

Beijing Jiaotong University , Standard Code Library

SpadeAce, Fengdalu

December 8, 2016

## Contents

基础	2
头文件	2
二进制函数 枚举组合数	2
日期时间公式	2
IO	2
Cpp 快速读入	2
Java 模板	2
数据结构	3
BIT	3
LCA	3
RMQ	4
二维线段树	4
主席树	5
树链剖分	6
树分治	7
Treap	8
Splay	9
KD-Tree	9
KD-Tree 欧几里得距离	10
DLX 精确覆盖	12
DLX 多重覆盖	13
图论	14
2-SAT	14
KM	15
ISAP	16
SAP	16
Dinic	17
Dijkstra 费用流	18
zkw 费用流	19
欧拉回路	20
二分图最大匹配 匈牙利算法	20
最小树形图	20
哈密尔顿回路	21
增广路费用流	22
无向图最小割	23
一般图最大匹配 带花树	23
割点/割边	24
斯坦纳树	25
字符串	27
Hash	27
KMP	27
EXKMP	27
SA	28
DC3	28
Manacher 最长回文串	29
最小表示法	29
SAM	29
Trie	30

数学	31
Fib	31
矩阵	31
高斯消元	31
四边形不等式	32
Java 开根号	32
Cpp 大数	32
线性逆元	34
勒让德定理	34
欧拉函数	34
行列式求值	35
生成树计数	35
Simpson 积分	36
常用结论	36
vimrc	36
矩阵乘法	36
四边形体积公式	37
卡特兰数	37
牛顿迭代法	37
康托展开	37
高阶等差数列	37
割建图	37
哈密尔顿判定	37
弦图	37
五边形数	37
重心	37
第二类 Bernoulli number	37
Stirling 数	37
数据范围	38
三角公式	38
积分表	38

## 基础 头文件

```
1 #include <bits/stdc++.h>
2 typedef long long ll;
3 #define MP make_pair
4 #define AA first
5 #define BB second
6 #define PB push_back
7 #define SZ size
8 #define OP begin()
9 #define ED end()
10 typedef std::pair<int, int> PII;
11 /* ===== */
```

## 二进制函数 枚举组合数

```
1 /* __builtin_clz 前导 0
2 * __builtin_ctz 后缀 0
3 */
4 #define ONES(x) __builtin_popcount(x) // 1 数目
5
6 // 枚举 C(n, k) 所有可能, 复杂度 O(C(n, k))
7 int next_combination(int n, int k) {
8     int ret, b = k & -k, t = (k + b);
9     ret = (((t ^ k) >> 2) / b) | t;
10    if (ret >= (1 << n)) return 0;
11    return ret;
12 }
13
14 void run(int n, int k) {
15     int ik = (1 << k) - 1;
16     do {
17
18     } while(ik = next_combination(n, ik));
19 }
```

## 日期时间公式

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

## IO Cpp 快速读入

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 struct FastIO {
6     static const int S = 65536;
7     char buf[S];
8     int pos, len;
9     bool eof;
10    FILE *in;
11    FastIO(FILE *_in = stdin) {
12        in = _in;
13        pos = len = 0;
14        eof = false;
15    }
16    int nextChar() {
17        if (pos == len)
18            pos = 0, len = fread(buf, 1, S, in);
19        if (pos == len) {eof = true; return -1;}
20        return buf[pos++];
21    }
22    int nextUInt() {
23        int c = nextChar(), x = 0;
24        while (c <= 32) c = nextChar();
25        for (; '0' <= c && c <= '10'; c = nextChar()) x = x * 10 + c - '0';
26        return x;
27    }
28    int nextInt() {
29        int s = 1, c = nextChar(), x = 0;
30        while (c <= 32) c = nextChar();
31        if (c == '-') s = -1, c = nextChar();
32        for (; '0' <= c && c <= '9'; c = nextChar()) x = x * 10 + c - '0';
33        return x * s;
34    }
35    void nextString(char *s) {
36        int c = nextChar();
37        while (c <= 32) c = nextChar();
38        for(; c > 32; c = nextChar()) *s++ = c;
39        *s = 0;
40    }
41 };
```

## Java 模板

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

```

3 import java.io.InputStreamReader;
4 import java.io.IOException;
5 import java.io.PrintWriter;
6 import java.util.Arrays;
7 import java.util.LinkedList;
8 import java.util.Queue;
9 import java.util.StringTokenizer;
10
11 public class Main {
12     public static void main(String[] args) {
13         new Main().run();
14     }
15
16     public void run() {
17         InputReader reader = new InputReader(System.in);
18         PrintWriter writer = new PrintWriter(System.out);
19         try {
20             } catch (Exception e) {
21             } finally {
22                 writer.close();
23             }
24     }
25
26     class InputReader {
27         InputReader(InputStream in) {
28             this.reader = new BufferedReader(new InputStreamReader(in));
29             this.tokenizer = new StringTokenizer("");
30         }
31
32         public String nextToken() throws IOException {
33             while (!tokenizer.hasMoreTokens()) {
34                 tokenizer = new StringTokenizer(reader.readLine());
35             }
36             return tokenizer.nextToken();
37         }
38
39         public int nextInt() throws IOException {
40             return Integer.parseInt(nextToken());
41         }
42
43         private BufferedReader reader;
44         private StringTokenizer tokenizer;
45     }
46
47     private void debug(Object...o) {

```

```

48         System.err.println(Arrays.deepToString(o));
49     }
50 }

```

## 数据结构 BIT

```

1  /*
2   * 支持第 k 大的 BIT
3   * Kyb
4   */
5  inline int LB(int x) {return x & (-x);}
6  const int MXN = 1e5;
7  template <typename T>
8  struct BIT {
9      T _[MXN+5];
10     int n;
11     void init(int m) {
12         n = m + 5;
13         for (int i = 0; i <= n; i++) _[i] = 0;
14     }
15     T query(int w) {
16         T ret = 0;
17         for (w += 3; w > 0; w -= LB(w)) ret += _[w];
18         return ret;
19     }
20     void update(int w, T d) {
21         for (w += 3; w < n; w += LB(w)) _[w] += d;
22     }
23     /*
24     * 待验证
25     */
26     int find_Kth(int k) {          // UESTC_Dagon
27         int ans=0,cnt=0;
28         for(int i=22;i>=0;i--){
29             ans+=1<<i;
30             if(ans>=n||cnt+_[ans]>=k)ans-=1<<i;
31             else cnt+=_[ans];
32         }
33         return ans - 2;
34     }
35 };

```

## LCA

```

1 #include <bits/stdc++.h>
2 const int POW = 18;
3 const int N = 1e5;
4 /*

```

```

5  * p[i][j]: i的第j倍祖先
6  * d[i]: i在树中的深度
7  * edge[N]: 边集合
8  * dfs(u, fa): 求出p, d
9  * lca(a, b): 求出(a, b)的最近公共祖先
10 */
11 int p[N][POW];
12 int d[N];
13 std::vector<int>edge[N];
14 void dfs(int u, int fa){
15     d[u] = d[fa] + 1;
16     p[u][0] = fa;
17     for(int i = 1; i < POW; i++) p[u][i] = p[p[u][i - 1]][i - 1];
18     int sz = edge[u].size();
19     for(int i = 0; i < sz; i++){
20         int v = edge[u][i];
21         if(v == fa) continue;
22         dfs(v, u);
23     }
24 }
25
26 int lca(int a, int b) {
27     if(d[a] > d[b]) std::swap(a, b);
28     if(d[a] < d[b]) {
29         int del = d[b] - d[a];
30         for(int i = 0; i < POW; i++) if(del & (1 << i)) b = p[b][i];
31     }
32     if(a != b) {
33         for(int i = POW - 1; i >= 0; i--)
34             if(p[a][i] != p[b][i])
35                 a = p[a][i], b = p[b][i];
36         a = p[a][0], b = p[b][0];
37     }
38     return a;
39 }

```

## RMQ

```

1  /*
2  * f[i][j]: i开头2^j长度区间的最大值
3  *
4  */
5  #include <algorithm>
6
7  const int N = 1e5;
8  const int POW = 32;
9  int f[N][POW];
10 int mm[N];
11 void init(int n) {
12     for(int j = 0; j < POW; j++)

```

```

13     for(int i = 0; i < n; i++)
14         if(i + (1 << (j + 1)) <= n) {
15             f[i][j + 1] = std::max(f[i][j], f[i + (1 << j)][j]);
16         }
17     mm[0] = -1;
18     for(int i = 1; i < N; i++) {
19         mm[i] = mm[i >> 1] + 1;
20     }
21 }
22
23 int cal(int l, int r) {
24     int k = mm[r - l + 1];
25     return std::max(f[l][k], f[r - (1 << k) + 1][k]);
26 }

```

## 二维线段树

```

1  /*
2  * 点修改区间查询
3  */
4  #include <bits/stdc++.h>
5  const int N = 5000;
6  int c[N][N];
7  int lx[N], ly[N];
8  int n, Q;
9  struct Node
10 {
11     struct node
12     {
13         int a, b;
14         int Min, Max;
15     }f[N];
16     int a, b;
17
18     void update(int x)
19     {
20         f[x].Min = std::min(f[x << 1].Min, f[x << 1 | 1].Min);
21         f[x].Max = std::max(f[x << 1].Max, f[x << 1 | 1].Max);
22     }
23
24     void build(int x, int a, int b)
25     {
26         f[x].a = a; f[x].b = b;
27         if(a < b)
28         {
29             int mid = (a + b) / 2;
30             build(x << 1, a, mid);
31             build(x << 1 | 1, mid + 1, b);
32             update(x);

```

```

33     } else { f[x].Min = f[x].Max = 0; ly[a] = x; }
34 }
35
36 int queryMin(int x, int a, int b)
37 {
38     if(a <= f[x].a && f[x].b <= b) return f[x].Min;
39     else
40     {
41         int mid = (f[x].a + f[x].b) / 2;
42         if(b <= mid) return queryMin(x << 1, a, b);
43         else if(a > mid) return queryMin(x << 1 | 1, a, b);
44         else return std::min(queryMin(x << 1, a, b), queryMin(x << 1 | 1, a, b));
45     }
46 }
47
48 int queryMax(int x, int a, int b)
49 {
50     if(a <= f[x].a && f[x].b <= b) return f[x].Max;
51     else
52     {
53         int mid = (f[x].a + f[x].b) / 2;
54         if(b <= mid) return queryMax(x << 1, a, b);
55         else if(a > mid) return queryMax(x << 1 | 1, a, b);
56         else return std::max(queryMax(x << 1, a, b), queryMax(x << 1 | 1, a, b));
57     }
58 }
59 }f[N];
60
61 void build(int x, int a, int b, int p, int q)
62 {
63     f[x].a = a; f[x].b = b;
64     f[x].build(1, p, q);
65     if(a < b)
66     {
67         int mid = (a + b) / 2;
68         build(x << 1, a, mid, p, q);
69         build(x << 1 | 1, mid + 1, b, p, q);
70     }
71     else lx[a] = x;
72 }
73
74 void update(int x, int y, int c)
75 {
76     x = lx[x]; y = ly[y];
77     for(int X = x; X != 0; X >>= 1)
78         for(int Y = y; Y != 0; Y >>= 1)
79         {
80             if(f[X].f[Y].a == f[X].f[Y].b)
81             {
82                 if(f[X].a == f[X].b)

```

```

83     {
84         f[X].f[Y].Min = f[X].f[Y].Max = c;
85     }
86     else
87     {
88         f[X].f[Y].Min = std::min(f[X << 1].f[Y].Min, f[X << 1 | 1].f[Y].Min);
89         f[X].f[Y].Max = std::max(f[X << 1].f[Y].Max, f[X << 1 | 1].f[Y].Max);
90     }
91 }
92     else { f[X].update(Y); }
93 }
94 }
95
96 int queryMin(int x, int a, int b, int p, int q)
97 {
98     if(a <= f[x].a && f[x].b <= b) return f[x].queryMin(1, p, q);
99     else
100     {
101         int mid = (f[x].a + f[x].b) / 2;
102         if(b <= mid) return queryMin(x << 1, a, b, p, q);
103         else if(a > mid) return queryMin(x << 1 | 1, a, b, p, q);
104         else return std::min(queryMin(x << 1, a, b, p, q), queryMin(x << 1 | 1, a, b, p,
            ↪ q));
105     }
106 }
107
108 int queryMax(int x, int a, int b, int p, int q)
109 {
110     if(a <= f[x].a && f[x].b <= b) return f[x].queryMax(1, p, q);
111     else
112     {
113         int mid = (f[x].a + f[x].b) / 2;
114         if(b <= mid) return queryMax(x << 1, a, b, p, q);
115         else if(a > mid) return queryMax(x << 1 | 1, a, b, p, q);
116         else return std::max(queryMax(x << 1, a, b, p, q), queryMax(x << 1 | 1, a, b, p,
            ↪ q));
117     }
118 }

```

## 主席树

```

1 #include <iostream>
2 #include <cstring>
3 #include <cstdio>
4 #include <algorithm>
5 #include <set>
6 #include <map>
7 #include <vector>
8 #include <queue>
9

```

```

10 typedef std::pair<int, int> PII;
11 #define PB(x) push_back(x)
12 #define SZ size()
13 #define AA first
14 #define BB second
15 #define MP(x, y) make_pair(x, y)
16 namespace ST {
17     /*
18      *   e:   线段树节点数组
19      *   l, r: 左右节点指针
20      *   sum: 区间和
21      *   rt:  主席树根节点
22      *   tot: 总节点个数
23      */
24     struct E {
25         int l, r;
26         int sum;
27     }e[10000000];
28     int rt[100010];
29     int tot;
30
31     /*
32      * 建树，初始化调用，建立一颗空树
33      */
34     void build(int &rt, int l, int r) {
35         rt = tot++;
36         e[rt].sum = 0;
37         if(l == r) return;
38         int mid = (l + r) >> 1;
39         build(e[rt].l, l, mid);
40         build(e[rt].r, mid + 1, r);
41     }
42
43     /*
44      * 插入新节点，向w位置插入节点，主席书中父亲结点为fa
45      */
46     void update(int &rt, int l, int r, int w, int fa) {
47         rt = tot++;
48         e[rt].l = e[fa].l;
49         e[rt].r = e[fa].r;
50         e[rt].sum = e[fa].sum + 1;
51         if(l == r) return;
52         int mid = (l + r) >> 1;
53         if(w <= mid) update(e[rt].l, l, mid, w, e[fa].l);
54         else update(e[rt].r, mid + 1, r, w, e[fa].r);
55     }
56
57     /*
58      * 查询，查询原图中(a, b)点对的信息，其中c = lca(a, b)
59      * 计算时候加入c信息

```

```

60     */
61     int query(int a, int b, int c, int l, int r, int k, int nd) {
62         int mid = (l + r) >> 1;
63         int sum = e[e[a].l].sum + e[e[b].l].sum - e[e[c].l].sum * 2 + (int)(nd <= mid);
64         if(l == r) return 1;
65         if(k <= sum) return query(e[a].l, e[b].l, e[c].l, l, mid, k, nd);
66         else return query(e[a].r, e[b].r, e[c].r, mid + 1, r, k - sum, (nd <= mid ? 1e9 :
        ↪ nd));
67     }
68 };

```

## 树链剖分

