1 DP

1.1 Bounded_Knapsack

```cpp
namespace {
    static const int MAXW = 1000005;
    static const int MAXN = 1005;
    struct BB {
        int w, v, c;
        BB(int w = 0, int v = 0, int c = 0): w(w), v(v), c(c) {}
        bool operator<(const BB &x) const {
            return w * c  < x.w * x.c;
        }
    };
    static int run(BB A[], int dp[], int W, int N) {
        static int MQ[MAXW][2];
        for (int i = 0, sum = 0; i < N; i++) {
            int w = A[i].w, v = A[i].v, c = A[i].c;
            sum = min(sum + w*c, W);
            for (int j = 0; j < w; j++) {
                int l = 0, r = 0;
                MQ[l][0] = 0, MQ[l][1] = dp[j];
                for (int k = 1, tw = w+j, tv = v; tw <= sum && k <= c; k++, tw += w, tv += v) {
                    int dpv = dp[tw] - tv;
                    while (l <= r && MQ[r][1] <= dpv) r--;
                    r++;
                    MQ[r][0] = k, MQ[r][1] = dpv;
                    dp[tw] = max(dp[tw], MQ[l][1] + tv);
                }
                for (int k = c+1, tw = (c+1)*w+j, tv = (c+1)*v; tw <= sum; k++, tw += w, tv += v) {
                    if (k - MQ[l][0] > c) l++;
                    int dpv = dp[tw] - tv;
                    while (l <= r && MQ[r][1] <= dpv) r--;
                    r++;
                    MQ[r][0] = k, MQ[r][1] = dpv;
                    dp[tw] = max(dp[tw], MQ[l][1] + tv);
                }
            }
        }
    }
    static int knapsack(int C[][3], int N, int W) { // O(WN)
        vector<BB> A;
        for (int i = 0; i < N; i++) {
            int w = C[i][0], v = C[i][1], c = C[i][2];
            A.push_back(BB(w, v, c));
        }
        assert(N < MAXN);
        static int dp1[MAXW+1], dp2[MAXW+1];
        BB Ar[2][MAXN];
        int ArN[2] = {};
        memset(dp1, 0, sizeof(dp1[0])*(W+1));
        memset(dp2, 0, sizeof(dp2[0])*(W+1));
        sort(A.begin(), A.end());
        int sum[2] = {};
        for (int i = 0; i < N; i++) {
            int ch = sum[1] < sum[0];
            Ar[ch][ArN[ch]] = A[i];
            ArN[ch]++;
            sum[ch] = min(sum[ch] + A[i].w*A[i].c, W);
        }
        run(Ar[0], dp1, W, ArN[0]);
        run(Ar[1], dp2, W, ArN[1]);
        int ret = 0;
        for (int i = 0, j = W, mx = 0; i <= W; i++, j--) {
            mx = max(mx, dp2[i]);
            ret = max(ret, dp1[j] + mx);
        }
        return ret;
    }
}
int main() {
    int W, N;
    assert(scanf("%d %d", &W, &N) == 2);
    int C[MAXN][3];
    for (int i = 0; i < N; i++)
        assert(scanf("%d %d %d", &C[i][1], &C[i][0], &C[i][2]) == 3);
    printf("%d\n", knapsack(C, N, W));
    return 0;
}
```

1.2 DP_1D1D

```cpp
int t, n, L, p;
char s[MAXN][35];
ll sum[MAXN] = {0};
long double dp[MAXN] = {0};
int prevd[MAXN] = {0};
long double pw(long double a, int n) {
    if ( n == 1 ) return a;
    long double b = pw(a, n/2);
    if ( n & 1 ) return b*b*a;
    else return b*b;
}
long double f(int i, int j) {
    // cout << (sum[i] - sum[j]+i-j-1-L) << endl;
    return pw(abs(sum[i] - sum[j]+i-j-1-L), p) + dp[j];
}
struct INV {
    int L, R, pos;
};
INV stk[MAXN*10];
int top = 1, bot = 1;
void update(int i) {
    while ( top > bot && i < stk[top].L && f(stk[top].L, i) < f(stk[top].L, stk[top].pos) ) {
        stk[top - 1].R = stk[top].R;
        top--;
    }
    int lo = stk[top].L, hi = stk[top].R, mid, pos = stk[top].pos;
    // if ( i >= lo ) lo = i + 1;
    while ( lo != hi ) {
        mid = lo + (hi - lo) / 2;
        if ( f(mid, i) < f(mid, pos) ) hi = mid;
        else lo = mid + 1;
    }
    if ( hi < stk[top].R ) {
        stk[top + 1] = (INV) { hi, stk[top].R, i };
        stk[top++].R = hi;
    }
}
int main() {
    cin >> t;
    while ( t-- ) {
        cin >> n >> L >> p;
        dp[0] = sum[0] = 0;
        for ( int i = 1 ; i <= n ; i++ ) {
            cin >> s[i];
            sum[i] = sum[i-1] + strlen(s[i]);
            dp[i] = numeric_limits<long double>::max();
        }
        stk[top] = (INV) {1, n + 1, 0};
        for ( int i = 1 ; i <= n ; i++ ) {
            if ( i >= stk[bot].R ) bot++;
            dp[i] = f(i, stk[bot].pos);
            update(i);
            // cout << (ll) f(i, stk[bot].pos) << endl;
        }
        if ( dp[n] > 1e18 ) {
            cout << "Too hard to arrange" << endl;
        } else {
            vector<PI> as;
            cout << (ll)dp[n] << endl;
        }
    } return 0;
}
```

1.3 LCIS

```cpp
vector<int> LCIS(vector<int> a, vector<int> b) {
    int n = a.size(), m = b.size();
    int dp[LEN][LEN] = {}, pre[LEN][LEN] = {};
    for(int i=1; i<=n; i++) {
        int p = 0;
        for(int j=1; j<=m; j++)
            if(a[i-1]!=b[j-1]) {
                dp[i][j] = dp[i-1][j], pre[i][j] = j;
                if( a[i-1]>b[j-1] && dp[i-1][j]>dp[i-1][p] )
                    p = j;
            } else {
                dp[i][j] = dp[i-1][p]+1, pre[i][j] = p;
            }
    }
    int len = 0, p = 0;
    for(int j=1; j<=m; j++)
        if(dp[n][j]>len) len = dp[n][j], p = j;
    vector<int> ans;
    for(int i=n; i>=1; i--) {
        if(a[i-1]==b[p-1]) ans.push_back(b[p-1]);
        p = pre[i][p];
    }
    reverse(ans.begin(), ans.end());
    return ans;
}
```

2 Data_Structure

2.1 Dynamic_KD_tree

```cpp
template<typename T,size_t kd>//有kd個維度
struct kd_tree{
  struct point{
    T d[kd];
    T dist(const point &x)const{
      T ret=0;
      for(size_t i=0;i<kd;++i)ret+=abs(d[i]-x.d[i]);
      return ret;
    }
    bool operator==(const point &p){
      for(size_t i=0;i<kd;++i)
        if(d[i]!=p.d[i])return 0;
      return 1;
    }
    bool operator<(const point &b)const{
      return d[0]<b.d[0];
    }
  };
private:
  struct node{
    node *l,*r;
    point pid;
    int s;
    node(const point &p):l(0),r(0),pid(p),s(1){}
    ~node(){delete l,delete r;}
    void up(){s=(l?l->s:0)+1+(r?r->s:0);}
  }*root;
  const double alpha,loga;
  const T INF;//記得要給INF，表示極大值
  int maxn;
  struct __cmp{
    int sort_id;
    bool operator()(const node*x,const node*y)const{
      return operator()(x->pid,y->pid);
    }
    bool operator()(const point &x,const point &y)const{
      if(x.d[sort_id]!=y.d[sort_id])
        return x.d[sort_id]<y.d[sort_id];
      for(size_t i=0;i<kd;++i)
        if(x.d[i]!=y.d[i])return x.d[i]<y.d[i];
      return 0;
    }
  }cmp;
  int size(node *o){return o?o->s:0;}
  vector<node*> A;
  node* build(int k,int l,int r){
    if(l>r) return 0;
    if(k==kd) k=0;
    int mid=(l+r)/2;
    cmp.sort_id = k;
    nth_element(A.begin()+l,A.begin()+mid,A.begin()+r+1,cmp);
    node *ret=A[mid];
    ret->l = build(k+1,l,mid-1);
    ret->r = build(k+1,mid+1,r);
    ret->up();
    return ret;
  }
  bool isbad(node*o){
    return size(o->l)>alpha*o->s||size(o->r)>alpha*o->s;
  }
  void flatten(node *u,typename vector<node*>::iterator &it){
    if(!u)return;
    flatten(u->l,it);
    *it=u;
    flatten(u->r,++it);
  }
  void rebuild(node*&u,int k){
    if((int)A.size()<u->s)A.resize(u->s);
    auto it=A.begin();
    flatten(u,it);
    u=build(k,0,u->s-1);
  }
  bool insert(node*&u,int k,const point &x,int dep){
    if(!u) return u=new node(x), dep<=0;
    ++u->s;
    cmp.sort_id=k;
    if(insert(cmp(x,u->pid)?u->l:u->r,(k+1)%kd,x,dep-1)){
      if(!isbad(u))return 1;
      rebuild(u,k);
    }
    return 0;
  }
  node *findmin(node*o,int k){
    if(!o)return 0;
    if(cmp.sort_id==k)return o->l?findmin(o->l,(k+1)%kd):o;
    node *l=findmin(o->l,(k+1)%kd);
    node *r=findmin(o->r,(k+1)%kd);
    if(l&&!r)return cmp(l,o)?l:o;
    if(!l&&r)return cmp(r,o)?r:o;
    if(!l&&!r)return o;
    if(cmp(l,r))return cmp(l,o)?l:o;
    return cmp(r,o)?r:o;
  }
  bool erase(node *&u,int k,const point &x){
    if(!u)return 0;
    if(u->pid==x){
      if(u->r);
      else if(u->l) u->r=u->l, u->l=0;
      else return delete(u),u=0, 1;
      --u->s;
      cmp.sort_id=k;
      u->pid=findmin(u->r,(k+1)%kd)->pid;
      return erase(u->r,(k+1)%kd,u->pid);
    }
    cmp.sort_id=k;
    if(erase(cmp(x,u->pid)?u->l:u->r,(k+1)%kd,x))
      return --u->s, 1;
    return 0;
  }
  T heuristic(const T h[])const{
    T ret=0;
    for(size_t i=0;i<kd;++i)ret+=h[i];
    return ret;
  }
  int qM;
  priority_queue<pair<T,point>> pQ;
  void nearest(node *u,int k,const point &x,T *h,T &mndist){
    if(u==0||heuristic(h)>=mndist)return;
    T dist=u->pid.dist(x),old=h[k];
    /*mndist=std::min(mndist,dist);*/
    if(dist<mndist){
      pQ.push(std::make_pair(dist,u->pid));
      if((int)pQ.size()==qM+1)
        mndist=pQ.top().first,pQ.pop();
    }
    if(x.d[k]<u->pid.d[k]){
      nearest(u->l,(k+1)%kd,x,h,mndist);
      h[k] = abs(x.d[k]-u->pid.d[k]);
      nearest(u->r,(k+1)%kd,x,h,mndist);
    }else{
      nearest(u->r,(k+1)%kd,x,h,mndist);
      h[k] = abs(x.d[k]-u->pid.d[k]);
      nearest(u->l,(k+1)%kd,x,h,mndist);
    }
    h[k]=old;
  }
  vector<point>in_range;
  void range(node *u,int k,const point&mi,const point&ma){
    if(!u)return;
    bool is=1;
    for(int i=0;i<kd;++i)
      if(u->pid.d[i]<mi.d[i]||ma.d[i]<u->pid.d[i])
        { is=0;break; }
    if(is) in_range.push_back(u->pid);
    if(mi.d[k]<=u->pid.d[k])range(u->l,(k+1)%kd,mi,ma);
    if(ma.d[k]>=u->pid.d[k])range(u->r,(k+1)%kd,mi,ma);
  }
public:
  kd_tree(const T &INF,double a=0.75):
  root(0),alpha(a),loga(log2(1.0/a)),INF(INF),maxn(1){}
  ~kd_tree(){delete root;}
  void clear(){delete root,root=0,maxn=1;}
  void build(int n,const point *p){
    delete root,A.resize(maxn=n);
    for(int i=0;i<n;++i)A[i]=new node(p[i]);
    root=build(0,0,n-1);
  }
  void insert(const point &x){
    insert(root,0,x,__lg(size(root))/loga);
    if(root->s>maxn)maxn=root->s;
  }
  bool erase(const point &p){
    bool d=erase(root,0,p);
    if(root&&root->s<alpha*maxn)rebuild();
    return d;
  }
  void rebuild(){
    if(root)rebuild(root,0);
    maxn=root->s;
  }
  T nearest(const point &x,int k){
    qM=k;
    T mndist=INF,h[kd]={};
    nearest(root,0,x,h,mndist);
    mndist=pQ.top().first;
    pQ = priority_queue<pair<T,point>>();
    return mndist;//回傳離x第k近的點的距離
  }
  const vector<point> &range(const point&mi,const point&ma){
    in_range.clear();
    range(root,0,mi,ma);
    return in_range;//回傳介於mi到ma之間的點vector
  }
  int size(){return root?root->s:0;}
};
```

2.2 HeavyLight

```cpp
#include<vector>
#define MAXN 100005
int siz[MAXN],max_son[MAXN],pa[MAXN],dep[MAXN];
int link_top[MAXN],link[MAXN],cnt;
vector<int> G[MAXN];
```

```cpp
void find_max_son(int u){
  siz[u]=1;
  max_son[u]=-1;
  for(auto v:G[u]){
    if(v==pa[u])continue;
    pa[v]=u;
    dep[v]=dep[u]+1;
    find_max_son(v);
    if(max_son[u]==-1||siz[v]>siz[max_son[u]])max_son[u]=v;
    siz[u]+=siz[v];
  }
}
void build_link(int u,int top){
  link[u]=++cnt;
  link_top[u]=top;
  if(max_son[u]==-1)return;
  build_link(max_son[u],top);
  for(auto v:G[u]){
    if(v==max_son[u]||v==pa[u])continue;
    build_link(v,v);
  }
}
int find_lca(int a,int b){
  //求LCA，可以在過程中對區間進行處理
  int ta=link_top[a],tb=link_top[b];
  while(ta!=tb){
    if(dep[ta]<dep[tb]){
      swap(ta,tb);
      swap(a,b);
    }
    //這裡可以對a所在的鏈做區間處理
    //區間為(link[ta],link[a])
    ta=link_top[a=pa[ta]];
  }
  //最後a,b會在同一條鏈，若a!=b還要在進行一次區間處理
  return dep[a]<dep[b]?a:b;
}
```

2.3 SegmentTree

```cpp
/** 普通線段樹，為了加速打字時間，所以只支援 1-based。 **/
/**
 * 把 df 設為:
 *   0     for 區間和/gcd/bit-or/bit-xor
 *   1     for 區間積/lcm
 *   9e18  for 區間最小值
 *   -9e18 for 區間最大值
 *   -1    for 區間 bit-and
 */
const ll df = 0;
const int N = ? ;  // maxn
#define ls i << 1  // 加速打字
#define rs i << 1 | 1
struct SegmentTree {
    ll a[N << 2];
    inline ll cal(ll a, ll b) {
        /**
         * 把回傳值設為對應的操作，例如 a+b 為區間和，還有像是
         * a*b, min(a,b), max(a,b), gcd(a,b), lcm(a,b),
         * a|b, a&b, a^b 等等。 */
            return a + b;
    }
    // 單點設值。外部呼叫的時候後三個參數不用填。注意只支援
    // 1-based ！
    ll set(int q, ll v, int i = 1, int l = 1, int r = N) {
        if (r < q || l > q) return a[i];
        if (l == r) return a[i] = v;
        int m = (l + r) >> 1;
        ll lo = set(q, v, ls, l, m);
        ll ro = set(q, v, rs, m + 1, r);
        return a[i] = cal(lo, ro);
    }
    // 查詢區間 [l, r] 總和
    // (或極值等等，看你怎麼寫)。外部呼叫的時
    // 候後三個參數不用填。注意只支援 1-based ！
    ll query(int ql, int qr, int i = 1, int l = 1,
             int r = N) {
        if (r < ql || l > qr) return df;
        if (ql <= l && r <= qr) return a[i];
        int m = (l + r) >> 1;
        ll lo = query(ql, qr, ls, l, m);
        ll ro = query(ql, qr, rs, m + 1, r);
        return cal(lo, ro);
    }
    // 建立 size = N 的空線段樹，所有元素都是 0 。注意只支援
    // 1-based ！
    SegmentTree() { memset(a, 0, sizeof(a)); }
};
```

2.4 MaxSumSegmentTree

```cpp
/** 計算最大子區間連續和的線段樹，限定 1-based 。
 * 複雜度 O(Q*log(N)) **/
#define ls i << 1
#define rs i << 1 | 1
class MaxSumSegmentTree {
   private:
    struct node {
        ll lss, rss, ss, ans;
        void set(ll v) { lss = rss = ss = ans = v; }
    };
    int n;
    vector<node> a;  // 萬萬不可用普通陣列，要用 vector
    vector<ll> z;
    void pull(int i) {
        a[i].ss = a[ls].ss + a[rs].ss;
        a[i].lss = max(a[ls].lss, a[ls].ss + a[rs].lss);
        a[i].rss = max(a[rs].rss, a[rs].ss + a[ls].rss);
        a[i].ans = max(max(a[ls].ans, a[rs].ans),
                       a[ls].rss + a[rs].lss);
    }
    void build(int i, int l, int r) {
        if (l == r) return a[i].set(z[l]), void();
        int m = (l + r) >> 1;
        build(ls, l, m), build(rs, m + 1, r), pull(i);
    }
    void set(int i, int l, int r, int q, ll v) {
        if (l == r) return a[i].set(v), void();
        int m = (l + r) >> 1;
        if (q <= m) set(ls, l, m, q, v);
        else set(rs, m + 1, r, q, v);
        pull(i);
    }
    node query(int i, int l, int r, int ql, int qr) {
        if (ql <= l && r <= qr) return a[i];
        int m = (l + r) >> 1;
        if (qr <= m) return query(ls, l, m, ql, qr);
        if (m < ql) return query(rs, m + 1, r, ql, qr);
        node lo = query(ls, l, m, ql, qr),
             ro = query(rs, m + 1, r, ql, qr), ans;
        ans.ss = lo.ss + ro.ss;
        ans.lss = max(lo.lss, lo.ss + ro.lss);
        ans.rss = max(ro.rss, ro.ss + lo.rss);
        ans.ans = max(max(lo.ans, ro.ans), lo.rss + ro.lss);
        return ans;
    }

   public:
    MaxSumSegmentTree(int n) : n(n) {
        a.resize(n << 2), z.resize(n << 2);
        build(1, 1, n);
    }
    // 單點設值。限定 1-based 。
    inline void set(int i, ll v) { set(1, 1, n, i, v); }
    // 問必區間 [l, r] 的最大子區間連續和。限定 1-based 。
    inline ll query(int l, int r) {
        return query(1, 1, n, l, r).ans;
    }
};
```

