ACM TEMPLATE



UESTC_Dagon

Last build at July 9, 2014

Contents

1	Data	astructure 4								
	1.1	KD tree								
	1.2	Binary indexed tree								
	1.3	Splay								
	1.4	Dynamic tree								
	1.5	Partition tree								
	_									
2	•	namic programming 9								
	2.1	RMQ								
	2.2	2D-LIS								
3	Geometry									
	3.1	2D								
		3.1.1 Point								
		3.1.2 Line								
		3.1.3 Functions								
		3.1.4 Half plane intersection								
		3.1.5 Convex hull								
		3.1.6 Intersections of line and polygon								
	3.2	3D								
	J	3.2.1 Point								
		3.2.2 Functions								
		3.2.3 Convex hull								
	3.3	Circle								
	0.0	3.3.1 Functions								
		3.3.2 Union								
		3.3.3 Area of intersection part with polygon								
	3.4	Matrix								
	5.4	3.4.1 基本矩阵								
		9.4.1 全年2件								
4	Gra	ph 20								
	4.1	Sap								
	4.2	Minimal cost maximal flow								
	4.3	Johnson Minimal cost flow								
	4.4	Bi-connect								
	4.5	Cut and bridge								
	4.6	Stoer-Wagner								
	4.7	Euler path								
	4.8	Strongly connected component								
	4.9	Match								
		4.9.1 Bipartite graph								
		4.9.2 Edmonds								
		4.9.3 KM								
	4.10	Clique								
		Spanning tree								
	1.11	4.11.1 Count the number of spanning tree								
		4.11.2 Spanning tree on directed graph								
	4 19	Kth shortest path								
		Stable marriage problem								
	4.10	Diamond marriage problem								

				U	\mathbf{E}	\mathbf{S}'	\mathbf{T}	C	<u>;_</u>	_I	Эa	gon
9	App									64		
	9.1	Templa	ate by elfness									64
		9.1.1	AC machine									64
		9.1.2	E-KMP									65
		9.1.3	KM (list)									66
		9.1.4	Nearest point pair									67
		9.1.5	SA									69
		9.1.6	SAP									71
		9.1.7	一般图最大匹配									72
		9.1.8	上下界最大流									73
		9.1.9	上下界最小流									75
		9.1.10	全局最小割									77

1 Datastructure

1.1 KD tree

1 //Node

2 | struct Node {

```
|bool Div[MaxN];
   void BuildKD(int deep,int l, int r, Point p[]) {
     if (l > r) return;
 3
     int mid = l + r >> 1;
 4
 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;
     maxX = max_element(p + l, p + r + 1, cmpX) -> x;
 8
     maxY = max_element(p + l, p + r + 1, cmpY) -> y;
 9
     Div[mid] = (maxX - minX >= maxY - minY);
10
     nth_element(p + l, p + mid, p + r + 1, Div[mid] ? cmpX : cmpY);
11
     BuildKD(l, mid -1, p);
12
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;
     int mid = l + r >> 1;
18
     long long dist = dist2(a, p[mid]);
19
     if (dist > 0)//NOTICE
20
        res = min(res, dist);
21
22
     long long d = Div[mid] ? (a.x - p[mid].x) : (a.y - p[mid].y);
23
     int l1, l2, r1, r2;
     l1 = l, l2 = mid + 1;
24
     r1 = mid - 1, r2 = r;
25
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
   int read(int k) {
 2
     int sum = 0;
     for (; k; k^=k&-k) sum+=tree[k];
 3
 4
      return sum;
 5
 6
   void update(int k, int v) {
 7
     for (; k<=MaxN; k+=k&-k) tree[k]+=v;</pre>
 8
9
   int find_Kth(int k) {
10
      int idx = 0;
     for(int i=20; i>=0; i—) {
11
12
        idx |= 1 << i;
        if(idx <= MaxN && tree[idx] < k)</pre>
13
14
          k -= tree[idx];
        else idx ^= 1 << i;
15
16
17
      return idx + 1;
18
   1.3 Splay
```

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

```
67
         if (f == nil)
 68
            root = x;
 69
       void Select(int k, Node *f) {
 70
         Node *x = root;
 71
         Pushdown(x);
 72
 73
         int tmp;
 74
         while ((tmp = x->c[0]->size) != k) {
 75
            if (k < tmp) x = x -> c[0];
 76
 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);
         x\rightarrow c[0] = Make\_tree(a, l, mid - 1, x);
 89
 90
         x\rightarrow c[1] = Make\_tree(a, mid + 1, r, x);
 91
         Pushup(x);
 92
         return x;
 93
       void Insert(int pos, int a[], int n) {
 94
         Select(pos, nil), Select(pos + 1, root);
 95
          root \rightarrow c[1] \rightarrow c[0] = Make\_tree(a, 0, n - 1, root \rightarrow c[1]);
 96
 97
         Splay(root->c[1]->c[0], nil);
 98
 99
       void Insert(int v) {
100
         Node *x = root, *y = nil;
101
          //Need pushdown
         while (x != nil) {
102
            y = x;
103
104
            y->size++;
105
            x = x - c[v > = x - key];
106
107
         y \rightarrow c[v >= y \rightarrow key] = x = newNode(v, y);
108
         Splay(x, nil);
109
       }
       void Remove(int l, int r) {
110
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
     struct SplayTree {
  1
  2
       void Pushup(Node *x) {
  3
         if (x == nil)
                            return;
  4
         Pushdown(x);
  5
         Pushdown(x \rightarrow c[0]);
  6
         Pushdown(x\rightarrowc[1]);
  7
         x\rightarrow size = x\rightarrow c[0]\rightarrow size + x\rightarrow c[1]\rightarrow size + 1;
  8
  9
       void Pushdown(Node *x) {
 10
         if (x == nil)
                            return;
```

