Contents

1 Basic 1
  1.1 debug list 1

2 Geometry 1
  2.1 2D Point Template 1
  2.2 外心 Circumcentre 1
  2.3 Convex Hull 1
  2.4 半平面交 1
  2.5 圓交 2
  2.6 線段交 2
  2.7 Smallest Covering Circle 2

3 Mathmatics 2
  3.1 ax+by=gcd(a,b) 2
  3.2 Inverse 2
  3.3 LinearPrime 2
  3.4 數論基本工具 2
  3.5 Theorem 2

4 Graph 3
  4.1 BCC 3
  4.2 Dijkstra 3
  4.3 Strongly Connected Component(SCC) 3
  4.4 DominatorTree 3
  4.5 Manhattan MST 4
  4.6 Hungarian 5
  4.7 KM 5
  4.8 Theorm - Matching 6
  4.9 Maximum General Matching 6
  4.10 Minimum General Weighted Matching 6
  4.11 Maximum Clique 7
  4.12 Steiner Tree 7
  4.13 最小平均環 8
  4.14 Tarjan 8

5 Data Structure 8
  5.1 Sparse Table 8
  5.2 Segment Tree 9
  5.3 Djs 9
  5.4 Binary Indexed Tree 9

6 String 10
  6.1 KMP 10

7 Dark Code 10
  7.1 PBDS 10

8 Search 10
  8.1 LIS 10
  8.2 Merge sort 10
  8.3 離散化 10

9 Others 10
  9.1 數位統計 10
  9.2 1D/1D dp 優化 11
  9.3 Theorm - DP optimization 11
  9.4 Mo′ s algorithm 11

10 Persistence 11

1 Basic

1.1 debug list

```
模板要記得 init
priority_queue 要清空
把邊界條件都加入測資
邊界條件 (過程溢位，題目數據範圍)，會不會爆 long long
是否讀錯題目，想不到時可以自己讀一次題目
環狀or凸包問題一定要每種都算n次
比較容易有問題的地方換人寫
注意公式有沒有推錯或抄錯
精度誤差 sqrt(大大的東西) + EPS
測試 %lld or %I64d
喇分 random_suffle 隨機演算法
用long long int記得要算MLE
```

2 Geometry

2.1 2D Point Template

```cpp
typedef double Double;
struct Point {
  Double x,y;

  bool operator < (const Point &b)const{
    //return tie(x,y) < tie(b.x,b.y);
    //return atan2(y,x) < atan2(b.y,b.x);
    assert(0 && "choose compare");
  }
  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 Double &d)const{
    return Point(d*x,d*y);
  }
  Double operator * (const Point &b)const{
    return x*b.x + y*b.y;
  }
  Double operator % (const Point &b)const{
    return x*b.y - y*b.x;
  }
  friend Double abs2(const Point &p){
    return p.x*p.x + p.y*p.y;
  }
  friend Double abs(const Point &p){
    return sqrt( abs2(p) );
  }
};
typedef Point Vector;

struct Line{
  Point P; Vector v;
  bool operator < (const Line &b)const{
    return atan2(v.y,v.x) < atan2(b.v.y,b.v.x);
  }
};
```

2.2 外心 Circumcentre

```cpp
#include "2Dpoint.cpp"

Point circumcentre(Point &p0, Point &p1, Point &p2){
  Point a = p1-p0;
  Point b = p2-p0;
  Double c1 = abs2(a)*0.5;
  Double c2 = abs2(b)*0.5;
  Double d = a % b;
  Double x = p0.x + ( c1*b.y - c2*a.y ) / d;
  Double y = p0.y + ( c2*a.x - c1*b.x ) / d;
  return {x,y};
}
```

2.3 Convex Hull

```cpp
#include "2Dpoint.cpp"

// retunr H, 第一個點會在 H 出現兩次
void ConvexHull(vector<Point> &P, vector<Point> &H){
    int n = P.size(), m=0;
    sort(P.begin(),P.end());
    H.clear();

    for (int i=0; i<n; i++){
        while (m>=2 && (P[i]-H[m-2]) % (H[m-1]-H[m-2])
            <0)H.pop_back(), m--;
        H.push_back(P[i]), m++;
    }

    for (int i=n-2; i>=0; i--){
        while (m>=2 && (P[i]-H[m-2]) % (H[m-1]-H[m-2])
            <0)H.pop_back(), m--;
        H.push_back(P[i]), m++;
    }
}
```

2.4 半平面交

```
bool OnLeft(const Line& L,const Point& p){
  return Cross(L.v,p-L.P)>0;
}
Point GetIntersection(Line a,Line b){
  Vector u = a.P-b.P;
  Double t = Cross(b.v,u)/Cross(a.v,b.v);
  return a.P + a.v*t;
}
int HalfplaneIntersection(Line* L,int n,Point* poly){
  sort(L,L+n);

  int first,last;
  Point *p = new Point[n];
  Line *q = new Line[n];
  q[first=last=0] = L[0];
  for(int i=1;i<n;i++){
    while(first < last && !OnLeft(L[i],p[last-1])) last
        --;
    while(first < last && !OnLeft(L[i],p[first])) first
        ++;
    q[++last]=L[i];
    if(fabs(Cross(q[last].v,q[last-1].v))<EPS){
      last--;
      if(OnLeft(q[last],L[i].P)) q[last]=L[i];
    }
    if(first < last) p[last-1]=GetIntersection(q[last
        -1],q[last]);
  }
  while(first<last && !OnLeft(q[first],p[last-1])) last
      --;
  if(last-first<=1) return 0;
  p[last]=GetIntersection(q[last],q[first]);

  int m=0;
  for(int i=first;i<=last;i++) poly[m++]=p[i];
  return m;
}
```