2.5 FenwickTree2D

```cpp
/** 支援單點增值和區間查詢，O((A+Q)*log(A))，A
 * 是矩陣面積。只能 用於 1-based **/
const int R = 256, C = 256;
class BIT2D {
   private:
    ll a[R + 1][C + 1];
    ll sum(int x, int y) {
        ll ret = 0;
        for (int i = x; i; i -= (i & -i))
            for (int j = y; j; j -= (j & -j))
                ret += a[i][j];
        return ret;
    }
   public:
    // 建立元素都是零的 R*C 大小的矩陣。
    BIT2D() { memset(a, 0, sizeof(a)); }
    // 單點增值，注意 1-based 。
    void add(int x, int y, ll v) {
        for (int i = x; i <= R; i += (i & -i))
            for (int j = y; j <= C; j += (j & -j))
                a[i][j] += v;
    }
    // 區間和，注意 1-based 。二維都是閉區間。
    ll sum(int x0, int y0, int x1, int y1) {
        return sum(x1, y1) - sum(x0 - 1, y1) -
               sum(x1, y0 - 1) + sum(x0 - 1, y0 - 1);
    }
};
```

2.6 PersistentSegmentTree

```cpp
int a[maxn], b[maxn], root[maxn], cnt;
struct node {
    int sum, L_son, R_son;
} tree[maxn << 5];
int create(int _sum, int _L_son, int _R_son) {
    int idx = ++cnt;
    tree[idx].sum = _sum, tree[idx].L_son = _L_son, tree[idx].R_son = _R_son;
    return idx;
}
void Insert(int &root, int pre_rt, int pos, int L, int R) {
    root = create(tree[pre_rt].sum+1, tree[pre_rt].L_son, tree[pre_rt].R_son);
    if(L==R) return;
    int M = (L+R)>>1;
    if(pos<=M) Insert(tree[root].L_son, tree[pre_rt].L_son, pos, L, M);
    else Insert(tree[root].R_son, tree[pre_rt].R_son, pos, M+1, R);
}
int query(int L_id, int R_id, int L, int R, int K) {
    if(L==R) return L;
    int M = (L+R)>>1;
    int s = tree[tree[R_id].L_son].sum - tree[tree[L_id].L_son].sum;
    if(K<=s) return query(tree[L_id].L_son, tree[R_id].L_son, L, M, K);
    return query(tree[L_id].R_son, tree[R_id].R_son, M+1, R, K-s);
}
int main() {
    int n,m; cin >> n >> m
    for(int i=1; i<=n; i++) {
        cin >> a[i]; b[i] = a[i];
    } sort(b+1,b+1+n); //離散化
    int b_sz = unique(b+1, b+1+n) - (b+1);
    cnt = root[0] = 0;
    for(int i=1; i<=n; i++) {
        int pos = lower_bound(b+1, b+1+b_sz, a[i]) - b;
        Insert(root[i], root[i-1], pos, 1, b_sz);
    }
    while(m--) {
        int l, r, k; cin >> l >> r >> k;
        int pos = query(root[l-1],root[r],1,b_sz,k);
        cout << b[pos] << endl;
    } return 0;
}
```

2.7 RangeUpdateSegmentTree

```cpp
/**
 * 修改功能最強的線段樹，但只能查詢區間和以及極值，所有區間操作都
 * 是閉區間。只支援 1-based 。 **/
#define ls i << 1
#define rs i << 1 | 1
const ll rr = 0x6891139;  // 亂數，若跟題目碰撞會吃 WA 或 RE
class RangeUpdateSegmentTree {
   private:
    // 程式碼重複性略高 (已盡力)。若不需要區間和，刪除所有含有 .s
    // 的行；若不需要 max ，刪除所有含有 .x 的行。
    struct node {
        int l, r; ll adt = 0, stt = rr, s = 0, x = 0;
    };
    vector<node> a; // 萬萬不可以用普通陣列，要用 vector
    void push(int i) {
        if (a[i].stt != rr) {
            a[ls].stt = a[rs].stt = a[i].stt;
            a[ls].adt = a[rs].adt = 0;
            a[ls].x = a[rs].x = a[i].stt;
            a[ls].s = (a[ls].r - a[ls].l + 1) * a[i].stt;
            a[rs].s = (a[rs].r - a[rs].l + 1) * a[i].stt;
            a[i].stt = rr;
        }
        if (a[i].adt) {
            a[ls].adt += a[i].adt, a[rs].adt += a[i].adt;
            a[ls].x += a[i].adt, a[rs].x += a[i].adt;
            a[ls].s += a[i].adt * (a[ls].r - a[ls].l + 1);
            a[rs].s += a[i].adt * (a[rs].r - a[rs].l + 1);
            a[i].adt = 0;
        }
    }
    void pull(int i) {
        a[i].s = a[ls].s + a[rs].s;
        a[i].x = max(a[ls].x, a[rs].x);
    }
    void build(int l, int r, int i) {
        a[i].l = l, a[i].r = r;
        if (l == r) return;
        int mid = (l + r) >> 1;
        build(l, mid, ls), build(mid + 1, r, rs);
    }
   public:
    RangeUpdateSegmentTree(int n) : a(n << 2) {
        build(1, n, 1);
    }
    void set(int l, int r, ll val, int i = 1) {
        if (a[i].l >= l && a[i].r <= r) {
            a[i].s = val * (a[i].r - a[i].l + 1);
            a[i].x = a[i].stt = val;
            a[i].adt = 0;
            return;
        }
        push(i);
        int mid = (a[i].l + a[i].r) >> 1;
        if (l <= mid) set(l, r, val, ls);
        if (r > mid) set(l, r, val, rs);
        pull(i);
    }
    void add(int l, int r, ll val, int i = 1) {
        if (a[i].l >= l && a[i].r <= r) {
            a[i].s += val * (a[i].r - a[i].l + 1);
            a[i].x += val;
            a[i].adt += val;
            return;
        }
        push(i);
        int mid = (a[i].l + a[i].r) >> 1;
        if (l <= mid) add(l, r, val, ls);
        if (r > mid) add(l, r, val, rs);
        pull(i);
    }
    ll maxx(int l, int r, int i = 1) {
        if (l <= a[i].l && a[i].r <= r) return a[i].x;
        push(i);
        ll ret = -9e18;
        int mid = (a[i].l + a[i].r) >> 1;
        if (l <= mid) ret = max(ret, maxx(l, r, ls));
        if (r > mid) ret = max(ret, maxx(l, r, rs));
        pull(i);
        return ret;
    }
    ll sum(int l, int r, int i = 1) {
        if (l <= a[i].l && a[i].r <= r) return a[i].s;
        push(i);
        ll ret = 0;
        int mid = (a[i].l + a[i].r) >> 1;
        if (l <= mid) ret += sum(l, r, ls);
        if (r > mid) ret += sum(l, r, rs);
        pull(i);
        return ret;
    }
};
```

2.8 Treap

```cpp
// 支援區間加值、區間反轉、區間 rotate 、區間刪除、插入元素、求區間
// 最小值的元素的 Treap。使用前建議 srand(time(0)); 除了 size()
// 方法以外，所有操作都是 O(log N) 。所有 public 方法各自獨立，請
// 斟酌要使用到哪些方法，有需要的才抄。
class Treap {
   private:
    struct Node {
        int pri = rand(), size = 1;
        ll val, mn, inc = 0;
        bool rev = 0;
        Node *lc = 0, *rc = 0;
        Node(ll v) { val = mn = v; }
    };
    Node* root = 0;
    void rev(Node* t) {
        if (!t) return;
        swap(t->lc, t->rc), t->rev ^= 1;
    }
    void update(Node* t, ll v) {
        if (!t) return;
        t->val += v, t->inc += v, t->mn += v;
    }
    void push(Node* t) {
        if (t->rev) rev(t->lc), rev(t->rc), t->rev = 0;
        update(t->lc, t->inc), update(t->rc, t->inc);
        t->inc = 0;
    }
    void pull(Node* t) {
        t->size = 1 + size(t->lc) + size(t->rc);
        t->mn = t->val;
        if (t->lc) t->mn = min(t->mn, t->lc->mn);
        if (t->rc) t->mn = min(t->mn, t->rc->mn);
    }
    // 看你要不要釋放記憶體
    void discard(Node* t) {
        if (!t) return;
        discard(t->lc), discard(t->rc);
        delete t;
    }
    void split(Node* t, Node*& a, Node*& b, int k) {
        if (!t) return a = b = 0, void();
        push(t);
        if (size(t->lc) < k) {
            a = t;
            split(t->rc, a->rc, b, k - size(t->lc) - 1);
            pull(a);
        } else {
            b = t;
            split(t->lc, a, b->lc, k);
            pull(b);
        }
    }
    Node* merge(Node* a, Node* b) {
        if (!a || !b) return a ? a : b;
        if (a->pri > b->pri) {
            push(a);
            a->rc = merge(a->rc, b);
            pull(a);
            return a;
        } else {
            push(b);
            b->lc = merge(a, b->lc);
            pull(b);
            return b;
        }
    }
    inline int size(Node* t) { return t ? t->size : 0; }
   public:
    int size() { return size(root); }
    void add(int l, int r, ll val) {
        Node *a, *b, *c, *d;
        split(root, a, b, r);
        split(a, c, d, l - 1);
        update(d, val);
        root = merge(merge(c, d), b);
    }
    // 反轉區間 [l, r]
    void reverse(int l, int r) {
        Node *a, *b, *c, *d;
        split(root, a, b, r);
        split(a, c, d, l - 1);
        swap(d->lc, d->rc);
        d->rev ^= 1;
        root = merge(merge(c, d), b);
    }
    // 區間 [l, r] 向右 rotate k 次， k < 0 表向左 rotate
    void rotate(int l, int r, int k) {
        int len = r - l + 1;
        Node *a, *b, *c, *d, *e, *f;
        split(root, a, b, r);
        split(a, c, d, l - 1);
        k = (k + len) % len;
        split(d, e, f, len - k);
        root = merge(merge(c, merge(f, e)), b);
    }
    // 插入一個元素 val 使其 index = i
    // 注意 i <= size
    void insert(int i, ll val) {
        if (i == size() + 1) {
            push_back(val);
            return;
        }
        assert(i <= size());
        Node *a, *b;
        split(root, a, b, i - 1);
        root = merge(merge(a, new Node(val)), b);
    }
    void push_back(ll val) {
        root = merge(root, new Node(val));
    }
    void remove(int l, int r) {
        int len = r - l + 1;
        Node *a, *b, *c, *d;
        split(root, a, b, l - 1);
        split(b, c, d, len);
        discard(c); // 看你要不要釋放記憶體
        root = merge(a, d);
    }
    ll minn(int l, int r) {
        Node *a, *b, *c, *d;
        split(root, a, b, r);
        split(a, c, d, l - 1);
        int ans = d->mn;
        root = merge(merge(c, d), b);
        return ans;
    }
};
```

2.9 link_cut_tree

```
struct splay_tree{
  int ch[2],pa;//子節點跟父母
  bool rev;//反轉的懶惰標記
  splay_tree():pa(0),rev(0){ch[0]=ch[1]=0;}
};
vector<splay_tree> nd;
//有的時候用vector會TLE，要注意
//這邊以node[0]作為null節點
bool isroot(int x){//判斷是否為這棵splay tree的根
  return nd[nd[x].pa].ch[0]!=x&&nd[nd[x].pa].ch[1]!=x;
}
void down(int x){//懶惰標記下推
  if(nd[x].rev){
    if(nd[x].ch[0])nd[nd[x].ch[0]].rev^=1;
    if(nd[x].ch[1])nd[nd[x].ch[1]].rev^=1;
    swap(nd[x].ch[0],nd[x].ch[1]);
    nd[x].rev=0;
  }
}
void push_down(int x){//所有祖先懶惰標記下推
  if(!isroot(x))push_down(nd[x].pa);
  down(x);
}
void up(int x){}//將子節點的資訊向上更新
void rotate(int x){//旋轉，會自行判斷轉的方向
  int y=nd[x].pa,z=nd[y].pa,d=(nd[y].ch[1]==x);
  nd[x].pa=z;
  if(!isroot(y))nd[z].ch[nd[z].ch[1]==y]=x;
  nd[y].ch[d]=nd[x].ch[d^1];
  nd[nd[y].ch[d]].pa=y;
  nd[y].pa=x,nd[x].ch[d^1]=y;
  up(y),up(x);
}
void splay(int x){//將x伸展到splay tree的根
  push_down(x);
  while(!isroot(x)){
    int y=nd[x].pa;
    if(!isroot(y)){
      int z=nd[y].pa;
      if((nd[z].ch[0]==y)^(nd[y].ch[0]==x))rotate(y);
      else rotate(x);
    }
    rotate(x);
  }
}
int access(int x){
  int last=0;
  while(x){
    splay(x);
    nd[x].ch[1]=last;
    up(x);
    last=x;
    x=nd[x].pa;
  }
  return last;//access後splay tree的根
}
void access(int x,bool is=0){//is=0就是一般的access
  int last=0;
  while(x){
    splay(x);
    if(is&&!nd[x].pa){
      //printf("%d\n",max(nd[last].ma,nd[nd[x].ch[1]].ma));
    }
    nd[x].ch[1]=last;
    up(x);
    last=x;
    x=nd[x].pa;
  }
}
void query_edge(int u,int v){
  access(u);
  access(v,1);
}
void make_root(int x){
  access(x),splay(x);
  nd[x].rev^=1;
}
void make_root(int x){
  nd[access(x)].rev^=1;
  splay(x);
}
void cut(int x,int y){
  make_root(x);
  access(y);
  splay(y);
  nd[y].ch[0]=0;
  nd[x].pa=0;
}
void cut_parents(int x){
  access(x);
  splay(x);
  nd[nd[x].ch[0]].pa=0;
  nd[x].ch[0]=0;
}
void link(int x,int y){
  make_root(x);
  nd[x].pa=y;
}
int find_root(int x){
  x=access(x);
  while(nd[x].ch[0])x=nd[x].ch[0];
  splay(x);
  return x;
}
int query(int u,int v){
//傳回uv路徑splay tree的根結點
//這種寫法無法求LCA
  make_root(u);
  return access(v);
}
int query_lca(int u,int v){
//假設求鏈上點權的總和，sum是子樹的權重和，data是節點的權重
  access(u);
  int lca=access(v);
  splay(u);
  if(u==lca){
    //return nd[lca].data+nd[nd[lca].ch[1]].sum
  }else{
    //return nd[lca].data+nd[nd[lca].ch[1]].sum+nd[u].sum
  }
}
struct EDGE{
  int a,b,w;
}e[10005];
int n;
vector<pair<int,int>> G[10005];
//first表示子節點，second表示邊的編號
int pa[10005],edge_node[10005];
//pa是父母節點，暫存用的，edge_node是每個編被存在哪個點裡面的陣列
void bfs(int root){
//在建構的時候把每個點都設成一個splay tree
  queue<int > q;
  for(int i=1;i<=n;++i)pa[i]=0;
  q.push(root);
  while(q.size()){
    int u=q.front();
    q.pop();
    for(auto P:G[u]){
      int v=P.first;
      if(v!=pa[u]){
        pa[v]=u;
        nd[v].pa=u;
        nd[v].data=e[P.second].w;
        edge_node[P.second]=v;
        up(v);
        q.push(v);
      }
    }
  }
}
void change(int x,int b){
  splay(x);
  //nd[x].data=b;
  up(x);
}
```

2.10 SparseTable

```
#define flg(a) floor(log2(a))
struct SparseTable {
    vector<vector<ll>> a;
    SparseTable(vector<ll>& data) {
        int n = data.size();
        a.assign(flg(n) + 1, vector<ll>(n));
        a[0] = data;
        for (int i = 1; (1 << i) <= n; i++)
            for (int j = 0, k = n - (1 << i); j <= k; j++)
                a[i][j] = max(a[i - 1][j],
                              a[i - 1][j + (1 << (i - 1))]);
    }
    ll maxx(int l, int r) { // [l, r], 0/1-based
        int k = flg(r - l + 1);
        return max(a[k][l], a[k][r - (1 << k) + 1]);
    }
};
```

2.11 FenwickTree

```
/** 普通 BIT ，為了加速打字只支援 1-based **/
const int maxn = ? ;  // 開全域加速打字
class BIT {
   private:
    ll a[maxn];
    ll sum(int i) {
        ll r = 0;
        while (i > 0) r += a[i], i -= i & -i;
        return r;
    }
   public:
    // size = maxn 的空 BIT ，所有元素都是零
    BIT() { memset(a, 0, sizeof(a)); }
    // 注意 1-based
    void add(int i, ll v) {
        while (i < maxn) a[i] += v, i += i & -i;
    }
    // 注意 1-based
    ll sum(int l, int r) { return sum(r) - sum(l - 1); }
};
/** 區間加值 BIT，只支援 1-based。複雜度 O(Q*log(N)) **/
const int maxn = ? ;  // 開全域加速打字
class RangeUpdateBIT {
   private:
    ll d[maxn], dd[maxn];
    ll sum(int i) {
        ll s = 0, ss = 0;
        int c = i + 1;  // 這行不是打錯！要加！
        while (i > 0) s += d[i], ss += dd[i], i -= i & -i;
        return c * s - ss;
    }
    void add(int i, ll v) {
        int c = i;
        while (i < maxn)
            d[i] += v, dd[i] += c * v, i += i & -i;
    }
   public:
    // 空 BIT，size = maxn，所有元素都是零，注意 1-based
    RangeUpdateBIT() {
        memset(d, 0, sizeof(d));
        memset(dd, 0, sizeof(dd));
    }
    // 必區間區間求和，注意 1-based
    ll sum(int l, int r) { return sum(r) - sum(l - 1); }
    // 必區間區間加值，注意 1-based
    void add(int l, int r, ll v) {
        add(l, v), add(r + 1, -v);
    }
};
```