```

1  /*
2  * siz[v]表示以v为根的子树的节点数
3  * dep[v]表示v的深度
4  * top[v]表示v所在的重链的顶端节点
5  * fa[v]表示v的父亲
6  * son[v]表示与v在同一重链上的v的儿子节点
7  * w[v]表示v与其父亲节点的连边在线段树中的位置
8  * 初始需要调用cnt1 = cnt2 = cnt3 = 0; dfs1(ROOT, 0); dfs2(ROOT, 1); bt(1, cnt2);
9  * 模板为边带权值，点带权值需要修改query(x, y)
10 * update(x, p, c)的p为线段树中的编号，更新x需要调用w[x]
11 */
12 const int N = 1e5;
13 const int M = 2 * N;
14 typedef long long ll;
15 #define MID(x, y) (((x) + (y)) >> 1)
16 #include <bits/stdc++.h>
17
18 int fa[N], top[N], w[N], son[N], dep[N], sz[N], r[N];
19 int a[N], b[N];
20 ll c[N];
21 int ind[N];
22 int t[M], nt[M];
23 int cnt1, cnt2, cnt3;
24 int n, m;
25
26 struct node
27 {
28     int l, r;
29     int a, b;
30     ll sum;
31 }f[M];
32 int rt;
33
34 void dfs1(int x, int d)
35 {
36     dep[x] = d;
37     son[x] = 0;

```

```

38  sz[x] = 1;
39  for(int k = ind[x]; k != -1; k = nt[k])
40      if(t[k] != fa[x])
41      {
42          fa[t[k]] = x;
43          dfs1(t[k], d + 1);
44          sz[x] += sz[t[k]];
45          if(sz[t[k]] > sz[son[x]]) son[x] = t[k];
46      }
47  }
48
49  void dfs2(int x, int tt)
50  {
51      w[x] = ++cnt2;
52      top[x] = tt;
53      if(son[x]) dfs2(son[x], tt);
54      for(int k = ind[x]; k != -1; k = nt[k]) if(t[k] != fa[x] && t[k] != son[x])
55          dfs2(t[k], t[k]);
56  }
57
58  void add(int a, int b)
59  {
60      t[cnt1] = b;
61      nt[cnt1] = ind[a];
62      ind[a] = cnt1++;
63  }
64
65  void update(int x)
66  {
67      f[x].sum = f[f[x].l].sum + f[f[x].r].sum;
68  }
69
70  int bt(int a, int b)
71  {
72      int x = cnt3++;
73      f[x].a = a; f[x].b = b;
74      if(a < b)
75      {
76          int mid = MID(a, b);
77          f[x].l = bt(a, mid);
78          f[x].r = bt(mid + 1, b);
79          f[x].sum = 0;
80      }
81      else
82      {
83          f[x].sum = 0;
84      }
85      return x;
86  }
87

```

```

88  // 在线段树上查询，不要直接调用
89  ll query(int x, int a, int b)
90  {
91      if(a <= f[x].a && f[x].b <= b) return f[x].sum;
92      int mid = MID(f[x].a, f[x].b);
93      ll ans = 0;
94      if(a <= mid) ans += query(f[x].l, a, b);
95      if(b > mid) ans += query(f[x].r, a, b);
96      return ans;
97  }
98
99  // 单调修改
100 void update(int x, int p, int cc)
101 {
102     if(f[x].a == f[x].b) { f[x].sum = cc; return; }
103     int mid = MID(f[x].a, f[x].b);
104     if(p <= mid) update(f[x].l, p, cc);
105     else update(f[x].r, p, cc);
106     update(x);
107 }
108
109 // 树上查询
110 ll query(int x, int y)
111 {
112     int fx = top[x], fy = top[y];
113     ll sum = 0;
114     while(fx != fy)
115     {
116         if(dep[fx] < dep[fy])
117         {
118             std::swap(x, y);
119             std::swap(fx, fy);
120         }
121         sum += query(rt, w[fx], w[x]);
122         x = fa[top[x]];
123         fx = top[x];
124     }
125     if(dep[x] > dep[y]) std::swap(x, y);
126     if(x == y) return sum;
127     return sum + query(rt, w[son[x]], w[y]);
128 }

```

## 树分治

```

1  /*
2  * 树分治
3  */
4
5  // 点分治
6  #include <bits/stdc++.h>

```



```

7  const int N = 100000 + 5;
8  std::vector<int> edges[N];
9  int n;
10 bool vis[N];
11 int parent[N];
12 int sz[N];
13 int que[N];
14 int balance[N];
15
16 int bfs(int source,int fa = -1) {
17     int qf = 0,qe = 0;
18     que[qe++] = source;
19     parent[source] = fa;
20     while (qf != qe) {
21         int u = que[qf++];
22         sz[u] = 1;
23         balance[u] = 0;
24         for (int v : edges[u]) {
25             if (!vis[v] && parent[u] != v) {
26                 parent[v] = u;
27                 que[qe++] = v;
28             }
29         }
30     }
31     for (int i = qe - 1; i > 0; -- i) {
32         int u = que[i];
33         sz[parent[u]] += sz[u];
34         balance[parent[u]] = std::max(balance[parent[u]],sz[u]);
35     }
36     return qe;
37 }
38
39 void divide(int root) {
40     int tot = bfs(root);
41     for (int i = tot - 1; i > 0; -- i) {
42         int u = que[i];
43         balance[u] = std::max(balance[u],tot - sz[u]);
44         if (balance[u] < balance[root]) {
45             root = u;
46         }
47     }
48     bfs(root);
49
50     // balabalalah
51
52     vis[root] = true;
53     for (int u : edges[root]) {
54         if (!vis[u]) {
55             divide(u);
56         }
57     }
58 }

```

```

57 }
58 }

```

## Treap

```

1  /*
2   * Poj 3481
3   */
4  #include <bits/stdc++.h>
5  using namespace std;
6  #define AA first
7  #define BB second
8  #define MP make_pair
9  #define PII pair<int, int>
10
11 const int N = 1000000+1000, MOD = 1e9+7;
12 const int nil = 0;
13 struct node {
14     int ch[2], key, sz;
15     PII data;
16 } f[N];
17 int cnt, rt;
18
19 void up(int cur) {
20     f[cur].sz = f[f[cur].ch[0]].sz + f[f[cur].ch[1]].sz + 1;
21 }
22
23 int newNode(PII data) {
24     cnt++;
25     f[cnt].ch[0] = f[cnt].ch[1] = 0;
26     f[cnt].data = data;
27     f[cnt].sz = 1;
28     f[cnt].key = rand();
29     return cnt;
30 }
31
32 PII split(int p, int n) {
33     if(n == 0) return MP(nil, p);
34     int sz = f[f[p].ch[0]].sz;
35     if(n == sz) {
36         int x = f[p].ch[0];
37         f[p].ch[0] = nil;
38         up(p);
39         return MP(x, p);
40     }
41     else if(n == sz + 1) {
42         int x = f[p].ch[1];
43         f[p].ch[1] = nil;
44         up(p);
45         return MP(p, x);
46     }
47 }

```

```

46     }
47     else if(n < sz) {
48         PII res = split(f[p].ch[0], n);
49         f[p].ch[0] = res.BB;
50         up(p);
51         return MP(res.AA, p);
52     }
53     else {
54         PII res = split(f[p].ch[1], n - sz - 1);
55         f[p].ch[1] = res.AA;
56         up(p);
57         return MP(p, res.BB);
58     }
59 }
60
61 int merge(int p, int q) {
62     if(p == nil) return q;
63     if(q == nil) return p;
64     if(f[p].key > f[q].key) {
65         f[q].ch[0] = merge(p, f[q].ch[0]);
66         up(q);
67         return q;
68     }
69     else {
70         f[p].ch[1] = merge(f[p].ch[1], q);
71         up(p);
72         return p;
73     }
74 }
75
76 int getrank(int p, int w) {
77     int ans = 0;
78     while(p != nil) {
79         if(w == f[p].data.BB) return ans + f[f[p].ch[0]].sz;
80         if(w < f[p].data.BB) p = f[p].ch[0];
81         else {
82             ans += f[f[p].ch[0]].sz + 1;
83             p = f[p].ch[1];
84         }
85     }
86     return ans;
87 }

```

## Splay

```

1 // 注意初始化内存池和 null 节点
2 /*
3  * hdu3487 hdu1908
4  */
5 #include <algorithm>

```

```

6 const int MAX_NODE = 1e5;
7 struct Node{
8     int rev,size; Node *ch[2],*p;
9     void set(Node*,int); int dir(); void update(); void relax(); void appRev();
10 } nodePool[MAX_NODE],*curNode,*null;
11 Node *newNode(){
12     Node *t=curNode++; t->rev=0, t->size=1;
13     t->ch[0]=t->ch[1]=t->p=null; return t;
14 }
15 struct Splay{
16     Node *root;
17     Splay(){ root=newNode(); root->set(newNode(),0); root->update(); }
18     void rot(Node *t){
19         Node *p=t->p; int d=t->dir();
20         p->relax(); t->relax();
21         if(p==root) root=t;
22         p->set(t->ch[!d],d); p->p->set(t,p->dir()); t->set(p,!d);
23         p->update();
24     }
25     void splay(Node *t,Node *f=null){
26         for(t->relax();t->p!=f;)
27             if(t->p->p==f) rot(t);
28             else t->dir()==t->p->dir()?(rot(t->p),rot(t)):(rot(t),rot(t));
29         t->update();
30     }
31 };
32 void initNull(){ curNode=nodePool;null=curNode++;null->size=0; }
33 void Node::set(Node *t,int _d){ ch[_d]=t; t->p=this; }
34 int Node::dir(){ return this==p->ch[1]; }
35 void Node::update(){ size=ch[0]->size+ch[1]->size+1;}
36 void Node::relax(){ if(rev) ch[0]->appRev(), ch[1]->appRev(), rev=false; }
37 void Node::appRev(){ if(this==null) return; rev^=true; std::swap(ch[0],ch[1]); }

```

## KD-Tree

```

1 /*
2  * KDT 2016 Qingdao 11
3  * 估价函数 :
4  * 欧几里德距离下界 : sqr(max(max(X-x.Max[0],x.Min[0]-X,0)))+ ...
5  * 欧几里德距离上界 : max(sqr(X-x.Max[0]),sqr(x.Min[0]-X))+ ...
6  * 曼哈顿距离下界 : max(X-x.Max[0],0)+max(x.Min[0]-X,0)+ ...
7  * 曼哈顿距离上界 : max(abs(X-x.Max[0]),abs(x.Min[0]-X))+ ...
8  *
9  */
10
11 #include <bits/stdc++.h>
12 using namespace std;
13 typedef long long LL;
14 #define cmin(x, y) x = min(x, y)

```

```

15 #define cmax(x, y) x = max(x, y)
16
17 const int N = 1000000;
18 struct point {
19     LL _[2], op, p;
20     LL& operator [] (int x) { return _[x]; }
21     int operator < (const point &t) const {
22         return p < t.p;
23     }
24 } a[N], ans[N];
25
26 template <typename T> inline T SQ(T x) { return x * x; }
27 inline LL dis(point a, point b) {
28     return SQ(1LL * (a[0] - b[0])) + SQ(1LL * (a[1] - b[1]));
29 }
30 struct node {
31     int l, r, fa;
32     bool vis, has;
33     point p;
34 } b[N];
35 int n, m, cnt;
36
37 inline int getnode() {
38     assert(cnt < N);
39     b[cnt].l = b[cnt].r = -1;
40     b[cnt].vis = b[cnt].has = false;
41     b[cnt].fa = -1;
42     return cnt++;
43 }
44
45 void sol(int u, point pt, point& ans, LL &value, int d) {
46     if(u == -1 || !b[u].has) return;
47     if(b[u].vis) {
48         LL w1 = dis(b[u].p, pt);
49         if(w1 < value || (w1 == value && b[u].p.op < ans.op)) {
50             ans = b[u].p;
51             value = w1;
52         }
53     }
54     int l = b[u].l, r = b[u].r;
55     if(pt[d] > b[u].p[d]) swap(l, r);
56     sol(l, pt, ans, value, d ^ 1);
57     if(1LL * SQ(pt[d] - b[u].p[d]) <= value)
58         sol(r, pt, ans, value, d ^ 1);
59 }
60
61 void build(int &u, int l, int r, int d, int fa=-1) {
62
63     int mid = l;
64     if(l != r) mid += rand() % (r - l);

```

```

65     point p = a[mid];
66     swap(a[r], a[mid]);
67     int j = l;
68     for(int i = l; i < r; i++)
69         if(a[i][d] < p[d]) {
70             swap(a[i], a[j++]);
71         }
72     swap(a[j], a[r]);
73     u = getnode();
74     b[u].fa = fa;
75     b[u].p = a[j];
76
77     if(l < j) build(b[u].l, l, j - 1, d ^ 1, u);
78     if(j < r) build(b[u].r, j + 1, r, d ^ 1, u);
79 }
80
81 void add(int u, point p, int d) {
82     if(u == -1) return;
83     if(b[u].p[0] == p[0] && b[u].p[1] == p[1]) {
84         b[u].has = b[u].vis = true;
85         return ;
86     }
87     if(p[d] > b[u].p[d]) add(b[u].r, p, d ^ 1);
88     else add(b[u].l, p, d ^ 1);
89     b[u].has |= b[u].vis;
90     if(~b[u].l) b[u].has |= b[b[u].l].has;
91     if(~b[u].r) b[u].has |= b[b[u].r].has;
92 }

```

## KD-Tree 欧几里得距离

```

1  /*****
2      Problem: 1941
3      User: Ceva
4      Language: C++
5      Result: Accepted
6      Time:1476 ms
7      Memory:63792 kb
8  *****/
9  /*
10 * 求所有点 距离最大和最小值差值的最小值
11 */
12
13 #include <iostream>
14 #include <cstdio>
15 #include <algorithm>
16 #include <cmath>
17 #include <cstring>
18 #include <string>
19 using namespace std;

```

```

20 #define cmin(x, y) x = min(x, y)
21 #define cmax(x, y) x = max(x, y)
22
23 const int maxn = 1000010;
24 const int inf = 1e9;
25 struct point {
26     int x[2];
27 } p[maxn];
28 struct node {
29     int l, r, p;
30     int mn[2], mx[2];
31 } f[maxn * 2];
32 int n;
33 int cur;
34 int cnt;
35 int mm, nn;
36 int operator < (const point &a, const point &b) { return a.x[cur] < b.x[cur]; }
37
38 void update(int rt) {
39     int id = f[rt].p;
40     int l = f[rt].l, r = f[rt].r;
41     for(int i = 0; i < 2; i++) {
42         f[rt].mn[i] = f[rt].mx[i] = p[id].x[i];
43         if(l) cmin(f[rt].mn[i], f[l].mn[i]), cmax(f[rt].mx[i], f[l].mx[i]);
44         if(r) cmin(f[rt].mn[i], f[r].mn[i]), cmax(f[rt].mx[i], f[r].mx[i]);
45     }
46 }
47
48 int mn(int rt, point P) {
49     int tot = 0;
50     for(int i = 0; i < 2; i++) {
51         int a = 0;
52         if(P.x[i] < f[rt].mn[i]) tot += f[rt].mn[i] - P.x[i];
53         else if(P.x[i] > f[rt].mx[i]) tot += -f[rt].mx[i] + P.x[i];
54         tot += a;
55     }
56     return tot;
57 }
58
59 int mx(int rt, point P) {
60     int tot = 0;
61     for(int i = 0; i < 2; i++) {
62         int a = -inf;
63         cmax(a, abs(P.x[i] - f[rt].mn[i]));
64         cmax(a, abs(P.x[i] - f[rt].mx[i]));
65         tot += a;
66     }
67     return tot;
68 }
69

```