2.5 圓交

```
vector<Double> interCircle(Double o1, Double r1, Double
     o2, Double r2) {
  Double d2 = abs2(o1 - o2);
  Double d = sqrt(d2);
  if (d < fabs(r1-r2) || r1+r2 < d) return {};
  Double u = 0.5*(o1+o2) + ((r2*r2-r1*r1)/(2.0*d2))*(o1
      -o2);
  Double A = sqrt((r1+r2+d) * (r1-r2+d) * (r1+r2-d) *
      (-r1+r2+d));
  Double v = A / (2.0*d2) * Double(o1.S-o2.S, -o1.F+o2.
      F);
  return {u+v, u-v};
}
```

2.6 線段交

```
Point interPnt(Point p1, Point p2, Point q1, Point q2,
    bool &res){
  Double f1 = cross(p2, q1, p1);
  Double f2 = -cross(p2, q2, p1);
  Double f = (f1 + f2);

  if(fabs(f) < EPS) {
    res = false;
    return {};
    }

  res = true;
  return (f2 / f) * q1 + (f1 / f) * q2;
}
```

2.7 Smallest Covering Circle

```
#include "circumcentre.cpp"
pair<Point,Double> SmallestCircle(int n, Point _p[]){
  Point *p = new Point[n];
  memcpy(p,_p,sizeof(Point)*n);
  random_shuffle(p,p+n);

  Double r2=0;
  Point cen;
  for (int i=0; i<n; i++){
    if ( abs2(cen-p[i]) <= r2)continue;
    cen = p[i], r2=0;
    for (int j=0; j<i; j++){
      if ( abs2(cen-p[j]) <= r2)continue;
      cen = (p[i]+p[j])*0.5;
      r2 = abs2(cen-p[i]);
      for (int k=0; k<j; k++){
        if ( abs2(cen-p[k]) <= r2)continue;
        cen = circumcentre(p[i],p[j],p[k]);
        r2 = abs2(cen-p[k]);
      }
    }
  }

  delete[] p;
  return {cen,r2};
}
// auto res = SmallestCircle(,);
```

3 Mathmatics

3.1 ax+by=gcd(a,b)

```
typedef pair<int, int> pii;
pii extgcd(int a, int b){
  if(b == 0) return make_pair(1, 0);
  else{
    int p = a / b;
    pii q = extgcd(b, a % b);
    return make_pair(q.second, q.first - q.second * p);
  }
}
```

3.2 Inverse

```
int inverse[100000];
void invTable(int b, int p) {
  inverse[1] = 1;
  for( int i = 2; i <= b; i++ ) {
    inverse[i] = (long long)inverse[p%i] * (p-p/i) % p;
  }
}

int inv(int b, int p) {
  return b == 1 ? 1 : ((long long)inv(p % b, p) * (p-p/
      b) % p);
}
```

3.3 LinearPrime

```
const int MAXP = 100; //max prime
vector<int> P;  // primes
void build_prime(){
  static bitset<MAXP> ok;
  int np=0;
  for (int i=2; i<MAXP; i++){
    if (ok[i]==0)P.push_back(i), np++;
    for (int j=0; j<np && i*P[j]<MAXP; j++){
      ok[ i*P[j] ] = 1;
      if ( i%P[j]==0 )break;
    }
  }
}
```

3.4 數論基本工具

3.5 Theorem

```
/*
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.
-------------------------------------------------------
Pick's Theorem
  A = i + b/2 - 1
-------------------------------------------------------
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
*/
```

4 Graph

4.1 BCC

```
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;
```

4.2 Dijkstra

```
typedef struct Edge{
    int v; long long len;
    bool operator > (const Edge &b)const { return len>b.len; }
} State;

const long long INF = 1LL<<60;

void Dijkstra(int n, vector<Edge> G[], long long d[],
    int s, int t=-1){
    static priority_queue<State, vector<State>, greater
        <State> > pq;
    while ( pq.size() )pq.pop();
    for (int i=1; i<=n; i++)d[i]=INF;
    d[s]=0; pq.push( (State){s,d[s]} );
    while ( pq.size() ){
        auto x = pq.top(); pq.pop();
        int u = x.v;
        if (d[u]<x.len)continue;
        if (u==t)return;
        for (auto &e:G[u]){
            if (d[e.v] > d[u]+e.len){
                d[e.v] = d[u]+e.len;
                pq.push( (State) {e.v,d[e.v]} );
            }
        }
    }
}
```