3 Flow_Matching

3.1 KM

```
/*
時間複雜度 O(N^3)
求完美匹配中的最大權匹配
如果不存在完美匹配，求最大匹配
如果存在數個最大匹配，求數個最大匹配當中最大權匹配
*/
const ll INF = 5e18;
const int N = ?;  // max n
int n;            // count of vertex (one side)
ll g[N][N];       // weights
class KM {
   private:
    ll lx[N], ly[N], s[N];
    int px[N], py[N], m[N], p[N];

    void adj(int y) {  // 把增廣路上所有邊反轉
        m[y] = py[y];
        if (px[m[y]] != -2)
            adj(px[m[y]]);
    }
    bool dfs(int x) {  // DFS找增廣路

        for (int y = 0; y < n; ++y) {
            if (py[y] != -1) continue;
            ll t = lx[x] + ly[y] - g[x][y];
            if (t == 0) {
                py[y] = x;
                if (m[y] == -1) {
                    adj(y);
                    return 1;
                }
                if (px[m[y]] != -1) continue;
                px[m[y]] = y;
                if (dfs(m[y])) return 1;
            } else if (s[y] > t) {
                s[y] = t;
                p[y] = x;
            }
        }
        return 0;
    }

   public:
    ll max_weight() {
        memset(ly, 0, sizeof(ly));
        memset(m, -1, sizeof(m));
        for (int x = 0; x < n; ++x) {
```

```cpp
            lx[x] = -INF;
            for (int y = 0; y < n; ++y) {
                lx[x] = max(lx[x], g[x][y]);
            }
        }
        for (int x = 0; x < n; ++x) {
            for (int y = 0; y < n; ++y) s[y] = INF;
            memset(px, -1, sizeof(px));
            memset(py, -1, sizeof(py));
            px[x] = -2;
            if (dfs(x)) continue;
            bool flag = 1;
            while (flag) {
                ll cut = INF;
                for (int y = 0; y < n; ++y)
                    if (py[y] == -1 && cut > s[y]) cut = s[y];
                for (int j = 0; j < n; ++j) {
                    if (px[j] != -1) lx[j] -= cut;
                    if (py[j] != -1)
                        ly[j] += cut;
                    else
                        s[j] -= cut;
                }
                for (int y = 0; y < n; ++y) {
                    if (py[y] == -1 && s[y] == 0) {
                        py[y] = p[y];
                        if (m[y] == -1) {
                            adj(y);
                            flag = 0;
                            break;
                        }
                        px[m[y]] = y;
                        if (dfs(m[y])) {
                            flag = 0;
                            break;
                        }
                    }
                }
            }
        }
        ll ans = 0;
        for (int y = 0; y < n; ++y)
            if (g[m[y]][y] != -INF) ans += g[m[y]][y];
        return ans;
    }
};
```

3.2 Min_Cost_Max_Flow

```cpp
/** Min cost max flow 。0/1-based 都安全。 **/
class MCMF {
   private:
    struct edge { int to, r; ll rest, c; };
    int n; ll f = 0, c = 0;
    vector<vector<edge>> g;
    vector<int> pre, prel;
    bool run(int s, int t) {
        vector<ll> dis(n, inf); vector<bool> vis(n);
        dis[s] = 0; queue<int> q; q.push(s);
        while (q.size()) {
            int u = q.front(); q.pop(); vis[u] = 0;
            for (int i = 0; i < g[u].size(); i++) {
                int v = g[u][i].to; ll w = g[u][i].c;
                if (g[u][i].rest <= 0 ||
                    dis[v] <= dis[u] + w)
                    continue;
                pre[v] = u, prel[v] = i;
                dis[v] = dis[u] + w;
                if (!vis[v]) vis[v] = 1, q.push(v);
            }
        }
        if (dis[t] == inf) return 0;
        ll tf = inf;
        for (int v = t, u, l; v != s; v = u) {
            u = pre[v], l = prel[v];
            tf = min(tf, g[u][l].rest);
        }
        for (int v = t, u, l; v != s; v = u) {
            u = pre[v], l = prel[v], g[u][l].rest -= tf;
            g[v][g[u][l].r].rest += tf;
        }
        c += tf * dis[t], f += tf;
        return 1;
    }
   public:
    // 建立空圖，n 是節點數量 (包含 source 和 sink)
    MCMF(int n)
        : n(n + 1), g(n + 1), pre(n + 1), prel(n + 1) {}
    // 加有向邊 u->v ，cap 容量 cost 成本
    void add_edge(int u, int v, ll cap, ll cost) {
        g[u].push_back({v, (int)g[v].size(), cap, cost});
        g[v].push_back({u, (int)g[u].size() - 1, 0, -cost});
    }
    pair<ll, ll> query(int src, int sink) {
        while (run(src, sink));
        return {f, c};  //{min cost, max flow}
    }
};
```

3.3 Ford_Fulkerson

```cpp
const int maxn = 1e5 + 10, INF = 1e9;
const long long INF64 = 1e18;
struct edge{ int to, cap, rev; };
vector<edge> G[maxn];
int n, m, s, t, a, b, c;
bool vis[maxn];
int dfs(int v, int t, int f) {
    cout << v << ' ' << t << ' ' << f << '\n';
    if (v == t) return f;
    vis[v] = true;
    for (edge &e: G[v]) {
        if (!vis[e.to] && e.cap > 0) {
            int d = dfs(e.to, t, min(f, e.cap));
            if (d > 0) {
                e.cap -= d, G[e.to][e.rev].cap += d;
                return d;
            }
        }
    }
    return 0;
}
int ford_fulkerson(int s, int t) {
    int flow = 0, f;
    for (int i = 0; i < n; i++) {
        cout << i << " : ";
        for (edge e: G[i])
            cout << '(' << e.to << ',' << e.cap << ')' << ' ';
        cout << '\n';
    }
    do {
        memset(vis, false, sizeof(vis));
        f = dfs(s, t, INF);
        for (int i = 0; i < n; i++) {
            cout << i << " : ";
            for (edge e: G[i])
                cout << '(' << e.to << ',' << e.cap << ')' << ' ';
            cout << '\n';
        }
        cout << f << '\n';
        flow += f;
    } while (f > 0);
    return flow;
}
void init(int n) {
    for (int i = 0; i < n; i++) G[i].clear();
}
int main() {
    cin >> n >> m >> s >> t;
    init(n);
    while (m--) {
        cin >> a >> b >> c;
        G[a].push_back((edge){b, c, (int)G[b].size()});
        G[b].push_back((edge){a, 0, (int)G[a].size() - 1});
    }
    cout << ford_fulkerson(s, t) << '\n';
    return 0;
}
```

3.4 Hungarian

```cpp
/*
時間複雜度 O(VE)
*/
const int INF = 2e9;
const int N = ? ;        // 男女總人數；女性 index 0 ~ p ，男性 index p+1 ~ N-1
int vis[N], rnd, m[N];  // 跑完匈牙利之後配對結果儲存於此，-1 表示人醜
vector<int> g[N];       // 關係表

int dfs(int s) {
    for (int x : g[s]) {
        if (vis[x]) continue;
        vis[x] = 1;
        if (m[x] == -1 || dfs(m[x])) {
            m[x] = s, m[s] = x;
            return 1;
        }
    }
    return 0;
}

// 回傳成功結婚對數
```

```
int hungarian(int p) {  // 傳入女性人數
    memset(m, -1, sizeof(m));
    int c = 0;
    for (int i = 0; i < p; i++) {
        if (m[i] == -1) {
            memset(vis, 0, sizeof(vis));
            c += dfs(i);
        }
    }
    return c;
}
```

3.5 Hopcroft_Karp

```
/*
等價於匈牙利算法，只是匈牙利算法的優化，都是做二分圖最大匹
    配。
時間複雜度為 O(E√V)
*/

int n, m, vis[maxn], level[maxn], pr[maxn], pr2[maxn];
vector<int> edge[maxn];  // for Left
bool dfs(int u) {
    vis[u] = true;
    for (vector<int>::iterator it = edge[u].begin();
         it != edge[u].end(); ++it) {
        int v = pr2[*it];
        if (v == -1 ||
            (!vis[v] && level[u] < level[v] && dfs(v))) {
            pr[u] = *it, pr2[*it] = u;
            return true;
        }
    }
    return false;
}
int hopcroftKarp() {
    memset(pr, -1, sizeof(pr));
    memset(pr2, -1, sizeof(pr2));
    for (int match = 0;;) {
        queue<int> Q;
        for (int i = 1; i <= n; ++i) {
            if (pr[i] == -1) {
                level[i] = 0;
                Q.push(i);
            } else
                level[i] = -1;
        }
        while (!Q.empty()) {
            int u = Q.front();
            Q.pop();
            for (vector<int>::iterator it = edge[u].begin();
                 it != edge[u].end(); ++it) {
                int v = pr2[*it];
                if (v != -1 && level[v] < 0) {
                    level[v] = level[u] + 1;
                    Q.push(v);
                }
            }
        }
        for (int i = 1; i <= n; ++i) vis[i] = false;
        int d = 0;
        for (int i = 1; i <= n; ++i)
            if (pr[i] == -1 && dfs(i)) ++d;
        if (d == 0) return match;
        match += d;
    }
}
```

3.6 SW_MinCut

```
// all pair min cut
// global min cut
struct SW { // O(V^3)
    static const int MXN = 514;
    int n, vst[MXN], del[MXN];
    int edge[MXN][MXN], wei[MXN];
    void init(int _n){
        n = _n; FZ(edge); FZ(del);
    }
    void addEdge(int u, int v, int w) {
        edge[u][v] += w; edge[v][u] += w;
    }
    void search(int &s, int &t) {
        FZ(vst); FZ(wei);
        s = t = -1;
        while (true){
            int mx=-1, cur=0;
            for (int i=0; i<n; i++)
                if (!del[i] && !vst[i] && mx<wei[i])
                    cur = i, mx = wei[i];
            if (mx == -1) break;
            vst[cur] = 1;
            s = t; t = cur;
            for (int i=0; i<n; i++)
                if (!vst[i] && !del[i]) wei[i] += edge[cur][i
                    ];
        }
    }
    int solve() {
        int res = 2147483647;
        for (int i=0, x, y; i<n-1; i++) {
            search(x,y);
            res = min(res,wei[y]);
            del[y] = 1;
            for (int j=0; j<n; j++)
                edge[x][j] = (edge[j][x] += edge[y][j]);
        }
        return res;
    }
} graph;
```

3.7 Dinic

```
/*
一般來說複雜度遠低於 O(EV^2)
特殊情況：
  Bipartite 約 O(E * sqrt(v))
  Unit network 約 O(E * min(V√V,√E))
0/1-based 都安全。
*/
class Dinic {
    struct edge {
        int d, r; ll c;
        edge(int d, ll c, int r) : d(d), c(c), r(r){};
    };
  private:
    vector<vector<edge>> adj; vector<int> lv, ve; int n;
    bool mklv(int s, int d) {
        lv.assign(n, -1); lv[s] = 0;
        queue<int> q; q.push(s);
        while (!q.empty()) {
            int v = q.front(); q.pop();
            for (auto& e : adj[v]) {
                if (e.c == 0 || lv[e.d] != -1) continue;
                lv[e.d] = lv[v] + 1, q.push(e.d);
            }
        }
        return lv[d] > 0;
    }
    ll aug(int v, ll f, int d) {
        if (v == d) return f;
        for (; ve[v] < adj[v].size(); ve[v]++) {
            auto& e = adj[v][ve[v]];
            if (lv[e.d] != lv[v] + 1 || !e.c) continue;
            ll sent = aug(e.d, min(f, e.c), d);
            if (sent > 0) {
                e.c -= sent, adj[e.d][e.r].c += sent;
                return sent;
            }
        }
        return 0;
    }
  public:
    // 建立空圖， n 是節點 (包含 source, sink) 數量
    Dinic(int n) : n(n + 1) { clear(); }
    // 清空整個圖，這需要重複使用 dinic 時 (如二分搜) 很方便
    void clear() { adj.assign(n, vector<edge>()); }
    // 加有向邊 src->dst ，cap 是容量
    void add_edge(int src, int dst, ll cap) {
        edge ss(dst, cap, adj[dst].size());
        edge dd(src, 0, adj[src].size());
        adj[src].push_back(ss), adj[dst].push_back(dd);
    }
    ll max_flow(int s, int d) {
        ll ret = 0;
        while (mklv(s, d)) {
            ve.assign(n, 0);
            while (ll f = aug(s, 9e18, d)) ret += f;
        }
        return ret;
    }
};
```

