Beijing Jiaotong University , Standard Code Library

SpadeAce, Fengdalu

December 8, 2016

Contents

基础

头文件

```cpp
#include <bits/stdc++.h>
typedef long long ll;
#define MP make_pair
#define AA first
#define BB second
#define PB push_back
#define SZ size
#define OP begin()
#define ED end()
typedef std::pair<int, int> PII;
/* ========================================================== */
```

二进制函数 枚举组合数

```cpp
/* __builtin_clz  前导 0
 * __builtin_ctz 后缀 0
 */
#define ONES(x) __builtin_popcount(x) // 1 数目

// 枚举 C(n, k) 所有可能, 复杂度 O(C(n, k))
int next_combination(int n, int k) {
  int ret, b = k & -k, t = (k + b);
  ret = (((t ^ k) >> 2) / b) | t;
  if (ret >= (1 << n)) return 0;
  return ret;
}

void run(int n, int k) {
  int ik = (1 << k) - 1;
  do {

  } while(ik = next_combination(n, ik));
}
```

日期时间公式

```cpp
using namespace std;

int zeller(int y, int m, int d) {
    if(m <= 2)  y--, m += 12; int c=y/100; y%=100;
    int w=((c>>2)-(c<<1)+y+(y>>2)+(13*(m+1)/5)+d-1)%7;
    if(w<0) w+=7; return (w);
}
```

IO

Cpp 快速读入

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

using namespace std;

struct FastIO {
  static const int S = 65536;
  char buf[S];
  int pos, len;
  bool eof;
  FILE *in;
  FastIO(FILE *_in = stdin) {
    in = _in;
    pos = len = 0;
    eof = false;
  }
  int nextChar() {
    if (pos == len)
      pos = 0, len = fread(buf, 1, S, in);
    if (pos == len) {eof = true; return -1;}
    return buf[pos++];
  }
  int nextUInt() {
    int c = nextChar(), x = 0;
    while (c <= 32) c = nextChar();
    for (;'0' <= c && c <= '10'; c = nextChar()) x = x * 10 + c - '0';
    return x;
  }
  int nextInt() {
    int s = 1, c = nextChar(), x = 0;
    while (c <= 32) c = nextChar();
    if (c == '-') s = -1, c = nextChar();
    for (; '0' <= c && c <= '9'; c = nextChar()) x = x * 10 + c - '0';
    return x * s;
  }
  void nextString(char *s) {
    int c = nextChar();
    while (c <= 32) c = nextChar();
    for(; c > 32; c = nextChar()) *s++ = c;
    *s = 0;
  }
};
```

Java 模板

```java
import java.io.BufferedReader;
import java.io.InputStream;
```

```java
import java.io.InputStreamReader;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Arrays;
import java.util.LinkedList;
import java.util.Queue;
import java.util.StringTokenizer;

public class Main {
  public static void main(String[] args) {
    new Main().run();
  }

  public void run() {
    InputReader reader = new InputReader(System.in);
    PrintWriter writer = new PrintWriter(System.out);
    try {
    } catch (Exception e) {
    } finally {
      writer.close();
    }
  }

  class InputReader {
    InputReader(InputStream in) {
      this.reader    = new BufferedReader(new InputStreamReader(in));
      this.tokenizer = new StringTokenizer("");
    }

    public String nextToken() throws IOException {
      while (!tokenizer.hasMoreTokens()) {
        tokenizer = new StringTokenizer(reader.readLine());
      }
      return tokenizer.nextToken();
    }

    public int nextInt() throws IOException {
      return Integer.parseInt(nextToken());
    }

    private BufferedReader  reader;
    private StringTokenizer tokenizer;
  }

  private void debug(Object...o) {
    System.err.println(Arrays.deepToString(o));
  }
}
```

数据结构

BIT

```cpp
/*
 * 支持第 k 大的 BIT
 * Kyb
 */
inline int LB(int x) {return x & (-x);}
const int MXN = 1e5;
template <typename T>
struct BIT {
  T _[MXN+5];
  int n;
  void init(int m) {
    n = m + 5;
    for (int i = 0; i <= n; i++)_[i] = 0;
  }
  T query(int w) {
    T ret = 0;
    for (w += 3; w > 0; w -= LB(w))ret += _[w];
    return ret;
  }
  void update(int w, T d) {
    for (w += 3; w < n; w += LB(w))_[w] += d;
  }
  /*
   * 待验证
   */
  int find_Kth(int k) {        // UESTC_Dagon
    int ans=0,cnt=0;
    for(int i=22;i>=0;i--){
      ans+=1<<i;
      if(ans>=n||cnt+_[ans]>=k)ans-=1<<i;
      else cnt+=_[ans];
    }
    return ans - 2;
  }
};
```

LCA

```cpp
#include <bits/stdc++.h>
const int POW = 18;
const int N = 1e5;
/*
```

```
 * p[i][j]: i的第j倍祖先
 * d[i]: i在树中的深度
 * edge[N]: 边集合
 * dfs(u, fa): 求出p, d
 * lca(a, b): 求出(a, b)的最近公共祖先
 */
int p[N][POW];
int d[N];
std::vector<int>edge[N];
void dfs(int u, int fa){
  d[u] = d[fa] + 1;
  p[u][0] = fa;
  for(int i = 1; i < POW; i++) p[u][i] = p[p[u][i - 1]][i - 1];
  int sz = edge[u].size();
  for(int i = 0;i < sz; i++){
    int v = edge[u][i];
    if(v == fa) continue;
    dfs(v, u);
  }
}

int lca(int a, int b) {
  if(d[a] > d[b]) std::swap(a, b);
  if(d[a] < d[b]) {
    int del = d[b] - d[a];
    for(int i = 0; i < POW; i++) if(del & (1 << i)) b = p[b][i];
  }
  if(a != b) {
    for(int i = POW - 1; i >= 0; i--)
      if(p[a][i] != p[b][i])
        a = p[a][i], b = p[b][i];
    a = p[a][0], b = p[b][0];
  }
  return a;
}
```

RMQ

```
/*
 * f[i][j]: i开头2^j长度区间的最大值
 *
 */
#include <algorithm>

const int N = 1e5;
const int POW = 32;
int f[N][POW];
int mm[N];
void init(int n) {
  for(int j = 0; j < POW; j++)
    for(int i = 0; i < n; i++)
      if(i + (1 << (j + 1)) <= n) {
        f[i][j + 1] = std::max(f[i][j], f[i + (1 << j)][j]);
      }
  mm[0] = -1;
  for(int i = 1; i < N; i++) {
    mm[i] = mm[i >> 1] + 1;
  }
}

int cal(int l, int r) {
  int k = mm[r - l + 1];
  return std::max(f[l][k], f[r - (1 << k) + 1][k]);
}
```

二维线段树

```
/*
 * 点修改区间查询
 */
#include <bits/stdc++.h>
const int N = 5000;
int c[N][N];
int lx[N], ly[N];
int n, Q;
struct Node
{
  struct node
  {
    int a, b;
    int Min, Max;
  }f[N];
  int a, b;

  void update(int x)
  {
    f[x].Min = std::min(f[x << 1].Min, f[x << 1 | 1].Min);
    f[x].Max = std::max(f[x << 1].Max, f[x << 1 | 1].Max);
  }

  void build(int x, int a, int b)
  {
    f[x].a = a; f[x].b = b;
    if(a < b)
    {
      int mid = (a + b) / 2;
      build(x << 1, a, mid);
      build(x << 1 | 1, mid + 1, b);
      update(x);
```

```
    } else { f[x].Min = f[x].Max = 0; ly[a] = x; }
  }

  int queryMin(int x, int a, int b)
  {
    if(a <= f[x].a && f[x].b <= b) return f[x].Min;
    else
    {
      int mid = (f[x].a + f[x].b) / 2;
      if(b <= mid) return queryMin(x << 1, a, b);
      else if(a > mid) return queryMin(x << 1 | 1, a, b);
      else return std::min(queryMin(x << 1, a, b), queryMin(x << 1 | 1, a, b));
    }
  }

  int queryMax(int x, int a, int b)
  {
    if(a <= f[x].a && f[x].b <= b) return f[x].Max;
    else
    {
      int mid = (f[x].a + f[x].b) / 2;
      if(b <= mid) return queryMax(x << 1, a, b);
      else if(a > mid) return queryMax(x << 1 | 1, a, b);
      else return std::max(queryMax(x << 1, a, b), queryMax(x << 1 | 1, a, b));
    }
  }
}f[N];

void build(int x, int a, int b, int p, int q)
{
  f[x].a = a; f[x].b = b;
  f[x].build(1, p, q);
  if(a < b)
  {
    int mid = (a + b) / 2;
    build(x << 1, a, mid, p, q);
    build(x << 1 | 1, mid + 1, b, p, q);
  }
  else lx[a] = x;
}

void update(int x, int y, int c)
{
  x = lx[x]; y = ly[y];
  for(int X = x; X != 0; X >>= 1)
    for(int Y = y; Y != 0; Y >>= 1)
    {
      if(f[X].f[Y].a == f[X].f[Y].b)
      {
        if(f[X].a == f[X].b)
        {
          f[X].f[Y].Min = f[X].f[Y].Max = c;
        }
        else
        {
          f[X].f[Y].Min = std::min(f[X << 1].f[Y].Min, f[X << 1 | 1].f[Y].Min);
          f[X].f[Y].Max = std::max(f[X << 1].f[Y].Max, f[X << 1 | 1].f[Y].Max);
        }
      }
      else { f[X].update(Y); }
    }
}

int queryMin(int x, int a, int b, int p, int q)
{
  if(a <= f[x].a && f[x].b <= b) return f[x].queryMin(1, p, q);
  else
  {
        int mid = (f[x].a + f[x].b) / 2;
        if(b <= mid) return queryMin(x << 1, a, b, p, q);
        else if(a > mid) return queryMin(x << 1 | 1, a, b, p, q);
        else return std::min(queryMin(x << 1, a, b, p, q), queryMin(x << 1 | 1, a, b, p,
          ↪ q));
    }
}

int queryMax(int x, int a, int b, int p, int q)
{
    if(a <= f[x].a && f[x].b <= b) return f[x].queryMax(1, p, q);
    else
    {
        int mid = (f[x].a + f[x].b) / 2;
        if(b <= mid) return queryMax(x << 1, a, b, p, q);
        else if(a > mid) return queryMax(x << 1 | 1, a, b, p, q);
        else return std::max(queryMax(x << 1, a, b, p, q), queryMax(x << 1 | 1, a, b, p,
          ↪ q));
    }
}
```

主席树

```
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#include <set>
#include <map>
#include <vector>
#include <queue>

```

```cpp
typedef std::pair<int, int> PII;
#define PB(x) push_back(x)
#define SZ size()
#define AA first
#define BB second
#define MP(x, y) make_pair(x, y)
namespace ST {
  /*
   *     e:    线段树节点数组
   *  l, r:    左右节点指针
   *   sum:    区间和
   *    rt:    主席树根节点
   *   tot:    总节点个数
   */
  struct E {
    int l, r;
    int sum;
  }e[10000000];
  int rt[100010];
  int tot;

  /*
   * 建树 , 初始化调用 , 建立一颗空树
   */
  void build(int &rt, int l, int r) {
    rt = tot++;
    e[rt].sum = 0;
    if(l == r) return;
    int mid = (l + r) >> 1;
    build(e[rt].l, l, mid);
    build(e[rt].r, mid + 1, r);
  }

  /*
   * 插入新节点，向$w$位置插入节点，主席书中父亲结点为$fa$
   */
  void update(int &rt, int l, int r, int w, int fa) {
    rt = tot++;
    e[rt].l = e[fa].l;
    e[rt].r = e[fa].r;
    e[rt].sum = e[fa].sum + 1;
    if(l == r) return ;
    int mid = (l + r) >> 1;
    if(w <= mid) update(e[rt].l, l, mid, w, e[fa].l);
    else update(e[rt].r, mid + 1, r, w, e[fa].r);
  }

  /*
   * 查询，查询原图中$(a, b)$点对的信息，其中$c = lca(a, b)$
   * 计算时候加入$c$信息
   */
  int query(int a, int b, int c, int l, int r, int k, int nd) {
    int mid = (l + r) >> 1;
    int sum = e[e[a].l].sum + e[e[b].l].sum - e[e[c].l].sum * 2 + (int)(nd <= mid);
    if(l == r) return l;
    if(k <= sum) return query(e[a].l, e[b].l, e[c].l, l, mid, k, nd);
    else return query(e[a].r, e[b].r, e[c].r, mid + 1, r, k - sum, (nd <= mid ? 1e9 :
      ↪ nd));
  }
};
```

树链剖分

```cpp
/*
 * $siz[v]$表示以$v$为根的子树的节点数
 * $dep[v]$表示$v$的深度
 * $top[v]$表示$v$所在的重链的顶端节点
 * $fa[v]$表示$v$的父亲
 * $son[v]$表示与$v$在同一重链上的$v$的儿子节点
 * $w[v]$表示$v$与其父亲节点的连边在线段树中的位置
 * 初始需要调用$cnt1 = cnt2 = cnt3 = 0; dfs1(ROOT, 0); dfs2(ROOT, 1); bt(1, cnt2);$
 * 模板为边带权值，点带权值需要修改$query(x, y)$
 * $update(x, p, c)$的$p$为线段树中的编号，更新$x$需要调用$w[x]$
 */
const int N = 1e5;
const int M = 2 * N;
typedef long long ll;
#define MID(x, y) (((x) + (y)) >> 1)
#include <bits/stdc++.h>

int fa[N], top[N], w[N], son[N], dep[N], sz[N], r[N];
int a[N], b[N];
ll c[N];
int ind[N];
int t[M], nt[M];
int cnt1, cnt2, cnt3;
int n, m;

struct node
{
  int l, r;
  int a, b;
  ll sum;
}f[M];
int rt;

void dfs1(int x, int d)
{
  dep[x] = d;
  son[x] = 0;
```

```
  sz[x] = 1;
  for(int k = ind[x]; k != -1; k = nt[k])
    if(t[k] != fa[x])
    {
      fa[t[k]] = x;
      dfs1(t[k], d + 1);
      sz[x] += sz[t[k]];
      if(sz[t[k]] > sz[son[x]]) son[x] = t[k];
    }
}

void dfs2(int x, int tt)
{
  w[x] = ++cnt2;
  top[x] = tt;
  if(son[x]) dfs2(son[x], tt);
  for(int k = ind[x]; k != -1; k = nt[k]) if(t[k] != fa[x] && t[k] != son[x])
    dfs2(t[k], t[k]);
}

void add(int a, int b)
{
  t[cnt1] = b;
  nt[cnt1] = ind[a];
  ind[a] = cnt1++;
}

void update(int x)
{
  f[x].sum = f[f[x].l].sum + f[f[x].r].sum;
}

int bt(int a, int b)
{
  int x = cnt3++;
  f[x].a = a; f[x].b = b;
  if(a < b)
  {
    int mid = MID(a, b);
    f[x].l = bt(a, mid);
    f[x].r = bt(mid + 1, b);
    f[x].sum = 0;
  }
  else
  {
    f[x].sum = 0;
  }
  return x;
}

```

