
ACM TEMPLATE



ATM_osphere

Last build at October 23, 2017

Contents

1	Datastructure	4
1.1	KD tree	4
1.2	Binary indexed tree	4
1.3	Splay	4
1.4	Dynamic tree	6
1.5	Partition tree	8
2	Dynamic programming	9
2.1	RMQ	9
2.2	2D-LIS	9
3	Geometry	10
3.1	2D	10
3.1.1	Point	10
3.1.2	Line	10
3.1.3	Functions	10
3.1.4	Half plane intersection	11
3.1.5	Convex hull	12
3.1.6	Intersections of line and polygon	13
3.2	3D	14
3.2.1	Point	14
3.2.2	Functions	14
3.2.3	Convex hull	14
3.3	Circle	17
3.3.1	Functions	17
3.3.2	Union	17
3.3.3	Area of intersection part with polygon	18
3.4	Matrix	19
3.4.1	基本矩阵	19
4	Graph	20
4.1	Sap	20
4.2	Minimal cost maximal flow	21
4.3	Johnson Minimal cost flow	22
4.4	Bi-connect	25
4.5	Cut and bridge	26
4.6	Stoer-Wagner	27
4.7	Euler path	27
4.8	Strongly connected component	28
4.9	Match	29
4.9.1	Bipartite graph	29
4.9.2	Edmonds	29
4.9.3	KM	31
4.10	Clique	32
4.11	Spanning tree	33
4.11.1	Count the number of spanning tree	33
4.11.2	Spanning tree on directed graph	33
4.12	Kth shortest path	34
4.13	LCA	36
4.14	VirtualTree	37
4.15	Stable marriage problem	39

5	Math	40
5.1	Hill climbing	40
5.2	Linear Seq	40
5.3	Farey Sequence	41
5.4	FFT	41
5.4.1	Bit operation	41
5.4.2	Standard	42
5.4.3	Usage	43
5.5	Euler function	44
5.6	Ex-GCD	44
5.7	Möbius	45
5.7.1	Möbius 用于容斥	45
5.8	Prime	46
5.8.1	Get primes	46
5.8.2	Get factors	46
5.8.3	区间筛	47
5.9	Simpson	48
5.10	Chinese remainder theorem	48
5.11	Lucas	49
5.12	Primitive root	49
5.13	Inverse element	49
5.14	Calculator	49
5.15	Linear programming	51
5.16	Factorization prime number p into $x^2 + y^2$	54
5.17	Partition ways of an integer	55
5.18	Pell's equation	55
5.19	Polya	57
5.20	拉格朗日插值法	57
5.21	正多面体顶点着色	58
5.22	求和公式	58
5.23	几何公式	58
5.24	小公式	59
6	Search	59
6.1	Dancing links	59
6.1.1	Usage	61
6.2	Dancing links (A-star)	61
7	String	63
7.1	Aho-Corasick automation	63
7.1.1	Static memory version	63
7.1.2	Pointer version	63
7.2	KMP	64
7.3	E-KMP	65
7.4	Manacher	65
7.5	Suffix array	66
7.5.1	Longest common prefix	66
7.6	Smallest representation	67
7.7	Hash	67

8	Tool	68
8.1	日期函数	68
8.2	Bit compression	69
8.3	Random	69
8.4	Hash map	69
8.5	120 bit integer	70
8.6	Bash script	71
8.7	Codeblocks settings	71
8.8	Bit operation	71
8.8.1	基本操作	71
8.8.2	枚举长为 n 含 k 个 1 的 01 串	71
8.9	vimrc	72
9	Appendix	73
9.1	Template by elfness	73
9.1.1	AC machine	73
9.1.2	E-KMP	74
9.1.3	KM (list)	75
9.1.4	Nearest point pair	76
9.1.5	SA	78
9.1.6	SAP	79
9.1.7	一般图最大匹配	81
9.1.8	上下界最大流	82
9.1.9	上下界最小流	84
9.1.10	全局最小割	86
9.1.11	最小树型图	87

1 Datastructure

1.1 KD tree

```

1 bool Div[MaxN];
2 void BuildKD(int deep,int l, int r, Point p[]) {
3     if (l > r) return;
4     int mid = l + r >> 1;
5     int minX, minY, maxX, maxY;
6     minX = min_element(p + l, p + r + 1, cmpX)->x;
7     minY = min_element(p + l, p + r + 1, cmpY)->y;
8     maxX = max_element(p + l, p + r + 1, cmpX)->x;
9     maxY = max_element(p + l, p + r + 1, cmpY)->y;
10    Div[mid] = (maxX - minX >= maxY - minY);
11    nth_element(p + l, p + mid, p + r + 1, Div[mid] ? cmpX : cmpY);
12    BuildKD(l, mid - 1, p);
13    BuildKD(mid + 1, r, p);
14 }
15 long long res;
16 void Find(int l, int r, Point a, Point p[]) {
17     if (l > r) return;
18     int mid = l + r >> 1;
19     long long dist = dist2(a, p[mid]);
20     if (dist > 0)//NOTICE
21         res = min(res, dist);
22     long long d = Div[mid] ? (a.x - p[mid].x) : (a.y - p[mid].y);
23     int l1, l2, r1, r2;
24     l1 = l, l2 = mid + 1;
25     r1 = mid - 1, r2 = r;
26     if (d > 0)
27         swap(l1, l2), swap(r1, r2);
28     Find(l1, r1, a, p);
29     if (d * d < res)
30         Find(l2, r2, a, p);
31 }

```

1.2 Binary indexed tree

```

1 int read(int k) {
2     int sum = 0;
3     for (; k; k^=k&-k) sum+=tree[k];
4     return sum;
5 }
6 void update(int k, int v) {
7     for (; k<=MaxN; k+=k&-k) tree[k]+=v;
8 }
9 int find_Kth(int k) {
10    int idx = 0;
11    for(int i=20; i>=0; i--) {
12        idx |= 1 << i;
13        if(idx <= MaxN && tree[idx] < k)
14            k -= tree[idx];
15        else idx ^= 1 << i;
16    }
17    return idx + 1;
18 }

```

1.3 Splay

```

1 //Node
2 struct Node {

```

```

3   int size,key;
4   Node *c[2], *p;
5 } mem[MaxN], *cur, *nil;
6 //Initialize functions without memory pool
7 Node *newNode(int v, Node *p) {
8     cur->c[0] = cur->c[1] = nil, cur->p = p;
9     cur->size = 1;
10    cur->key = v;
11    return cur++;
12 }
13 void Init() {
14     cur = mem;
15     nil = newNode(0, cur);
16     nil->size = 0;
17 }
18 //Splay tree
19 struct SplayTree {
20     Node *root;
21     void Init() {
22         root = nil;
23     }
24     void Pushup(Node *x) {
25         if (x == nil) return;
26         Pushdown(x);
27         Pushdown(x->c[0]);
28         Pushdown(x->c[1]);
29         x->size = x->c[0]->size + x->c[1]->size + 1;
30     }
31     void Pushdown(Node *x) {
32         if (x == nil) return;
33         //do something
34     }
35     void Rotate(Node *x, int f) {
36         if (x == nil) return;
37         Node *y = x->p;
38         y->c[f ^ 1] = x->c[f], x->p = y->p;
39         if (x->c[f] != nil)
40             x->c[f]->p = y;
41         if (y->p != nil)
42             y->p->c[y->p->c[1] == y] = x;
43         x->c[f] = y, y->p = x;
44         Pushup(y);
45     }
46     void Splay(Node *x, Node *f) {
47         static Node *stack[maxn];
48         int top = 0;
49         stack[top++] = x;
50         for (Node *y = x; y != f; y = y->p)
51             stack[top++] = y->p;
52         while (top)
53             Pushdown(stack[--top]);
54         while (x->p != f) {
55             Node *y = x->p;
56             if (y->p == f)
57                 Rotate(x, x == y->c[0]);
58             else {
59                 int fd = y->p->c[0] == y;
60                 if (y->c[fd] == x)
61                     Rotate(x, fd ^ 1), Rotate(x, fd);
62                 else
63                     Rotate(y, fd), Rotate(x, fd);
64             }
65         }
66         Pushup(x);

```

```

67     if (f == nil)
68         root = x;
69 }
70 void Select(int k, Node *f) {
71     Node *x = root;
72     Pushdown(x);
73     int tmp;
74     while ((tmp = x->c[0]->size) != k) {
75         if (k < tmp) x = x->c[0];
76         else
77             x = x->c[1], k -= tmp + 1;
78         Pushdown(x);
79     }
80     Splay(x, f);
81 }
82 void Select(int l, int r) {
83     Select(l, nil), Select(r + 2, root);
84 }
85 Node *Make_tree(int a[], int l, int r, Node *p) {
86     if (l > r) return nil;
87     int mid = l + r >> 1;
88     Node *x = newNode(a[mid], p);
89     x->c[0] = Make_tree(a, l, mid - 1, x);
90     x->c[1] = Make_tree(a, mid + 1, r, x);
91     Pushup(x);
92     return x;
93 }
94 void Insert(int pos, int a[], int n) {
95     Select(pos, nil), Select(pos + 1, root);
96     root->c[1]->c[0] = Make_tree(a, 0, n - 1, root->c[1]);
97     Splay(root->c[1]->c[0], nil);
98 }
99 void Insert(int v) {
100     Node *x = root, *y = nil;
101     //Need pushdown
102     while (x != nil) {
103         y = x;
104         y->size++;
105         x = x->c[v >= x->key];
106     }
107     y->c[v >= y->key] = x = newNode(v, y);
108     Splay(x, nil);
109 }
110 void Remove(int l, int r) {
111     Select(l, r);
112     //Recycle(root->c[1]->c[0]);
113     root->c[1]->c[0] = nil;
114     Splay(root->c[1], nil);
115 }
116 };

```

1.4 Dynamic tree

```

1 struct SplayTree {
2     void Pushup(Node *x) {
3         if (x == nil) return;
4         Pushdown(x);
5         Pushdown(x->c[0]);
6         Pushdown(x->c[1]);
7         x->size = x->c[0]->size + x->c[1]->size + 1;
8     }
9     void Pushdown(Node *x) {
10        if (x == nil) return;

```

```

11     if (x->rev) {
12         x->rev = 0;
13         x->c[0]->rev ^= 1;
14         x->c[1]->rev ^= 1;
15         swap(x->c[0], x->c[1]);
16     }
17 }
18 bool isRoot(Node *x) {
19     return (x == nil) || (x->p->c[0] != x && x->p->c[1] != x);
20 }
21 void Rotate(Node *x, int f) {
22     if (isRoot(x)) return;
23     Node *y = x->p;
24     y->c[f ^ 1] = x->c[f], x->p = y->p;
25     if (x->c[f] != nil)
26         x->c[f]->p = y;
27     if (y != nil) {
28         if (y == y->p->c[1])
29             y->p->c[1] = x;
30         else if (y == y->p->c[0])
31             y->p->c[0] = x;
32     }
33     x->c[f] = y, y->p = x;
34     Pushup(y);
35 }
36 void Splay(Node *x) {
37     static Node *stack[MaxN];
38     int top = 0;
39     stack[top++] = x;
40     for (Node *y = x; !isRoot(y); y = y->p)
41         stack[top++] = y->p;
42     while (top)
43         Pushdown(stack[--top]);
44     while (!isRoot(x)) {
45         Node *y = x->p;
46         if (isRoot(y))
47             Rotate(x, x == y->c[0]);
48         else {
49             int fd = y->p->c[0] == y;
50             if (y->c[fd] == x)
51                 Rotate(x, fd ^ 1), Rotate(x, fd);
52             else
53                 Rotate(y, fd), Rotate(x, fd);
54         }
55     }
56     Pushup(x);
57 }
58 Node *Access(Node *u) {
59     Node *v = nil;
60     while (u != nil) {
61         Splay(u);
62         v->p = u;
63         u->c[1] = v;
64         Pushup(u);
65         u = (v = u)->p;
66         if (u == nil)
67             return v;
68     }
69 }
70 Node *LCA(Node *u, Node *v) {
71     Access(u);
72     return Access(v);
73 }
74 Node *Link(Node *u, Node *v) {

```



```

75     Access(u);
76     Splay(u);
77     u->rev = true;
78     u->p = v;
79 }
80 void ChangeRoot(Node *u) {
81     Access(u)->rev ^= 1;
82 }
83 Node *GetRoute(Node *u, Node *v) {
84     ChangeRoot(u);
85     return Access(v);
86 }
87 };

```

1.5 Partition tree

```

1  int n,m;
2  struct elem {
3      int v,index;
4  } a[120000];
5  int d[30][120000];
6  int s[30][120000];
7  bool cmp(elem a,elem b) {
8      if (a.v == b.v)
9          return a.index <= b.index;
10     return a.v < b.v;
11 }
12 void build(int depth,int l,int r) {
13     if (l == r)
14         return;
15     int mid = (l+r)/2;
16     int tl,tr;
17     tl = tr = 0;
18     for (int i = l; i <= r; i++) {
19         if (cmp(a[d[depth][i]],a[mid])) {
20             d[depth+1][l+tl] = d[depth][i];
21             tl++;
22         } else {
23             d[depth+1][mid+1+tr] = d[depth][i];
24             tr++;
25         }
26     }
27     s[depth][i] = tl;
28     build(depth+1,l,mid);
29     build(depth+1,mid+1,r);
30 }
31 int find(int depth,int dl,int dr,int fl,int fr,int k) {
32     if (fl == fr)
33         return a[d[depth][fl]].v;
34     int ls,rs;
35     int mid = (dl+dr)/2;
36     ls = (fl == dl)? 0 : s[depth][fl-1];
37     rs = s[depth][fr];
38     return (rs-ls < k)?
39         find(depth+1,mid+1,dr,mid+fl-dl-ls+1,mid+fr-dl-rs+1,k-(rs-ls))
40         : find(depth+1,dl,mid,dl+ls,dl+rs-1,k);
41 }
42 int main() {
43     while (scanf("%d%d",&n,&m) != EOF) {
44         for (int i = 1; i <= n; i++) {
45             scanf("%d",&a[i].v);
46             a[i].index = i;
47         }

```

```

48     sort(a+1,a+n+1,cmp);
49     for (int i = 1; i <= n; i++)
50         d[0][a[i].index] = i;
51     build(0,1,n);
52     int l,r,k;
53     for (int i = 1; i <= m; i++) {
54         scanf("%d%d%d",&l,&r,&k);
55         printf("%d\n",find(0,1,n,l,r,k));
56     }
57 }
58 return 0;
59 }

```

2 Dynamic programming

2.1 RMQ

```

1 void init() {
2     int i,j;
3     int n=N,k=1,l=0;
4     for (i=0; i<n; i++) {
5         f[i][0]=ele[i].num;
6         if (i+1>k*2) {
7             k*=2;
8             l++;
9         }
10        lent[i+1]=l;
11    }
12    for (j=1; (1<<j)-1<n; j++)
13        for (i=0; i+(1<<j)-1<n; i++)
14            f[i][j]=max(f[i][j-1],f[i+(1<<(j-1))][j-1]);
15 }
16 int fint(int x,int y) {
17     int k=lent[y-x+1];
18     return max(f[x][k],f[y-(1<<k)+1][k]);
19 }

```

2.2 2D-LIS

```

1 #include<cstdio>
2 #include<map>
3 using namespace std;
4 map<int,int> mp[100001];
5 bool check(int idx,int x,int y) {
6     if (!idx) return 1;
7     if (mp[idx].begin()->first>=x) return 0;
8     map<int,int> ::iterator it=mp[idx].lower_bound(x);
9     it--;
10    if (it->second<y) return 1;
11    else return 0;
12 }
13 int main() {
14     int n;
15     scanf("%d",&n);
16     int l=0,r=0;
17     for (int i=0; i<n; i++) {
18         int x,y;
19         scanf("%d%d",&x,&y);
20         int tl=l,tr=r;
21         while (tl<tr) {
22             int mid=(tl+tr+1)/2;

```

```

23     if (check(mid,x,y))
24         tl=mid;
25     else
26         tr=mid-1;
27 }
28 if (tl==r) r++;
29 int idx=tl+1;
30 map<int,int> ::iterator itl=mp[idx].lower_bound(x),itr=itl;
31 while (itr!=mp[idx].end() && itr->second>y) itr++;
32 if (mp[idx].find(x)!=mp[idx].end())
33     y=min(y,mp[idx][x]);
34 if (itl!=itr) mp[idx].erase(itl,itr);
35 if (mp[idx].find(x)==mp[idx].end() || mp[idx][x]>y)
36     mp[idx][x]=y;
37 }
38 printf("%d\n",r);
39 return 0;
40 }

```

3 Geometry

3.1 2D

3.1.1 Point

```

1 //Use cross product instead of atan2
2 bool cmp(const Point& a,const Point& b) {
3     if (a.y*b.y <= 0) {
4         if (a.y > 0 || b.y > 0) return a.y < b.y;
5         if (a.y == 0 && b.y == 0) return a.x < b.x;
6     }
7     return a*b > 0;
8 }

```

3.1.2 Line

```

1 Point operator &(const Line& b) const {
2     Point res = s;
3     double t = ((s - b.s) * (b.s - b.e)) / ((s - e) * (b.s - b.e));
4     res.x += (e.x - s.x) * t;
5     res.y += (e.y - s.y) * t;
6     return res;
7 }

```

3.1.3 Functions

```

1 Point nearestPointToLine(Point P, Line L) {
2     Point result;
3     double a, b, t;
4     a = L.e.x-L.s.x;
5     b = L.e.y-L.s.y;
6     t = ( (P.x-L.s.x)*a+(P.y-L.s.y)*b )/(a*a+b*b);
7     if (t >= 0 && t <= 1) {
8         result.x = L.s.x+a*t;
9         result.y = L.s.y+b*t;
10    }
11    return result;
12 }
13 //Segment
14 bool inter(Line l1,Line l2) {

```

```

15  return
16      max(l1.s.x,l1.e.x) >= min(l2.s.x,l2.e.x) &&
17      max(l2.s.x,l2.e.x) >= min(l1.s.x,l1.e.x) &&
18      max(l1.s.y,l1.e.y) >= min(l2.s.y,l2.e.y) &&
19      max(l2.s.y,l2.e.y) >= min(l1.s.y,l1.e.y) &&
20      sgn((l2.s-l1.s)*(l1.e-l1.s))*sgn((l2.e-l1.s)*(l1.e-l1.s)) <= 0 &&
21      sgn((l1.s-l2.s)*(l2.e-l2.s))*sgn((l1.e-l2.s)*(l2.e-l2.s)) <= 0;
22  }
23  bool onSeg(Line a,Point b) {
24      return ((a.s-b)*(a.e-b) == 0 &&
25          (b.x-a.s.x)*(b.x-a.e.x) <= 0 &&
26          (b.y-a.s.y)*(b.y-a.e.y) <= 0);
27  }
28  int inPoly(Point p,Point poly[], int n) {
29      int i, count;
30      Line ray, side;
31      count = 0;
32      ray.s = p;
33      ray.e.y = p.y;
34      ray.e.x = -1;//-∞
35      for (i = 0; i < n; i++) {
36          side.s = poly[i];
37          side.e = poly[(i+1)%n];
38          if(OnSeg(p, side))
39              return 1;
40          if (side.s.y == side.e.y)
41              continue;
42          if (OnSeg(side.s, ray)) {
43              if (side.s.y > side.e.y) count++;
44          } else if (OnSeg(side.e, ray)) {
45              if (side.e.y > side.s.y) count++;
46          } else if (inter(ray, side)) {
47              count++;
48          }
49      }
50      return ((count % 2 == 1) ? 0 : 2);
51  }
52  Point centerOfPolygon(Point poly[],int n) {
53      Point p, p0, p1, p2, p3;
54      double m, m0;
55      p1 = poly[0];
56      p2 = poly[1];
57      p.x = p.y = m = 0;
58      for (int i = 2; i < n; i++) {
59          p3 = poly[i];
60          p0.x = (p1.x + p2.x + p3.x) / 3.0;
61          p0.y = (p1.y + p2.y + p3.y) / 3.0;
62          m0 = p1.x*p2.y+p2.x*p3.y+p3.x*p1.y-p1.y*p2.x-p2.y*p3.x-p3.y*p1.x;
63          if (cmp(m + m0,0.0) == 0)
64              m0 += eps;
65          p.x = (m * p.x + m0 * p0.x) / (m + m0);
66          p.y = (m * p.y + m0 * p0.y) / (m + m0);
67          m = m + m0;
68          p2 = p3;
69      }
70      return p;
71  }

```

3.1.4 Half plane intersection

```

1  bool HPIcmp(Line a, Line b) {
2      if (fabs(a.k - b.k) > EPS) return a.k < b.k;
3      return ((a.s - b.s) * (b.e - b.s)) < 0;

```

```

4 }
5 Line Q[MAXN];
6 void HPI(Line line[], int n, Point res[], int &resn) {
7     int tot = n;
8     sort(line, line + n, HPIcmp);
9     tot = 1;
10    for (int i = 1; i < n; i++)
11        if (fabs(line[i].k - line[i - 1].k) > EPS)
12            line[tot++] = line[i];
13    int head = 0, tail = 1;
14    Q[0] = line[0];
15    Q[1] = line[1];
16    resn = 0;
17    for (int i = 2; i < tot; i++) {
18        if (fabs((Q[tail].e - Q[tail].s) * (Q[tail - 1].e - Q[tail - 1].s)) < EPS ||
19            fabs((Q[head].e - Q[head].s) * (Q[head + 1].e - Q[head + 1].s)) < EPS)
20            return;
21        while (head < tail && (((Q[tail] & Q[tail - 1]) - line[i].s) * (line[i].e - line[
22            i].s)) > EPS)
23            tail--;
24        while (head < tail && (((Q[head] & Q[head + 1]) - line[i].s) * (line[i].e - line[
25            i].s)) > EPS)
26            head++;
27        Q[++tail] = line[i];
28    }
29    while (head < tail && (((Q[tail] & Q[tail - 1]) - Q[head].s) * (Q[head].e - Q[head
30        ].s)) > EPS)
31        tail--;
32    while (head < tail && (((Q[head] & Q[head + 1]) - Q[tail].s) * (Q[tail].e - Q[tail
33        ].s)) > EPS)
34        head++;
35    if (tail <= head + 1) return;
36    for (int i = head; i < tail; i++)
37        res[resn++] = Q[i] & Q[i + 1];
38    if (head < tail + 1)
39        res[resn++] = Q[head] & Q[tail];
40 }