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

```
75
        Access(u);
76
        Splay(u);
77
        u->rev = true;
78
        u \rightarrow p = v;
79
      }
      void ChangeRoot(Node *u) {
80
81
        Access(u)->rev ^= 1;
82
83
      Node *GetRoute(Node *u, Node *v) {
84
        ChangeRoot(u);
85
        return Access(v);
86
87
   };
        Partition tree
    1.5
 1
   | int n,m;
   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;</pre>
10
      return a.v < b.v;</pre>
11
   void build(int depth,int l,int r) {
12
13
      if (l == r)
14
        return;
15
      int mid = (l+r)/2;
      int tl,tr;
16
17
      tl = tr = 0;
18
      for (int i = l; i <= r; i++) {
        if (cmp(a[d[depth][i]],a[mid])) {
19
20
          d[depth+1][l+tl] = d[depth][i];
21
          tl++;
        } else {
22
          d[depth+1][mid+1+tr] = d[depth][i];
23
24
25
        }
26
        s[depth][i] = tl;
27
      build(depth+1,l,mid);
28
29
      build(depth+1,mid+1,r);
30
    int find(int depth,int dl,int dr,int fl,int fr,int k) {
31
32
      if (fl == fr)
33
        return a[d[depth][fl]].v;
34
      int ls,rs;
35
      int mid = (dl+dr)/2;
      ls = (fl == dl)? 0 : s[depth][fl-1];
36
37
      rs = s[depth][fr];
38
      return (rs-ls < k)?
             find(depth+1,mid+1,dr,mid+fl-dl-ls+1,mid+fr-dl-rs+1,k-(rs-ls))
39
40
             : find(depth+1,dl,mid,dl+ls,dl+rs-1,k);
41
42
    int main() {
      while (scanf("%d%d",&n,&m) != EOF) {
43
44
        for (int i = 1; i <= n; i++) {
          scanf("%d",&a[i].v);
45
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;
        build(0,1,n);
51
52
        int l,r,k;
53
        for (int i = 1; i <= m; i++) {</pre>
          scanf("%d%d%d",&l,&r,&k);
54
55
          printf("%d\n",find(0,1,n,l,r,k));
56
57
58
      return 0;
59
    \mathbf{2}
        Dynamic programming
   2.1
        RMQ
   void init() {
 1
 2
      int i,j;
 3
      int n=N,k=1,l=0;
      for (i=0; i<n; i++) {</pre>
 4
 5
        f[i][0]=ele[i].num;
        if (i+1>k*2) {
 6
 7
          k*=2;
 8
          l++;
 9
10
        lent[i+1]=l;
11
      for (j=1; (1<<j)-1<n; j++)</pre>
12
        for (i=0; i+(1<<j)-1<n; i++)
13
          f[i][j]=max(f[i][j-1],f[i+(1<<(j-1))][j-1]);
14
15
16
    int fint(int x,int y) {
      int k=lent[y-x+1];
17
      return \max(f[x][k], f[y-(1<< k)+1][k]);
18
19
   }
    2.2
         2D-LIS
   |#include<cstdio>
 1
   #include<map>
 2
   using namespace std;
 3
   map<int,int> mp[100001];
 5
   bool check(int idx,int x,int y) {
      if (!idx) return 1;
 6
 7
      if (mp[idx].begin()->first>=x) return 0;
 8
      map<int, int> ::iterator it=mp[idx].lower_bound(x);
 9
      if (it->second<y) return 1;</pre>
10
11
      else return 0;
12
13
    int main() {
      int n;
14
      scanf("%d",&n);
15
16
      int l=0,r=0;
      for (int i=0; i<n; i++) {</pre>
17
18
        int x,y;
        scanf("%d%d",&x,&y);
19
        int tl=l,tr=r;
20
21
        while (tl<tr) {</pre>
          int mid=(tl+tr+1)/2;
22
```

```
23
          if (check(mid,x,y))
24
            tl=mid;
25
          else
            tr=mid-1;
26
27
        if (tl==r) r++;
28
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]);
        if (itl!=itr) mp[idx].erase(itl,itr);
34
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
   //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;</pre>
 5
        if (a.y == 0 && b.y == 0) return a.x < b.x;
 6
 7
     return a*b > 0;
 8
   3.1.2 Line
