i][k],a[j][k]);
        ret = -ret;
      }
    }
    if(a[i][i] == 0) return 0;
    ret = ret*a[i][i]%mod;
  }
  return (ret+mod)%mod;
}
```

8.5 Primitive Root

P is prime, if not, replace P-1 with phi(P)

```
int solve(int P) {
  if(P == 2) {
    return 1;
  }
  vecotr<int> v = getFactors(P-1);
  for(int g = 2; g < P; g++) {
    bool flag = true;
    for(int i = 0; i < v.size(); i++) {
      int t = (P-1)/v[i];
      if(pow_mod(g,t,P) == 1) {
        flag = false;
        break;
      }
    }
    if(flag) {
      return g;
    }
  }
  return -1;
}
```

8.6 Pollard's Rho

```
ll mul(ll a, ll b, ll m) {
  ll ret = a * (b & 0x1fff) % m;
  (ret += ((a <<= 13) %= m) * ((b >>= 13) & 0x1fff)) %= m;
  (ret += ((a <<= 13) %= m) * ((b >>= 13) & 0x1fff)) %= m;
  (ret += ((a <<= 13) %= m) * ((b >>= 13) & 0x1fff)) %= m;
  return ret;
}

ll pollard(ll n) {
  if(n % 2 == 0) return 2;
  ll x, y, d, c;
  for(c = 1; ;++c) {
    for(x = y = 2, d = 1; d == 1; d = __gcd(abs(x - y), n)
    ) {
      x = mul(x, x, n) + c;
      y = mul(y, y, n) + c;
      y = mul(y, y, n) + c;
    }
    if(d < n) return d;
  }
}
```

8.7 Miller-Rabin

```
ll mul(ll a, ll b, ll m) {
  ll ret = a * (b & 0x1fff) % m;
  (ret += ((a <<= 13) %= m) * ((b >>= 13) & 0x1fff)) %= m;
  (ret += ((a <<= 13) %= m) * ((b >>= 13) & 0x1fff)) %= m;
  (ret += ((a <<= 13) %= m) * ((b >>= 13) & 0x1fff)) %= m;
  return ret;
}

ll power(ll a, ll e, ll m) {
  ll y = 1;
  for(; e; e >>= 1, a = mul(a, a, m)) if(e & 1) y = mul(y,
   a, m);
  return y;
}

// n < 341550071728321 < 2^49
ll MillerRabin[] = {2, 3, 5, 7, 11, 13, 17, -1};
bool isPrime(ll n) {
  if (n <= 1 || n % 2 == 0) return n == 2;
  int r, s = 0;
  ll d, *a, ad;
  for(d = n - 1; d % 2 == 0; d >>= 1, ++s);
  for(a = MillerRabin; ~*a && *a < n; ++a) {
    ad = power(*a, d, n);
    if(ad == 1) continue;
    REP(r, s) {
      if(ad == n-1) break;
      ad = mul(ad, ad, n);
    }
    if(r == s) return 0;
  }
  return 1;
}
```

9 Matrix

9.1 Matrix Mul n Pow

```
long[][] mul(long[][] a, long[][] b) {
  int n = a.length;
  long[][] c = new long[n][n];
  for (int i = 0; i < n; i++) {
```

```
    for (int k = 0; k < n; k++) if (a[i][k] != 0) {
      for (int j = 0; j < n; j++) {
        c[i][j] = c[i][j] + a[i][k] * b[k][j];
      }
    }
  }
  return c;
}
long[][] pow(long[][] a, long b) {
  int n = a.length;
  long[][] c = new long[n][n];
  for (int i = 0; i < n; i++)
    c[i][i] = 1;
  while (b > 0) {
    if ((b & 1) != 0) c = mul(c, a);
    a = mul(a, a);
    b >>>= 1;
  }
  return c;
}
```

9.2 Solution Space

```
long[][] solutionSpace(long[][] A, long[] b, long mod) {
  int n = A.length, m = A[0].length;
  Big MOD = Big.vOf(mod);
  long[][] a = new long[n][m + 1];
  for (int i = 0; i < n; i++) {
    System.arraycopy(A[i], 0, a[i], 0, m);
    a[i][m] = b[i];
  }
  int[] id = new int[n + 1];
  第 i 行的第一个非零元 1 所在的位置是 id[i]
  Arrays.fill(id, -1);
  int pi = 0; 矩阵 A 的秩
  for (int pj = 0; pi < n && pj < m; pj++) {
    for (int i = pi + 1; i < n; i++) {
      if (Math.abs(a[i][pj]) > Math.abs(a[pi][pj])) {
        long[] t = a[i];
        a[i] = a[pi];
        a[pi] = t;
      }
    }
    if (Math.abs(a[pi][pj]) < EPS) 当前列已经全零
      continue;
    //double inv = 1 / a[pi][pj];
    long inv = Big.vOf(a[pi][pj]).modInv(MOD).long();
    for (int j = 0; j <= m; j++)
      a[pi][j] = (a[pi][j] * inv) % mod;
    for (int i = 0; i < n; i++)
      if (i != pi) {
        long d = a[i][pj];
        for (int j = 0; j <= m; j++)
          a[i][j] = (a[i][j] - d * a[pi][j] % mod) % mod;
      }
    id[pi++] = pj;
  }
  for (int i = pi; i < n; i++)
    if (Math.abs(a[i][m]) > EPS)
      增广矩阵的秩更大，无解
      return null;
  long[][] X = new long[1 + m - pi][m];
  for (int j = 0, k = 0; j < m; j++) {
    if (id[k] == j)
      X[0][j] = a[k++][m];
    else {
      for (int i = 0; i < k; i++)
        X[1 + j - k][id[i]] = -a[i][j];
      X[1 + j - k][j] = 1;
    }
  }
  return X;
}
```

9.3 Int Solve

大约 $O(n^5)$