4 Geometry

4.1 Geometry

```
//Copy from Jinkela
const double PI=atan2(0.0,-1.0);
template<typename T>
struct point{
```

```cpp
  T x,y;
  point(){}
  point(const T&x,const T&y):x(x),y(y){}
  point operator+(const point &b)const{
    return point(x+b.x,y+b.y); }
  point operator-(const point &b)const{
    return point(x-b.x,y-b.y); }
  point operator*(const T &b)const{
    return point(x*b,y*b); }
  point operator/(const T &b)const{
    return point(x/b,y/b); }
  bool operator==(const point &b)const{
    return x==b.x&&y==b.y; }
  T dot(const point &b)const{
    return x*b.x+y*b.y; }
  T cross(const point &b)const{
    return x*b.y-y*b.x; }
  point normal()const{//求法向量
    return point(-y,x); }
  T abs2()const{//向量長度的平方
    return dot(*this); }
  T rad(const point &b)const{//兩向量的弧度
return fabs(atan2(fabs(cross(b)),dot(b))); }
  T getA()const{//對x軸的弧度
    T A=atan2(y,x);//超過180度會變負的
    if(A<=-PI/2)A+=PI*2;
    return A;
  }
};
template<typename T>
struct line{
  line(){}
  point<T> p1,p2;
  T a,b,c;//ax+by+c=0
  line(const point<T>&x,const point<T>&y):p1(x),p2(y){}
  void pton(){//轉成一般式
    a=p1.y-p2.y;
    b=p2.x-p1.x;
    c=-a*p1.x-b*p1.y;
  }
  T ori(const point<T> &p)const{//點和有向直線的關係，>0左邊、=0在線上<0右邊
    return (p2-p1).cross(p-p1);
  }
  T btw(const point<T> &p)const{//點投影落在線段上<=0
    return (p1-p).dot(p2-p);
  }
  bool point_on_segment(const point<T>&p)const{//點是否在線段上
    return ori(p)==0&&btw(p)<=0;
  }
  T dis2(const point<T> &p,bool is_segment=0)const{//點跟直線/線段的距離平方
    point<T> v=p2-p1,v1=p-p1;
    if(is_segment){
      point<T> v2=p-p2;
      if(v.dot(v1)<=0)return v1.abs2();
      if(v.dot(v2)>=0)return v2.abs2();
    }
    T tmp=v.cross(v1);
    return tmp*tmp/v.abs2();
  }
  T seg_dis2(const line<T> &l)const{//兩線段距離平方
    return min({dis2(l.p1,1),dis2(l.p2,1),l.dis2(p1,1),l.dis2(p2,1)});
  }
  point<T> projection(const point<T> &p)const{//點對直線的投影
    point<T> n=(p2-p1).normal();
    return p-n*(p-p1).dot(n)/n.abs2();
  }
  point<T> mirror(const point<T> &p)const{
    //點對直線的鏡射，要先呼叫pton轉成一般式
    point<T> R;
    T d=a*a+b*b;
    R.x=(b*b*p.x-a*a*p.x-2*a*b*p.y-2*a*c)/d;
    R.y=(a*a*p.y-b*b*p.y-2*a*b*p.x-2*b*c)/d;
    return R;
  }
  bool equal(const line &l)const{//直線相等
    return ori(l.p1)==0&&ori(l.p2)==0;
  }
  bool parallel(const line &l)const{
    return (p1-p2).cross(l.p1-l.p2)==0;
  }
  bool cross_seg(const line &l)const{
    return (p2-p1).cross(l.p1-p1)*(p2-p1).cross(l.p2-p1)<=0;//直線是否交線段
  }
  int line_intersect(const line &l)const{//直線相交情況，-1無限多點、1交於一點、0不相交
    return parallel(l)?(ori(l.p1)==0?-1:0):1;
  }
  int seg_intersect(const line &l)const{
    T c1=ori(l.p1), c2=ori(l.p2);
    T c3=l.ori(p1), c4=l.ori(p2);
    if(c1==0&&c2==0){//共線
      bool b1=btw(l.p1)>=0,b2=btw(l.p2)>=0;
      T a3=l.btw(p1),a4=l.btw(p2);
      if(b1&&b2&&a3==0&&a4>=0) return 2;
      if(b1&&b2&&a3>=0&&a4==0) return 3;
      if(b1&&b2&&a3>=0&&a4>=0) return 0;
      return -1;//無限交點
    }else if(c1*c2<=0&&c3*c4<=0)return 1;
    return 0;//不相交
  }
  point<T> line_intersection(const line &l)const{/*直線交點*/
    point<T> a=p2-p1,b=l.p2-l.p1,s=l.p1-p1;
    //if(a.cross(b)==0)return INF;
    return p1+a*(s.cross(b)/a.cross(b));
  }
  point<T> seg_intersection(const line &l)const{//線段交點
    int res=seg_intersect(l);
    if(res<=0) assert(0);
    if(res==2) return p1;
    if(res==3) return p2;
    return line_intersection(l);
  }
};
template<typename T>
struct polygon{
  polygon(){}
  vector<point<T> > p;//逆時針順序
  T area()const{//面積
    T ans=0;
    for(int i=p.size()-1,j=0;j<(int)p.size();i=j++)
      ans+=p[i].cross(p[j]);
    return ans/2;
  }
  point<T> center_of_mass()const{//重心
    T cx=0,cy=0,w=0;
    for(int i=p.size()-1,j=0;j<(int)p.size();i=j++){
      T a=p[i].cross(p[j]);
      cx+=(p[i].x+p[j].x)*a;
      cy+=(p[i].y+p[j].y)*a;
      w+=a;
    }
    return point<T>(cx/3/w,cy/3/w);
  }
  char ahas(const point<T>& t)const{//點是否在簡單多邊形內，是的話回傳1、在邊上回傳-1、否則回傳0
    bool c=0;
    for(int i=0,j=p.size()-1;i<p.size();j=i++)
      if(line<T>(p[i],p[j]).point_on_segment(t))return -1;
      else if((p[i].y>t.y)!=(p[j].y>t.y)&&
      t.x<(p[j].x-p[i].x)*(t.y-p[i].y)/(p[j].y-p[i].y)+p[i].x)
        c=!c;
    return c;
  }
  char point_in_convex(const point<T>&x)const{
    int l=1,r=(int)p.size()-2;
    while(l<=r){//點是否在凸多邊形內，是的話回傳1、在邊上回傳-1、否則回傳0
      int mid=(l+r)/2;
      T a1=(p[mid]-p[0]).cross(x-p[0]);
      T a2=(p[mid+1]-p[0]).cross(x-p[0]);
      if(a1>=0&&a2<=0){
        T res=(p[mid+1]-p[mid]).cross(x-p[mid]);
        return res>0?1:(res>=0?-1:0);
      }else if(a1<0)r=mid-1;
      else l=mid+1;
    }
    return 0;
  }
  vector<T> getA()const{//凸包邊對x軸的夾角
    vector<T>res;//一定是遞增的
    for(size_t i=0;i<p.size();++i)
      res.push_back((p[(i+1)%p.size()]-p[i]).getA());
    return res;
  }
  bool line_intersect(const vector<T>&A,const line<T> &l)const{//O(logN)
    int f1=upper_bound(A.begin(),A.end(),(l.p1-l.p2).getA())-A.begin();
    int f2=upper_bound(A.begin(),A.end(),(l.p2-l.p1).getA())-A.begin();
    return l.cross_seg(line<T>(p[f1],p[f2]));
  }
  polygon cut(const line<T> &l)const{//凸包對直線切割，得到直線l左側的凸包
    polygon ans;
    for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
      if(l.ori(p[i])>=0){
        ans.p.push_back(p[i]);
        if(l.ori(p[j])<0)
          ans.p.push_back(l.line_intersection(line<T>(p[i],p[j])));
      }else if(l.ori(p[j])>0)
      ans.p.push_back(l.line_intersection(line<T>(p[i],p[j])));
    }
    return ans;
  }
  static bool graham_cmp(const point<T>& a,const point<T>& b){//凸包排序函數
    return (a.x<b.x)||(a.x==b.x&&a.y<b.y);
  }
  void graham(vector<point<T> > &s){//凸包
    sort(s.begin(),s.end(),graham_cmp);
    p.resize(s.size()+1);
    int m=0;
    for(size_t i=0;i<s.size();++i){
      while(m>=2&&(p[m-1]-p[m-2]).cross(s[i]-p[m-2])<=0)--m;
      p[m++]=s[i];
    }
    for(int i=s.size()-2,t=m+1;i>=0;--i){
      while(m>=t&&(p[m-1]-p[m-2]).cross(s[i]-p[m-2])<=0)--m;
      p[m++]=s[i];
    }
    if(s.size()>1)--m;
    p.resize(m);
  }
  T diam(){//直徑
    int n=p.size(),t=1;
    T ans=0;p.push_back(p[0]);
    for(int i=0;i<n;i++){
      point<T> now=p[i+1]-p[i];
      while(now.cross(p[t+1]-p[i])>now.cross(p[t]-p[i]))t=(t+1)%n;
      ans=max(ans,(p[i]-p[t]).abs2());
    }
    return p.pop_back(),ans;
  }
  T min_cover_rectangle(){//最小覆蓋矩形
    int n=p.size(),t=1,r=1,l;
    if(n<3)return 0;//也可以做最小周長矩形
    T ans=1e99;p.push_back(p[0]);
    for(int i=0;i<n;i++){
      point<T> now=p[i+1]-p[i];
      while(now.cross(p[t+1]-p[i])>now.cross(p[t]-p[i]))t=(t+1)%n;
      while(now.dot(p[r+1]-p[i])>now.dot(p[r]-p[i]))r=(r+1)%n;
      if(!i)l=r;
      while(now.dot(p[l+1]-p[i])<=now.dot(p[l]-p[i]))l=(l+1)%n;
      T d=now.abs2();
      T tmp=now.cross(p[t]-p[i])*(now.dot(p[r]-p[i])-now.dot(p[l]-p[i]))/d;
      ans=min(ans,tmp);
    }
    return p.pop_back(),ans;
  }
  T dis2(polygon &pl){//凸包最近距離平方
    vector<point<T> > &P=p,&Q=pl.p;
    int n=P.size(),m=Q.size(),l=0,r=0;
  for(int i=0;i<n;++i)if(P[i].y<P[l].y)l=i;
  for(int i=0;i<m;++i)if(Q[i].y<Q[r].y)r=i;
    P.push_back(P[0]),Q.push_back(Q[0]);
    T ans=1e99;
    for(int i=0;i<n;++i){
      while((P[l]-P[l+1]).cross(Q[r+1]-Q[r])<0)r=(r+1)%m;
      ans=min(ans,line<T>(P[l],P[l+1]).seg_dis2(line<T>(Q[r],Q[r+1])));
      l=(l+1)%n;
    }
    return P.pop_back(),Q.pop_back(),ans;
  }
  static char sign(const point<T>&t){
    return (t.y==0?t.x:t.y)<0;
  }
  static bool angle_cmp(const line<T>& A,const line<T>& B){
    point<T> a=A.p2-A.p1,b=B.p2-B.p1;
    return sign(a)<sign(b)||(sign(a)==sign(b)&&a.cross(b)>0);
  }
  int halfplane_intersection(vector<line<T> > &s){//半平面交
    sort(s.begin(),s.end(),angle_cmp);//線段左側為該線段半平面
    int L,R,n=s.size();
    vector<point<T> > px(n);
    vector<line<T> > q(n);
    q[L=R=0]=s[0];
    for(int i=1;i<n;++i){
      while(L<R&&s[i].ori(px[R-1])<=0)--R;
      while(L<R&&s[i].ori(px[L])<=0)++L;
      q[++R]=s[i];
      if(q[R].parallel(q[R-1])){
        --R;
        if(q[R].ori(s[i].p1)>0)q[R]=s[i];
      }
      if(L<R)px[R-1]=q[R-1].line_intersection(q[R]);
    }
    while(L<R&&q[L].ori(px[R-1])<=0)--R;
    p.clear();
    if(R-L<=1)return 0;
    px[R]=q[R].line_intersection(q[L]);
    for(int i=L;i<=R;++i)p.push_back(px[i]);
    return R-L+1;
  }
};
template<typename T>
struct triangle{
  point<T> a,b,c;
  triangle(){}
  triangle(const point<T> &a,const point<T> &b,const point<T> &c):a(a),b(b),c(c){}
  T area()const{
    T t=(b-a).cross(c-a)/2;
    return t>0?t:-t;
  }
  point<T> barycenter()const{//重心
    return (a+b+c)/3;
  }
  point<T> circumcenter()const{//外心
    static line<T> u,v;
    u.p1=(a+b)/2;
    u.p2=point<T>(u.p1.x-a.y+b.y,u.p1.y+a.x-b.x);
    v.p1=(a+c)/2;
    v.p2=point<T>(v.p1.x-a.y+c.y,v.p1.y+a.x-c.x);
    return u.line_intersection(v);
  }
  point<T> incenter()const{//內心
    T A=sqrt((b-c).abs2()),B=sqrt((a-c).abs2()),C=sqrt((a-b).abs2());
    return point<T>(A*a.x+B*b.x+C*c.x,A*a.y+B*b.y+C*c.y)/(A+B+C);
  }
  point<T> perpencenter()const{//垂心
    return barycenter()*3-circumcenter()*2;
  }
};
template<typename T>
struct point3D{
  T x,y,z;
  point3D(){}
  point3D(const T&x,const T&y,const T&z):x(x),y(y),z(z){}
  point3D operator+(const point3D &b)const{
    return point3D(x+b.x,y+b.y,z+b.z);}
  point3D operator-(const point3D &b)const{
    return point3D(x-b.x,y-b.y,z-b.z);}
  point3D operator*(const T &b)const{
    return point3D(x*b,y*b,z*b);}
  point3D operator/(const T &b)const{
    return point3D(x/b,y/b,z/b);}
  bool operator==(const point3D &b)const{
    return x==b.x&&y==b.y&&z==b.z;}
  T dot(const point3D &b)const{
    return x*b.x+y*b.y+z*b.z;}
  point3D cross(const point3D &b)const{
    return point3D(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);}
  T abs2()const{//向量長度的平方
    return dot(*this);}
  T area2(const point3D &b)const{//和b、原點圍成面積的平方
    return cross(b).abs2()/4;}
};
template<typename T>
struct line3D{
  point3D<T> p1,p2;
  line3D(){}
  line3D(const point3D<T> &p1,const point3D<T> &p2):p1(p1),p2(p2){}
  T dis2(const point3D<T> &p,bool is_segment=0)const{//點跟直線/線段的距離平方
    point3D<T> v=p2-p1,v1=p-p1;
    if(is_segment){
      point3D<T> v2=p-p2;
      if(v.dot(v1)<=0)return v1.abs2();
      if(v.dot(v2)>=0)return v2.abs2();
    }
    point3D<T> tmp=v.cross(v1);
    return tmp.abs2()/v.abs2();
  }
  pair<point3D<T>,point3D<T> > closest_pair(const line3D<T> &l)const{
    point3D<T> v1=(p1-p2),v2=(l.p1-l.p2);
    point3D<T> N=v1.cross(v2),ab(p1-l.p1);
    //if(N.abs2()==0)return NULL;平行或重合
    T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//最近點對距離
    point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.cross(d2),G=l.p1-p1;
    T t1=(G.cross(d2)).dot(D)/D.abs2();
    T t2=(G.cross(d1)).dot(D)/D.abs2();
    return make_pair(p1+d1*t1,l.p1+d2*t2);
  }
  bool same_side(const point3D<T> &a,const point3D<T> &b)const{
    return (p2-p1).cross(a-p1).dot((p2-p1).cross(b-p1))>0;
  }
};
template<typename T>
struct plane{
  point3D<T> p0,n;//平面上的點和法向量
  plane(){}
  plane(const point3D<T> &p0,const point3D<T> &n):p0(p0),n(n){}
  T dis2(const point3D<T> &p)const{//點到平面距離的平方
    T tmp=(p-p0).dot(n);
    return tmp*tmp/n.abs2();
  }
  point3D<T> projection(const point3D<T> &p)const{
    return p-n*(p-p0).dot(n)/n.abs2();
  }
  point3D<T> line_intersection(const line3D<T> &l)const{
    T tmp=n.dot(l.p2-l.p1);//等於0表示平行或重合該平面
    return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/tmp);
  }
  line3D<T> plane_intersection(const plane &pl)const{
    point3D<T> e=n.cross(pl.n),v=n.cross(e);
    T tmp=pl.n.dot(v);//等於0表示平行或重合該平面
    point3D<T> q=p0+(v*(pl.n.dot(pl.p0-p0))/tmp);
    return line3D<T>(q,q+e);
  }
};
template<typename T>
struct triangle3D{
  point3D<T> a,b,c;
  triangle3D(){}
  triangle3D(const point3D<T> &a,const point3D<T> &b,const point3D<T> &c):a(a),b(b),c(c){}
  bool point_in(const point3D<T> &p)const{//點在該平面上的投影在三角形中
    return line3D<T>(b,c).same_side(p,a)&&line3D<T>(a,c).same_side(p,b)&&line3D<T>(a,b).same_side(p,c);
  }
};
template<typename T>
struct tetrahedron{//四面體
  point3D<T> a,b,c,d;
  tetrahedron(){}
  tetrahedron(const point3D<T> &a,const point3D<T> &b,const point3D<T> &c,const point3D<T> &d):a(a),b(b),c(c),d(d){}
  T volume6()const{//體積的六倍
    return (d-a).dot((b-a).cross(c-a));
  }
  point3D<T> centroid()const{
    return (a+b+c+d)/4;
  }
  bool point_in(const point3D<T> &p)const{
    return triangle3D<T>(a,b,c).point_in(p)&&triangle3D<T>(c,d,a).point_in(p);
  }
};
template<typename T>
struct convexhull3D{
  static const int MAXN=1005;
  struct face{
    int a,b,c;
    face(int a,int b,int c):a(a),b(b),c(c){}
  };
  vector<point3D<T>> pt;
  vector<face> ans;
  int fid[MAXN][MAXN];
  void build(){
    int n=pt.size();
    ans.clear();
    memset(fid,0,sizeof(fid));
    ans.emplace_back(0,1,2);//注意不能共線
    ans.emplace_back(2,1,0);
    int ftop = 0;
    for(int i=3, ftop=1; i<n; ++i,++ftop){
      vector<face> next;
      for(auto &f:ans){
        T d=(pt[i]-pt[f.a]).dot((pt[f.b]-pt[f.a]).cross(pt[f.c]-pt[f.a]));
        if(d<=0) next.push_back(f);
        int ff=0;
        if(d>0) ff=ftop;
        else if(d<0) ff=-ftop;
        fid[f.a][f.b]=fid[f.b][f.c]=fid[f.c][f.a]=ff;
      }
      for(auto &f:ans){
        if(fid[f.a][f.b]>0 && fid[f.a][f.b]!=fid[f.b][f.a])
          next.emplace_back(f.a,f.b,i);
        if(fid[f.b][f.c]>0 && fid[f.b][f.c]!=fid[f.c][f.b])
          next.emplace_back(f.b,f.c,i);
        if(fid[f.c][f.a]>0 && fid[f.c][f.a]!=fid[f.a][f.c])
          next.emplace_back(f.c,f.a,i);
      }
      ans=next;
    }
  }
  point3D<T> centroid()const{
    point3D<T> res(0,0,0);
    T vol=0;
    for(auto &f:ans){
      T tmp=pt[f.a].dot(pt[f.b].cross(pt[f.c]));
      res=res+(pt[f.a]+pt[f.b]+pt[f.c])*tmp;
      vol+=tmp;
    }
    return res/(vol*4);
  }
};
```

4.2 SmallestCircle

```
using PT = point<T>;
using CPT = const PT;
PT circumcenter(CPT &a, CPT &b, CPT &c) {
  PT u = b-a, v =c-a;
  T c1 = u.abs2()/2, c2 = v.abs2()/2;
  T d = u.cross(v);
  return PT(a.x+(v.y*c1-u.y*c2)/d, a.y+(u.x*c2-v.x*c1)/d);
}
void solve(PT p[], int n, PT &c, T &r2){
  random_shuffle(p,p+n);
  c = p[0]; r2 = 0; // c,r2 = 圓心,半徑平方
  for(int i=1; i<n; i++)
    if( (p[i]-c).abs2() > r2) {
      c=p[i]; r2=0;
      for(int j=0; j<i; j++)
        if( (p[j]-c).abs2() > r2) {
          c.x = (p[i].x+p[j].x)/2;
          c.y = (p[i].y+p[j].y)/2;
          r2 = (p[j]-c).abs2();
          for(int k=0; k<j; k++)
            if( (p[k]-c).abs2() > r2) {
              c = circumcenter(p[i], p[j], p[k]);
              r2 = (p[i]-c).abs2();
            }
        }
    }
}
```

4.3 Rectangle_Union_Area

```
const int maxn = 1e5 + 10;
struct rec{
    int t, b, l, r;
} r[maxn];
int n, cnt[maxn << 2];
long long st[maxn << 2], ans = 0;
vector<int> x, y;
vector<pair<pair<int, int>, pair<int, int>>> v;
void modify(int t, int l, int r, int ql, int qr, int v) {
    if (ql <= l && r <= qr) cnt[t] += v;
    else {
        int m = (l + r) >> 1;
        if (qr <= m) modify(t << 1, l, m, ql, qr, v);
        else if (ql >= m) modify(t << 1 | 1, m, r, ql, qr, v);
        else modify(t << 1, l, m, ql, m, v), modify(t << 1 | 1, m, r, m, qr, v);
    }
    if (cnt[t]) st[t] = y[r] - y[l];
    else if (r - l == 1) st[t] = 0;
    else st[t] = st[t << 1] + st[t << 1 | 1];
}
int main() {
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> r[i].l >> r[i].r >> r[i].b >> r[i].t;
        if (r[i].l > r[i].r) swap(r[i].l, r[i].r);
        if (r[i].b > r[i].t) swap(r[i].b, r[i].t);
        x.push_back(r[i].l);
        x.push_back(r[i].r);
        y.push_back(r[i].b);
        y.push_back(r[i].t);
    }
    sort(x.begin(), x.end());
    sort(y.begin(), y.end());
    x.erase(unique(x.begin(), x.end()), x.end());
    y.erase(unique(y.begin(), y.end()), y.end());
    for (int i = 0; i < n; i++) {
        r[i].l = lower_bound(x.begin(), x.end(), r[i].l) - x.begin();
        r[i].r = lower_bound(x.begin(), x.end(), r[i].r) - x.begin();
        r[i].b = lower_bound(y.begin(), y.end(), r[i].b) - y.begin();
        r[i].t = lower_bound(y.begin(), y.end(), r[i].t) - y.begin();
        v.emplace_back(make_pair(r[i].l, 1), make_pair(r[i].b, r[i].t));
        v.emplace_back(make_pair(r[i].r, -1), make_pair(r[i].b, r[i].t));
    }
    sort(v.begin(), v.end(), [](pair<pair<int, int>, pair<int, int>> a, pair<pair<int, int>, pair<int, int>> b){
        if (a.first.first != b.first.first) return a.first.first < b.first.first;
```

```cpp
            return a.first.second > b.first.second;
    });
    for (int i = 0; i < v.size(); i++) {
        if (i) ans += (x[v[i].first.first] - x[v[i - 1].first
             .first]) * st[1];
        modify(1, 0, y.size(), v[i].second.first, v[i].second
             .second, v[i].first.second);
    }
    cout << ans << '\n';
    return 0;
}
```

4.4 旋轉卡尺

```cpp
typedef pair<ll, ll> pii;
#define x first
#define y second
#define ii (i + 1) % n  // 打字加速！
inline pii operator-(const pii& a, const pii& b) {
    return {a.x - b.x, a.y - b.y};
}  // const 不可省略
inline ll operator*(const pii& a, const pii& b) {
    return a.x * b.y - a.y * b.x;
}
inline ll crzf(const pii& o, const pii& a, const pii& b) {
    return (a - o) * (b - o)
}
inline ll dd(const pii& a, const pii& b) {
    ll dx = a.x - b.x, dy = a.y - b.y;
    return dx * dx + dy * dy;
}
// 給平面上任意個點，求其凸包。返回順序為逆時針。此方法會移除
     重複點。
#define jud \
    crzf(ret[ret.size() - 2], ret.back(), pp[i]) <= 0
vector<pii> makepoly(vector<pii>& pp) {
    int n = pp.size();
    sort(pp.begin(), pp.end());
    pp.erase(unique(pp.begin(), pp.end()), pp.end());
    vector<pii> ret;
    for (int i = 0; i < n; i++) {
        while (ret.size() >= 2 && jud) ret.pop_back();
        ret.push_back(pp[i]);
    }
    for (int i = n - 2, t = ret.size() + 1; i >= 0; i--) {
        while (ret.size() >= t && jud) ret.pop_back();
        ret.push_back(pp[i]);
    }
    if (n >= 2) ret.pop_back();
    return ret;
}
// (shoelace formula)
// 給凸包，問其面積「的兩倍」。若凸包少於三個點，回傳零。
ll area(vector<pii>& poly) {
    int n = poly.size();
    ll ret = 0;
    for (int i = 0; i < n; i++)
        ret += (poly[i].x * poly[ii].y);
    for (int i = 0; i < n; i++)
        ret -= (poly[i].y * poly[ii].x);
    return ret;
}
// 給凸包，問其兩點最遠距離「的平方」。若要問平面上任意個點的
     兩點最遠
// 距離，請先轉成凸包。若凸包少於兩個點，回傳零。
#define kk (k + 1) % n
ll maxdist(vector<pii>& poly) {
    int k = 1, n = poly.size();
    if (n < 2) return 0;
    if (n == 2) return dd(poly[0], poly[1]);
    ll ret = 0;
    for (int i = 0; i < n; i++) {
        while (abs(crzf(poly[kk], poly[i], poly[ii])) >=
               abs(crzf(poly[k], poly[i], poly[ii])))
            k = kk;
        ret = max(ret, max(dd(poly[i], poly[k]),
                           dd(poly[ii], poly[k])));
    }
    return ret;
}
```