4.3 Strongly Connected Component(SCC)

4.4 DominatorTree

```
// PEC VER

// idom[n] is the unique node that strictly dominates n
     but does
// not strictly dominate any other node that strictly
    dominates n.
// idom[n] = 0 if n is entry or the entry cannot reach
    n.
struct DominatorTree{
  static const int MAXN = 200010;
  int n,s;
  vector<int> g[MAXN],pred[MAXN];
  vector<int> cov[MAXN];
  int dfn[MAXN],nfd[MAXN],ts;
  int par[MAXN];
  int sdom[MAXN],idom[MAXN];
  int mom[MAXN],mn[MAXN];

  inline bool cmp(int u,int v) { return dfn[u] < dfn[v
      ]; }

  int eval(int u) {
    if(mom[u] == u) return u;
    int res = eval(mom[u]);
    if(cmp(sdom[mn[mom[u]]],sdom[mn[u]]))
      mn[u] = mn[mom[u]];
    return mom[u] = res;
  }

  void init(int _n, int _s) {
    n = _n;
    s = _s;
    REP1(i,1,n) {
      g[i].clear();
      pred[i].clear();
      idom[i] = 0;
    }
  }
  void add_edge(int u, int v) {
    g[u].push_back(v);
    pred[v].push_back(u);
  }
  void DFS(int u) {
    ts++;
    dfn[u] = ts;
    nfd[ts] = u;
    for(int v:g[u]) if(dfn[v] == 0) {
      par[v] = u;
      DFS(v);
    }
  }
  void build() {
    ts = 0;
    REP1(i,1,n) {
      dfn[i] = nfd[i] = 0;
      cov[i].clear();
      mom[i] = mn[i] = sdom[i] = i;
    }
    DFS(s);
    for (int i=ts; i>=2; i--) {
      int u = nfd[i];
      if(u == 0) continue ;
      for(int v:pred[u]) if(dfn[v]) {
        eval(v);
        if(cmp(sdom[mn[v]],sdom[u])) sdom[u] = sdom[mn[
            v]];
      }
      cov[sdom[u]].push_back(u);
      mom[u] = par[u];
      for(int w:cov[par[u]]) {
        eval(w);
        if(cmp(sdom[mn[w]],par[u])) idom[w] = mn[w];
        else idom[w] = par[u];
      }
      cov[par[u]].clear();
    }
    REP1(i,2,ts) {
      int u = nfd[i];
      if(u == 0) continue ;
      if(idom[u] != sdom[u]) idom[u] = idom[idom[u]];
    }
  }
}dom;
```

```
#define MXN 100005
#define PB push_back
#define FZ(s) memset(s,0,sizeof(s))

struct Scc{
int n, nScc, vst[MXN], bln[MXN];
vector<int> E[MXN], rE[MXN], vec;
void init(int _n){
  n = _n;
  for (int i=0; i<MXN; i++){
    E[i].clear();
    rE[i].clear();
  }
}
void add_edge(int u, int v){
  E[u].PB(v);
  rE[v].PB(u);
}
void DFS(int u){
  vst[u]=1;
  for (auto v : E[u])
    if (!vst[v]) DFS(v);
  vec.PB(u);
}
void rDFS(int u){
  vst[u] = 1;
  bln[u] = nScc;
  for (auto v : rE[u])
    if (!vst[v]) rDFS(v);
}
void solve(){
  nScc = 0;
  vec.clear();
  FZ(vst);
  for (int i=0; i<n; i++)
    if (!vst[i]) DFS(i);
  reverse(vec.begin(),vec.end());
  FZ(vst);
  for (auto v : vec){
    if (!vst[v]){
      rDFS(v);
      nScc++;
    }
  }
}
};
```