```

3.1.5 Convex hull

```

1 bool GScmp(Point a, Point b) {
2     if (fabs(a.x - b.x) < eps)
3         return a.y < b.y - eps;
4     return a.x < b.x - eps;
5 }
6 void GS(Point p[], int n, Point res[], int &resn) {
7     resn = 0;
8     int top = 0;
9     sort(p, p+n, GScmp);
10    if (conPoint(p, n)) {
11        res[resn++] = p[0];
12        return;
13    }
14    if (conLine(p, n)) {
15        res[resn++] = p[0];
16        res[resn++] = p[n-1];
17        return;
18    }
19    for (int i = 0; i < n; i++)
20        if (resn < 2 ||
21            (res[resn-1] - res[resn-2]) * (p[i] - res[resn-1]) > 0)
22            res[resn++] = p[i];
23    else

```

```

24     --resn;
25     top = resn-1;
26     for (int i = n-2; i >= 0;)
27         if (resn < top+2 ||
28             (res[resn-1]-res[resn-2])*(p[i]-res[resn-1]) > 0)
29             res[resn++] = p[i--];
30         else
31             --resn;
32     resn--;
33 }

```

3.1.6 Intersections of line and polygon

```

1 //Intersecting segment between [la,lb]
2 int Gao(int la,int lb,Line line) {
3     if (la > lb)
4         lb += n;
5     int l = la,r = lb,mid;
6     while (l < r) {
7         mid = l+r+1>>1;
8         if (cmp((line.e-line.s)*(p[la]-line.s),0)*cmp((line.e-line.s)*(p[mid]-line.s),0)
9             >= 0)
10             l = mid;
11         else
12             r = mid-1;
13     }
14     return l%n;
15 }
16 double theta[maxn];
17 void Gettheta() {
18     for (int i = 0; i < n; i++) {
19         Point v = p[(i+1)%n]-p[i];
20         theta[i] = atan2(v.y,v.x);
21     }
22     for (int i = 1; i < n; i++)
23         if (theta[i-1] > theta[i]+eps)
24             theta[i] += 2*pi;
25 }
26 void Calc(Line l) {
27     double tnow;
28     Point v = l.e-l.s;
29     tnow = atan2(v.y,v.x);
30     if (cmp(tnow,theta[0]) < 0) tnow += 2*pi;
31     int pl = lower_bound(theta,theta+n,tnow)-theta;
32     tnow = atan2(-v.y,-v.x);
33     if (cmp(tnow,theta[0]) < 0) tnow += 2*pi;
34     int pr = lower_bound(theta,theta+n,tnow)-theta;
35     //Farest points with l on polygon
36     pl = pl%n;
37     pr = pr%n;
38     if (cmp(v*(p[pl]-l.s),0)*cmp(v*(p[pr]-l.s),0) >= 0)
39         return 0.0;
40     int xa = Gao(pl,pr,l);
41     int xb = Gao(pr,pl,l);
42     if (xa > xb) swap(xa,xb);
43     //Intersecting with line  $P_{xa} \rightarrow P_{xa+1}$  and  $P_{xb} \rightarrow P_{xb+1}$ 
44     if (cmp(v*(p[xa+1]-p[xa]),0) == 0) return 0.0;
45     if (cmp(v*(p[xb+1]-p[xb]),0) == 0) return 0.0;
46     Point pa,pb;
47     //Intersections
48     pa = Line(p[xa],p[xa+1])&l;
49     pb = Line(p[xb],p[xb+1])&l;
50 }

```

3.2 3D

3.2.1 Point

```

1 Point3D operator *(const Point3D& b) const {
2     return Point3D(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);
3 }
4 //Rotate around V, notice that |V|=1
5 Point3D Trans(Point3D pa,Point3D V,double theta) {
6     double s = sin(theta);
7     double c = cos(theta);
8     double x,y,z;
9     x = V.x;
10    y = V.y;
11    z = V.z;
12    Point3D pp =
13        Point3D(
14            (x*x*(1-c)+c)*pa.x+(x*y*(1-c)-z*s)*pa.y+(x*z*(1-c)+y*s)*pa.z,
15            (y*x*(1-c)+z*s)*pa.x+(y*y*(1-c)+c)*pa.y+(y*z*(1-c)-x*s)*pa.z,
16            (x*z*(1-c)-y*s)*pa.x+(y*z*(1-c)+x*s)*pa.y+(z*z*(1-c)+c)*pa.z);
17    return pp;
18 }

```

3.2.2 Functions

```

1 bool lineIntersect(Line3D L1, Line3D L2) {
2     Point3D s = L1.s-L1.e;
3     Point3D e = L2.s-L2.e;
4     Point3D p = s*e;
5     if (ZERO(p)) return false; //Parallel
6     p = (L2.s-L1.e)*(L1.s-L1.e);
7     return ZERO(p&L2.e); //Common face
8 }
9 //Please check whether a, b, c, d on a plane first
10 bool segmentIntersect(Point a,Point b,Point c,Point d) {
11     Point ret = (a-b)*(c-d);
12     Point t1 = (b-a)*(c-a);
13     Point t2 = (b-a)*(d-a);
14     Point t3 = (d-c)*(a-c);
15     Point t4 = (d-c)*(b-c);
16     return sgn(t1&ret)*sgn(t2&ret) < 0 &&
17         sgn(t3&ret)*sgn(t4&ret) < 0;
18 }
19 //Distance from point p to line L
20 double distance(Point3D p, Line3D L) {
21     return (Norm((p-L.s)*(L.e-L.s))/Norm(L.e-L.s));
22 }
23 //Angle between line L1 and L2,  $\theta \in [0, \pi]$ 
24 double calcTheta(Line3D L1, Line3D L2) {
25     Point3D u = L1.e - L1.s;
26     Point3D v = L2.e - L2.s;
27     return acos( (u & v) / (Norm(u)*Norm(v)) );
28 }

```

3.2.3 Convex hull

Don't forget Randomshuffle!

```

1 struct pt {
2     double x, y, z;
3     pt() {}
4     pt(double _x, double _y, double _z): x(_x), y(_y), z(_z) {}

```

```

5   pt operator - (const pt p1) {}
6   pt operator * (pt p) {}
7   double operator ^ (pt p) {}
8 };
9 struct _3DCH {
10  struct fac {
11      int a, b, c;
12      bool ok;
13  };
14  int n;
15  pt P[MAXV];
16  int cnt;
17  fac F[MAXV*8];
18  int to[MAXV][MAXV];
19  double vlen(pt a) {
20      return sqrt(a.x*a.x+a.y*a.y+a.z*a.z);
21  }
22  double area(pt a, pt b, pt c) {
23      return vlen((b-a)*(c-a));
24  }
25  double volume(pt a, pt b, pt c, pt d) {
26      return (b-a)*(c-a)^(d-a);
27  }
28  double ptof(pt &p, fac &f) {
29      pt m = P[f.b]-P[f.a], n = P[f.c]-P[f.a], t = p-P[f.a];
30      return (m * n) ^ t;
31  }
32  void deal(int p, int a, int b) {
33      int f = to[a][b];
34      fac add;
35      if (F[f].ok) {
36          if (ptof(P[p], F[f]) > eps)
37              dfs(p, f);
38          else {
39              add.a = b, add.b = a, add.c = p, add.ok = 1;
40              to[p][b] = to[a][p] = to[b][a] = cnt;
41              F[cnt++] = add;
42          }
43      }
44  }
45  void dfs(int p, int cur) {
46      F[cur].ok = 0;
47      deal(p, F[cur].b, F[cur].a);
48      deal(p, F[cur].c, F[cur].b);
49      deal(p, F[cur].a, F[cur].c);
50  }
51  bool same(int s, int t) {
52      pt &a = P[F[s].a], &b = P[F[s].b], &c = P[F[s].c];
53      return fabs(volume(a, b, c, P[F[t].a])) < eps && fabs(volume(a, b, c,
54          P[F[t].b])) < eps && fabs(volume(a, b, c, P[F[t].c])) < eps;
55  }
56  void construct() {
57      cnt = 0;
58      if (n < 4)
59          return;
60      bool sb = 1;
61      for (int i = 1; i < n; i++) {
62          if (vlen(P[0] - P[i]) > eps) {
63              swap(P[1], P[i]);
64              sb = 0;
65              break;
66          }
67      }
68      if (sb) return;

```



```

69     sb = 1;
70     for (int i = 2; i < n; i++) {
71         if (vlen((P[0] - P[1]) * (P[1] - P[i])) > eps) {
72             swap(P[2], P[i]);
73             sb = 0;
74             break;
75         }
76     }
77     if (sb) return;
78     sb = 1;
79     for (int i = 3; i < n; i++) {
80         if (fabs((P[0] - P[1]) * (P[1] - P[2]) ^ (P[0] - P[i])) > eps) {
81             swap(P[3], P[i]);
82             sb = 0;
83             break;
84         }
85     }
86     if (sb) return;
87     fac add;
88     for (int i = 0; i < 4; i++) {
89         add.a = (i+1)%4, add.b = (i+2)%4, add.c = (i+3)%4, add.ok = 1;
90         if (ptof(P[i], add) > 0)
91             swap(add.b, add.c);
92         to[add.a][add.b] = to[add.b][add.c] = to[add.c][add.a] = cnt;
93         F[cnt++] = add;
94     }
95     for (int i = 4; i < n; i++) {
96         for (int j = 0; j < cnt; j++) {
97             if (F[j].ok && ptof(P[i], F[j]) > eps) {
98                 dfs(i, j);
99                 break;
100             }
101         }
102     }
103     int tmp = cnt;
104     cnt = 0;
105     for (int i = 0; i < tmp; i++) {
106         if (F[i].ok) {
107             F[cnt++] = F[i];
108         }
109     }
110 }
111 double area() {
112     double ret = 0.0;
113     for (int i = 0; i < cnt; i++) {
114         ret += area(P[F[i].a], P[F[i].b], P[F[i].c]);
115     }
116     return ret / 2.0;
117 }
118 double volume() {
119     pt O(0, 0, 0);
120     double ret = 0.0;
121     for (int i = 0; i < cnt; i++) {
122         ret += volume(O, P[F[i].a], P[F[i].b], P[F[i].c]);
123     }
124     return fabs(ret / 6.0);
125 }
126 int facetCnt_tri() {
127     return cnt;
128 }
129 int facetCnt() {
130     int ans = 0;
131     for (int i = 0; i < cnt; i++) {
132         bool nb = 1;

```

```

133     for (int j = 0; j < i; j++) {
134         if (same(i, j)) {
135             nb = 0;
136             break;
137         }
138     }
139     ans += nb;
140 }
141 return ans;
142 }
143 pt Fc[MAXV*8];
144 double V[MAXV*8];
145 pt Center() {
146     pt O(0,0,0);
147     for (int i = 0; i < cnt; i++) {
148         Fc[i].x = (O.x+P[F[i].a].x+P[F[i].b].x+P[F[i].c].x)/4.0;
149         Fc[i].y = (O.y+P[F[i].a].y+P[F[i].b].y+P[F[i].c].y)/4.0;
150         Fc[i].z = (O.z+P[F[i].a].z+P[F[i].b].z+P[F[i].c].z)/4.0;
151         V[i] = volume(O,P[F[i].a],P[F[i].b],P[F[i].c]);
152     }
153     pt res = Fc[0],tmp;
154     double m = V[0];
155     for (int i = 1; i < cnt; i++) {
156         if (fabs(m+V[i]) < eps)
157             V[i] += eps;
158         tmp.x = (m*res.x+V[i]*Fc[i].x)/(m+V[i]);
159         tmp.y = (m*res.y+V[i]*Fc[i].y)/(m+V[i]);
160         tmp.z = (m*res.z+V[i]*Fc[i].z)/(m+V[i]);
161         m += V[i];
162         res = tmp;
163     }
164     return res;
165 }
166 };

```

3.3 Circle

3.3.1 Functions

```

1 //Common area of two circle
2 double area(int x1,int y1,int x2,int y2,double r1,double r2) {
3     double s=dis(x2-x1,y2-y1);
4     if(r1+r2<s) return 0;
5     else if(r2-r1>s) return PI*r1*r1;
6     else if(r1-r2>s) return PI*r2*r2;
7     double q1=acos((r1*r1+s*s-r2*r2)/(2*r1*s));
8     double q2=acos((r2*r2+s*s-r1*r1)/(2*r2*s));
9     return (r1*r1*q1+r2*r2*q2-r1*s*sin(q1));
10 }

```

3.3.2 Union

```

1 for (int i = 1; i <= n; i++)
2     ans[i] = 0.0;
3 for (int i = 0; i < n; i++) {
4     tote = 0;
5     e[tote++] = Event(-pi,1);
6     e[tote++] = Event(pi,-1);
7     for (int j = 0; j < n; j++)
8         if (j != i) {
9             lab = Point(c[j].c.x-c[i].c.x,c[j].c.y-c[i].c.y);
10            AB = lab.Length();

```

```

11     AC = c[i].r;
12     BC = c[j].r;
13     if (cmp(AB+AC,BC) <= 0) {
14         e[tote++] = Event(-pi,1);
15         e[tote++] = Event(pi,-1);
16         continue;
17     }
18     if (cmp(AB+BC,AC) <= 0) continue;
19     if (cmp(AB,AC+BC) > 0) continue;
20     theta = atan2(lab.y,lab.x);
21     fai = acos((AC*AC+AB*AB-BC*BC)/(2.0*AC*AB));
22     a0 = theta-fai;
23     if (cmp(a0,-pi) < 0) a0 += 2*pi;
24     a1 = theta+fai;
25     if (cmp(a1,pi) > 0) a1 -= 2*pi;
26     if (cmp(a0,a1) > 0) {
27         e[tote++] = Event(a0,1);
28         e[tote++] = Event(pi,-1);
29         e[tote++] = Event(-pi,1);
30         e[tote++] = Event(a1,-1);
31     } else {
32         e[tote++] = Event(a0,1);
33         e[tote++] = Event(a1,-1);
34     }
35 }
36 sort(e,e+tote,Eventcmp);
37 cur = 0;
38 for (int j = 0; j < tote; j++) {
39     if (cur != 0 && cmp(e[j].tim,pre[cur]) != 0) {
40         ans[cur] += Area(e[j].tim-pre[cur],c[i].r);
41         ans[cur] += xmult(Point(c[i].c.x+c[i].r*cos(pre[cur]),c[i].c.y+c[i].r*sin(pre[
42             cur])),
43             Point(c[i].c.x+c[i].r*cos(e[j].tim),c[i].c.y+c[i].r*sin(e[j].
44                 tim)))/2.0;
45     }
46     cur += e[j].typ;
47     pre[cur] = e[j].tim;
48 }
49 for (int i = 1; i < n; i++)
50     ans[i] -= ans[i+1];

```

3.3.3 Area of intersection part with polygon

```

1 bool InCircle(Point a,double r) {
2     return cmp(a.x*a.x+a.y*a.y,r*r) <= 0;
3     //ε should big enough
4 }
5 double CalcArea(Point a,Point b,double r) {
6     Point p[4];
7     int tot = 0;
8     p[tot++] = a;
9     Point tv = Point(a,b);
10    Line tmp = Line(Point(0,0),Point(tv.y,-tv.x));
11    Point near = LineToLine(Line(a,b),tmp);
12    if (cmp(near.x*near.x+near.y*near.y,r*r) <= 0) {
13        double A,B,C;
14        A = near.x*near.x+near.y*near.y;
15        C = r;
16        B = C*C-A;
17        double tvl = tv.x*tv.x+tv.y*tv.y;
18        double tmp = sqrt(B/tvl);
19        p[tot] = Point(near.x+tmp*tv.x,near.y+tmp*tv.y);

```

```

20     if (OnSeg(Line(a,b),p[tot]) == true) tot++;
21     p[tot] = Point(near.x-tmp*tv.x,near.y-tmp*tv.y);
22     if (OnSeg(Line(a,b),p[tot]) == true) tot++;
23 }
24 if (tot == 3) {
25     if (cmp(Point(p[0],p[1]).Length(),Point(p[0],p[2]).Length()) > 0)
26         swap(p[1],p[2]);
27 }
28 p[tot++] = b;
29 double res = 0.0,theta,a0,a1,sgn;
30 for (int i = 0; i < tot-1; i++) {
31     if (InCircle(p[i],r) == true && InCircle(p[i+1],r) == true) {
32         res += 0.5*xmult(p[i],p[i+1]);
33     } else {
34         a0 = atan2(p[i+1].y,p[i+1].x);
35         a1 = atan2(p[i].y,p[i].x);
36         if (a0 < a1) a0 += 2*pi;
37         theta = a0-a1;
38         if (cmp(theta,pi) >= 0) theta = 2*pi-theta;
39         sgn = xmult(p[i],p[i+1])/2.0;
40         if (cmp(sgn,0) < 0) theta = -theta;
41         res += 0.5*r*r*theta;
42     }
43 }
44 return res;
45 }
46 area2 = 0.0;
47 for (int i = 0; i < resn; i++) //counterclockwise
48     area2 += CalcArea(p[i],p[(i+1)%resn],r);

```

3.4 Matrix

3.4.1 基本矩阵

按向量 $\overrightarrow{(x,y,z)}$ 平移:

$$\begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

按比例 (x,y,z) 缩放:

$$\begin{pmatrix} x & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

绕单位向量 $\overrightarrow{(x,y,z)}$ 旋转 $angle$ 角度:

$$\begin{pmatrix} x^2 \times (1-c) + c & x \times y \times (1-c) - z \times s & x \times z \times (1-c) + y \times s & 0 \\ y \times x \times (1-c) + z \times s & y^2 \times (1-c) + c & y \times z \times (1-c) - x \times s & 0 \\ x \times z \times (1-c) - y \times s & y \times z \times (1-c) + x \times s & z^2 \times (1-c) + c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{cases} s = \sin(angle) \\ c = \cos(angle) \end{cases}$$

以上矩阵变换都把点当作列向量, 旋转角度的正负由右手定则决定

4 Graph

4.1 Sap

```

1  const int MAXEDGE=50000;
2  const int MAXN=3000;
3  const int inf=0x3fffffff;
4  struct edges {
5      int cap,to,next,flow;
6  } edge[MAXEDGE+100];
7  struct nodes {
8      int head,label,pre,cur;
9  } node[MAXN+100];
10 int L,N;
11 int gap[MAXN+100];
12 void init(int n) {
13     L=0;
14     N=n;
15     for (int i=0; i<N; i++)
16         node[i].head=-1;
17 }
18 void add_edge(int x,int y,int z,int w) {
19     edge[L].cap=z;
20     edge[L].flow=0;
21     edge[L].to=y;
22     edge[L].next=node[x].head;
23     node[x].head=L++;
24     edge[L].cap=w;
25     edge[L].flow=0;
26     edge[L].to=x;
27     edge[L].next=node[y].head;
28     node[y].head=L++;
29 }
30 int maxflow(int s,int t) {
31     memset(gap,0,sizeof(gap));
32     gap[0]=N;
33     int u,ans=0;
34     for (int i=0; i<N; i++) {
35         node[i].cur=node[i].head;
36         node[i].label=0;
37     }
38     u=s;
39     node[u].pre=-1;
40     while (node[s].label<N) {
41         if (u==t) {
42             int min=inf;
43             for (int i=node[u].pre; i!=-1; i=node[edge[i^1].to].pre)
44                 if (min>edge[i].cap-edge[i].flow)
45                     min=edge[i].cap-edge[i].flow;
46             for (int i=node[u].pre; i!=-1; i=node[edge[i^1].to].pre) {
47                 edge[i].flow+=min;
48                 edge[i^1].flow-=min;
49             }
50             u=s;
51             ans+=min;
52             continue;
53         }
54         bool flag=false;
55         int v;
56         for (int i=node[u].cur; i!=-1; i=edge[i].next) {
57             v=edge[i].to;
58             if (edge[i].cap-edge[i].flow &&
59                 node[v].label+1==node[u].label) {

```

```

60     flag=true;
61     node[u].cur=node[v].pre=i;
62     break;
63 }
64 }
65 if (flag) {
66     u=v;
67     continue;
68 }
69 node[u].cur=node[u].head;
70 int min=N;
71 for (int i=node[u].head; i!=-1; i=edge[i].next)
72     if (edge[i].cap-edge[i].flow && node[edge[i].to].label<min)
73         min=node[edge[i].to].label;
74 gap[node[u].label]--;
75 if (!gap[node[u].label]) return ans;
76 node[u].label=min+1;
77 gap[node[u].label]++;
78 if (u!=s) u=edge[node[u].pre^1].to;
79 }
80 return ans;
81 }

```

4.2 Minimal cost maximal flow

```

1 //Use stack instead of queue when get TLE
2 int L,N;
3 int K;
4 struct edges {
5     int to,next,cap,flow,cost;
6 } edge[MAXM];
7 struct nodes {
8     int dis,pre,head;
9     bool visit;
10 } node[MAXN];
11 void init(int n) {
12     N=n;
13     L=0;
14     for (int i=0; i<N; i++)
15         node[i].head=-1;
16 }
17 void add_edge(int x,int y,int cap,int cost) {
18     edge[L].to=y;
19     edge[L].cap=cap;
20     edge[L].cost=cost;
21     edge[L].flow=0;
22     edge[L].next=node[x].head;
23     node[x].head=L++;
24     edge[L].to=x;
25     edge[L].cap=0;
26     edge[L].cost=-cost;
27     edge[L].flow=0;
28     edge[L].next=node[y].head;
29     node[y].head=L++;
30 }
31 bool spfa(int s,int t) {
32     queue<int> q;
33     for (int i=0; i<N; i++) {
34         node[i].dis=0x3fffffff;
35         node[i].pre=-1;
36         node[i].visit=0;
37     }
38     node[s].dis=0;