   Point operator &(const Line& b) const {
 1
 2
     Point res = s;
     double t = ((s - b.s) * (b.s - b.e)) / ((s - e) * (b.s - b.e));
 3
 4
     res.x += (e.x - s.x) * t;
 5
     res.y += (e.y - s.y) * t;
      return res;
 6
 7
   3.1.3 Functions
   Point nearestPointToLine(Point P, Line L) {
 1
 2
     Point result;
 3
     double a, b, t;
 4
     a = L.e.x-L.s.x;
 5
     b = L.e.y-L.s.y;
     t = ( (P.x-L.s.x)*a+(P.y-L.s.y)*b )/(a*a+b*b);
 6
     if (t >= 0 && t <= 1) {
 7
 8
        result.x = L.s.x+a*t;
 9
        result.y = L.s.y+b*t;
10
11
      return result;
12
   //Segment
13
14 | bool inter(Line l1, Line l2) {
```

```
15
     return
        max(l1.s.x, l1.e.x) >= min(l2.s.x, l2.e.x) &&
16
17
        max(l2.s.x, l2.e.x) >= min(l1.s.x, l1.e.x) &&
        max(l1.s.y, l1.e.y) >= min(l2.s.y, l2.e.y) &&
18
19
        max(l2.s.y,l2.e.y) >= min(l1.s.y,l1.e.y) &&
        sgn((l2.s-l1.s)*(l1.e-l1.s))*sgn((l2.e-l1.s)*(l1.e-l1.s)) <= 0 \&\&
20
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
   int inPoly(Point p,Point poly[], int n) {
28
29
     int i, count;
30
     Line ray, side;
     count = 0;
31
32
      ray.s = p;
33
      ray.e.y = p.y;
      ray.e.x = -1;//-\infty
34
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++;
        } else if (OnSeg(side.e, ray)) {
44
45
          if (side.e.y > side.s.y) count++;
46
        } else if (inter(ray, side)) {
47
          count++;
48
49
      return ((count % 2 == 1) ? 0 : 2);
50
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;
      for (int i = 2; i < n; i++) {</pre>
58
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;
        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;
62
63
        if (cmp(m + m0, 0.0) == 0)
          m0 += eps;
64
65
        p.x = (m * p.x + m0 * p0.x) / (m + m0);
        p.y = (m * p.y + m0 * p0.y) / (m + m0);
66
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;</pre>
```

```
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;
      for (int i = 1; i < n; i++)
10
        if (fabs(line[i].k - line[i - 1].k) > EPS)
11
12
          line[tot++] = line[i];
13
      int head = 0, tail = 1;
14
     Q[0] = line[0];
     Q[1] = line[1];
15
      resn = 0;
16
      for (int i = 2; i < tot; i++) {
17
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:
        while (head < tail && (((Q[tail] & Q[tail - 1]) - line[i].s) * (line[i].e - line[
21
           i].s)) > EPS)
22
          tail--:
23
        while (head < tail && (((Q[head] & Q[head + 1]) - line[i].s) * (line[i].e - line[
           i].s)) > EPS)
24
          head++;
25
        Q[++tail] = line[i];
26
27
     while (head < tail && (((Q[tail] & Q[tail - 1]) - Q[head].s) * (Q[head].e - Q[head]
         ].s)) > EPS)
28
        tail--:
     while (head < tail && (((Q[head] & Q[head + 1]) - Q[tail].s) * (Q[tail].e - Q[tail]
29
         ].s)) > EPS)
        head++;
30
31
     if (tail <= head + 1)
                               return;
      for (int i = head; i < tail; i++)</pre>
32
        res[resn++] = Q[i] & Q[i + 1];
33
34
     if (head < tail + 1)
        res[resn++] = Q[head] & Q[tail];
35
36
   3.1.5 Convex hull
   bool GScmp(Point a, Point b) {
 1
 2
      if (fabs(a.x - b.x) < eps)
        return a.y < b.y - eps;
 3
      return a.x < b.x - eps;
 4
 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);
     if (conPoint(p,n)) {
10
        res[resn++] = p[0];
11
12
        return;
13
     if (conLine(p,n)) {
14
        res[resn++] = p[0];
15
```