4.5 MinRect

```cpp
// 全部浮點數運算，先製作凸包，然後呼叫 minrect
typedef long double dd;
typedef pair<dd, dd> pii;
#define x first
#define y second
#define in inline
#define cp const pii&
#define op operator
#define ab (cp a, cp b)
const dd eps = 1e-8;
in pii op+ab { return {a.x + b.x, a.y + b.y}; }
in pii op-ab { return {a.x - b.x, a.y - b.y}; }
in pii op*(cp p, dd v) { return {v * p.x, v * p.y}; }
in dd  op^ab { return a.x * b.x + a.y * b.y; }
in dd  op*ab { return a.x * b.y - a.y * b.x; }
in dd  op%ab {
    dd dx = a.x - b.x, dy = a.y - b.y;
    return dx * dx + dy * dy;
}
in dd crzf(cp o, cp a, cp b) { return (a - o) * (b - o); }
in dd dotf(cp o, cp a, cp b) { return (a - o) ^ (b - o); }

#define judge \
    crzf(ret[ret.size() - 2], ret.back(), pp[i]) <= eps
vector<pii> makepoly(vector<pii>& pp) {
    sort(pp.begin(), pp.end());
    pp.erase(unique(pp.begin(), pp.end()), pp.end());
    int n = pp.size(); vector<pii> ret;
    for (int i = 0; i < n; i++) {
        while (ret.size() >= 2 && judge) ret.pop_back();
        ret.push_back(pp[i]);
    }
    for (int i = n - 2, s = ret.size() + 1; i >= 0; i--) {
        while (ret.size() >= s && judge) ret.pop_back();
        ret.push_back(pp[i]);
    }
    if (n >= 2) ret.pop_back(); return ret;
}

// 給凸包，問最小覆蓋矩形面積以及該矩形頂點座標 (存於 rec)
// 。頂點座標按照凸包製作方式排序。如果不需要矩形座標，把跟
// rec 有關的程式碼移除。
#define xx(i) ((i + 1) % n)
in pii foot(cp s1, cp s2, cp q) {
return s1 + (s2 - s1) * dotf(s1, s2, q) * (1 / (s1 % s2));
}
dd minrect(const vector<pii>& poly, vector<pii>& rec) {
    int n = poly.size(); if (n < 3) return 0;
    dd minn = 1e50; rec.resize(4);
    int j = 1, k = 1, r;
    for (int i = 0; i < n; i++) {
        while (crzf(poly[i], poly[xx(i)], poly[xx(j)]) -
               crzf(poly[i], poly[xx(i)], poly[j]) > -eps)
            j = xx(j);
        while (dotf(poly[i], poly[xx(i)], poly[xx(k)]) -
               dotf(poly[i], poly[xx(i)], poly[k]) > -eps)
            k = xx(k);
        if (i == 0) r = k;
        while (dotf(poly[i], poly[xx(i)], poly[xx(r)]) -
               dotf(poly[i], poly[xx(i)], poly[r]) < eps)
            r = xx(r);
        dd a = crzf(poly[i], poly[xx(i)], poly[j]) *
               (dotf(poly[i], poly[xx(i)], poly[k]) -
                dotf(poly[i], poly[xx(i)], poly[r])) /
               (poly[i] % poly[xx(i)]);
        a = abs(a); if (a < minn) { minn = a;
            rec[0] = foot(poly[i], poly[xx(i)], poly[r]);
            rec[1] = foot(poly[i], poly[xx(i)], poly[k]);
            pii toss = foot(poly[i], poly[xx(i)], poly[j]);
            rec[2] = poly[j] + rec[0] - toss;
            rec[3] = poly[j] + rec[1] - toss;
        }
    }
    rec = makepoly(rec); return minn;
}
```

4.6 ClosestPair

```cpp
typedef pair<ll, ll> pii;
#define x first
#define y second
ll dd(const pii& a, const pii& b) {
    ll dx = a.x - b.x, dy = a.y - b.y;
    return dx * dx + dy * dy;
}
const ll inf = 1e18;
ll dac(vector<pii>& p, int l, int r) {
    if (l >= r) return inf;
    int m = (l + r) / 2;
    ll d = min(dac(p, l, m), dac(p, m + 1, r));
    vector<pii> t;
    for (int i = m; i >= l && p[m].x - p[i].x < d; i--)
        t.push_back(p[i]);
    for (int i = m + 1; i <= r && p[i].x - p[m].x < d; i++)
        t.push_back(p[i]);
    sort(t.begin(), t.end(),
         [](pii& a, pii& b) { return a.y < b.y; });
    int n = t.size();
    for (int i = 0; i < n - 1; i++)
        for (int j = 1; j < 4 && i + j < n; j++)
            // 這裡可以知道是哪兩點是最小點對
            d = min(d, dd(t[i], t[i + j]));
    return d;
}
```

```cpp
// 給一堆點，求最近點對的距離「的平方」。
ll closest_pair(vector<pii>& pp) {
    sort(pp.begin(), pp.end());
    return dac(pp, 0, pp.size() - 1);
}
```

5 Graph

5.1 Dijkstra

```cpp
// 0/1-based, edge = {cost，dest}, -1 : unconnected
typedef pair<ll, int> pii;
vector<ll> dijkstra (int s, vector<vector<pii>>& edge) {
    vector<ll> sum(edge.size(), -1);
    priority_queue<pii, vector<pii>, greater<pii>> q;
    q.emplace(0, s);
    while (q.size()) {
        int v = q.top().second; ll d = q.top().first;
        q.pop();
        if (sum[v] != -1) continue;
        sum[v] = d;
        for (auto& e : edge[v])
            if (sum[e.second] == -1)
                q.emplace(d + e.first, e.second);
    } return sum;
}
```

5.2 MahattanMST

```cpp
#define REP(i,n) for(int i=0;i<n;i++)
typedef long long LL;
const int N=200100;
int n,m;
struct PT {int x,y,z,w,id;} p[N];
inline int dis(const PT &a,const PT &b){return abs(a.xb.x)+
     abs(a.y-b.y);}
inline bool cpx(const PT &a,const PT &b)
{return a.x!=b.x? a.x>b.x:a.y>b.y;}
inline bool cpz(const PT &a,const PT &b){return a.z<b.z;}
struct E{int a,b,c;}e[8*N];
bool operator<(const E&a,const E&b){return a.c<b.c;}
struct Node{ int L,R,key; } node[4*N];
int s[N];
int F(int x) {return s[x]==x ? x : s[x]=F(s[x]); }
void U(int a,int b) {s[F(b)]=F(a);}
void init(int id,int L,int R) {
    node[id] = (Node){L,R,-1};
    if(L==R)return;
    init(id*2,L,(L+R)/2);
    init(id*2+1,(L+R)/2+1,R);
}
void ins(int id,int x) {
    if(node[id].key==-1 || p[node[id].key].w>p[x].w)
        node[id].key=x;
    if(node[id].L==node[id].R) return;
    if(p[x].z<=(node[id].L+node[id].R)/2) ins(id*2,x);
    else ins(id*2+1,x);
}
int Q(int id,int L,int R){
    if(R<node[id].L || L>node[id].R)return -1;
    if(L<=node[id].L && node[id].R<=R)return node[id].key;
    int a=Q(id*2,L,R),b=Q(id*2+1,L,R);
    if(b==-1 || (a!=-1 && p[a].w<p[b].w)) return a;
    else return b;
}
void calc() {
    REP(i,n) {
        p[i].z = p[i].y-p[i].x;
        p[i].w = p[i].x+p[i].y;
    }
    sort(p,p+n,cpz);
    int cnt = 0, j, k;
    for(int i=0; i<n; i=j){
        for(j=i+1; p[j].z==p[i].z && j<n; j++);
        for(k=i, cnt++; k<j; k++) p[k].z = cnt;
    }
    init(1,1,cnt);
    sort(p,p+n,cpx);
    REP(i,n) {
        j=Q(1,p[i].z,cnt);
        if(j!=-1) e[m++] = (E){p[i].id, p[j].id, dis(p[i],p[j
             ])};
        ins(1,i);
    }
}
LL MST() {
    LL r=0;
    sort(e, e+m);
    REP(i, m) {
        if(F(e[i].a)==F(e[i].b)) continue;
        U(e[i].a, e[i].b);
        r += e[i].c;
    }
    return r;
}
int main() {
    int ts;
    scanf("%d", &ts);
    while (ts--) {
        m = 0;
        scanf("%d",&n);
        REP(i,n) {scanf("%d%d",&p[i].x,&p[i].y);p[i].id=s[i]=
             i;}
        calc();
        REP(i,n)p[i].y= -p[i].y;
        calc();
        REP(i,n)swap(p[i].x,p[i].y);
        calc();
        REP(i,n)p[i].x=-p[i].x;
        calc();
        printf("%lld\n",MST()*2);
    }
    return 0;
}
```

5.3 LCA

```cpp
/* 三種 0/1-based。 只支援無向樹 */
/* Time: O(N+Q) Space: O(N^2) online */
class SsadpTarjan {
   private:
    int n;
    vector<int> par, dep; vector<vector<int>> ca;
    int dfs(int u, vector<vector<int>>& edge, int d) {
        dep[u] = d;
        for (int a = 0; a < n; a++)
            if (dep[a] != -1)
                ca[a][u] = ca[u][a] = parent(a);
        for (int a : edge[u]) {
            if (dep[a] != -1) continue;
            dfs(a, edge, d + 1);
            par[a] = u;
        }
    }
    int parent(int x) {
        if (par[x] == x) return x;
        return par[x] = parent(par[x]);
    }
   public:
    SsadpTarjan(vector<vector<int>>& edge, int root)
        : n(edge.size()) {
        dep.assign(n, -1); par.resize(n);
        ca.assign(n, vector<int>(n));
        for (int i = 0; i < n; i++) par[i] = i;
        dfs(root, edge, 0);
    }
    int lca(int a, int b) { return ca[a][b]; }
    int dist(int a, int b) {
        return dep[a] + dep[b] - 2 * dep[ca[a][b]];
    }
};
/* Time: O(N+Q)  Space: O(N+Q) only offline */
#define x first
#define y second
class OfflineTarjan {
   private:
    vector<int> par, anc, dep, ans, rank;
    vector<vector<pii>> qry;
    vector<vector<int>>& edge; // 安全考量可把 & 去掉
    int root, n;
    void merge(int a, int b) {
        a = parent(a), b = parent(b);
        if (rank[a] < rank[b]) swap(a, b);
        else if (rank[a] == rank[b]) rank[a]++;
        par[b] = a;
    }
    void dfs(int u, int d) {
        anc[parent(u)] = u, dep[u] = d;
        for (int a : edge[u]) {
            if (dep[a] != -1) continue;
            dfs(a, d + 1);
            merge(a, u);
            anc[parent(u)] = u;
        }
        for (auto q : qry[u])
            if (dep[q.first] != -1)
                ans[q.second] = anc[parent(q.first)];
    }
    int parent(int x) {
        if (par[x] == x) return x;
        return par[x] = parent(par[x]);
    }
    void solve(vector<pii>& query) {
        dep.assign(n, -1), rank.assign(n, 0);
        par.resize(n), anc.resize(n), qry.resize(n);
        for (int i = 0; i < n; i++) anc[i] = par[i] = i;
        ans.resize(query.size());
        for (int i = 0; i < query.size(); i++) {
            auto& q = query[i];
            qry[q.first].emplace_back(q.second, i);
            qry[q.second].emplace_back(q.first, i);
        }
        dfs(root, 0);
    }
   public:
    // edge 是傳 reference ，完成所有查詢不可改。
    OfflineTarjan(vector<vector<int>>& edge, int root)
        : edge(edge), root(root), n(edge.size()) {}
    // 離線查詢， query 陣列包含所有詢問 {src, dst} 。呼叫一
         次無
    // 論 query 量多少，複雜度都是 O(N) 。所以應盡量只呼叫一
         次。
    vector<int> lca(vector<pii>& query) {
        solve(query); return ans;
    }
    vector<int> dist(vector<pii>& query) {
        solve(query);
        for (int i = 0; i < query.size(); i++) {
            auto & q = query[i];
            ans[i] = dep[q.first] + dep[q.second]
                     - 2 * dep[ans[i]];
        } return ans;
    }
};
/* Udchen Time: O(QlgN) Space: O(NlgN) 。支援非離線。*/
class SparseTableTarjan {
   private:
    int maxlg;
    vector<vector<int>> anc;
    vector<int> dep;
    void dfs(int u, vector<vector<int>>& edge, int d) {
        dep[u] = d;
        for (int i = 1; i < maxlg; i++)
            if (anc[u][i - 1] == -1) break;
            else anc[u][i] = anc[anc[u][i - 1]][i - 1];
        for (int a : edge[u]) {
            if (dep[a] != -1) continue;
            anc[a][0] = u;
            dfs(a, edge, d + 1);
        }
    }
   public:
    SparseTableTarjan(vector<vector<int>>& edge, int root) {
        int n = edge.size();
        maxlg = ceil(log2(n));
        anc.assign(n, vector<int>(maxlg, -1));
        dep.assign(n, -1);
        dfs(root, edge, 0);
    }
    int lca(int a, int b) {
        if (dep[a] > dep[b]) swap(a, b);
        for (int k = 0; dep[b] - dep[a]; k++)
            if (((dep[b] - dep[a]) >> k) & 1) b = anc[b][k];
        if (a == b) return a;
        for (int k = maxlg - 1; k >= 0; k--)
            if (anc[a][k] != anc[b][k])
                a = anc[a][k], b = anc[b][k];
        return anc[a][0];
    }
    int dist(int a, int b) {
        return dep[a] + dep[b] - 2 * dep[lca(a, b)];
    }
};
```

5.4 BCC_edge

```cpp
邊雙連通
任意兩點間至少有兩條不重疊的路徑連接，找法：
1. 標記出所有的橋
2. 對全圖進行 DFS，不走橋，每一次 DFS 就是一個新的邊雙連通
// from BCW
struct BccEdge {
  static const int MXN = 100005;
  struct Edge { int v,eid; };
  int n,m,step,par[MXN],dfn[MXN],low[MXN];
  vector<Edge> E[MXN];
  DisjointSet djs;
  void init(int _n) {
    n = _n; m = 0;
    for (int i=0; i<n; i++) E[i].clear();
    djs.init(n);
  }
  void add_edge(int u, int v) {
    E[u].PB({v, m});
    E[v].PB({u, m});
    m++;
  }
  void DFS(int u, int f, int f_eid) {
    par[u] = f;
    dfn[u] = low[u] = step++;
    for (auto it:E[u]) {
      if (it.eid == f_eid) continue;
      int v = it.v;
      if (dfn[v] == -1) {
        DFS(v, u, it.eid);
        low[u] = min(low[u], low[v]);
      } else {
        low[u] = min(low[u], dfn[v]);
      }
    }
  }
  void solve() {
    step = 0;
    memset(dfn, -1, sizeof(int)*n);
    for (int i=0; i<n; i++) {
      if (dfn[i] == -1) DFS(i, i, -1);
    }
    djs.init(n);
    for (int i=0; i<n; i++) {
      if (low[i] < dfn[i]) djs.uni(i, par[i]);
    }
  }
} graph;
```

5.5 SPFA

```cpp
vector<pii> G[maxn];
int dis[maxn]; bool inque[maxn];
void SPFA (int n, int s) { //O(kE) k~2.
    for(int i = 1; i <= n; i++) dis[i] = INF;
    queue<int> q; q.push(s); dis[s] = 0;
    while (!q.empty()) {
        int u = q.front(); q.pop(); inque[u] = 0;
        for (pii e : G[u]) {
            int v = e.first , w = e.second;
            if( dis[u] + w < dis[v]) {
                if (!inque[v]) q.push(v), inque[v] = true;
                dis[v] = dis[u] + w;
            }
        }
    }
}
```

5.6 Tarjan

```cpp
割點
點 u 為割點 if and only if 滿足 1. or 2.
1. u 爲樹根，且 u 有多於一個子樹。
2. u 不爲樹根，且滿足存在 (u,v) 爲樹枝邊 (或稱父子邊，即 u 爲
      v 在搜索樹中的父親)，使得 DFN(u) <= Low(v)。
-------------------------------------------------------
橋
一條無向邊 (u,v) 是橋 if and only if (u,v) 爲樹枝邊，且滿足
     DFN(u) < Low(v)。
// 0 base
struct TarjanSCC{
    static const int MAXN = 1000006;
    int n, dfn[MAXN], low[MAXN], scc[MAXN], scn, count;
    vector<int> G[MAXN];
    stack<int> stk;
    bool ins[MAXN];
    void tarjan(int u) {
        dfn[u] = low[u] = ++count;
        stk.push(u);
        ins[u] = true;
        for(auto v:G[u]) {
            if(!dfn[v]) {
                tarjan(v);
                low[u] = min(low[u], low[v]);
            } else if(ins[v]) {
                low[u] = min(low[u], dfn[v]);
            }
        }
        if(dfn[u] == low[u]) {
            int v;
            do {
            v = stk.top(); stk.pop();
            scc[v] = scn;
            ins[v] = false;
            } while(v != u);
            scn++;
        }
    }
    void getSCC(){
        memset(dfn,0,sizeof(dfn));
        memset(low,0,sizeof(low));
        memset(ins,0,sizeof(ins));
        memset(scc,0,sizeof(scc));
        count = scn = 0;
        for(int i = 0 ; i < n ; i++ )
            if(!dfn[i]) tarjan(i);
    }
} SCC;
```

5.7 BellmanFord

```cpp
vector<pii> G[maxn];
int dis[maxn];
bool BellmanFord (int n, int s) {
    for (int i = 1; i <= n; i++) dis[i] = INF;
    dis[s] = 0;
    bool relax;
    for (int r = 1; r <= n; r++) { //O(VE)
        relax = false;
        for (int i = 1; i <= n; i++)
            for (pii e : G[i])
                if ( dis[i] + e.second < dis[e.first] )
                    dis[e.first] = dis[i] + e.second, relax = true;
    } return relax; //有負環
}
```

5.8 KirchhoffMatrixTree

```cpp
// D : degree-matrix
// A : adjacent-matrix
// 無向圖
    // (u,v)
    // A[u][v]++, A[v][u]++
    // D[u][u]++, D[v][v]++
    // G = D-A
    // abs(det(G去掉i-col和i-row))
    // 生成樹的數量
// 有向圖
    // A[u][v]++
    // D[v][v]++ (in-deg)
    // 以i為root的樹形圖數量
    // 所有節點都能到達root
```

5.9 Two_SAT

```cpp
const int N = 5010 * 2;  // 變數最大數量的兩倍
namespace Two_Sat {
vector<int> a[N], b[N], stk;
int vis[N], res[N];
void dfs(int u, vector<int>* g, int sc) {
    vis[u] = 1, res[u] = sc;
    for (int v : g[u]) if (!vis[v]) dfs(v, g, sc);
    if (g == a) stk.push_back(u);
}
// 先呼叫 imply 來設定約束，然後呼叫 scc 跑分析。
// var[x] 的真值對應 i = x * 2 ; var[x] 的假值對應 i = x * 2 + 1
// e.g. 若 var[3] 為真則 var[6] 必為假，則呼叫 imply(6, 13)
void imply(int u, int v) {  // if u then v
    a[u].push_back(v), b[v].push_back(u);
}
// 跑 two_sat ，回傳 true 表示有解。解答存於 Two_Sat::res
// e.g. 若 res[13] == 1 表 var[6] 必為假
// e.g. 若 res[0] == 1 且 res[1] == 1 ，表 var[0] 必為真且必為假，矛盾，無解。
int scc(int n /*變數實際數量的兩倍*/) {
    memset(vis, 0, sizeof(vis));
    for (int i = 0; i < n; i++) if (!vis[i]) dfs(i, a, -1);
    memset(vis, 0, sizeof(vis));
    int sc = 0;
    while (!stk.empty()) {
        if (!vis[stk.back()]) dfs(stk.back(), b, sc++);
        stk.pop_back();
    }
    for (int i = 0; i < n; i += 2) {
        if (res[i] == res[i + 1]) return 0;
        if (res[i] > res[i + 1]) res[i] = 1, res[i + 1] = 0;
        else res[i] = 0, res[i + 1] = 1;
    }
    return 1;
}
}  // namespace Two_Sat
```