```

```

39 node[s].visit=1;
40 q.push(s);
41 while (!q.empty()) {
42     int u=q.front();
43     node[u].visit=0;
44     for (int i=node[u].head; i!=-1; i=edge[i].next) {
45         int v=edge[i].to;
46         if (edge[i].cap>edge[i].flow &&
47             node[v].dis>node[u].dis+edge[i].cost) {
48             node[v].dis=node[u].dis+edge[i].cost;
49             node[v].pre=i;
50             if (!node[v].visit) {
51                 node[v].visit=1;
52                 q.push(v);
53             }
54         }
55     }
56     q.pop();
57 }
58 if (node[t].pre==-1)
59     return 0;
60 else
61     return 1;
62 }
63 int mcmf(int s,int t,int &cost) {
64     int flow=0;
65     while (spfa(s,t)) {
66         int max=inf;
67         for (int i=node[t].pre; i!=-1; i=node[edge[i^1].to].pre) {
68             if (max>edge[i].cap-edge[i].flow)
69                 max=edge[i].cap-edge[i].flow;
70         }
71         for (int i=node[t].pre; i!=-1; i=node[edge[i^1].to].pre) {
72             edge[i].flow+=max;
73             edge[i^1].flow-=max;
74             cost+=edge[i].cost*max;
75         }
76         flow+=max;
77     }
78     return flow;
79 }

```

4.3 Johnson Minimal cost flow

```

1 #include <cstdio>
2 #include <cstring>
3 #include <algorithm>
4 #include <queue>
5 #include <stack>
6 using namespace std;
7 const int MAXN = 2003;
8 const int MAXM = 2000 * 1999 / 2 + 2000 * 3;
9 int N, L;
10 int head[MAXN];
11 struct Edge {
12     int to, next, flow, cost;
13 } edge[MAXM * 2];
14 int h[MAXN], dis[MAXN], pre[MAXN];
15 struct Heap {
16     int value[MAXN + 1], id[MAXN + 1];
17     int pos[MAXN];
18     int size;
19     void init() {

```

```

20     size = 1;
21 }
22 void swap2(int p, int q) {
23     swap(value[p], value[q]);
24     swap(id[p], id[q]);
25     pos[id[p]] = p;
26     pos[id[q]] = q;
27 }
28 void push_up(int p) {
29     while (p > 1 && value[p / 2] > value[p]) {
30         swap2(p, p / 2);
31         p /= 2;
32     }
33 }
34 void push_down(int p) {
35     while (p * 2 < size) {
36         int best = p;
37         if (p * 2 < size && value[p] > value[p * 2])
38             best = p * 2;
39         if (p * 2 + 1 < size && value[best] > value[p * 2 + 1])
40             best = p * 2 + 1;
41         if (p == best)
42             break;
43         swap2(p, best);
44         p = best;
45     }
46 }
47 void push(int _value, int _id) {
48     value[size] = _value;
49     id[size] = _id;
50     pos[_id] = size;
51     push_up(size++);
52 }
53 int top() {
54     return id[1];
55 }
56 void pop() {
57     value[1] = value[size - 1];
58     id[1] = id[--size];
59     pos[id[1]] = 1;
60     push_down(1);
61 }
62 void update(int _value, int _id) {
63     int p = pos[_id];
64     value[p] = _value;
65     push_up(p);
66 }
67 } heap;
68 bool inque[MAXN];
69 void init(int n) {
70     N = n;
71     L = 0;
72     memset(head, -1, 4 * n);
73 }
74 void add_edge(int u, int v, int flow, int cost) {
75     edge[L].to = v;
76     edge[L].flow = flow;
77     edge[L].cost = cost;
78     edge[L].next = head[u];
79     head[u] = L++;
80     edge[L].to = u;
81     edge[L].flow = 0;
82     edge[L].cost = -cost;
83     edge[L].next = head[v];

```



```

84     head[v] = L++;
85 }
86 void spfa(int s) {
87     memset(dis, 63, 4 * N);
88     memset(inque, 0, N);
89     memset(pre, -1, 4 * N);
90     dis[s] = 0;
91     queue<int> que;
92     que.push(s);
93     while (!que.empty()) {
94         int u = que.front();
95         inque[u] = 0;
96         que.pop();
97         for (int i = head[u]; i != -1; i = edge[i].next)
98             if (edge[i].flow) {
99                 int v = edge[i].to;
100                 if (dis[v] > dis[u] + edge[i].cost) {
101                     dis[v] = dis[u] + edge[i].cost;
102                     pre[v] = i;
103                     if (!inque[v]) {
104                         inque[v] = 1;
105                         que.push(v);
106                     }
107                 }
108             }
109     }
110 }
111 void dijkstra(int s) {
112     for (int i = 0; i < N; ++i)
113         h[i] += dis[i];
114     memset(dis, 63, 4 * N);
115     memset(pre, -1, 4 * N);
116     memset(inque, 0, N);
117     dis[s] = 0;
118     inque[s] = 1;
119     heap.init();
120     heap.push(0, s);
121     while (heap.size > 1) {
122         int u = heap.top();
123         heap.pop();
124         for (int i = head[u]; i != -1; i = edge[i].next)
125             if (edge[i].flow) {
126                 int v = edge[i].to;
127                 if (dis[v] > dis[u] + edge[i].cost + h[u] - h[v]) {
128                     dis[v] = dis[u] + edge[i].cost + h[u] - h[v];
129                     pre[v] = i;
130                     if (!inque[v]) {
131                         heap.push(dis[v], v);
132                         inque[v] = 1;
133                     } else
134                         heap.update(dis[v], v);
135                 }
136             }
137     }
138 }
139 int minimumCostFlow(int s, int t, int &cost) {
140     int flow = 0;
141     memset(h, 0, 4 * N);
142     for (spfa(s); pre[t] != -1; dijkstra(s)) {
143         int maxs = edge[pre[t]].flow;
144         for (int i = pre[t]; i != -1; i = pre[edge[i ^ 1].to])
145             maxs = min(maxs, edge[i].flow);
146         for (int i = pre[t]; i != -1; i = pre[edge[i ^ 1].to]) {
147             edge[i].flow -= maxs;

```

```

148     edge[i ^ 1].flow += maxs;
149     cost += edge[i].cost * maxs;
150 }
151     flow += maxs;
152 }
153     return flow;
154 }
155 int main() {
156     return 0;
157 }

```

4.4 Bi-connect

```

1 struct edges {
2     int to,next;
3     bool cut,visit;
4 } edge[MAXM<<1];
5 int head[MAXN],low[MAXN],dpt[MAXN],L;
6 bool visit[MAXN],cut[MAXN];
7 void init(int n) {
8     L=0;
9     memset(head,-1,4*n);
10    memset(visit,0,n);
11 }
12 void add_edge(int u,int v) {
13     edge[L].cut=edge[L].visit=0;
14     edge[L].to=v;
15     edge[L].next=head[u];
16     head[u]=L++;
17 }
18 int idx;
19 stack<int> st;
20 int bcc[MAXM];
21 void dfs(int u,int fu,int deg) {
22     cut[u]=0;
23     visit[u]=1;
24     low[u]=dpt[u]=deg;
25     int tot=0;
26     for (int i=head[u]; i!=-1; i=edge[i].next) {
27         int v=edge[i].to;
28         if (edge[i].visit)
29             continue;
30         st.push(i/2);
31         edge[i].visit=edge[i^1].visit=1;
32         if (visit[v]) {
33             low[u]=dpt[v]>low[u]?low[u]:dpt[v];
34             continue;
35         }
36         dfs(v,u,deg+1);
37         edge[i].cut=edge[i^1].cut=(low[v]>dpt[u] || edge[i].cut);
38         if (u!=fu) cut[u]=low[v]>=dpt[u]?1:cut[u];
39         if (low[v]>=dpt[u] || u==fu) {
40             while (st.top()!=i/2) {
41                 int x=st.top()*2,y=st.top()*2+1;
42                 bcc[st.top()]=idx;
43                 st.pop();
44             }
45             bcc[i/2]=idx++;
46             st.pop();
47         }
48         low[u]=low[v]>low[u]?low[u]:low[v];
49         tot++;
50     }

```

```

51     if (u==fu && tot>1) cut[u]=1;
52 }
53 int main() {
54     int n,m;
55     while (scanf("%d%d",&n,&m)!=EOF) {
56         init(n);
57         for (int i=0; i<m; i++) {
58             int u,v;
59             scanf("%d%d",&u,&v);
60             add_edge(u,v);
61             add_edge(v,u);
62         }
63         idx=0;
64         for (int i=0; i<n; i++)
65             if (!visit[i])
66                 dfs(i,i,0);
67     }
68     return 0;
69 }

```

4.5 Cut and bridge

```

1  vector<int> G[maxn];
2  int dfn[maxn], low[maxn], dfs_clock;
3  //割点答案
4  bool iscut[maxn];
5  //桥答案
6  vector<pair<int,int> > bridge;
7  void init()
8  {
9      dfs_clock = 1;
10     memset(dfn, 0, sizeof(dfn));
11     for (int i = 1; i <= n; i++)
12     {
13         G[i].clear();
14     }
15     memset(iscut, 0, sizeof(iscut));
16     bridge.clear();
17 }
18 void addedge(int u, int v)
19 {
20     G[u].push_back(v);
21     G[v].push_back(u);
22 }
23 void dfs(int u, int fa)
24 {
25     low[u] = dfn[u] = dfs_clock++;
26     int cnt = 0;
27     for (int v: G[u])
28     {
29         if (v != fa)
30         {
31             if (!dfn[v])
32             {
33                 dfs(v, u);
34                 cnt++;
35                 low[u] = min(low[u], low[v]);
36                 //判断割点 u?=1用于判断树根
37                 if (u == 1 && cnt > 1) iscut[u] = true;
38                 if (u != 1 && low[v] >= dfn[u]) iscut[u] = true;
39                 //判断桥
40                 if (low[v] > dfn[u]) bridge.push_back({u, v});
41             }

```

```

42     else
43     {
44         low[u] = min(low[u], dfn[v]);
45     }
46 }
47 }
48 if (cnt <= 1 && u == 1) iscut[u] = false;
49 }

```

4.6 Stoer-Wagner

```

1  int map[maxn][maxn];
2  int n;
3  void contract(int x,int y) {
4      int i,j;
5      for (i=0; i<n; i++)
6          if (i!=x) map[x][i]+=map[y][i],map[i][x]+=map[i][y];
7      for (i=y+1; i<n; i++) for (j=0; j<n; j++) {
8          map[i-1][j]=map[i][j];
9          map[j][i-1]=map[j][i];
10     }
11     n--;
12 }
13 int w[maxn],c[maxn];
14 int sx,tx;
15 int mincut() {
16     int i,j,k,t;
17     memset(c,0,sizeof(c));
18     c[0]=1;
19     for (i=0; i<n; i++) w[i]=map[0][i];
20     for (i=1; i+1<n; i++) {
21         t=k=-1;
22         for (j=0; j<n; j++) if (c[j]==0&&w[j]>k)
23             k=w[t=j];
24         c[sx=t]=1;
25         for (j=0; j<n; j++) w[j]+=map[t][j];
26     }
27     for (i=0; i<n; i++) if (c[i]==0) return w[tx=i];
28 }
29 int main() {
30     int i,j,k,m;
31     while (scanf("%d%d",&n,&m)!=EOF) {
32         memset(map,0,sizeof(map));
33         while (m--) {
34             scanf("%d%d%d",&i,&j,&k);
35             map[i][j]+=k;
36             map[j][i]+=k;
37         }
38         int mint=999999999;
39         while (n>1) {
40             k=mincut();
41             if (k<mint) mint=k;
42             contract(sx,tx);
43         }
44         printf("%d\n",mint);
45     }
46     return 0;
47 }

```

4.7 Euler path

```

1 //Directed graph
2 void solve(int x) {
3     int i;
4     if (!match[x]) {
5         path[++l]=x;
6         return ;
7     }
8     for (i=1; i<=n; i++)
9         if (b[x][i]) {
10             b[x][i]--;
11             match[x]--;
12             solve(i);
13         }
14     path[++l]=x;
15 }
16 //Undirected graph
17 void solve(int x) {
18     int i;
19     if (!match[x]) {
20         path[++l]=x;
21         return ;
22     }
23     for (i=1; i<=n; i++)
24         if (b[x][i]) {
25             b[x][i]--;
26             b[i][x]--;
27             match[x]--;
28             match[i]--;
29             solve(i);
30         }
31     path[++l]=x;
32 }

```

4.8 Strongly connected component

```

1 int dfsnum[2000];
2 int low[2000];
3 int stack[2000];
4 int top;
5 int ans;
6 int an;
7 int be[2000];
8 int flag[2000];
9 void dfs(int x) {
10     dfsnum[x] = low[x] = ans++;
11     stack[++top] = x;
12     flag[x] = 1;
13     for (int i = head[x]; i != -1; i = edge[i].next) {
14         int y = edge[i].to;
15         if (dfsnum[y] == -1) {
16             dfs(y);
17             low[x] = min(low[x], low[y]);
18         } else if (flag[y] == 1)
19             low[x] = min(low[x], dfsnum[y]);
20     }
21     if (dfsnum[x] == low[x]) {
22         while (stack[top] != x) {
23             flag[stack[top]] = 0;
24             be[stack[top]] = an;
25             top--;
26         }
27         flag[x] = 0;
28         be[x] = an++;

```

```

29     top--;
30 }
31 }
32 void SC() {
33     memset(dfsnum,-1,sizeof(dfsnum));
34     memset(flag,0,sizeof(flag));
35     top = 0;
36     an = 0;
37     ans = 0;
38     for (int i = 0; i < n; i++)
39         if (dfsnum[i] == -1)
40             dfs(i);
41 }

```

4.9 Match

4.9.1 Bipartite graph

```

1 bool check(int u) {
2     for (int i=head[u]; i!=-1; i=edge[i].next) {
3         int v=edge[i].to;
4         if (!use[v]) {
5             use[v]=1;
6             if (pre[v]==-1 || check(pre[v])) {
7                 pre[v]=u;
8                 return 1;
9             }
10        }
11    }
12    return 0;
13 }
14 int match() {
15     int ret=0;
16     memset(pre,-1,sizeof(pre));
17     for (int u=1; u<=N; u++) {
18         memset(use,0,sizeof(use));
19         if (check(u))
20             ret++;
21     }
22     return ret;
23 }

```

4.9.2 Edmonds

```

1 int N;
2 bool Graph[MaxN+1][MaxN+1];
3 int Match[MaxN+1];
4 bool InQueue[MaxN+1], InPath[MaxN+1], InBlossom[MaxN+1];
5 int Head, Tail;
6 int Queue[MaxN+1];
7 int Start, Finish;
8 int NewBase;
9 int Father[MaxN+1], Base[MaxN+1];
10 int Count;
11 void CreateGraph() {}
12 void Push(int u) {
13     Queue[Tail] = u;
14     Tail++;
15     InQueue[u] = true;
16 }
17 int Pop() {
18     int res = Queue[Head];

```

```

19     Head++;
20     return res;
21 }
22 int FindCommonAncestor(int u,int v) {
23     memset(InPath,false,sizeof(InPath));
24     while (true) {
25         u = Base[u];
26         InPath[u] = true;
27         if (u == Start) break;
28         u = Father[Match[u]];
29     }
30     while (true) {
31         v = Base[v];
32         if (InPath[v]) break;
33         v = Father[Match[v]];
34     }
35     return v;
36 }
37 void ResetTrace(int u) {
38     int v;
39     while (Base[u] != NewBase) {
40         v = Match[u];
41         InBlossom[Base[u]] = InBlossom[Base[v]] = true;
42         u = Father[v];
43         if (Base[u] != NewBase) Father[u] = v;
44     }
45 }
46 void BlossomContract(int u,int v) {
47     NewBase = FindCommonAncestor(u,v);
48     memset(InBlossom,false,sizeof(InBlossom));
49     ResetTrace(u);
50     ResetTrace(v);
51     if (Base[u] != NewBase) Father[u] = v;
52     if (Base[v] != NewBase) Father[v] = u;
53     for (int tu = 1; tu <= N; tu++)
54         if (InBlossom[Base[tu]]) {
55             Base[tu] = NewBase;
56             if (!InQueue[tu]) Push(tu);
57         }
58 }
59 void FindAugmentingPath() {
60     memset(InQueue,false,sizeof(InQueue));
61     memset(Father,0,sizeof(Father));
62     for (int i = 1; i <= N; i++)
63         Base[i] = i;
64     Head = Tail = 1;
65     Push(Start);
66     Finish = 0;
67     while (Head < Tail) {
68         int u = Pop();
69         for (int v = 1; v <= N; v++)
70             if (Graph[u][v] && (Base[u] != Base[v]) && (Match[u] != v)) {
71                 if ((v == Start) ||
72                     ((Match[v] > 0) && (Father[Match[v]] > 0)))
73                     BlossomContract(u,v);
74                 else if (Father[v] == 0) {
75                     Father[v] = u;
76                     if (Match[v] > 0)
77                         Push(Match[v]);
78                     else {
79                         Finish = v;
80                         return;
81                     }
82                 }

```

```

83     }
84 }
85 }
86 void AugmentPath() {
87     int u,v,w;
88     u = Finish;
89     while (u > 0) {
90         v = Father[u];
91         w = Match[v];
92         Match[v] = u;
93         Match[u] = v;
94         u = w;
95     }
96 }
97 void Edmonds() {
98     memset(Match,0,sizeof(Match));
99     for (int u = 1; u <= N; u++)
100         if (Match[u] == 0) {
101             Start = u;
102             FindAugmentingPath();
103             if (Finish > 0) AugmentPath();
104         }
105 }
106 void PrintMatch() {}
107 int main() {
108     CreateGraph();
109     Edmonds();
110     PrintMatch();
111 }

```

4.9.3 KM

```

1 bool visx[N],visy[N];
2 int lx[N],ly[N];
3 int matchy[N];
4 int map[N][N];
5 bool find(int x) {
6     visx[x]=true;
7     int t;
8     for (int y=0; y<ycnt; y++) {
9         if (!visy[y]) {
10             t=lx[x]+ly[y]-map[x][y];
11             if (t==0) {
12                 visy[y]=true;
13                 if (matchy[y]==-1 || find(matchy[y])) {
14                     matchy[y]=x;
15                     return true;
16                 }
17             } else if (lack>t) lack=t;
18         }
19     }
20     return false;
21 }
22 void KM() {
23     memset(lx,0,sizeof(lx));
24     memset(ly,0,sizeof(ly));
25     memset(matchy,-1,sizeof(matchy));
26     for (int i=0; i<xcnt; i++)
27         for (int j=0; j<ycnt; j++)
28             if (map[i][j]>lx[i])
29                 lx[i]=map[i][j];
30     for (int x=0; x<xcnt; x++) {
31         while (true) {

```



```

32     memset(visx,false,sizeof(visx));
33     memset(visy,false,sizeof(visy));
34     lack=INFI;
35     if (find(x)) break;
36     for (int i=0; i<xcnt; i++) {
37         if (visx[i]) lx[i]-=lack;
38         if (visy[i]) ly[i]+=lack;
39     }
40 }
41 }
42 int cost=0;
43 for (int i=0; i<ycnt; i++)
44     cost+=map[matchy[i]][i];
45 }

```

4.10 Clique

```

1  bool am[100][100];
2  int ans;
3  int c[100];
4  int U[100][100];
5  int n;
6  bool dfs(int rest,int num) {
7      if (!rest) {
8          if (num>=ans)
9              return 1;
10         else
11             return 0;
12     }
13     int pre=-1;
14     for (int i=0; i<rest && rest-i+num>=ans; i++) {
15         int idx=U[num][i];
16         if (num+c[idx]<ans)
17             return 0;
18         int nrest=0;
19         for (int j=i+1; j<rest; j++)
20             if (am[idx][U[num][j]])
21                 U[num+1][nrest++]=U[num][j];
22         if (dfs(nrest,num+1))
23             return 1;
24     }
25     return 0;
26 }
27 int main() {
28     while (scanf("%d",&n),n) {
29         for (int i=0; i<n; i++)
30             for (int j=0; j<n; j++)
31                 scanf("%d",&am[i][j]);
32         ans=0;
33         for (int i=n-1; i>=0; i--) {
34             int rest=0;
35             for (int j=i+1; j<n; j++)
36                 if (am[i][j])
37                     U[0][rest++]=j;
38             ans+=dfs(rest,0);
39             c[i]=ans;
40         }
41         printf("%d\n",ans);
42     }
43     return 0;
44 }

```

最大团的压位做法 by Claris

```

1 typedef unsigned long long U;
2 typedef long long ll;
3 const int N=45;
4 //0为有边, 1为无边
5 int n,K,x,i,j,ans;bool flag;U g[N];double res;
6 inline int ctz(U s){return s?__builtin_ctzll(s):64;}
7 void BornKerbosch(U cur,U allow,U forbid){
8     if(!allow&&!forbid){
9         ans=max(ans,__builtin_popcountll(cur));
10        return;
11    }
12    if(!allow)return;
13    int pivot=ctz(allow|forbid);
14    U z=allow&~g[pivot];
15    for(int u=ctz(z);u<n;u+=ctz(z>>(u+1))+1){
16        BornKerbosch(cur|(1ULL<<u),allow&g[u],forbid&g[u]);
17        allow^=1ULL<<u,forbid|=1ULL<<u;
18    }
19 }
20 int main(){
21     scanf("%d",&n);
22     for(i=0;i<n;i++)g[i]=(1ULL<<n)-1-(1ULL<<i);
23     for(i=0;i<n;i++)for(j=0;j<n;j++){
24         scanf("%d",&x);
25         //0为有边, 1为无边
26         if(!x&&i!=j)g[i]^=1ULL<<j;
27     }
28     BornKerbosch(0,(1ULL<<n)-1,0);
29     //ans为最大团大小
30     printf("%d",ans);
31 }

```

4.11 Spanning tree

4.11.1 Count the number of spanning tree

```

1 Matrix laplacian;
2 laplacian.clear();
3 for (int i = 0; i < n; i++)
4     for (int j = 0; j < n; j++)
5         if (i != j && G[i][j]) {
6             laplacian.a[i][j] = -1;
7             laplacian.a[i][i]++;
8         }
9 printf("%d\n",laplacian.det(n-1));

```

4.11.2 Spanning tree on directed graph

```

1 struct Edge {
2     int u,v,cost;
3 };
4 Edge e[1001*1001];
5 int pre[1001],id[1001],visit[1001],in[1001];
6 int zhuliu(int root,int n,int m,Edge e[]) {
7     int res = 0,u,v;
8     while (true) {
9         for (int i = 0; i < n; i++)
10            in[i] = inf;
11        for (int i = 0; i < m; i++)