16

17

18

19

20

21 22

23

res[resn++] = p[n-1];

for (int i = 0; i < n;)</pre>

res[resn++] = p[i++];

(res[resn-1]-res[resn-2])*(p[i]-res[resn-1]) > 0)

if (resn < 2 ||

return;

else

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

49 | }

$3.2 \quad 3D$

```
3.2.1 Point
   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
   //Rotate around V, notice that |V|=1
 4
 5
   Point3D Trans(Point3D pa,Point3D V,double theta) {
     double s = sin(theta);
 6
 7
     double c = cos(theta);
 8
     double x,y,z;
 9
     x = V.x;
     y = V.y;
10
     z = V.z;
11
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
   | bool lineIntersect(Line3D L1, Line3D L2) {
 1
 2
      Point3D s = L1.s-L1.e;
 3
     Point3D e = L2.s-L2.e;
 4
     Point3D p = s*e;
                                     //Parallel
 5
     if (ZERO(p)) return false;
 6
     p = (L2.s-L1.e)*(L1.s-L1.e);
 7
                                     //Common face
      return ZERO(p&L2.e);
 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);
     Point t1 = (b-a)*(c-a);
12
13
     Point t2 = (b-a)*(d-a);
14
     Point t3 = (d-c)*(a-c);
     Point t4 = (d-c)*(b-c);
15
      return sgn(t1&ret)*sgn(t2&ret) < 0 &&</pre>
16
17
             sgn(t3\&ret)*sgn(t4\&ret) < 0;
18
19
   //Distance from point p to line L
   double distance(Point3D p, Line3D L) {
20
21
      return (Norm((p-L.s)*(L.e-L.s))/Norm(L.e-L.s));
22
   //Angle between line L_1 and L_2, \theta \in [0,\pi]
23
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 {
```

```
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) {}
      pt operator * (pt p) {}
 6
 7
      double operator ^ (pt p) {}
 8
   struct _3DCH {
9
      struct fac {
10
11
        int a, b, c;
12
        bool ok;
13
      };
14
      int n;
15
      pt P[MAXV];
16
      int cnt;
      fac F[MAXV*8];
17
      int to[MAXV][MAXV];
18
19
      double vlen(pt a) {
20
        return sqrt(a.x*a.x+a.y*a.y+a.z*a.z);
21
      double area(pt a, pt b, pt c) {
22
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) {
        pt m = P[f.b]-P[f.a], n = P[f.c]-P[f.a], t = p-P[f.a];
29
30
        return (m * n) ^ t;
31
32
      void deal(int p, int a, int b) {
        int f = to[a][b];
33
        fac add;
34
35
        if (F[f].ok) {
          if (ptof(P[p], F[f]) > eps)
36
            dfs(p, f);
37
38
39
            add.a = b, add.b = a, add.c = p, add.ok = 1;
            to[p][b] = to[a][p] = to[b][a] = cnt;
40
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];
        return fabs(volume(a, b, c, P[F[t].a])) < eps && fabs(volume(a, b, c,</pre>
53
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;
        for (int i = 1; i < n; i++) {</pre>
61
62
          if (vlen(P[0] - P[i]) > eps) {
63
            swap(P[1], P[i]);
            sb = 0;
64
65
            break;
          }
66
67
        if (sb)return;
68
```

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

```
133
           for (int j = 0; j < i; j++) {
             if (same(i, j)) {
134
135
               nb = 0;
136
               break;
137
             }
138
139
           ans += nb;
140
141
         return ans;
142
143
       pt Fc[MAXV*8];
       double V[MAXV*8];
144
145
       pt Center() {
146
         pt 0(0,0,0);
147
         for (int i = 0; i < cnt; i++) {</pre>
148
           Fc[i].x = (0.x+P[F[i].a].x+P[F[i].b].x+P[F[i].c].x)/4.0;
           Fc[i].y = (0.y+P[F[i].a].y+P[F[i].b].y+P[F[i].c].y)/4.0;
149
150
           Fc[i].z = (0.z+P[F[i].a].z+P[F[i].b].z+P[F[i].c].z)/4.0;
151
           V[i] = volume(0,P[F[i].a],P[F[i].b],P[F[i].c]);
         }
152
153
         pt res = Fc[0],tmp;
         double m = V[0];
154
         for (int i = 1; i < cnt; i++) {</pre>
155
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]);
           m += V[i];
161
162
           res = tmp;
163
164
         return res;
165
       }
166 | };
     3.3
         Circle
     3.3.1 Functions
    //Common area of two circle
  1
    double area(int x1,int y1,int x2,int y2,double r1,double r2) {
  2
  3
       double s=dis(x2-x1,y2-y1);
  4
       if(r1+r2<s) return 0;</pre>
       else if(r2-r1>s) return PI*r1*r1;
  5
       else if(r1-r2>s) return PI*r2*r2;
  6
  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
    for (int i = 1; i <= n; i++)
  1
  2
       ans[i] = 0.0;
  3
     for (int i = 0; i < n; i++) {
  4
       tote = 0;
       e[tote++] = Event(-pi,1);
  5
       e[tote++] = Event(pi,-1);
  6
  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);
           AB = lab.Length();
 10
```

```
11
          AC = c[i].r;
          BC = c[j].r;
12
13
          if (cmp(AB+AC,BC) <= 0) {
            e[tote++] = Event(-pi,1);
14
15
            e[tote++] = Event(pi,-1);
            continue;
16
17
          if (cmp(AB+BC,AC) <= 0) continue;</pre>
18
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));
          a0 = theta-fai;
22
          if (cmp(a0,-pi) < 0) a0 += 2*pi;
23
          a1 = theta+fai;
24
25
          if (cmp(a1,pi) > 0) a1 -= 2*pi;
26
          if (cmp(a0,a1) > 0) {
27
            e[tote++] = Event(a0,1);
            e[tote++] = Event(pi,-1);
28
29
            e[tote++] = Event(-pi,1);
30
            e[tote++] = Event(a1,-1);
          } else {
31
            e[tote++] = Event(a0,1);
32
33
            e[tote++] = Event(a1,-1);
34
35
        }
36
      sort(e,e+tote,Eventcmp);
37
      cur = 0;
      for (int j = 0; j < tote; j++) {</pre>
38
        if (cur != 0 && cmp(e[j].tim,pre[cur]) != 0) {
39
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[
             cur])),
                             Point(c[i].c.x+c[i].r*cos(e[j].tim),c[i].c.y+c[i].r*sin(e[j].
42
                                 tim)))/2.0;
43
        }
        cur += e[j].typ;
44
45
        pre[cur] = e[j].tim;
46
47
48
   for (int i = 1; i < n; i++)</pre>
49
     ans[i] = ans[i+1];
    3.3.3 Area of intersection part with polygon
   bool InCircle(Point a, double r) {
 1
 2
      return cmp(a.x*a.x+a.y*a.y,r*r) <= 0;
      //\epsilon should big enough
 3
 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);
      Line tmp = Line(Point(0,0),Point(tv.y,-tv.x));
10
      Point near = LineToLine(Line(a,b),tmp);
11
12
      if (cmp(near.x*near.x*near.y*near.y,r*r) <= 0) {</pre>
13
        double A,B,C;
        A = near.x*near.x+near.y*near.y;
14
        C = r;
15
        B = C*C-A;
16
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);
```