4.5 Manhattan MST

```
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 100005;
const int OFFSET = 2000; // y-x may < 0, offset it, if y-x too large, please write a unique function
const int INF = 0xFFFFFFF;
int n;
int x[MAXN], y[MAXN], p[MAXN];

typedef pair<int, int> pii;
pii bit[MAXN]; // [ val, pos ]

struct P {
    int x, y, id;
    bool operator<(const P&b ) const {
        if ( x == b.x ) return y > b.y;
        else return x > b.x;
    }
};
vector<P> op;

struct E {
    int x, y, cost;
    bool operator<(const E&b ) const {
        return cost < b.cost;
    }
};
vector<E> edges;

int find(int x) {
    return p[x] == x ? x : p[x] = find(p[x]);
}

void update(int i, int v, int p) {
    while ( i ) {
        if ( bit[i].first > v ) bit[i] = {v, p};
        i -= i & (-i);
    }
}

pii query(int i) {
    pii res = {INF, INF};
    while ( i < MAXN ) {
        if ( bit[i].first < res.first ) res = {bit[i].first, bit[i].second};
        i += i & (-i);
    }
    return res;
}

void input() {
    cin >> n;
    for ( int i = 0 ; i < n ; i++ ) cin >> x[i] >> y[i], op.push_back((P) {x[i], y[i], i});
}

void mst() {
    for ( int i = 0 ; i < MAXN ; i++ ) p[i] = i;
    int res = 0;
    sort(edges.begin(), edges.end());
    for ( auto e : edges ) {
        int x = find(e.x), y = find(e.y);
        if ( x != y ) {
            p[x] = y;
            res += e.cost;
        }
    }
    cout << res << endl;
}

void construct() {
    sort(op.begin(), op.end());
    for ( int i = 0 ; i < n ; i++ ) {
        pii q = query(op[i].y - op[i].x + OFFSET);
        update(op[i].y - op[i].x + OFFSET, op[i].x + op[i].y, op[i].id);
        if ( q.first == INF ) continue;
        edges.push_back((E) {op[i].id, q.second, abs(x[op[i].id]-x[q.second]) + abs(y[op[i].id]-y[q.second]) });
    }
}

void solve() {

    // [45 ~ 90 deg]
    for ( int i = 0 ; i < MAXN ; i++ ) bit[i] = {INF, INF};
    construct();

    // [0 ~ 45 deg]
    for ( int i = 0 ; i < MAXN ; i++ ) bit[i] = {INF, INF};
    for ( int i = 0 ; i < n ; i++ ) swap(op[i].x, op[i].y);
    construct();
    for ( int i = 0 ; i < n ; i++ ) swap(op[i].x, op[i].y);

    // [-90 ~ -45 deg]
    for ( int i = 0 ; i < MAXN ; i++ ) bit[i] = {INF, INF};
    for ( int i = 0 ; i < n ; i++ ) op[i].y *= -1;
    construct();

    // [-45 ~ 0 deg]
    for ( int i = 0 ; i < MAXN ; i++ ) bit[i] = {INF, INF};
    for ( int i = 0 ; i < n ; i++ ) swap(op[i].x, op[i].y);
    construct();

    // mst
    mst();

}

int main () {
    input();
    solve();
    return 0;
}
```

4.6 Hungarian

```
// Maximum Cardinality Bipartite Matching

struct Graph {
    static const int MAXN = 5005;
    vector<int> G[MAXN];
    int n;
    int match[MAXN]; // Matching Result
    int vis[MAXN];

    void init(int _n) {
        n = _n;
        for ( int i = 0 ; i < n ; i++ ) G[i].clear();
    }

    bool dfs(int u) {
        for ( auto v:G[u] ) {
            if (!vis[v]) {
                vis[v] = true;
                if (match[v] == -1 || dfs(match[v])) {
                    match[v] = u;
                    match[u] = v;
                    return true;
                }
            }
        }
        return false;
    }

    int solve() {
        int res = 0;
        memset(match, -1, sizeof(match));
        for (int i = 0; i < n; i++) {
            if (match[i] == -1) {
                memset(vis, 0, sizeof(vis));
                if (dfs(i)) res += 1;
            }
        }
        return res;
    }
} graph;
```

4.7 KM

```
Detect non-perfect-matching:
1. set all edge[i][j] as INF
2. if solve() >= INF, it is not perfectmatching.
-------------------------------------------------------
// Maximum Weight Perfect Bipartite Matching
// allow negative weight!

typedef long long Int;
struct KM {
    static const int MAXN = 1050;
    static const int INF = 1LL<<60;
    int n, match[MAXN], vx[MAXN], vy[MAXN];
    Int edge[MAXN][MAXN], lx[MAXN], ly[MAXN], slack[
        MAXN];
    void init(int _n){
        n = _n;
        for ( int i = 0 ; i < n ; i++ )
            for ( int j = 0; j < n ; j++ )
                edge[i][j] = 0;
    }
    void add_edge(int x, int y, Int w){
        edge[x][y] = w;
    }
    bool DFS(int x){
        vx[x] = 1;
        for ( int y = 0 ; y < n ; y++ ) {
            if ( vy[y] ) continue;
            if ( lx[x] + ly[y] > edge[x][y] ) {
                slack[y] = min(slack[y], lx[x] + ly[y]
                    - edge[x][y]);
            } else {
                vy[y] = 1;
                if ( match[y] == -1 || DFS(match[y]) ){
                    match[y] = x;
                    return true;
                }
            }
        }
        return false;
    }
    Int solve() {
        fill(match, match + n, -1);
        fill(lx, lx + n, -INF);
        fill(ly, ly + n, 0);
        for ( int i = 0; i < n; i++ )
            for ( int j = 0; j < n; j++ )
                lx[i] = max(lx[i], edge[i][j]);
        for ( int i = 0 ; i < n; i++ ) {
            fill(slack, slack + n, INF);
            while (true){
                fill(vx, vx + n, 0);
                fill(vy, vy + n, 0);
                if ( DFS(i) ) break;
                Int d = INF;
                for ( int j = 0 ; j < n ; j++ )
                    if ( !vy[j] ) d = min(d, slack[j]);
                for ( int j = 0 ; j < n ; j++ ) {
                    if (vx[j]) lx[j] -= d;
                    if (vy[j]) ly[j] += d;
                    else slack[j] -= d;
                }
            }
        }
        Int res = 0;
        for ( int i = 0 ; i < n ; i++ ) {
            res += edge[ match[i] ][i];
        }
        return res;
    }
} graph;
```