```

70
71 void build(int &rt, int l, int r, int mk) {
72     if(l > r) return;
73     int mid = (l + r) >> 1;
74     rt = ++cnt;
75     cur = mk;
76     nth_element(p + l, p + mid, p + r + 1);
77     //cout << p[rt].x[0] << " " << p[rt].x[1] << " " << l << " " << r << endl;
78     f[rt].p = mid;
79     f[rt].l = f[rt].r = 0;
80     build(f[rt].l, l, mid - 1, mk ^ 1);
81     build(f[rt].r, mid + 1, r, mk ^ 1);
82     update(rt);
83 }
84
85 int getdis(point &x, point &y) {
86     int ans = 0;
87     for(int i = 0; i < 2; i++) ans += abs(x.x[i] - y.x[i]);
88     return ans;
89 }
90
91 void dfs1(int rt, point &P, int cur, int &m) {
92     int l = f[rt].l, r = f[rt].r;
93     point &t = p[f[rt].p];
94     int tot = getdis(t, P);
95     if(tot != 0) cmin(m, tot);
96     int a = mn(l, P), b = mn(r, P);
97     if(a < b) {
98         if(l) if(m > a) dfs1(f[rt].l, P, cur ^ 1, m);
99         if(r) if(m > b) dfs1(f[rt].r, P, cur ^ 1, m);
100     }
101     else {
102         if(r) if(m > b) dfs1(f[rt].r, P, cur ^ 1, m);
103         if(l) if(m > a) dfs1(f[rt].l, P, cur ^ 1, m);
104     }
105 }
106
107 void dfs2(int rt, point &P, int cur, int &m) {
108     int l = f[rt].l, r = f[rt].r;
109     point &t = p[f[rt].p];
110     cmax(m, getdis(t, P));
111     int a = mx(l, P), b = mx(r, P);
112     if(a > b) {
113         if(l) if(m < a) dfs2(f[rt].l, P, cur ^ 1, m);
114         if(r) if(m < b) dfs2(f[rt].r, P, cur ^ 1, m);
115     }
116     else {
117         if(r) if(m < b) dfs2(f[rt].r, P, cur ^ 1, m);
118         if(l) if(m < a) dfs2(f[rt].l, P, cur ^ 1, m);
119     }

```

```

120 }
121
122 int main() {
123     scanf("%d", &n);
124     for(int i = 0; i < n; i++) scanf("%d%d", &p[i].x[0], &p[i].x[1]);
125     int root;
126     build(root, 0, n - 1, 0);
127     int ans = 1e9;
128     cur = 0;
129     cnt = 0;
130     for(int i = 0; i < n; i++) {
131         int mx = 0, mn = inf;
132         dfs2(root, p[i], 0, mx); dfs1(root, p[i], 0, mn);
133         cmin(ans, mx - mn);
134     }
135     printf("%d\n", ans);
136     return 0;
137 }

```

## DLX 精确覆盖

```

1 #include <iostream>
2 #include <cstdio>
3 #include <cstring>
4
5 const int Maxm = 1000;
6 const int Maxn = 1200;
7 int ans[Maxn];
8 struct DLX{
9     struct Node{
10         Node *L, *R, *U, *D;
11         int col, row;
12     } *head, *row[Maxn], *col[Maxm], node[Maxn * Maxm];
13     int colsum[Maxm], cnt;
14
15     /* dancing link
16      * 精确覆盖问题
17      * 可以添加迭代加深优化
18      * 1) 举深度 h
19      * 2) 若当前深度 + predeep > h : return false
20      * 3) mat 下标 1 开始
21      */
22     /*
23     int predeep() {
24         bool vis[Maxm];
25         memset(vis, 0, sizeof(vis));
26         int ret = 0;
27         for (Node *p = head->R; p != head; p = p->R)
28             if (!vis[p->col]) {
29                 ret ++ ;

```

```

30         vis[p->col] ++ ;
31         for (Node *q = p->D; q != p; q = p->D)
32             for (Node *r = q->R; r != q; r = r->R)
33                 vis[r->col] = true;
34         }
35         return ret;
36     }
37     */
38     void init(int mat[][Maxm], int n, int m) {
39         cnt = 0;
40         memset(colsum, 0, sizeof (colsum ));
41         head = &node[cnt ++ ];
42         for(int i = 1; i <= n; i ++ )
43             row[i] = &node[cnt ++ ];
44         for(int j = 1; j <= m; j ++ )
45             col[j] = &node[cnt ++ ];
46         head->D = row[1], row[1]->U = head;
47         head->R = col[1], col[1]->L = head;
48         head->U = row[n], row[n]->D = head;
49         head->L = col[m], col[m]->R = head;
50         head->row = head->col = 0;
51         for(int i = 1; i <= n; i ++ ) {
52             if (i != n) row[i]->D = row[i + 1];
53             if (i != 1) row[i]->U = row[i - 1];
54             row[i]->L = row[i]->R = row[i];
55             row[i]->row = i, row[i]->col = 0;
56         }
57         for(int i = 1; i <= m; i ++ ) {
58             if (i != m) col[i]->R = col[i + 1];
59             if (i != 1) col[i]->L = col[i - 1];
60             col[i]->U = col[i]->D = col[i];
61             col[i]->col = i, col[i]->row = 0;
62         }
63         for(int i = n; i > 0; i -- )
64             for(int j = m; j > 0; j -- )
65                 if(mat[i][j]) {
66                     Node *p = &node[cnt ++ ];
67                     p->R = row[i]->R, row[i]->R->L = p;
68                     p->L = row[i], row[i]->R = p;
69                     p->D = col[j]->D, col[j]->D->U = p;
70                     p->U = col[j], col[j]->D = p;
71                     p->row = i;
72                     p->col = j;
73                     colsum[j] ++ ;
74                 }
75     }
76     void remove(Node *c) {
77         c->L->R = c->R;
78         c->R->L = c->L;
79         for(Node *p = c->D; p != c; p = p->D) {

```

```

80     for(Node *q = p->R; q != p; q = q->R) {
81         q->U->D = q->D;
82         q->D->U = q->U;
83         colsum[q->col] -- ;
84     }
85 }
86 }
87 void resume(Node *c) {
88     for(Node *p = c->U; p != c; p = p->U) {
89         for(Node *q = p->L; q != p; q = q->L) {
90             q->U->D = q;
91             q->D->U = q;
92             colsum[q->col] ++ ;
93         }
94     }
95     col[c->col]->L->R = col[c->col];
96     col[c->col]->R->L = col[c->col];
97 }
98 int dfs(int deep) {
99     if(head->R == head) return deep;
100     Node *p, *q = head->R;
101     for(p = head->R; p != head; p = p->R)
102         if(colsum[p->col] < colsum[q->col])
103             q = p;
104     remove(q);
105     for(p = q->D; p != q; p = p->D) {
106         for(Node* r = p->R; r != p; r = r->R)
107             if (r->col != 0)
108                 remove (col[r->col]);
109         // ----- 可修改区域 -----
110         ans[deep] = p->row;
111         // -----
112         int sta = dfs (deep + 1);
113         if(sta) return sta;
114         for(Node* r = p->L; r != p; r = r->L)
115             if(r->col != 0)
116                 resume (col[r->col]);
117     }
118     resume(q);
119     return false;
120 }
121 } dlx;
122 int mat[Maxn][Maxn];
123 int mem[Maxn][3]; // 记录每行代表哪一格填几
124 // col = 324
125 int n;
126 // 数独填充 (x, y) = v
127 void addline(int x, int y, int v) {
128     int i, j;
129     n++;

```

```

130     mem[n][0] = x;
131     mem[n][1] = y;
132     mem[n][2] = v;
133     for(i = 0; i < Maxm; i++) mat[n][i] = 0;
134     mat[n][x * 9 + y + 1] = 1;
135     mat[n][81 + x * 9 + v] = 1;
136     mat[n][162 + y * 9 + v] = 1;
137     mat[n][243 + (3 * (x / 3) + y / 3) * 9 + v] = 1;
138 }

```

## DLX 多重覆盖

```

1 #include <bits/stdc++.h>
2 #define cmin(x, y) x = std::min(x, y)
3 const int Maxm = 62;
4 const int Maxn = 62;
5
6 int K;
7 struct DLX{
8     struct Node{
9         Node *L, *R, *U, *D;
10        int col, row;
11    } *head, *row[Maxn], *col[Maxm], node[Maxn * Maxm];
12    int colsum[Maxm], cnt;
13    /* dancing link
14     * 重复覆盖问题
15     * 可以添加迭代加深优化
16     * 1) 举深度 h
17     * 2) 若当前深度 + predeep > h : return false
18     * 3) mat 下标 1 开始
19     */
20
21    int predeep(){
22        bool vis[Maxm];
23        Node * p, *q, *r;
24        memset(vis, 0, sizeof(vis));
25        int ret = 0;
26        for(p = head->R; p != head; p = p->R) {
27            if(!vis[p->col]) {
28                ret++;
29                vis[p->col]++;
30                for(q = p->D; q != p; q = q->D) {
31                    for(r = q->R; r != q; r = r->R) {
32                        vis[r->col] = true;
33                    }
34                }
35            }
36        }
37        return ret;
38    }

```

```

39 void init(int mat[][Maxm], int n, int m) {
40     cnt = 0;
41     int i, j;
42     Node * p;
43     memset(colsum, 0, sizeof(colsum));
44     head = &node[cnt++];
45     for(i = 1; i <= n; i++) row[i] = &node[cnt++];
46     for(j = 1; j <= m; j++) col[j] = &node[cnt++];
47     head->D = row[1], row[1]->U = head;
48     head->R = col[1], col[1]->L = head;
49     head->U = row[n], row[n]->D = head;
50     head->L = col[m], col[m]->R = head;
51     head->row = head->col = 0;
52     for(i = 1; i <= n; i++) {
53         if(i != n) row[i]->D = row[i + 1];
54         if(i != 1) row[i]->U = row[i - 1];
55         row[i]->L = row[i]->R = row[i];
56         row[i]->row = i; row[i]->col = 0;
57     }
58     for(i = 1; i <= m; i++) {
59         if(i != m) col[i]->R = col[i + 1];
60         if(i != 1) col[i]->L = col[i - 1];
61         col[i]->U = col[i]->D = col[i];
62         col[i]->col = i; col[i]->row = 0;
63     }
64     for(i = n; i > 0; i--) {
65         for(j = m; j > 0; j--) {
66             if(mat[i][j]) {
67                 p = &node[cnt++];
68                 p->R = row[i]->R, row[i]->R->L = p;
69                 p->L = row[i], row[i]->R = p;
70                 p->D = col[j]->D, col[j]->D->U = p;
71                 p->U = col[j], col[j]->D = p;
72                 p->row = i;
73                 p->col = j;
74                 colsum[j]++;
75             }
76         }
77     }
78 }
79 void remove(Node *c) {
80     Node * p;
81     for(p = c->D; row[p->row] != row[c->row]; p = p->D) {
82         p->R->L = p->L; p->L->R = p->R;
83     }
84 }
85 void resume(Node *c) {
86     Node * p;
87     for(p = c->U; row[p->row] != row[c->row]; p = p->U) {
88         p->L->R = p->R->L = p;

```

```

89     }
90 }
91
92 bool dfs(int deep) {
93     if(head->R == head) {
94         if(deep <= K) return true;
95         return false;
96     }
97     if(deep + predeep() > K) return false;
98     Node *p, *q = head->R, *r;
99     for(p = head->R; p != head; p = p->R) {
100         if(colsum[p->col] < colsum[q->col]) q = p;
101     }
102     for(p = q->D; p != q; p = p->D) {
103         remove(p);
104         for(r = p->R; r != p; r = r->R) {
105             if(r->col != 0) remove(r);
106         }
107         // ----- 可修改区域 -----
108         //         ans[deep] = p->row;
109         // -----
110         int sta = 0;
111         sta = dfs(deep + 1);
112         if(sta) return sta;
113         for(r = p->L; r != p; r = r->L) {
114             if(r->col != 0) resume(r);
115         }
116         resume(p);
117     }
118     return false;
119 }
120 } dlx;

```

## 图论 2-SAT

```

1 #include <iostream>
2 #include <cstdio>
3 #include <iostream>
4 #include <stack>
5 #include <cstring>
6 #include <algorithm>
7 #include <cstring>
8
9 const int N = 500;
10 const int M = 2e6;
11 /*
12  * dfn: 标号数组
13  * low: 回向边数组
14  */

```

```

15 int ind[N];
16 int to[M], nt[M];
17 int dfn[N], low[N];
18 int color[N];
19 bool vis[N];
20 int cnt, num, idn;
21 int ncnt;
22 std::stack<int>s;
23
24 void add(int i, int j) {
25     cnt++;
26     to[cnt] = j;
27     nt[cnt] = ind[i];
28     ind[i] = cnt;
29 }
30
31 void tarjan(int x) {
32     if(dfn[x] != 0) return;
33     low[x] = dfn[x] = ++num;
34     vis[x] = true;
35     s.push(x);
36     for(int k = ind[x]; k != -1; k = nt[k]) {
37         tarjan(to[k]);
38         if(vis[to[k]]) low[x] = std::min(low[x], low[to[k]]);
39     }
40     if(dfn[x] == low[x]) {
41         ncnt++;
42         while(true) {
43             int p = s.top();
44             s.pop();
45             color[p] = ncnt;
46             vis[p] = false;
47             if(p == x) break;
48         }
49     }
50 }
51
52 const int Maxn = 500;
53 const int Maxm = 50000;
54 int n, m;
55 int a[Maxm], b[Maxm], c[Maxm];
56
57 bool check() {
58     memset(dfn, 0, sizeof dfn);
59     memset(low, 0, sizeof low);
60     memset(vis, 0, sizeof vis);
61     for(int i = 0; i < 2 * n; i++) ind[i] = -1;
62     int tot = n;
63     num = idn = 0;
64     cnt = 0;

```

```

65 ncnt = 0;
66 /*
67  * 这里建图
68  * 每个节点分为两个，选和不选
69  * ( $p - > q$ ), 表示选 $p$ 一定要选 $q$ 
70  * 检查每个节点选和不选是否在同一个集合
71  */
72 for(int i = 0; i < 2 * n; i++) tarjan(i);
73 for(int i = 0; i < n; i++) {
74     if(color[i] == color[tot + i]) return false;
75 }
76 return true;
77 }

```

## KM

```

1 /*
2  * 二分图最大权匹配
3  * SJTU
4  * Hdu 2255
5  */
6 #include <bits/stdc++.h>
7 using namespace std;
8
9 namespace graph {
10     const int maxn=400; const int oo=0x7fffffff;
11     int w[maxn][maxn], x[maxn], y[maxn], px[maxn], py[maxn], sy[maxn], slack[maxn];
12     int par[maxn]; int n; int pa[200][2], pb[200][2], n0, m0, na, nb; char s[200][200];
13     void adjust(int v){ sy[v]=py[v]; if (px[sy[v]]!=-2) adjust(px[sy[v]]);}
14     bool find(int v){for (int i=0; i<n; i++){
15         if (py[i]==-1){
16             if (slack[i]>x[v]+y[i]-w[v][i]) slack[i]=x[v]+y[i]-w[v][i], par[i]=v;
17             if (x[v]+y[i]==w[v][i]){
18                 py[i]=v; if (sy[i]==-1){adjust(i); return 1;}
19                 if (px[sy[i]]!=-1) continue; px[sy[i]]=i;
20                 if (find(sy[i])) return 1;
21             }}return 0;}
22     int km(){int i,j,m,flag; for (i=0; i<n; i++) sy[i]=-1, y[i]=0;
23     for (i=0; i<n; i++){x[i]=0; for (j=0; j<n; j++) x[i]=max(x[i], w[i][j]);}
24     for (i=0; i<n; i++){
25         for (j=0; j<n; j++) px[j]=py[j]=-1, slack[j]=oo;
26         px[i]=-2; if (find(i)) continue; flag=false;
27         for (; !flag;){ m=oo;
28             for (j=0; j<n; j++) if (py[j]==-1) m=min(m, slack[j]);
29             for (j=0; j<n; j++){ if (px[j]!=-1) x[j]-=m;
30                 if (py[j]!=-1) y[j]+=m; else slack[j]-=m;}
31             for (j=0; j<n; j++){ if (py[j]==-1&&!slack[j]){
32                 py[j]=par[j];
33                 if (sy[j]==-1){ adjust(j); flag=true; break;}
34                 px[sy[j]]=j; if (find(sy[j])){flag=true; break;}

```