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

3.4 Matrix

3.4.1 基本矩阵

按向量 (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:

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

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

```
const int MAXEDGE=50000;
   const int MAXN=3000;
   const int inf=0x3fffffff;
 3
 4
   struct edges {
 5
      int cap,to,next,flow;
 6
   } edge[MAXEDGE+100];
   struct nodes {
 7
      int head, label, pre, cur;
 8
 9
   } node[MAXN+100];
10
   int L,N;
    int gap[MAXN+100];
11
    void init(int n) {
12
13
      L=0;
      N=n;
14
15
      for (int i=0; i<N; i++)</pre>
16
        node[i].head=-1;
17
    void add_edge(int x,int y,int z,int w) {
18
19
      edge[L].cap=z;
20
      edge[L].flow=0;
      edge[L].to=y;
21
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++) {</pre>
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) {</pre>
41
        if (u==t) {
42
          int min=inf;
          for (int i=node[u].pre; i!=-1; i=node[edge[i^1].to].pre)
43
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
        bool flag=false;
54
        int ∨;
55
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;
            node[u].cur=node[v].pre=i;
61
62
            break;
          }
63
64
        if (flag) {
65
66
          u=v;
67
          continue;
68
69
        node[u].cur=node[u].head;
70
        int min=N;
        for (int i=node[u].head; i!=-1; i=edge[i].next)
71
72
          if (edge[i].cap-edge[i].flow && node[edge[i].to].label<min)</pre>
73
            min=node[edge[i].to].label;
74
        gap[node[u].label]--;
75
        if (!gap[node[u].label]) return ans;
        node[u].label=min+1;
76
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

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

```
39
     node[s].visit=1;
40
     q.push(s);
41
     while (!q.empty()) {
        int u=q.front();
42
43
        node[u].visit=0;
        for (int i=node[u].head; i!=-1; i=edge[i].next) {
44
45
          int v=edge[i].to;
          if (edge[i].cap>edge[i].flow &&
46
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
   int mcmf(int s,int t,int &cost) {
63
64
     int flow=0;
65
     while (spfa(s,t)) {
        int max=inf;
66
        for (int i=node[t].pre; i!=-1; i=node[edge[i^1].to].pre) {
67
          if (max>edge[i].cap-edge[i].flow)
68
69
            max=edge[i].cap-edge[i].flow;
70
        for (int i=node[t].pre; i!=-1; i=node[edge[i^1].to].pre) {
71
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
   |#include <cstdio>
 1
 2
   #include <cstring>
   #include <algorithm>
 3
   #include <queue>
   #include <stack>
 5
   using namespace std;
 6
 7
   const int MAXN = 2003;
   const int MAXM = 2000 * 1999 / 2 + 2000 * 3;
 8
 9
   int N, L;
10
   int head[MAXN];
11
   struct Edge {
12
      int to, next, flow, cost;
13
   } edge[MAXM * 2];
   int h[MAXN], dis[MAXN], pre[MAXN];
14
15
   struct Heap {
16
      int value[MAXN + 1], id[MAXN + 1];
17
      int pos[MAXN];
18
      int size;
```

void init() {

19

```
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
      void push_up(int p) {
28
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) {
          int best = p;
36
          if (p * 2 < size && value[p] > value[p * 2])
37
38
            best = p * 2;
          if (p * 2 + 1 < size && value[best] > value[p * 2 + 1])
39
40
            best = p * 2 + 1;
          if (p == best)
41
42
            break;
43
          swap2(p, best);
44
          p = best;
45
        }
46
      void push(int _value, int _id) {
47
        value[size] = _value;
48
        id[size] = _id;
49
        pos[_id] = size;
50
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
      void update(int _value, int _id) {
62
        int p = pos[_id];
63
64
        value[p] = _value;
65
        push_up(p);
      }
66
67
    } heap;
   bool inque[MAXN];
68
   void init(int n) {
69
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;
      edge[L].next = head[v];
83
```