```
Big[] intSolve(Big[][] A, Big[] b) {
  int n = A.length, m = A[0].length;
  Big[][] a = new Big[n][m + 1];
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < m; j++) a[i][j] = A[i][j];
    a[i][m] = b[i];
  }
  Stack<Trans> trans = new Stack<Trans>();
  for (int p = 0; p < m && p < n; p++) {
    for (;;) {
      int pi = p, pj = p;
      for (int i = p; i < n; i++) {
        for (int j = p; j < m; j++) {
          if (a[i][j].signum() != 0 && (a[pi][pj].signum()
            == 0
              || a[pi][pj].abs().compareTo(a[i][j].abs())
              > 0)) {
            pi = i;
            pj = j;
          }
        }
      }
      swap(a, pi, p);
      for (int i = p; i < n; i++) swap(a[i], pj, p);
      if (pj != p) trans.push(new Trans(0, pj, p, null));
      if (a[p][p].signum() == 0) break;
      boolean end = true;
      for (int i = p + 1; i < n; i++) {
        Big d = a[i][p].div(a[p][p]);
        end = end && a[i][p].signum() == 0;
        for (int j = p; j <= m; j++) {
          a[i][j] = a[i][j].sub(a[p][j].mul(d));
        }
      }
      for (int j = p + 1; j < m; j++) {
        Big d = a[p][j].div(a[p][p]);
        end = end && a[p][j].signum() == 0;
        trans.push(new Trans(1, j, p, d));
        for (int i = p; i < n; i++) {
          a[i][j] = a[i][j].sub(a[i][p].mul(d));
        }
      }
      if (end) break;
    }
  }
  Big[] res = new Big[m];
  fill(res, ZERO);
  for (int i = 0; i < m && i < n; i++) {
    if (a[i][i].signum() < 0) {
      a[i][i] = a[i][i].negate();
      a[i][m] = a[i][m].negate();
    }
    if (a[i][i].signum() == 0) {
      if (a[i][m].signum() != 0) return null;
    } else if (a[i][m].mod(a[i][i]).signum() == 0) {
      res[i] = a[i][m].div(a[i][i]);
    } else return null;
  }
  for (int i = min(m, n); i < n; i++) if (a[i][m].signum()
    != 0) return null;
  while (!trans.isEmpty()) {
    Trans t = trans.pop();
    if (t.type == 0) swap(res, t.a, t.b);
    else res[t.b] = res[t.b].sub(res[t.a].mul(t.c));
  }
  return res;
}
class Trans {
  int type, a, b; Big c;
}
```

9.4 Simplx

求解：$max\{cx|Ax \le b, x \ge 0\}$

```
double[] simplex(double[][] A, double[] b, double[] c) {
  int n = A.length, m = A[0].length + 1, r = n, s = m - 1;
  double[][] D = new double[n + 2][m + 1];
  int[] ix = new int[n + m];
  for (int i = 0; i < n + m; i++) ix[i] = i;
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < m - 1; j++)
      D[i][j] = -A[i][j];
    D[i][m - 1] = 1;
    D[i][m] = b[i];
    if (D[r][m] > D[i][m]) r = i;
  }
  for (int j = 0; j < m - 1; j++) D[n][j] = c[j];
  D[n + 1][m - 1] = -1;
```

```
  for (double d;;) {
    if (r < n) {
      int t = ix[s]; ix[s] = ix[r + m]; ix[r + m] = t;
      D[r][s] = 1.0 / D[r][s];
      for (int j = 0; j <= m; j++)
        if (j != s) D[r][j] *= -D[r][s];
      for (int i = 0; i <= n + 1; i++) if (i != r) {
          for (int j = 0; j <= m; j++) if (j != s)
              D[i][j] += D[r][j] * D[i][s];
          D[i][s] *= D[r][s];
        }
    }
    r = -1; s = -1;
    for (int j = 0; j < m; j++)
      if (s < 0 || ix[s] > ix[j]) {
        if (D[n + 1][j] > EPS ||
            D[n + 1][j] > -EPS && D[n][j] > EPS)
          s = j;
      }
    if (s < 0) break;
    for (int i = 0; i < n; i++) if (D[i][s] < -EPS) {
        if (r < 0 ||
          (d = D[r][m] / D[r][s] - D[i][m] / D[i][s]) < -
          EPS
          || d < EPS && ix[r + m] > ix[i + m]) r = i;
      }
    if (r < 0) return null;
  }
  if (D[n + 1][m] < -EPS) return null;
  double[] x = new double[m - 1];
  for (int i = m; i < n + m; i++)
    if (ix[i] < m - 1) x[ix[i]] = D[i - m][m];
  return x;
}
```

10 Geo

10.1 P