```

35     }}}}
36     int ans=0; for (i=0;i<n;i++) ans+=w[sy[i]][i];return ans;}
37 }
38 using namespace graph;

```

## ISAP

```

1  /*
2  * 用于边数较多的情况, 调用ISAP(intst,inted,intn), n要稍大
3  */
4
5  #include <bits/stdc++.h>
6  const int Maxn = 1000;
7  const int Maxm = Maxn * Maxn;
8  struct node {
9      int u, v, c, next;
10 }e[Maxm];
11 int tot, last[Maxn];
12 void adde(int u, int v, int c, int c1) {
13     e[tot].u = u; e[tot].v = v; e[tot].c = c; e[tot].next = last[u]; last[u] = tot++;
14     e[tot].u = v; e[tot].v = u; e[tot].c = c1; e[tot].next = last[v]; last[v] = tot++;
15 }
16
17 int dist[Maxn], cur[Maxn], gap[Maxn], pre[Maxn];
18 int ISAP(int s, int t, int n) {
19     int i, j, u, v, det;
20     int maxflow = 0;
21     memset(dist, 0, sizeof(dist[0]) * (n + 3));
22     memset(gap, 0, sizeof(gap[0]) * (n + 3));
23     for (i = 0; i < n; i++) {
24         cur[i] = last[i];
25     }
26     u = s;
27     gap[0] = n;
28     pre[s] = -1;
29     while (dist[s] <= n) {
30         bool flag = false;
31         for (j = cur[u]; j != -1; j = e[j].next) {
32             v = e[j].v;
33             if (e[j].c > 0 && dist[u] == dist[v] + 1) {
34                 flag = true;
35                 pre[v] = u;
36                 cur[u] = j;
37                 u = v;
38                 break;
39             }
40         }
41         if (flag) {
42             if (u == t) {
43                 int det = INF;
44                 for (i = u; i != s; i = pre[i])

```

```

44         det = min(det, e[cur[pre[i]]].c);
45         for (i = u; i != s; i = pre[i]) {
46             e[cur[pre[i]]].c -= det;
47             e[cur[pre[i]] ^ 1].c += det;
48         }
49         maxflow += det;
50         u = s;
51     }
52 }
53 else {
54     int mind = n;
55     for (j = last[u]; j != -1; j = e[j].next) {
56         v = e[j].v;
57         if (e[j].c > 0 && dist[v] < mind) {
58             mind = dist[v];
59             cur[u] = j;
60         }
61     }
62     if ((-- gap[dist[u]]) == 0) break;
63     gap[dist[u] = mind + 1]++;
64     if (u != s) u = pre[u];
65 }
66 }
67 return maxflow;
68 }

```

## SAP

```

1  #include <bits/stdc++.h>
2  #define cmin(x, y) x = std::min(x, y)
3  typedef int ft;
4  const ft inf = 0x3f3f3f;
5  const int M = 500000+5, N = 20000+5;
6
7  struct SAP{
8      int y[M],nxt[M],gap[N],fst[N],c[N],pre[N],q[N],dis[N];
9      ft f[M];
10     int S,T,tot,Tn;
11     void init(int s,int t,int tn){
12         tot=1;
13         memset(fst,0,sizeof (int) * tn);
14         memset(dis, 0, sizeof(int) * tn);
15         S=s;T=t;Tn=tn;
16     }
17     void add(int u,int v,ft c1,ft c2=0){
18         tot++;y[tot]=v;f[tot]=c1;nxt[tot]=fst[u];fst[u]=tot;
19         tot++;y[tot]=u;f[tot]=c2;nxt[tot]=fst[v];fst[v]=tot;
20     }
21     ft sap(){

```

```

22     int u=S,t=1;ft flow=0;
23     for(int i = 0; i < t; i++){
24         int u=q[i];
25         for(int j=fst[u];j;j=nxt[j])
26             if(dis[y[j]]>dis[u]+1&&f[j^1])
27                 q[t++]=y[j],dis[y[j]]=dis[u]+1;
28     }
29     for(int i = 0; i < Tn; i++)gap[dis[i]]++;
30     while(dis[S]<=Tn){
31         while(c[u]&&(!f[c[u]]||dis[y[c[u]]+1!=dis[u]))
32             c[u]=nxt[c[u]];
33         if(c[u]){
34             pre[y[c[u]]]=c[u]^1;
35             u=y[c[u]];
36             if(u==T){
37                 ft minf=inf;
38                 for(int p=pre[T];p;p=pre[y[p]])
39                     cmin(minf,f[p^1]);
40                 for(int p=pre[T];p;p=pre[y[p]])
41                     f[p^1]-=minf,f[p]+=minf;
42                 flow+=minf;u=S;
43             }
44         }else {
45             if(!(--gap[dis[u]]))break;
46             int mind=Tn;
47             c[u]=fst[u];
48             for(int j=fst[u];j;j=nxt[j])
49                 if(f[j]&&dis[y[j]]<mind)
50                     mind=dis[y[j]],c[u]=j;
51             dis[u]=mind+1;
52             gap[dis[u]]++;
53             if(u!=S)u=y[pre[u]];
54         }
55     }
56     return flow;
57 }
58 };

```

## Dinic

```

1  /*
2  * dinic 模板
3  *
4  */
5  #include <bits/stdc++.h>
6  namespace dinic {
7      const int N = 1e3;
8      const int M = 1e4;
9      const int INF = 1e9+7;
10     int f[N];

```

```

11     int q[N];
12     bool vis[N];
13     int h[N];
14     int ind[N];
15     int t[M], c[M], nt[M], opp[M];
16     int cnt;
17     int n, m;
18     int ST, ED;
19
20     int add(int a, int b, int C) {
21         t[cnt] = b;
22         nt[cnt] = ind[a];
23         ind[a] = cnt;
24         c[cnt] = C;
25         return cnt++;
26     }
27
28     void add1(int a, int b, int c) {
29         static int x, y;
30         x = add(a, b, c);
31         y = add(b, a, 0);
32         opp[x] = y; opp[y] = x;
33     }
34
35     bool bfs() {
36         int l = 0, r = 0;
37         q[l] = ST;
38         memset(h, 0, sizeof h);
39         h[ST] = 1;
40         while(l <= r) {
41             int x = q[l++];
42             for(int k = ind[x]; k != -1; k = nt[k])
43                 if(!h[t[k]] && c[k] > 0) {
44                     h[t[k]] = h[x] + 1;
45                     q[++r] = t[k];
46                 }
47         }
48         return h[ED] != 0;
49     }
50
51     int dfs(int x, int p) {
52         if(x == ED) return p;
53         bool flg = false;
54         int tot = 0;
55         for(int k = ind[x]; k != -1; k = nt[k])
56             if(h[t[k]] == h[x] + 1) {
57                 int d = dfs(t[k], min(p, c[k]));
58                 if(d) {
59                     p -= d;
60                     tot += d;

```

```

61         c[k] -= d;
62         flg = true;
63         c[opp[k]] += d;
64     }
65 }
66 return tot;
67 }
68
69 int dinic() {
70     int ans = 0, tmp;
71     while(bfs()) {
72         while(true) {
73             int fw = dfs(ST, INF);
74             if(!fw) break;
75             else ans += fw;
76         }
77     }
78     return ans;
79 }
80 }

```

## Dijkstra 费用流

```

1  /*
2  * dijkstra 费用流
3  */
4  #include <bits/stdc++.h>
5  typedef long long LL;
6  typedef std::pair<int, int> PII;
7  #define MP(x, y) std::make_pair(x, y)
8  #define COST_INF 1e9
9  #define AA first
10 #define BB second
11 #define SZ size()
12 #define PB(x) push_back(x)
13 #define cmin(x, y) x = min(x, y)
14 template <typename T> class MinCostFlow{
15 private:
16     struct edge{int to;LL cap;T cost;int rev;};
17
18     int V;
19     std::vector<std::vector<edge> >adj;
20     std::vector<T>pot;
21
22     std::pair<LL,T>dijkstra(int s,int t,LL FLOW_BOUND){
23         std::vector<int>used(V,0);
24         std::vector<T>dist(V,COST_INF);
25         std::vector<PII>path(V,MP(-1,-1));
26         std::priority_queue<std::pair<T,int> >Q;
27         dist[s]=0;

```

```

28     Q.push(MP(0,s));
29     while(!Q.empty()){
30         int x=Q.top().BB;
31         Q.pop();
32         if(used[x])continue;
33         used[x]=1;
34         for(int i=0;i<adj[x].SZ;i++){if(adj[x][i].cap>0){
35             edge e=adj[x][i];
36             int y=e.to;
37             T d=dist[x]+e.cost+pot[x]-pot[y];
38             if(d<dist[y]&&!used[y]){
39                 dist[y]=d;
40                 path[y]=MP(x,i);
41                 Q.push(MP(-d,y));
42             }
43         }
44     }
45     for(int i=0;i<V;i++){
46         pot[i]+=dist[i];
47         if(dist[t]==COST_INF)
48             return MP(0,0);
49     LL f=FLOW_BOUND;
50     T sum=0;
51     int x=t;
52     while(x!=s){
53         int y=path[x].AA;
54         int id=path[x].BB;
55         sum+=adj[y][id].cost;
56         cmin(f,adj[y][id].cap);
57         x=y;
58     }
59     x=t;
60     while(x!=s){
61         int y=path[x].AA;
62         int id=path[x].BB;
63         adj[y][id].cap-=f;
64         int id2=adj[y][id].rev;
65         adj[x][id2].cap+=f;
66         x=y;
67     }
68     return MP(f,sum);
69 }
70 public:
71     MinCostFlow(int n){//[0,n)
72         V=n;
73         adj.resize(V,std::vector<edge>(0));
74         pot.resize(V,0);
75     }
76     void add_edge(int s,int t,LL f,T c){
77         edge e1={t,f,c,(int)adj[t].SZ};

```

```

78     edge e2={s,0LL,-c,(int)adj[s].SZ};
79     adj[s].PB(e1);
80     adj[t].PB(e2);
81 }
82 std::pair<LL,T>mincostflow(int s,int t,LL FLOW_BOUND=(1LL<<48)){
83     std::pair<LL,T>ans=MP(0LL,0);
84     while(FLOW_BOUND>0){
85         std::pair<LL,T>tmp=dijkstra(s,t,FLOW_BOUND);
86         if(tmp.AA==0)break;
87         ans.AA+=tmp.AA;
88         ans.BB+=tmp.BB;
89         FLOW_BOUND-=tmp.AA;
90     }
91     return ans;
92 }
93 };

```

## zkw 费用流

```

1  /*
2   * luoshiying 版本
3   * 使用前需要给src,des赋值
4   * 调用zkw(src,des,n)中的tn为节点数目, 要稍大于总数目
5   */
6  #include <bits/stdc++.h>
7  #define MOD 0x3f3f3f3f
8  const int Maxn = 3000;
9  const int Maxm = 100000;
10 struct edge
11 {
12     int u, v, c, w, next;
13 }e[Maxm];
14 int last[Maxn];
15 int tot;
16 int flow, cost, value;
17 int dist[Maxn], visit[Maxn], src, des;
18 std::deque<int> Q;
19 int n, m;
20
21 void adde(int u, int v, int c, int w) {
22     e[tot].u = u; e[tot].v = v; e[tot].c = c; e[tot].w = w; e[tot].next = last[u]; last[u]
        ↳ = tot++;
23     e[tot].u = v; e[tot].v = u; e[tot].c = 0; e[tot].w = -w; e[tot].next = last[v];
        ↳ last[v] = tot++;
24 }
25
26 int Aug(int u, int m) {
27     if(u == des) {
28         cost += value * m;
29         flow += m;

```

```

30     return m;
31 }
32 visit[u] = true;
33 int l = m;
34 int j, v, c, w;
35 for(j = last[u]; j != -1; j = e[j].next) {
36     v = e[j].v; c = e[j].c; w = e[j].w;
37     if(c && !w && !visit[v]) {
38         int del = Aug(v, l < c ? l : c);
39         e[j].c -= del; e[j ^ 1].c += del; l -= del;
40         if(!l) return m;
41     }
42 }
43 return m - l;
44 }
45
46 bool Modlabel(int src, int des, int n) {
47     int i, j, u, v, c, w, del;
48     memset(dist, 0x3f, sizeof(dist[0])*(n + 3));
49     dist[src] = 0;
50     while(!Q.empty()) Q.pop_back();
51     Q.push_back(src);
52     while(!Q.empty()) {
53         u = Q.front(); Q.pop_front();
54         for(j = last[u]; j != -1; j = e[j].next) {
55             v = e[j].v; c = e[j].c; w = e[j].w;
56             if(c && (del = dist[u] + w) < dist[v]) {
57                 dist[v] = del;
58                 if(Q.empty() || del <= dist[Q.front()]) {
59                     Q.push_front(v);
60                 }
61                 else {
62                     Q.push_back(v);
63                 }
64             }
65         }
66     }
67     for(i = 0; i < n; i++) {
68         for(j = last[i]; j != -1; j = e[j].next) {
69             e[j].w -= dist[e[j].v] - dist[i];
70         }
71     }
72     value += dist[des];
73     return dist[des] < MOD;
74 }
75
76 void zkw(int src, int des, int n) {
77     value = cost = flow = 0;
78     while(Modlabel(src, des, n)){
79         do {

```

```
80     memset(visit, 0, sizeof(visit[0]) * (n + 3));
81     }while(Aug(src, MOD));
82 }
83 }
```

## 欧拉回路

```
1  /*
2  * 存在条件为奇数度的点 0 个
3  * 有向图所有点入度等于出度
4  * 求欧拉回路需要注意访问过的边需要删除
5  * 2016 NEERC Moscow G.
6  */
7  #include <stack>
8  const int N = 1000;
9  const int M = N * N;
10 struct Graph {
11     int ind[N], vis[N];
12     int nt[M], t[M], opp[M], chose[M];
13     std::stack<int> stk;
14
15     void dfs(int x) {
16         stk.push(x);
17         while(stk.size()) {
18             x = stk.top(); stk.pop();
19             for(int k = ind[x]; ind[x] != -1; ind[x] = nt[k], k = ind[x])
20                 if(!vis[k]) {
21                     // 无向图加入下面这句
22                     vis[k] = vis[opp[k]] = true;
23                     // =====
24                     chose[k] = true;
25                     stk.push(x);
26                     stk.push(t[k]);
27                     break;
28                 }
29         }
30         if(ind[x] == -1) {
31             // 这里记录 答案
32         }
33     }
34 };
```

## 二分图最大匹配 匈牙利算法

```
1  /*
2  *  kuangbin模板 ,
3  *  边方向为u到v ,
4  *  uN为u类节点个数 ,
5  *  vN为v类节点个数
```

```
6  */
7  #include <bits/stdc++.h>
8  const int MAXN=1000;
9  int uN, vN;
10 int g[MAXN][MAXN];
11 int linker[MAXN];
12 bool used[MAXN];
13 bool dfs(int u) {
14     int v;
15     for(v = 0; v < vN; v++)
16         if(g[u][v] && !used[v]) {
17             used[v] = true;
18             if(linker[v] == -1 || dfs(linker[v])) {
19                 linker[v] = u;
20                 return true;
21             }
22         }
23     return false;
24 }
25
26 int hungary() {
27     int res = 0;
28     int u;
29     memset(linker, -1, sizeof(linker));
30     for(u = 0; u < uN; u++)
31     {
32         memset(used, 0, sizeof(used));
33         if(dfs(u)) res++;
34     }
35     return res;
36 }
```

## 最小树形图