```

```

12     if (e[i].u != e[i].v && e[i].cost < in[e[i].v]) {
13         pre[e[i].v] = e[i].u;
14         in[e[i].v] = e[i].cost;
15     }
16     for (int i = 0; i < n; i++)
17         if (i != root)
18             if (in[i] == inf) return -1;
19     int tn = 0;
20     memset(id, -1, sizeof(id));
21     memset(visit, -1, sizeof(visit));
22     in[root] = 0;
23     for (int i = 0; i < n; i++) {
24         res += in[i];
25         v = i;
26         while (visit[v] != i && id[v] == -1 && v != root) {
27             visit[v] = i;
28             v = pre[v];
29         }
30         if (v != root && id[v] == -1) {
31             for (int u = pre[v]; u != v; u = pre[u])
32                 id[u] = tn;
33             id[v] = tn++;
34         }
35     }
36     if (tn == 0) break;
37     for (int i = 0; i < n; i++)
38         if (id[i] == -1)
39             id[i] = tn++;
40     for (int i = 0; i < m; i++) {
41         int v = e[i].v;
42         e[i].u = id[e[i].u];
43         e[i].v = id[e[i].v];
44         if (e[i].u != e[i].v)
45             e[i++].cost -= in[v];
46         else
47             swap(e[i], e[--m]);
48     }
49     n = tn;
50     root = id[root];
51 }
52 return res;
53 }

```

4.12 Kth shortest path

```

1  #include<cstdio>
2  #include<cstring>
3  #include<queue>
4  using namespace std;
5  int K;
6  class states {
7  public:
8      int cost, id;
9  };
10 int dist[1000];
11 class cmp {
12 public:
13     bool operator ()(const states &i, const states &j) {
14         return i.cost > j.cost;
15     }
16 };
17 class cmp2 {
18 public:

```

```

19  bool operator ()(const states &i,const states &j) {
20      return i.cost+dist[i.id]>j.cost+dist[j.id];
21  }
22  };
23  struct edges {
24      int to,next,cost;
25  } edger[100000],edge[100000];
26  int headr[1000],head[1000],Lr,L;
27  void dijkstra(int s) {
28      states u;
29      u.id=s;
30      u.cost=0;
31      dist[s]=0;
32      priority_queue<states,vector<states>,cmp> q;
33      q.push(u);
34      while (!q.empty()) {
35          u=q.top();
36          q.pop();
37          if (u.cost!=dist[u.id]) continue;
38          for (int i=headr[u.id]; i!=-1; i=edger[i].next) {
39              states v=u;
40              v.id=edger[i].to;
41              if (dist[v.id]>dist[u.id]+edger[i].cost) {
42                  v.cost=dist[v.id]=dist[u.id]+edger[i].cost;
43                  q.push(v);
44              }
45          }
46      }
47  }
48  int num[1000];
49  void init(int n) {
50      Lr=L=0;
51      memset(head,-1,4*n);
52      memset(headr,-1,4*n);
53      memset(dist,63,4*n);
54      memset(num,0,4*n);
55  }
56  void add_edge(int u,int v,int x) {
57      edge[L].to=v;
58      edge[L].cost=x;
59      edge[L].next=head[u];
60      head[u]=L++;
61      edger[Lr].to=u;
62      edger[Lr].cost=x;
63      edger[Lr].next=headr[v];
64      headr[v]=Lr++;
65  }
66  int a_star(int s,int t) {
67      if (dist[s]==0x3f3f3f3f)
68          return -1;
69      priority_queue<states,vector<states>,cmp2> q;
70      states tmp;
71      tmp.id=s;
72      tmp.cost=0;
73      q.push(tmp);
74      while (!q.empty()) {
75          states u=q.top();
76          q.pop();
77          num[u.id]++;
78          if (num[t]==K)
79              return u.cost;
80          for (int i=head[u.id]; i!=-1; i=edge[i].next) {
81              int v=edge[i].to;
82              tmp.id=v;

```

```

83     tmp.cost=u.cost+edge[i].cost;
84     q.push(tmp);
85 }
86 }
87 return -1;
88 }
89 int main() {
90     int n,m;
91     scanf("%d%d",&n,&m);
92     init(n);
93     for (int i=0; i<m; i++) {
94         int u,v,x;
95         scanf("%d%d%d",&u,&v,&x);
96         add_edge(u-1,v-1,x);
97     }
98     int s,t;
99     scanf("%d%d%d",&s,&t,&K);
100    if (s==t)
101        K++;
102    dijkstra(t-1);
103    printf("%d\n",a_star(s-1,t-1));
104 }

```

4.13 LCA

```

1  typedef long long ll;
2  const int maxn = 100000 + 100;
3  const int maxk = 20;
4  struct edge
5  {
6      int v, w;
7  } es[maxn * 2];
8  int tot;
9  vector<int> G[maxn];
10 ll dis[maxn];
11 int fa[maxn][maxk];
12 int depth[maxn];
13 int n, m;
14 void init()
15 {
16     for (int i = 1; i <= n; i++)
17     {
18         G[i].clear();
19     }
20     tot = 0;
21 }
22 void addedge(int u, int v, int w)
23 {
24     G[u].push_back(tot);
25     es[tot++] = {v, w};
26     G[v].push_back(tot);
27     es[tot++] = {u, w};
28 }
29 int lca(int x, int y)
30 {
31     if (depth[x] > depth[y]) swap(x, y);
32     for (int k = maxk - 1; k >= 0; k--)
33     {
34         if (depth[fa[y][k]] >= depth[x])
35         {
36             y = fa[y][k];
37         }
38     }

```

```

39  if (x == y) return x;
40  for (int k = maxk - 1; k >= 0; k--)
41  {
42      if (fa[x][k] != fa[y][k])
43      {
44          x = fa[x][k];
45          y = fa[y][k];
46      }
47  }
48  return fa[x][0];
49 }
50 void dfs(int cur, int parent)
51 {
52     fa[cur][0] = parent;
53     for (int k = 1; k < maxk; k++)
54     {
55         fa[cur][k] = fa[fa[cur][k - 1]][k - 1];
56     }
57     for (int eno: G[cur])
58     {
59         edge & e = es[eno];
60         if (e.v != parent)
61         {
62             dis[e.v] = e.w + dis[cur];
63             depth[e.v] = 1 + depth[cur];
64             dfs(e.v, cur);
65         }
66     }
67 }
68 ll dist(int u, int v)
69 {
70     int _lca = lca(u, v);
71     return dis[u] + dis[v] - 2 * dis[_lca];
72 }

```

4.14 VirtualTree

dfs 部分参照 lca 部分自己写，碰上部分点问题优先想 dfs 序

```

1  typedef long long ll;
2  const int inf = 0x3f3f3f3f;
3  const int maxn = 100000;
4  const int maxk = 21;
5  int dfn[maxn], dfs_clock;
6  inline bool cmp(const int & i, const int & j)
7  {
8      return dfn[i] < dfn[j];
9  }
10 int fa[maxn][maxk], depth[maxn];
11 struct edge
12 {
13     int v;
14     ll c;
15 } es[maxn * 2];
16 vector<int> G[maxn], H[maxn];
17 void addH(int u, int v)
18 {
19     H[u].push_back(v);
20 }
21 int lca(int x, int y)
22 {
23     if (depth[x] < depth[y]) swap(x, y);
24     for (int k = maxk - 1; k >= 0; k--)

```

```

25 {
26     if (depth[fa[x][k]] >= depth[y])
27     {
28         x = fa[x][k];
29     }
30 }
31 if (x == y) return x;
32 for (int k = maxk - 1; k >= 0; k--)
33 {
34     if (fa[x][k] != fa[y][k])
35     {
36         x = fa[x][k];
37         y = fa[y][k];
38     }
39 }
40 return fa[x][0];
41 }
42 //vertices need to be arrange
43 int key_node[maxn], kcnt;
44 void build()
45 {
46     static int stk[maxn];
47     sort(key_node, key_node + kcnt, cmp);
48     //注释部分可用于去除关键点子树中的多余关键点
49     // int p = 0;
50     // for (int i = 1; i < kcnt; i++)
51     // {
52     //     if (lca(key_node[i], key_node[p]) != key_node[p])
53     //     {
54     //         key_node[++p] = key_node[i];
55     //     }
56     // }
57     // int sz = p + 1;
58     int sz = 0;
59     stk[sz++] = 0;
60     for (int i = 0; i < kcnt; i++)
61     {
62         int f = lca(stk[sz - 1], key_node[i]);
63         if (f == stk[sz - 1])
64         {
65             stk[sz++] = key_node[i];
66         }
67         else
68         {
69             while (sz - 2 >= 0 && depth[stk[sz - 2]] >= depth[f])
70             {
71                 addH(stk[sz - 2], stk[sz - 1]);
72                 sz--;
73             }
74             if (stk[sz - 1] != f)
75             {
76                 addH(f, stk[sz - 1]);
77                 stk[sz++] = f;
78             }
79             stk[sz++] = key_node[i];
80         }
81     }
82     for (int i = 1; i < sz; i++)
83     {
84         addH(stk[i - 1], stk[i]);
85     }
86 }

```

4.15 Stable marriage problem

假定有 n 个男生和 n 个女生，理想的拍拖状态就是对于每对情侣 (a, b) ，找不到另一对情侣 (c, d) 使得 c 更喜欢 b ， b 也更喜欢 c ，同理，对 a 来说也没有 (e, f) 使得 a 更喜欢 e 而 e 更喜欢 a ，当然最后会有一些人落单。这样子一个状态可以称为理想拍拖状态，它也有一个专业的名词叫稳定婚姻。求解这个问题可以用一个专有的算法，延迟认可算法，其核心就是让每个男生按自己喜欢的顺序逐个向女生表白，例如 leokan 向一个女生求爱，这个过程中，若这个女生没有男朋友，那么这个女生就暂时成为 leokan 的女朋友，或这个女生喜欢她现有男朋友的程度没有喜欢 leokan 高，这个女生也暂时成为 leokan 的女朋友，而她原有的男朋友则再将就找下一个次喜欢的女生来当女朋友。

```

1  #include<string.h>
2  #include<stdio.h>
3  #define N 1050
4  int boy[N][N];
5  int girl[N][N];
6  int ans[N];
7  int cur[N];
8  int n;
9  void getMarry(int g) {
10     for (int i=ans[g]+1; i<n; i++) {
11         int b=girl[g][i]-1;
12         if (cur[b]<0) {
13             ans[g]=i;
14             cur[b]=g;
15             return;
16         }
17         int og=cur[b];
18         if (boy[b][og] > boy[b][g]) {
19             cur[b]=g;
20             ans[g]=i;
21             getMarry(og);
22             return;
23         }
24     }
25 };
26 int main() {
27     int t,a;
28     scanf("%d",&t);
29     while(t--) {
30         memset(girl,0,sizeof(girl));
31         memset(boy,0,sizeof(boy));
32         scanf("%d",&n);
33         for (int i=0; i<n; i++)
34             for (int j=0; j<n; j++)
35                 scanf("%d",&girl[i][j]);
36         for (int i=0; i<n; i++)
37             for (int j=0; j<n; j++) {
38                 scanf("%d",&a);
39                 boy[i][a-1]=j;
40             }
41         memset(cur,0xff,sizeof(cur));
42         memset(ans,0xff,sizeof(ans));
43         for (int i=0; i<n; i++)
44             getMarry(i);
45         for (int i=0; i<n; i++)
46             printf("%d\n",girl[i][ans[i]]);
47     }
48     return 0;
49 }

```


5 Math

5.1 Hill climbing

Hill climbing is an useful function to get the maximum value if you don't know what to do! just make a function and follow the instruction below:

```

1 for (s = 1; s > 1e-6; f = 0)
2 {
3     if (F(dx, dy) > F(dx + s, dy)) dx += s, f = 1;
4     else if (F(dx, dy) > F(dx - s, dy)) dx -= s, f = 1;
5     else if (F(dx, dy) > F(dx, dy + s)) dy += s, f = 1;
6     else if (F(dx, dy) > F(dx, dy - s)) dy -= s, f = 1;
7     if (!f) s *= 0.7;
8 }
```

5.2 Linear Seq

杜教的递推板子，目测大概需要暴力递推阵的大小的两倍。

```

1 const ll moder = 998244353;
2 typedef vector<int> VI;
3 ll p_m(ll base, ll index)
4 {
5     ll ret = 1;
6     while(index)
7     {
8         if (index & 1) ret = ret * base % moder;
9         base = base * base % moder;
10        index >>= 1;
11    }
12    return ret;
13 }
14 int n;
15 namespace linear_seq
16 {
17     const int N = 10000 + 10;
18     ll res[N], base[N], _c[N], _md[N];
19     vector<int> Md;
20     void mul(ll *a, ll *b, int k)
21     {
22         for (int i = 0; i < k+k; i++) _c[i] = 0;
23         for (int i = 0; i < k; i++)
24             if (a[i])
25                 for (int j = 0; j < k; j++)
26                     _c[i+j] = (_c[i+j] + a[i] * b[j]) % moder;
27         for (int i = k + k - 1; i >= k; i--)
28             if (_c[i])
29                 for (int j = 0; j < Md.size(); j++)
30                     _c[i-k+Md[j]] = (_c[i-k+Md[j]] - _c[i] * _md[Md[j]]) % moder;
31         for (int i = 0; i < k; i++) a[i] = _c[i];
32     }
33     int solve(ll n, VI a, VI b)
34     {
35         ll ans = 0, pnt = 0;
36         int k = a.size();
37         for (int i = 0; i < k; i++) _md[k-1-i] = -a[i];
38         _md[k] = 1;
39         Md.clear();
40         for (int i = 0; i < k; i++) if (_md[i] != 0) Md.push_back(i);
41         for (int i = 0; i < k; i++) res[i] = base[i] = 0;
42         res[0] = 1;
43         while((1LL << pnt) <= n) pnt++;
44         for (int p = pnt; p >= 0; p--)
```

```

45     {
46         mul(res,res,k);
47         if ((n>>p) & 1)
48         {
49             for (int i = k - 1; i >= 0; i--) res[i+1] = res[i];
50             res[0] = 0;
51             for (int j = 0; j < Md.size(); j++) res[Md[j]] = (res[Md[j]] - res[k] * _md[
                Md[j]]) % moder;
52         }
53     }
54     for (int i = 0; i < k; i++) ans = (ans + res[i] * b[i]) % moder;
55     if (ans < 0) ans += moder;
56     return ans;
57 }
58 VI BM(VI s)
59 {
60     VI C(1,1), B(1,1);
61     int L = 0, m = 1, b = 1;
62     for (int n = 0; n < s.size(); n++)
63     {
64         ll d = 0;
65         for (int i = 0; i <= L; i++) d = (d + (ll)C[i]*s[n-i])%moder;
66         if (d == 0) ++m;
67         else if (2 * L <= n)
68         {
69             VI T = C;
70             ll c = moder - d * p_m(b, moder - 2) % moder;
71             while(C.size() < B.size() + m) C.push_back(0);
72             for (int i = 0; i < B.size(); i++) C[i+m] = (C[i+m] + c * B[i]) % moder;
73             L = n + 1 - L; B = T; b = d; m = 1;
74         }
75         else
76         {
77             ll c = moder - d * p_m(b, moder - 2) % moder;
78             while(C.size() < B.size() + m) C.push_back(0);
79             for (int i = 0; i < B.size(); i++) C[i+m] = (C[i+m] + c * B[i]) % moder;
80             ++m;
81         }
82     }
83     return C;
84 }
85 int gao(VI a, ll n)
86 {
87     VI c = BM(a);
88     c.erase(c.begin());
89     for (int i = 0; i < c.size(); i++) c[i] = (moder - c[i]) % moder;
90     return solve(n,c,VI(a.begin(), a.begin() + c.size()));
91 }
92 }
93 int main() {
94     while(~scanf("%d",&n))
95         printf("%d\n",linear_seq::gao(VI{1, 4, 12, 33, 88, 232, 609},n));
96 }

```

5.3 Farey Sequence

用于求最接近某个真值的有理数分数表示 TODO

5.4 FFT

5.4.1 Bit operation

$$tf(X1, X2) = (tf(X1) - tf(X2), tf(X1) + tf(X2))$$

异或: $tf(X1, X2) = (tf(X1) - tf(X2), tf(X1) + tf(X2))$

与: $tf(x1, x2) = (tf(x1) + tf(x2), tf(x1))$

```

1 // Transforms the interval [x, y) in a.
2 void transform(int x, int y) {
3     if ( x == y - 1) {
4         return;
5     }
6     int l2 = ( y - x ) / 2;
7     int z = x + l2;
8     transform(x, z);
9     transform(z, y);
10    for (int i=x; i<z; i++) {
11        int x1 = a[i];
12        int x2 = a[i+l2];
13        a[i] = (x1 - x2 + MOD) % MOD;
14        a[i+l2] = (x1 + x2) % MOD;
15    }
16 }
17 // Reverses the transform in
18 // the interval [x, y) in a.
19 void untransform(int x, int y) {
20     if ( x == y - 1) {
21         return;
22     }
23     int l2 = ( y - x ) / 2;
24     int z = x + l2;
25     for (int i=x; i<z; i++) {
26         long long y1 = a[i];
27         long long y2 = a[i+l2];
28         // x1 - x2 = y1
29         // x1 + x2 = y2
30         // 2 * x1 = y1 + y2
31         // 2 * x2 = y2 - y1
32
33         // In order to solve those equations, we need to divide by 2
34         // But we are performing operations modulo 1000000007
35         // that needs us to find the modular multiplicative inverse of 2.
36         // That is saved in the INV2 variable.
37
38         a[i] = (int)((y1 + y2)*INV2) % MOD ;
39         a[i+l2] = (int)((y2 - y1 + MOD)*INV2) % MOD ;
40     }
41     untransform(x, z);
42     untransform(z, y);
43 }

```

5.4.2 Standard

```

1 struct vir {
2     long double re, im;
3     vir(long double a = 0, long double b = 0) {
4         re = a;
5         im = b;
6     }
7     vir operator +(const vir& b) const {
8         return vir(re + b.re, im + b.im);
9     }
10    vir operator -(const vir& b) const {
11        return vir(re - b.re, im - b.im);
12    }
13    vir operator *(const vir& b) const {
14        return vir(re * b.re - im * b.im, re * b.im + im * b.re);

```

```

15     };
16 };
17 void change(vir *x, int len, int loglen) {
18     int i, j, k, t;
19     for (i = 0; i < len; i++) {
20         t = i;
21         for (j = k = 0; j < loglen; j++, t >>= 1)
22             k = (k << 1) | (t & 1);
23         if (k < i) {
24             vir wt = x[k];
25             x[k] = x[i];
26             x[i] = wt;
27         }
28     }
29 }
30 void fft(vir *x, int len, int loglen) {
31     int i, j, t, s, e;
32     change(x, len, loglen);
33     t = 1;
34     for (i = 0; i < loglen; i++, t <<= 1) {
35         s = 0;
36         e = s + t;
37         while (s < len) {
38             vir a, b, wo(cos(PI / t), sin(PI / t)), wn(1, 0);
39             for (j = s; j < s + t; j++) {
40                 a = x[j];
41                 b = x[j + t] * wn;
42                 x[j] = a + b;
43                 x[j + t] = a - b;
44                 wn = wn * wo;
45             }
46             s = e + t;
47             e = s + t;
48         }
49     }
50 }
51 void dit_fft(vir *x, int len, int loglen) {
52     int i, j, s, e, t = 1 << loglen;
53     for (i = 0; i < loglen; i++) {
54         t >>= 1;
55         s = 0;
56         e = s + t;
57         while (s < len) {
58             vir a, b, wn(1, 0), wo(cos(PI / t), -sin(PI / t));
59             for (j = s; j < s + t; j++) {
60                 a = x[j] + x[j + t];
61                 b = (x[j] - x[j + t]) * wn;
62                 x[j] = a;
63                 x[j + t] = b;
64                 wn = wn * wo;
65             }
66             s = e + t;
67             e = s + t;
68         }
69     }
70     change(x, len, loglen);
71     for (i = 0; i < len; i++)
72         x[i].re /= len;
73 }

```

5.4.3 Usage

```
1 | vir x1[MAXN], x2[MAXN];
```

```

2 void solve(long long *a, int lena, long long *b, int lenb, long long *ret, int& len)
  {
3   int len1 = lena << 1;
4   int len2 = lenb << 1;
5   len = 1;
6   int loglen = 0;
7   while (len < len1 || len < len2) {
8     len <<= 1;
9     loglen++;
10  }
11  for (int i = 0; i < lena; i++)
12    x1[i] = vir(a[i], 0);
13  for (int i = lena; i < len; i++)
14    x1[i] = vir(0, 0);
15  for (int i = 0; i < lenb; i++)
16    x2[i] = vir(b[i], 0);
17  for (int i = lenb; i < len; i++)
18    x2[i] = vir(0, 0);
19  fft(x1, len, loglen);
20  fft(x2, len, loglen);
21  for (int i = 0; i < len; i++)
22    x1[i] = x1[i] * x2[i];
23  dit_fft(x1, len, loglen);
24  for (int i = 0; i < len; i++)
25    ret[i] = (long long)(x1[i].re + 0.5);
26 }

```

5.5 Euler function

```

1 int getEuler(int x) {
2   getFactor(x);
3   int ret=x;
4   for (int i=0; i<N; i++)
5     ret = ret/fac[i]*(fac[i]-1);
6   return ret;
7 }
8 void getEuler2() {
9   memset(euler,0,sizeof(euler));
10  euler[1] = 1;
11  for (int i = 2; i <= 3000000; i++) {
12    if (!euler[i]) {
13      for (int j = i; j <= 3000000; j += i) {
14        if (!euler[j])
15          euler[j] = j;
16        euler[j] = euler[j]/i*(i-1);
17      }
18    }
19  }
20 }

```

5.6 Ex-GCD

```

1 //Find one solution (x,y) of  $ax + by = gcd(a,b)$ 
2 long long ex_gcd(long long a, long long b, long long &x, long long &y) {
3   if (b) {
4     long long ret = ex_gcd(b, a%b, x, y), tmp = x;
5     x = y;
6     y = tmp - (a/b)*y;
7     return ret;
8   } else {
9     x = 1;