4.8 Theorm - Matching

最大匹配 + 最小邊覆蓋 = V
最大獨立集 + 最小點覆蓋 = V
最大匹配 = 最小點覆蓋
最小路徑覆蓋數 = V - 最大匹配數

4.9 Maximum General Matching

```
// Maximum Cardinality Matching

struct Graph {
  vector<int> G[MAXN];
  int pa[MAXN], match[MAXN], st[MAXN], S[MAXN], vis[
      MAXN];
  int t, n;

  void init(int _n) {
    n = _n;
    for ( int i = 1 ; i <= n ; i++ ) G[i].clear();
  }
  void add_edge(int u, int v) {
    G[u].push_back(v);
    G[v].push_back(u);
  }
  int lca(int u, int v){
    for ( ++t ; ; swap(u, v) ) {
      if ( u == 0 ) continue;
      if ( vis[u] == t ) return u;
      vis[u] = t;
      u = st[ pa[ match[u] ] ];
    }
  }
  void flower(int u, int v, int l, queue<int> &q) {
    while ( st[u] != l ) {
      pa[u] = v;
      if ( S[ v = match[u] ] == 1 ) {
        q.push(v);
        S[v] = 0;
      }
      st[u] = st[v] = l;
      u = pa[v];
    }
  }
  bool bfs(int u){
    for ( int i = 1 ; i <= n ; i++ ) st[i] = i;
    memset(S, -1, sizeof(S));
    queue<int>q;
    q.push(u);
    S[u] = 0;
    while ( !q.empty() ) {
      u = q.front(); q.pop();
```

```
   for ( int i = 0 ; i < (int)G[u].size(); i++) {
    int v = G[u][i];
    if ( S[v] == -1 ) {
     pa[v] = u;
     S[v] = 1;
     if ( !match[v] ) {
      for ( int lst ; u ; v = lst, u = pa[v] ) {
       lst = match[u];
       match[u] = v;
       match[v] = u;
      }
      return 1;
     }
     q.push(match[v]);
     S[ match[v] ] = 0;
    } else if ( !S[v] && st[v] != st[u] ) {
     int l = lca(st[v], st[u]);
     flower(v, u, l, q);
     flower(u, v, l, q);
    }
   }
  }
  return 0;
 }
 int solve(){
  memset(pa, 0, sizeof(pa));
  memset(match, 0, sizeof(match));
  int ans = 0;
  for ( int i = 1 ; i <= n ; i++ )
   if ( !match[i] && bfs(i) ) ans++;
  return ans;
 }
} graph;
```

4.10 Minimum General Weighted Matching

```
// Minimum Weight Perfect Matching (Perfect Match)

struct Graph {
  static const int MAXN = 105;
  int n, e[MAXN][MAXN];
  int match[MAXN], d[MAXN], onstk[MAXN];
  vector<int> stk;
  void init(int _n) {
    n = _n;
    for( int i = 0 ; i < n ; i ++ )
      for( int j = 0 ; j < n ; j ++ )
        e[i][j] = 0;
  }
  void add_edge(int u, int v, int w) {
    e[u][v] = e[v][u] = w;
  }
  bool SPFA(int u){
    if (onstk[u]) return true;
    stk.push_back(u);
    onstk[u] = 1;
    for ( int v = 0 ; v < n ; v++ ) {
      if (u != v && match[u] != v && !onstk[v] )
        {
        int m = match[v];
        if ( d[m] > d[u] - e[v][m] + e[u][v] )
          {
          d[m] = d[u] - e[v][m] + e[u][v];
          onstk[v] = 1;
          stk.push_back(v);
          if (SPFA(m)) return true;
          stk.pop_back();
          onstk[v] = 0;
        }
      }
    }
    onstk[u] = 0;
    stk.pop_back();
    return false;
  }
  int solve() {
    for ( int i = 0 ; i < n ; i += 2 ) {
      match[i] = i+1;
      match[i+1] = i;
    }
    while (true){
      int found = 0;
      for ( int i = 0 ; i < n ; i++ )
        onstk[ i ] = d[ i ] = 0;
      for ( int i = 0 ; i < n ; i++ ) {
        stk.clear();
        if ( !onstk[i] && SPFA(i) ) {
          found = 1;
          while ( stk.size() >= 2 ) {
            int u = stk.back(); stk.
              pop_back();
            int v = stk.back(); stk.
              pop_back();
            match[u] = v;
            match[v] = u;
          }
        }
      }
      if (!found) break;
    }
    int ret = 0;
    for ( int i = 0 ; i < n ; i++ )
      ret += e[i][match[i]];
    ret /= 2;
    return ret;
  }
} graph;
```