5.10 MinMeanCycle

```cpp
#include<cfloat> //for DBL_MAX
int dp[MAXN][MAXN]; // 1-base,O(NM)
vector<tuple<int,int,int>> edge;
double mmc(int n){ //allow negative weight
    const int INF = 0x3f3f3f3f;
    for(int t=0; t<n; ++t){
        memset(dp[t+1],0x3f,sizeof(dp[t+1]));
        for(const auto &e:edge) {
            int u, v, w; tie(u,v,w) = e;
            dp[t+1][v] = min(dp[t+1][v],dp[t][u]+w);
        }
    }
    double res = DBL_MAX;
    for(int u=1; u<=n; ++u) {
        if(dp[n][u]==INF) continue;
        double val = -DBL_MAX;
        for(int t=0;t<n;++t)
            val = max(val,(dp[n][u]-dp[t][u])*1.0/(n-t));
        res = min(res,val);
    } return res;
}
```

5.11 Prim

```cpp
// 0/1-based n(必須剛好) edge:{cost, dest}
typedef pair<ll, int> pii;
ll MST(vector<vector<pii>>& edge, int n) {
    vector<bool> vis(n + 1);
    priority_queue<pii, vector<pii>, greater<pii>> q;
    q.emplace(0, 1);
    ll ret = 0; int nvis = 0;
    while (nvis < n && q.size()) {
        ll d = q.top().first;
        int v = q.top().second; q.pop();
        if (vis[v]) continue;
        vis[v] = 1; ret += d;
        if (++nvis == n) return ret;
        for (auto& e : edge[v])
            if (!vis[e.second]) q.push(e);
    } return -1; // unconnected
}
```

6 Math

6.1 Simplex

```cpp
/*target:
  max \sum_{j=1}^n A_{0,j}*x_j
condition:
  \sum_{j=1}^n A_{i,j}*x_j <= A_{i,0} |i=1~m
  x_j >= 0 |j=1~n
VDB = vector<double>*/
template<class VDB>
VDB simplex(int m,int n,vector<VDB> a){
  vector<int> left(m+1), up(n+1);
  iota(left.begin(), left.end(), n);
  iota(up.begin(), up.end(), 0);
  auto pivot = [&](int x, int y){
    swap(left[x], up[y]);
    auto k = a[x][y]; a[x][y] = 1;
    vector<int> pos;
    for(int j = 0; j <= n; ++j){
      a[x][j] /= k;
      if(a[x][j] != 0) pos.push_back(j);
    }
    for(int i = 0; i <= m; ++i){
      if(a[i][y]==0 || i == x) continue;
      k = a[i][y], a[i][y] = 0;
      for(int j : pos) a[i][j] -= k*a[x][j];
    }
  };
  for(int x,y;;){
    for(int i=x=1; i <= m; ++i)
      if(a[i][0]<a[x][0]) x = i;
    if(a[x][0]>=0) break;
    for(int j=y=1; j <= n; ++j)
      if(a[x][j]<a[x][y]) y = j;
    if(a[x][y]>=0) return VDB();//infeasible
    pivot(x, y);
  }
  for(int x,y;;){
    for(int j=y=1; j <= n; ++j)
      if(a[0][j] > a[0][y]) y = j;
    if(a[0][y]<=0) break;
    x = -1;
    for(int i=1; i<=m; ++i) if(a[i][y] > 0)
      if(x == -1 || a[i][0]/a[i][y]
        < a[x][0]/a[x][y]) x = i;
    if(x == -1) return VDB();//unbounded
    pivot(x, y);
  }
  VDB ans(n + 1);
  for(int i = 1; i <= m; ++i)
    if(left[i] <= n) ans[left[i]] = a[i][0];
  ans[0] = -a[0][0];
  return ans;
}
```

6.2 Fraction

```
#define cfl(str) (const frac& f) const { return str; }
#define cll(str) (ll l) const { return str; }
#define lfl(str) (ll l, const frac& f) { return str; }
#define ff inline frac operator
#define bb inline bool operator
#define fff inline friend frac operator
#define fbb inline friend bool operator

class frac {
   private: ll x, y;
   public:
    frac() : x(0), y(1) {}
    frac(ll v) : x(v), y(1) {}
    frac(ll xx, ll yy, bool f = 0) : x(xx), y(yy) {
        assert(y != 0);
        if (!f) {
            ll g = __gcd(x, y);
            x /= g, y /= g;
            if (y < 0) x *= -1, y *= -1;
        }
    }
    // 以下斟酌使用，不必全抄
    ff = (ll l) { return frac(l); }
    ff - () const { return frac(-x, y, 1); }
    ff ! () const {  // 倒數
        return x > 0 ? frac(y, x, 1) : frac(-y, -x, 1);
    }

    bb >  cfl(x * f.y > y * f.x)
    bb <  cfl(x * f.y < y * f.x)
    bb <= cfl(x * f.y <= y * f.x)
    bb >= cfl(x * f.y >= y * f.x)
    bb == cfl(x == f.x && y == f.y)
    bb != cfl(x != f.x || y != f.y)
    ff +  cfl(frac(x * f.y + y * f.x, y * f.y))
    ff -  cfl(frac(x * f.y - y * f.x, y * f.y))
    ff *  cfl(frac(x * f.x, y * f.y))
    ff /  cfl(frac(x * f.y, y * f.x))

    bb >  cll(x > l * y)
    bb <  cll(x < l * y)
    bb >= cll(x >= l * y)
    bb <= cll(x <= l * y)
    bb == cll(x == l * y)
    bb != cll(x != l * y)
    ff +  cll(frac(x + l * y, y))
    ff -  cll(frac(x - l * y, y))
    ff *  cll(frac(l * x, y))
    ff /  cll(frac(x, l * y))

    fbb <  lfl(f > l)
    fbb >  lfl(f < l)
    fbb <= lfl(f >= l)
    fbb >= lfl(f <= l)
    fbb == lfl(f == l)
    fbb != lfl(f != l)
    fff +  lfl(f + l)
    fff -  lfl(-f + l)
    fff *  lfl(f * l)
    fff /  lfl(!f * l)

    inline operator double() { return (double)x / y; }
    inline friend frac abs(const frac& f) {
        return frac(abs(f.x), f.y, 1);
    }
    inline friend ostream& operator <<
         (ostream & out, const frac& f) {
        out << f.x;
        if (f.y != 1) out << '/' << f.y;
        return out;
    }
};
```

6.3 FFT

```
template<typename T,typename VT=vector<complex<T> > >
struct FFT{
    const T pi;
    FFT(const T pi=acos((T)-1)):pi(pi){}
    unsigned bit_reverse(unsigned a,int len){
        a=((a&0x55555555U)<<1)|((a&0xAAAAAAAAU)>>1);
        a=((a&0x33333333U)<<2)|((a&0xCCCCCCCCU)>>2);
        a=((a&0x0F0F0F0FU)<<4)|((a&0xF0F0F0F0U)>>4);
        a=((a&0x00FF00FFU)<<8)|((a&0xFF00FF00U)>>8);
        a=((a&0x0000FFFFU)<<16)|((a&0xFFFF0000U)>>16);
        return a>>(32-len);
    }
    void fft(bool is_inv,VT &in,VT &out,int N){
        int bitlen=__lg(N),num=is_inv?-1:1;
        for(int i=0;i<N;++i) out[bit_reverse(i,bitlen)]=in[i];
        for(int step=2; step<=N; step<<=1){
            const int mh = step>>1;
            for(int i=0; i<mh; ++i){
                complex<T> wi = exp(complex<T>(0,i*num*pi/mh));
                for(int j=i; j<N; j+=step){
                    int k = j+mh;
                    complex<T> u = out[j], t = wi*out[k];
                    out[j] = u+t;
                    out[k] = u-t;
                }
            }
        }
        if(is_inv) for(int i=0;i<N;++i) out[i]/=N;
    }
};
```

6.4 FindRealRoot

```
// an*x^n + ... + a1x + a0 = 0;
int sign(double x){
  return x < -eps ? -1 : x > eps;
}
double get(const vector<double>&coef, double x){
  double e = 1, s = 0;
  for(auto i : coef) s += i*e, e *= x;
  return s;
}
double find(const vector<double>&coef, int n, double lo, double hi){
  double sign_lo, sign_hi;
  if( !(sign_lo = sign(get(coef,lo))) ) return lo;
  if( !(sign_hi = sign(get(coef,hi))) ) return hi;
  if(sign_lo * sign_hi > 0) return INF;
  for(int stp = 0; stp < 100 && hi - lo > eps; ++stp){
    double m = (lo+hi)/2.0;
    int sign_mid = sign(get(coef,m));
    if(!sign_mid) return m;
    if(sign_lo*sign_mid < 0) hi = m;
    else lo = m;
  }
  return (lo+hi)/2.0;
}
vector<double> cal(vector<double>coef, int n){
  vector<double>res;
  if(n == 1){
    if(sign(coef[1])) res.pb(-coef[0]/coef[1]);
    return res;
  }
  vector<double>dcoef(n);
  for(int i = 0; i < n; ++i) dcoef[i] = coef[i+1]*(i+1);
  vector<double>droot = cal(dcoef, n-1);
  droot.insert(droot.begin(), -INF);
  droot.pb(INF);
  for(int i = 0; i+1 < droot.size(); ++i){
    double tmp = find(coef, n, droot[i], droot[i+1]);
    if(tmp < INF) res.pb(tmp);
  }
  return res;
}
int main () {
  vector<double>ve;
  vector<double>ans = cal(ve, n);
  // 視情況把答案 +eps，避免 -0
}
```

6.5 質因數分解

```
LL func(const LL n,const LL mod,const int c) {
  return (LLmul(n,n,mod)+c+mod)%mod;
}
LL pollorrho(const LL n, const int c) {//循環節長度
  LL a=1, b=1;
  a=func(a,n,c)%n;
  b=func(b,n,c)%n; b=func(b,n,c)%n;
  while(gcd(abs(a-b),n)==1) {
    a=func(a,n,c)%n;
    b=func(b,n,c)%n; b=func(b,n,c)%n;
  }
  return gcd(abs(a-b),n);
}
void prefactor(LL &n, vector<LL> &v) {
  for(int i=0;i<12;++i) {
    while(n%prime[i]==0) {
      v.push_back(prime[i]);
      n/=prime[i];
    }
  }
}
```

```
void smallfactor(LL n, vector<LL> &v) {
  if(n<MAXPRIME) {
    while(isp[(int)n]) {
      v.push_back(isp[(int)n]);
      n/=isp[(int)n];
    }
    v.push_back(n);
  } else {
    for(int i=0;i<primecnt&&prime[i]*prime[i]<=n;++i) {
      while(n%prime[i]==0) {
        v.push_back(prime[i]);
        n/=prime[i];
      }
    }
    if(n!=1) v.push_back(n);
  }
}
void comfactor(const LL &n, vector<LL> &v) {
  if(n<1e9) {
    smallfactor(n,v);
    return;
  }
  if(Isprime(n)) {
    v.push_back(n);
    return;
  }
  LL d;
  for(int c=3;;++c) {
    d = pollorrho(n,c);
    if(d!=n) break;
  }
  comfactor(d,v);
  comfactor(n/d,v);
}
void Factor(const LL &x, vector<LL> &v) {
  LL n = x;
  if(n==1) { puts("Factor 1"); return; }
  prefactor(n,v);
  if(n==1) return;
  comfactor(n,v);
  sort(v.begin(),v.end());
}
void AllFactor(const LL &n,vector<LL> &v) {
  vector<LL> tmp;
  Factor(n,tmp);
  v.clear();
  v.push_back(1);
  int len;
  LL now=1;
  for(int i=0;i<tmp.size();++i) {
    if(i==0 || tmp[i]!=tmp[i-1]) {
      len = v.size();
      now = 1;
    }
    now*=tmp[i];
    for(int j=0;j<len;++j)
      v.push_back(v[j]*now);
  }
}
```

6.6 Karatsuba

```
// N is power of 2
template<typename Iter>
void DC(int N, Iter tmp, Iter A, Iter B, Iter res){
    fill(res,res+2*N,0);
    if (N<=32){
        for (int i=0; i<N; i++)
            for (int j=0; j<N; j++)
                res[i+j] += A[i]*B[j];
        return;
    }
    int n = N/2;
    auto a = A+n, b = A;
    auto c = B+n, d = B;
    DC(n,tmp+N,a,c,res+2*N);
    for (int i=0; i<N; i++){
        res[i+N] += res[2*N+i];
        res[i+n] -= res[2*N+i];
    }
    DC(n,tmp+N,b,d,res+2*N);
    for (int i=0; i<N; i++){
        res[i] += res[2*N+i];
        res[i+n] -= res[2*N+i];
    }
    auto x = tmp;
    auto y = tmp+n;
    for (int i=0; i<n; i++) x[i] = a[i]+b[i];
    for (int i=0; i<n; i++) y[i] = c[i]+d[i];
    DC(n,tmp+N,x,y,res+2*N);
    for (int i=0; i<N; i++)
        res[i+n] += res[2*N+i];
}
// DC(1<<16,tmp.begin(),A.begin(),B.begin(),res.begin());
```

6.7 FastPow

```
ll fastpow(ll a, int p) { // a ^ p
    ll ret = 1;
    while (p) {
        if (p & 1) ret *= a;
        a *= a, p >>= 1;
    } return ret;
}
ll fastpow(ll a, ll p, ll m) { // (a ^ p) % m
    ll ret = 1;
    while (p) {
        if (p & 1) ret = ret * a % m;
        a = a * a % m, p >>= 1;
    } return ret;
}
```

6.8 MillerRabin

```
//From jacky860226
typedef long long LL;
inline LL mul(LL a,LL b,LL m){//a*b%m
    return (a%m)*(b%m)%m;
}
/*LL mul(LL a,LL b,LL m){//a*b%m
    a %= m, b %= m;
    LL y = (LL)((double)a*b/m+0.5); //fast for m < 2^58
    LL r = (a*b-y*m)%m;
    return r<0 ? r+m : r;
}*/
template<typename T> T pow(T a,T b,T mod) { //a^b%mod
    T ans = 1;
    while(b) {
        if(b&1) ans = mul(ans,a,mod);
        a = mul(a,a,mod);
        b >>= 1;
    } return ans;
}
template<typename T> bool isprime(T n, int num) { //num = 3,7
    int sprp[3] = {2,7,61}; //int範圍可解
    //int llsprp[7] = {2,325,9375,28178,450775,9780504,1795265022}; //至少unsigned long long範圍
    if(n==2) return true;
    if(n<2 || n%2==0) return false;
    //n-1 = u * 2^t
    int t = 0;  T u = n-1;
    while(u%2==0) u >>= 1, t++;
    for(int i=0; i<num; i++) {
        T a = sprp[i]%n;
        if(a==0 || a==1 || a==n-1) continue;
        T x = pow(a,u,n);
        if(x==1 || x==n-1) continue;
        for(int j=1; j<t; j++) {
            x = mul(x,x,n);
            if(x==1) return false;
            if(x==n-1) break;
        }
        if(x!=n-1) return false;
    } return true;
}
```

6.9 Discrete_sqrt

```
int order(ll b, ll p) {
    if (__gcd(b, p) != 1) return -1;
    int ret = 2;
    while (++ret)
        if (fastpow(b, ret, p) == 1) break;
    return ret;
}
// 把 fastpow 也抄過來，會用到。
// 問 (x^2 = y) mod p 的解。回傳 -1 表示 x 無解。
ll dsqrt(ll y, ll p) {
    if (__gcd(y, p) != 1) return -1;
    if (fastpow(y, (p - 1 / 2), p) == p - 1) return -1;
    int e = 0;
    ll s = p - 1;
    while (!(s & 1)) s >>= 1, e++;
    int q = 2;
    while (1)
        if (fastpow(q, (p - 1) / 2, p) == p - 1)
            break;
        else q++;
    ll x = fastpow(y, (s + 1) / 2, p);
    ll b = fastpow(y, s, p);
    ll g = fastpow(q, s, p);
    while (1) {
        int m;
        for (m = 0; m < e; m++) {
            int o = order(p, b);
            if (o == -1) return -1;
            if (o == fastpow(2, m, p)) break;
        }
        if (m == 0) return x;
        x = x * fastpow(g, fastpow(2, e - m - 1), p) % p;
        g = fastpow(g, fastpow(2, e - m, p), p);
        b = b * g % p;
        if (b == 1) return x;
        e = m;
    }
}
```

6.10 PrimeList

```
12721       13331       14341       75577
123457      222557      556679      880301
999983      1e6+99      1e9+9       2e9+99
1e12+39     1e15+37     1e9+7       1e7+19
1097774749  1076767633  100102021
999997771   1001010013  1000512343
987654361   999991231   999888733
98789101    987777733   999991921
1010101333  1010102101
2305843009213693951     4611686018427387847
9223372036854775783     18446744073709551557
```

6.11 Matrix

```
struct Matrix {
    int r, c;
    vector<vector<ll>> m;
    Matrix(int r, int c): r(r), c(c), m(r, vector<ll>(c)) {}
    vector<ll> &operator[](int i) { return m[i]; }
    Matrix operator+(const Matrix &a) {
        Matrix rev(r, c);
        for (int i = 0; i < r; ++i)
            for (int j = 0; j < c; ++j)
                rev[i][j] = m[i][j] + a.m[i][j];
        return rev;
    }
    Matrix operator-(const Matrix &a) {
        Matrix rev(r, c);
        for (int i = 0; i < r; ++i)
            for (int j = 0; j < c; ++j)
                rev[i][j] = m[i][j] - a.m[i][j];
        return rev;
    }
    Matrix operator*(const Matrix &a) {
        Matrix rev(r, a.c);
        Matrix tmp(a.c, a.r);
        for (int i = 0; i < a.r; ++i)
            for (int j = 0; j < a.c; ++j)
                tmp[j][i] = a.m[i][j];
        for (int i = 0; i < r; ++i)
            for (int j = 0; j < a.c; ++j)
                for (int k = 0; k < c; ++k)
                    rev.m[i][j] += m[i][k] * tmp[j][k];
        return rev;
    }
    // 回傳反矩陣。注意這是 const 方法所以原矩陣不受影響。
    Matrix inverse() const {
        Matrix t(r, r + c);
        for (int y = 0; y < r; y++) {
            t.m[y][c + y] = 1;
            for (int x = 0; x < c; x++) t.m[y][x] = m[y][x];
        }
        if (!t.gauss()) return Matrix(0, 0);
        Matrix ret(c, r);
        for (int y = 0; y < r; y++)
            for (int x = 0; x < c; x++)
                ret[y][x] = t.m[y][c + x] / t.m[y][y];
        return ret;
    }
    // 做高斯消去 (最高次係數應置於最左，常數應置於最右) 並回傳 det
    // 行列式值。複雜度 O(n^3)。如果不是方陣，回傳值無意義。
    ll gauss() {
        vector<ll> lazy(r, 1);
        bool sign = false;
        for (int i = 0; i < r; ++i) {
            if (m[i][i] == 0) {
                int j = i + 1;
                while (j < r && !m[j][i]) j++;
                if (j == r) continue;
                m[i].swap(m[j]); sign = !sign;
            }
            for (int j = 0; j < r; ++j) {
                if (i == j) continue;
                lazy[j] = lazy[j] * m[i][i];
                ll mx = m[j][i];
                for (int k = 0; k < c; ++k)
                    m[j][k] =
                        m[j][k] * m[i][i] - m[i][k] * mx;
            }
        }
        ll det = sign ? -1 : 1;
        for (int i = 0; i < r; ++i) {
            det = det * m[i][i] / lazy[i];
            for (auto &j : m[i]) j /= lazy[i];
        }
        return det;
    }
};
```