```
class P implements Comparable<P> {
  考虑 double 的精度是 15 位来设置
  static final double EPS = 1e-9;
  为了减少误差
  static double add(double a, double b) {
    if (Math.abs(a + b) < EPS * (Math.abs(a) + Math.abs(b)
    ))
      return 0;
    return a + b;
  }
  final double x, y;

  double det(P p) {
    return add(x * p.y, -y * p.x);
  }
  double dot(P p) {
    return add(x * p.x, y * p.y);
  }
  double abs() {
    return Math.sqrt(abs2());
  }
  double abs2() {
    return dot(this);
  }
  饶原点旋转角度B（弧度值）产生的新点
  P rot(double rad) {
    return new P(
      add(x * Math.cos(rad), -y * Math.sin(rad)),
      add(x * Math.sin(rad), y * Math.cos(rad))
    );
  }
  P rot90() {
    return new P(-y, x);
  }
  int compareTo(P p) {
    int b = sig(x - p.x);
    if (b != 0) return b;
    return sig(y - p.y);
  }
  static int sig(double x) {
    if (Math.abs(x) < EPS) return 0;
    return x < 0 ? -1 : 1;
  }
  返回两点和原点形成的夹角
  注意这两点都不能为原点
  static double rad(P p1, P p2) {
    return Math.acos(p1.dot(p2) / p1.abs() / p2.abs());
  }
  返回 0~2PI 的角
  static double rad2(P p1, P p2) {
    double r = (p1.det(p2) < -EPS ? -1 : 1) * rad(p1, p2);
    if (r < 0) r += 2 * Math.PI;
    return r;
  }
}
```

10.2 Line

```
线段相交判定
boolean crsSS(P p1, P p2, P q1, P q2) {
  if (Math.max(p1.x, p2.x) + EPS < Math.min(q1.x, q2.x))
    return false;
  if (Math.max(q1.x, q2.x) + EPS < Math.min(p1.x, p2.x))
    return false;
  if (Math.max(p1.y, p2.y) + EPS < Math.min(q1.y, q2.y))
    return false;
  if (Math.max(q1.y, q2.y) + EPS < Math.min(p1.y, p2.y))
    return false;
  return p2.sub(p1).det(q1.sub(p1)) * p2.sub(p1).det(q2.
  sub(p1)) <= 0
      && q2.sub(q1).det(p1.sub(q1)) * q2.sub(q1).det(p2.
      sub(q1)) <= 0;
}
直线和线段的相交判定
boolean crsLS(P l1, P l2, P s1, P s2) {
  return s1.sub(l2).det(l1.sub(l2)) * s2.sub(l2).det(l1.
  sub(l2)) <= 0;
}
直线相交判定
返回-1表示重合，为0表示平行，为1表示相交
int crsLL(P p1, P p2, P q1, P q2) {
  if (sig(p1.sub(p2).det(q1.sub(q2))) != 0) return 1;
  if (sig(p1.sub(q2).det(q1.sub(p2))) != 0) return 0;
  return -1;
}
直线和直线的交点
P isLL(P p1, P p2, P q1, P q2) {
  double d = q2.sub(q1).det(p2.sub(p1));
  if (sig(d) == 0) return null;
  return p1.add(p2.sub(p1).mul(q2.sub(q1).det(q1.sub(p1))
  / d));
}
点到直线的垂足
P proj(P p1, P p2, P q) {
  return p1.add(p2.sub(p1).mul(p2.sub(p1).dot(q.sub(p1)) /
   p2.sub(p1).abs2()));
}
计算多边形的*有向*面积
点不需要有顺序
double directedArea(P... ps) {
  double res = 0;
  for (int i = 0; i < ps.length; i++) {
    res += ps[i].det(ps[(i + 1) % ps.length]);
  }
  return res / 2;
}
线段到点的距离
double disSP(P p1, P p2, P q) {
  if (p2.sub(p1).dot(q.sub(p1)) <= 0)
    return q.sub(p1).abs();
  if (p1.sub(p2).dot(q.sub(p2)) <= 0)
    return q.sub(p2).abs();
  return disLP(p1, p2, q);
}
直线到点的距离
double disLP(P p1, P p2, P q) {
  return Math.abs(p2.sub(p1).det(q.sub(p1))) / p2.sub(p1).
  abs();
}
```

10.3 Circle

```
圆和线段的相交判定
boolean crsCS(P c, double r, P p1, P p2) {
  return disSP(p1, p2, c) < r + EPS &&
```