```

```

10 |     y = 0;
11 |     return a;
12 | }
13 | }

```

5.7 Möbius

两个公式:

$$F(n) = \sum_{d|n} f(d) \implies f(n) = \sum_{d|n} \mu(d) F\left(\frac{n}{d}\right) \quad (1)$$

$$F(n) = \sum_{n|d} f(d) \implies f(n) = \sum_{n|d} \mu\left(\frac{d}{n}\right) F(d) \quad (2)$$

$$\mu(n) = \begin{cases} 1 & n = 1 \\ (-1)^k & n = p_1 p_2 \dots p_k \\ 0 & \text{其余情况} \end{cases}$$

5.7.1 Möbius 用于容斥

容斥原理: 在集合 S 中至少具有 $P_1 P_2 \dots P_m$ 中一个元素的个数是:

$$|S_1 \cup S_2 \cup S_3 \dots \cup S_n| = \sum |S_i| - \sum |S_i \cup S_j| + \dots + \sum (-1)^{m+1} |S_1 \cup S_2 \dots \cup S_m|$$

常用转化式:

$$\sum_{i=1}^n \left\lfloor \frac{n}{i} \right\rfloor = \sum_{i=1}^n d(i), d(n) \text{ 是 } n \text{ 的正因子数目 (埃筛)}$$

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

```

1 | const int maxn = 10000000 + 100;
2 | int primes[maxn], ptot;
3 | int mu[maxn];
4 | bool nprime[maxn];
5 | void init()
6 | {
7 |     nprime[1] = true;
8 |     mu[1] = 1;
9 |     for (int i = 2; i < maxn; i++)
10 |     {
11 |         if (!nprime[i])
12 |         {
13 |             primes[ptot++] = i;
14 |             mu[i] = -1;
15 |         }
16 |         for (int j = 0; j < ptot && i * primes[j] < maxn; j++)
17 |         {
18 |             nprime[i * primes[j]] = true;
19 |             if (i % primes[j] == 0)
20 |             {
21 |                 mu[i * primes[j]] = -mu[i];
22 |                 break;
23 |             }
24 |         }
25 |     }
26 | }

```

5.8 Prime

5.8.1 Get primes

```

1 int N;
2 bool isPrime[10001];
3 int prime[10000];
4 void getPrime(int n) {
5     memset(isPrime,1,++n);
6     N=0;
7     isPrime[0]=isPrime[1]=0;
8     for (int i=2; i<n; i++) {
9         if (isPrime[i])
10            prime[N++]=i;
11        for (int j=0; j<N && prime[j]*i<n; j++) {
12            isPrime[i*prime[j]]=0;
13            if (i%prime[j]==0)
14                break;
15        }
16    }
17 }
```

5.8.2 Get factors

```

1 const int TIME = 8;
2 int factor[100], fac_top = -1;
3 //GCD of bint
4 bint gcd(bint small,bint big) {
5     while(small) {
6         swap(small,big);
7         small%=big;
8     }
9     return abs(big);
10 }
11 //ret = (a*b)%n (n<2^62)
12 bint muti_mod(bint a,bint b,bint n) {
13     bint exp = a%n, res = 0;
14     while(b) {
15         if(b&1) {
16             res += exp;
17             if(res>n) res -= n;
18         }
19         exp <<= 1;
20         if (exp>n) exp -= n;
21         b>>=1;
22     }
23     return res;
24 }
25 // ret = (a^b)%n
26 bint mod_exp(bint a,bint p,bint m) {
27     bint exp=a%m, res=1;
28     while(p>1) {
29         if(p&1)
30             res=muti_mod(res,exp,m);
31         exp = muti_mod(exp,exp,m);
32         p>>=1;
33     }
34     return muti_mod(res,exp,m);
35 }
36 //miller-rabin
37 bool miller_rabin(bint n, int times) {
38     if(n==2)return 1;
39     if(n<2||!(n&1))return 0;
```

```

40  bint a, u=n-1, x, y;
41  int t=0;
42  while(u%2==0) {
43      t++;
44      u/=2;
45  }
46  srand(time(0));
47  for(int i=0; i<times; i++) {
48      a = rand() % (n-1) + 1;
49      x = mod_exp(a, u, n);
50      for(int j=0; j<t; j++) {
51          y = muti_mod(x, x, n);
52          if ( y == 1 && x != 1 && x != n-1 )
53              return false; //must not
54          x = y;
55      }
56      if( y!=1) return false;
57  }
58  return true;
59 }
60 bint pollard_rho(bint n,int c) {
61     bint x,y,d,i = 1,k = 2;
62     srand(time(0));
63     x = rand()%(n-1)+1;
64     y = x;
65     while(true) {
66         i++;
67         x = (muti_mod(x,x,n) + c) % n;
68         d = gcd(y-x, n);
69         if (1 < d && d < n) return d;
70         if( y == x) return n;
71         if(i == k) {
72             y = x;
73             k <<= 1;
74         }
75     }
76 }
77 void findFactor(bint n,int k) {
78     if(n==1)return;
79     if(miller_rabin(n, TIME)) {
80         factor[++fac_top] = n;
81         return;
82     }
83     bint p = n;
84     while(p >= n)
85         p = pollard_rho(p,k--);
86     findFactor(p,k);
87     findFactor(n/p,k);
88 }

```

5.8.3 区间筛

```

1  const int maxn = 1000000 + 10;
2  int primes[maxn];
3  int ptot;
4  bool nprime[maxn];
5  void intervalprime(int L, int U)
6  {
7      int i, j;
8      int SU = sqrt((double) U);
9      int d = U - L + 1;
10     for (int i = 0; i < d; i++) nprime[i] = 0;
11     //去偶数, 可删(改下面起始点为2, 步长为1)

```



```

12  for (int i = (L % 2 != 0); i < d; i+= 2) nprime[i] = 1;
13
14  for (int i = 3; i <= SU; i += 2)
15  {
16      if (i > L && nprime[i - L]) continue;
17      j = (L / i) * i;
18      if (j < L) j += i;
19      j = j - L;
20      for (; j < d; j += i) nprime[j] = 1;
21  }
22  if (L <= 1) nprime[1 - L] = 1;
23  if (L <= 2) nprime[2 - L] = 0;
24  ptot = 0;
25  for (int i = 0; i < d; i++) if (!nprime[i]) primes[ptot++] = i + L;
26  }

```

5.9 Simpson

```

1  double Simp(double l,double r) {
2      double h = (r-l)/2.0;
3      return h*(calc(l)+4*calc((l+r)/2.0)+calc(r))/3.0;
4  }
5  double rSimp(double l,double r) {
6      double mid = (l+r)/2.0;
7      if (abs((Simp(l,r)-Simp(l,mid)-Simp(mid,r)))/15 < eps)
8          return Simp(l,r);
9      else
10         return rSimp(l,mid)+rSimp(mid,r);
11  }

```

5.10 Chinese remainder theorem

```

1  int m[10],a[10]; //x mod mi = ai
2  bool solve(int &m0,int &a0,int m,int a) {
3      int y,x;
4      int g=ex_gcd(m0,m,x,y);
5      if (abs(a-a0)%g) return 0;
6      x*=(a-a0)/g;
7      x%=m/g;
8      a0=(x*m0+a0);
9      m0*=m/g;
10     a0%=m0;
11     if (a0<0) a0+=m0;
12     return 1;
13 }
14 int MLES() {
15     bool flag=1;
16     int m0=1,a0=0;
17     for (int i=0; i<n; i++)
18         if (!solve(m0,a0,m[i],a[i])) {
19             flag=0;
20             break;
21         }
22     if (flag)
23         return a0;
24     else
25         return -1;
26 }

```

5.11 Lucas

```

1 //num[i] = i!
2 int comLucus(int n,int m,int p) {
3     int ans=1;
4     for (; n && m && ans; n/=p,m/=p) {
5         if (n%p>=m%p)
6             ans = ans*num[n%p]%p*getInv(num[m%p]%p)%p
7                 *getInv(num[n%p-m%p])%p;
8         else
9             ans=0;
10    }
11    return ans;
12 }

```

5.12 Primitive root

```

1 int getPriRoot(int p) {
2     if (p==2) return 1;
3     int phi = p - 1;
4     getFactor(phi);
5     for (int g = 2; g < p; ++g) {
6         bool flag=1;
7         for (int i = 0; flag && i < N; ++i)
8             if (power(g, phi/fac[i], p) == 1)
9                 flag=0;
10        if (flag)
11            return g;
12    }
13 }

```

5.13 Inverse element

```

1 void getInv2(int x) {
2     inv[1]=1;
3     for (int i=2; i<=x; i++)
4         inv[i]=(mod-(mod/i)*inv[mod%i]%mod)%mod;
5 }

```

5.14 Calculator

注意灵活运用。

双目运算符在 calc() 中，左结合单目运算符在 P() 中，右结合单目运算符在 calc_exp 中。（但是还没遇到过。。）

```

1 #include <iostream>
2 #include <cstdio>
3 #include <cstring>
4 #include <algorithm>
5 #include <string>
6 using namespace std;
7
8 char s[1000000];
9 int n,cur;
10 const string OP = "+-*/";
11
12 char next_char() {
13     if (cur >= n) return EOF;
14     return s[cur];

```

```

15 }
16
17 int get_priority(char ch) {
18     if (ch == '*') return 2;
19     return 1;
20 }
21
22 int P();
23
24 int calc(int a, char op, int b) {
25     if (op == '+')
26         return a+b;
27     if (op == '-')
28         return a-b;
29     if (op == '*')
30         return a*b;
31 }
32
33 int calc_exp(int p) {
34     int a = P();
35     while ((OP.find(next_char()) != OP.npos) &&
36           (get_priority(next_char()) >= p)) {
37         char op = next_char();
38         cur++;
39         a = calc(a, op, calc_exp(get_priority(op)+1));
40     }
41     return a;
42 }
43
44 int totvar, m, var[26], varid[26];
45
46 int P() {
47     if (next_char() == '-') {
48         cur++;
49         return -P();
50     } else if (next_char() == '+') {
51         cur++;
52         return P();
53     } else if (next_char() == '(') {
54         cur++;
55         int res = calc_exp(0);
56         cur++;
57         return res;
58     } else {
59         cur++;
60         return var[varid[s[cur-1]-'a']];
61     }
62 }
63
64 int id[26], minid;
65
66 int main() {
67     while (true) {
68         scanf("%d%d", &totvar, &var[0]);
69         if (totvar == 0 && var[0] == 0) break;
70         for (int i = 1; i < totvar; i++)
71             scanf("%d", &var[i]);
72         scanf("%d", &m);
73         scanf("%s", s);
74         for (int i = 0; i < 26; i++)
75             id[i] = -1;
76         minid = 0;
77         n = strlen(s);
78         for (int i = 0; i < n; i++)

```

```

79     if (s[i] >= 'a' && s[i] <= 'z') {
80         if (id[s[i]-'a'] == -1) {
81             id[s[i]-'a'] = minid;
82             minid++;
83         }
84         s[i] = 'a'+id[s[i]-'a'];
85     }
86     for (int i = 0; i < totvar; i++)
87         varid[i] = i;
88     int res = 0;
89     do {
90         cur = 0;
91         int tmp = calc_exp(0);
92         if (tmp == m) {
93             res++;
94             break;
95         }
96     } while (next_permutation(varid,varid+totvar));
97     //puts(s);
98     if (res > 0)
99         puts("YES");
100     else
101         puts("NO");
102 }
103 return 0;
104 }

```

5.15 Linear programming

```

1  #define MAXM 20 //max num of basic variables
2  #define INF 1E200
3
4  double A[MAXM+5][MAXN+MAXM+5];
5  double b[MAXM+5],c[MAXN+MAXM+5];
6  int N[MAXN+5],B[MAXM+5];
7  double X[MAXN+MAXM+5],V;
8  int n,m,R,C,nCnt,bCnt;
9  int v1[MAXN],v2[MAXN];
10
11 int fcmp(double a,double b) {
12     if(fabs(a-b)<1E-7) return 0;
13     if(a>b) return 1;
14     return -1;
15 }
16
17 void Pivot(int l,int e) {
18     double t=A[l][e],p=c[e];
19     b[l]=b[l]/t;
20     for(int i=1; i<=C; i++)
21         A[l][i]/=t;
22     V=V-c[e]*b[l];
23     for(int i=1; i<=R; i++) {
24         if(i==l || fcmp(A[i][e],0.0)==0)
25             continue;
26         t=A[i][e];
27         b[i]=b[i]-t*b[l];
28         for(int j=1; j<=C; j++)
29             A[i][j]=A[i][j]-t*A[l][j];
30     }
31     for(int i=1; i<=C; i++)
32         c[i]=c[i]-p*A[l][i];
33     for(int i=1; i<=nCnt; i++) {
34         if(N[i]==e) {

```

```

35     N[i]=B[l];
36     break;
37 }
38 }
39 B[l]=e;
40 }
41
42 bool Process(double P[]) {
43     while(true) {
44         int e=-1;
45         double mV=-INF;
46         for(int i=1; i<=nCnt; i++)
47             if(fcmp(P[N[i]],mV)==1)
48                 mV=P[N[i]],e=N[i];
49
50         if(fcmp(mV,0.0)<=0) break;
51         int l=-1;
52         mV=INF;
53         for(int i=1; i<=bCnt; i++) {
54             if(fcmp(A[i][e],0.0)==1) {
55                 double t=b[i]/A[i][e];
56                 if(fcmp(mV,t)==1 || (fcmp(mV,t)==0&&(l==-1 || B[l]>B[i])))
57                     mV=t,l=i;
58             }
59         }
60         if(l==-1) return false;
61         Pivot(l,e);
62     }
63     return true;
64 }
65
66 bool initSimplex() {
67     nCnt=bCnt=0;
68     for(int i=1; i<=n; i++)
69         N[++nCnt]=i;
70     for(int i=1; i<=m; i++)
71         B[++bCnt]=i+n,A[i][n+i]=1.0;
72     R=bCnt,C=bCnt+nCnt;
73     double minV=INF;
74     int p=-1;
75     for(int i=1; i<=m; i++)
76         if(fcmp(minV,b[i])==1)
77             minV=b[i],p=i;
78     if(fcmp(minV,0.0)>=0)
79         return true;
80     N[++nCnt]=n+m+1;
81     R++,C++;
82     for(int i=0; i<=C; i++)
83         A[R][i]=0.0;
84     for(int i=1; i<=R; i++)
85         A[i][n+m+1]=-1.0;
86     Pivot(p,n+m+1);
87     if(!Process(A[R])) return false;
88     if(fcmp(b[R],0.0)!=0)
89         return false;
90     p=-1;
91     for(int i=1; i<=bCnt&&p==-1; i++)
92         if(B[i]==n+m+1) p=i;
93     if(p!=-1) {
94         for(int i=1; i<=nCnt; i++) {
95             if(fcmp(A[p][N[i]],0.0)!=0) {
96                 Pivot(p,N[i]);
97                 break;
98             }

```

```

99     }
100 }
101 bool f=false;
102 for(int i=1; i<=nCnt; i++) {
103     if(N[i]==n+m+1) f=true;
104     if(f&& i+1<=nCnt)
105         N[i]=N[i+1];
106 }
107 nCnt--;
108 R--,C--;
109 return true;
110 }
111
112 //-1: no solution 1: no bound 0: has a solution -V
113 int Simplex() {
114     if(!initSimplex())
115         return -1;
116     if(!Process(c))
117         return 1;
118     for(int i=1; i<=nCnt; i++)
119         X[N[i]]=0.0;
120     for(int i=1; i<=bCnt; i++)
121         X[B[i]]=b[i];
122     return 0;
123 }
124
125 int main() {
126     //n = 1;m=1;
127     //V= 0.0;
128     //c[1] = 1.0;
129     //A[1][1] = 1.0;
130     //b[1] = 5.0;
131     //Simplex();
132     //printf("V = %.3f\n",V);
133
134     while(scanf("%d",&v1[1]) == 1) {
135         for(int i = 2; i<=6; i++)
136             scanf("%d",&v1[i]);
137         n = 4;
138         m = 6;
139         for(int i = 0 ; i<=m+1; i++)
140             for(int j=0; j<=n+m+2; j++)
141                 A[i][j] = c[j] = 0;
142         memset(b,0,sizeof(b));
143         V = 0.0;
144         /*
145         n 为未知数个数
146         m 为约束个数
147         目标: siama(c[i]*xi)
148         约束: sigma(A[i][j]*xj) <=b[i]; j = 1 ... n
149         解存在X里面
150         */
151         b[1] = v1[1] ;
152         A[1][1] = 1;
153         A[1][4] = 1;
154         b[2] = v1[2] ;
155         A[2][1] = 1;
156         A[2][3] = 1;
157         b[3] = v1[3] ;
158         A[3][3] = 1;
159         A[3][4] = 1;
160         b[4] = v1[4] ;
161         A[4][2] = 1;
162         A[4][3] = 1;

```

```

163     b[5] = v1[5] ;
164     A[5][2] = 1;
165     A[5][4] = 1;
166     b[6] = v1[6] ;
167     A[6][1] = 1;
168     A[6][2] = 1;
169     c[1] = 1;
170     c[2] = 1;
171     c[3] = 1;
172     c[4] = 1;
173     Simplex();
174     //printf("V = %.3f\n",V);
175     printf("%.3f_%.3f_%.3f_%.3f\n",X[1],X[2],X[3],X[4]);
176
177 }
178 return 0;
179 }

```

5.16 Factorization prime number p into $x^2 + y^2$

```

1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  int p,expp,A,B,aa,ans,tt;
5  long long M;
6  long long exp(int a,int b,long long mod) {
7      long long ans=1,num=a;
8      while (b!=0) {
9          if (b&1) {
10             ans=((ans%mod)*(num%mod))%mod;
11         }
12         num=((num%mod)*(num%mod))%mod;
13         b>>=1;
14     }
15     return ans;
16 }
17 int calcu(int p,int &x,int &y) {
18     if (p%4!=1) return -1;
19     else {
20         expp=(p-1)/4;
21         A,B;
22         while (1) {
23             aa=rand()%p;
24             if (aa==0) continue;
25             A=exp(aa,expp,p);
26             ans=((long long)A%p)*((long long)A%p)%p;
27             if (ans==p-1) break;
28         }
29         B=1;
30         M=((long long)A*(long long)A+(long long)B*(long long)B)/p;
31         if (M!=1) B=p;
32         while (M!=1) {
33             if (B>A) {
34                 tt=A;
35                 A=B;
36                 B=tt;
37             }
38             tt=A;
39             A=B;
40             B=tt%B;
41             M=((long long)A*(long long)A
42                +(long long)B*(long long)B)/p;
43         }

```

```

44     if (B<=A) {
45         x=B;
46         y=A;
47     } else {
48         x=A;
49         y=B;
50     }
51 }
52 }
53 int main() {
54     while (scanf("%d",&p)!=EOF) {
55         int x,y;
56         if (calcu(p,x,y)!=-1)
57             ;
58     }
59     return 0;
60 }

```

5.17 Partition ways of an integer

$O(n\sqrt{n})$

```

1  #include <cstdio>
2  #include <cmath>
3  #include <cstring>
4  #include <map>
5  #include <algorithm>
6  using namespace std;
7  bool check(int x) {
8      for (int i=2; i*i<=x; i++)
9          if (x%i==0)
10             return 0;
11     return 1;
12 }
13 int p[1000000];
14 inline int calc(int x) {
15     return x*(x+1)/2;
16 }
17 int main() {
18     p[0]=1;
19     for (int i=1; i<1000000; i++) {
20         for (int j=1,k=1; calc(j)<=i; j++,k*=-1) {
21             p[i]+=k*p[i-calc(j)];
22             if (p[i]<0)
23                 p[i]+=1000000;
24             if (p[i]>=1000000)
25                 p[i]-=1000000;
26             if (calc(-j)<=i)
27                 p[i]+=k*p[i-calc(-j)];
28             if (p[i]<0)
29                 p[i]+=1000000;
30             if (p[i]>=1000000)
31                 p[i]-=1000000;
32         }
33         if (!p[i])
34             printf("%d\n",i);
35     }
36     return 0;
37 }

```

5.18 Pell's equation

```

1  import java.math.BigInteger;

```



```

2 import java.util.*;
3 public class Main {
4     public static class Fraction {
5         public BigInteger num,den;
6         public Fraction() {
7             num=BigInteger.ZERO;
8             den=BigInteger.ONE;
9         }
10        public Fraction(int _num,int _den) {
11            num=BigInteger.valueOf(_num);
12            den=BigInteger.valueOf(_den);
13        }
14        public Fraction(BigInteger _num,BigInteger _den) {
15            num=_num;
16            den=_den;
17        }
18        public Fraction gen() {
19            BigInteger g=num.gcd(den);
20            return new Fraction(num.divide(g),den.divide(g));
21        }
22        public Fraction add(Fraction x) {
23            return new Fraction(x.num.multiply(den).add(num.multiply(x.den)),x.den.multiply
                (den)).gen();
24        }
25        public Fraction reciprocal() {
26            return new Fraction(den,num);
27        }
28        public void out() {
29            System.out.println(num+"/"+den);
30        }
31    }
32    public static BigInteger sqrt(BigInteger a) {
33        BigInteger b=a;
34        while (a.compareTo(b.multiply(b))<0)
35            b=b.multiply(b).add(a).divide(b.multiply(BigInteger.valueOf(2)));
36        return b;
37    }
38    public static boolean check(Fraction x,int n) {
39        return x.num.multiply(x.num).add(x.den.multiply(x.den.multiply(BigInteger.valueOf
            (n))).negate()).compareTo(BigInteger.ONE)==0;
40    }
41    static int p[]=new int[1000];
42    static int l;
43    public static void main(String[] args) {
44        BigInteger ans=BigInteger.ZERO;
45        int idx=0;
46        for (int n=2,r=2; n<=1000; n++) {
47            if (n==r*r) {
48                r++;
49                continue;
50            }
51            int tmp=calc(n,0,1),a=tmp,b=n-tmp*tmp;
52            p[0]=tmp;
53            l=1;
54            while (true) {
55                tmp=calc(n,a,b);
56                p[l++]=tmp;
57                a=a-tmp*b;
58                Fraction x=getFrac();
59                if (check(x,n)) {
60                    if (ans.compareTo(x.num)<0) {
61                        ans=x.num;
62                        idx=n;
63                    }