4.11 Maximum Clique

```
const int MAXN = 105;
int best;
int m ,n;
int num[MAXN];
// int x[MAXN];
int path[MAXN];
int g[MAXN][MAXN];

bool dfs( int *adj, int total, int cnt ){
  int i, j, k;
  int t[MAXN];
  if( total == 0 ){
    if( best < cnt ){
      // for( i = 0; i < cnt; i++) path[i] = x[i
        ];
      best = cnt; return true;
    }
    return false;
  }
  for( i = 0; i < total; i++){
    if( cnt+(total-i) <= best ) return false;
    if( cnt+num[adj[i]] <= best ) return false;
    // x[cnt] = adj[i];
    for( k = 0, j = i+1; j < total; j++ )
      if( g[ adj[i] ][ adj[j] ] )
        t[ k++ ] = adj[j];
        if( dfs( t, k, cnt+1 ) ) return true;
  } return false;
}
int MaximumClique(){
  int i, j, k;
  int adj[MAXN];
  if( n <= 0 ) return 0;
  best = 0;
  for( i = n-1; i >= 0; i-- ){
    // x[0] = i;
    for( k = 0, j = i+1; j < n; j++ )
      if( g[i][j] ) adj[k++] = j;
    dfs( adj, k, 1 );
    num[i] = best;
  }
  return best;
}
```

4.12 Steiner Tree

```
// Minimum Steiner Tree
// O(V 3^T + V^2 2^T)
struct SteinerTree{
#define V 33
#define T 8
#define INF 1023456789
  int n , dst[V][V] , dp[1 << T][V] , tdst[V];
  void init( int _n ){
    n = _n;
    for( int i = 0 ; i < n ; i ++ ){
      for( int j = 0 ; j < n ; j ++ )
        dst[ i ][ j ] = INF;
      dst[ i ][ i ] = 0;
    }
  }
  void add_edge( int ui , int vi , int wi ){
    dst[ ui ][ vi ] = min( dst[ ui ][ vi ] , wi );
    dst[ vi ][ ui ] = min( dst[ vi ][ ui ] , wi );
  }
  void shortest_path(){
    for( int k = 0 ; k < n ; k ++ )
      for( int i = 0 ; i < n ; i ++ )
        for( int j = 0 ; j < n ; j ++ )
          dst[ i ][ j ] = min( dst[ i ][ j ],
                dst[ i ][ k ] + dst[ k ][ j ] );
  }
  int solve( const vector<int>& ter ){
    int t = (int)ter.size();
    for( int i = 0 ; i < ( 1 << t ) ; i ++ )
      for( int j = 0 ; j < n ; j ++ )
        dp[ i ][ j ] = INF;
    for( int i = 0 ; i < n ; i ++ )
      dp[ 0 ][ i ] = 0;
    for( int msk = 1 ; msk < ( 1 << t ) ; msk ++ ){
      if( msk == ( msk & (-msk) ) ){
        int who = __lg( msk );
        for( int i = 0 ; i < n ; i ++ )
          dp[ msk ][ i ] = dst[ ter[ who ] ][ i ];
        continue;
      }
      for( int i = 0 ; i < n ; i ++ )
        for( int submsk = ( msk - 1 ) & msk ; submsk ;
             submsk = ( submsk - 1 ) & msk )
            dp[ msk ][ i ] = min( dp[ msk ][ i ],
                    dp[ submsk ][ i ] +
                    dp[ msk ^ submsk ][ i ] );
      for( int i = 0 ; i < n ; i ++ ){
        tdst[ i ] = INF;
        for( int j = 0 ; j < n ; j ++ )
          tdst[ i ] = min( tdst[ i ],
               dp[ msk ][ j ] + dst[ j ][ i ] );
      }
      for( int i = 0 ; i < n ; i ++ )
        dp[ msk ][ i ] = tdst[ i ];
    }
    int ans = INF;
    for( int i = 0 ; i < n ; i ++ )
      ans = min( ans , dp[ ( 1 << t ) - 1 ][ i ] );
    return ans;
  }
} solver;
```

4.13 最小平均環

```
// from BCW

/* minimum mean cycle */
const int MAXE = 1805;
const int MAXN = 35;
const double inf = 1029384756;
const double eps = 1e-6;
struct Edge {
  int v,u;
  double c;
};
int n,m,prv[MAXN][MAXN], prve[MAXN][MAXN], vst[MAXN];
Edge e[MAXE];
vector<int> edgeID, cycle, rho;
double d[MAXN][MAXN];
inline void bellman_ford() {
  for(int i=0; i<n; i++) d[0][i]=0;
  for(int i=0; i<n; i++) {
    fill(d[i+1], d[i+1]+n, inf);
    for(int j=0; j<m; j++) {
      int v = e[j].v, u = e[j].u;
      if(d[i][v]<inf && d[i+1][u]>d[i][v]+e[j].c) {
        d[i+1][u] = d[i][v]+e[j].c;
        prv[i+1][u] = v;
        prve[i+1][u] = j;
      }
    }
  }
}
double karp_mmc() {
  // returns inf if no cycle, mmc otherwise
  double mmc=inf;
  int st = -1;
  bellman_ford();
  for(int i=0; i<n; i++) {
    double avg=-inf;
    for(int k=0; k<n; k++) {
      if(d[n][i]<inf-eps) avg=max(avg,(d[n][i]-d[k][i])
          /(n-k));
      else avg=max(avg,inf);
    }
    if (avg < mmc) tie(mmc, st) = tie(avg, i);
  }
  for(int i=0; i<n; i++) vst[i] = 0;
  edgeID.clear(); cycle.clear(); rho.clear();
  for (int i=n; !vst[st]; st=prv[i--][st]) {
    vst[st]++;
    edgeID.PB(prve[i][st]);
    rho.PB(st);
  }
  while (vst[st] != 2) {
    int v = rho.back(); rho.pop_back();
    cycle.PB(v);
    vst[v]++;
  }
  reverse(ALL(edgeID));
  edgeID.resize(SZ(cycle));
  return mmc;
}
```