```
      (r < c.sub(p1).abs() + EPS || r < c.sub(p2).abs() +
      EPS);
}
圆和圆的相交判定
boolean crsCC(P c1, double r1, P c2, double r2) {
  double dis = c1.sub(c2).abs();
  return dis < r1 + r2 + EPS && Math.abs(r1 - r2) < dis +
  EPS;
}
四点共圆判定
boolean onC(P p1, P p2, P p3, P p4) {
  P c = CCenter(p1, p2, p3);
  if (c == null) return false; 有三点共线，返回false
  return add(c.sub(p1).abs2(), -c.sub(p4).abs2()) == 0;
}
三点共圆的圆心
P CCenter(P p1, P p2, P p3) {
  if (disLP(p1, p2, p3) < EPS) return null; 三点共线
  P q1 = p1.add(p2).mul(0.5);
  P q2 = q1.add(p1.sub(p2).rot90());
  P s1 = p3.add(p2).mul(0.5);
  P s2 = s1.add(p3.sub(p2).rot90());
  return isLL(q1, q2, s1, s2);
}
直线和圆的交点
P[] isCL(P c, double r, P p1, P p2) {
  double x = p1.sub(c).dot(p2.sub(p1));
  double y = p2.sub(p1).abs2();
  double d = add(x * x, -y * (add(p1.sub(c).abs2(), -r * r
  )));
  if (d < -EPS) return new P[0];
  if (d < 0) d = 0;
  P q1 = p1.sub(p2.sub(p1).mul(x / y));
  P q2 = p2.sub(p1).mul(Math.sqrt(d) / y);
  return new P[]{q1.sub(q2), q1.add(q2)};
}
两圆的交点
P[] isCC(P c1, double r1, P c2, double r2) {
  double x = c1.sub(c2).abs2();
  double y = (add(r1 * r1, -r2 * r2) / x + 1) / 2;
  double d = add(r1 * r1 / x, -y * y);
  if (d < -EPS) return new P[0];
  if (d < 0) d = 0;
  P q1 = c1.add(c2.sub(c1).mul(y));
  P q2 = c2.sub(c1).mul(Math.sqrt(d)).rot90();
  return new P[]{q1.sub(q2), q1.add(q2)};
}
点和圆的两个切点
P[] tanCP(P c, double r, P p) {
  double x = p.sub(c).abs2();
  double d = add(x, -r * r);
  if (d < -EPS) return new P[0];
  if (d < 0) d = 0;
  P q1 = p.sub(c).mul(r * r / x);
  P q2 = p.sub(c).mul(-r * Math.sqrt(d) / x).rot90();
  return new P[]{c.add(q1.sub(q2)), c.add(q1.add(q2))};
}
两圆的公切线
返回的是切点对
P[][] tanCC(P c1, double r1, P c2, double r2) {
  List<P[]> list = new ArrayList<P[]>();
  if (Math.abs(r1 - r2) < EPS) {
    P dir = c2.sub(c1);
    dir = dir.mul(r1 / dir.abs()).rot90();
    list.add(new P[]{c1.add(dir), c2.add(dir)});
    list.add(new P[]{c1.sub(dir), c2.sub(dir)});
  } else {
    P p = c1.mul(-r2).add(c2.mul(r1)).div(r1 - r2);
    P[] ps = tanCP(c1, r1, p);
    P[] qs = tanCP(c2, r2, p);
    for (int i = 0; i < ps.length && i < qs.length; i++) {
      list.add(new P[]{ps[i], qs[i]});
    }
  }
  P p = c1.mul(r2).add(c2.mul(r1)).div(r1 + r2);
  P[] ps = tanCP(c1, r1, p);
  P[] qs = tanCP(c2, r2, p);
  for (int i = 0; i < ps.length && i < qs.length; i++) {
    list.add(new P[]{ps[i], qs[i]});
  }
  return list.toArray(new P[0][]);
}
两圆公共部分的面积
double areaCC(P c1, double r1, P c2, double r2) {
  double d = c1.sub(c2).abs();
  if (r1 + r2 < d + EPS) return 0;
  if (d < Math.abs(r1 - r2) + EPS) {
    double r = Math.min(r1, r2);
    return r * r * Math.PI;
  }
  double x = (d * d + r1 * r1 - r2 * r2) / (2 * d);
  double t1 = Math.acos(x / r1);
  double t2 = Math.acos((d - x) / r2);
  return r1 * r1 * t1 + r2 * r2 * t2 - d * r1 * Math.sin(
  t1);
}
以r为半径的圆O与三角形Op1p2的公共面积
O为坐标原点
注意返回值可能为负
double areaCT(double r, P p1, P p2) {
  P[] qs = isCL(new P(0, 0), r, p1, p2);
  if (qs.length == 0) return r * r * rad(p1, p2) / 2;
  boolean b1 = p1.abs() > r + EPS, b2 = p2.abs() > r + EPS
  ;
  if (b1 && b2) {
    if (p1.sub(qs[0]).dot(p2.sub(qs[0])) < EPS &&
        p1.sub(qs[1]).dot(p2.sub(qs[1])) < EPS) {
      return (r * r * (rad(p1, p2) - rad(qs[0], qs[1])) +
      qs[0].det(qs[1])) / 2;
    } else {
      return r * r * rad(p1, p2) / 2;
    }
  } else if (b1) {
    return (r * r * rad(p1, qs[0]) + qs[0].det(p2)) / 2;
  } else if (b2) {
    return (r * r * rad(qs[1], p2) + p1.det(qs[1])) / 2;
  } else {
    return p1.det(p2) / 2;
  }
}
```

10.4 Polygon Contains P

点在多边形内外的判定内部返回 1，边上返回 0，外部返回-1

```
int contains(P[] ps, P q) {
  int n = ps.length;
  int res = -1;
  for (int i = 0; i < n; i++) {
    P a = ps[i].sub(q), b = ps[(i + 1) % n].sub(q);
    if (a.y > b.y) { P t = a; a = b; b = t; }
    if (a.y < EPS && b.y > EPS && a.det(b) > EPS) {
      res = -res;
    }
    if (Math.abs(a.det(b)) < EPS && a.dot(b) < EPS)
      return 0;
  }
  return res;
}
```

10.5 Convex Hull

```
凸包
逆时针 不包含线上的点
如果需要包含线上的点 将 <= 0 改成 < 0
但是需要注意此时不能有重点
P[] convexHull(P[] ps) {
  int n = ps.length, k = 0;
  if (n <= 1) return ps;
  Arrays.sort(ps);
  P[] qs = new P[n * 2];
  for (int i = 0; i < n; qs[k++] = ps[i++]) {
    while (k > 1 && qs[k - 1].sub(qs[k - 2]).det(ps[i].sub
    (qs[k - 1])) < EPS)
      k--;
  }
  for (int i = n - 2, t = k; i >= 0; qs[k++] = ps[i--]) {
    while (k > t && qs[k - 1].sub(qs[k - 2]).det(ps[i].sub
    (qs[k - 1])) < EPS)
      k--;
  }
  P[] res = new P[k - 1];
  System.arraycopy(qs, 0, res, 0, k - 1);
  return res;
}
按相对于 p0 的极角逆时针排序
```