6.12 SG

```
Anti Nim (取走最後一個石子者敗) :
先手必勝 if and only if
1. 「所有」堆的石子數都為 1 且遊戲的 SG 值為 0。
2. 「有些」堆的石子數大於 1 且遊戲的 SG 值不為 0。
-------------------------------------------------------
Anti-SG (決策集合為空的遊戲者贏) :
定義 SG 值為 0 時，遊戲結束，
則先手必勝 if and only if
1. 遊戲中沒有單一遊戲的 SG 函數大於 1 且遊戲的 SG 函數為 0。
2. 遊戲中某個單一遊戲的 SG 函數大於 1 且遊戲的 SG 函數不為 0。
-------------------------------------------------------
Sprague-Grundy :
1. 雙人、回合制
2. 資訊完全公開
3. 無隨機因素
4. 可在有限步內結束
5. 沒有和局
6. 雙方可採取的行動相同

SG(S) 的值為 0：後手(P)必勝
不為 0：先手(N)必勝
int mex(set S) {
  // find the min number >= 0 that not in the S
  // e.g. S = {0, 1, 3, 4} mex(S) = 2
}
state = []
int SG(A) {
  if (A not in state) {
    S = sub_states(A)
    if( len(S) > 1 ) state[A] = reduce(operator.xor, [SG(B) for B in S])
    else state[A] = mex(set(SG(B) for B in next_states(A)))
  } return state[A]
}
```

6.13 ModInv

```
// 解 (ax == 1) mod b。a、b必須是互質整數，否則不存在mod inverse。
int phi(int x) {
    int r = x;
    for (int p = 2; p * p <= x; p++) {
        if (x % p == 0) {
            while (x % p == 0) x /= p;
            r -= r / p;
        }
    }
    if (x > 1) r -= r / x;
    return r;
}

ll modinv(ll a, ll b){
  if(__gcd(a, b) != 1){
    return -1;
  }
  // Euler 定理: a^phi(b) == 1 (mod b)
  // -> a^(phi(b) - 1) is the mod inverse to b of a
  int mod_inv_pow = phi(b) - 1;
  int ans = 1, base = a % b;
  while(mod_inv_pow > 0){
    if(mod_inv_pow & 1){
      ans = ans * base % b;
    }
    base = base * base % b;
    mod_inv_pow >>= 1;
  }
  return ans;
}

// 解 (ax == 1) mod p 。p 必須是質數，a 是正整數。
ll modinv(ll a, ll p) {
    if (p == 1) return 0;
    ll pp = p, y = 0, x = 1;
    while (a > 1) {
```

```cpp
        ll q = a / p, t = p;
        p = a % p, a = t, t = y, y = x - q * y, x = t;
    }
    if (x < 0) x += pp;
    return x;
}
// 解 (ax == b) mod p 。p 必須是質數，a 和 b 是正整數。
ll modinv(ll a, ll b, ll p) {
    ll ret = modinv(a, p);
    return ret * b % p;
}
```

6.14 外星模運算

```cpp
//a[0]^(a[1]^a[2]^...)
#define maxn 1000000
int euler[maxn+5];
bool is_prime[maxn+5];
void init_euler(){
  is_prime[1] = 1; //一不是質數
  for(int i=1; i<=maxn; i++) euler[i]=i;
  for(int i=2; i<=maxn; i++) {
    if(!is_prime[i]) { //是質數
      euler[i]--;
      for(int j=i<<1; j<=maxn; j+=i) {
        is_prime[j]=1;
        euler[j] = euler[j]/i*(i-1);
      }
    }
  }
}
LL pow(LL a, LL b, LL mod) { //a^b%mod
  LL ans=1;
  for(; b; a=a*a%mod, b>>=1)
    if(b&1) ans = ans*a%mod;
  return ans;
}
bool isless(LL *a, int n, int k) {
  if(*a==1)return k>1;
  if(--n==0)return *a<k;
  int next=0;
  for(LL b=1;b<k;++next)
    b *= *a;
  return isless(a+1, n, next);
}
LL high_pow(LL *a, int n, LL mod){
  if(*a==1||--n==0)return *a%mod;
  int k = 0, r = euler[mod];
  for(LL tma=1;tma!=pow(*a,k+r,mod);++k)
    tma = tma*(*a)%mod;
  if(isless(a+1,n,k))return pow(*a,high_pow(a+1,n,k),mod);
  int tmd = high_pow(a+1,n,r), t = (tmd-k+r)%r;
  return pow(*a,k+t,mod);
}
LL a[1000005]; int t,mod;
int main(){
  init_euler();
  scanf("%d", &t);
  #define n 4
  while(t--){
    for(int i=0;i<n;++i)scanf("%lld", &a[i]);
    scanf("%d", &mod);
    printf("%lld\n", high_pow(a,n,mod));
  }
  return 0;
}
```

6.15 ax+by=gcd(a,b)

```cpp
// 給 a,b ，解 ax+by=gcd(a,b)
typedef pair<ll, ll> pii;
pii extgcd(ll a, ll b) {
    if (b == 0) return {1, 0};
    ll k = a / b;
    pii p = extgcd(b, a - k * b);
    return {p.second, p.first - k * p.second};
}
```

6.16 Expression

```cpp
/**
 * 支援處理四則運算的工具。給四則運算的字串，檢查格式並計算其值。如果
 * 格式不合法，會丟出錯誤。複雜度 O(字串長度) 。支援的符號有四則運算
 * 和求餘數，先乘除後加減。可以使用括號、或前置正負號。數字開頭可以為
 * 零或禁止為零。可以兼容或禁止多重前置號 (例如 --1 視為 1 、+-+-1
 * 視為 -1) 。空字串視為不合法。運算範圍限於 long long 。如果試圖除
 * 以零或對零求餘也會丟出錯誤。
 */
void req(bool b) { if (!b) throw ""; }
const int B = 2; // 可以調整成 B 進位
class Expr {
   private:
    deque<char> src;
    Expr(const string& s) : src(s.begin(), s.end()) {}
    inline char top() {
        return src.empty() ? '\0' : src.front();
    }
    inline char pop() {
        char c = src.front(); src.pop_front(); return c;
    }
    ll n() {
        ll ret = pop() - '0';
        // 若要禁止數字以 0 開頭，加上這行
        // req(ret || !isdigit(top()));
        while (isdigit(top())) ret = B * ret + pop() - '0';
        return ret;
    }
    ll fac() {
        if (isdigit(top())) return n();
        if (top() == '-') { pop(); return -fac(); }
        if (top() == '(') {
            pop();
            ll ret = expr(1);
            req(pop() == ')');
            return ret;
        }
        // 若要允許前置正號，加上這行
        // if(top() == '+') { pop(); return fac(); }
        throw "";
    }
    ll term() {
        ll ret = fac(); char c = top();
        while (c == '*' || c == '/' || c == '%') {
            pop();
            if (c == '*') ret *= fac();
            else {
                ll t = fac(); req(t);
                if (c == '/') ret /= t; else ret %= t;
            }
            c = top();
        } return ret;
    }
    ll expr(bool k) {
        ll ret = term();
        while (top() == '+' || top() == '-')
            if (pop() == '+') ret += term();
            else ret -= term();
        req(top() == (k ? ')' : '\0'));
        return ret;
    }
   public:
    // 給定數學運算的字串，求其值。若格式不合法，丟出錯誤。
    static ll eval(const string& s) {
        // 若要禁止多重前置號，加上這四行
        // req(s.find("--") == -1); // 禁止多重負號
        // req(s.find("-+") == -1);
        // req(s.find("+-") == -1);
        // req(s.find("++") == -1);
        return Expr(s).expr(0);
    }
};
```

6.17 NTT

```cpp
template<typename T,typename VT=std::vector<T> >
struct NTT{
  const T P,G;
  NTT(T p=(1<<23)*7*17+1,T g=3):P(p),G(g){}
  inline unsigned int bit_reverse(unsigned int a,int len){
    a=((a&0x55555555U)<<1)|((a&0xAAAAAAAAU)>>1);
    a=((a&0x33333333U)<<2)|((a&0xCCCCCCCCU)>>2);
    a=((a&0x0F0F0F0FU)<<4)|((a&0xF0F0F0F0U)>>4);
    a=((a&0x00FF00FFU)<<8)|((a&0xFF00FF00U)>>8);
    a=((a&0x0000FFFFU)<<16)|((a&0xFFFF0000U)>>16);
    return a>>(32-len);
  }
  inline T pow_mod(T n,T k,T m){
    T ans=1;
    for(n=(n>=m?n%m:n);k;k>>=1){
      if(k&1)ans=ans*n%m;
      n=n*n%m;
    } return ans;
  }
  inline void ntt(bool is_inv,VT &in,VT &out,int N){
    int bitlen=std::__lg(N);
    for(int i=0;i<N;++i)out[bit_reverse(i,bitlen)]=in[i];
    for(int step=2,id=1;step<=N;step<<=1,++id){
      T wn=pow_mod(G,(P-1)>>id,P),wi=1,u,t;
      const int mh=step>>1;
      for(int i=0;i<mh;++i){
        for(int j=i;j<N;j+=step){
          u = out[j], t = wi*out[j+mh]%P;
          out[j] = u+t;
          out[j+mh] = u-t;
          if(out[j]>=P)out[j]-=P;
          if(out[j+mh]<0)out[j+mh]+=P;
        }
        wi = wi*wn%P;
      }
    }
    if(is_inv){
      for(int i=1;i<N/2;++i)std::swap(out[i],out[N-i]);
      T invn=pow_mod(N,P-2,P);
      for(int i=0;i<N;++i)out[i]=out[i]*invn%P;
    }
  }
};
#endif
```

6.18 EulerFunction

```
// 查詢 phi(x) 亦即比 x 小且與 x 互質的數的數量。
int phi(int x) {
    int r = x;
    for (int p = 2; p * p <= x; p++) {
        if (x % p == 0) {
            while (x % p == 0) x /= p;
            r -= r / p;
        }
    }
    if (x > 1) r -= r / x;
    return r;
}
// 查詢所有 phi(x) ，且 x in [0, n) 。注意右開區間，回傳陣列。
vector<int> phi_in(int n) {
    vector<bool> p(n, 1); vector<int> r(n);
    p[0] = p[1] = 0;
    for (int i = 0; i < n; i++) r[i] = i;
    for (int i = 2; i < n; i++) {
        if (!p[i]) continue;
        r[i]--;
        for (int j = i * 2; j < n; j += i)
            p[j] = 0, r[j] = r[j] / i * (i - 1);
    }
    r[1] = 0;
    return r;
}
```

7 Other

7.1 莫隊算法 _ 區間眾數

```
using namespace std;
const int maxn = 1e6 + 10;
struct query { int id, bk, l, r; };
int arr[maxn], cnt[maxn], d[maxn], n, m, bk, mx;
pair<int,int> ans[maxn];
vector<query> q;
bool cmp(query x,query y) {
    return (x.bk < y.bk || (x.bk == y.bk) && x.r < y.r);
}
void add(int pos) {
    d[cnt[arr[pos]]]--;
    cnt[arr[pos]]++;
    d[cnt[arr[pos]]]++;
    if(d[mx + 1] > 0) mx++;
}
void del(int pos) {
    d[cnt[arr[pos]]]--;
    cnt[arr[pos]]--;
    d[cnt[arr[pos]]]++;
    if(d[mx] == 0) mx--;
}
void mo(int n, int m) {
    sort(q.begin(), q.end(), cmp);
    for(int i = 0, cl = 1, cr = 0; i < m; i++) {
        while(cr < q[i].r) add(++cr);
        while(cl > q[i].l) add(--cl);
        while(cr > q[i].r) del(cr--);
        while(cl < q[i].l) del(cl++);
        ans[q[i].id] = make_pair(mx, d[mx]);
    }
}
int main(){
    cin >> n >> m;
    bk = (int)sqrt(n + 0.5);
    for(int i = 1; i <= n; i++) cin >> arr[i];
    q.resize(m);
    for(int i = 0; i < m; i++) {
        cin >> q[i].l >> q[i].r;
        q[i].id = i,q[i].bk = (q[i].l - 1) / bk;
    }
    mo(n, m);
    for(int i = 0; i < m; i++)
        cout << ans[i].first << ' ' << ans[i].second << '\n';
    return 0;
}
```

7.2 CNF

```
#define MAXN 55
struct CNF{
  int s,x,y;//s->xy | s->x, if y==-1
  int cost;
  CNF(){}
  CNF(int s,int x,int y,int c):s(s),x(x),y(y),cost(c){}
};
int state;//規則數量
map<char,int> rule;//每個字元對應到的規則，小寫字母為終端字符
vector<CNF> cnf;
void init(){
  state=0;
  rule.clear();
  cnf.clear();
}
void add_to_cnf(char s,const string &p,int cost){
  //加入一個s -> <p>的文法，代價為cost
  if(rule.find(s)==rule.end())rule[s]=state++;
  for(auto c:p)if(rule.find(c)==rule.end())rule[c]=state++;
  if(p.size()==1){
    cnf.push_back(CNF(rule[s],rule[p[0]],-1,cost));
  }else{
    int left=rule[s];
    int sz=p.size();
    for(int i=0;i<sz-2;++i){
      cnf.push_back(CNF(left,rule[p[i]],state,0));
      left=state++;
    }
    cnf.push_back(CNF(left,rule[p[sz-2]],rule[p[sz-1]],cost));
  }
}
vector<long long> dp[MAXN][MAXN];
vector<bool> neg_INF[MAXN][MAXN];//如果花費是負的可能會有無限小的情形
void relax(int l,int r,const CNF &c,long long cost,bool neg_c=0){
  if(!neg_INF[l][r][c.s]&&(neg_INF[l][r][c.x]||cost<dp[l][r][c.s])){
    if(neg_c||neg_INF[l][r][c.x]){
      dp[l][r][c.s]=0;
      neg_INF[l][r][c.s]=true;
    }else dp[l][r][c.s]=cost;
  }
}
void bellman(int l,int r,int n){
  for(int k=1;k<=state;++k)
    for(auto c:cnf)
      if(c.y==-1)relax(l,r,c,dp[l][r][c.x]+c.cost,k==n);
}
void cyk(const vector<int> &tok){
  for(int i=0;i<(int)tok.size();++i){
    for(int j=0;j<(int)tok.size();++j){
      dp[i][j]=vector<long long>(state+1,INT_MAX);
      neg_INF[i][j]=vector<bool>(state+1,false);
    }
    dp[i][i][tok[i]]=0;
    bellman(i,i,tok.size());
  }
  for(int r=1;r<(int)tok.size();++r){
    for(int l=r-1;l>=0;--l){
      for(int k=l;k<r;++k)
        for(auto c:cnf)
          if(~c.y)relax(l,r,c,dp[l][k][c.x]+dp[k+1][r][c.y]+c.cost);
      bellman(l,r,tok.size());
    }
  }
}
```

7.3 pbds

```
#include<bits/extc++.h>
using namespace __gnu_pbds;

// hash_table：用法和map差不多 //均攤O(1)
gp_hash_table <string,int> mp;
```

```
mp.find(); mp[]=;
mp.insert(make_pair())

// heaps
priority_queue<int, greater<int>, TAG> Q;
/*
Tag                  | push  | pop    | join   | modify |
pairing_heap_tag     | O(1)  | O(lgN)| O(1)   | O(lgN) |
binary_heap_tag      | O(lgN)| O(lgN)| 慢     | 慢      |
binomial_heap_tag    | O(1)  | O(lgN)| O(lgN)| O(lgN) |
rc_binomial_heap_tag| O(1)  | O(lgN)| O(lgN)| O(lgN) |
thin_heap_tag        | O(1)  | O(lgN)| 慢     | O(lgN) |
*/   //可以用迭代器遍歷
Q.push(x); Q.pop(); Q.top();
Q.join(b); //merge two heap
Q.empty(); Q.size();
Q.modify(it, 6); Q.erase(it);

// k-th
typedef tree<int,null_type,less<int>,rb_tree_tag,
        tree_order_statistics_node_update> set_t;
set_t s; s.insert(12); s.insert(505);
assert(*s.find_by_order(0) == 12);
assert(*s.find_by_order(3) == 505);
assert(s.order_of_key(12) == 0);
assert(s.order_of_key(505) == 1);
s.erase(12);
assert(*s.find_by_order(0) == 505);
assert(s.order_of_key(505) == 0);
```

7.4 BuiltIn

```
//gcc專用
//unsigned int ffs
//unsigned long ffsl
//unsigned long long ffsll
unsigned int x; scanf("%u",&x)
printf("右起第一個1:的位置");
printf("%d\n",__builtin_ffs(x));
printf("左起第一個1之前0的個數:");
printf("%d\n",__builtin_clz(x));
printf("右起第一個1之後0的個數:");
printf("%d\n",__builtin_ctz(x));
printf("1的個數:");
printf("%d\n",__builtin_popcount(x));
printf("1的個數的奇偶性:");
printf("%d\n",__builtin_parity(x));
```