```
// 在线段树上查询，不要直接调用
ll query(int x, int a, int b)
{
  if(a <= f[x].a && f[x].b <= b) return f[x].sum;
  int mid = MID(f[x].a, f[x].b);
  ll ans = 0;
  if(a <= mid) ans += query(f[x].l, a, b);
  if(b > mid) ans += query(f[x].r, a, b);
  return ans;
}

// 单调修改
void update(int x, int p, int cc)
{
    if(f[x].a == f[x].b) { f[x].sum = cc; return; }
    int mid = MID(f[x].a, f[x].b);
    if(p <= mid) update(f[x].l, p, cc);
    else update(f[x].r, p, cc);
    update(x);
}

// 树上查询
ll query(int x, int y)
{
    int fx = top[x], fy = top[y];
    ll sum = 0;
    while(fx != fy)
    {
        if(dep[fx] < dep[fy])
        {
            std::swap(x, y);
            std::swap(fx, fy);
        }
        sum += query(rt, w[fx], w[x]);
        x = fa[top[x]];
        fx = top[x];
    }
    if(dep[x] > dep[y]) std::swap(x, y);
    if(x == y) return sum;
    return sum + query(rt, w[son[x]], w[y]);
}
```

树分治

```
/*
 * 树分治
 */

// 点分治
#include <bits/stdc++.h>
```

```
const int N = 100000 + 5;
std::vector<int> edges[N];
int n;
bool vis[N];
int parent[N];
int sz[N];
int que[N];
int balance[N];

int bfs(int source,int fa = -1) {
  int qf = 0,qe = 0;
  que[qe++] = source;
  parent[source] = fa;
  while (qf != qe) {
    int u = que[qf++];
    sz[u] = 1;
    balance[u] = 0;
    for (int v : edges[u]) {
      if (!vis[v] && parent[u] != v) {
        parent[v] = u;
        que[qe++] = v;
      }
    }
  }
  for (int i = qe - 1; i > 0; -- i) {
    int u = que[i];
    sz[parent[u]] += sz[u];
    balance[parent[u]] = std::max(balance[parent[u]],sz[u]);
  }
  return qe;
}

void divide(int root) {
  int tot = bfs(root);
  for (int i = tot - 1; i > 0; -- i) {
    int u = que[i];
    balance[u] = std::max(balance[u],tot - sz[u]);
    if (balance[u] < balance[root]) {
      root = u;
    }
  }
  bfs(root);

  // balabalah

  vis[root] = true;
  for (int u : edges[root]) {
    if (!vis[u]) {
      divide(u);
    }
  }
}
```

Treap

```
/*
 * Poj 3481
 */
#include <bits/stdc++.h>
using namespace std;
#define AA first
#define BB second
#define MP make_pair
#define PII pair<int, int>

const int N = 1000000+1000, MOD = 1e9+7;
const int nil = 0;
struct node {
    int ch[2], key, sz;
    PII data;
} f[N];
int cnt, rt;

void up(int cur) {
    f[cur].sz = f[f[cur].ch[0]].sz + f[f[cur].ch[1]].sz + 1;
}

int newNode(PII data) {
    cnt++;
    f[cnt].ch[0] = f[cnt].ch[1] = 0;
    f[cnt].data = data;
    f[cnt].sz = 1;
    f[cnt].key = rand();
    return cnt;
}

PII split(int p, int n) {
    if(n == 0) return MP(nil, p);
    int sz = f[f[p].ch[0]].sz;
    if(n == sz) {
        int x = f[p].ch[0];
        f[p].ch[0] = nil;
        up(p);
        return MP(x, p);
    }
    else if(n == sz + 1) {
        int x = f[p].ch[1];
        f[p].ch[1] = nil;
        up(p);
        return MP(p, x);
```

```
    }
    else if(n < sz) {
        PII res = split(f[p].ch[0], n);
        f[p].ch[0] = res.BB;
        up(p);
        return MP(res.AA, p);
    }
    else {
        PII res = split(f[p].ch[1], n - sz - 1);
        f[p].ch[1] = res.AA;
        up(p);
        return MP(p, res.BB);
    }
}

int merge(int p, int q) {
    if(p == nil) return q;
    if(q == nil) return p;
    if(f[p].key > f[q].key) {
        f[q].ch[0] = merge(p, f[q].ch[0]);
        up(q);
        return q;
    }
    else {
        f[p].ch[1] = merge(f[p].ch[1], q);
        up(p);
        return p;
    }
}

int getrank(int p, int w) {
    int ans = 0;
    while(p != nil) {
        if(w == f[p].data.BB) return ans + f[f[p].ch[0]].sz;
        if(w < f[p].data.BB) p = f[p].ch[0];
        else {
            ans += f[f[p].ch[0]].sz + 1;
            p = f[p].ch[1];
        }
    }
    return ans;
}
```

Splay

```
// 注意初始化内存池和 null 节点
/*
 * hdu3487 hdu1908
 */
#include <algorithm>
const int MAX_NODE = 1e5;
struct Node{
  int rev,size; Node *ch[2],*p;
  void set(Node*,int); int dir(); void update(); void relax(); void appRev();
} nodePool[MAX_NODE],*curNode,*null;
Node *newNode(){
  Node *t=curNode++; t->rev=0, t->size=1;
  t->ch[0]=t->ch[1]=t->p=null; return t;
}
struct Splay{
  Node *root;
  Splay(){ root=newNode(); root->set(newNode(),0); root->update(); }
  void rot(Node *t){
    Node *p=t->p; int d=t->dir();
    p->relax(); t->relax();
    if(p==root) root=t;
    p->set(t->ch[!d],d); p->p->set(t,p->dir()); t->set(p,!d);
    p->update();
  }
  void splay(Node *t,Node *f=null){
    for(t->relax();t->p!=f;)
      if(t->p->p==f) rot(t);
      else t->dir()==t->p->dir()?(rot(t->p),rot(t)):(rot(t),rot(t));
    t->update();
  }
};
void initNull(){ curNode=nodePool;null=curNode++;null->size=0; }
void Node::set(Node *t,int _d){ ch[_d]=t; t->p=this; }
int Node::dir(){ return this==p->ch[1]; }
void Node::update(){ size=ch[0]->size+ch[1]->size+1;}
void Node::relax(){ if(rev) ch[0]->appRev(), ch[1]->appRev(), rev=false; }
void Node::appRev(){ if(this==null) return; rev^=true; std::swap(ch[0],ch[1]); }
```

KD-Tree

```
/*
 * KDT 2016 Qingdao 11
 * 估价函数 :
 * 欧几里德距离下界 : sqr(max(max(X-x.Max[0],x.Min[0]-X,0)))+ ...
 * 欧几里德距离上界 : max(sqr(X-x.Max[0]),sqr(x.Min[0]-X))+ ...
 * 曼哈顿距离下界 : max(X-x.Max[0],0)+max(x.Min[0]-X,0)+ ...
 * 曼哈顿距离上界 : max(abs(X-x.Max[0]),abs(x.Min[0]-X))+ ...
 *
 */

#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
#define cmin(x, y) x = min(x, y)
```

```
#define cmax(x, y) x = max(x, y)

const int N = 1000000;
struct point {
  LL _[2], op, p;
  LL& operator [] (int x) { return _[x]; }
  int operator < (const point &t) const {
    return p < t.p;
  }
} a[N], ans[N];

template <typename T> inline T SQ(T x) { return x * x; }
inline LL dis(point a, point b) {
  return SQ(1LL * (a[0] - b[0])) + SQ(1LL * (a[1] - b[1]));
}
struct node {
  int l, r, fa;
  bool vis, has;
  point p;
} b[N];
int n, m, cnt;

inline int getnode() {
  assert(cnt < N);
  b[cnt].l = b[cnt].r = -1;
  b[cnt].vis = b[cnt].has = false;
  b[cnt].fa = -1;
  return cnt++;
}

void sol(int u, point pt, point& ans, LL &value, int d) {
  if(u == -1 || !b[u].has) return;
  if(b[u].vis) {
    LL w1 = dis(b[u].p, pt);
    if(w1 < value || (w1 == value && b[u].p.op < ans.op)) {
      ans = b[u].p;
      value = w1;
    }
  }
  int l = b[u].l, r = b[u].r;
  if(pt[d] > b[u].p[d]) swap(l, r);
  sol(l, pt, ans, value, d ^ 1);
  if(1LL * SQ(pt[d] - b[u].p[d]) <= value)
    sol(r, pt, ans, value, d ^ 1);
}

void build(int &u, int l, int r, int d, int fa=-1) {

  int mid = l;
  if(l != r) mid += rand() % (r - l);
  point p = a[mid];
  swap(a[r], a[mid]);
  int j = l;
  for(int i = l; i < r; i++)
    if(a[i][d] < p[d]) {
      swap(a[i], a[j++]);
    }
  swap(a[j], a[r]);
  u = getnode();
  b[u].fa = fa;
  b[u].p = a[j];

  if(l < j) build(b[u].l, l, j - 1, d ^ 1, u);
  if(j < r) build(b[u].r, j + 1, r, d ^ 1, u);
}

void add(int u, point p, int d) {
  if(u == -1) return;
  if(b[u].p[0] == p[0] && b[u].p[1] == p[1]) {
    b[u].has = b[u].vis = true;
    return ;
  }
  if(p[d] > b[u].p[d]) add(b[u].r, p, d ^ 1);
  else add(b[u].l, p, d ^ 1);
  b[u].has |= b[u].vis;
  if(~b[u].l) b[u].has |= b[b[u].l].has;
  if(~b[u].r) b[u].has |= b[b[u].r].has;
}
```

KD-Tree 欧几里得距离

```
/*************************************************************
    Problem: 1941
    User: Ceva
    Language: C++
    Result: Accepted
    Time:1476 ms
    Memory:63792 kb
*************************************************************/
/*
 * 求所有点 距离最大和最小值差值的最小值
 */

#include <iostream>
#include <cstdio>
#include <algorithm>
#include <cmath>
#include <cstring>
#include <string>
using namespace std;
```

```cpp
#define cmin(x, y) x = min(x, y)
#define cmax(x, y) x = max(x, y)

const int maxn = 1000010;
const int inf = 1e9;
struct point {
    int x[2];
} p[maxn];
struct node {
    int l, r, p;
    int mn[2], mx[2];
} f[maxn * 2];
int n;
int cur;
int cnt;
int mm, nn;
int operator < (const point &a, const point &b) { return a.x[cur] < b.x[cur]; }

void update(int rt) {
    int id = f[rt].p;
    int l = f[rt].l, r = f[rt].r;
    for(int i = 0; i < 2; i++) {
        f[rt].mn[i] = f[rt].mx[i] = p[id].x[i];
        if(l) cmin(f[rt].mn[i], f[l].mn[i]), cmax(f[rt].mx[i], f[l].mx[i]);
        if(r) cmin(f[rt].mn[i], f[r].mn[i]), cmax(f[rt].mx[i], f[r].mx[i]);
    }
}

int mn(int rt, point P) {
    int tot = 0;
    for(int i = 0; i < 2; i++) {
        int a = 0;
        if(P.x[i] < f[rt].mn[i]) tot += f[rt].mn[i] - P.x[i];
        else if(P.x[i] > f[rt].mx[i]) tot += -f[rt].mx[i] + P.x[i];
        tot += a;
    }
    return tot;
}

int mx(int rt, point P) {
    int tot = 0;
    for(int i = 0; i < 2; i++) {
        int a = -inf;
        cmax(a, abs(P.x[i] - f[rt].mn[i]));
        cmax(a, abs(P.x[i] - f[rt].mx[i]));
        tot += a;
    }
    return tot;
}

void build(int &rt, int l, int r, int mk) {
    if(l > r) return;
    int mid = (l + r) >> 1;
    rt = ++cnt;
    cur = mk;
    nth_element(p + l, p + mid, p + r + 1);
    //cout << p[rt].x[0] << " " << p[rt].x[1]  << " " << l << " " << r << endl;
    f[rt].p = mid;
    f[rt].l = f[rt].r = 0;
    build(f[rt].l, l, mid - 1, mk ^ 1);
    build(f[rt].r, mid + 1, r, mk ^ 1);
    update(rt);
}

int getdis(point &x, point &y) {
    int ans = 0;
    for(int i = 0; i < 2; i++) ans += abs(x.x[i] - y.x[i]);
    return ans;
}

void dfs1(int rt, point &P, int cur, int &m) {
    int l = f[rt].l, r = f[rt].r;
    point &t = p[f[rt].p];
    int tot = getdis(t, P);
    if(tot != 0) cmin(m, tot);
    int a = mn(l, P), b = mn(r, P);
    if(a < b) {
        if(l) if(m > a) dfs1(f[rt].l, P, cur ^ 1, m);
        if(r) if(m > b) dfs1(f[rt].r, P, cur ^ 1, m);
    }
    else {
        if(r) if(m > b) dfs1(f[rt].r, P, cur ^ 1, m);
        if(l) if(m > a) dfs1(f[rt].l, P, cur ^ 1, m);
    }
}

void dfs2(int rt, point &P, int cur, int &m) {
    int l = f[rt].l, r = f[rt].r;
    point &t = p[f[rt].p];
    cmax(m, getdis(t, P));
    int a = mx(l, P), b = mx(r, P);
    if(a > b) {
        if(l) if(m < a) dfs2(f[rt].l, P, cur ^ 1, m);
        if(r) if(m < b) dfs2(f[rt].r, P, cur ^ 1, m);
    }
    else {
        if(r) if(m < b) dfs2(f[rt].r, P, cur ^ 1, m);
        if(l) if(m < a) dfs2(f[rt].l, P, cur ^ 1, m);
    }
```

```
}

int main() {
    scanf("%d", &n);
    for(int i = 0; i < n; i++) scanf("%d%d", &p[i].x[0], &p[i].x[1]);
    int root;
    build(root, 0, n - 1, 0);
    int ans = 1e9;
    cur = 0;
    cnt = 0;
    for(int i = 0; i < n; i++) {
        int mx = 0, mn = inf;
        dfs2(root, p[i], 0, mx); dfs1(root, p[i], 0, mn);
        cmin(ans, mx - mn);
    }
    printf("%d\n", ans);
    return 0;
}
```

DLX 精确覆盖

```
#include <iostream>
#include <cstdio>
#include <cstring>

const int Maxm = 1000;
const int Maxn = 1200;
int ans[Maxn];
struct DLX{
  struct Node{
    Node *L, *R, *U, *D;
    int col, row;
  } *head, *row[Maxn], *col[Maxm], node[Maxn * Maxm];
  int colsum[Maxm], cnt;