```
1  /*
2  * 最小树形图 , 就是给有向带权图中指定一个特殊的点 root ,
3  * 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 .
4  * 最小树形图 ( 根固定 ) ,  $O(VE)$  .
5  *
6  * 求一个有向图的最小生成树 ,
7  * 如果根不固定 , 添加一个根节点与所有点连无穷大的边 ,
8  * 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 ;
9  * 求根 , 则求和虚拟根相连的结点 .
10 *
11 * 根据pre的信息能构造出这棵树 , 注意结点必须从[0, n) ,
12 * 因为要考虑重新标号建图的统一 , mytype 根据实际情况确定 .
13 */
14 const int Maxn = 1000;
15 const double MOD = 1e9;
```

```

16 struct obj {
17     int u, v;
18     double w;
19 } e[Maxn * Maxn];
20 int n, m;
21
22 typedef double mytype;
23 int visit[Maxn], pre[Maxn], belong[Maxn], ROOT;
24 mytype inv[Maxn];
25 mytype dirtree(int n, int m, int root) {
26     mytype sum = 0;
27     int i, j, k, u, v;
28     while (true) {
29         for (i = 0; i < n; i++) {
30             inv[i] = MOD;
31             pre[i] = -1;
32             belong[i] = -1;
33             visit[i] = -1;
34         }
35         inv[root] = 0;
36         for (i = 0; i < m; i++) { // 除原点外, 找每个点的最小入边
37             u = e[i].u; v = e[i].v;
38             if (u != v) {
39                 if (e[i].w < inv[v]) {
40                     inv[v] = e[i].w;
41                     pre[v] = u;
42                     if (u == root) ROOT = i; // 记录根所在的边
43                     // 输出根时利用  $ROOT - m$  计算是原图哪个结点
44                 }
45             }
46         }
47         for (i = 0; i < n; i++) {
48
49             if (inv[i] == MOD) return -1;
50         }
51         int num = 0;
52         for (i = 0; i < n; i++) { // 找圈, 收缩圈
53             if (visit[i] == -1) {
54                 j = i;
55                 for (j = i; j != -1 && visit[j] == -1 && j != root; j = pre[j]) {
56                     visit[j] = i;
57                 }
58                 if (j != -1 && visit[j] == i) {
59                     for (k = pre[j]; k != j; k = pre[k]) {
60                         belong[k] = num;
61                     }
62                     belong[j] = num++;
63                 }
64             }
65             sum += inv[i];

```

```

66     }
67     if (num == 0) return sum;
68     for (i = 0; i < n; i++) {
69         if (belong[i] == -1) {
70             belong[i] = num++;
71         }
72     }
73     for (i = 0; i < m; i++) { // 重新构图
74         e[i].w = e[i].w - inv[e[i].v];
75         e[i].v = belong[e[i].v];
76         e[i].u = belong[e[i].u];
77     }
78     n = num;
79     root = belong[root];
80 }
81 }

```

### 哈密尔顿回路

```

1  /*
2   * 设  $dp[i][j]$  表示站在  $i$  点, 盘面状态为  $j$  的最短路,
3   * 最后答案需要遍历所有  $j == (1 < n) - 1$ 
4   * 注意有的题需要用 Floyd 预处理最短路
5   */
6  #include <bits/stdc++.h>
7  const int N = 18;
8  const int INF = 1e9+7;
9  int f[N][N];
10 int dp[N][(1 << N)];
11 int n, m;
12 int main() {
13     int T; scanf("%d", &T);
14     while(T--) {
15         scanf("%d%d", &n, &m);
16         for(int i = 1; i <= n; i++)
17             for(int j = 1; j <= n; j++) f[i][j] = INF;
18         for(int i = 1; i <= n; i++) f[i][i] = 0;
19         for(int i = 0; i < m; i++) {
20             int x, y, z; scanf("%d%d%d", &x, &y, &z);
21             // 处理重边
22             f[x][y] = std::min(f[x][y], z);
23             f[y][x] = std::min(f[y][x], z);
24         }
25         for(int k = 1; k <= n; k++)
26             for(int i = 1; i <= n; i++)
27                 for(int j = 1; j <= n; j++)
28                     f[i][j] = std::min(f[i][k] + f[k][j], f[i][j]);
29
30         for(int i = 1; i <= n; i++)
31             for(int j = 0; j < 1 << (n + 1); j++) dp[i][j] = INF;

```

```

32
33     dp[1][2] = 0;
34
35     for(int j = 0; j < (1 << (n + 1)); j++) {
36         for(int i = 1; i <= n; i++)
37             if(((1 << i) & j) != 0) {
38                 for(int k = 1; k <= n; k++)
39                     dp[k][(1 << k) | j] =
40                         std::min(dp[i][j] + f[i][k]
41                             , dp[k][(1 << k) | j]);
42             }
43     }
44     int ans = INF;
45     for(int i = 1; i <= n; i++) {
46         ans = std::min(ans, dp[i][(1 << (n + 1)) - 2] + f[i][1]);
47     }
48     printf("%d\n", ans);
49 }
50 return 0;
51 }

```

## 增广路费用流

```

1  /*
2   * 增广路版费用流, 复杂度 $O(n^2m)$ 
3   * 调用之前需要清空ind并令cnt = 1
4   */
5  #include <bits/stdc++.h>
6  #define N 3000
7  #define M 100000
8  #define INF 0x7fffffff
9  typedef long long ll;
10
11 int ind[N], pre[N], f[N];
12 bool vis[N];
13 int bg[M], t[M], nt[M], c[M], op[M], v[M];
14 int cnt;
15 int S, T;
16
17 int add(int a, int b, int C, int V) { // 不要直接调用
18     bg[cnt] = a;
19     t[cnt] = b;
20     v[cnt] = V;
21     c[cnt] = C;
22     nt[cnt] = ind[a];
23     ind[a] = cnt;
24     return cnt++;
25 }
26
27 int ADD(int a, int b, int c, int v) {

```

```

28     int x = add(a, b, c, v);
29     int y = add(b, a, 0, -v);
30     op[x] = y; op[y] = x;
31     return x;
32 }
33
34 int h[N], q[N];
35
36 bool spfa() {
37     memset(vis, 0, sizeof vis);
38     for(int i = S; i <= T; i++) f[i] = INF;
39     for(int i = S; i <= T; i++) pre[i] = -1;
40     pre[S] = -1;
41     f[S] = 0;
42     int l = 0, r = 0; q[l] = S; vis[S] = true;
43     while(l <= r)
44     {
45         int x = q[l % N];
46         l++; vis[x] = false;
47         for(int k = ind[x]; k != -1; k = nt[k])
48             if(c[k] > 0 && f[t[k]] > f[x] + v[k])
49             {
50                 f[t[k]] = f[x] + v[k];
51                 pre[t[k]] = k;
52                 if(!vis[t[k]])
53                 {
54                     r++;
55                     q[r % N] = t[k];
56                     vis[t[k]] = true;
57                 }
58             }
59     }
60     return pre[T] != -1;
61 }
62
63 int dfs()
64 {
65     int ans = INF;
66     int k = pre[T];
67     while(k != -1)
68     {
69         ans = std::min(ans, c[k]);
70         k = pre[bg[k]];
71     }
72     k = pre[T];
73     while(k != -1)
74     {
75         c[k] -= ans;
76         c[op[k]] += ans;
77         k = pre[bg[k]];

```

```

78 }
79 return ans;
80 }
81
82 ll dinic() { // 程序入口
83     ll ans = 0, tmp;
84     while(spfa())
85     {
86         ans += (ll)f[T] * dfs();
87     }
88     return ans;
89 }

```

### 无向图最小割

```

1 #include <bits/stdc++.h>
2
3 const int maxn = 400;
4 const int inf = 1e9;
5 int cost[maxn][maxn], seq[maxn], len[maxn], n, m, pop, ans;
6 bool used[maxn];
7 void Init(){
8     int i, j, a, b, c;
9     for(i=0; i<n; i++) for(j=0; j<n; j++) cost[i][j]=0;
10    for(i=0; i<m; i++){
11        scanf("%d %d %d", &a, &b, &c);
12        a--; b--;
13        cost[a][b] += c; cost[b][a] += c;
14    }
15    pop = n; for(i=0; i<n; i++) seq[i] = i;
16 }
17
18 void Work(){
19     ans = inf; int i, j, k, l, mm, sum, pk;
20     while(pop > 1){
21         for(i=1; i<pop; i++) used[seq[i]] = 0; used[seq[0]] = 1;
22         for(i=1; i<pop; i++) len[seq[i]] = cost[seq[0]][seq[i]];
23         pk = 0; mm = -inf; k = -1;
24         for(i=1; i<pop; i++) if(len[seq[i]] > mm){ mm = len[seq[i]]; k = i; }
25         for(i=1; i<pop; i++){
26             used[seq[l=k]] = 1;
27             if(i == pop-2) pk = k;
28             if(i == pop-1) break;
29             mm = -inf;
30             for(j=1; j<pop; j++) if(!used[seq[j]])
31                 if((len[seq[j]] + cost[seq[l]][seq[j]]) > mm)
32                     mm = len[seq[j]], k = j;
33         }
34         sum = 0;
35         for(i=0; i<pop; i++) if(i != k) sum += cost[seq[k]][seq[i]];

```

```

36     ans = std::min(ans, sum);
37     for(i=0; i<pop; i++)
38         cost[seq[k]][seq[i]] = cost[seq[i]][seq[k]] + cost[seq[pk]][seq[i]];
39     seq[pk] = seq[--pop];
40 }
41 printf("%d\n", ans);
42 }
43
44 void solve() {
45     int K;
46     scanf("%d%d%d", &n, &m, &K);
47     if(n == 0 && m == 0 && K == 0) exit(0);
48     Init();
49     Work();
50 }
51
52 int main() {
53     while(true) solve();
54     return 0;
55 }

```

### 一般图最大匹配 带花树

```

1 #include <bits/stdc++.h>
2 /* 求一般图的最大匹配，复杂度  $O(n^3)$  .
3  *  $g[i][j]$  存放关系图： $i, j$  是否有边， $match[i]$  存放  $i$  所匹配的点
4  * 建图开始初始化  $g$  .
5  * 最终匹配方案为  $match$  .
6  * 复杂度  $O(n^3)$ 
7  * 点是从 1 到  $n$  的
8  * URAL 1099
9  */
10 const int MAXN = 500;
11 std::deque<int> Q;
12 bool g[MAXN][MAXN], inque[MAXN], inblossom[MAXN], inpath[MAXN];
13 int match[MAXN], pre[MAXN], base[MAXN];
14
15 // 找公共祖先
16 int findancestor(int u, int v)
17 {
18     memset(inpath, false, sizeof(inpath));
19     while(1)
20     {
21         u = base[u];
22         inpath[u] = true;
23         if(match[u] == -1) break;
24         u = pre[match[u]];
25     }
26     while(1)

```



```

27 {
28     v=base[v];
29     if(inpath[v])return v;
30     v=pre[match[v]];
31 }
32 }
33
34 // 压缩花
35 void reset(int u,int anc)
36 {
37     while(u!=anc)
38     {
39         int v=match[u];
40         inblossom[base[u]]=1;
41         inblossom[base[v]]=1;
42         v=pre[v];
43         if(base[v]!=anc)pre[v]=match[u];
44         u=v;
45     }
46 }
47
48 void contract(int u,int v,int n)
49 {
50     int anc=findancestor(u,v);
51     memset(inblossom,0, sizeof inblossom);
52     reset(u,anc);reset(v,anc);
53     if(base[u]!=anc)pre[u]=v;
54     if(base[v]!=anc)pre[v]=u;
55     for(int i=1;i<=n;i++)
56         if(inblossom[base[i]])
57         {
58             base[i]=anc;
59             if(!inque[i])
60             {
61                 Q.push_back(i);
62                 inque[i]=1;
63             }
64         }
65 }
66
67 bool bfs(int S,int n)
68 {
69     for(int i=0;i<=n;i++)pre[i]=-1,inque[i]=0,base[i]=i;
70     Q.clear();Q.push_back(S);inque[S]=1;
71     while(!Q.empty())
72     {
73         int u=Q.front();Q.pop_front();
74         for(int v=1;v<=n;v++)
75         {
76             if(g[u][v]&&base[v]!=base[u]&&match[u]!=v)

```

```

77 {
78     if(v==S||(match[v]!=-1&&pre[match[v]]!=-1))contract(u,v,n);
79     else if(pre[v]==-1)
80     {
81         pre[v]=u;
82         if(match[v]!=-1)Q.push_back(match[v]),inque[match[v]]=1;
83         else
84         {
85             u=v;
86             while(u!=-1)
87             {
88                 v=pre[u];
89                 int w=match[v];
90                 match[u]=v;
91                 match[v]=u;
92                 u=w;
93             }
94             return true;
95         }
96     }
97 }
98 }
99 }
100 return false;
101 }
102
103 int solve(int n)
104 {
105     memset(match,-1, sizeof match);
106     int ans=0;
107     for(int i=1;i<=n;i++)
108         if(match[i]==-1 && bfs(i,n))
109             ans++;
110     return ans;
111 }

```

### 割点/割边

```

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

```

```

13
14 class Articulation {
15 public:
16     static const int SIZE = N; // 最大结点个
17     // keyE 为割边集, keyV 为割点集
18     std::vector<PII> keyE;
19     // cc[p] 表示结点p在哪些双连通分量中
20     std::vector<int> keyV, cc[SIZE];
21     int cnt;
22     // 对于旧版编译器, 将上面 cc[SIZE] 改成 vector 的形式
23     // 传入结点个 n 及各结点的出边 e[], 返回双连通分量的个数 cnt
24     int run(int n, const std::vector<int> G[]){
25         memset(dfn, use = 0, sizeof(dfn[0]) * n);
26         memset(low, cnt = 0, sizeof(low[0]) * n);
27         keyE.clear();
28         fill_n(cc, n, keyV = std::vector<int>());
29         for(int i = 0; i < n; ++i) {
30             if(!dfn[i]) dfs(i, 1, G);
31         }
32         return cnt;
33     }
34 private:
35     int dfn[SIZE], low[SIZE], dot[SIZE], use;
36     void dfs(int x, int dep, const std::vector<int> G[]){
37         int src = 0, out = 1 < dep;
38         dot[use++] = x;
39         dfn[x] = low[x] = dep;
40         for (int i = 0; i < G[x].size(); i++) {
41             int y = G[x][i];
42             if (!dfn[y]){
43                 dfs(y, dep + 1, G);
44                 low[x] = std::min(low[x], low[y]);
45                 if (low[y] > dfn[x]) keyE.push_back(PII(x, y));
46                 if (low[y] >= dfn[x]){
47                     if (++out == 2) keyV.push_back(x);
48                     while (dot[--use] != y) cc[dot[use]].push_back(cnt);
49                     cc[x].push_back(cnt);
50                     cc[y].push_back(cnt++);
51                 }
52             } else if (dfn[y] != dfn[x] - 1 || src++){
53                 low[x] = std::min(low[x], dfn[y]);
54             }
55         }
56     }
57 } ;

```

### 斯坦纳树