```

```

64         break;
65     }
66     a=-a;
67     b=(n-a*a)/b;
68 }
69 }
70 System.out.println(idx);
71 }
72 private static Fraction getFrac() {
73     Fraction ret=new Fraction(p[l-1],1);
74     for (int i=l-2; i>=0; i--)
75         ret=new Fraction(p[i],1).add(ret.reciprocal());
76     return ret;
77 }
78 private static int calc(int n, int a, int b) {
79     for (long i=2;; i++)
80         if ((i*b-a)*(i*b-a)>n)
81             return (int)i-1;
82 }
83 }

```

5.19 Polya

设 G 是 p 个对象的一个置换群，用 k 种颜色去染这 p 个对象，若一种染色方案在群 G 的作用下变为另一种方案，则这两个方案当作是同一种方案，这样的不同染色方案数为：

$$L = \frac{1}{|G|} \times \sum (k^{C(f)}), f \in G$$

$C(f)$ 为循环节， $|G|$ 表示群的置换方法数

对于有 n 个位置的手镯，有 n 种旋转置换和 n 种翻转置换

对于旋转置换：

$$C(f_i) = \gcd(n, i), i \text{ 表示一次转过 } i \text{ 颗宝石}, i = 0 \text{ 时 } c = n;$$

对于翻转置换：

如果 n 为偶数： 则有 $\frac{n}{2}$ 个置换 $C(f) = \frac{n}{2}$ ，有 $\frac{n}{2}$ 个置换 $C(f) = \frac{n}{2} + 1$

如果 n 为奇数： $C(f) = \frac{n}{2} + 1$

5.20 拉格朗日插值法

已知 $y = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1}$ 曲线上的 n 个点 $(x_1, y_1), (x_2, y_2), (x_3, y_3) \cdots (x_n, y_n)$ 用拉格朗日插值法可以不求系数可知任意 x 对应的 y 值。

$$\begin{aligned}
 y &= y_1 \frac{(x-x_2)(x-x_3) \cdots (x-x_n)}{(x_1-x_2)(x_1-x_3) \cdots (x_1-x_n)} \\
 &+ y_2 \frac{(x-x_1)(x-x_3) \cdots (x-x_n)}{(x_2-x_1)(x_2-x_3) \cdots (x_2-x_n)} \\
 &+ \cdots \\
 &+ y_n \frac{(x-x_1)(x-x_2) \cdots (x-x_{n-1})}{(x_n-x_1)(x_n-x_2) \cdots (x_n-x_{n-1})}
 \end{aligned}$$

特别的，如果 $x_1 \sim x_n$ 为连续自然数，那么对于下一个自然数对应的 y 值为：

$$y_{n+1} = (-1)^{n-1} C_n^0 y_1 + (-1)^{n-2} C_n^1 y_2 + \cdots + (-1)^0 C_n^{n-1} y_n$$

这个组合系数可以通过高斯消元求出来，前提是要猜到它满足递推关系。

5.21 正多面体顶点着色

$$\text{正四面体: } N = \frac{(n^4 + 11 \times n^2)}{12}$$

$$\text{正六面体: } N = \frac{(n^8 + 17 \times n^4 + 6 \times n^2)}{24}$$

$$\text{正八面体: } N = \frac{(n^6 + 3 \times n^4 + 12 \times n^3 + 8 \times n^2)}{24}$$

$$\text{正十二面体: } N = \frac{(n^{20} + 15 \times n^{10} + 20 \times n^8 + 24 \times n^4)}{60}$$

$$\text{正二十面体: } N = \frac{(n^{12} + 15 \times n^6 + 44 \times n^4)}{60}$$

5.22 求和公式

$$\sum k = \frac{n \times (n+1)}{2}$$

$$\sum 2k - 1 = n^2$$

$$\sum k^2 = \frac{n \times (n+1) \times (2n+1)}{6}$$

$$\sum (2k-1)^2 = \frac{n \times (4n^2-1)}{3}$$

$$\sum k^3 = \left(\frac{n \times (n+1)}{2}\right)^2$$

$$\sum (2k-1)^3 = n^2 \times (2n^2 - 1)$$

$$\sum k^4 = \frac{n \times (n+1) \times (2n+1) \times (3n^2+3n-1)}{30}$$

$$\sum k^5 = \frac{n^2 \times (n+1)^2 \times (2n^2+2n-1)}{12}$$

$$\sum k \times (k+1) = \frac{n \times (n+1) \times (n+2)}{3}$$

$$\sum k \times (k+1) \times (k+2) = \frac{n \times (n+1) \times (n+2) \times (n+3)}{4}$$

$$\sum k \times (k+1) \times (k+2) \times (k+3) = \frac{n \times (n+1) \times (n+2) \times (n+3) \times (n+4)}{5}$$

$$\sum i \times \binom{n}{i} = n \times 2^{n-1} \quad (\text{子集长度和})$$

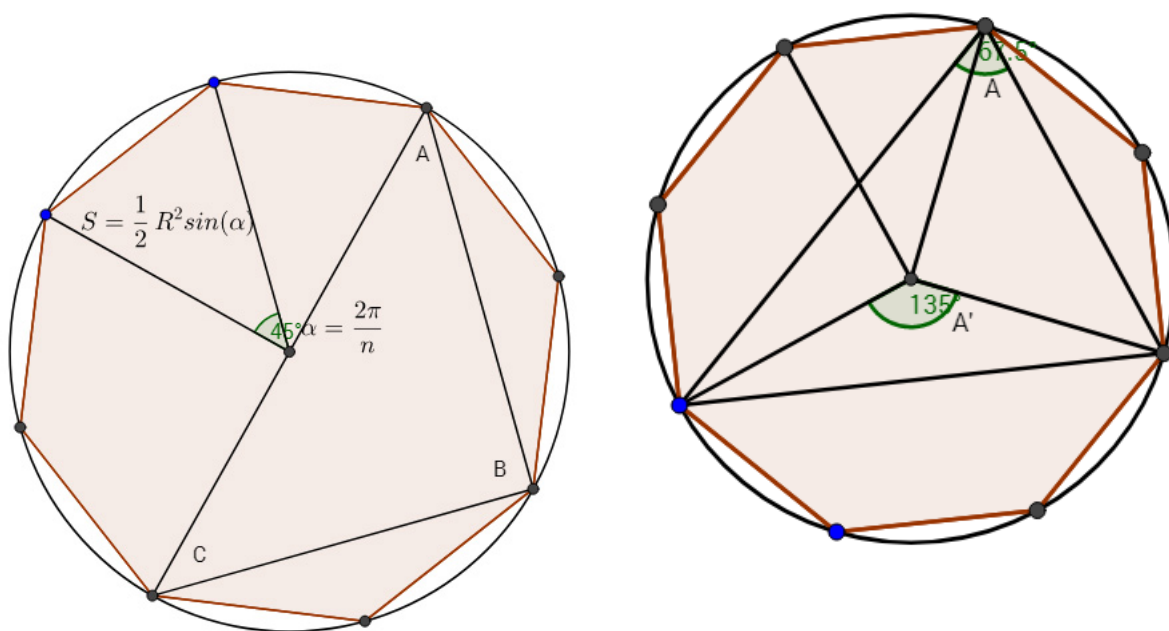
$$\sum_{i=1}^n \lfloor \frac{n}{i} \rfloor = \sum_{i=1}^n d(i), d(n) \text{ 是 } n \text{ 的正因子数目 (埃筛)}$$

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

5.23 几何公式

$$n\text{-Polygon: } \frac{n}{2} * R^2 * \sin(\frac{2\pi}{n})$$

已知任意三点求边数最少的（同时也是面积最小）正多边形：由于 $A = \frac{A'}{2}$ 所以有 $A = k \frac{\alpha}{2}$ ，结论：
 $\alpha = 2gcd(A, B, C)$ 其中 A, B, C 为三角形内角



球扇形：

全面积： $T = \pi r(2h + r_0)$ ， h 为球冠高， r_0 为球冠底面半径

体积： $V = \frac{2\pi r^2 h}{3}$

5.24 小公式

Pick 公式： $A = E \times 0.5 + I - 1$ (A 是多边形面积， E 是边界上的整点， I 是多边形内部的整点)

海伦公式： $S = \sqrt{p(p-a)(p-b)(p-c)}$ ，其中 $p = \frac{(a+b+c)}{2}$ ， abc 为三角形的三条边长

求 $\binom{n}{k}$ 中素因子 P 的个数：

1. 把 n 转化为 P 进制，并记它每个位上的和为 $S1$

2. 把 $n - k$ ， k 做同样的处理，得到 $S2$ ， $S3$

则 $\binom{n}{k}$ 中素因子 P 的个数： $\frac{S2+S3-S1}{P-1}$

部分错排公式：

$n + m$ 个数中 m 个数必须错排求排列数

```
1 dp[i] = n*dp[i-1]+(i-1)*(dp[i-1]+dp[i-2]);
2 dp[0] = n!;
3 dp[1] = n*n!;
```

$dp[m]$ 为所求解

6 Search

6.1 Dancing links

```
1 struct DLX {
2     int h,n,m,tot;
3     int U[MaxN*MaxM],D[MaxN*MaxM],L[MaxN*MaxM],R[MaxN*MaxM],Row[MaxN*MaxM],Col[MaxN*MaxM];
4 }
```

```

4  int S[MaxM],O[MaxN];
5  bool hasans;
6  void init() {
7      h = 0;
8      hasans = false;
9      tot = m+n;
10     for (int i = 0; i <= m; i++) {
11         D[i] = U[i] = Col[i] = i;
12         Row[i] = S[i] = 0;
13         L[i] = (i+m)%(m+1);
14         R[i] = (i+1)%(m+1);
15     }
16     for (int i = 1; i <= n; i++) {
17         R[i+m] = L[i+m] = i+m;
18         Row[i+m] = i;
19         Col[i+m] = 0;
20     }
21 }
22 void insert(int x,int y) {
23     tot++;
24     Row[tot] = x;
25     Col[tot] = y;
26     S[y]++;
27     int colPos,rowPos;
28     colPos = y;
29     while (true) {
30         colPos = D[colPos];
31         if (colPos == y || Row[colPos] > x) break;
32     }
33     colPos = U[colPos];
34     if (Row[colPos] == x) return;
35     U[tot] = colPos;
36     D[tot] = D[colPos];
37     U[D[tot]] = D[U[tot]] = tot;
38     rowPos = x+m;
39     while (true) {
40         rowPos = R[rowPos];
41         if (rowPos == x+m || Col[rowPos] > y) break;
42     }
43     rowPos = L[rowPos];
44     if (Col[rowPos] == y) return;
45     L[tot] = rowPos;
46     R[tot] = R[rowPos];
47     L[R[tot]] = R[L[tot]] = tot;
48 }
49 void print(int deep) {
50     for (int i = 0; i < deep; i++)
51         printf("%d_", O[i]);
52     printf("\n");
53 }
54 void cover(int col) {
55     L[R[col]] = L[col];
56     R[L[col]] = R[col];
57     for (int i = D[col]; i != col; i = D[i])
58         for (int j = R[i]; j != i; j = R[j])
59             if (Col[j] != col) {
60                 U[D[j]] = U[j];
61                 D[U[j]] = D[j];
62                 S[Col[j]]--;
63             }
64 }
65 void resume(int col) {
66     for (int i = U[col]; i != col; i = U[i])
67         for (int j = L[i]; j != i; j = L[j])

```

```

68     if (Col[j] != col) {
69         S[Col[j]]++;
70         U[D[j]] = j;
71         D[U[j]] = j;
72     }
73     L[R[col]] = col;
74     R[L[col]] = col;
75 }
76 void initDFS() {
77     for (int i = 1; i <= n; i++) {
78         L[R[i+m]] = L[i+m];
79         R[L[i+m]] = R[i+m];
80     }
81 }
82 void DFS(int deep) {
83     if (hasans == true) return;
84     if (R[0] == 0) {
85         hasans = true;
86         print(deep);
87         return;
88     };
89     int tc = R[0];
90     for (int i = R[0]; i != 0; i = R[i])
91         if (S[i] < S[tc]) tc = i;
92     cover(tc);
93     for (int i = D[tc]; i != tc; i = D[i]) {
94         int temp = O[deep];
95         O[deep] = Row[i];
96         for (int j = R[i]; j != i; j = R[j])
97             cover(Col[j]);
98         DFS(deep+1);
99         for (int j = L[i]; j != i; j = L[j])
100             resume(Col[j]);
101         O[deep] = temp;
102     }
103     resume(tc);
104 }
105 }

```

6.1.1 Usage

```

1 DLX g;
2 g.n = ROW_SIZE;
3 g.m = COL_SIZE;
4 g.init();
5 g.insert(ROW, COL);
6 g.initDFS();
7 g.DFS(0);

```

6.2 Dancing links (A-star)

```

1 namespace DLX {
2     const int MAXN = 1000;
3     const int MAXM = 400;
4     const int INF = 0x3f3f3f3f;
5     int D[MAXN * MAXM], U[MAXN * MAXM], L[MAXN * MAXM], R[MAXN * MAXM], COL[MAXN * MAXM],
        ROW[MAXN * MAXM];
6     int CNT, BEG[MAXN * MAXM], END[MAXN * MAXM], ANS, USE[MAXM], _USE[MAXM];
7     int SUM[MAXM];
8     bool vis[MAXM];
9     void init(int n) {

```

```

10  memset(BEG, 0xff, sizeof(BEG));
11  for(int i = 1; i <= n; i++)
12      SUM[L[i + 1] = R[i - 1] = D[i] = U[i] = i] = 0;
13  L[L[1] = R[n] = 0] = n, CNT = n + 1;
14  ANS = n + 1;
15  }
16  void link(int r, int c) {
17      D[CNT] = D[c], U[CNT] = c, U[D[c]] = CNT, D[c] = CNT, COL[CNT] = c, ROW[CNT] = r,
          SUM[c]++;
18      if (BEG[r] == -1) BEG[r] = END[r] = CNT;
19      R[END[r]] = CNT, L[CNT] = END[r], R[CNT] = BEG[r], L[BEG[r]] = CNT, END[r] = CNT++;
20  }
21  void DLX_Remove_Repeat(int c) {
22      for (int i = D[c]; i != c; i = D[i])
23          L[R[i]] = L[i], R[L[i]] = R[i], SUM[COL[i]]--;
24  }
25  void DLX_Resume_Repeat(int c) {
26      for (int i = U[c]; i != c; i = U[i])
27          L[R[i]] = i, R[L[i]] = i, SUM[COL[i]]++;
28  }
29  int Heuristics() {
30      memset(vis, true, sizeof(vis));
31      int c, i, j, cnt=0;
32      for(c=R[0]; c; c=R[c])
33          if(vis[c])
34              for(cnt++, vis[c] = false, i = D[c]; i != c; i = D[i])
35                  for(j = R[i]; j != i; j = R[j])
36                      vis[COL[j]] = false;
37      return cnt;
38  }
39  void DLX_Dfs(int n) {
40      if (Heuristics() + n >= ANS) return;
41      if (R[0] == 0) {
42          ANS = n;
43          for (int i = 0; i < n; i++)
44              USE[i] = _USE[i];
45          return ;
46      }
47      int i,now = INF,c;
48      for (i = R[0]; i; i = R[i])
49          if (now > SUM[i])
50              now = SUM[c = i];
51      for(i = D[c]; i != c; i = D[i]) {
52          DLX_Remove_Repeat(i);
53          for(int j = R[i]; j != i; j = R[j])
54              DLX_Remove_Repeat(j);
55          _USE[n] = ROW[i];
56          DLX_Dfs(n + 1);
57          for(int j = L[i]; j != i; j = L[j])
58              DLX_Resume_Repeat(j);
59          DLX_Resume_Repeat(i);
60      }
61  }
62  void solve() {
63      //ANS = m
64      DLX_Dfs(0);
65  }
66  };

```

7 String

7.1 Aho-Corasick automation

Don't forget running **BUILD**!

7.1.1 Static memory version

```

1 struct Trie {
2     int next[50][10], fail[50];
3     bool end[50];
4     int L, root;
5     int newNode() {
6         for (int i = 0; i < 10; i++)
7             next[L][i] = -1;
8         end[L] = false;
9         return L++;
10    }
11    void Init() {
12        L = 0;
13        root = newNode();
14    }
15    void Insert(char s[]) {
16        int now = root;
17        for (int i = 0; s[i] != 0; i++) {
18            if (next[now][s[i] - '0'] == -1)
19                next[now][s[i] - '0'] = newNode();
20            now = next[now][s[i] - '0'];
21        }
22        end[now] = true;
23    }
24    void Build() {
25        queue<int> Q;
26        for (int i = 0; i < 10; i++)
27            if (next[root][i] == -1)
28                next[root][i] = root;
29            else {
30                fail[next[root][i]] = root;
31                Q.push(next[root][i]);
32            }
33        while (!Q.empty()) {
34            int now = Q.front();
35            Q.pop();
36            end[now] |= end[fail[now]];
37            for (int i = 0; i < 10; i++)
38                if (next[now][i] == -1)
39                    next[now][i] = next[fail[now]][i];
40                else {
41                    fail[next[now][i]] = next[fail[now]][i];
42                    Q.push(next[now][i]);
43                }
44        }
45    }
46 };

```

7.1.2 Pointer version

```

1 const int CHAR=26;
2 const int TOTLEN=5000000;
3 const int MAXLEN=10000000;
4 struct Vertex {
5     Vertex *fail,*next[CHAR];

```



```

6   Vertex() {}
7   Vertex(bool flag) { //为什么要这样写?
8       fail=0;
9       memset(next,0,sizeof(next));
10  }
11 };
12 int size;
13 Vertex vertex[TOTLEN+1];
14 void init() {
15     vertex[0]=Vertex(0);
16     size=1;
17 }
18 void add(Vertex *pos,int cha) {
19     vertex[size]=Vertex(0);
20     pos->next[cha]=&vertex[size++];
21 }
22 void add(vector<int> s) {
23     int l=s.size();
24     Vertex *pos=&vertex[0];
25     for (int i=0; i<l; i++) {
26         if (pos->next[s[i]]==NULL)
27             add(pos,s[i]);
28         pos=pos->next[s[i]];
29     }
30 }
31 void bfs() {
32     queue<Vertex *> que;
33     Vertex *u=&vertex[0];
34     for (int i=0; i<CHAR; i++)
35         if (u->next[i]!=NULL) {
36             que.push(u->next[i]);
37             u->next[i]->fail=u;
38         } else
39             u->next[i]=u;
40     u->fail=NULL;
41     while (!que.empty()) {
42         u=que.front();
43         que.pop();
44         for (int i=0; i<CHAR; i++)
45             if (u->next[i]!=NULL) {
46                 que.push(u->next[i]);
47                 u->next[i]->fail=u->fail->next[i];
48             } else
49                 u->next[i]=u->fail->next[i];
50     }
51 }

```

7.2 KMP

Match the suffix of $A[\cdots i]$ and the prefix of B

```

1 //Self match
2 int j;
3 p[0] = j = -1;
4 for (int i = 1; i < lb; i++) {
5     while (j >= 0 && b[j + 1] != b[i]) j = p[j];
6     if (b[j + 1] == b[i]) j++;
7     p[i] = j;
8 }
9 //Match
10 j = -1;
11 for (int i = 0; i < la; i++) {
12     while (j >= 0 && b[j + 1] != a[i]) j = p[j];

```

```

13 |   if (b[j + 1] == a[i]) j ++;
14 |   KMP[i] = j + 1;
15 | }

```

7.3 E-KMP

Common prefix of $A[i \dots]$ and B

```

1 | //Self match
2 | int j = 0;
3 | while (j < lb && b[j] == b[j + 1])
4 |     j++;
5 | p[0] = lb, p[1] = j;
6 | int k = 1;
7 | for (int i = 2; i < lb; i++) {
8 |     int Len = k + p[k] - 1, L = p[i - k];
9 |     if (L < Len - i + 1)
10 |         p[i] = L;
11 |     else {
12 |         j = max(0, Len - i + 1);
13 |         while (i + j < lb && b[i + j] == b[j])
14 |             j++;
15 |         p[i] = j, k = i;
16 |     }
17 | }
18 | //Match
19 | j = 0;
20 | while (j < la && j < lb && a[j] == b[j])
21 |     j++;
22 | eKMP[0] = j;
23 | k = 0;
24 | for (int i = 1; i < la; i++) {
25 |     int Len = k + eKMP[k] - 1, L = p[i - k];
26 |     if (L < Len - i + 1)
27 |         eKMP[i] = L;
28 |     else {
29 |         j = max(0, Len - i + 1);
30 |         while (i + j < la && j < lb && a[i + j] == b[j])
31 |             j++;
32 |         eKMP[i] = j, k = i;
33 |     }
34 | }

```

7.4 Manacher

```

1 | const int maxn = 110000;
2 |
3 | char Ma[maxn*2];
4 | int Mp[maxn*2];
5 | void Manacher(char s[], int len) {
6 |     int l = 0;
7 |     Ma[l++] = '.';
8 |     Ma[l++] = ',';
9 |     for (int i = 0; i < len; i++) {
10 |         Ma[l++] = s[i];
11 |         Ma[l++] = ',';
12 |     }
13 |     Ma[l] = 0;
14 |     int pnow = 0, pid = 0;
15 |     for (int i = 1; i < l; i++) {
16 |         if (pnow > i)
17 |             Mp[i] = min(Mp[2*pid-i], pnow-i);

```

```

18     else
19         Mp[i] = 1;
20         for (; Ma[i-Mp[i]] == Ma[i+Mp[i]]; Mp[i]++);
21         if (i+Mp[i] > pnow) {
22             pnow = i+Mp[i];
23             pid = i;
24         }
25     }
26 }
27 /*
28 abaaba
29 . , a , b , a , a , b , a ,
30 0 1 2 1 4 1 2 7 2 1 4 1 2 1
31 */