  /* dancing link
   * 精确覆盖问题
   * 可以添加迭代加深优化
   * 1) 举深度 h
   * 2) 若当前深度 + predeep > h : return false□
   * 3) mat 下标 1 开始
   */
  /*
     int predeep() {
     bool vis[Maxm];
     memset(vis, 0, sizeof(vis));
     int ret = 0;
     for (Node *p = head->R; p != head; p = p->R)
     if (!vis[p->col]) {
     ret ++ ;
     vis[p->col] ++ ;
     for (Node *q = p->D; q != p; q = p->D)
     for (Node *r = q->R; r != q; r = r->R)
     vis[r->col] = true;
     }
     return ret;
     }
   */
  void init(int mat[][Maxm], int n, int m) {
    cnt = 0;
    memset(colsum, 0, sizeof (colsum) );
    head = &node[cnt ++ ];
    for(int i = 1; i <= n; i ++ )
      row[i] = &node[cnt ++ ];
    for(int j = 1; j <= m; j ++ )
      col[j] = &node[cnt ++ ];
    head->D = row[1], row[1]->U = head;
    head->R = col[1], col[1]->L = head;
    head->U = row[n], row[n]->D = head;
    head->L = col[m], col[m]->R = head;
    head->row = head->col = 0;
    for(int i = 1; i <= n; i ++ ) {
      if (i != n) row[i]->D = row[i + 1];
      if (i != 1) row[i]->U = row[i - 1];
      row[i]->L = row[i]->R = row[i];
      row[i]->row = i, row[i]->col = 0;
    }
    for(int i = 1; i <= m; i ++ ) {
      if (i != m) col[i]->R = col[i + 1];
      if (i != 1) col[i]->L = col[i - 1];
      col[i]->U = col[i]->D = col[i];
      col[i]->col = i, col[i]->row = 0;
    }
    for(int i = n; i > 0; i -- )
      for(int j = m; j > 0; j -- )
        if(mat[i][j]) {
          Node *p = &node[cnt ++ ];
          p->R = row[i]->R, row[i]->R->L = p;
          p->L = row[i], row[i]->R = p;
          p->D = col[j]->D, col[j]->D->U = p;
          p->U = col[j], col[j]->D = p;
          p->row = i;
          p->col = j;
          colsum[j] ++ ;
        }
  }
  void remove(Node *c) {
    c->L->R = c->R;
    c->R->L = c->L;
    for(Node *p = c->D; p != c; p = p->D) {
```

```
      for(Node *q = p->R; q != p; q = q->R) {
        q->U->D = q->D;
        q->D->U = q->U;
        colsum[q->col] -- ;
      }
    }
  }
  void resume(Node *c) {
    for(Node *p = c->U; p != c; p = p->U) {
      for(Node *q = p->L; q != p; q = q->L) {
        q->U->D = q;
        q->D->U = q;
        colsum[q->col] ++ ;
      }
    }
    col[c->col]->L->R = col[c->col];
    col[c->col]->R->L = col[c->col];
  }
  int dfs(int deep) {
    if(head->R == head) return deep;
    Node *p, *q = head->R;
        for(p = head->R; p != head; p = p->R)
            if(colsum[p->col] < colsum[q->col])
                q = p;
        remove(q);
        for(p = q->D; p != q; p = p->D) {
            for(Node* r = p->R; r != p; r = r->R)
                if (r->col != 0)
                    remove (col[r->col]);
            // -------- 可修改区域 -----------
            ans[deep] = p->row;
            // -------------------------------
            int sta = dfs (deep + 1);
            if(sta) return sta;
            for(Node* r = p->L; r != p; r = r->L)
                if(r->col != 0)
                    resume (col[r->col]);
        }
        resume(q);
        return false;
    }
} dlx;
int mat[Maxn][Maxm];
int mem[Maxn][3];   // 记录每行代表哪一格填几
// col = 324
int n;
// 数独填充 (x, y) = v
void addline(int x, int y, int v) {
    int i, j;
    n++;
    mem[n][0] = x;
    mem[n][1] = y;
    mem[n][2] = v;
    for(i = 0; i < Maxm; i++) mat[n][i] = 0;
    mat[n][x * 9 + y + 1] = 1;
    mat[n][81 + x * 9 + v] = 1;
    mat[n][162 + y * 9 + v] = 1;
    mat[n][243 + (3 * (x / 3) + y / 3) * 9 + v] = 1;
}
```

DLX 多重覆盖

```
#include <bits/stdc++.h>
#define cmin(x, y) x = std::min(x, y)
const int Maxm = 62;
const int Maxn = 62;

int K;
struct DLX{
  struct Node{
    Node *L, *R, *U, *D;
    int col, row;
  } *head, *row[Maxn], *col[Maxm], node[Maxn * Maxm];
  int colsum[Maxm], cnt;
  /* dancing link
   * 重复覆盖问题
   * 可以添加迭代加深优化
   * 1) 举深度 h
   * 2) 若当前深度 + predeep > h : return false
   * 3) mat 下标 1 开始
   */

  int predeep(){
    bool vis[Maxm];
    Node * p, *q, *r;
    memset(vis, 0, sizeof(vis));
    int ret = 0;
    for(p = head->R; p != head; p = p->R) {
      if(!vis[p->col]) {
        ret++;
        vis[p->col]++;
        for(q = p->D; q != p; q = q->D) {
          for(r = q->R; r != q; r = r->R) {
            vis[r->col] = true;
          }
        }
      }
    }
    return ret;
  }
```

```
  void init(int mat[][Maxm], int n, int m) {
    cnt = 0;
    int i, j;
    Node * p;
    memset(colsum, 0, sizeof(colsum));
    head = &node[cnt++];
    for(i = 1; i <= n; i++) row[i] = &node[cnt++];
    for(j = 1; j <= m; j++) col[j] = &node[cnt++];
    head->D = row[1], row[1]->U = head;
    head->R = col[1], col[1]->L = head;
    head->U = row[n], row[n]->D = head;
    head->L = col[m], col[m]->R = head;
    head->row = head->col = 0;
    for(i = 1; i <= n; i++) {
      if(i != n) row[i]->D = row[i + 1];
      if(i != 1) row[i]->U = row[i - 1];
      row[i]->L = row[i]->R = row[i];
      row[i]->row = i; row[i]->col = 0;
    }
    for(i = 1; i <= m; i++) {
      if(i != m) col[i]->R = col[i + 1];
      if(i != 1) col[i]->L = col[i - 1];
      col[i]->U = col[i]->D = col[i];
      col[i]->col = i; col[i]->row = 0;
    }
    for(i = n; i > 0; i--) {
      for(j = m; j > 0; j--) {
        if(mat[i][j]) {
          p = &node[cnt++];
          p->R = row[i]->R, row[i]->R->L = p;
          p->L = row[i], row[i]->R = p;
          p->D = col[j]->D, col[j]->D->U = p;
          p->U = col[j], col[j]->D = p;
          p->row = i;
          p->col = j;
          colsum[j]++;
        }
      }
    }
  }
  void remove(Node *c) {
    Node * p;
    for(p = c->D; row[p->row] != row[c->row]; p = p->D) {
      p->R->L = p->L; p->L->R = p->R;
    }
  }
  void resume(Node *c) {
    Node * p;
    for(p = c->U; row[p->row] != row[c->row]; p = p->U) {
      p->L->R = p->R->L = p;
    }
  }

  bool dfs(int deep) {
    if(head->R == head) {
      if(deep <= K) return true;
      return false;
    }
    if(deep + predeep() > K) return false;
    Node *p, *q = head->R, *r;
    for(p = head->R; p != head; p = p->R) {
      if(colsum[p->col] < colsum[q->col]) q = p;
        }
        for(p = q->D; p != q; p = p->D) {
            remove(p);
            for(r = p->R; r != p; r = r->R) {
                if(r->col != 0) remove(r);
            }
            // -------- 可修改区域 -----------
            //       ans[deep] = p->row;
            // ------------------------------
            int sta = 0;
            sta = dfs(deep + 1);
            if(sta) return sta;
            for(r = p->L; r != p; r = r->L) {
                if(r->col != 0) resume(r);
            }
            resume(p);
        }
        return false;
    }
} dlx;
```

图论

2-SAT

```
#include <iostream>
#include <cstdio>
#include <iostream>
#include <stack>
#include <cstring>
#include <algorithm>
#include <cstring>

const int N = 500;
const int M = 2e6;
/*
 * dfn: 标号数组
 * low: 回向边数组
 */
```

```
int ind[N];
int to[M], nt[M];
int dfn[N], low[N];
int color[N];
bool vis[N];
int cnt, num, idn;
int ncnt;
std::stack<int>s;

void add(int i, int j) {
  cnt++;
  to[cnt] = j;
  nt[cnt] = ind[i];
  ind[i] = cnt;
}

void tarjan(int x) {
  if(dfn[x] != 0) return;
  low[x] = dfn[x] = ++num;
  vis[x] = true;
  s.push(x);
  for(int k = ind[x]; k != -1; k = nt[k]) {
    tarjan(to[k]);
    if(vis[to[k]]) low[x] = std::min(low[x], low[to[k]]);
  }
  if(dfn[x] == low[x]) {
    ncnt++;
    while(true) {
      int p = s.top();
      s.pop();
      color[p] = ncnt;
      vis[p] = false;
      if(p == x) break;
    }
  }
}

const int Maxn = 500;
const int Maxm = 50000;
int n, m;
int a[Maxm], b[Maxm], c[Maxm];

bool check() {
  memset(dfn, 0, sizeof dfn);
  memset(low, 0, sizeof low);
  memset(vis, 0, sizeof vis);
  for(int i = 0; i < 2 * n; i++) ind[i] = -1;
  int tot = n;
  num = idn = 0;
  cnt = 0;
  ncnt = 0;
  /*
   * 这里建图
   * 每个节点分为两个，选和不选
   * $(p-> q)$，表示选$p$一定要选$q$
   * 检查每个节点选和不选是否在同一个集合
   */
  for(int i = 0; i < 2 * n; i++) tarjan(i);
  for(int i = 0; i < n; i++) {
    if(color[i] == color[tot + i]) return false;
  }
  return true;
}
```

KM

```
/*
 * 二分图最大权匹配
 * SJTU
 * Hdu 2255
 */
#include <bits/stdc++.h>
using namespace std;

namespace graph {
  const int maxn=400; const int oo=0x7fffffff;
  int w[maxn][maxn],x[maxn],y[maxn],px[maxn],py[maxn],sy[maxn],slack[maxn];
  int par[maxn];int n;int pa[200][2],pb[200][2],n0,m0,na,nb;char s[200][200];
  void adjust(int v){   sy[v]=py[v]; if (px[sy[v]]!=-2) adjust(px[sy[v]]);}
  bool find(int v){for (int i=0;i<n;i++)
    if (py[i]==-1){
      if (slack[i]>x[v]+y[i]-w[v][i]) slack[i]=x[v]+y[i]-w[v][i],   par[i]=v;
      if (x[v]+y[i]==w[v][i]){
        py[i]=v; if (sy[i]==-1){adjust(i);   return 1;}
        if (px[sy[i]]!=-1) continue;   px[sy[i]]=i;
        if (find(sy[i])) return 1;
      }}return 0;}
  int km(){int i,j,m,flag; for (i=0;i<n;i++) sy[i]=-1,y[i]=0;
    for (i=0;i<n;i++) {x[i]=0; for (j=0;j<n;j++) x[i]=max(x[i],w[i][j]);}
    for (i=0;i<n;i++){
      for (j=0;j<n;j++) px[j]=py[j]=-1,slack[j]=oo;
      px[i]=-2;   if (find(i)) continue; flag=false;
      for (;!flag;){ m=oo;
        for (j=0;j<n;j++) if (py[j]==-1) m=min(m,slack[j]);
        for (j=0;j<n;j++){ if (px[j]!=-1) x[j]-=m;
          if (py[j]!=-1) y[j]+=m; else slack[j]-=m;}
        for (j=0;j<n;j++){ if (py[j]==-1&&!slack[j]){
          py[j]=par[j];
          if (sy[j]==-1){   adjust(j);   flag=true;   break;}
          px[sy[j]]=j;    if (find(sy[j])){flag=true;break;}
```

```
        }}}}
    int ans=0; for (i=0;i<n;i++) ans+=w[sy[i]][i];return ans;}
}
using namespace graph;
```

ISAP

```
/*
 * 用于边数较多的情况，调用ISAP(intst,inted,intn)，n要稍大
 */

#include <bits/stdc++.h>
const int Maxn = 1000;
const int Maxm = Maxn * Maxn;
struct node {
  int u, v, c, next;
}e[Maxm];
int tot, last[Maxn];
void adde(int u, int v, int c, int c1) {
  e[tot].u = u; e[tot].v = v; e[tot].c = c; e[tot].next = last[u]; last[u] = tot++;
  e[tot].u = v; e[tot].v = u; e[tot].c = c1; e[tot].next = last[v]; last[v] = tot++;
}

int dist[Maxn], cur[Maxn], gap[Maxn], pre[Maxn];
int ISAP(int s, int t, int n) {
  int i, j, u, v, det;
  int maxflow = 0;
  memset(dist, 0, sizeof(dist[0]) * (n + 3));
  memset(gap, 0, sizeof(gap[0]) * (n + 3));
  for (i = 0; i < n; i ++ )
    cur[i] = last[i];
  u = s;
  gap[0] = n;
  pre[s] = -1;
  while (dist[s] <= n) {
    bool flag = false;
    for (j = cur[u]; j != -1; j = e[j].next) {
      v = e[j].v;
      if (e[j].c > 0 && dist[u] == dist[v] + 1) {
        flag = true;
        pre[v] = u;
        cur[u] = j;
        u = v;
        break;
      }
    }
    if (flag) {
      if (u == t) {
        int det = INF;
        for (i = u; i != s; i = pre[i])
          det = min(det, e[cur[pre[i]]].c);
        for (i = u; i != s; i = pre[i]) {
          e[cur[pre[i]]].c -= det;
          e[cur[pre[i]] ^ 1].c += det;
        }
        maxflow += det;
        u = s;
      }
    }
    else {
      int mind = n;
      for (j = last[u]; j != -1; j = e[j].next ) {
        v = e[j].v;
        if (e[j].c > 0 && dist[v] < mind) {
          mind = dist[v];
          cur[u] = j;
        }
      }
      if (( -- gap[dist[u]]) == 0) break;
      gap[dist[u] = mind + 1] ++ ;
      if (u != s) u = pre[u];
    }
  }
  return maxflow;
}
```

SAP