```

1 /*
2 * 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 ,

```

```

3 * 初始化将 k 个点和 n 个点 dp[u][1<<i] 初始化为最短路 ,
4 * , dp[u][0]=0, 加入队列
5 * 利用 spfa 求出 dp 数组
6 * 状态转移分三部分 :
7 * 1) dp[u][su] 利用 dp[u][sub] + dp[u][su^sub] 更新, sub 为 su 子集
8 * 2) dp[u][su] 更新相邻的 dp[v][sv]
9 * 3) 将 k 中不属于 su 的点与u连接, 利用 u, k 的最短路
10 */
11
12 // hdu 4085
13 // 求斯坦纳森林, 再森林合并为全集
14 #include <bits/stdc++.h>
15 #define MOD 0x3f3f3f3f
16 typedef std::pair<int, int> PII;
17 #define MP std::make_pair
18 #define PB push_back
19 #define cmin(x, y) x = std::min(x, y)
20 #define SZ size()
21 #define AA first
22 #define BB second
23 using namespace std;
24 const int Maxn = 55;
25 const int Maxk = 14;
26 const int MSta = 1 << 15;
27 int dp[Maxn][MSta];
28 vector<PII> g[Maxn];
29 int gg[Maxn][Maxn];
30 int mp1[Maxn], mp2[Maxn];
31 int n, m, K;
32 vector<int> split[Maxk];
33 int gao[MSta];
34 int has(int sta) {
35     int i, a = 0, b = 0, u;
36     for (i = 0; i < 2 * K; i++) {
37         if ((1<<i) & sta) {
38             u = mp1[i];
39             if (u <= K) a++;
40             else b++;
41         }
42     }
43     if (a == b) return true;
44     else return false;
45 }
46 int ans;
47 queue<PII> que;
48 int inque[Maxn][MSta];
49 int doit() {
50     int i, j, k, u, v, w, su, sv, sub, vv, tp;
51     int kk = 2 * K;
52     for (i = 1; i <= n; i++) {

```

```

53     for (j = 0; j < (1<<kk); j++) dp[i][j] = MOD, inque[i][j] = 0;
54     dp[i][0] = 0; inque[i][0] = 0;
55 }
56 while(!que.empty()) que.pop();
57 memset(mp2, 0, sizeof(mp2));
58 for(i = 1, j = 0; i <= K; i++, j++) mp1[j] = i, mp2[i] = 1<<j;
59 for(i = n - K + 1; i <= n; i++, j++) mp1[j] = i, mp2[i] = 1<<j;
60 for(i = 0; i < kk; i++) {
61     for(u = 1; u <= n; u++) {
62         cmin(dp[u][1<<i], gg[u][mp1[i]]);
63         inque[u][1<<i] = 1;
64         que.push(MP(u, 1<<i));
65     }
66 }
67 while(!que.empty()) {
68     u = que.front().AA; su = que.front().BB; que.pop();
69     sub = su;
70     tp = dp[u][su];
71     do {
72         sv = sub ^ su;
73         cmin(tp, dp[u][sub] + dp[u][sv]);
74         sub = (sub - 1) & su;
75     } while(sub != su);
76     if(tp < dp[u][su]) dp[u][su] = tp;
77     for(j = 0; j < g[u].SZ; j++) {
78         v = g[u][j].AA; w = g[u][j].BB;
79         if(su & mp2[v]) continue;
80         sv = su | mp2[v];
81         if(dp[v][sv] > dp[u][su] + w) {
82             dp[v][sv] = dp[u][su] + w;
83             if(!inque[v][sv]) {
84                 inque[v][sv] = 1;
85                 que.push(MP(v, sv));
86             }
87         }
88     }
89     for(j = 0; j < kk; j++) {
90         if(su & (1<<j)) continue;
91         sv = su | (1<<j);
92         if(dp[u][sv] > dp[u][su] + gg[u][mp1[j]]) {
93             dp[u][sv] = dp[u][su] + gg[u][mp1[j]];
94             if(!inque[u][sv]) {
95                 inque[u][sv] = 1;
96                 que.push(MP(u, sv));
97             }
98         }
99     }
100     inque[u][su] = 0;
101 }
102

```

```

103     for(i = 0; i <= kk; i++) split[i].clear();
104     for(i = 0; i < (1<<kk); i++) {
105         split[__builtin_popcount(i)].PB(i);
106         gao[i] = MOD;
107         if(has(i)) {
108             for(j = 1; j <= n; j++) {
109                 cmin(gao[i], dp[j][i]);
110             }
111         }
112     }
113     for(i = 1; i <= kk; i++) {
114         for(j = 0; j < split[i].SZ; j++) {
115             su = split[i][j];
116             for(sub = su - 1 & su; sub; sub = sub - 1 & su) {
117                 sv = su ^ sub;
118                 cmin(gao[su], gao[sub] + gao[sv]);
119                 sub = (sub - 1) & su;
120             }
121         }
122     }
123     return gao[(1<<kk) - 1];
124 }
125 int main() {
126     int i,j,k,u,v,w;
127     int te;
128     scanf ("%d", &te);
129     for (int ca = 1; ca <= te; ca++) {
130         scanf ("%d%d%d", &n, &m, &K);
131         for (i = 1; i <= n; i++) g[i].clear();
132         for (i = 1; i <= n; i++) {
133             for (j = 1; j <= n; j++) gg[i][j] = MOD;
134             gg[i][i] = 0;
135         }
136         for (i = 0; i < m; i++) {
137             scanf ("%d%d%d", &u, &v, &w);
138             cmin (gg[u][v], w);
139             cmin (gg[v][u], w);
140         }
141         for (i = 1; i <= n; i++) {
142             for (j = i + 1; j <= n; j++) {
143                 if (gg[i][j] < MOD) {
144                     g[i].PB (MP (j, gg[i][j]) );
145                     g[j].PB (MP (i, gg[i][j]) );
146                 }
147             }
148         }
149         for (k = 1; k <= n; k++) {
150             for (i = 1; i <= n; i++) {
151                 for (j = 1; j <= n; j++) {
152                     cmin (gg[i][j], gg[i][k] + gg[k][j]);
153

```

```

153     }
154 }
155 }
156 ans = 0;
157 ans = doit();
158 if(ans >= MOD) printf("No solution\n");
159 else printf ("%d\n", ans);
160 }
161 return 0;
162 }

```

## 字符串 Hash

```

1  /*
2   * 全局 init()
3   * get(u64* H, int L, int w) 获取 L 开头 长度为 w 的 hash
4   */
5  const int N = 1e5+100;
6  typedef unsigned long long ull;
7  const ull mod = 1e9 + 7;
8  struct Hash {
9      ull P[N];
10     ull H[N];
11     void init() {
12         P[0] = 1;
13         for(int i = 1; i < N; i++)
14             P[i] = P[i - 1] * mod;
15     }
16
17     template <typename T>
18     void make(T *a, int n) {
19         H[n] = 0;
20         for(int i = n - 1; i >= 0; i--)
21             H[i] = H[i + 1] * mod + a[i] + 1;
22     }
23
24     ull get(int L, int w) {
25         return H[L] - H[L + w] * P[w];
26     }
27 } T;

```

## KMP

```

1  /*
2   * 初始把 b[n + 1] 赋为 -1
3   * Hdu 1711
4   */
5  template<typename T>

```

```

6  void build(T b[], int next[], int m) {
7      int k = -1;
8      for(int i = 0; i < m; i++) {
9          while(k != -1 && b[k + 1] != b[i]) k = next[k];
10         if(k + 1 != i && b[k + 1] == b[i]) k++;
11         next[i] = k;
12     }
13 }
14 // 得到 b 在 a 中的第一个匹配位置
15 template <typename T>
16 int kmp(T a[], T b[], int next[], int n, int m) {
17     int k = -1;
18     for(int i = 0; i < n; i++) {
19         while(k != -1 && b[k + 1] != a[i]) k = next[k];
20         if(b[k + 1] == a[i]) k++;
21         if(k == m - 1) return i - m + 2;
22     }
23     return -1;
24 }

```

## EXKMP

```

1  /* Hdu 4300
2   * S 为主串 , T 为子串 ,
3   * LS 为 S 长度 , LT 为 T 长度 ,
4   * B[i] 表示 S[i] 匹配了 B[i] 长度的 T
5   */
6  #include <bits/stdc++.h>
7  const int Maxn = 1e5;
8  char S[Maxn], T[Maxn];
9  int Next[Maxn], B[Maxn];
10
11 void preExKmp(char T[], int LT, int next[]) {
12     int i, ind = 0, k = 1;
13     next[0] = LT;
14     while(ind + 1 < LT && T[ind + 1] == T[ind]) ind++;
15     next[1] = ind;
16     for(i = 2; i < LT; i++) {
17         if(i <= k + next[k] - 1 && next[i - k] + i < k + next[k])
18             next[i] = next[i - k];
19         else {
20             ind = std::max(0, k + next[k] - i);
21             while(ind + i < LT && T[ind + i] == T[ind]) ind++;
22             next[i] = ind; k = i;
23         }
24     }
25 }
26
27 void exKmp(char S[], int LS, char T[], int LT, int next[], int B[]) {

```

```

28  int i, ind = 0, k = 0;
29  preExKmp(T, LT, next);
30  while(ind < LS && ind < LT && T[ind] == S[ind]) ind++;
31  B[0] = ind;
32  for(i = 1; i < LS; i++) {
33      int p = k + B[k] - 1, L = next[i - k];
34      if((i - 1) + L < p)
35          B[i] = L;
36      else {
37          ind = std::max(0, p - i + 1);
38          while(ind + i < LS && ind < LT && S[ind + i] == T[ind]) ind++;
39          B[i] = ind;
40          k = i;
41      }
42  }
43  }

```

## SA

```

1  /*
2  * 倍增算法 , r 为待匹配数组,
3  * n 为总长度 +1 , m 为字符范围 , num 保存字符串
4  * 使用时注意 num[] 有效位为 [0, n) ,
5  * 但是需要将 num[n] = 0 ,
6  * 另外 , 对于模板的处理将空串也处理了 ,
7  * 作为rank最小的串 ,
8  * 因此有效串为 [0, n] 共 n-1 个 ,
9  * 在调用da()函数时, 需要调用 da(num, n + 1, m)
10 * 对于 sa[], rank[], height[] 数组都将空串考虑在内, 作为 rank 最小的后缀
11 * 注意 rank , height 范围从 [0, n]
12 */
13
14 const int N = 1e5;
15 namespace SA
16 {
17     int len;
18     int num[N];
19     // sa[1~n] value(0~n-1) ; rank[0..n-1] value(1..n) ; height[2..n]
20     int sa[N], rank[N], height[N];
21     int wa[N], wb[N], wv[N], wd[N];
22
23     int cmp(int *r, int a, int b, int x) {
24         return r[a] == r[b] && r[a + x] == r[b + x];
25     }
26
27     // 倍增算法 r 为待匹配数组 n 为总长度 +1 , m 为字符范围
28     void da(int *r, int n, int m) {
29         int i, j, k, p, *x = wa, *y = wb, *t;
30         for(i = 0; i < m; i++) wd[i] = 0;
31         for(i = 0; i < n; i++) wd[x[i] = r[i]]++;

```

```

32         for(i = 1; i < m; i++) wd[i] += wd[i - 1];
33         for(i = n - 1; i >= 0; i--) sa[--wd[x[i]]] = i;
34         for(j = 1, p = 1; p < n; j <= 1, m = p) {
35             for(p = 0, i = n - j; i < n; i++) y[p++] = i;
36             for(i = 0; i < n; i++) if(sa[i] >= j) y[p++] = sa[i] - j;
37             for(i = 0; i < n; i++) wv[i] = x[y[i]];
38             for(i = 0; i < m; i++) wd[i] = 0;
39             for(i = 0; i < n; i++) wd[wv[i]]++;
40             for(i = 1; i < m; i++) wd[i] += wd[i - 1];
41             for(i = n - 1; i >= 0; i--) sa[--wd[wv[i]]] = y[i];
42             for(t = x, x = y, y = t, p = 1, x[sa[0]] = 0, i = 1; i < n; i++) {
43                 x[sa[i]] = cmp(y, sa[i - 1], sa[i], j) ? p - 1 : p++;
44             }
45         }
46
47         for(i = 0, k = 0; i < n; i++) rank[sa[i]] = i;
48         for(i = 0; i < n - 1; height[rank[i+1]] = k) {
49             for(k ? k-- : 0, j = sa[rank[i] - 1]; r[i + k] == r[j + k]; k++);
50         }
51     }
52 }

```

## DC3

```

1  // 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
2  // 约定除 r[n-1] 外所有的 r[i] 都大于 0, r[n-1]=0
3  // 函数结束后, 结果放在 sa 数组中, 从 sa[0] 到 sa[n-1]
4  #define maxn 10000
5  #define F(x) ((x)/3+((x)%3==1?0:tb))
6  #define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
7  int wa[maxn],wb[maxn],wv[maxn],wss[maxn]; // 必须这么大
8  int s[maxn*3],sa[maxn*3];
9  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];}
10 int c12(int k,int *r,int a,int b){
11     if(k==2) return r[a]<r[b]||r[a]==r[b]&&c12(1,r,a+1,b+1);
12     else return r[a]<r[b]||r[a]==r[b]&&wv[a+1]<wv[b+1];
13 }
14 void sort(int *r,int *a,int *b,int n,int m){
15     int i; for(i=0;i<n;i++) wv[i]=r[a[i]];
16     for(i=0;i<m;i++) wss[i]=0; for(i=0;i<n;i++) wss[wv[i]]++;
17     for(i=1;i<m;i++) wss[i]+=wss[i-1];
18     for(i=n-1;i>=0;i--) b[--wss[wv[i]]]=a[i];
19 }
20 void dc3(int *r,int *sa,int n,int m){
21     int i,j,*rn=r+n,*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;
22     r[n]=r[n+1]=0;
23     for(i=0;i<n;i++) if(i%3!=0) wa[tbc++]=i;
24     sort(r+2,wa,wb,tbc,m); sort(r+1,wb,wa,tbc,m); sort(r,wa,wb,tbc,m);
25     for(p=1,rn[F(wb[0])]=0,i=1;i<tbc;i++)

```

```

26     rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;
27     if(p<tb) dc3(rn,san,tbc,p);
28     else for(i=0;i<tbc;i++) san[rn[i]]=i;
29     for (i=0;i<tbc;i++) if(san[i]<tb) wb[ta++]=san[i]*3;
30     if(n%3==1) wb[ta++]=n-1;
31     sort(r,wb,wa,ta,m); for(i=0;i<tbc;i++) wv[wb[i]=G(san[i])]=i;
32     for(i=0,j=0,p=0;i<ta && j<tbc;p++)
33         sa[p]=c12(wb[j]%3,r,wa[i],wb[j])?wa[i++]:wb[j++];
34     for(;i<ta;p++) sa[p]=wa[i++]; for(;j<tbc;p++) sa[p]=wb[j++];
35     int main(){
36         int n,m=0; scanf("%d",&n);
37         for (int i=0;i<n;i++) scanf("%d",&s[i]),s[i]++,m=max(s[i]+1,m);
38         printf("%d\n",m); s[n++]=0; dc3(s,sa,n,m);
39         for (int i=0;i<n;i++) printf("%d ",sa[i]);printf("\n");
40     }

```

## Manacher 最长回文串

```

1  /*
2  * 求以 i 为中心的最长回文串长度 ,
3  * 结果保存在 pk[i] 中 , 下标 [0,n) ,
4  * 开头 , 末尾 , 字符之间插入 \# , 例如 ababa 变为 \#a\#b\#a\#b\#a\#
5  */
6  #include <bits/stdc++.h>
7  const int N = 1e5;
8  int pk[N];
9  template <typename T>
10 // [0, n)
11 void manacher(T *a, int n) {
12     int mx = 0;
13     int p;
14     for(int i = 0; i < n; i++) {
15         if(i < mx) pk[i] = std::min(pk[2 * p - i], mx - i);
16         else pk[i] = 1;
17         while(i + pk[i] < n && i - pk[i] > -1 && a[i + pk[i]] == a[i - pk[i]]) pk[i]++;
18         if(i + pk[i] > mx) { p = i; mx = i + pk[i]; }
19     }
20 }

```

## 最小表示法