```

7.5 Suffix array

```

1  const int maxn = 200010;
2  int wx[maxn],wy[maxn],*x,*y,wss[maxn],wv[maxn];
3
4  bool cmp(int *r,int n,int a,int b,int l) {
5      return a+l<n && b+l<n && r[a]==r[b]&&r[a+l]==r[b+l];
6  }
7  void da(int str[],int sa[],int rank[],int height[],int n,int m) {
8      int *s = str;
9      int *x=wx,*y=wy,*t,p;
10     int i,j;
11     for(i=0; i<m; i++)wss[i]=0;
12     for(i=0; i<n; i++)wss[x[i]=s[i]]++;
13     for(i=1; i<m; i++)wss[i]+=wss[i-1];
14     for(i=n-1; i>=0; i--)sa[--wss[x[i]]]=i;
15     for(j=1,p=1; p<n && j<n; j*=2,m=p) {
16         for(i=n-j,p=0; i<n; i++)y[p++]=i;
17         for(i=0; i<n; i++)if(sa[i]-j>=0)y[p++]=sa[i]-j;
18         for(i=0; i<n; i++)wv[i]=x[y[i]];
19         for(i=0; i<m; i++)wss[i]=0;
20         for(i=0; i<n; i++)wss[wv[i]]++;
21         for(i=1; i<m; i++)wss[i]+=wss[i-1];
22         for(i=n-1; i>=0; i--)sa[--wss[wv[i]]]=y[i];
23         for(t=x,x=y,y=t,p=1,i=1,x[sa[0]]=0; i<n; i++)
24             x[sa[i]]=cmp(y,n,sa[i-1],sa[i],j)?p-1:p++;
25     }
26     for(int i=0; i<n; i++) rank[sa[i]]=i;
27     for(int i=0,j=0,k=0; i<n; height[rank[i++]]=k)
28         if(rank[i]>0)
29             for(k?k--:0,j=sa[rank[i]-1];
30                 i+k < n && j+k < n && str[i+k]==str[j+k];
31                 k++);
32 }

```

7.5.1 Longest common prefix

```

1  int lcp(int x,int y) {
2      if (x > y) swap(x,y);
3      if (x == y)
4          return len-sa[x]; //NOTICE!
5      x++;
6      int k = lent[y-x+1];
7      return min(f[x][k],f[y-(1<<k)+1][k]);
8  }
9  //Interval

```

```

10 void getinterval(int pos,int comlen,int& pl,int& pr) {
11     int l,r,mid,cp;
12     l = 0;
13     r = pos;
14     while (l < r) {
15         mid = l+r>>1;
16         cp = lcp(mid,pos);
17         if (cp < comlen)
18             l = mid+1;
19         else
20             r = mid;
21     }
22     pl = l;
23     l = pos;
24     r = len-1;
25     while (l < r) {
26         mid = l+r+1>>1;
27         cp = lcp(pos,mid);
28         if (cp < comlen)
29             r = mid-1;
30         else
31             l = mid;
32     }
33     pr = l;
34 }

```

7.6 Smallest representation

```

1 int Gao(char a[],int len) {
2     int i = 0,j = 1,k = 0;
3     while (i < len && j < len && k < len) {
4         int cmp = a[(j+k)%len]-a[(i+k)%len];
5         if (cmp == 0)
6             k++;
7         else {
8             if (cmp > 0)
9                 j += k+1;
10            else
11                i += k+1;
12            if (i == j) j++;
13            k = 0;
14        }
15    }
16    return min(i,j);
17 }

```

7.7 Hash

```

1 typedef long long ll;
2 typedef unsigned long long ull;
3 const int MAXN = 1000000 + 100;
4 //ll gao[MAXN];
5 ll BL, BR, ML, MR;
6 ull psl[MAXN],psr[MAXN];
7 //call this before everything
8 void init(){
9     int maxx = 1e9;
10    srand(time(0));
11    BL = (ll)maxx + rand() % maxx;
12    BR = (ll)maxx + rand() % maxx;
13    ML = (ll)maxx + rand() % maxx;

```

```

14     MR = (ll)maxx + rand() % maxx;
15 }
16 //n is the max length you need
17 void Hash2(int n){
18     for(int i = 0; i <= n ; i ++){
19         psl[i] = (i == 0 ? 1 : psl[i - 1] * BL) % ML;
20     }
21     for(int i = 0 ; i <= n; i ++){
22         psr[i] = (i == 0 ? 1 : psr[i - 1] * BR) % MR;
23     }
24 }
25 struct _hash{
26     //read your string in here.
27     char str[MAXN];
28     ull hs[MAXN];
29     void build(){
30         int n = strlen(str);
31         ll l = 0, r = 0;
32         for(int i = 0 ; i < n ; i ++){
33             l = (l * BL + str[i]) % ML;
34             r = (r * BR + str[i]) % MR;
35             if(l < 0) l += ML;
36             if(r < 0) r += MR;
37             hs[i + 1] = (l << 32) | r;
38         }
39     }
40     //hash(str[b: e])
41     ll get(int b,int e){
42         ull el = (hs[e] >>32);
43         ull er = (hs[e] & 0xffffffffFULL);
44         ull bl = (hs[b] >> 32);
45         ull br = (hs[b] & 0xffffffffFULL);
46         ll l = el - bl * psl[e-b] % ML;
47         ll r = er - br *psr[e-b] % MR;
48         if(l < 0) l += ML;
49         if(r < 0) r += MR;
50         return (l << 32) | r;
51     }
52 }Hash;

```

8 Tool

8.1 日期函数

```

1 int days[12]={31,28,31,30,31,30,31,31,30,31,30,31};
2 struct date{
3     int year,month,day;
4 };
5 //是否闰年
6 inline bool isleap(int year)
7 {
8     return (year%4==0&&year%100!=0)||year%400==0;
9 }
10 //判合法性
11 inline bool islegal(date a){
12     if (a.month<0||a.month>12) return 0;
13     if (a.month==2)
14         return a.day>0&&a.day<=28+isleap(a.year);
15     return a.day>0&&a.day<=days[a.month-1];
16 }
17 //比较日期大小，正/负表示大于/小于，0表示相等
18 //如果用于sort等，请把-改成<

```

```

19 inline int datecmp(date a,date b){
20     if (a.year!=b.year)
21         return a.year-b.year;
22     if (a.month!=b.month)
23         return a.month-b.month;
24     return a.day-b.day;
25 }
26 //日期转天数偏移({0,1,1}第1天)
27 int date2int(date a){
28     int ret=a.year*365+(a.year-1)/4-(a.year-1)/100+(a.year-1)/400,i;
29     days[1]+=isleap(a.year);
30     for (i=0;i<a.month-1;ret+=days[i++]);
31     days[1]=28;
32     return ret+a.day;
33 }
34 //天数偏移转日期
35 date int2date(int a){
36     date ret;
37     ret.year=a/146097*400;
38     for (a%=146097;a>=365+isleap(ret.year);a-=365+isleap(ret.year),ret.year++);
39     days[1]+=isleap(ret.year);
40     for (ret.month=1;a>=days[ret.month-1];a-=days[ret.month-1],ret.month++);
41     days[1]=28;
42     ret.day=a+1;
43     return ret;
44 }

```

8.2 Bit compression

```

1 int bit[5];
2 inline int getbit26(int sta, int pos) {
3     return sta / bit[pos] % bit[1];
4 }
5 inline int setbit26(int sta, int pos, int val) {
6     return sta / bit[pos + 1] * bit[pos + 1] + val * bit[pos] + sta % bit[pos];
7 }
8 //bin
9 inline int getbit(int sta, int pos) {
10     return (sta >> pos) & 1;
11 }
12 inline int setbit(int sta, int pos, int val) {
13     return ((sta >> (pos + 1)) << (pos + 1)) | (val << pos) | (sta & ((1 << pos) - 1));
14 }

```

8.3 Random

之所以增加这一部分是由于 rand() 和 random_device 在 Win + MinGw 的环境下并不是一个安全的随机（返回的是固定的一个数列）
因此用以下代码将会使得你的随机代码更加靠谱。

```

1 std::mt19937 rng_engine{randutils::auto_seed_128{}.base()};
2 std::uniform_int_distribution<int> dist{1, 1000}; //1-1000 inclusive
3 int rand_integer = dist(rng_engine);

```

8.4 Hash map

```

1 struct hash_map {
2     int head[MOD];
3     struct hash_tables {

```

```

4     int key1, key2;
5     long long val;
6     int next;
7 } ele[ELE];
8 int N;
9 int getHash(int key1, int key2) {
10     return (key1 * 1000000 + key2) % MOD;
11 }
12 void init() {
13     memset(head, -1, sizeof(head));
14     N = 0;
15 }
16 void clear() {
17     for (int i = 0; i < N; i++)
18         head[getHash(ele[i].key1, ele[i].key2)] = -1;
19     N = 0;
20 }
21 int fint(int key1, int key2) {
22     for (int i = head[getHash(key1, key2)]; i != -1; i = ele[i].next) {
23         if (ele[i].key1 == key1 && ele[i].key2 == key2)
24             return i;
25     }
26     return -1;
27 }
28 void insert(int key1, int key2) {
29     int tmp = getHash(key1, key2);
30     ele[N].key1 = key1;
31     ele[N].key2 = key2;
32     ele[N].val = 0;
33     ele[N].next = head[tmp];
34     head[tmp] = N++;
35 }
36 long long get(int key1, int key2) {
37     int tmp = fint(key1, key2);
38     if (tmp == -1) {
39         insert(key1, key2);
40         return ele[N - 1].val;
41     } else
42         return ele[tmp].val;
43 }
44 void set(int key1, int key2, long long val) {
45     int tmp = fint(key1, key2);
46     if (tmp == -1) {
47         insert(key1, key2);
48         ele[N - 1].val = val;
49     } else
50         ele[tmp].val = val;
51 }
52 void add(int key1, int key2, long long val) {
53     int tmp = fint(key1, key2);
54     if (tmp == -1) {
55         insert(key1, key2);
56         ele[N - 1].val += val;
57     } else
58         ele[tmp].val += val;
59 }
60 };

```

8.5 120 bit integer

```

1 struct integer {
2     long long pa, pb;
3     integer() {}

```

```

4   integer(long long _pa, long long _pb) {
5       pa = _pa;
6       pb = _pb;
7   }
8   integer negate() {
9       if (pa == 0 && pb == 0)
10          return integer(pa, pb);
11       else if (pa == 0)
12          return integer(pa, -pb);
13       else
14          return integer(-pa, pb);
15   }
16   integer operator +(const integer& b) const {
17       integer ret = integer(pa + b.pa, pb + b.pb);
18       if (ret.pb >= MOD) {
19           ret.pa += 1;
20           ret.pb -= MOD;
21       }
22       return ret;
23   }
24   bool operator <(const integer& b) const {
25       if (pa == b.pa)
26          return pb < b.pb;
27       return pa < b.pa;
28   }
29 };

```

8.6 Bash script

```

1 while true; do
2     ./gen > input
3     ./sol < input > output.sol
4     ./bf < input > output.bf
5
6     diff output.sol output.bf
7     if [ $? -ne 0 ] ; then break; fi
8 done

```

8.7 Codeblocks settings

```
1 |gnome-terminal -t $TITLE -x
```

8.8 Bit operation

8.8.1 基本操作

注意括号

8.8.2 枚举长为 n 含 k 个 1 的 01 串

```

1 int n = 5, k = 3;
2 for (int s = (1 << k) - 1, u = 1 << n; s < u;) {
3     for (int i = 0; i < n; i++)
4         printf("%d", (((s >> (n - 1 - i)) & 1) == 1));
5     printf("\n");
6
7     int b = s & -s;
8     s = (s + b) | (((s ^ (s + b)) >> 2) / b);
9 }

```


功能	示例	位运算
返回 lsb 之后的 0 的个数	(1100010) \rightarrow 1D	__builtin_ctz(x)[x==0 时 UB]
统计二进制 1 的个数	(1100110 \rightarrow 4D)	__builtin_popcount(x)
取最后一个 1 的 pos+1(ffs)	(1000010 \rightarrow 2D)	__builtin_ffs(x)
取最后一个 1 的 mask(lsb)	(1000010 \rightarrow 10)	(x & (-x))
去掉最后一位	(101101 \rightarrow 10110)	x shr 1
在最后加一个 0	(101101 \rightarrow 1011010)	x shl 1
在最后加一个 1	(101101 \rightarrow 1011011)	x shl 1+1
把最后一位变成 1	(101100 \rightarrow 101101)	x or 1
把最后一位变成 0	(101101 \rightarrow 101100)	x or 1-1
最后一位取反	(101101 \rightarrow 101100)	x xor 1
把右数第 k 位变成 1	(101001 \rightarrow 101101, $k=3$)	x or (1 shl (k-1))
把右数第 k 位变成 0	(101101 \rightarrow 101001, $k=3$)	x and not (1 shl (k-1))
右数第 k 位取反	(101001 \rightarrow 101101, $k=3$)	x xor (1 shl (k-1))
取末三位	(1101101 \rightarrow 101)	x and 7
取末 k 位	(1101101 \rightarrow 1101, $k=5$)	x and (1 shl k-1)
取右数第 k 位	(1101101 \rightarrow 1, $k=4$)	x shr (k-1) and 1
把末 k 位变成 1	(101001 \rightarrow 101111, $k=4$)	x or (1 shl k-1)
末 k 位取反	(101001 \rightarrow 100110, $k=4$)	x xor (1 shl k-1)
把右边连续的 1 变成 0	(100101111 \rightarrow 100100000)	x and (x+1)
把右起第一个 0 变成 1	(100101111 \rightarrow 100111111)	x or (x+1)
把右边连续的 0 变成 1	(11011000 \rightarrow 11011111)	x or (x-1)
取右边连续的 1	(100101111 \rightarrow 1111)	(x xor (x+1)) shr 1
去掉右起第一个 1 的左边	(100101000 \rightarrow 1000)	x and (x xor (x-1))

8.9 vimrc

```

1 syntax on
2
3 set backspace=start,indent,eol
4 set showmode
5 set showcmd
6 set hlsearch
7 set nowrap
8 set smarttab
9 set autoindent
10 set tabstop=4
11 set softtabstop=4
12 set shiftwidth=4
13 set number
14 filetype indent on
15
16 set makeprg=g++\ '%:p'\ -o\ '%:p.mzry'\ -Wall\ -g
17 function! Gao()
18     exec "silent!w"
19     exec "silent!rm_f '%:p.mzry1992'"
20     exec "silent!make"
21     exec "cw"
22 endfunction
23 function! Run()
24     call Gao()
25     let execFile = expand("%:p").".mzry"
26     if filereadable(execFile)
27         exec "silent!gnome-terminal -t '%:p.mzry' --working-directory='%:p:h' -x /usr/
                bin/cb_console_runner '%:p.mzry'"
28     endif
29 endfunction

```

```

30
31 colorscheme slate
32 set gfn=Monospace\ 14
33
34 map <C-F9> :call Gao()<Enter>
35 imap <C-F9> <Esc>:call Gao()<Enter>
36 map <F9> :call Run()<Enter>
37 imap <F9> <Esc>:call Run()<Enter>
38
39 map <C-c> :s!^!//<Enter>:noh<Enter>
40 imap <C-c> <Esc>:s!^!//<Enter>:noh<Enter>
41 map <C-x> :s!//!<Enter>:noh<Enter>
42 imap <C-x> <Esc>:s!//!<Enter>:noh<Enter>

```

9 Appendix

9.1 Template by elfness

9.1.1 AC machine

```

1  #include<stdio>
2  #include<cstring>
3  #include<cstdlib>
4  #include<cmath>
5  #include<algorithm>
6  #include<iostream>
7  using namespace std;
8  typedef long long LL;
9  struct tree {
10     tree *ne[26],*fail;
11     int ct;
12 } tr[500100],VD,*root,*Q[500100];
13 int tn;
14 void init() {
15     tr[tn=0]=VD;
16     root=tr+(tn++);
17 }
18 char s[1000100];
19 void build() {
20     tree *p=root;
21     for(int i=0; s[i]; i++) {
22         if(p->ne[s[i]-'a']==NULL) {
23             tr[tn]=VD;
24             p->ne[s[i]-'a']=tr+(tn++);
25         }
26         p=p->ne[s[i]-'a'];
27     }
28     p->ct++;
29 }
30 void pre() {
31     int i,top,tail;
32     tree *p,*q;
33     top=0;
34     tail=0;
35     for(i=0; i<26; i++)
36         if(root->ne[i]!=NULL) {
37             Q[++tail]=root->ne[i];
38             root->ne[i]->fail=root;
39         } else root->ne[i]=root;
40     while(top<tail) {
41         p=Q[++top];
42         for(i=0; i<26; i++)

```

```

43     if(p->ne[i]!=NULL) {
44         q=p->ne[i];
45         Q[++tail]=q;
46         q->fail=p->fail->ne[i];
47         if(q->fail==NULL)q->fail=root;
48     } else p->ne[i]=p->fail->ne[i];
49 }
50 }
51 int doit() {
52     int ret=0;
53     tree *p=root,*q;
54     for(int i=0; s[i]; i++) {
55         p=p->ne[s[i]-'a'];
56         q=p;
57         while(root!=q&&q->ct!=-1) {
58             ret+=q->ct;
59             q->ct=-1;
60             q=q->fail;
61         }
62     }
63     return ret;
64 }
65 int i,n,_;
66 int main() {
67     for(i=0; i<26; i++)VD.ne[i]=NULL;
68     VD.ct=0;
69     scanf("%d",&_);
70     while(____) {
71         scanf("%d",&n);
72         init();
73         for(i=0; i<n; i++) {
74             scanf("%s",s);
75             build();
76         }
77         pre();
78         scanf("%s",s);
79         printf("%d\n",doit());
80     }
81 }

```

9.1.2 E-KMP

```

1  #include<cstdio>
2  #include<cstring>
3  #include<cstdlib>
4  #include<cmath>
5  #include<algorithm>
6  #include<iostream>
7  using namespace std;
8  typedef long long LL;
9  void e_kmp(char *s,char *t,int *has,int *e_has) {
10     int sp,p,mx,tn;
11     for(sp=p=mx=0; s[p]>0; p++) {
12         if(mx==p || p+e_has[p-sp]>=mx ) {
13             for(tn=mx-p; s[mx]==t[tn]; tn++)mx++;
14             has[sp=p]=mx-p;
15             if(mx==p)sp=++mx;
16         } else has[p]=e_has[p-sp];
17     }
18 }
19 const int V=1001000;
20 char t[V],s[V];
21 int e_has[V],has[V],tn;

```

```

22 int main() {
23     scanf("%s%s",s,t);
24     tn=strlen(t);
25     t[tn]=-1;
26     e_has[0] = tn;
27     e_kmp(t+1,t,e_has+1,e_has);
28     e_kmp(s,t,has,e_has);
29 }

```

9.1.3 KM (list)

```

1  #include<cstdio>
2  #include<cstring>
3  #include<cstdlib>
4  #include<cmath>
5  #include<algorithm>
6  using namespace std;
7  const int V=1200;
8  const int En=21000;
9  const int oo=10000000000;
10 struct Edge {
11     int num,ne,w;
12 } e[En];
13 int p[V],K;
14 void add(int x,int y,int z) {
15     e[K].num=y;
16     e[K].w=z;
17     e[K].ne=p[x];
18     p[x]=K++;
19 }
20 bool sx[V],sy[V];
21 int lx[V],ly[V],mat[V];
22 bool path(int u) {
23     sx[u]=true;
24     for(int i=p[u]; i!=-1; i=e[i].ne) {
25         int v=e[i].num;
26         if(!sy[v]&&lx[u]+ly[v]==e[i].w) {
27             sy[v]=true;
28             if(mat[v]==-1||path(mat[v])) {
29                 mat[v]=u;
30                 return true;
31             }
32         }
33     }
34     return false;
35 }
36 int N;
37 int KM() {
38     int i,j;
39     for(i=0; i<N; i++) {
40         lx[i]=-oo;
41         for(j=p[i]; j!=-1; j=e[j].ne)
42             lx[i]=max(lx[i],e[j].w);
43     }
44     for(i=0; i<N; i++)ly[i]=0,mat[i]=-1;
45     for(int u=0; u<N; u++)
46         while(1) {
47             for(i=0; i<N; i++)sx[i]=0,sy[i]=0;
48             if(path(u))break;
49             int dx=oo;
50             for(i=0; i<N; i++)if(sx[i])
51                 for(j=p[i]; j!=-1; j=e[j].ne)
52                     if(!sy[e[j].num])

```

```

53         dx=min(dx,lx[i]+ly[e[j].num]-e[j].w);
54         if(dx==oo) return -1;
55         for(i=0; i<N; i++) if(sx[i]) lx[i]-=dx;
56         for(i=0; i<N; i++) if(sy[i]) ly[i]+=dx;
57     }
58     int ret=0;
59     for(i=0; i<N; i++) ret+=lx[i]+ly[i];
60     return -ret;
61 }
62 int _,ca,n,m,i,x,y,z,te;
63 int main() {
64     scanf("%d",&_);
65     ca=0;
66     while(_--) {
67         ca++;
68         scanf("%d%d",&n,&m);
69         N=n;
70         for(i=0; i<n; i++) p[i]=-1;
71         K=0;
72         for(i=0; i<m; i++) {
73             scanf("%d%d%d",&x,&y,&z);
74             x--;
75             y--;
76             add(x,y,-z);
77             add(y,x,-z);
78         }
79         te=KM();
80         printf("Case_%d: ",ca);
81         if(te==-1) puts("NO");
82         else printf("%d\n",te);
83     }
84 }

```

9.1.4 Nearest point pair

```

1  /*
2   * nearestPointPair.cpp
3   *
4   * Created on: 2011-10-10
5   * Author: Fish
6   */
7
8  #include <cstdio>
9  #include <cstring>
10 #include <cstdlib>
11 #include <cmath>
12 #include <algorithm>
13
14 using namespace std;
15
16 const int MaxN = 120000;
17 const int Log = 20;
18
19 struct Point {
20     double x, y;
21     Point() {
22     }
23     Point(double x, double y) :
24         x(x), y(y) {
25     }
26     Point operator-(const Point& p) const {
27         return Point(x - p.x, y - p.y);
28     }