```
#include <bits/stdc++.h>
#define cmin(x, y) x = std::min(x, y)
typedef int ft;
const ft inf = 0x3f3f3f;
const int M = 500000+5,N = 20000+5;

struct SAP{
  int y[M],nxt[M],gap[N],fst[N],c[N],pre[N],q[N],dis[N];
  ft f[M];
  int S,T,tot,Tn;
  void init(int s,int t,int tn){
    tot=1;
    memset(fst,0,sizeof (int) * tn);
    memset(dis, 0, sizeof(int) * tn);
    S=s;T=t;Tn=tn;
  }
  void add(int u,int v,ft c1,ft c2=0){
    tot++;y[tot]=v;f[tot]=c1;nxt[tot]=fst[u];fst[u]=tot;
    tot++;y[tot]=u;f[tot]=c2;nxt[tot]=fst[v];fst[v]=tot;
  }
  ft sap(){
```

```
    int u=S,t=1;ft flow=0;
    for(int i = 0; i < t; i++){
      int u=q[i];
      for(int j=fst[u];j;j=nxt[j])
        if(dis[y[j]]>dis[u]+1&&f[j^1])
          q[t++]=y[j],dis[y[j]]=dis[u]+1;
    }
    for(int i = 0; i < Tn; i++)gap[dis[i]]++;
    while(dis[S]<=Tn){
      while(c[u]&&(!f[c[u]]||dis[y[c[u]]]+1!=dis[u]))
        c[u]=nxt[c[u]];
      if(c[u]){
        pre[y[c[u]]]=c[u]^1;
        u=y[c[u]];
        if(u==T){
          ft minf=inf;
          for(int p=pre[T];p;p=pre[y[p]])
            cmin(minf,f[p^1]);
          for(int p=pre[T];p;p=pre[y[p]])
            f[p^1]-=minf,f[p]+=minf;
          flow+=minf;u=S;
        }
      }else {
        if(!(--gap[dis[u]]))break;
        int mind=Tn;
        c[u]=fst[u];
        for(int j=fst[u];j;j=nxt[j])
          if(f[j]&&dis[y[j]]<mind)
            mind=dis[y[j]],c[u]=j;
        dis[u]=mind+1;
        gap[dis[u]]++;
        if(u!=S)u=y[pre[u]];
      }
    }
    return flow;
  }
};
```

Dinic

```
/*
 * dinic 模板
 *
 */
#include <bits/stdc++.h>
namespace dinic {
    const int N = 1e3;
    const int M = 1e4;
    const int INF = 1e9+7;
    int f[N];
    int q[N];
    bool vis[N];
    int h[N];
    int ind[N];
    int t[M], c[M], nt[M], opp[M];
    int cnt;
    int n, m;
    int ST, ED;

    int add(int a, int b, int C) {
      t[cnt] = b;
      nt[cnt] = ind[a];
      ind[a] = cnt;
      c[cnt] = C;
      return cnt++;
    }

    void add1(int a, int b, int c) {
      static int x, y;
      x = add(a, b, c);
      y = add(b, a, 0);
      opp[x] = y; opp[y] = x;
    }

    bool bfs() {
      int l = 0, r = 0;
      q[l] = ST;
      memset(h, 0, sizeof h);
      h[ST] = 1;
      while(l <= r) {
        int x = q[l++];
        for(int k = ind[x]; k != -1; k = nt[k])
          if(!h[t[k]] && c[k] > 0) {
            h[t[k]] = h[x] + 1;
            q[++r] = t[k];
          }
      }
      return h[ED] != 0;
    }

    int dfs(int x, int p) {
      if(x == ED) return p;
      bool flg = false;
      int tot = 0;
      for(int k = ind[x]; k != -1; k = nt[k])
        if(h[t[k]] == h[x] + 1) {
          int d = dfs(t[k], min(p, c[k]));
          if(d) {
            p -= d;
            tot += d;
```

```
            c[k] -= d;
            flg = true;
            c[opp[k]] += d;
          }
        }
      return tot;
    }

    int dinic() {
      int ans = 0, tmp;
      while(bfs()) {
        while(true) {
          int fw = dfs(ST, INF);
          if(!fw) break;
          else ans += fw;
        }
      }
      return ans;
    }
}
```

Dijkstra 费用流

```
/*
 * dijkstra 费用流
 */
#include <bits/stdc++.h>
typedef long long LL;
typedef std::pair<int, int> PII;
#define MP(x, y) std::make_pair(x, y)
#define COST_INF 1e9
#define AA first
#define BB second
#define SZ size()
#define PB(x) push_back(x)
#define cmin(x, y) x = min(x, y)
template <typename T> class MinCostFlow{
  private:
    struct edge{int to;LL cap;T cost;int rev;};

    int V;
    std::vector<std::vector<edge> >adj;
    std::vector<T>pot;

    std::pair<LL,T>dijkstra(int s,int t,LL FLOW_BOUND){
      std::vector<int>used(V,0);
      std::vector<T>dist(V,COST_INF);
      std::vector<PII>path(V,MP(-1,-1));
      std::priority_queue<std::pair<T,int> >Q;
      dist[s]=0;
      Q.push(MP(0,s));
      while(!Q.empty()){
        int x=Q.top().BB;
        Q.pop();
        if(used[x])continue;
        used[x]=1;
        for(int i=0;i<adj[x].SZ;i++)if(adj[x][i].cap>0){
          edge e=adj[x][i];
          int y=e.to;
          T d=dist[x]+e.cost+pot[x]-pot[y];
          if(d<dist[y]&&!used[y]){
            dist[y]=d;
            path[y]=MP(x,i);
            Q.push(MP(-d,y));
          }
        }
      }
      for(int i=0;i<V;i++)
        pot[i]+=dist[i];
      if(dist[t]==COST_INF)
        return MP(0,0);
      LL f=FLOW_BOUND;
      T sum=0;
      int x=t;
      while(x!=s){
        int y=path[x].AA;
        int id=path[x].BB;
        sum+=adj[y][id].cost;
        cmin(f,adj[y][id].cap);
        x=y;
      }
      x=t;
      while(x!=s){
        int y=path[x].AA;
        int id=path[x].BB;
        adj[y][id].cap-=f;
        int id2=adj[y][id].rev;
        adj[x][id2].cap+=f;
        x=y;
      }
      return MP(f,f*sum);
    }
  public:
    MinCostFlow(int n){//[0,n)
      V=n;
      adj.resize(V,std::vector<edge>(0));
      pot.resize(V,0);
    }
    void add_edge(int s,int t,LL f,T c){
      edge e1={t,f,c,(int)adj[t].SZ};
```

```
      edge e2={s,0LL,-c,(int)adj[s].SZ};
      adj[s].PB(e1);
      adj[t].PB(e2);
    }
    std::pair<LL,T>mincostflow(int s,int t,LL FLOW_BOUND=(1LL<<48)){
      std::pair<LL,T>ans=MP(0LL,0);
      while(FLOW_BOUND>0){
        std::pair<LL,T>tmp=dijkstra(s,t,FLOW_BOUND);
        if(tmp.AA==0)break;
        ans.AA+=tmp.AA;
        ans.BB+=tmp.BB;
        FLOW_BOUND-=tmp.AA;
      }
      return ans;
    }
};
```

zkw 费用流

```
/*
 * luoshiying 版本
 * 使用前需要给src, des赋值
 * 调用zkw(src, des, n)中的tn为节点数目，要稍大于总数目
 */
#include <bits/stdc++.h>
#define MOD 0x3f3f3f3f
const int Maxn = 3000;
const int Maxm = 100000;
struct edge
{
  int u, v, c, w, next;
}e[Maxm];
int last[Maxn];
int tot;
int flow, cost, value;
int dist[Maxn], visit[Maxn], src, des;
std::deque<int> Q;
int n, m;

void adde(int u, int v, int c, int w) {
  e[tot].u = u; e[tot].v = v; e[tot].c = c; e[tot].w = w; e[tot].next = last[u]; last[u] = tot++;
  e[tot].u = v; e[tot].v = u; e[tot].c = 0; e[tot].w = -w; e[tot].next = last[v]; last[v] = tot++;
}

int Aug(int u, int m) {
  if(u == des) {
    cost += value * m;
    flow += m;
    return m;
  }
  visit[u] = true;
  int l = m;
  int j, v, c, w;
  for(j = last[u]; j != -1; j = e[j].next) {
    v = e[j].v; c = e[j].c; w = e[j].w;
    if(c && !w && !visit[v]) {
      int del = Aug(v, l < c ? l : c);
      e[j].c -= del; e[j ^ 1].c += del; l -= del;
      if(!l) return m;
    }
  }
  return m - l;
}

bool Modlabel(int src, int des, int n) {
  int i, j, u, v, c, w, del;
  memset(dist, 0x3f, sizeof(dist[0])*(n + 3));
  dist[src] = 0;
  while(!Q.empty()) Q.pop_back();
  Q.push_back(src);
  while(!Q.empty()) {
    u = Q.front(); Q.pop_front();
    for(j = last[u]; j != -1; j = e[j].next) {
      v = e[j].v; c = e[j].c; w = e[j].w;
      if(c && (del = dist[u] + w) < dist[v]) {
        dist[v] = del;
        if(Q.empty() || del <= dist[Q.front()]) {
          Q.push_front(v);
        }
        else {
          Q.push_back(v);
        }
      }
    }
  }
  for(i = 0; i < n; i++) {
    for(j = last[i]; j != -1; j = e[j].next) {
      e[j].w -= dist[e[j].v] - dist[i];
    }
  }
  value += dist[des];
  return dist[des] < MOD;
}

void zkw(int src, int des, int n) {
  value = cost = flow = 0;
  while(Modlabel(src, des, n)){
    do {
```

```
      memset(visit, 0, sizeof(visit[0]) * (n + 3));
    }while(Aug(src, MOD));
  }
}
```

欧拉回路

```
/*
 * 存存在条件为奇数度的点 0 个
 * 有向图所有点入度等于出度
 * 求欧拉回路需要注意访问过的边需要删除
 * 2016 NEERC Moscow G.
 */
#include <stack>
const int N = 1000;
const int M = N * N;
struct Graph {
  int ind[N], vis[N];
  int nt[M], t[M], opp[M], chose[M];
  std::stack<int> stk;

  void dfs(int x) {
    stk.push(x);
    while(stk.size()) {
      x = stk.top(); stk.pop();
      for(int k = ind[x]; ind[x] != -1; ind[x] = nt[k], k = ind[x])
        if(!vis[k]) {
          // 无向图加入下面这句
          vis[k] = vis[opp[k]] = true;
          // ==================
          chose[k] = true;
          stk.push(x);
          stk.push(t[k]);
          break;
        }
    }
    if(ind[x] == -1) {
      // 这里记录 答案
    }
  }
};
```

二分图最大匹配 匈牙利算法

```
/*
 * kuangbin模板 ,
 * 边方向为u到v ,
 * uN为u类节点个数 ,
 * vN为v类节点个数
 */
#include <bits/stdc++.h>
const int MAXN=1000;
int uN, vN;
int g[MAXN][MAXN];
int linker[MAXN];
bool used[MAXN];
bool dfs(int u) {
  int v;
  for(v = 0; v < vN; v++)
    if(g[u][v] && !used[v]) {
      used[v] = true;
      if(linker[v] == -1 || dfs(linker[v])) {
        linker[v] = u;
        return true;
      }
    }
  return false;
}

int hungary() {
  int res = 0;
  int u;
  memset(linker, -1, sizeof(linker));
  for(u = 0; u < uN; u++)
  {
    memset(used, 0, sizeof(used));
    if(dfs(u)) res++;
  }
  return res;
}
```

最小树形图

```
/*
 * 最小树形图 ， 就是给有向带权图中指定一个特殊的点 root ，
 * 求一棵以 root 为根的有向生成树T ，并且T中所有边的总权值最小 .
 * 最小树形图 ( 根固定 ) , O(VE) .
 *
 * 求一个有向图的最小生成树 ，
 * 如果根不固定 ，添加一个根节点与所有点连无穷大的边 ，
 * 或者总边权+1 ，如果求出比2 * MOD大或者返回值为-1 ，则不连通 ；
 * 求根 ，则求和虚拟根相连的结点 .
 *
 * 根据pre的信息能构造出这棵树 ，注意结点必须从[0, n) ，
 * 因为要考虑重新标号建图的统一 ，mytype 根据实际情况确定 .
 */
const int Maxn = 1000;
const double MOD = 1e9;
```

```
struct obj {
  int u, v;
  double w;
} e[Maxn * Maxn];
int n, m;

typedef double mytype;
int visit[Maxn], pre[Maxn], belong[Maxn], ROOT;
mytype inv[Maxn];
mytype dirtree(int n, int m, int root) {
  mytype sum = 0;
  int i, j, k, u, v;
  while (true) {
    for (i = 0; i < n; i++) {
      inv[i] = MOD;
      pre[i] = -1;
      belong[i] = -1;
      visit[i] = -1;
    }
    inv[root] = 0;
    for (i = 0; i < m; i++) { // 除原点外，找每个点的最小入边
      u = e[i].u; v = e[i].v;
      if (u != v) {
        if (e[i].w < inv[v]) {
          inv[v] = e[i].w;
          pre[v] = u;
          if(u == root) ROOT = i; // 记录根所在的边
          // 输出根时利用ROOT − m计算是原图哪个结点
        }
      }
    }
    for (i = 0; i < n; i++) {

      if (inv[i] == MOD) return -1;
    }
    int num = 0;
    for (i = 0; i < n; i++) { // 找圈，收缩圈
      if (visit[i] == -1) {
        j = i;
        for(j = i; j != -1 && visit[j] == -1 && j != root; j = pre[j]) {
          visit[j] = i;
        }
        if (j != -1 && visit[j] == i) {
          for (k = pre[j]; k != j; k = pre[k]) {
            belong[k] = num;
          }
          belong[j] = num++ ;
        }
      }
      sum += inv[i];
    }
    if (num == 0) return sum;
    for (i = 0; i < n; i++){
      if (belong[i] == -1) {
        belong[i] = num ++ ;
      }
    }
    for (i = 0; i < m; i++) { // 重新构图
      e[i].w = e[i].w - inv[e[i].v];
      e[i].v = belong[e[i].v];
      e[i].u = belong[e[i].u];
    }
    n = num;
    root = belong[root];
  }
}
```

哈密尔顿回路

```
/*
 * 设dp[i][j]表示站在i点，盘面状态为j的最短路，
 * 最后答案需要遍历所有j == (1 << n) − 1
 * 注意有的题需要用Floyd预处理最短路
 */
#include <bits/stdc++.h>
const int N = 18;
const int INF = 1e9+7;
int f[N][N];
int dp[N][(1 << N)];
int n, m;
int main() {
  int T; scanf("%d", &T);
  while(T--) {
    scanf("%d%d", &n, &m);
    for(int i = 1; i <= n; i++)
      for(int j = 1; j <= n; j++) f[i][j] = INF;
    for(int i = 1; i <= n; i++) f[i][i] = 0;
    for(int i = 0; i < m; i++) {
      int x, y, z; scanf("%d%d%d", &x, &y, &z);
      // 处理重边
      f[x][y] = std::min(f[x][y], z);
      f[y][x] = std::min(f[y][x], z);
    }
    for(int k = 1; k <= n; k++)
      for(int i = 1; i <= n; i++)
        for(int j = 1; j <= n; j++)
          f[i][j] = std::min(f[i][k] + f[k][j], f[i][j]);

    for(int i = 1; i <= n; i++)
      for(int j = 0; j < 1 << (n + 1); j++) dp[i][j] = INF;
```