```
角度相同，则离 p0 距离更近的放在前面
class CmpByAngle implements Comparator<P> {
  P p0;
  CmpByAngle(P p0) {
    this.p0 = p0;
  }
  public int compare(P o1, P o2) {
    double det = o1.sub(p0).det(o2.sub(p0));
    if (det != 0) return det > 0 ? -1 : 1;
    double dis = add(o1.sub(p0).abs2(), -o2.sub(p0).abs2()
    );
    if (dis != 0) return dis > 0 ? 1 : -1;
    return 0;
  }
}
P[] convexHullByAngle(P[] ps) {
  int n = ps.length, k = 0;
  if (n <= 1) return ps;
  for (int i = 1; i < n; i++) {
    if (ps[i].y < ps[0].y ||
        ps[i].y == ps[0].y && ps[i].x < ps[0].x) {
      Algo.swap(ps, 0, i);
    }
  }
  Arrays.sort(ps, 1, n, new CmpByAngle(ps[0]));
  P[] qs = new P[n];
  for (int i = 0; i < n; qs[k++] = ps[i++]) {
    while (k > 1 && qs[k - 1].sub(qs[k - 2]).det(ps[i].sub
    (qs[k - 1])) < EPS)
      k--;
  }
  return Arrays.copyOf(qs, k);
}
```

10.6 Convex Cut

凸多边形的切断，返回 p1p2 左侧凸包

```
P[] convexCut(P[] ps, P p1, P p2) {
  int n = ps.length;
  ArrayList<P> res = new ArrayList<P>();
  for (int i = 0; i < n; i++) {
    int d1 = sig(p2.sub(p1).det(ps[i].sub(p1)));
    int d2 = sig(p2.sub(p1).det(ps[(i + 1) % n].sub(p1)));
    if (d1 >= 0) res.add(ps[i]);
    if (d1 * d2 < 0) res.add(isLL(p1, p2, ps[i], ps[(i +
    1) % n]));
  }
  return res.toArray(new P[0]);
}
```

10.7 Convex Diameter

凸多边形的直径 aka. 凸多边形上最远点的距离。$O(n)$

```
double convexDiameter(P[] ps) {
  int n = ps.length;
  int is = 0, js = 0;
  for (int i = 1; i < n; i++) {
    if (ps[i].x > ps[is].x) is = i;
    if (ps[i].x < ps[js].x) js = i;
  }
  double maxD = ps[is].sub(ps[js]).abs();
  int i = is, j = js;
  do {
    if (ps[(i + 1) % n].sub(ps[i]).det(ps[(j + 1) % n].sub
    (ps[j])) >= 0) {
      j = (j + 1) % n;
    } else {
      i = (i + 1) % n;
    }
    maxD = Math.max(maxD, ps[i].sub(ps[j]).abs());
  } while (i != is || j != js);
  return maxD;
}
```

10.8 Dis Convex P

凸多边形与外部点的距离

```
double disConvexP(P[] ps, P q) {
  int n = ps.length;
  int left = 0, right = n;
  while (right - left > 1) {
    int mid = (left + right) / 2;
    if (in(ps[(left + n - 1) % n], ps[left], ps[mid], ps[(
    mid + 1) % n], q)) {
      right = mid;
    } else {
      left = mid;
    }
  }
  return disSP(ps[left], ps[right % n], q);
}
boolean in(P p1, P p2, P p3, P p4, P q) {
  P o12 = p1.sub(p2).rot90();
  P o23 = p2.sub(p3).rot90();
  P o34 = p3.sub(p4).rot90();
  return in(o12, o23, q.sub(p2))
      || in(o23, o34, q.sub(p3))
      || in(o23, p3.sub(p2), q.sub(p2))
       && in(p2.sub(p3), o23, q.sub(p3));
}
boolean in(P p1, P p2, P q) {
  return p1.det(q) > -EPS && p2.det(q) < EPS;
}
```

11 Other

11.1 Scanner

```
class Scanner {
  BufferedReader br;
  StringTokenizer st;
  Scanner(InputStream in) {
    br = new BufferedReader(new InputStreamReader(in));
    eat("");
  }
  void eat(String s) {
    st = new StringTokenizer(s);
  }
  String nextLine() {
    try {
      return br.readLine();
    } catch (IOException e) {
      return null;
    }
  }
  boolean hasNext() {
    while (!st.hasMoreTokens()) {
      String s = nextLine();
      if (s == null)
        return false;
      eat(s);
    }
    return true;
  }
  String next() {
    hasNext();
    return st.nextToken();
  }
  int nextInt() {
    return Integer.parseInt(next());
  }
}
```

11.2 Date Time

```
int[] ds = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31,
30, 31 };
int days(int y, int m, int d) {
  m = (m + 9) % 12;
  y = y - m / 10;
  return 365 * y + y / 4 - y / 100 + y / 400 + (m * 306 +
  5) / 10 + (d - 1);
}
int[] nextDay(int y, int m, int d) {
  if (d < ds[m]) return new int[] { y, m, d + 1 };
  if (d == 28 && m == 2 && isLeapYear(y))
    return new int[] { y, 2, 29 };
  m++;
  if (m == 13) {
    m = 1;
    y++;
  }
```

```
  return new int[] { y, m, 1 };
}
boolean isLeapYear(int year) {
  return new GregorianCalendar().isLeapYear(year);
}
```

11.3 Comb Permutation