```

```

29  double norm() const {
30      return hypot(x, y);
31  }
32  void init() {
33      scanf("%lf%lf", &x, &y);
34  }
35 } p[MaxN];
36 int x[MaxN], y[Log][MaxN], tmp[MaxN], n;
37 bool vst[MaxN];
38
39 bool comp_x(const int& i, const int& j) {
40     return p[i].x < p[j].x;
41 }
42
43 bool comp_y(const int& i, const int& j) {
44     return p[i].y < p[j].y;
45 }
46
47 double dfs(int k, int l, int r) {
48     double ret = 1e100;
49     if (r - l <= 2) {
50         for (int i = l; i < r; i++)
51             for (int j = i + 1; j <= r; j++)
52                 ret = min(ret, (p[x[i]] - p[x[j]]).norm());
53     }
54     return ret;
55
56     int mid = (l + r) >> 1;
57     int lp = l, rp = mid + 1;
58     for (int i = l; i <= r; i++)
59         vst[x[i]] = i <= mid;
60     for (int i = l; i <= r; i++)
61         if (vst[y[k][i]])
62             y[k + 1][lp++] = y[k][i];
63         else
64             y[k + 1][rp++] = y[k][i];
65     double lhs = dfs(k + 1, l, mid);
66     double rhs = dfs(k + 1, mid + 1, r);
67     double mx = (p[x[mid + 1]].x + p[x[mid]].x) / 2.0;
68     ret = min(lhs, rhs);
69
70     lp = 0;
71     for (int i = l; i <= r; i++)
72         if (fabs(mx - p[y[k][i]].x) < ret)
73             tmp[lp++] = y[k][i];
74
75     for (int i = 0; i < lp; i++)
76         for (int j = 1; j < 8 && i + j < lp && (p[tmp[i + j]].y - p[tmp[i]].y) < ret; j++)
77             ret = min(ret, (p[tmp[i]] - p[tmp[i + j]]).norm());
78
79     return ret;
80 }
81
82 int main() {
83 #ifdef __FISH__
84     freopen("data.in", "r", stdin);
85     freopen("nlogn.out", "w", stdout);
86 #endif
87     while (scanf("%d", &n) == 1 && n) {
88         for (int i = 0; i < n; i++) {
89             p[i].init();
90             x[i] = y[0][i] = i;
91         }

```

```

92     sort(x, x + n, comp_x);
93     sort(y[0], y[0] + n, comp_y);
94     printf("%.2f\n", dfs(0, 0, n - 1) / 2.0);
95     // printf("%.6f\n", dfs(0, 0, n - 1));
96 }
97
98 return 0;
99 }

```

9.1.5 SA

```

1  #include<cstdio>
2  #include<cstring>
3  #include<cstdlib>
4  #include<cmath>
5  #include<algorithm>
6  #include<iostream>
7  #include<vector>
8  #include<string>
9  using namespace std;
10 typedef long long LL;
11 const int N=100100;
12 char s[N]; /// 长度+1, 对于非字符串, 加一个小于最小值的元素,
13 int sa[N]; /// 倍增算法, 结果 下标 1~n, 第 i 大的是 sa[i]
14 int rk[N]; /// 第 i 位开始的后缀, 的排名为 rk[i]
15 int wa[N],wb[N],wv[N],rmq[20][N];
16 int sn,to[N];
17 bool cmp(int *y,int a,int b,int L) {
18     return y[a]==y[b]&&y[a+L]==y[b+L];
19 }
20 void da(char *s,int *sa,int len,int dn) {
21     int i,j,p;
22     int *x,*y,*t;
23     x=wa;
24     y=wb;
25     for(i=0; i<dn; i++)rk[i]= 0;
26     for(i=0; i<len; i++)rk[x[i]=s[i]]++;
27     for(i=0; i<dn; i++)rk[i+1]+=rk[i];
28     for(i=len-1; i>=0; i--)sa[--rk[x[i]]]=i;
29     for(j=1,p=1; p<len; j*=2,dn=p) {
30         for(p=0; p<j; p++)y[p]=len-j+p;
31         for(i=0; i<len; i++)if(sa[i]>=j)y[p++]=sa[i]-j;
32         for(i=0; i<len; i++)wv[i]=x[y[i]];
33         for(i=0; i<dn; i++)rk[i]=0;
34         for(i=0; i<len; i++)rk[wv[i]]++;
35         for(i=0; i<dn; i++)rk[i+1]+=rk[i];
36         for(i=len-1; i>=0; i--)sa[--rk[wv[i]]]=y[i];
37         swap(x,y);
38         x[sa[0]]=0;
39         for(p=i=1; i<len; i++) {
40             p+=!cmp(y,sa[i],sa[i-1],j);
41             x[sa[i]]=p-1;
42         }
43     }
44 }
45 void find_height(char *s,int *sa,int len) {
46     int *h=rmq[0];
47     int i,j,k=0;
48     for(i=1; i<=len; i++)
49         rk[sa[i]] = i;
50     for(i=0; i<len; i++) {
51         if(k>0)k--;
52         j=sa[rk[i]-1];

```

```

53     while(s[i+k]==s[j+k])k++;
54     h[rk[i]]=k;
55 }
56 }
57 void RMQ(int n) {
58     int i,j;
59     int rn=(int)floor(log(n*2.0)/log(2.0));
60     for(i=1; i<rn; i++)
61         for(j=0; j<n+2-(1<<(i-1)); j++)
62             rmq[i][j]=min(rmq[i-1][j],rmq[i-1][j+(1<<(i-1))]);
63 }
64 int askRMQ(int a,int b) { /// [a,b] 闭区间
65     int rq=to[b-a];
66     return min(rmq[rq][a],rmq[rq][b+1-(1<<rq)]);
67 }
68 void PT(char *s,int *sa) {
69     int i,sn;
70     sn=strlen(s);
71     for(i=0; i<sn; i++)
72         puts(s+sa[i+1]);
73     puts("");
74     for(i=0; i<sn; i++)
75         printf("rank_%d=%d\n",i,rk[i]);
76 }
77 int lcp(int a,int b,int len) {
78     if(a==b)
79         return len-a;
80     a=rk[a];
81     b=rk[b];
82     if(a>b)swap(a,b);
83     return askRMQ(a+1,b);
84 }
85 void pre_log() {
86     int i;
87     to[0]=to[1]=0;
88     for(i=1; i*2<N; i++)
89         to[i*2]=to[i*2+1]=to[i]+1;
90 }
91 int main() {
92     int T,_=0;
93     pre_log();
94     while(~scanf("%s",s)) {
95         sn=strlen(s);
96         da(s,sa,sn+1,128);
97         find_height(s,sa,sn);
98         RMQ(sn);
99         PT(s,sa);
100         scanf("%d",&T);
101         while(T--) {
102             int a,b;
103             scanf("%d%d",&a,&b);
104             a--,b--;/// 求原串的 a b 开始的后缀的公共前缀
105             printf("lcp_%d=%d\n",lcp(a,b,sn));
106         }
107     }
108     return 0;
109 }

```

9.1.6 SAP

```

1 #include<stdio>
2 #include<cstring>
3 #include<stdlib>

```



```

4  #include<cmath>
5  #include<algorithm>
6  using namespace std;
7  const int V=220;
8  const int En=200000;
9  const int oo=0x3f3f3f3f;
10 struct Edge {
11     int num,ne,c;
12 } e[En];
13 int d[V],p[V],pre[V],low[V];
14 int gap[V],cur[V];
15 int N,K,st,ed;
16 void add(int x,int y,int c) {
17     e[K].num=y;
18     e[K].c=c;
19     e[K].ne=p[x];
20     p[x]=K++;
21     e[K].num=x;
22     e[K].c=0;
23     e[K].ne=p[y];
24     p[y]=K++;
25 }
26 int sap() {
27     int ret=0;
28     bool fail;
29     for(int i=0; i<=N; i++) {
30         low[i]=gap[i]=d[i]=0;
31         cur[i]=p[i];
32     }
33     low[st]=oo;
34     gap[0]=N;
35     int u=st;
36     while(d[st]<N) {
37         fail=true;
38         for(int i=cur[u]; i!=-1; i=e[i].ne) {
39             int v=e[i].num;
40             cur[u]=i;
41             if(e[i].c&&d[u]==d[v]+1) {
42                 pre[v]=i;
43                 low[v]=min(low[u],e[i].c);
44                 u=v;
45                 if(u==ed) {
46                     do {
47                         e[pre[u]].c-=low[ed];
48                         e[pre[u]^1].c+=low[ed];
49                         u=e[pre[u]^1].num;
50                     } while(u!=st);
51                     ret+=low[ed];
52                 }
53                 fail=false;
54                 break;
55             }
56         }
57         if(fail) {
58             gap[d[u]]--;
59             if(!gap[d[u]])return ret;
60             d[u]=N;
61             for(int i=p[u]; i!=-1; i=e[i].ne)
62                 if(e[i].c)d[u]=min(d[u],d[e[i].num]+1);
63             gap[d[u]]++;
64             cur[u]=p[u];
65             if(u!=st)u=e[pre[u]^1].num;
66         }
67     }

```

```

68     return ret;
69 }

```

9.1.7 一般图最大匹配

```

1  #include <stdio.h>
2  #include <string.h>
3  #include <algorithm>
4  #include <vector>
5  #define maxn 300
6  #define maxm 90010
7
8  using namespace std;
9
10 int match[maxn];           //标记是否匹配
11 int st[maxn],aim[maxm],nxt[maxm],ln; //边表
12 int q[maxn];               //bfs队列
13 int level[maxn];           //离根深度的奇偶性
14 vector<int> ar[maxn];       //存每个点到根的路径
15 vector<int> a;              //找到的一条增广路
16 int n;
17 void init() {
18     for(int i=0; i<n; i++)st[i]=-1;
19     ln=0;
20 }
21 void in_edge(int x,int y) {
22     aim[ln]=y;
23     nxt[ln]=st[x];
24     st[x]=ln++;
25 }
26 int lca(int p,int q) {      //求p和q的最近公共祖先
27     int ret=0;
28     while (ret<ar[p].size() && ret<ar[q].size() && ar[p][ret]==ar[q][ret]) ret++;
29     return ret-1;
30 }
31 int FindAlterRoad(int sp) {
32     int qn=1;
33     memset(level,-1,sizeof(level));
34     level[q[0]=sp]=1;
35     ar[sp].clear();
36     ar[sp].push_back(sp);
37     for (int p=0; p<qn; p++) {
38         int x=q[p];
39         for (int i=st[x]; i!=-1; i=nxt[i]) {
40             int u=aim[i];
41             if (match[u]==u) continue;
42             if (level[u]==-1) { //u是未访问的点
43                 if (match[u]==-1) { //u是未匹配的,找到增广路
44                     a=ar[x];
45                     a.push_back(u);
46                     return 1;
47                 } else { //u是已匹配的点
48                     int v=match[u];
49                     if (level[v]!=-1) continue;
50                     ar[v]=ar[x];
51                     ar[v].push_back(u);
52                     ar[v].push_back(v);
53                     level[u]=0;
54                     level[v]=1;
55                     q[qn++]=v;
56                 }
57             } else if (level[u]==1) { //u和x同为偶点.形成花
58                 int root=lca(u,x);

```

```

59     vector<int> tmp=ar[x];
60     for (int i=ar[u].size()-1; i>root; i--) {
61         int y=ar[u][i];
62         tmp.push_back(y);
63         if (level[y]==0) {
64             level[y]=1;
65             ar[y]=tmp;
66             level[y]=1;
67             q[qn++]=y;
68         }
69     }
70     tmp=ar[u];
71     for (int i=ar[x].size()-1; i>root; i--) {
72         int y=ar[x][i];
73         tmp.push_back(y);
74         if (level[y]==0) {
75             level[y]=1;
76             ar[y]=tmp;
77             level[y]=1;
78             q[qn++]=y;
79         }
80     }
81 }
82 }
83 }
84 return 0;
85 }
86 int MaximumMatch() {
87     int ret=0; //最大匹配数
88     memset(match,-1,sizeof(match));
89     for (int i=0; i<n; i++)
90         if (match[i]==-1)
91             if (FindAlterRoad(i)) {
92                 for (int i=0; i<a.size(); i+=2) {
93                     int u=a[i],v=a[i+1];
94                     match[u]=v;
95                     match[v]=u;
96                 }
97                 ret++;
98             } else match[i]=i;
99     return ret;
100 }

```

9.1.8 上下界最大流

```

1  /*
2  Author: elfness@UESTC
3  */
4  #include<cstdio>
5  #include<cstring>
6  #include<cstdlib>
7  #include<cmath>
8  #include<algorithm>
9  #include<iostream>
10 #include<vector>
11 #include<string>
12 using namespace std;
13 typedef long long LL;
14 const int V=1500;
15 const int En=900000;
16 const int inf=0x3f3f3f3f;
17 struct Edge {
18     int num,ne;

```

```

19     int c;
20 } e[En];
21 int p[V],K;
22 void add(int x,int y,int c) {
23     e[K].num=y;
24     e[K].c=c;
25     e[K].ne=p[x];
26     p[x]=K++;
27     e[K].num=x;
28     e[K].c=0;
29     e[K].ne=p[y];
30     p[y]=K++;
31 }
32 int d[V],pre[V],pree[V],gap[V],cur[V];
33 int N,st,ed;
34 int low[V];
35 int sap() {
36     int ret=0;
37     bool fail;
38     for(int i=0; i<=N; i++) {
39         d[i]=0;
40         gap[i]=0;
41         cur[i]=p[i];
42         low[i]=0;
43     }
44     low[st]=inf;
45     gap[0]=N;
46     int u=st;
47     while(d[st]<N) {
48         fail=true;
49         for(int i=cur[u]; i!=-1; i=e[i].ne) {
50             int v=e[i].num;
51             cur[u]=i;
52             if(e[i].c&& d[u]==d[v]+1) {
53                 pre[v]=u;
54                 pree[v]=i;
55                 low[v]=min(low[u],e[i].c);
56                 u=v;
57                 if(u==ed) {
58                     do {
59                         e[pree[u]].c-=low[ed];
60                         e[pree[u]^1].c+=low[ed];
61                         u=pree[u];
62                     } while(u!=st);
63                     ret+=low[ed];
64                 }
65                 fail=false;
66                 break;
67             }
68         }
69         if(fail) {
70             gap[d[u]]--;
71             if(!gap[d[u]])return ret;
72             d[u]=N;
73             for(int i=p[u]; i!=-1; i=e[i].ne)
74                 if(e[i].c)d[u]=min(d[u],d[e[i].num]+1);
75             gap[d[u]]++;
76             cur[u]=p[u];
77             if(u!=st)u=pree[u];
78         }
79     }
80     return ret;
81 }
82 int n,m,s,t;

```

```

83 struct Elf {
84     int u,v,lo,up;
85 } b[12000];
86 int lb[12000];
87 int doit() {
88     int i;
89     N=n+2;
90     st=n;
91     ed=n+1;
92     for(i=0; i<N; i++)p[i]=-1;
93     K=0;
94     for(i=0; i<n; i++)lb[i]=0;
95     for(i=0; i<m; i++) {
96         lb[b[i].u]-=b[i].lo;
97         lb[b[i].v]+=b[i].lo;
98         add(b[i].u,b[i].v,b[i].up-b[i].lo);
99     }
100    for(i=0; i<n; i++) {
101        if(lb[i]>0)add(st,i,lb[i]);
102        else add(i,ed,-lb[i]);
103    }
104    add(t,s,inf);
105    int te=sap();
106    for(i=p[st]; i!=-1; i=e[i].ne)
107        if(e[i].c!=0)return -1;
108    st=s;
109    ed=t;
110    te=sap();
111    return te;
112 }

```

9.1.9 上下界最小流

```

1  #include<cstdio>
2  #include<cstdlib>
3  #include<cstring>
4  #include<cmath>
5  #include<algorithm>
6  using namespace std;
7  const int V=600;
8  const int En=50000;
9  const int oo=0x3f3f3f3f;
10 struct Edge {
11     int num,ne,c;
12 } e[En];
13 int p[V],K;
14 void add(int x,int y,int c) {
15     e[K].num=y;
16     e[K].c=c;
17     e[K].ne=p[x];
18     p[x]=K++;
19     e[K].num=x;
20     e[K].c=0;
21     e[K].ne=p[y];
22     p[y]=K++;
23 }
24 int d[V],cur[V],low[V],pre[V],gap[V],pree[V];
25 int st,ed,N;
26 int sap() {
27     int ret=0;
28     bool fail;
29     memset(gap,0,sizeof(gap));
30     memset(low,0,sizeof(low));

```

```

31  memset(d,0,sizeof(d));
32  for(int i=0; i<N; i++)cur[i]=p[i];
33  gap[0]=N;
34  low[st]=oo;
35  int u=st;
36  while(d[st]<N) {
37      fail=true;
38      for(int i=cur[u]; i!=-1; i=e[i].ne) {
39          int v=e[i].num;
40          cur[u]=i;
41          if(e[i].c&& d[u]==d[v]+1) {
42              pre[v]=u;
43              pree[v]=i;
44              low[v]=min(low[u],e[i].c);
45              u=v;
46              if(u==ed) {
47                  do {
48                      e[pree[u]].c-=low[ed];
49                      e[pree[u]^1].c+=low[ed];
50                      u=pree[u];
51                  } while(u!=st);
52                  ret+=low[ed];
53              }
54              fail=false;
55              break;
56          }
57      }
58      if(fail) {
59          gap[d[u]]--;
60          if(!gap[d[u]])return ret;
61          d[u]=N;
62          for(int i=p[u]; i!=-1; i=e[i].ne)
63              if(e[i].c)d[u]=min(d[u],d[e[i].num]+1);
64          gap[d[u]]++;
65          cur[u]=p[u];
66          if(u!=st)u=pree[u];
67      }
68  }
69  return ret;
70 }
71 struct ELF {
72     int u,v,lo;
73 } b[En];
74 int n,m,lb[V],ts,tt;
75 void solve() {
76     N=n+4;
77     ts=0;
78     tt=n+1;
79     st=n+2;
80     ed=n+3;
81     memset(lb,0,sizeof(lb));
82     int i,u,v;
83     for(i=0; i<N; i++)p[i]=-1;
84     K=0;
85     for(i=0; i<m; i++) {
86         u=b[i].u;
87         v=b[i].v;
88         lb[v]+=b[i].lo;
89         lb[u]-=b[i].lo;
90         add(u,v,oo-b[i].lo);
91     }
92     for(i=1; i<=n; i++) {
93         add(ts,i,oo);
94         add(i,tt,oo);

```

```

95     }
96     for(i=0; i<n+2; i++) {
97         if(lb[i]>0)add(st,i,lb[i]);
98         else add(i,ed,-lb[i]);
99     }
100     int ans=sap();
101     add(tt,ts,oo);
102     printf("%d\n",sap());
103 }
104 int _,ca,i;
105 int main() {
106     scanf("%d",&_);
107     ca=0;
108     while(_--) {
109         ca++;
110         scanf("%d%d",&n,&m);
111         for(i=0; i<m; i++) {
112             scanf("%d%d%d",&b[i].u,&b[i].v,&b[i].lo);
113         }
114         printf("Case_#%d:_",ca);
115         solve();
116     }
117 }

```

9.1.10 全局最小割

```

1  using namespace std;
2  #define inf 1000000000
3  bool visit[502],com[502];
4  int map[502][502],W[502],s,t;
5  int maxadj(int N,int V) {
6      int CUT;
7      memset(visit,0,sizeof(visit));
8      memset(W,0,sizeof(W));
9      for(int i=0; i<N; i++) {
10         int Num=0,Max=-inf;
11         for(int j=0; j<V; j++)
12             if(!com[j]&&!visit[j]&&W[j]>Max) {
13                 Max=W[j];
14                 Num=j;
15             }
16         visit[Num]=true;
17         s=t;
18         t=Num;
19         CUT=W[t];
20         for(int j=0; j<V; j++)
21             if(!com[j]&&!visit[j])W[j]+=map[Num][j];
22     }
23     return CUT;
24 }
25 int stoer(int V) {
26     int Mincut=inf;
27     int N=V;
28     memset(com,0,sizeof(com));
29     for(int i=0; i<V-1; i++) {
30         int Cut;
31         s=0,t=0;
32         Cut=maxadj(N,V);
33         N--;
34         if(Cut<Mincut)Mincut=Cut;
35         com[t]=true;
36         for(int j=0; j<V; j++)
37             if(!com[j]) {

```

```

38     map[j][s]+=map[j][t];
39     map[s][j]+=map[t][j];
40 }
41 }
42 return Mincut;
43 }

```

9.1.11 最小树型图

```

1  #include<cstdio>
2  #include<cstring>
3  #include<cstdlib>
4  #include<cmath>
5  #include<algorithm>
6  using namespace std;
7  const int V=1200;
8  const int En=2100000;
9  struct Elf {
10     int u,v,len;
11 } b[En];
12 const int oo=1000000000;
13 int ret;
14 int N,M,Root;//点数，边数，根，默认从0开始
15 int id[V],pre[V],cnt,vis[V];
16 int in[V];
17 bool TreeMST() {
18     ret=0;
19     int i,u,v;
20     while(1) {
21         for(i=0; i<N; i++)
22             in[i]=oo;
23         memset(pre,-1,sizeof(pre));
24         for(i=0; i<M; i++) {
25             u=b[i].u;
26             v=b[i].v;
27             if(b[i].len<in[v]&&u!=v) {
28                 pre[v]=u;
29                 in[v]=b[i].len;
30             }
31         }
32         for(i=0; i<N; i++) {
33             if(i==Root)continue;
34             if(pre[i]==-1)return false;
35         }
36         in[Root]=0;
37         cnt=0;
38         memset(id,-1,sizeof(id));
39         memset(vis,-1,sizeof(vis));
40         for(i=0; i<N; i++) {
41             ret+=in[i];
42             v=i;
43             while(vis[v]!=i&&id[v]==-1&&v!=Root) {
44                 vis[v]=i;
45                 v=pre[v];
46             }
47             if(v!=Root&&id[v]==-1) {
48                 for(u=pre[v]; u!=v; u=pre[u])
49                     id[u]=cnt;
50                 id[v]=cnt++;
51             }
52         }
53         if(cnt==0)return true;
54         for(i=0; i<N; i++)

```



```
55     if(id[i]==-1)id[i]=cnt++;
56     for(i=0; i<M; i++) {
57         v=b[i].v;
58         b[i].u=id[b[i].u];
59         b[i].v=id[b[i].v];
60         if(b[i].u!=b[i].v)
61             b[i].len-=in[v];
62     }
63     N=cnt;
64     Root=id[Root];
65 }
66 return true;
67 }
```