4.14 Tarjan

```
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 Data Structure

5.1 Sparse Table

```
const int MAXN = 200005;
const int lgN = 20;

struct SP{ //sparse table
  int Sp[MAXN][lgN];
  function<int(int,int)> opt;
  void build(int n, int *a){ // 0 base
    for (int i=0 ;i<n; i++) Sp[i][0]=a[i];

    for (int h=1; h<lgN; h++){
      int len = 1<<(h-1), i=0;
      for (; i+len<n; i++)
        Sp[i][h] = opt( Sp[i][h-1] , Sp[i+len][h-1] );
      for (; i<n; i++)
        Sp[i][h] = Sp[i][h-1];
    }
  }
  int query(int l, int r){
    int h = __lg(r-l+1);
    int len = 1<<h;
    return opt( Sp[l][h] , Sp[r-len+1][h] );
  }
};
```

5.2 Segment Tree

```
int n,m,i,a,b,c;
int ans[MAXN<<2],add[MAXN<<2],inp[MAXN<<2];

inline int ls(const int&p){
    return p<<1;
}

inline int rs(const int&p){
    return p<<1|1;
}

inline int Max(const int&x,const int&y){
    return x>y?x:y;
}

inline void push_up(const int&p,const int&tag){
    ans[p]=Max(ans[ls(p)],ans[rs(p)])+tag;
}

void build(const int l=1,const int r=n,const int p=1){
    if(l==r){
        get(ans[p]);
        inp[l]=ans[p];
        return;
    }
    int mid=(l+r)>>1;
    build(l, mid, ls(p));
    build(mid+1,r,rs(p));
    push_up(p,0);
}

inline void update(const int&x,const int&y,const int&k,
    const int&l=1,const int&r=n,const int&p=1){
    if(l>=x&&r<=y){
        add[p]+=k;
        ans[p]+=k;
        return;
    }
    int mid=(l+r)>>1;
    if(x<=mid){
        update(x,y,k,l, mid, ls(p));
    }
    if(y>mid){
        update(x,y,k,mid+1,r,rs(p));
    }
    push_up(p,add[p]);
}

inline int query(const int &x,const int &y,const int &
    tag=0,const int &l=1,const int &r=n,const int &p=1)
    {
    if(l>=x&&r<=y){
        return ans[p]+tag;
    }
    int mx=-1;
    int mid=(l+r)>>1;
    if(x<=mid){
        mx=Max(mx,query(x,y,tag+add[p],l, mid, ls(p)));
    }
    if(y>mid){
        mx=Max(mx,query(x,y,tag+add[p],mid+1,r,rs(p)));
    }
    return mx;
}
```

5.3 Djs

```
struct DisjointSet{
    int n, fa[MAXN];

      void init(int size) {
            for (int i = 0; i <= size; i++) {
                    fa[i] = i;
                        }
              }

        void find(int x) {
```

```
       return fa[x] == x ? x : find(fa[x]);
        }

     void unite(int x, int y) {
        p[find(x)] = find(y);
         }

} djs;
```

5.4 Binary Indexed Tree

```
vector<int> bit;
int size;
int lowbit(int x){
    return x & (-x);
}
void update(int p, int val){
    while(p <= size){
        bit[p] += val;
        p += lowbit(p);
    }
}
int sum(int p){
    int ans = 0;
    while(p > 0){
        ans += bit[p];
        p -= lowbit(p);
    }
    return ans;
}
vector<int> countSmaller(vector<int>& nums) {
    if(nums.empty()){
        return vector<int>{};
    }
    size = nums.size();
    vector<int> ans(size, 0);
    bit = vector<int>(size + 1, 0);
    vector<int> tmp = nums;
    unordered_map<int, int> m;
    sort(tmp.begin(), tmp.end());
    for(int i = 0;i < size; ++i){
        m[tmp[i]] = i + 1;
    }
    for(int i = size - 1;i >= 0; --i){
        ans[i] = sum(m[nums[i]] - 1);
        update(m[nums[i]], 1);
    }
    return ans;
}
```

6 String

6.1 KMP

```
template<typename T>
void build_KMP(int n, T *s, int *f){ // 1 base
 f[0]=-1, f[1]=0;
 for (int i=2; i<=n; i++){
  int w = f[i-1];
  while (w>=0 && s[w+1]!=s[i])w = f[w];
  f[i]=w+1;
 }
}

template<typename T>
int KMP(int n, T *a, int m, T *b){
 build_KMP(m,b,f);
 int ans=0;

 for (int i=1, w=0; i<=n; i++){
  while ( w>=0 && b[w+1]!=a[i] )w = f[w];
  w++;
  if (w==m){
   ans++;
   w=f[w];
  }
 }
 return ans;
}
```

7 Dark Code

7.1 PBDS