```
for (int comb = (1 << k) - 1; comb < 1 << n;) {
  //...
  int x = comb & -comb, y = comb + x;
  comb = ((comb & ~y) / x >>> 1) | y;
}
```

11.4 Next Permutation

```
boolean nextPermutation(int[] is) {
  int n = is.length;
  for (int i = n - 1; i > 0; i--) {
    if (is[i - 1] < is[i]) {
      int j = n;
      while (is[i - 1] >= is[--j]) ;
      swap(is, i - 1, j);
      reverse(is, i, n);
      return true;
    }
  }
  reverse(is, 0, n);
  return false;
}
```

11.5 Nth Element

```
void nth_element(int[] a, int low, int high, int n) {
  while (true) {
    int k = randomizedPartition(a, low, high);
    if (n < k) high = k;
    else if (n > k) low = k + 1;
    else return;
  }
}
int randomizedPartition(int[] a, int low, int high) {
  swap(a, low + (int) (Math.random() * (high - low)), high
   - 1);
  int separator = a[high - 1];
  int i = low - 1;
  for (int j = low; j < high; j++)
    if (a[j] <= separator)
      swap(a, ++i, j);
  return i;
}
```

11.6 Radix Sort

```
void radixSort(int[] a) {
  final int d = 8;
  final int w = 32;
  int[] t = new int[a.length];
  for (int p = 0; p < w / d; p++) {
    // counting-sort
    int[] cnt = new int[1 << d];
    for (int i = 0; i < a.length; i++)
      ++cnt[((a[i] ^ Integer.MIN_VALUE) >>> (d * p)) & ((1
       << d) - 1)];
    for (int i = 1; i < cnt.length; i++)
      cnt[i] += cnt[i - 1];
    for (int i = a.length - 1; i >= 0; i--)
      t[--cnt[((a[i] ^ Integer.MIN_VALUE) >>> (d * p)) &
      ((1 << d) - 1)]] = a[i];
    System.arraycopy(t, 0, a, 0, a.length);
  }
}
```

12 Other G

12.1 C++ Template

```
#include <bits/stdc++.h>
using namespace std;
#define INF 0x3F3F3F3F
#define MP(X,Y) make_pair(X,Y)
#define PB(X) push_back(X)
#define REP(X,N) for(int X=0;X<N;X++)
#define REP2(X,L,R) for(int X=L;X<=R;X++)
#define DEP(X,R,L) for(int X=R;X>=L;X--)
#define CLR(A,X) memset(A,X,sizeof(A))
#define IT iterator
typedef long long ll;
typedef unsigned long long ull;
typedef pair<int,int> PII;
typedef vector<PII> VII;
typedef vector<int> VI;
#define X first
#define Y second
#define lson(X) ((X)<<1)
#define rson(X) ((X)<<1|1)
//pb_ds
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
#include <ext/pb_ds/priority_queue.hpp>
using namespace __gnu_pbds;
typedef tree<int, null_type, greater<int>, rb_tree_tag,
tree_order_statistics_node_update > rb_tree_set;
typedef tree<int, int, greater<int>, rb_tree_tag,
tree_order_statistics_node_update > rb_tree;
#define PQ std::priority_queue
#define HEAP __gnu_pbds::priority_queue

//gnu_cxx
using namespace __gnu_cxx;
#include <ext/hash_map>
#include <ext/hash_set>
#include <ext/rope>

namespace __gnu_cxx { //hash_map
  template<>
  struct hash<pair<int, int> > {
    size_t operator()(const pair<int, int>& key) const {
      return key.first * key.second;
    }
  };
}
```

12.2 Usage

```
//=== Bitset ===
bitset<length> b;
b.any()   b中是否存在置为1的二进制位?
b.none()  b中不存在置为1的二进制位吗?
b.count() b中置为1的二进制位的个数
b.size()  b中二进制位的个数
b[pos]    访问b中在pos处的二进制位
b.test(pos) b中在pos处的二进制位是否为1?
b.set()   把b中所有二进制位都置为1
b.set(pos)  把b中在pos处的二进制位置为1
b.reset() 把b中所有二进制位都置为0
b.reset(pos)  把b中在pos处的二进制位置为0
b.flip()  把b中所有二进制位逐位取反
b.flip(pos) 把b中在pos处的二进制位取反
b.to_ulong()  用b中同样的二进制位返回一个ul值
os << b   把b中的位集输出到os流

//=== Tree ===
order_of_key(x)   求Rank
find_by_order(k)  求第K小
*第二项可能是null_mapped_type
Heap:
a.join(b);
pairing_heap_tag 或 thin_heap_tag

//=== Rope ===
push_back(x)  在末尾添加x
insert(pos,x)   在pos插入x
erase(pos,x)  从pos开始删除x个
replace(pos,x) 从pos开始换成x
substr(pos,x)   提取pos开始x个
at(x)/[x]     访问第x个元素
count(begin, end, char);
```

12.3 Fast IO

```
int Scan() {
```

```
  int res=0, ch;
  while(ch=getchar(), ch<'0'||ch>'9');
  res=ch-'0';
  while((ch=getchar())>='0'&&ch<='9')
    res=res*10+ch-'0';
  return res;
}
void Out(int a) {
  if(a>9)
    Out(a/10);
  putchar(a%10+'0');
}

fread(buff, 1, MAX_LEN, stdin);
fwrite(buff_out, 1, len_out, stdout);
```

12.4 Stack