```
    dp[1][2] = 0;

    for(int j = 0; j < (1 << (n + 1)); j++) {
      for(int i = 1; i <= n; i++)
        if(((1 << i) & j) != 0) {
          for(int k = 1; k <= n; k++)
            dp[k][(1 << k) | j] =
              std::min(dp[i][j] + f[i][k]
                  , dp[k][(1 << k) | j]);
        }
    }
    int ans = INF;
    for(int i = 1; i <= n; i++) {
      ans = std::min(ans, dp[i][(1 << (n + 1)) - 2] + f[i][1]);
    }
    printf("%d\n", ans);
  }
  return 0;
}
```

增广路费用流

```
/*
 * 增广路版费用流，复杂度O(n^2m)
 * 调用之前需要清空ind并令cnt = 1
 */
#include <bits/stdc++.h>
#define N 3000
#define M 100000
#define INF 0x7fffffff
typedef long long ll;

int ind[N], pre[N], f[N];
bool vis[N];
int bg[M], t[M], nt[M], c[M], op[M], v[M];
int cnt;
int S, T;

int add(int a, int b, int C, int V) { //  不要直接调用
  bg[cnt] = a;
  t[cnt] = b;
  v[cnt] = V;
  c[cnt] = C;
  nt[cnt] = ind[a];
  ind[a] = cnt;
  return cnt++;
}

int ADD(int a, int b, int c, int v) {
  int x = add(a, b, c, v);
  int y = add(b, a, 0, -v);
  op[x] = y; op[y] = x;
  return x;
}

int h[N], q[N];

bool spfa() {
  memset(vis, 0, sizeof vis);
  for(int i = S; i <= T; i++) f[i] = INF;
  for(int i = S; i <= T; i++) pre[i] = -1;
  pre[S] = -1;
  f[S] = 0;
  int l = 0, r = 0; q[l] = S; vis[S] = true;
  while(l <= r)
  {
    int x = q[l % N];
    l++; vis[x] = false;
    for(int k = ind[x]; k != -1; k = nt[k])
      if(c[k] > 0 && f[t[k]] > f[x] + v[k])
      {
        f[t[k]] = f[x] + v[k];
        pre[t[k]] = k;
        if(!vis[t[k]])
        {
          r++;
          q[r % N] = t[k];
          vis[t[k]] = true;
        }
      }
  }
  return pre[T] != -1;
}

int dfs()
{
  int ans = INF;
  int k = pre[T];
  while(k != -1)
  {
    ans = std::min(ans, c[k]);
    k = pre[bg[k]];
  }
  k = pre[T];
  while(k != -1)
  {
    c[k] -= ans;
    c[op[k]] += ans;
    k = pre[bg[k]];
```

```
  }
  return ans;
}

ll dinic() {// 程序入口
  ll ans = 0, tmp;
  while(spfa())
  {
    ans += (ll)f[T] * dfs();
  }
  return ans;
}
```

无向图最小割

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

const int maxn = 400;
const int inf = 1e9;
int cost[maxn][maxn],seq[maxn],len[maxn],n,m,pop,ans;
bool used[maxn];
void Init(){
  int i,j,a,b,c;
  for(i=0;i<n;i++) for(j=0;j<n;j++) cost[i][j]=0;
  for(i=0;i<m;i++){
    scanf("%d %d %d",&a,&b,&c);
    a--; b--;
    cost[a][b]+=c; cost[b][a]+=c;
  }
  pop=n; for(i=0;i<n;i++) seq[i]=i;
}

void Work(){
  ans=inf; int i,j,k,l,mm,sum,pk;
  while(pop > 1){
    for(i=1;i<pop;i++) used[seq[i]]=0; used[seq[0]]=1;
    for(i=1;i<pop;i++) len[seq[i]]=cost[seq[0]][seq[i]];
    pk=0; mm=-inf; k=-1;
    for(i=1;i<pop;i++) if(len[seq[i]] > mm){ mm=len[seq[i]]; k=i; }
    for(i=1;i<pop;i++){
      used[seq[l=k]]=1;
      if(i==pop-2) pk=k;
      if(i==pop-1) break;
      mm=-inf;
      for(j=1;j<pop;j++) if(!used[seq[j]])
        if((len[seq[j]]+=cost[seq[l]][seq[j]]) > mm)
          mm=len[seq[j]], k=j;
    }
    sum=0;
    for(i=0;i<pop;i++) if(i != k) sum+=cost[seq[k]][seq[i]];
    ans=std::min(ans,sum);
    for(i=0;i<pop;i++)
      cost[seq[k]][seq[i]]=cost[seq[i]][seq[k]]+=cost[seq[pk]][seq[i]];
    seq[pk]=seq[--pop];
  }
  printf("%d\n",ans);
}

void solve() {
  int K;
  scanf("%d%d%d", &n, &m, &K);
  if(n == 0 && m == 0 && K == 0) exit(0);
  Init();
  Work();
}

int main() {
  while(true) solve();
  return 0;
}
```

一般图最大匹配 带花树

```
#include <bits/stdc++.h>
/* 求一般图的最大匹配 ， 复杂度O(n^3) .
 * g[i][j] 存放关系图 : i,j 否有边 ， match[i] 存放i所匹配的点
 * 建图开始初始化g .
 * 最终匹配方案为match .
 * 复杂度O(n^3)
 * 点是从1到n的
 * URAL 1099
 */
const int MAXN = 500;
std::deque<int> Q;
bool g[MAXN][MAXN],inque[MAXN],inblossom[MAXN],inpath[MAXN];
int match[MAXN],pre[MAXN],base[MAXN];

// 找公共祖先
int findancestor(int u,int v)
{
  memset(inpath,false,sizeof(inpath));
  while(1)
  {
    u=base[u];
    inpath[u]=true;
    if(match[u]==-1)break;
    u=pre[match[u]];
  }
  while(1)
```

```
  {
    v=base[v];
    if(inpath[v])return v;
    v=pre[match[v]];
  }
}

// 压缩花
void reset(int u,int anc)
{
  while(u!=anc)
  {
    int v=match[u];
    inblossom[base[u]]=1;
    inblossom[base[v]]=1;
    v=pre[v];
    if(base[v]!=anc)pre[v]=match[u];
    u=v;
  }
}

void contract(int u,int v,int n)
{
  int anc=findancestor(u,v);
  memset(inblossom,0, sizeof inblossom);
  reset(u,anc);reset(v,anc);
  if(base[u]!=anc)pre[u]=v;
  if(base[v]!=anc)pre[v]=u;
  for(int i=1;i<=n;i++)
    if(inblossom[base[i]])
    {
      base[i]=anc;
      if(!inque[i])
      {
        Q.push_back(i);
        inque[i]=1;
      }
    }
}

bool bfs(int S,int n)
{
  for(int i=0;i<=n;i++)pre[i]=-1,inque[i]=0,base[i]=i;
  Q.clear();Q.push_back(S);inque[S]=1;
  while(!Q.empty())
  {
    int u=Q.front();Q.pop_front();
    for(int v=1;v<=n;v++)
    {
      if(g[u][v]&&base[v]!=base[u]&&match[u]!=v)
      {
        if(v==S||(match[v]!=-1&&pre[match[v]]!=-1))contract(u,v,n);
        else if(pre[v]==-1)
        {
          pre[v]=u;
          if(match[v]!=-1)Q.push_back(match[v]),inque[match[v]]=1;
          else
          {
            u=v;
            while(u!=-1)
            {
              v=pre[u];
              int w=match[v];
              match[u]=v;
              match[v]=u;
              u=w;
            }
            return true;
          }
        }
      }
    }
  }
  return false;
}

int solve(int n)
{
    memset(match, -1, sizeof match);
    int ans=0;
    for(int i=1;i<=n;i++)
        if(match[i]== -1 && bfs(i,n))
            ans++;
    return ans;
}
```

割点/割边

```
#include <bits/stdc++.h>
using namespace std;
#define MP make_pair
#define AA first
#define BB second
#define SZ size()
#define PB push_back
#define OP begin()
#define ED end()
#define cmin(a, b) a = min(a, b)
typedef std::pair<int, int> PII;
const int N = 50010, M = 1000200, POW = 20, INF = 0x3f3f3f3f;
```

```

class Articulation {
public:
  static const int SIZE = N; // 最大结点个数
  // keyE 为割边集 , keyV 为割点集
  std::vector<PII> keyE;
  // cc[p] 表示结点p在哪些双连通分量中
  std::vector<int> keyV, cc[SIZE];
  int cnt;
  // 对于旧版编译器 , 将上面 cc[SIZE] 改成 vector 的形式
  // 传入结点个数 n 及各结点的出边 e[] , 返回双连通分量的个数 cnt
  int run(int n, const std::vector<int> G[]){
    memset(dfn, use = 0, sizeof(dfn[0]) * n);
    memset(low, cnt = 0, sizeof(low[0]) * n);
    keyE.clear();
    fill_n(cc, n, keyV = std::vector<int>());
    for(int i = 0; i < n; ++i) {
      if(!dfn[i]) dfs(i, 1, G);
    }
    return cnt;
  }
private:
  int dfn[SIZE], low[SIZE], dot[SIZE], use;
  void dfs(int x, int dep, const std::vector<int> G[]){
    int src = 0, out = 1 < dep;
    dot[use++] = x;
    dfn[x] = low[x] = dep;
    for (int i = 0; i < G[x].size(); i++) {
      int y = G[x][i];
      if (!dfn[y]){
        dfs(y, dep + 1, G);
        low[x] = std::min(low[x], low[y]);
        if (low[y] > dfn[x]) keyE.push_back(PII(x, y));
        if (low[y] >= dfn[x]){
          if (++out == 2) keyV.push_back(x);
          while (dot[--use] != y) cc[dot[use]].push_back(cnt);
          cc[x].push_back(cnt);
          cc[y].push_back(cnt++);
        }
      } else if (dfn[y] != dfn[x] - 1 || src++){
        low[x] = std::min(low[x], dfn[y]);
      }
    }
  }
} ;
```

斯坦纳树

```
/*
 * dp[u][i] 表示结点 u 已经和要连通的结点集合 ( 2^k 表示 ) i 连通的最小花费 ,
 * 初始化将 k 个点和 n 个点 dp[u][1<<i] 初始化为最短路 ,
 * , dp[u][0]=0, 加入队列
 * 利用 spfa 求出 dp 数组
 * 状态转移分三部分 :
 *  1) dp[u][su] 利用 dp[u][sub] + dp[u][su^sub] 更新 , sub 为 su 子集
 *  2) dp[u][su] 更新相邻的 dp[v][sv]
 *  3) 将 k 中不属于 su 的点与u连接 , 利用 u, k 的最短路
 */

// hdu 4085
// 求斯坦纳森林，再森林合并为全集
#include <bits/stdc++.h>
#define MOD 0x3f3f3f3f
typedef std::pair<int, int> PII;
#define MP std::make_pair
#define PB push_back
#define cmin(x, y) x = std::min(x, y)
#define SZ size()
#define AA first
#define BB second
using namespace std;
const int Maxn = 55;
const int Maxk = 14;
const int MSta = 1 << 15;
int dp[Maxn][MSta];
vector<PII> g[Maxn];
int gg[Maxn][Maxn];
int mp1[Maxn], mp2[Maxn];
int n, m, K;
vector<int> split[Maxk];
int gao[MSta];
int has (int sta) {
  int i, a = 0, b = 0, u;
  for (i = 0; i < 2 * K; i++) {
    if ((1<<i) & sta) {
      u = mp1[i];
      if (u <= K) a++;
      else b++;
    }
  }
  if (a == b) return true;
  else return false;
}
int ans;
queue<PII> que;
int inque[Maxn][MSta];
int doit() {
  int i, j, k, u, v, w, su, sv, sub, vv, tp;
  int kk = 2 * K;
  for (i = 1; i <= n; i++) {
```

```
    for (j = 0; j < (1<< kk); j++) dp[i][j] = MOD, inque[i][j] = 0;
    dp[i][0] = 0; inque[i][0] = 0;
  }
  while(!que.empty()) que.pop();
  memset(mp2, 0, sizeof(mp2));
  for(i = 1, j = 0; i <= K; i++, j++)mp1[j] = i, mp2[i] = 1<<j;
  for(i = n - K + 1; i <= n; i++, j++)mp1[j] = i, mp2[i] = 1<<j;
  for(i = 0; i < kk; i++) {
    for(u = 1; u <= n; u++) {
      cmin(dp[u][1<<i], gg[u][mp1[i]]);
      inque[u][1<<i] = 1;
      que.push(MP(u, 1<<i));
    }
  }
  while(!que.empty()) {
    u = que.front().AA; su = que.front().BB; que.pop();
    sub = su;
    tp = dp[u][su];
    do {
      sv = sub ^ su;
      cmin(tp, dp[u][sub] + dp[u][sv]);
      sub = (sub - 1) & su;
    }while(sub != su);
    if(tp < dp[u][su]) dp[u][su] = tp;
    for(j = 0; j < g[u].SZ; j++) {
      v = g[u][j].AA; w = g[u][j].BB;
      if(su & mp2[v]) continue;
      sv = su | mp2[v];
      if(dp[v][sv] > dp[u][su] + w) {
        dp[v][sv] = dp[u][su] + w;
        if(!inque[v][sv]) {
          inque[v][sv] = 1;
          que.push(MP(v, sv));
        }
      }
    }
    for(j = 0; j < kk; j++) {
      if(su & (1<<j)) continue;
      sv = su | (1<<j);
      if(dp[u][sv] > dp[u][su] + gg[u][mp1[j]]) {
        dp[u][sv] = dp[u][su] + gg[u][mp1[j]];
        if(!inque[u][sv]) {
          inque[u][sv] = 1;
          que.push(MP(u, sv));
        }
      }
    }
    inque[u][su] = 0;
    }

    for(i = 0; i <= kk; i++) split[i].clear();
    for(i = 0; i < (1<<kk); i++) {
        split[__builtin_popcount(i)].PB(i);
        gao[i] = MOD;
        if(has(i)) {
            for(j = 1; j <= n; j++) {
                cmin(gao[i], dp[j][i]);
            }
        }
    }
    for(i = 1; i <= kk; i++) {
        for(j = 0; j <split[i].SZ; j++) {
            su = split[i][j];
            for(sub = su - 1 & su; sub; sub = sub - 1 & su) {
                sv = su ^ sub;
                cmin(gao[su], gao[sub] + gao[sv]);
                sub = (sub - 1) & su;
            }
        }
    }
    return gao[(1<<kk) - 1];
}
int main() {
    int i,j,k,u,v,w;
    int te;
    scanf ("%d", &te);
    for (int ca = 1; ca <= te; ca ++) {
        scanf ("%d%d%d", &n, &m, &K);
        for (i = 1; i <= n; i++) g[i].clear();
        for (i = 1; i <= n; i++) {
            for (j = 1; j <= n; j++) gg[i][j] = MOD;
            gg[i][i] = 0;
        }
        for (i = 0; i < m; i++) {
            scanf ("%d%d%d", &u, &v, &w);
            cmin (gg[u][v], w);
            cmin (gg[v][u], w);
        }
        for (i = 1; i <= n; i++) {
            for (j = i + 1; j <= n; j++) {
                if (gg[i][j] < MOD) {
                    g[i].PB (MP (j, gg[i][j]) );
                    g[j].PB (MP (i, gg[i][j]) );
                }
            }
        }
        for (k = 1; k <= n; k++) {
            for (i = 1; i <= n; i++) {
                for (j = 1; j <= n; j++) {
                    cmin (gg[i][j], gg[i][k] + gg[k][j]);
```

```
                }
            }
        }
        ans = 0;
        ans = doit();
        if(ans >= MOD) printf("No solution\n");
        else printf ("%d\n", ans);
    }
    return 0;
}
```