```
84
       head[v] = L++;
 85
 86
    void spfa(int s) {
       memset(dis, 63, 4 * N);
 87
 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();
         inque[u] = 0;
 95
         que.pop();
 96
         for (int i = head[u]; i != -1; i = edge[i].next)
 97
 98
           if (edge[i].flow) {
 99
             int v = edge[i].to;
             if (dis[v] > dis[u] + edge[i].cost) {
100
               dis[v] = dis[u] + edge[i].cost;
101
102
               pre[v] = i;
103
               if (!inque[v]) {
104
                 inque[v] = 1;
105
                 que.push(v);
106
107
             }
108
           }
109
       }
110
     void dijkstra(int s) {
111
       for (int i = 0; i < N; ++i)</pre>
112
         h[i] += dis[i];
113
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) {
         int u = heap.top();
122
123
         heap.pop();
         for (int i = head[u]; i != -1; i = edge[i].next)
124
125
           if (edge[i].flow) {
126
             int v = edge[i].to;
             if (dis[v] > dis[u] + edge[i].cost + h[u] - h[v]) {
127
               dis[v] = dis[u] + edge[i].cost + h[u] - h[v];
128
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
     int minimumCostFlow(int s, int t, int &cost) {
139
140
       int flow = 0;
141
       memset(h, 0, 4 * N);
142
       for (spfa(s); pre[t] != −1; dijkstra(s)) {
         int maxs = edge[pre[t]].flow;
143
         for (int i = pre[t]; i != -1; i = pre[edge[i ^ 1].to])
144
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
         flow += maxs;
151
152
       }
153
       return flow;
154
     int main() {
155
156
       return 0;
157
         Bi-connect
  1
    struct edges {
  2
       int to,next;
  3
       bool cut, visit;
  4
    } edge[MAXM<<1];</pre>
    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;
       for (int i=head[u]; i!=-1; i=edge[i].next) {
 26
 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];
         if (low[v]>=dpt[u] || u==fu) {
 39
 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
         low[u]=low[v]>low[u]?low[u]:low[v];
 48
 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++) {
          int u,v;
58
          scanf("%d%d",&u,&v);
59
60
          add_edge(u,v);
61
          add_edge(v,u);
62
        }
        idx=0;
63
        for (int i=0; i<n; i++)</pre>
64
65
          if (!visit[i])
66
            dfs(i,i,0);
67
68
      return 0;
69
         Cut and bridge
 1
   struct edges {
 2
      int to,next;
 3
      bool cut, visit;
 4
      int from;
    } edge[MAXN-1<<1];</pre>
 5
   int head[MAXN],low[MAXN],dfn[MAXN],L;
 6
 7
    bool visit[MAXN], cut[MAXN];
 8
   void init(int n) {
 9
      L=0;
      memset(head, -1, 4*n);
10
11
      memset(cut, 0, 4*n);
12
      memset(visit,0,4*n);
13
    void add_edge(int u,int v) {
14
      edge[L].from=u;
15
16
      edge[L].cut=edge[L].visit=0;
17
      edge[L].to=v;
18
      edge[L].next=head[u];
19
      head[u]=L++;
20
21
    int idx;
22
    void dfs(int u,int fu) {
23
      visit[u]=1;
24
      low[u]=dfn[u]=idx++;
25
      int tot=0;
26
      for (int i=head[u]; i!=-1; i=edge[i].next) {
27
        int v=edge[i].to;
        if (edge[i].visit)
28
29
          continue;
30
        edge[i].visit=edge[i^1].visit=1;
31
        if (visit[v]) {
32
          low[u]=dfn[v]>low[u]?low[u]:dfn[v];
33
          continue;
34
        }
35
        dfs(v,u);
36
        edge[i].cut=edge[i^1].cut=low[v]>dfn[u] || edge[i].cut;
37
        if (u!=fu) cut[u]=low[v]>=dfn[u]?1:cut[u];
38
        low[u] = low[v] > low[u] ? low[u] : low[v];
39
        tot++;
40
      if (u==fu && tot>1) cut[u]=1;
41
```

```
42
   }
43
    int main() {
44
      int t;
45
      scanf("%d",&t);
46
      while (t---) {
        int n,m;
47
        scanf("%d%d",&n,&m);
48
49
        init(n);
50
        for (int i=0; i<m; i++) {
51
          int u,v;
          scanf("%d%d",&u,&v);
52
53
          add_edge(--u,--v);
54
          add_edge(v,u);
55
56
        for (int i=0; i<n; i++)
57
          if (!visit[i]) {
58
            idx=0;
59
            dfs(i,i);
60
61
62
      return 0;
63
        Stoer-Wagner
   4.6
   int map[maxn][maxn];
   int n;
 2
   void contract(int x,int y) {
 3
      int i,j;
 4
 5
      for (i=0; i<n; i++)
        if (i!=x) map[x][i]+=map[y][i],map[i][x]+=map[i][y];
 6
 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
      n--;
11
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;
      for (i=0; i<n; i++) w[i]=map[0][i];</pre>
19
20
      for (i=1; i+1<n; i++) {</pre>
        t=k=-1;
21
        for (j=0; j<n; j++) if (c[j]==0&&w[j]>k)
22
23
            k=w[t=j];
        c[sx=t]=1;
24
25
        for (j=0; j<n; j++) w[j]+=map[t][j];</pre>
26
27
      for (i=0; i<n; i++) if (c[i]==0) return w[tx=i];</pre>
28
29
    int main() {
30
      int i,j,k,m;
      while (scanf("%d%d",&n,&m)!=EOF) {
31
32
        memset(map,0,sizeof(map));
33
        while (m——) {
          scanf("%d%d%d",&i,&j,&k);
34
35
          map[i][j]+=k;
36
          map[j][i]+=k;
37
        int mint=999999999;
38
```