```
#pragma comment(linker, "/STACK:1024000000,1024000000")

int size = 256 << 20; // 256MB
char *p = (char*)malloc(size) + size;
__asm__("movl %0, %%esp\n" :: "r"(p));

extern int main2(void) __asm__ ("_main2");
int main2() {
  char test[255 << 20];
  memset(test, 42, sizeof(test));
  printf(":)\n");
  exit(0);
}
int main() {
  int size = 256 << 20;  // 256Mb
  char *p = (char*)malloc(size) + size;
  __asm__("movl  %0, %%esp\n"
    "pushl $exit\n"  // if you get a compile error here
    under mingw gwin,
    "jmp   main2\n"  // replace exit with _exit, and main2
     with _main2.
    :: "r"(p));
}
__asm__ __volatile__(  // replace for 64-bit version
   "movq  %0, %%rsp\n"
   "pushq $exit\n"
   "jmp main2\n"
   :: "r"(p));

//test
int st(int s) {
  if(s==0) return 0;
  return min(s, st(s-1));
}
//ans += st(1e6);
```

12.5 Formula Set

线性规划

$$max\{cx \mid Ax \le b,\ x \ge 0\} = min\{yb \mid yA \ge c,\ y \ge 0\}$$

$$max\{cx \mid Ax = b,\ x \ge 0\} = min\{yb \mid yA \ge c\}$$

$$max\{cx \mid Ax = b\} = min\{yb \mid yA = c\}$$

三角形内心

$$\frac{a\overrightarrow{A} + b\overrightarrow{B} + c\overrightarrow{C}}{a+b+c}$$

三角形外心

$$(\overrightarrow{A} + \overrightarrow{B} - \frac{\overrightarrow{BC}\cdot\overrightarrow{CA}}{\overrightarrow{AB}\times\overrightarrow{BC}}\overrightarrow{AB}^T)/2$$

三角形垂心

$$\overrightarrow{H} = 3\overrightarrow{G} - 2\overrightarrow{O}$$

三角形外接圆半径

$$R = \frac{abc}{4S}$$

海伦公式

$$2s := a+b+c$$

$$S = \sqrt{s(s-a)(s-b)(s-c)}$$

皮克公式，简单多边形，面积 S，内部整点数 I，边上整点数 B

$$S = B/2 + I - 1$$

四面体 $O-ABC$ 体积与边长公式

$$a = AB, b = BC, c = CA, d = OC, e = OA, f = OB$$

$$(12V)^2 = a^2d^2(b^2+c^2+e^2+f^2-a^2-d^2)+b^2e^2(c^2+a^2+f^2+d^2-b^2-e^2)$$
$$+c^2f^2(a^2+b^2+d^2+e^2-c^2-f^2)-a^2b^2c^2-a^2e^2f^2-d^2b^2f^2-d^2e^2c^2$$

Polya

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

泰勒展开

$$e^x = \sum_{n=0}^{\infty}\frac{x^n}{n!}$$

$$sin\ x = \sum_{n=0}^{\infty}\frac{(-1)^n}{(2n+1)!}x^{2n+1}$$

$$cos\ x = \sum_{n=0}^{\infty}\frac{(-1)^n}{(2n)!}x^{2n}$$

辛普森积分 $h=(b-a)/n$、$x_i = a+ih$，误差 $O(h^4)$

$$\int_a^b f(x)dx \approx \frac{h}{3}\left[f(x_0) + 2\sum_{j=1}^{n/2-1}f(x_{2j}) + 4\sum_{j=1}^{n/2}f(x_{2j-1}) + f(x_n)\right]$$

莫比乌斯

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

$$g(x) = \sum_{n=1}^{\lfloor x\rfloor}f(\frac{x}{n}) \Leftrightarrow f(x) = \sum_{n=1}^{\lfloor x\rfloor}\mu(n)g(\frac{x}{n})$$

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

高神公式

$$\sum_{d|n}\mu(d) = [n=1]$$

$$\sum_{k=0}^{n}(-1)^k\binom{n}{k} = [n=0]$$

$$\sum_{r\subseteq p}(-1)^{|r|} = [p=0]$$

$$M(i) = \sum_{1}^{n}\mu(i) = 1 - \sum_{2}^{n}M(n/i)$$

模幂

$$a^e\ mod\ m = \begin{cases} a^e\ mod\ m & if\ e < \phi(m) \\ a^{e\ mod\ \phi(m)+\phi(m)}\ mod\ m & otherwise \end{cases}$$

Catalan 数