字符串

Hash

```
/*
 * 全局 init()
 * get(u64* H, int L, int w) 获取 L 开头 长度为 w 的 hash
 */
const int N = 1e5+100;
typedef unsigned long long ull;
const ull mod = 1e9 + 7;
struct Hash {
  ull P[N];
  ull H[N];
  void init() {
    P[0] = 1;
    for(int i = 1; i < N; i++)
      P[i] = P[i - 1] * mod;
  }

  template <typename T>
    void make(T *a, int n) {
      H[n] = 0;
      for(int i = n - 1; i >= 0; i--)
        H[i] = H[i + 1] * mod + a[i] + 1;
    }

  ull get(int L, int w) {
    return H[L] - H[L + w] * P[w];
  }
} T;
```

KMP

```
/*
 * 初始把 b[n + 1] 赋为 -1
 * Hdu 1711
 */
template<typename T>
void build(T b[], int next[], int m) {
    int k = -1;
    for(int i = 0; i < m; i++) {
        while(k != -1 && b[k + 1] != b[i]) k = next[k];
        if(k + 1 != i && b[k + 1] == b[i]) k++;
        next[i] = k;
    }
}
// 得到 b 在 a 中的第一个匹配位置
template <typename T>
int kmp(T a[], T b[], int next[], int n, int m) {
    int k = -1;
    for(int i = 0; i < n; i++) {
        while(k != -1 && b[k + 1] != a[i]) k = next[k];
        if(b[k + 1] == a[i]) k++;
        if(k == m - 1) return i - m + 2;
    }
    return -1;
}
```

EXKMP

```
/* Hdu 4300
 * S 为主串 , T 为子串 ,
 * LS 为 S 长度 , LT 为 T 长度 ,
 * B[i] 表示 S[i] 匹配了 B[i] 长度的 T
 */
#include <bits/stdc++.h>
const int Maxn = 1e5;
char S[Maxn], T[Maxn];
int Next[Maxn], B[Maxn];

void preExKmp(char T[], int LT, int next[]) {
  int i, ind = 0, k = 1;
  next[0] = LT;
  while(ind + 1 < LT && T[ind + 1] == T[ind]) ind++;
  next[1] = ind;
  for(i = 2; i < LT; i++) {
    if(i <= k + next[k] - 1 && next[i - k] + i < k + next[k])
      next[i] = next[i - k];
    else {
      ind = std::max(0, k + next[k] - i);
      while(ind + i < LT && T[ind + i] == T[ind]) ind++;
      next[i] = ind; k = i;
    }
  }
}

void exKmp(char S[], int LS, char T[], int LT, int next[], int B[]) {
```

```
  int i, ind = 0, k = 0;
  preExKmp(T, LT, next);
  while(ind < LS && ind < LT && T[ind] == S[ind]) ind++;
  B[0] = ind;
  for(i = 1; i < LS; i++) {
    int p = k + B[k] - 1, L = next[i - k];
    if((i - 1) + L < p)
      B[i] = L;
    else {
      ind = std::max(0, p - i + 1);
      while(ind + i < LS && ind < LT && S[ind + i] == T[ind]) ind++;
      B[i] = ind;
      k = i;
    }
  }
}
```

SA

```
/*
 * 倍增算法 , r 为待匹配数组,
 * n 为总长度 +1 , m 为字符范围 , num 保存字符串
 * 使用时注意 num[] 有效位为 [0, n) ,
 * 但是需要将 num[n] = 0 ,
 * 另外 , 对于模板的处理将空串也处理了 ,
 * 作为rank最小的串 ,
 * 因此有效串为 [0, n] 共 n-1 个 ,
 * 在调用da()函数时, 需要调用 da(num, n + 1, m)
 * 对于 sa[], rank[], height[] 数组都将空串考虑在内, 作为 rank 最小的后缀
 * 注意 rank , height 范围从 [0, n]
 */

const int N = 1e5;
namespace SA
{
  int len;
  int num[N];
  // sa[1~n]value(0~n-1) ; rank[0..n-1] value(1..n) ; height[2..n]
  int sa[N], rank[N], height[N];
  int wa[N], wb[N], wv[N], wd[N];

  int cmp(int *r, int a, int b, int x) {
    return r[a] == r[b] && r[a + x] == r[b + x];
  }

  //  倍增算法 r 为待匹配数组  n 为总长度 +1 , m 为字符范围
  void da(int *r, int n, int m) {
    int i, j, k, p, *x = wa, *y = wb, *t;
    for(i = 0; i < m; i++) wd[i] = 0;
    for(i = 0; i < n; i++) wd[x[i] = r[i]]++;
    for(i = 1; i < m; i++) wd[i] += wd[i - 1];
    for(i = n - 1; i >= 0; i--) sa[--wd[x[i]]] = i;
    for(j = 1, p = 1; p < n; j <<= 1, m = p) {
      for(p = 0, i = n - j; i < n; i++) y[p++] = i;
      for(i = 0; i < n; i++) if(sa[i] >= j) y[p++] = sa[i] - j;
      for(i = 0; i < n; i++) wv[i] = x[y[i]];
      for(i = 0; i < m; i++) wd[i] = 0;
      for(i = 0; i < n; i++) wd[wv[i]]++;
      for(i = 1; i < m; i++) wd[i] += wd[i - 1];
      for(i = n - 1; i >= 0; i--) sa[--wd[wv[i]]] = y[i];
      for(t = x, x = y, y = t, p = 1, x[sa[0]] = 0, i = 1; i < n; i++) {
        x[sa[i]] = cmp(y, sa[i - 1], sa[i], j) ? p - 1 : p++;
      }
    }

    for(i = 0, k = 0; i < n; i++) rank[sa[i]] = i;
    for(i = 0; i < n - 1; height[rank[i++]] = k) {
      for(k ? k-- : 0, j = sa[rank[i] - 1]; r[i + k] == r[j + k]; k++);
    }
  }
}
```

DC3

```
// 待排序的字符串放在 r 数组中, 从 r[0] 到 r[n-1], 长度为 n, 且最大值小于 m
// 约定除 r[n-1] 外所有的 r[i] 都大于 0, r[n-1]=0
// 函数结束后, 结果放在 sa 数组中, 从 sa[0] 到 sa[n-1]
#define maxn 10000
#define F(x) ((x)/3+((x)%3==1?0:tb))
#define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
int wa[maxn],wb[maxn],wv[maxn],wss[maxn]; // 必须这么大
int s[maxn*3],sa[maxn*3];
int c0(int *r,int a,int b){return r[a]==r[b]&&r[a+1]==r[b+1]&&r[a+2]==r[b+2];}
int c12(int k,int *r,int a,int b){
  if(k==2) return r[a]<r[b]||r[a]==r[b]&&c12(1,r,a+1,b+1);
  else return r[a]<r[b]||r[a]==r[b]&&wv[a+1]<wv[b+1];
}
void sort(int *r,int *a,int *b,int n,int m){
  int i; for(i=0;i<n;i++) wv[i]=r[a[i]];
  for(i=0;i<m;i++) wss[i]=0; for(i=0;i<n;i++) wss[wv[i]]++;
  for(i=1;i<m;i++) wss[i]+=wss[i-1];
  for(i=n-1;i>=0;i--) b[--wss[wv[i]]]=a[i];
}
void dc3(int *r,int *sa,int n,int m){
  int i,j,*rn=r+n,*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;
  r[n]=r[n+1]=0;
  for(i=0;i<n;i++) if(i%3!=0) wa[tbc++]=i;
  sort(r+2,wa,wb,tbc,m); sort(r+1,wb,wa,tbc,m); sort(r,wa,wb,tbc,m);
  for(p=1,rn[F(wb[0])]=0,i=1;i<tbc;i++)
```

```
    rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;
  if(p<tbc) dc3(rn,san,tbc,p);
  else for(i=0;i<tbc;i++) san[rn[i]]=i;
  for (i=0;i<tbc;i++) if(san[i]<tb) wb[ta++]=san[i]*3;
  if(n%3==1) wb[ta++]=n-1;
  sort(r,wb,wa,ta,m); for(i=0;i<tbc;i++) wv[wb[i]=G(san[i])]=i;
  for(i=0,j=0,p=0;i<ta && j<tbc;p++)
    sa[p]=c12(wb[j]%3,r,wa[i],wb[j])?wa[i++]:wb[j++];
  for(;i<ta;p++) sa[p]=wa[i++]; for(;j<tbc;p++) sa[p]=wb[j++];}
  int main(){
    int n,m=0;  scanf("%d",&n);
    for (int i=0;i<n;i++) scanf("%d",&s[i]),s[i]++,m=max(s[i]+1,m);
    printf("%d\n",m); s[n++]=0; dc3(s,sa,n,m);
    for (int i=0;i<n;i++) printf("%d ",sa[i]);printf("\n");
  }
```

Manacher 最长回文串

```
/*
 * 求以 i 为中心的最长回文串长度 ,
 * 结果保存在 pk[i] 中 , 下标 [0,n) ,
 * 开头 , 末尾 , 字符之间插入 \# , 例如 ababa 变为 \#a\#b\#a\#b\#a\#
 */
#include <bits/stdc++.h>
const int N = 1e5;
int pk[N];
template <typename T>
// [0, n)
void manacher(T *a, int n) {
  int mx = 0;
  int p;
  for(int i = 0; i < n; i++) {
    if(i < mx) pk[i] = std::min(pk[2 * p - i], mx - i);
    else pk[i] = 1;
    while(i + pk[i] < n && i - pk[i] > -1 && a[i + pk[i]] == a[i - pk[i]]) pk[i]++;
    if(i + pk[i] > mx) { p = i; mx = i + pk[i]; }
  }
}
```

最小表示法

```
/*
 * hdu 3374
 */
#include <string>
#include <algorithm>

std::string find(std::string s) {
  int i,j,k,l,N=s.length(); s+=s;
  for(i=0,j=1;j<N;){
    for(k=0;k<N&&s[i+k]==s[j+k];k++);
    if(k>=N) break;
    if(s[i+k]<s[j+k]) j+=k+1;
    else l=i+k,i=j,j=std::max(l,j)+1;
  }
  return s.substr(i,N);
}
```

SAM

```
/*
 * cxlove
 * spoj NSUBSTR
 * 给一个字符串 S,
 * 令 F(x) 表示 S 的所有长度为x的子串中 ,
 * 出现次数的最大值 ,
 * 求 F(1)..F(Length(S))
 * 结点的 len 值表示那一时刻的后缀长度
 */
#include <bits/stdc++.h>
#define inf 100000005
#define M 40
#define N 510005
#define maxn 300005
#define eps 1e-10
#define zero(a) fabs(a)<eps
#define pb(a) push_back(a)
#define mp(a,b) make_pair(a,b)
#define mem(a,b) memset(a,b,sizeof(a))
#define LL unsigned long long
#define MOD 1000000007
#define lson step<<1
#define rson step<<1|1
#define sqr(a) ((a)*(a))
#define Key_value ch[ch[root][1]][0]
#define test puts("OK");
#pragma comment(linker, "/STACK:1024000000,1024000000")
struct SAM {
  SAM *pre,*son[26];
  int len,g;
}que[N],*root,*tail,*b[N];
int tot;
void add(int c,int l) {
  SAM *p=tail,*np=&que[tot++];
  np->len=l;tail=np;
  while(p&&p->son[c]==NULL) p->son[c]=np,p=p->pre;
  if(p==NULL) np->pre=root;
  else {
```

```
    SAM *q=p->son[c];
    if(p->len+1==q->len) np->pre=q;
    else {
      SAM *nq=&que[tot++];
      *nq=*q;
      nq->len=p->len+1;
      np->pre=q->pre=nq;
      while(p&&p->son[c]==q) p->son[c]=nq,p=p->pre;
    }
  }
}
char str[N/2];
int dp[N/2];
int main() {
  while(scanf("%s",str)!=EOF) {
    int n=strlen(str);
    tot=0;
    root=tail=&que[tot++];
    for(int i=0;i<n;i++) add(str[i]-'a',i+1);
    int cnt[N/2];mem(cnt,0);
    for(int i=0;i<tot;i++) cnt[que[i].len]++;
    for(int i=1;i<=n;i++) cnt[i]+=cnt[i-1];
    for(int i=0;i<tot;i++)  b[--cnt[que[i].len]]=&que[i];
    for(int i=0;i<n;i++) (root=root->son[str[i]-'a'])->g++;
    mem(dp,0);
    for(int i=tot-1;i>0;i--) {
      dp[b[i]->len]=std::max(dp[b[i]->len],b[i]->g);
      if(b[i]->pre) b[i]->pre->g+=b[i]->g;
    }
    for(int i=n-1;i>0;i--) dp[i]=std::max(dp[i],dp[i+1]);
    for(int i=1;i<=n;i++) printf("%d\n",dp[i]);
  }
  return 0;
}
```

Trie

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

const int M = 26;
const int N = 2000;
struct trie {
  int to[N][M]; // 节点指针
  int w[256]; // 字母下标
  int fail[N]; // 失配指针
  int cnt;

  trie() {
    // 标注下标
  }

  int newNode() {
    memset(to[cnt], 0, sizeof to[cnt]);
    return cnt++;