```
#include<ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
#include <ext/pb_ds/detail/standard_policies.hpp>
typedef tree<int,null_type,less<int>,rb_tree_tag,
  tree_order_statistics_node_update> ordered_set;
```

8 Search

8.1 LIS

```
int LIS(vector<int>& s)
{
    if (s.size() == 0) return 0;

    vector<int> v;
    v.push_back(s[0]);

    for (int i = 1; i < s.size(); ++i)
    {
        int n = s[i];

        if (n > v.back())
            v.push_back(n);
        else
            *lower_bound(v.begin(), v.end(), n) = n;
    }

    return v.size();
}
```

8.2 Merge sort

```
void merge(int *vec, int start, int end)
{
    if (start >= end) return;
    int mid = start + ((end - start) >> 1);
    merge(vec, start, mid);
    merge(vec, mid + 1, end);
    for (int i = start; i <= mid; i++)
    {
        //ans += distance(lower_bound(vec + mid + 1,
          vec + end + 1, vec[i] + lower),
        //           upper_bound(vec + mid + 1,
          vec + end + 1, vec[i] + upper));
        //Do some cool stuffs
    }
    inplace_merge(vec + start, vec + mid + 1, vec + end
         + 1);
}
```

8.3 離散化

```
int main()
{
    int n, in;
    vector<int> data, lib;
    cin >> n;
    for (int i = 0; i < n; i++)
    {
        cin >> in;
        data.push_back(in);
    }
    lib = data;
    sort(lib.begin(), lib.end());
    lib.erase(unique(lib.begin(), lib.end()), lib.end()
        );

    for (auto i:data)
        cout << lower_bound(lib.begin(), lib.end(), i)
            - lib.begin() << " ";
    cout << endl;
}
```

9 Others

9.1 數位統計

```
int dfs(int pos, int state1, int state2 ....., bool
    limit, bool zero) {
    if ( pos == -1 ) return 是否符合條件;
    int &ret = dp[pos][state1][state2][....];
    if ( ret != -1 && !limit ) return ret;
    int ans = 0;
    int upper = limit ? digit[pos] : 9;
    for ( int i = 0 ; i <= upper ; i++ ) {
        ans += dfs(pos - 1, new_state1, new_state2,
            limit & ( i == upper), ( i == 0) && zero);
    }
    if ( !limit ) ret = ans;
    return ans;
}

int solve(int n) {
    int it = 0;
    for ( ; n ; n /= 10 ) digit[it++] = n % 10;
    return dfs(it - 1, 0, 0, 1, 1);
}
```

9.2 1D/1D dp 優化

```
#include<bits/stdc++.h>

int t, n, L;
int 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;
}
```

9.3 Theorm - DP optimization

```
Monotonicity & 1D/1D DP & 2D/1D DP
-------------------------------------------------------
Definition xD/yD
1D/1D DP[j] = min(0≤i<j) { DP[i] + w(i, j) }; DP[0] = k
2D/1D DP[i][j] = min(i<k≤j) { DP[i][k - 1] + DP[k][j] }
     + w(i, j); DP[i][i] = 0
-------------------------------------------------------
Monotonicity
       c       d
  -----------------
a | w(a, c) w(a, d)
b | w(b, c) w(b, d)

Monge Condition
Concave(凹四邊形不等式): w(a, c) + w(b, d) >= w(a, d) +
     w(b, c)
Convex (凸四邊形不等式): w(a, c) + w(b, d) <= w(a, d) +
     w(b, c)

Totally Monotone
Concave(凹單調): w(a, c) <= w(b, d) -----> w(a, d) <= w
    (b, c)
Convex (凸單調): w(a, c) >= w(b, d) -----> w(a, d) >= w
    (b, c)
-------------------------------------------------------
1D/1D DP O(n^2) -> O(nlgn)
```

```
**CONSIDER THE TRANSITION POINT**
Solve 1D/1D Concave by Stack
Solve 1D/1D Convex by Deque
-------------------------------------------------------
2D/1D Convex DP (Totally Monotone) O(n^3) -> O(n^2)
h(i, j − 1) ≤ h(i, j) ≤ h(i + 1, j)
```

9.4 Mo’ s algorithm

```
int l = 0, r = 0, nowAns = 0, BLOCK_SIZE, n, m;
int ans[];
struct QUE{
    int l, r, id;
    friend bool operator < (QUE a, QUE b){
        if(a.l / BLOCK_SIZE != b.l / BLOCK_SIZE)
            return a.l / BLOCK_SIZE < b.l / BLOCK_SIZE;
        return a.r < b.r;
    }
}querys[];

inline void move(int pos, int sign) {
    // update nowAns
}

void solve() {
    BLOCK_SIZE = int(ceil(pow(n, 0.5)));
    sort(querys, querys + m);
    for (int i = 0; i < m; ++i) {
        const QUE &q = querys[i];
        while (l > q.l) move(--l, 1);
        while (r < q.r) move(r++, 1);
        while (l < q.l) move(l++, -1);
        while (r > q.r) move(--r, -1);
        ans[q.id] = nowAns;
    }
}
```

10 Persistence