```

1  /*
2  * hdu 3374
3  */
4  #include <string>
5  #include <algorithm>
6
7  std::string find(std::string s) {
8      int i,j,k,l,N=s.length(); s+=s;

```

```

9      for(i=0,j=1;j<N;){
10         for(k=0;k<N&&s[i+k]==s[j+k];k++);
11         if(k>=N) break;
12         if(s[i+k]<s[j+k]) j+=k+1;
13         else l=i+k,i=j,j=std::max(l,j)+1;
14     }
15     return s.substr(i,N);
16 }

```

## SAM

```

1  /*
2  * cxlove
3  * spoj NSUBSTR
4  * 给一个字符串 S,
5  * 令 F(x) 表示 S 的所有长度为x的子串中 ,
6  * 出现次数的最大值 ,
7  * 求 F(1)..F(Length(S))
8  * 结点的 len 值表示那一时刻的后缀长度
9  */
10 #include <bits/stdc++.h>
11 #define inf 100000005
12 #define M 40
13 #define N 510005
14 #define maxn 300005
15 #define eps 1e-10
16 #define zero(a) fabs(a)<eps
17 #define pb(a) push_back(a)
18 #define mp(a,b) make_pair(a,b)
19 #define mem(a,b) memset(a,b,sizeof(a))
20 #define LL unsigned long long
21 #define MOD 1000000007
22 #define lson step<<1
23 #define rson step<<1|1
24 #define sqr(a) ((a)*(a))
25 #define Key_value ch[ch[root][1]][0]
26 #define test puts("OK");
27 #pragma comment(linker, "/STACK:1024000000,1024000000")
28 struct SAM {
29     SAM *pre,*son[26];
30     int len,g;
31 }que[N],*root,*tail,*b[N];
32 int tot;
33 void add(int c,int l) {
34     SAM *p=tail,*np=&que[tot++];
35     np->len=l;tail=np;
36     while(p&&p->son[c]==NULL) p->son[c]=np,p=p->pre;
37     if(p==NULL) np->pre=root;
38     else {

```

```

39     SAM *q=p->son[c];
40     if(p->len+1==q->len) np->pre=q;
41     else {
42         SAM *nq=&que[tot++];
43         *nq=*q;
44         nq->len=p->len+1;
45         np->pre=q->pre=nq;
46         while(p&& p->son[c]==q) p->son[c]=nq,p=p->pre;
47     }
48 }
49 }
50 char str[N/2];
51 int dp[N/2];
52 int main() {
53     while(scanf("%s",str)!=EOF) {
54         int n=strlen(str);
55         tot=0;
56         root=tail=&que[tot++];
57         for(int i=0;i<n;i++) add(str[i]-'a',i+1);
58         int cnt[N/2];mem(cnt,0);
59         for(int i=0;i<tot;i++) cnt[que[i].len]++;
60         for(int i=1;i<=n;i++) cnt[i]+=cnt[i-1];
61         for(int i=0;i<tot;i++) b[--cnt[que[i].len]]=&que[i];
62         for(int i=0;i<n;i++) (root=root->son[str[i]-'a'])->g++;
63         mem(dp,0);
64         for(int i=tot-1;i>0;i--) {
65             dp[b[i]->len]=std::max(dp[b[i]->len],b[i]->g);
66             if(b[i]->pre) b[i]->pre->g+=b[i]->g;
67         }
68         for(int i=n-1;i>0;i--) dp[i]=std::max(dp[i],dp[i+1]);
69         for(int i=1;i<=n;i++) printf("%d\n",dp[i]);
70     }
71     return 0;
72 }

```

## Trie

```

1  #include <bits/stdc++.h>
2
3  const int M = 26;
4  const int N = 2000;
5  struct trie {
6      int to[N][M]; // 节点指针
7      int w[256]; // 字母下标
8      int fail[N]; // 失配指针
9      int cnt;
10
11     trie() {
12         // 标注下标
13     }

```

```

14
15     int newNode() {
16         memset(to[cnt], 0, sizeof to[cnt]);
17         return cnt++;
18     }
19
20     void init() {
21         cnt = 0;
22         memset(to[0], 0, sizeof to[0]);
23     }
24
25     void insert(char *a, int mask) {
26         int p = 0;
27         while(*a) {
28             int v = w[*a];
29             if(!to[p][v]) {
30                 to[p][v] = newNode();
31             }
32             p = to[p][v];
33             a++;
34         }
35     }
36
37     void ac() {
38         std::queue<int>q;
39         memset(fail, 0, sizeof fail);
40         for(int i = 0; i < 4; i++) if(to[0][i]) {
41             q.push(to[0][i]);
42         }
43         while(q.size()) {
44             int p = q.front();
45             q.pop();
46             for(int i = 0; i < 4; i++) {
47                 int v = to[p][i];
48                 if(v) {
49                     fail[v] = to[fail[p]][i];
50                     q.push(v);
51                 }
52                 else
53                     to[p][i] = to[fail[p]][i];
54             }
55         }
56     }
57 };

```

## 数学 Fib

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  const int MOD = 1e9+7;
5  typedef long long LL;
6  LL mul(LL a, LL b) {
7      LL tot = 0;
8      while(b) {
9          if(b & 1) tot = (tot + a);
10         b >>= 1;
11         a <<= 1;
12         if(tot >= MOD) tot -= MOD;
13         if(a >= MOD) a -= MOD;
14     }
15     return tot;
16 };
17
18 void fib(LL n, LL &x, LL &y) {
19     if (n == 1) {
20         x = y = 1;
21         return;
22     } else if (n & 1) {
23         fib(n - 1, y, x);
24         y += x; y %= MOD;
25     } else {
26         LL a, b;
27         fib(n >> 1, a, b);
28         y = (mul(a, a) % MOD + mul(b, b) % MOD) % MOD;
29         x = (mul(a, b) % MOD + mul(a, (b + MOD - a)) % MOD) % MOD;
30     }
31 }

```

## 矩阵

```

1  #include <bits/stdc++.h>
2
3  const int MXN = 4;
4  const int mod = 1e4+7;
5  template <typename T> class Matrix{
6  public:
7      int n,m ;
8      T a[MXN][MXN];
9      Matrix() {memset(a , 0 , sizeof(a));}
10     Matrix(int _n , int _m) {
11         n = _n , m = _m , memset(a , 0 , sizeof(a));
12     };
13     T* operator[] (int i) {return a[i];}

```

```

14     Matrix friend operator * (Matrix A , Matrix B){
15         Matrix ans(A.n , B.m);
16         for (int i = 0 ; i < A.n ; ++ i)
17             for (int j = 0 ; j < A.m ; ++ j)if(A[i][j]!=0)
18                 for (int k = 0 ; k < B.m ; ++ k)
19                     ans[i][k]=(ans[i][k]+A[i][j]*B[j][k]) % mod;
20         return ans;
21     }
22 };

```

## 高斯消元

```

1  // 二进制版本 , hdu5833
2  #include <bits/stdc++.h>
3  const int N = 1000;
4
5  double a[N][N];
6  int gauss(int eq, int var) {
7      int line = 0, ret = 0;
8      for(int i = 0; i < var; i++) {
9          int target = line;
10         while(target < eq && a[target][i] == 0) target++;
11         if(target == eq) { ret++; continue; }
12         std::swap(a[target], a[line]);
13         for(int j = 0; j < eq; j++) if(j != line && a[j][i]) a[j] ^= a[line];
14         line++;
15     }
16     return ret;
17 }
18
19 // 数值版本
20 int Gauss(int equ, int var) {
21     int i, j, k;
22     int max_r; // 当前这列绝对值最大的行 .
23     int col; // 当前处理的列
24     // 转换为阶梯阵 .
25     col = 0; // 当前处理的列
26     for(k = 0; k < equ && col < var; k++,col++) {
27         // 枚举当前处理的行 .
28         // 找到该 col 列元素绝对值最大的那行与第 k 行交换 .
29         // ( 为了在除法时减小误差 )
30         max_r=k;
31         for(i = k + 1; i < equ; i++) {
32             if((a[i][col]) > (a[max_r][col])) max_r = i;
33         }
34         if(max_r != k) {
35             // 与第k行交换 .
36             for(j=k; j<var+1; j++) swap(a[k][j], a[max_r][j]);
37         }

```



```

38     if((a[k][col])==0) {
39         // 说明该 col 列第 k 行以下全是 0 了 , 则处理当前行的下一列 .
40         k--;
41         continue;
42     }
43     for(i=k+1; i<equ; i++) {
44         // 枚举要删去的行 .
45         if((a[i][col]) > 0) {
46             for(j = var; j >= col; j--) {
47                 a[i][j] -= a[k][j] * (a[i][col] / a[k][col]);
48             }
49         }
50     }
51 }
52 // 无穷解 , 返回自由变元个数
53 // 会改变解的顺序
54 if(k < var) {
55     for(int i=0; i<equ; i++) {
56         if(a[i][i]!= 0) continue;
57         else {
58             int flag=0;
59             for(int j=i+1; j<var; j++) {
60                 if(a[i][j] != 0) {
61                     flag = 1;
62                     for(int k = 0; k <= i; k++) {
63                         swap(a[k][i], a[k][j]);
64                     }
65                     break;
66                 }
67             }
68             if(!flag)
69                 return var - i;
70         }
71     }
72     return var - equ;
73 }
74 return 0;
75 }

```

## 四边形不等式

```

1  /*
2  * Kyb 示意代码 待验证
3  */
4  #define rep(i,a,n) for(int i=(a);i<(int)(n);i++)
5  #define better true
6
7  void dp(int n, int K[][100]) {
8      rep(r,1,n){
9          rep(i,1,n-r){

```

```

10         if(r==1)K[i][i+r]=i,dp;
11         else rep(k,K[i][i+r-1],K[i+1][i+r])
12             if(better)K[i][i+r]=k,dp;
13     }
14 }
15 }

```

## Java 开根号

```

1  /*
2  * Java 开根号
3  * Kyb , 待验证
4  */
5  public static BigInteger getsqrt(BigInteger n) {
6      if (n.compareTo(BigInteger.ZERO) <= 0) return n;
7      BigInteger x, xx, txx;      xx = x = BigInteger.ZERO;
8      for (int t = n.bitLength() / 2; t >= 0; t--) {
9          txx = xx.add(x.shiftLeft(t + 1)).add(BigInteger.ONE.shiftLeft(t + t));
10         if (txx.compareTo(n) <= 0) {
11             x = x.add(BigInteger.ONE.shiftLeft(t));      xx = txx;
12         }
13     }
14     return x;
15 }

```

## C++ 大数

```

1  /*
2  * C++ 大数运算 , Kyb , 待验证
3  */
4  #include <bits/stdc++.h>
5  class BigNum {
6  public:
7      static const int MOD = 100000000;
8      static const int BIT = 8, SIZE = 105;
9      mutable int n, o;
10     long long u[SIZE];
11     BigNum() {}
12     BigNum(const std::string& s) {
13         memset(this, 0, sizeof(BigNum));
14         int num = 0, cnt = 1;
15         for (int i = s.size() - 1; ~i; i--) {
16             if (s[i] == '-') o ^= 1;
17             if (s[i] >= '0' && s[i] <= '9') {
18                 num += (s[i] - '0') * cnt;
19                 cnt *= 10;

```

```

20     if (cnt == MOD) u[n++] = num, num = 0, cnt = 1;
21 }
22 }
23 if (!n || cnt >= 10) u[n++] = num;
24 if (!u[0] && n == 1) o = 0;
25 }
26 BigNum(long long x) {
27     memset(this, 0, sizeof(BigNum));
28     if (x < 0) o = 1, x = -x;
29     do u[n++] = x % MOD; while (x /= MOD);
30 }
31 operator std::string() const {
32     static char s[SIZE * BIT + 10];
33     char* c = s + sprintf(s, "%s%d", o ? "-" : "", int(u[n - 1]));
34     for (int i = n - 2; ~i; i--) c += sprintf(c, "%0*d", BIT, int(u[i]));
35     return s;
36 }
37 int operator [](int pos) const {
38     static int e[BIT] = {1};
39     for (static int i = 1; i < BIT; i++) e[i] = e[i - 1] * 10;
40     return u[pos / BIT] / e[pos % BIT] % 10;
41 }
42 int length() const {
43     int ret = (n - 1) * BIT + 1;
44     for (int x = u[n - 1] / 10; x; x /= 10) ret++;
45     return ret;
46 }
47 friend int cmp(const BigNum& l, const BigNum& r) {
48     if (l.o != r.o) return (l.o ? -1 : 1);
49     if (l.n != r.n) return (l.o ? -1 : 1) * (l.n - r.n);
50     for (int i = l.n - 1; ~i; i--) if (l.u[i] - r.u[i])
51         return (l.o ? -1 : 1) * (l.u[i] - r.u[i]);
52     return 0;
53 }
54 // 运算符
55 bool operator < (const BigNum& r) const {return cmp(*this, r) < 0;}
56 bool operator > (const BigNum& r) const {return cmp(*this, r) > 0;}
57 bool operator <= (const BigNum& r) const {return cmp(*this, r) <= 0;}
58 bool operator >= (const BigNum& r) const {return cmp(*this, r) >= 0;}
59 bool operator == (const BigNum& r) const {return cmp(*this, r) == 0;}
60 bool operator != (const BigNum& r) const {return cmp(*this, r) != 0;}
61 BigNum operator + (const BigNum& r) const {return BigNum(*this) += r;}
62 BigNum operator - (const BigNum& r) const {return BigNum(*this) -= r;}
63 BigNum operator * (int x) const {return BigNum(*this) *= x;}
64 BigNum operator / (int x) const {return BigNum(*this) /= x;}
65 BigNum& operator *= (const BigNum& r) {return *this = *this * r;}
66 BigNum& operator /= (const BigNum& r) {return *this = *this / r;}
67 BigNum& operator %= (const BigNum& r) {return *this = *this % r;}
68 BigNum& operator %=(int x) {return *this = *this % x;}
69 BigNum operator -() const {