  }
  void init() {
    cnt = 0;
    memset(to[0], 0, sizeof to[0]);
  }

  void insert(char *a, int mask) {
    int p = 0;
    while(*a) {
      int v = w[*a];
      if(!to[p][v]) {
        to[p][v] = newNode();
      }
      p = to[p][v];
      a++;
    }
  }

  void ac() {
    std::queue<int>q;
    memset(fail, 0, sizeof fail);
    for(int i = 0; i < 4; i++) if(to[0][i]) {
      q.push(to[0][i]);
    }
    while(q.size()) {
      int p = q.front();
      q.pop();
      for(int i = 0; i < 4; i++) {
        int v = to[p][i];
        if(v) {
          fail[v] = to[fail[p]][i];
          q.push(v);
        }
        else
          to[p][i] =  to[fail[p]][i];
      }
    }
  }
};
```

数学

Fib

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

using namespace std;
const int MOD = 1e9+7;
typedef long long LL;
LL mul(LL a, LL b) {
  LL tot = 0;
  while(b) {
      if(b & 1) tot = (tot + a);
      b >>= 1;
      a <<= 1;
      if(tot >= MOD) tot -= MOD;
      if(a >= MOD) a -= MOD;
    }
    return tot;
};

void fib(LL n, LL &x, LL &y) {
  if (n == 1) {
    x = y = 1;
    return;
  } else if (n & 1) {
    fib(n - 1, y, x);
    y += x; y %= MOD;
  } else {
    LL a, b;
    fib(n >> 1, a, b);
    y = (mul(a, a) % MOD + mul(b, b) % MOD) % MOD;
    x = (mul(a, b) % MOD + mul(a, (b + MOD - a)) % MOD) % MOD;
  }
}
```

矩阵

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

const int MXN = 4;
const int mod = 1e4+7;
template <typename T> class Matrix{
  public:
    int n,m ;
    T a[MXN][MXN];
    Matrix() {memset(a , 0 , sizeof(a));}
    Matrix(int _n , int _m) {
      n = _n , m = _m , memset(a , 0 , sizeof(a));
    };
    T* operator[] (int i) {return a[i];}
    Matrix friend operator * (Matrix A , Matrix B){
      Matrix ans(A.n , B.m);
      for (int i = 0 ; i < A.n ; ++ i)
        for (int j = 0 ; j < A.m ; ++ j)if(A[i][j]!=0)
          for (int k = 0 ; k < B.m ; ++ k)
            ans[i][k]=(ans[i][k]+A[i][j]*B[j][k]) % mod;
      return ans;
    }
};
```

高斯消元

```cpp
// 二进制版本 , hdu5833
#include <bits/stdc++.h>
const int N = 1000;

double a[N][N];
int gauss(int eq, int var) {
  int line = 0, ret = 0;
  for(int i = 0; i < var; i++) {
    int target = line;
    while(target < eq && a[target][i] == 0) target++;
    if(target == eq) { ret++; continue; }
    std::swap(a[target], a[line]);
    for(int j = 0; j < eq; j++) if(j != line && a[j][i]) a[j] ^= a[line];
    line++;
  }
  return ret;
}

// 数值版本
int Gauss(int equ, int var) {
  int i, j, k;
  int max_r; // 当前这列绝对值最大的行 .
  int col; // 当前处理的列
  // 转换为阶梯阵 .
  col = 0; // 当前处理的列
  for(k = 0; k < equ && col < var; k++,col++) {
    // 枚举当前处理的行 .
    // 找到该 col 列元素绝对值最大的那行与第 k 行交换 .
    // ( 为了在除法时减小误差 )
    max_r=k;
    for(i = k + 1; i < equ; i++) {
      if((a[i][col]) > (a[max_r][col])) max_r = i;
    }
    if(max_r != k) {
      // 与第k行交换 .
      for(j=k; j<var+1; j++) swap(a[k][j], a[max_r][j]);
    }
```

```
    if((a[k][col])==0) {
      // 说明该 col 列第 k 行以下全是 0 了 , 则处理当前行的下一列 .
      k--;
      continue;
    }
    for(i=k+1; i<equ; i++) {
      // 枚举要删去的行 .
      if((a[i][col]) > 0) {
        for(j = var; j >= col; j--) {
          a[i][j] -= a[k][j] * (a[i][col] / a[k][col]);
        }
      }
    }
  }
  // 无穷解 , 返回自由变元个数
  // 会改变解的顺序
  if(k < var) {
    for(int i=0; i<equ; i++) {
      if(a[i][i]!= 0) continue;
      else {
        int flag=0;
        for(int j=i+1; j<var; j++) {
          if(a[i][j] != 0) {
            flag = 1;
            for(int k = 0; k <= i; k++) {
              swap(a[k][i], a[i][j]);
            }
            break;
          }
        }
        if(!flag)
          return var - i;
      }
    }
    return var - equ;
  }
  return 0;
}
```

四边形不等式

```
/*
 * Kyb 示意代码 待验证
 */
#define rep(i,a,n) for(int i=(a);i<(int)(n);i++)
#define better true

void dp(int n, int K[][100])  {
  rep(r,1,n){
    rep(i,1,n-r){
      if(r==1)K[i][i+r]=i,dp;
      else rep(k,K[i][i+r-1],K[i+1][i+r])
        if(better)K[i][i+r]=k,dp;
    }
  }
}
```

Java 开根号

```
/*
 * Java 开根号
 * Kyb , 待验证
 */
public static BigInteger getsqrt(BigInteger n) {
  if (n.compareTo(BigInteger.ZERO) <= 0) return n;
  BigInteger x, xx, txx;        xx = x = BigInteger.ZERO;
  for (int t = n.bitLength() / 2; t >= 0; t--) {
    txx = xx.add(x.shiftLeft(t + 1)).add(BigInteger.ONE.shiftLeft(t + t));
    if (txx.compareTo(n) <= 0) {
      x = x.add(BigInteger.ONE.shiftLeft(t));   xx = txx;
    }
  }
  return x;
}
```

Cpp 大数

```
/*
 * C++ 大数运算 , Kyb , 待验证
 */
#include <bits/stdc++.h>
class BigNum {
  public:
    static const int MOD = 100000000;
    static const int BIT = 8, SIZE = 105;
    mutable int n, o;
    long long u[SIZE];
    BigNum() {}
    BigNum(const std::string& s) {
      memset(this, 0, sizeof(BigNum));
      int num = 0, cnt = 1;
      for (int i = s.size() - 1; ~i; i--) {
        if (s[i] == '-') o ^= 1;
        if (s[i] >= '0' && s[i] <= '9') {
          num += (s[i] - '0') * cnt;
          cnt *= 10;
```

```
        if (cnt == MOD) u[n++] = num, num = 0, cnt = 1;
      }
    }
    if (!n || cnt >= 10) u[n++] = num;
    if (!u[0] && n == 1) o = 0;
  }
  BigNum(long long x) {
    memset(this, 0, sizeof(BigNum));
    if (x < 0) o = 1, x = -x;
    do u[n++] = x % MOD; while (x /= MOD);
  }
  operator std::string() const {
    static char s[SIZE * BIT + 10];
    char* c = s + sprintf(s, "%s%d", o ? "-" : "", int(u[n - 1]));
    for (int i = n - 2; ~i; i--) c += sprintf(c, "%0*d", BIT, int(u[i]));
    return s;
  }
  int operator [](int pos) const {
    static int e[BIT] = {1};
    for (static int i = 1; i < BIT; i++) e[i] = e[i - 1] * 10;
    return u[pos / BIT] / e[pos % BIT] % 10;
  }
  int length() const {
    int ret = (n - 1) * BIT + 1;
    for (int x = u[n - 1] / 10; x; x /= 10) ret++;
    return ret;
  }
  friend int cmp(const BigNum& l, const BigNum& r) {
    if (l.o != r.o) return (l.o ? -1 : 1);
    if (l.n != r.n) return (l.o ? -1 : 1) * (l.n - r.n);
    for (int i = l.n - 1; ~i; i--) if (l.u[i] - r.u[i])
      return (l.o ? -1 : 1) * (l.u[i] - r.u[i]);
    return 0;
  }
  // 运算符
  bool operator < (const BigNum& r) const {return cmp(*this, r) < 0;}
  bool operator > (const BigNum& r) const {return cmp(*this, r) > 0;}
  bool operator <=(const BigNum& r) const {return cmp(*this, r) <= 0;}
  bool operator >=(const BigNum& r) const {return cmp(*this, r) >= 0;}
  bool operator ==(const BigNum& r) const {return cmp(*this, r) == 0;}
  bool operator !=(const BigNum& r) const {return cmp(*this, r) != 0;}
  BigNum operator +(const BigNum& r) const {return BigNum(*this) += r;}
  BigNum operator -(const BigNum& r) const {return BigNum(*this) -= r;}
  BigNum operator *(int x) const {return BigNum(*this) *= x;}
  BigNum operator /(int x) const {return BigNum(*this) /= x;}
  BigNum& operator *=(const BigNum& r) {return *this = *this * r;}
  BigNum& operator /=(const BigNum& r) {return *this = *this / r;}
  BigNum& operator %=(const BigNum& r) {return *this = *this % r;}
  BigNum& operator %=(int x) {return *this = *this % x;}
  BigNum operator -() const {
    BigNum s = *this;
    if (s.u[0] || s.n >= 2) s.o ^= 1;
    return s;
  }
  BigNum& operator +=(const BigNum& r) {
    if (r.n == 1 && !r.u[0]) return *this;
    if (r.o ^ o) return r.o ^= 1, *this -= r, r.o ^= 1, *this;
    if (r.n > n) n = r.n;
    for (int i = 0; i < r.n; i++) u[i] += r.u[i];
    for (int i = 0; i < n; i++) if (u[i] >= MOD) u[i + 1]++, u[i] -= MOD;
    if (u[n]) n++;
    return *this;
  }
  BigNum& operator -=(const BigNum& r) {
    if (r.n == 1 && !r.u[0]) return *this;
    if (r.o ^ o) return r.o ^= 1, *this += r, r.o ^= 1, *this;
    if (cmp(*this, r) * (r.o ? -1 : 1) < 0) {
      o ^= 1, n = r.n;
      for (int i = 0; i < r.n; i++) u[i] = r.u[i] - u[i];
    } else {
      for (int i = 0; i < r.n; i++) u[i] = u[i] - r.u[i];
    }
    for (int i = 0; i < n; i++) if (u[i] < 0) u[i + 1]--, u[i] += MOD;
    while (!u[n - 1] && n >= 2) --n;
    if (!u[0] && n == 1) o = 0;
    return *this;
  }
  BigNum operator *(const BigNum& r) const {
    BigNum s = 0;
    if (!u[n - 1] || !r.u[r.n - 1]) return s;
    s.n = r.n + n - 1;
    s.o = r.o ^ o;
    for (int i = 0; i < n; i++) for (int j = 0; j < r.n; j++)
      s.u[i + j] += u[i] * r.u[j];
    for (int i = 0; i < s.n; i++) if (s.u[i] >= MOD) {
      s.u[i + 1] += s.u[i] / MOD;
      s.u[i] %= MOD;
      if (i == s.n - 1) s.n++;
    }
    return s;
  }
  BigNum operator /(const BigNum& r) const {
    BigNum e[35], s = 0, c = 0;
    int m = 0, ro = r.o, lo = o;
    r.o ^= ro, o ^= lo;
    for (e[m] = r; MOD >> ++m; e[m] = e[m - 1] + e[m - 1]);
    for (int i = n - 1; ~i; i--) {
      int tag = 0;
      (s *= MOD) += u[i];
      for (int x = m - 1; ~x; x--) if (s >= e[x]) s -= e[x], tag |= 1 << x;
```

```
      (c *= MOD) += tag;
    }
    r.o ^= ro, o ^= lo;
    if (c.u[0] || c.n >= 2) c.o = r.o ^ o;
    return c;
  }
  BigNum operator %(const BigNum& r) const {
    BigNum e[35], s = 0;
    int m = 0, ro = r.o, lo = o;
    r.o ^= ro, o ^= lo;
    for (e[m] = r; MOD >> ++m; e[m] = e[m - 1] + e[m - 1]);
    for (int i = n - 1; ~i; i--) {
      (s *= MOD) += u[i];
      for (int x = m - 1; ~x; x--) if (s >= e[x]) s -= e[x];
    }
    r.o ^= ro, o ^= lo;
    if (s.u[0] || s.n >= 2) s.o = o;
    return s;
  }
  BigNum& operator *=(int x) {
    if (!x) return *this = 0;
    if (x < 0) o ^= 1, x = -x;
    for (int i = 0; i < n; i++) u[i] *= x;
    for (int i = 0; i < n; i++) if (u[i] >= MOD) {
      u[i + 1] += u[i] / MOD;
      u[i] %= MOD;
      if (i == n - 1) n++;
    }
    if (!u[0] && n == 1) o = 0;
    return *this;
  }
  BigNum& operator /=(int x) {
    if (x < 0) o ^= 1, x = -x;
    for (int i = n - 1; i; u[i--] /= x) u[i - 1] += u[i] % x * MOD;
    for (u[0] /= x; n >= 2; n--) if (u[n - 1]) break;
    if (!u[0] && n == 1) o = 0;
    return *this;
  }
  int operator %(int x) const {
    long long c = 0;
    for (int i = n - 1; ~i; i--) c = (c * MOD + u[i]) % x;
    return (1 - o - o) * int(c);
  }
};
```

线性逆元

```
// 返回 [0, n] 每个数关于 m 的逆元
// x * i = 1 mod m
// let m = i * k + r, where k = m / i, r = m % i
// i = (m - r) * k^-1
// x = -k * r^-1
// time complexity : O(n)

std::vector<LL> linear_inversion(int n, LL m) {
  std::vector<LL> inv(n + 1, 0);
  inv[1] = 1;
  for (int i = 2; i <= n; ++i) {
    inv[i] = (m - m / i) * inv[m % i] % m;
  }
  return inv;
}
```

勒让德定理

```
/*
 * 求出 n! 分解中 , 素因子 x 的个数
 */

typedef long long ll;
ll lrd(ll n, ll x) {
  ll ans = 0;
  for(ll k = x; k <= n; k *= x) {
    ans += n / k;
  }
  return ans;
}
```

欧拉函数

```
/*
 * 对正整数 n ,
 * 欧拉函数是少于或等于n的数中与n互质的数的数目
 * P是素数 :
 * 若p是x的约数 , 则E(xp) = E(x)p
 * p不是x的约数 , 则E(xp) = E(x)E(p) = E(x)(p-1)
 */
const int N = 1e5;
int prime[N];
int phi[N];
bool is_prime[N];

void get_phi() {
  int i, j, k;
  k = 0;
  for(i = 2; i < N; i++) {
    if(is_prime[i] == false) {
      prime[k++] = i;
```

```
      phi[i] = i - 1;
    }
    for(j = 0; j < k && i * prime[j] < N; j++) {
      is_prime[ i * prime[j] ] = true;
      if( i % prime[j] == 0) {
        phi[ i * prime[j] ] = phi[i] * prime[j];
        break;
      }
      else {
        phi[ i * prime[j] ] = phi[i] * ( prime[j] - 1 );
      }
    }
  }
}
```

行列式求值