$$h(n) = C(2n,n) - C(2n,n-1) = C(2n,n)/(n+1)$$

Bell 数，S 是第二类 Stirling 数

$$B_n = \sum_{k=0}^{n-1}C_{n-1}^{k}B_k = \sum_{k=1}^{n}S(n,k)$$

Bell 还有两个重要的同余性质：

$$B_{p+n} \equiv B_n + B_{n+1}\ (mod\ p)$$

$$B_{pm+n} \equiv mB_n + B_{n+1}\ (mod\ p)$$

其中这里的 p 是不大于 100 的素数，这样，我们可以通过上面的性质来计算 Bell 数模小于 100 的素数值。

Bell 数模素数 p 的周期为：$N_p = \frac{p^p-1}{p-1}$

第一类 Stirling 数 $s(p,k)$

$s(p,k)$ 的一个的组合学解释是：将 p 个物体排成 k 个非空循环排列的方法数。

$$s(p,k) = (p-1)*s(p-1,k) + s(p-1,k-1), 1 \le k \le p-1$$

$$s(p,0) = 0, p \ge 1, s(p,p) = 1, p \ge 0$$

递推关系的说明：考虑第 p 个物品，p 可以单独构成一个非空循环排列，这样前 $p-1$ 种物品构成 $k-1$ 个非空循环排列，方法数为 $s(p-1,k-1)$；也可以前 $p-1$ 种物品构成 k 个非空循环排列，而第 p 个物品插入第 i 个物品的左边，这有 $(p-1)*s(p-1,k)$ 种方法。

第二类 Stirling 数 $S(p,k)$

$S(p,k)$ 的一个组合学解释是：将 p 个物体划分成 k 个非空的不可辨别的（可以理解为盒子没有编号）集合的方法数。$k!S(p,k)$ 是把 p 个人分进 k 间有差别 (如：被标有房号）的房间 (无空房）的方法数。

$$S(p,k) = k*S(p-1,k) + S(p-1,k-1), 1 \le k \le p-1$$

$$S(p,p)=1,p\geq 0,S(p,0)=0,p\geq 1$$

递推关系的说明：考虑第 p 个物品，p 可以单独构成一个非空集合，此时前 $p-1$ 个物品构成 $k-1$ 个非空的不可辨别的集合，方法数为 $S(p-1,k-1)$；也可以前 $p-1$ 种物品构成 k 个非空的不可辨别的集合，第 p 个物品放入任意一个中，这样有 $k*S(p-1,k)$ 种方法。

将 n 个的小球放入 r 个的盒子

小球 | 盒子 | 空盒 | 方案数
相异 | 相异 | 允许 | r^n
相异 | 相同 | 不许 | $S_2(n,r)$
相异 | 相异 | 不许 | $r!\ S_2(n,r)$
相异 | 相同 | 允许 | $\sum_{k=1}^{r} S_2(n,k)$
相同 | 相异 | 允许 | C_{n+r-1}^{n}
相同 | 相异 | 不许 | C_{n-1}^{r-1}
相同 | 相同 | 不许 | $P_r(n)$
相同 | 相同 | 允许 | $P_r(n+r)$

从上之下编号 1-8 的话，1 最简单，2 是第二类 Stirling 数，2 继而推出 3、4。5 和 6 用插板法可以推出，7 和 8 是整数拆分问题。

第 1 种：n 个球，每个球都有 r 种不同的选择。

第 2 种：n 个相异的小球放入 r 个相同的盒子，不允许空盒，记为 $S(n,r)$。

第 3 种：相当于先进行第 2 种之后，再对所有的盒子做全排列。

第 4 种：枚举有几个空盒。

第 5 种：在 $(n+r-1)$ 个位置中，选出 n 个作为小球，剩下的位置自然变成盒子的边界。

第 6 种：在 $(n-1)$ 个位置中，选出 $(r-1)$ 个作为盒子的边界。

第 7 种：就是正整数 n 的 r 划分。而 $P_k(n)=P_{k-1}(n-1)+P_k(n-k)$（分是否包含整数 1 来讨论）。

第 8 种：允许空集合，那么相当于把每个集合都先加 1，然后再做不允许空集合的划分。

图论相关

在没有孤立点的图中，| 最大边独立集 | + | 最小边覆盖 | = |V|

一般图中，| 最大点独立集 | + | 最小点覆盖 | = |V|

一般图的最大边独立集可以使用 Tutte 矩阵来计算。一般图的最大点独立集是 NP-hard 的。

二分图中，最大边独立集可以使用匈牙利算法计算，而 | 最小点覆盖 | = | 最大边独立集 |，于是在一般图中 NP-hard 的最大点独立集问题，在二分图中也有多项式算法了。

一般图中，最大点权独立集的权值和 + 最小点权覆盖的权值和 = 所有点权值和

现在还没有发现最小边权覆盖和最大边权独立集之间的关系。

二分图的最小点权覆盖

最小点权覆盖即选出一个点集 V，使得原图中的边 <u, v>，u ∈ V, v ∈ V 至少有一个成立，同时最小化点集 V 的权值之和。

那么考虑如下最小割建图：

- 增加源点 s，汇点 t，从 s 连边到 X 部中的点 x，权值为 x 的权值；
- 从 Y 部中的点 y 连边到 t，权值为 y 的权值；
- 对原图中的所有边 <u, v>，从 u 连边到 v，权值为 INF。

这样建图后考虑最小割，<s, u> 和 <v, t> 至少有一个在最小割中，因为 <u, v> 权值为 INF，一定不会出现在最小割中。这正好与“u ∈ V, v ∈ V 至少有一个成立”形式相符。

2-SAT

2-SAT 是指合取范式形如：

$$(a\ \vee\ b)\wedge(c\ \vee\ d)\wedge\ldots\wedge(e\ \vee\ f)$$

将每个 $(a\ \vee\ b)$ 写成 $(\neg a\rightarrow b\wedge\neg b\rightarrow a)$。

对于每个变量 x，构造两个点 x 和 $\neg x$，以 $\rightarrow$ 为关系建有向图。则图中的一个强连通分量的真假性相同。

如果存在 x，x 和 $\neg x$ 在一个强连通分量中，则无解。如果不存在，则有解。构造一组解：

x 所在的强连通分量的拓扑序在 $\neg x$ 之后 $\Leftrightarrow$ x 为真

```
1  DP？贪心？
2
3  网络流？割？
4
5  线性规划？
6
7  逆向思维？
8
9  二分搜索？三分搜索？
10
11 小数据规律？
12
13 作假设？
```