```

```

70     BigNum s = *this;
71     if (s.u[0] || s.n >= 2) s.o ^= 1;
72     return s;
73 }
74 BigNum& operator += (const BigNum& r) {
75     if (r.n == 1 && !r.u[0]) return *this;
76     if (r.o ^ o) return r.o ^= 1, *this -= r, r.o ^= 1, *this;
77     if (r.n > n) n = r.n;
78     for (int i = 0; i < r.n; i++) u[i] += r.u[i];
79     for (int i = 0; i < n; i++) if (u[i] >= MOD) u[i + 1]++, u[i] -= MOD;
80     if (u[n]) n++;
81     return *this;
82 }
83 BigNum& operator -= (const BigNum& r) {
84     if (r.n == 1 && !r.u[0]) return *this;
85     if (r.o ^ o) return r.o ^= 1, *this += r, r.o ^= 1, *this;
86     if (cmp(*this, r) * (r.o ? -1 : 1) < 0) {
87         o ^= 1, n = r.n;
88         for (int i = 0; i < r.n; i++) u[i] = r.u[i] - u[i];
89     } else {
90         for (int i = 0; i < r.n; i++) u[i] = u[i] - r.u[i];
91     }
92     for (int i = 0; i < n; i++) if (u[i] < 0) u[i + 1]--, u[i] += MOD;
93     while (!u[n - 1] && n >= 2) --n;
94     if (!u[0] && n == 1) o = 0;
95     return *this;
96 }
97 BigNum operator * (const BigNum& r) const {
98     BigNum s = 0;
99     if (!u[n - 1] || !r.u[r.n - 1]) return s;
100     s.n = r.n + n - 1;
101     s.o = r.o ^ o;
102     for (int i = 0; i < n; i++) for (int j = 0; j < r.n; j++)
103         s.u[i + j] += u[i] * r.u[j];
104     for (int i = 0; i < s.n; i++) if (s.u[i] >= MOD) {
105         s.u[i + 1] += s.u[i] / MOD;
106         s.u[i] %= MOD;
107         if (i == s.n - 1) s.n++;
108     }
109     return s;
110 }
111 BigNum operator / (const BigNum& r) const {
112     BigNum e[35], s = 0, c = 0;
113     int m = 0, ro = r.o, lo = o;
114     r.o ^= ro, o ^= lo;
115     for (e[m] = r; MOD >> ++m; e[m] = e[m - 1] + e[m - 1]);
116     for (int i = n - 1; ~i; i--) {
117         int tag = 0;
118         (s == MOD) += u[i];
119         for (int x = m - 1; ~x; x--) if (s >= e[x]) s -= e[x], tag |= 1 << x;

```

```

120     (c *= MOD) += tag;
121 }
122 r.o ^= ro, o ^= lo;
123 if (c.u[0] || c.n >= 2) c.o = r.o ^ o;
124 return c;
125 }
126 BigNum operator %(const BigNum& r) const {
127     BigNum e[35], s = 0;
128     int m = 0, ro = r.o, lo = o;
129     r.o ^= ro, o ^= lo;
130     for (e[m] = r; MOD >> ++m; e[m] = e[m - 1] + e[m - 1]);
131     for (int i = n - 1; ~i; i--) {
132         (s *= MOD) += u[i];
133         for (int x = m - 1; ~x; x--) if (s >= e[x]) s -= e[x];
134     }
135     r.o ^= ro, o ^= lo;
136     if (s.u[0] || s.n >= 2) s.o = o;
137     return s;
138 }
139 BigNum& operator *=(int x) {
140     if (!x) return *this = 0;
141     if (x < 0) o ^= 1, x = -x;
142     for (int i = 0; i < n; i++) u[i] *= x;
143     for (int i = 0; i < n; i++) if (u[i] >= MOD) {
144         u[i + 1] += u[i] / MOD;
145         u[i] %= MOD;
146         if (i == n - 1) n++;
147     }
148     if (!u[0] && n == 1) o = 0;
149     return *this;
150 }
151 BigNum& operator /=(int x) {
152     if (x < 0) o ^= 1, x = -x;
153     for (int i = n - 1; i; u[i--] /= x) u[i - 1] += u[i] % x * MOD;
154     for (u[0] /= x; n >= 2; n--) if (u[n - 1]) break;
155     if (!u[0] && n == 1) o = 0;
156     return *this;
157 }
158 int operator %(int x) const {
159     long long c = 0;
160     for (int i = n - 1; ~i; i--) c = (c * MOD + u[i]) % x;
161     return (1 - o - o) * int(c);
162 }
163 };

```

## 线性逆元

```

1 // 返回 [0, n] 每个数关于 m 的逆元
2 // x * i = 1 mod m
3 // let m = i * k + r, where k = m / i, r = m % i

```

```

4 // i = (m - r) * k^-1
5 // x = -k * r^-1
6 // time complexity : O(n)
7
8 std::vector<LL> linear_inversion(int n, LL m) {
9     std::vector<LL> inv(n + 1, 0);
10    inv[1] = 1;
11    for (int i = 2; i <= n; ++i) {
12        inv[i] = (m - m / i) * inv[m % i] % m;
13    }
14    return inv;
15 }

```

## 勒让德定理

```

1 /*
2  * 求出 n! 分解中 , 素因子 x 的个数
3  */
4
5 typedef long long ll;
6 ll lrd(ll n, ll x) {
7     ll ans = 0;
8     for (ll k = x; k <= n; k *= x) {
9         ans += n / k;
10    }
11    return ans;
12 }

```

## 欧拉函数

```

1 /*
2  * 对正整数 n ,
3  * 欧拉函数是少于或等于n的数中与n互质的数的数目
4  * P是素数 :
5  * 若p是x的约数 , 则  $E(xp) = E(x)p$ 
6  * p不是x的约数 , 则  $E(xp) = E(x)E(p) = E(x)(p - 1)$ 
7  */
8 const int N = 1e5;
9 int prime[N];
10 int phi[N];
11 bool is_prime[N];
12
13 void get_phi() {
14     int i, j, k;
15     k = 0;
16     for (i = 2; i < N; i++) {
17         if (is_prime[i] == false) {
18             prime[k++] = i;

```

```

19     phi[i] = i - 1;
20 }
21 for(j = 0; j < k && i * prime[j] < N; j++) {
22     is_prime[ i * prime[j] ] = true;
23     if( i % prime[j] == 0) {
24         phi[ i * prime[j] ] = phi[i] * prime[j];
25         break;
26     }
27     else {
28         phi[ i * prime[j] ] = phi[i] * ( prime[j] - 1 );
29     }
30 }
31 }
32 }

```

## 行列式求值

```

1  /*
2  * 行列式 Mod
3  * Kyb 待验证
4  */
5  #include <algorithm>
6  const int MOD = (int)1e9 + 7;
7
8  inline void add(int &a,int b) {
9      a += b;
10     if (a >= MOD) a -= MOD;
11 }
12 int det(int A[16][16],int n) {
13     int ret = 1;
14     for (int i = 0; i < n; ++ i) {
15         if (A[i][i] == 0) {
16             for (int j = i + 1; j < n; ++ j) {
17                 if (A[j][i]) {
18                     for (int k = i; k < n; ++ k) {
19                         std::swap(A[i][k],A[j][k]);
20                     }
21                     ret = -ret;
22                     break;
23                 }
24             }
25             if (A[i][i] == 0) return 0;
26         }
27         for (int j = i + 1; j < n; ++ j) {
28             int a = 1,b = 0,c = 0,d = 1;
29             int x = A[i][i],y = A[j][i];
30             while (y) {
31                 int t = x / y;
32                 if (t < 0) t += MOD;
33                 add(a,MOD - c * 111 * t % MOD);

```

```

34         add(b,MOD - d * 111 * t % MOD);
35         std::swap(a,c);
36         std::swap(b,d);
37         x %= y;
38         std::swap(x,y);
39         ret = -ret;
40     }
41     for (int k = 0; k < n; ++ k) {
42         int q = A[i][k],w = A[j][k];
43         A[i][k] = (a * 111 * q + b * 111 * w) % MOD;
44         A[j][k] = (c * 111 * q + d * 111 * w) % MOD;
45     }
46 }
47 ret = A[i][i] * 111 * ret % MOD;
48 }
49 if (ret < 0) {
50     ret += MOD;
51 }
52 return ret;
53 }

```

## 生成树计数

```

1  /*
2  * 算法引入 :
3  * 给定一个无向图 G , 求它生成树的个数 t(G) ;
4  *
5  * 算法思想 :
6  * 1) G 的度数矩阵 D[G] 是一个 n*n 的矩阵 ,
7  * 并且满足 : 当 i != j 时 , dij=0 ; 当 i=j 时 , dij 等于 vi 的度数 ;
8  * 2) G 的邻接矩阵 A[G] 是一个 n*n 的矩阵 ,
9  * 并且满足 : 如果 vi , vj 之间有边直接相连 , 则 aij=1 , 否则为 0 ;
10 * 定义图 G 的 Kirchhoff 矩阵 C[G] 为 C[G]=D[G]-A[G] ;
11 * Matrix-Tree 定理 : G 的所有不同的生成树的个数等于
12 * 其 Kirchhoff 矩阵 C[G] 任何一个 n-1 阶主子式的行列式的绝对值 ;
13 * 所谓 n-1 阶主子式 , 就是对于 r(1 <= r <= n) , 将 C[G] 的第r行
14 * 第 r 列同时去掉后得到的新矩阵 , 用 Cr[G] 表示 ;
15 *
16 * Kirchhoff 矩阵的特殊性质 :
17 * 1) 对于任何一个图 G, 它的 Kirchhoff 矩阵 C 的行列式总是 0 ,
18 * 这是因为 C 每行每列所有元素的和均为 0 ;
19 * 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 ;
20 * 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 ;
21 *
22 * 算法举例 :
23 * SPOJ HIGH
24 *
25 * 题目地址 :
26 * http://www.spoj.com/problems/HIGH/

```

```

27  *
28  * 题目大意 :
29  * 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 ;
30  * 需要有选择的修建一些高速公路 , 从而组成一个交通网络 ;
31  * 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 ;
32  **/
33
34  #include <bits/stdc++.h>
35  const int N=15;
36  typedef long long ll;
37
38  int degree[N];
39  ll C[N][N];
40  // 生成树计数 : Matrix-Tree 定理
41  ll det(ll a[][N],int n) {
42      ll ret = 1;
43      for(int i=1; i<n; i++) {
44          for(int j=i+1; j<n; j++)
45              while(a[j][i]) {
46                  ll t = a[i][i] / a[j][i];
47                  for(int k = i; k < n; k++)
48                      a[i][k] = (a[i][k] - a[j][k] * t);
49                  for(int k = i; k < n; k++)
50                      std::swap(a[i][k], a[j][k]);
51                  ret = -ret;
52              }
53          if(a[i][i] == 0)
54              return 0;
55          ret = ret * a[i][i];
56      }
57      if(ret < 0)
58          ret = -ret;
59      return ret;
60  }
61
62  int main() {
63      int tcase;
64      scanf("%d",&tcase);
65      while(tcase--) {
66          memset(degree, 0, sizeof(degree));
67          memset(C, 0, sizeof(C));
68          int n ,m;
69          scanf("%d%d", &n, &m);
70          int u, v;
71          while(m--) {
72              scanf("%d%d", &u, &v);
73              u--; v--;
74              C[u][v] = C[v][u] = -1;
75              degree[u]++; degree[v]++;
76          }

```

```

77      for(int i = 0; i < n; ++i)
78          C[i][i] = degree[i];
79      printf("%lld\n", det(C, n));
80  }
81  return 0;
82  }

```

## Simpson 积分

```

1  /*
2  * 自适应 Simpson
3  * 验题 :hdu4978
4  * Kyb 待验证
5  */
6  #include <cmath>
7  const double eps = 1e-6;
8  double f(double x) { return x; }
9
10 long double simpson(long double a, long double b) {
11     long double c = a + (b - a) / 2;
12     return (f(a) + 4 * f(c) + f(b)) * (b - a) / 6;
13 }
14 long double asr(long double a, long double b, long double eps, long double A) {
15     long double c = a + (b - a) / 2;
16     long double L = simpson(a, c), R = simpson(c, b);
17     if (std::fabs(L + R - A) < 15 * eps) return L + R + (L + R - A) / 15.;
18     return asr(a, c, eps / 2, L) + asr(c, b, eps / 2, R);
19 }
20 long double asr(long double a, long double b, long double eps) {
21     return asr(a, b, eps, simpson(a, b));
22 }

```

## 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

$$X=a_nn!+a_{n-1}(n-1)!+...+a_2\cdot 2!+a_1\cdot 1!$$

a<sub>i</sub> 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 ∈ B 的 w 中最后的一个点 , 判断 v ∪ N(v) 是否为极大团 , 只需判断是否存在一个 w ∈ w\*, 满足 Next(w) = v 且 |N(v)| + 1 ≤ |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

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

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

$$\sin(a\pm b)=\sin a\cos b\pm\cos a\sin b\quad\cos(a\pm b)=\cos a\cos b\mp\sin a\sin b$$
$$\tan(a\pm b)=\frac{\tan(a)\pm\tan(b)}{1\mp\tan(a)\tan(b)}\quad\tan(a)\pm\tan(b)=\frac{\sin(a\pm b)}{\cos(a)\cos(b)}$$
$$\sin(a)+\sin(b)=2\sin(\frac{a+b}{2})\cos(\frac{a-b}{2})\quad\sin(a)-\sin(b)=2\cos(\frac{a+b}{2})\sin(\frac{a-b}{2})$$
$$\cos(a)+\cos(b)=2\cos(\frac{a+b}{2})\cos(\frac{a-b}{2})\quad\cos(a)-\cos(b)=-2\sin(\frac{a+b}{2})\sin(\frac{a-b}{2})$$
$$\sin(na)=n\cos^{n-1}a\sin a-\binom{n}{3}\cos^{n-3}a\sin^3a+\binom{n}{5}\cos^{n-5}a\sin^5a-\ldots$$
$$\cos(na)=\cos^na-\binom{n}{2}\cos^{n-2}a\sin^2a+\binom{n}{4}\cos^{n-4}a\sin^4a-\ldots$$

积分表

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

$$\int\sqrt{x^2\pm a^2}dx=\frac{1}{2}x\sqrt{x^2\pm a^2}\pm\frac{1}{2}a^2\ln\left|x+\sqrt{x^2\pm a^2}\right|$$
$$\int\sqrt{a^2-x^2}dx=\frac{1}{2}x\sqrt{a^2-x^2}+\frac{1}{2}a^2\tan^{-1}\frac{x}{\sqrt{a^2-x^2}}$$
$$\int\frac{x^2}{\sqrt{x^2\pm a^2}}dx=\frac{1}{2}x\sqrt{x^2\pm a^2}\mp\frac{1}{2}a^2\ln\left|x+\sqrt{x^2\pm a^2}\right|$$
$$\int\frac{1}{\sqrt{x^2\pm a^2}}dx=\ln\left|x+\sqrt{x^2\pm a^2}\right|$$
$$\int\frac{1}{\sqrt{a^2-x^2}}dx=\sin^{-1}\frac{x}{a}\quad\int\frac{x}{\sqrt{x^2\pm a^2}}dx=\sqrt{x^2\pm a^2}\quad\int\frac{x}{\sqrt{a^2-x^2}}dx=-\sqrt{a^2-x^2}$$
$$\int\sqrt{ax^2+bx+c}dx=\frac{b+2ax}{4a}\sqrt{ax^2+bx+c}+\frac{4ac-b^2}{8a^{3/2}}\ln\left|2ax+b+2\sqrt{a(ax^2+bx+c)}\right|$$
$$\int x^ne^{ax}\mathrm{d}x=\frac{x^ne^{ax}}{a}-\frac{n}{a}\int x^{n-1}e^{ax}\mathrm{d}x$$
$$\int\sin^2axdx=\frac{x}{2}-\frac{1}{4a}\sin2ax\quad\int\sin^3axdx=-\frac{3\cos ax}{4a}+\frac{\cos3ax}{12a}$$
$$\int\cos^2axdx=\frac{x}{2}+\frac{\sin2ax}{4a}\quad\int\cos^3axdx=\frac{3\sin ax}{4a}+\frac{\sin3ax}{12a}$$
$$\int\tan axdx=-\frac{1}{a}\ln\cos ax\quad\int\tan^2axdx=-x+\frac{1}{a}\tan ax$$
$$\int x\cos axdx=\frac{1}{a^2}\cos ax+\frac{x}{a}\sin ax\quad\int x^2\cos axdx=\frac{2x\cos ax}{a^2}+\frac{a^2x^2-2}{a^3}\sin ax$$
$$\int x\sin axdx=-\frac{x\cos ax}{a}+\frac{\sin ax}{a^2}\quad\int x^2\sin axdx=\frac{2-a^2x^2}{a^3}\cos ax+\frac{2x\sin ax}{a^2}$$