```
40
          k=mincut();
41
          if (k<mint) mint=k;</pre>
42
          contract(sx,tx);
43
        }
44
        printf("%d\n",mint);
45
46
      return 0;
47
        Euler path
   4.7
   //Directed graph
 1
   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]--;
          match[x]--;
11
12
          solve(i);
13
14
      path[++l]=x;
15
    //Undirected graph
16
    void solve(int x) {
17
      int i;
18
19
      if (!match[x]) {
20
        path[++l]=x;
21
        return ;
22
23
      for (i=1; i<=n; i++)</pre>
24
        if (b[x][i]) {
25
          b[x][i]--;
26
          b[i][x]--;
27
          match[x]--;
          match[i]--;
28
29
          solve(i);
30
      path[++l]=x;
31
32 | }
         Strongly connected component
   | int dfsnum[2000];
 2
   int low[2000];
   int stack[2000];
 3
 4
    int top;
   int ans;
 5
   int an;
 6
   int be[2000];
 7
 8
   int flag[2000];
   void dfs(int x) {
 9
      dfsnum[x] = low[x] = ans++;
10
      stack[++top] = x;
11
12
      flag[x] = 1;
13
      for (int i = head[x]; i != -1; i = edge[i].next) {
14
        int y = edge[i].to;
```

39

while (n>1) {

```
15
        if (dfsnum[y] == -1) {
          dfs(y);
16
17
          low[x] = min(low[x], low[y]);
        } else if (flag[y] == 1)
18
19
          low[x] = min(low[x], dfsnum[y]);
20
     if (dfsnum[x] == low[x]) {
21
        while (stack[top] != x) {
22
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
   void SC() {
32
     memset(dfsnum,-1,sizeof(dfsnum));
33
     memset(flag,0,sizeof(flag));
34
35
     top = 0;
     an = 0;
36
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
   bool check(int u) {
 1
      for (int i=head[u]; i!=-1; i=edge[i].next) {
 2
        int v=edge[i].to;
 3
 4
        if (!use[v]) {
 5
          use[v]=1;
          if (pre[v]==-1 || check(pre[v])) {
 6
 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));
     for (int u=1; u<=N; u++) {
17
18
        memset(use,0,sizeof(use));
19
        if (check(u))
20
          ret++;
21
22
      return ret;
23
   4.9.2 Edmonds
 1
   int N;
   bool Graph[MaxN+1][MaxN+1];
 2
   int Match[MaxN+1];
 4 | bool InQueue[MaxN+1],InPath[MaxN+1],InBlossom[MaxN+1];
```

```
5
   int Head, Tail;
   int Queue[MaxN+1];
 6
 7
   int Start,Finish;
   int NewBase;
 9
   int Father[MaxN+1], Base[MaxN+1];
    int Count;
10
11
    void CreateGraph() {}
   void Push(int u) {
12
13
      Queue[Tail] = u;
14
      Tail++;
15
      InQueue[u] = true;
16
17
    int Pop() {
      int res = Queue[Head];
18
19
      Head++;
20
      return res;
21
    int FindCommonAncestor(int u,int v) {
22
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) {
        v = Base[v];
31
        if (InPath[v]) break;
32
33
        v = Father[Match[v]];
34
35
      return v;
36
   void ResetTrace(int u) {
37
38
39
      while (Base[u] != NewBase) {
        v = Match[u];
40
41
        InBlossom[Base[u]] = InBlossom[Base[v]] = true;
        u = Father[v];
42
43
        if (Base[u] != NewBase) Father[u] = v;
44
      }
45
46
    void BlossomContract(int u,int v) {
      NewBase = FindCommonAncestor(u,v);
47
48
      memset(InBlossom, false, sizeof(InBlossom));
49
      ResetTrace(u);
      ResetTrace(v);
50
51
      if (Base[u] != NewBase) Father[u] = v;
52
      if (Base[v] != NewBase) Father[v] = u;
      for (int tu = 1; tu <= N; tu++)
53
54
        if (InBlossom[Base[tu]]) {
55
          Base[tu] = NewBase;
56
          if (!InQueue[tu]) Push(tu);
57
58
59
    void FindAugmentingPath() {
      memset(InQueue, false, sizeof(InQueue));
60
61
      memset(Father,0,sizeof(Father));
62
      for (int i = 1; i <= N; i++)</pre>
        Base[i] = i;
63
      Head = Tail = 1;
64
65
      Push(Start);
66
      Finish = 0;
67
      while (Head < Tail) {</pre>
        int u = Pop();
68
```