7.5 Reminder

```
Debug List:
1. Long Long !!
2. python3 整數除法 "//"
3. connected / unconnected
4. 範圍看清楚
5. eps夠小嗎！！
6. 浮點數的輸出用 printf("%.5lf", x); 其中.5 代表小數點後第五
   位
```

```
-------------------------------------------------------
Lucas's Theorem
    For non-negative integer n,m and prime P,
    C(m,n) mod P = C(m/M,n/M) * C(m%M,n%M) mod P
    = mult_i ( C(m_i,n_i) )
    where m_i is the i-th digit of m in base P.
-------------------------------------------------------
Kirchhoff's theorem
  A_{ii} = deg(i), A_{ij} = (i,j) \in E ? -1 : 0
  Deleting any one row, one column, and cal the det(A)
-------------------------------------------------------
Nth Catalan recursive function:
C_0 = 1, C_{n+1} = C_n * 2(2n + 1)/(n+2)
-------------------------------------------------------
Mobius Formula
u(n) = 1          , if n = 1
       (-1)^m     , 若 n 無平方數因數，且 n = p1*p2*p3*...*pk
       0          , 若 n 有大於 1 的平方數因數
- Property
1. (積性函數) u(a)u(b) = u(ab)
2. ∑_{d|n} u(d) = [n == 1]
-------------------------------------------------------
Mobius Inversion Formula
if      f(n) = ∑_{d|n} g(d)
then    g(n) = ∑_{d|n} u(n/d)f(d)
             = ∑_{d|n} u(d)f(n/d)
- Application
the number/power of gcd(i, j) = k
- Trick
分塊, O(sqrt(n))
-------------------------------------------------------
Chinese Remainder Theorem (m_i 兩兩互質)
  x = a_1 (mod m_1)
  x = a_2 (mod m_2)
  ....
  x = a_i (mod m_i)
construct a solution:
  Let M = m_1 * m_2 * m_3 * ... * m_n
  Let M_i = M / m_i
  t_i = 1 / M_i
  t_i * M_i = 1 (mod m_i)
  solution x = a_1 * t_1 * M_1 + a_2 * t_2 * M_2 + ... + a_n
       * t_n * M_n + k * M
  = k*M + ∑ a_i * t_i * M_i, k is positive integer.
  under mod M, there is one solution x = ∑ a_i * t_i * M_i
-------------------------------------------------------
Burnside's lemma
|G| * |X/G|  = sum( |X^g| ) where g in G
總方法數: 每一種旋轉下不動點的個數總和 除以 旋轉的方法數
-------------------------------------------------------
Linear Algebra
trace： tr(A) = 對角線和
eigen vector： Ax = cx  =>  (A-cI)x = 0
-------------------------------------------------------
Josephus Problem
f(n,k) = (f(n-1,k)+k)(mod n)
f(1,k) = 0
```

8 String

8.1 Manacher

```
// Longest Palindromic Substring
int manacher (string str) { // O(n)
  int len = (s.length() << 1) | 1;
  vector<int> z(len);
  string s(len, '$');
  for (int i = 1; i < len; i += 2)
    s[i] = str[i >> 1];
  int r = 0, p = 0, ans = 0;
  for (int i = 0, j = p << 1; i < len; i++, j--) {
    z[i] = (i >= r) ? 1 : min(z[j], r - i +1);
    while(0 <= i - z[i] && i + z[i] < len && s[i - z[i]] == s
        [i + z[i]])
      z[i]++;
    if (r < i + z[i] - 1)
      r = i + z[i] - 1, p = i;
    ans = max(ans, z[i]);
  }
  return ans - 1;
}
```

8.2 Edit_Distance

```
// 問從 src 到 dst 的最小 edit distance
// ins 插入一個字元的成本
// del 刪除一個字元的成本
// sst 替換一個字元的成本
ll edd(string& src, string& dst, ll ins, ll del, ll sst) {
    ll dp[src.size() + 1][dst.size() + 1]; // 不用初始化
    for (int i = 0; i <= src.size(); i++) {
        for (int j = 0; j <= dst.size(); j++) {
            if (i == 0) dp[i][j] = ins * j;
            else if (j == 0) dp[i][j] = del * i;
            else if (src[i - 1] == dst[j - 1])
                dp[i][j] = dp[i - 1][j - 1];
            else
                dp[i][j] = min(dp[i][j - 1] + ins,
                            min(dp[i - 1][j] + del,
                             dp[i - 1][j - 1] + sst));
        }
    }
    return dp[src.size()][dst.size()];
}
```

8.3 RollHash

```
// 問 pat 在 str 第一次出現的開頭 index 。-1 表示找不到。
int rollhash(string& str, string& pat) {
    const ll x = 1e6 + 99;  // 隨意大質數，建議 1e6
    const ll m = 1e9 + 9;   // 隨意大質數，建議 1e9
    assert(pat.size());     // pat 不能是空字串
    ll xx = 1, sh = 0;
    for (char c : pat)
```

```cpp
        sh = (sh * x + c) % m, xx = xx * x % m;
    deque<ll> hash = {0};
    int ret = 0;
    for (char c : str) {
        hash.push_back((hash.back() * x + c) % m);
        if (hash.size() <= pat.size()) continue;
        ll h = hash.back() - hash.front() * xx;
        h = (h % m + m) % m;
        if (h == sh) return ret;
        hash.pop_front();
        ret++;
    } return -1;
}
```

8.4 LPS

```cpp
char t[1001];           // 原字串
char s[1001 * 2];       // 穿插特殊字元之後的t
int z[1001 * 2], L, R;  // 源自Gusfield's Algorithm
// 由a往左、由b往右，對稱地作字元比對。
int extend(int a, int b) {
    int i = 0;
    while (a-i>=0 && b+i<N && s[a-i] == s[b+i]) i++;
    return i;
}
void longest_palindromic_substring() {
    int N = strlen(t);
    // t穿插特殊字元，存放到s。
    //  (實際上不會這麼做，都是細算索引值。)
    memset(s, '.', N*2+1);
    for (int i=0; i<N; ++i) s[i*2+1] = t[i];
    N = N*2+1;
    // s[N] = '\0';    // 可做可不做
    // Manacher's Algorithm
    z[0] = 1; L = R = 0;
    for (int i=1; i<N; ++i)  {
        int ii = L - (i - L);   // i的映射位置
        int n = R + 1 - i;
        if (i > R)  {
            z[i] = extend(i, i);
            L = i;
            R = i + z[i] - 1;
        } else if (z[ii] == n)  {
            z[i] = n + extend(i-n, i+n);
            L = i;
            R = i + z[i] - 1;
        } else z[i] = min(z[ii], n);
    }
    // 尋找最長迴文子字串的長度。
    int n = 0, p = 0;
    for (int i=0; i<N; ++i)
        if (z[i] > n)   n = z[p = i];
    // 記得去掉特殊字元。
    cout << "最長迴文子字串的長度是" << (n-1) / 2;
    // 印出最長迴文子字串，記得別印特殊字元。
    for (int i=p-z[p]+1; i<=p+z[p]-1; ++i)
        if (i & 1)  cout << s[i];
}
```

8.5 Trie

```cpp
class Trie {
private:
    struct Node {
        int cnt = 0, sum = 0;
        Node *tr[128] = {};
        ~Node() {
            for (int i = 0; i < 128; i++)
                if (tr[i]) delete tr[i];
        }
    };
    Node *root;
public:
    void insert(char *s) {
        Node *ptr = root;
        for (; *s; s++) {
            if (!ptr->tr[*s]) ptr->tr[*s] = new Node();
            ptr = ptr->tr[*s];
            ptr->sum++;
        }
        ptr->cnt++;
    }
    inline int count(char *s) {
        Node *ptr = find(s);
        return ptr ? ptr->cnt : 0;
    }
    Node *find(char *s) {
        Node *ptr = root;
        for (; *s; s++) {
            if (!ptr->tr[*s]) return 0;
            ptr = ptr->tr[*s];
        } return ptr;
    }
    bool erase(char *s) {
        Node *ptr = find(s);
        if (!ptr) return false;
        int num = ptr->cnt;
        if (!num) return false;
        ptr = root;
        for (; *s; s++) {
            Node *tmp = ptr;
            ptr = ptr->tr[*s];
            ptr->sum -= num;
            if (!ptr->sum) {
                delete ptr;
                tmp->tr[*s] = 0;
                return true;
            }
        }
    }
    Trie() { root = new Node(); }
    ~Trie() { delete root; }
};
```

8.6 Kmp

```cpp
// KMP fail function.
int* kmp_fail(string& s) {
    int* f = new int[s.size()]; int p = f[0] = -1;
    for (int i = 1; s[i]; i++) {
        while (p != -1 && s[p + 1] != s[i]) p = f[p];
        if (s[p + 1] == s[i]) p++;
        f[i] = p;
    }
    return f;
}
// 問 sub 在 str 中出現幾次。
int kmp_count(string& str, string& sub) {
    int* fail = kmp_fail(sub); int p = -1, ret = 0;
    for (int i = 0; i < str.size(); i++) {
        while (p != -1 && sub[p + 1] != str[i]) p = fail[p];
        if (sub[p + 1] == str[i]) p++;
        if (p == sub.size() - 1) p = fail[p], ret++;
    }
    delete[] fail; return ret;
}
// 問 sub 在 str 第一次出現的開頭 index 。-1 表示找不到。
int kmp(string& str, string& sub) {
    int* fail = kmp_fail(sub);
    int i, j = 0;
    while (i < str.size() && j < sub.size()) {
        if (sub[j] == str[i]) i++, j++;
        else if (j == 0) i++;
        else j = fail[j - 1] + 1;
    }
    delete[] fail;
    return j == sub.size() ? (i - j) : -1;
}
```

8.7 AC 自動機

```cpp
template<char L='a',char R='z'>
class ac_automaton{
  struct joe{
    int next[R-L+1], fail, efl, ed, cnt_dp, vis;
    joe():ed(0),cnt_dp(0),vis(0){
      for(int i=0; i<=R-L; i++) next[i]=0;
    }
  };
public:
  std::vector<joe> S;
  std::vector<int> q;
  int qs,qe,vt;
  ac_automaton():S(1),qs(0),qe(0),vt(0){}
  void clear(){
    q.clear();
    S.resize(1);
    for(int i=0; i<=R-L; i++) S[0].next[i] = 0;
    S[0].cnt_dp = S[0].vis = qs = qe = vt = 0;
  }
  void insert(const char *s){
    int o = 0;
    for(int i=0,id; s[i]; i++){
      id = s[i]-L;
      if(!S[o].next[id]){
        S.push_back(joe());
        S[o].next[id] = S.size()-1;
      }
      o = S[o].next[id];
    }
    ++S[o].ed;
  }
  void build_fail(){
    S[0].fail = S[0].efl = -1;
    q.clear();
    q.push_back(0);
    ++qe;
    while(qs!=qe){
      int pa = q[qs++], id, t;
      for(int i=0;i<=R-L;i++){
        t = S[pa].next[i];
        if(!t)continue;
        id = S[pa].fail;
        while(~id && !S[id].next[i]) id = S[id].fail;
        S[t].fail = ~id ? S[id].next[i] : 0;
        S[t].efl = S[S[t].fail].ed ? S[t].fail : S[S[t].fail].efl;
        q.push_back(t);
        ++qe;
      }
    }
  }
  /*DP出每個前綴在字串s出現的次數並傳回所有字串被s匹配成功的次數O(N+M)*/
  int match_0(const char *s){
    int ans = 0, id, p = 0, i;
    for(i=0; s[i]; i++){
      id = s[i]-L;
      while(!S[p].next[id] && p) p = S[p].fail;
      if(!S[p].next[id])continue;
      p = S[p].next[id];
      ++S[p].cnt_dp;/*匹配成功則它所有後綴都可以被匹配(DP計算)*/
    }
    for(i=qe-1; i>=0; --i){
      ans += S[q[i]].cnt_dp * S[q[i]].ed;
      if(~S[q[i]].fail) S[S[q[i]].fail].cnt_dp += S[q[i]].cnt_dp;
    }
    return ans;
  }
  /*多串匹配走efl邊並傳回所有字串被s匹配成功的次數O(N*M^1.5)*/
  int match_1(const char *s)const{
    int ans = 0, id, p = 0, t;
    for(int i=0; s[i]; i++){
      id = s[i]-L;
      while(!S[p].next[id] && p) p = S[p].fail;
      if(!S[p].next[id])continue;
      p = S[p].next[id];
      if(S[p].ed) ans += S[p].ed;
      for(t=S[p].efl; ~t; t=S[t].efl){
        ans += S[t].ed;/*因為都走efl邊所以保證匹配成功*/
      }
    }
    return ans;
  }
  /*枚舉(s的子字串□A)的所有相異字串各恰一次並傳回次數O(N*M^(1/3))*/
  int match_2(const char *s){
    int ans=0, id, p=0, t;
    ++vt;
    /*把戳記vt+=1，只要vt沒溢位，所有S[p].vis==vt就會變成false
    這種利用vt的方法可以O(1)歸零vis陣列*/
    for(int i=0; s[i]; i++){
      id = s[i]-L;
      while(!S[p].next[id]&&p)p = S[p].fail;
      if(!S[p].next[id])continue;
      p = S[p].next[id];
      if(S[p].ed && S[p].vis!=vt){
        S[p].vis = vt;
        ans += S[p].ed;
      }
      for(t=S[p].efl; ~t && S[t].vis!=vt; t=S[t].efl){
        S[t].vis = vt;
        ans += S[t].ed;/*因為都走efl邊所以保證匹配成功*/
      }
    }
    return ans;
  }
  /*把AC自動機變成真的自動機*/
  void evolution(){
    for(qs=1; qs!=qe;){
      int p = q[qs++];
      for(int i=0; i<=R-L; i++)
        if(S[p].next[i]==0) S[p].next[i] = S[S[p].fail].next[i];
    }
  }
};
```

8.8 BWT

```cpp
const int N = 8;             // 字串長度
int s[N+N+1] = "suffixes";  // 字串，後面預留一倍空間。
int sa[N];                   // 後綴陣列
int pivot;
int cmp(const void* i, const void* j) {
    return strncmp(s+*(int*)i, s+*(int*)j, N);
}
// 此處便宜行事，採用 O(N²logN) 的後綴陣列演算法。
void BWT() {
    strncpy(s + N, s, N);
    for (int i=0; i<N; ++i) sa[i] = i;
    qsort(sa, N, sizeof(int), cmp);
    // 當輸入字串的所有字元都相同，必須當作特例處理。
    // 或者改用stable sort。
    for (int i=0; i<N; ++i)
        cout << s[(sa[i] + N-1) % N];
    for (int i=0; i<N; ++i)
        if (sa[i] == 0) {
            pivot = i;
            break;
        }
}
// Inverse BWT
const int N = 8;             // 字串長度
char t[N+1] = "xuffessi";   // 字串
int pivot;
int next[N];
void IBWT() {
    vector<int> index[256];
    for (int i=0; i<N; ++i)
        index[t[i]].push_back(i);
    for (int i=0, n=0; i<256; ++i)
        for (int j=0; j<index[i].size(); ++j)
            next[n++] = index[i][j];
    int p = pivot;
    for (int i=0; i<N; ++i)
        cout << t[p = next[p]];
}
```

8.9 Z

```cpp
void z_build(string &s, vector<int> &z) {
    int bst = z[0] = 0;
    for (int i = 1; s[i]; i++) {
        if (z[bst] + bst < i) z[i] = 0;
        else z[i] = min(z[bst] + bst - i, z[i - bst]);
        while (s[z[i]] == s[i + z[i]]) z[i]++;
        if (z[i] + i > z[bst] + bst) bst = i;
    }
}
// Queries how many times s appears in t
int z_match(string &s, string &t) {
    int ans = 0;
    int lens = s.length(), lent = t.length();
    vector<int> z(lens + lent + 1);
    string st = s + "$" + t;
    z_build(st, z);
    for (int i = lens + 1; i <= lens + lent; i++)
        if (z[i] == lens) ans++;
    return ans;
}
```

8.10 suffix_array

```cpp
// qsort suffix array, 0-based only, O(T * log^2 T) 略慢但是好寫
const int N = ? ;  // 字串最大長度
namespace SA {
int sa[N], t0[N], t1[N];

struct CMP {
    int *r, n, X;
    bool operator()(int i, int j) {
        if (r[i] != r[j]) return r[i] < r[j];
        int a = (i + n < X) ? r[i + n] : -1;
        int b = (j + n < X) ? r[j + n] : -1;
        return a < b;
    }
};

// str = 字串，可為 vector 或 string 或 char[] 等
// n = 字串長(含$)
// 結果存在 SA::sa
template <typename T>
void build(const T &str) {
    int n = str.size();
    int *a = t0, *aa = t1;
    for (int i = 0; i < n; i++) sa[i] = i, a[i] = str[i];
    for (int m = 2; m <= n; m *= 2) {
        CMP cmp = {a, m / 2, n};
        sort(sa, sa + n, cmp);
        int r = 0;
        aa[sa[0]] = r;
        for (int i = 1; i < n; i++) {
            if (cmp(sa[i - 1], sa[i])) r++;
            aa[sa[i]] = r;
        }
        swap(a, aa);
        if (r == n - 1) break;
    }
}
}  // namespace SA


// 卦長的 IS suffix array ，0-based only
// N = 字串最大長度 , A = 最大字元 ascii
// 複雜度 O(N+A)
const int N = ?, A = ?;
namespace SA {

#define pushS(x) sa[--b[s[x]]] = x
#define pushL(x) sa[b[s[x]]++] = x
#define induce_sort(v)                                    \
    {                                                     \
        fill_n(sa, n, 0);                                 \
        copy_n(bb, A, b);                                 \
        for (i = n1 - 1; ~i; --i) pushS(v[i]);            \
        copy_n(bb, A - 1, b + 1);                         \
        for (i = 0; i < n; ++i)                           \
            if (sa[i] && !t[sa[i] - 1]) pushL(sa[i] - 1); \
        copy_n(bb, A, b);                                 \
        for (i = n - 1; ~i; --i)                          \
            if (sa[i] && t[sa[i] - 1]) pushS(sa[i] - 1);  \
    }

template <typename T>
void sais(const T s, int n, int *sa, int *bb, int *p, bool *t, int A) {
    int *r = p + n, *s1 = p + n / 2, *b = bb + A;
    int n1 = 0, i, j, x = t[n - 1] = 1, y = r[0] = -1, cnt = -1;
    for (i = n - 2; ~i; --i) t[i] = (s[i] == s[i + 1] ? t[i + 1] : s[i] < s[i + 1]);
    for (i = 1; i < n; ++i) r[i] = t[i] && !t[i - 1] ? (p[n1] = i, n1++) : -1;
    fill_n(bb, A, 0);
    for (i = 0; i < n; ++i) ++bb[s[i]];
    for (i = 1; i < A; ++i) bb[i] += bb[i - 1];
    induce_sort(p);
    for (i = 0; i < n; ++i)
        if (~(x = r[sa[i]]))
            j = y < 0 || memcmp(s + p[x], s + p[y], (p[x + 1] - p[x]) * sizeof(s[0])), s1[y = x] = cnt += j;
    if (cnt + 1 < n1)
        sais(s1, n1, sa, b, r, t + n, cnt + 1);
    else
        for (i = 0; i < n1; ++i) sa[s1[i]] = i;
    for (i = 0; i < n1; ++i) s1[i] = p[sa[i]];
    induce_sort(s1);
}

int sa[N];
int b[N + A], p[N * 2];
bool t[N * 2];

// 計算 suffix array ，字串須為 char[] 或 int[]， 不可為 string 或 vector
// s = 字串
// n = 字串長度(含$)
// 結果存在 SA::sa
template <typename T>
void build(const T s, int n) { sais(s, n, sa, b, p, t, A); }

}  // namespace SA
```

9 Surroudings

9.1 bashrc

```bash
oj() {
  ext=${1##*.}            #空格敏感
  filename=${1##*/}       #空格敏感
  filename=${filename%.*} #空格敏感
  case $ext in
    cpp ) g++ -o "/tmp/$filename" "$1" && "/tmp/$filename" ;; #空格不敏感
    py  ) python3 "$1" ;;                                     #空格不敏感
  esac
}
```

NCTU-Pusheen Codebook

Contents