```
/*
 * 行列式 Mod
 * Kyb 待验证
 */
#include <algorithm>
const int MOD = (int)1e9 + 7;

inline void add(int &a,int b) {
  a += b;
  if (a >= MOD) a -= MOD;
}
int det(int A[16][16],int n) {
  int ret = 1;
  for (int i = 0; i < n; ++ i) {
    if (A[i][i] == 0) {
      for (int j = i + 1; j < n; ++ j) {
        if (A[j][i]) {
          for (int k = i; k < n; ++ k) {
            std::swap(A[i][k],A[j][k]);
          }
          ret = -ret;
          break;
        }
      }
      if (A[i][i] == 0) return 0;
    }
    for (int j = i + 1; j < n; ++ j) {
      int a = 1,b = 0,c = 0,d = 1;
      int x = A[i][i],y = A[j][i];
      while (y) {
        int t = x / y;
        if (t < 0) t += MOD;
        add(a,MOD - c * 1ll * t % MOD);
        add(b,MOD - d * 1ll * t % MOD);
        std::swap(a,c);
        std::swap(b,d);
        x %= y;
        std::swap(x,y);
        ret = -ret;
      }
      for (int k = 0; k < n; ++ k) {
        int q = A[i][k],w = A[j][k];
        A[i][k] = (a * 1ll * q + b * 1ll * w) % MOD;
        A[j][k] = (c * 1ll * q + d * 1ll * w) % MOD;
      }
    }
    ret = A[i][i] * 1ll * ret % MOD;
  }
  if (ret < 0) {
    ret += MOD;
  }
  return ret;
}
```

生成树计数

```
/*
 * 算法引入 :
 * 给定一个无向图 G , 求它生成树的个数 t(G) ;
 *
 * 算法思想 :
 * 1) G 的度数矩阵 D[G] 是一个 n*n 的矩阵 ,
 * 并且满足 : 当 i != j 时 , dij=0 ; 当 i=j 时 , dij 等于 vi 的度数 ;
 * 2) G 的邻接矩阵 A[G] 是一个 n*n 的矩阵 ,
 * 并且满足 : 如果 vi , vj 之间有边直接相连 , 则 aij=1 , 否则为 0 ;
 * 定义图 G 的 Kirchhoff 矩阵 C[G] 为 C[G]=D[G]-A[G] ;
 * Matrix-Tree 定理 : G 的所有不同的生成树的个数等于
 * 其 Kirchhoff 矩阵 C[G] 任何一个 n-1 阶主子式的行列式的绝对值 ;
 * 所谓 n-1 阶主子式 , 就是对于 r(1 <= r <= n) , 将 C[G] 的第r行
 * 第 r 列同时去掉后得到的新矩阵 , 用 Cr[G] 表示 ;
 *
 * Kirchhoff 矩阵的特殊性质 :
 * 1) 对于任何一个图 G, 它的 Kirchhoff 矩阵 C 的行列式总是 0 ,
 * 这是因为 C 每行每列所有元素的和均为 0 ;
 * 2) 如果 G 是不连通的 , 则它的 Kirchhoff 矩阵 C 的任一个主子式的行列式均为 0 ;
 * 3) 如果 G 是一颗树 , 那么它的 Kirchhoff 矩阵 C 的任一个 n-1 阶主子式的行列式均为 1 ;
 *
 * 算法举例 :
 * SPOJ HIGH
 *
 * 题目地址 :
 * http://www.spoj.com/problems/HIGH/
```

```
 *
 * 题目大意 :
 * 一个有 n 座城市的组成国家 , 城市 1 至 n 编号 , 其中一些城市之间可以修建高速公路 ;
 * 需要有选择的修建一些高速公路 , 从而组成一个交通网络 ;
 * 计算有多少种方案 , 使得任意两座城市之间恰好只有一条路径 ;
 **/

#include <bits/stdc++.h>
const int N=15;
typedef long long ll;

int degree[N];
ll C[N][N];
// 生成树计数 : Matrix-Tree 定理
ll det(ll a[][N],int n) {
  ll ret = 1;
  for(int i=1; i<n; i++) {
    for(int j=i+1; j<n; j++)
      while(a[j][i]) {
        ll t = a[i][i] / a[j][i];
        for(int k = i; k < n; k++)
          a[i][k] = (a[i][k] - a[j][k] * t);
        for(int k = i; k < n; k++)
          std::swap(a[i][k], a[j][k]);
        ret = -ret;
      }
    if(a[i][i] == 0)
      return 0;
    ret = ret * a[i][i];
  }
  if(ret < 0)
    ret = -ret;
  return ret;
}

int main() {
  int tcase;
  scanf("%d",&tcase);
  while(tcase--) {
    memset(degree, 0, sizeof(degree) );
    memset(C, 0, sizeof(C));
    int n ,m;
    scanf("%d%d", &n, &m);
    int u, v;
    while(m--) {
      scanf("%d%d", &u, &v);
      u--; v--;
      C[u][v] = C[v][u] = -1;
      degree[u]++; degree[v]++;
    }
    for(int i = 0; i < n; ++i)
      C[i][i] = degree[i];
    printf("%lld\n", det(C, n));
  }
  return 0;
}
```

Simpson 积分

```
/*
 * 自适应 Simpson
 * 验题 :hdu4978
 * Kyb 待验证
 */
#include <cmath>
const double eps = 1e-6;
double f(double x) { return x; }

long double simpson(long double a, long double b) {
  long double c = a + (b - a) / 2;
  return (f(a) + 4 * f(c) + f(b)) * (b - a) / 6;
}
long double asr(long double a, long double b, long double eps, long double A) {
  long double c = a + (b - a) / 2;
  long double L = simpson(a, c), R = simpson(c, b);
  if (std::fabs(L + R - A) < 15 * eps) return L + R + (L + R - A) / 15.;
  return asr(a, c, eps / 2, L) + asr(c, b, eps / 2, R);
}
long double asr(long double a, long double b, long double eps) {
  return asr(a, b, eps, simpson(a, b));
}
```

常用结论

vimrc

syntax on
set nu
set cindent
set shiftwidth=2
set tabstop=2
set expandtab

矩阵乘法

for(i)for(j)if(A[i][j])for(k)C[i][k]+=A[i][j]*B[j][k]

四边形体积公式

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

卡特兰数

$Cat(n) = Comb(2n,n)/(n+1) = Comb(2n,n) - Comb(2n,n+1) = Cat(n-1)*(4n-2)/(n+1)$
n 节点二叉树个数 Cat(n)
正 n 边形划分为 n-2 个三角形的种数 Cat(n-2)
n 个矩阵连乘括号化种数 Cat(n-1)
n 个元素入栈的出栈顺序种数 Cat(n)
n 对括号的合法括号序列个数 Cat(n)

牛顿迭代法

$x_m = \frac{(xf'(x)-f(x))}{f'(x)}$

康托展开

$X = a_n n! + a_{n-1}(n-1)! + ... + a_2 \cdot 2! + a_1 \cdot 1!$
a_i 表示 i 开头的后缀中逆序对的个数

高阶等差数列

$$\sum n = \frac{1}{2}n(n+1) \tag{1}$$
$$\sum n^2 = \frac{1}{6}n(n+1)(2n+1) \tag{2}$$
$$\sum n^3 = \left(\sum n\right)^2 \tag{3}$$
$$\sum n^4 = \left(\sum n^2\right)\frac{1}{5}(3n^2+3n-1) \tag{4}$$
$$\sum n^5 = \left(\sum n\right)^2\frac{1}{3}(2n^2+2n-1) \tag{5}$$
$$\sum n^6 = \left(\sum n^2\right)\frac{1}{7}(3n^4+6n^3-3n+1) \tag{6}$$
$$\sum n^7 = \left(\sum n\right)^2\frac{1}{6}(3n^4+6n^3-n^2-4n+2) \tag{7}$$
$$\sum n^8 = \left(\sum n^2\right)\frac{1}{15}(5n^6+15n^5+5n^4-15n^3-n^2+9n-3) \tag{8}$$

割建图

对于图G=(V,E)中的一个点覆盖是一个集合S V使得每一条边至少有一个端点在S中 .
对于二分图 ,
最小路径覆盖 = | P | - 最大匹配数 .
最小点覆盖数 = 最大匹配数 ,
最大独立顶点集 = 总顶点数 - 最大匹配数 ,

哈密尔顿判定

对于$n >= 3$个点的图G , 如果对于任意u, v 都有 $deg(u)+deg(v) >= n$, 则G一定是哈密尔顿图 .

弦图

设 $next(v)$ 表示 $N(v)$ 中最前的点 . 令 $w*$ 表示所有满足 $A \in B$ 的 w 中最后的一个点 , 判断 $v \cup N(v)$ 是否为极大团 , 只需判断是否存在一个 $w \in w*$, 满足 $Next(w) = v$ 且 $|N(v)|+1 \leq |N(w)|$ 即可 .

五边形数

$\prod_{n=1}^{\infty}(1-x^n) = \sum_{n=0}^{\infty}(-1)^n(1-x^{2n+1})x^{n(3n+1)/2}$
$f(n) = \frac{n(3n-1)}{2}$

重心

半径为 r , 圆心角为 θ 的扇形重心与圆心的距离为 $\frac{4r\sin(\theta/2)}{3\theta}$
半径为 r , 圆心角为 θ 的圆弧重心与圆心的距离为 $\frac{4r\sin^3(\theta/2)}{3(\theta-\sin(\theta))}$

第二类 Bernoulli number

$$B_m = 1 - \sum_{k=0}^{m-1}\binom{m}{k}\frac{B_k}{m-k+1}$$
$$S_m(n) = \sum_{k=1}^{n}k^m = \frac{1}{m+1}\sum_{k=0}^{m}\binom{m+1}{k}B_k n^{m+1-k}$$

Stirling 数

第一类 :n 个元素的项目分作 k 个环排列的方法数目

$$s(n,k) = (-1)^{n+k}|s(n,k)|$$
$$|s(n,0)| = 0$$
$$|s(1,1)| = 1$$
$$|s(n,k)| = |s(n-1,k-1)| + (n-1)*|s(n-1,k)|$$

第二类 :n 个元素的集定义 k 个等价类的方法数

$$S(n,1)=S(n,n)=1$$
$$S(n,k)=S(n-1,k-1)+k*S(n-1,k)$$

数据范围

三角公式

$\sin(a\pm b)=\sin a\cos b\pm\cos a\sin b\quad\cos(a\pm b)=\cos a\cos b\mp\sin a\sin b$

$\tan(a\pm b)=\frac{\tan(a)\pm\tan(b)}{1\mp\tan(a)\tan(b)}\quad\tan(a)\pm\tan(b)=\frac{\sin(a\pm b)}{\cos(a)\cos(b)}$

$\sin(a)+\sin(b)=2\sin(\frac{a+b}{2})\cos(\frac{a-b}{2})\quad\sin(a)-\sin(b)=2\cos(\frac{a+b}{2})\sin(\frac{a-b}{2})$

$\cos(a)+\cos(b)=2\cos(\frac{a+b}{2})\cos(\frac{a-b}{2})\quad\cos(a)-\cos(b)=-2\sin(\frac{a+b}{2})\sin(\frac{a-b}{2})$

$\sin(na)=n\cos^{n-1}a\sin a-\binom{n}{3}\cos^{n-3}a\sin^3a+\binom{n}{5}\cos^{n-5}a\sin^5a-\dots$

$\cos(na)=\cos^n a-\binom{n}{2}\cos^{n-2}a\sin^2a+\binom{n}{4}\cos^{n-4}a\sin^4a-\dots$

积分表

$\int\frac{1}{1+x^2}dx=\tan^{-1}x\quad\int\frac{1}{a^2+x^2}dx=\frac{1}{a}\tan^{-1}\frac{x}{a}$

$\int\frac{x}{a^2+x^2}dx=\frac{1}{2}\ln|a^2+x^2|\quad\int\frac{x^2}{a^2+x^2}dx=x-a\tan^{-1}\frac{x}{a}$

$\int\sqrt{x^2\pm a^2}dx=\frac{1}{2}x\sqrt{x^2\pm a^2}\pm\frac{1}{2}a^2\ln\left|x+\sqrt{x^2\pm a^2}\right|$

$\int\sqrt{a^2-x^2}dx=\frac{1}{2}x\sqrt{a^2-x^2}+\frac{1}{2}a^2\tan^{-1}\frac{x}{\sqrt{a^2-x^2}}$

$\int\frac{x^2}{\sqrt{x^2\pm a^2}}dx=\frac{1}{2}x\sqrt{x^2\pm a^2}\mp\frac{1}{2}a^2\ln\left|x+\sqrt{x^2\pm a^2}\right|$

$\int\frac{1}{\sqrt{x^2\pm a^2}}dx=\ln\left|x+\sqrt{x^2\pm a^2}\right|$

$\int\frac{1}{\sqrt{a^2-x^2}}dx=\sin^{-1}\frac{x}{a}\quad\int\frac{x}{\sqrt{x^2\pm a^2}}dx=\sqrt{x^2\pm a^2}\quad\int\frac{x}{\sqrt{a^2-x^2}}dx=-\sqrt{a^2-x^2}$

$\int\sqrt{ax^2+bx+c}dx=\frac{b+2ax}{4a}\sqrt{ax^2+bx+c}+\frac{4ac-b^2}{8a^{3/2}}\ln\left|2ax+b+2\sqrt{a(ax^2+bx^+c)}\right|$

$\int x^ne^{ax}\,\mathrm{d}x=\frac{x^ne^{ax}}{a}-\frac{n}{a}\int x^{n-1}e^{ax}\,\mathrm{d}x$

$\int\sin^2axdx=\frac{x}{2}-\frac{1}{4a}\sin 2ax\quad\int\sin^3axdx=-\frac{3\cos ax}{4a}+\frac{\cos 3ax}{12a}$

$\int\cos^2axdx=\frac{x}{2}+\frac{\sin 2ax}{4a}\quad\int\cos^3axdx=\frac{3\sin ax}{4a}+\frac{\sin 3ax}{12a}$

$\int\tan axdx=-\frac{1}{a}\ln\cos ax\quad\int\tan^2axdx=-x+\frac{1}{a}\tan ax$

$\int x\cos axdx=\frac{1}{a^2}\cos ax+\frac{x}{a}\sin ax\quad\int x^2\cos axdx=\frac{2x\cos ax}{a^2}+\frac{a^2x^2-2}{a^3}\sin ax$

$\int x\sin axdx=-\frac{x\cos ax}{a}+\frac{\sin ax}{a^2}\quad\int x^2\sin axdx=\frac{2-a^2x^2}{a^3}\cos ax+\frac{2x\sin ax}{a^2}$