```
69
         for (int v = 1; v <= N; v++)
           if (Graph[u][v] && (Base[u] != Base[v]) && (Match[u] != v)) {
 70
 71
             if ((v == Start) ||
 72
                  ((Match[v] > 0) \&\& (Father[Match[v]] > 0)))
 73
               BlossomContract(u,v);
             else if (Father[v] == 0) {
 74
 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;
       u = Finish;
 88
 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) {
           Start = u;
101
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];
     int lx[N],ly[N];
     int matchy[N];
     int map[N][N];
  5
     bool find(int x) {
  6
       visx[x]=true;
       int t;
  7
  8
       for (int y=0; y<ycnt; y++) {</pre>
         if (!visy[y]) {
  9
 10
           t=lx[x]+ly[y]-map[x][y];
 11
           if (t==0) {
             visy[y]=true;
 12
             if (matchy[y]==-1 || find(matchy[y])) {
 13
 14
               matchy[y]=x;
 15
               return true;
 16
           } else if (lack>t) lack=t;
 17
```

```
18
        }
19
20
      return false;
21
    void KM() {
22
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++)</pre>
27
        for (int j=0; j<ycnt; j++)</pre>
28
          if (map[i][j]>lx[i])
29
             lx[i]=map[i][j];
30
      for (int x=0; x<xcnt; x++) {</pre>
        while (true) {
31
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++) {
             if (visx[i]) lx[i]-=lack;
37
38
             if (visy[i]) ly[i]+=lack;
39
40
        }
41
42
      int cost=0;
43
      for (int i=0; i<ycnt; i++)</pre>
44
        cost+=map[matchy[i]][i];
45
    4.10 Clique
   |bool am[100][100];
 1
    int ans;
    int c[100];
    int U[100][100];
 4
    int n;
 5
 6
    bool dfs(int rest,int num) {
 7
      if (!rest) {
 8
        if (num>=ans)
 9
           return 1;
10
        else
11
          return 0;
12
      int pre=-1;
13
14
      for (int i=0; i<rest && rest-i+num>=ans; i++) {
        int idx=U[num][i];
15
        if (num+c[idx]<ans)</pre>
16
17
          return 0;
18
        int nrest=0;
19
        for (int j=i+1; j<rest; j++)</pre>
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() {
      while (scanf("%d",&n),n) {
28
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++)
            if (am[i][j])
36
37
              U[0][rest++]=j;
          ans+=dfs(rest,0);
38
39
          c[i]=ans;
40
41
        printf("%d\n",ans);
42
43
      return 0;
44
```

4.11 Spanning tree

4.11.1 Count the number of spanning tree

4.11.2 Spanning tree on directed graph

```
struct Edge {
 1
 2
      int u,v,cost;
 3
   };
   Edge e[1001*1001];
   int pre[1001],id[1001],visit[1001],in[1001];
 5
    int zhuliu(int root,int n,int m,Edge e[]) {
 6
      int res = 0,u,v;
 7
 8
      while (true) {
 9
        for (int i = 0; i < n; i++)
10
          in[i] = inf;
        for (int i = 0; i < m; i++)
11
12
          if (e[i].u != e[i].v && e[i].cost < in[e[i].v]) {</pre>
            pre[e[i].v] = e[i].u;
13
14
            in[e[i].v] = e[i].cost;
15
16
        for (int i = 0; i < n; i++)</pre>
17
          if (i != root)
18
            if (in[i] == inf)
                                  return -1;
19
        int tn = 0;
        memset(id,-1,sizeof(id));
20
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;
          while (visit[v] != i && id[v] == -1 && v != root) {
26
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;
            id[v] = tn++;
33
```

```
34
          }
35
        if(tn == 0) break;
36
        for (int i = 0; i < n; i++)</pre>
37
          if (id[i] == -1)
38
            id[i] = tn++;
39
        for (int i = 0; i < m;) {</pre>
40
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
            swap(e[i],e[--m]);
47
48
        }
        n = tn;
49
50
        root = id[root];
51
52
      return res;
53
   }
    4.12 Kth shortest path
   #include<cstdio>
 1
 2
   #include<cstring>
   #include<queue>
   using namespace std;
 4
   int K;
 5
 6
   class states {
 7
   public:
 8
      int cost,id;
 9
10
   int dist[1000];
11
   class cmp {
12
   public:
      bool operator ()(const states &i,const states &j) {
13
14
        return i.cost>j.cost;
15
16
   };
17
   class cmp2 {
18
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];
    int headr[1000],head[1000],Lr,L;
26
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;
```

for (int i=headr[u.id]; i!=-1; i=edger[i].next) {

38

39

40

states v=u;

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++;
       edger[Lr].to=u;
61
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]==0x3f3f3f3f3f)
         return -1;
68
       priority_queue<states,vector<states>,cmp2> q;
69
70
       states tmp;
 71
       tmp.id=s;
       tmp.cost=0;
72
       q.push(tmp);
73
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;
       scanf("%d%d",&n,&m);
91
92
       init(n);
       for (int i=0; i<m; i++) {
93
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 Stable marriage problem

假定有 n 个男生和 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。

```
#include<string.h>
    #include<stdio.h>
 2
    #define N 1050
 3
 4
    int boy[N][N];
 5
    int girl[N][N];
    int ans[N];
 6
 7
    int cur[N];
    int n;
 8
    void getMarry(int g) {
 9
      for (int i=ans[g]+1; i<n; i++) {</pre>
10
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]) {
          cur[b]=g;
19
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));
        scanf("%d",&n);
32
33
        for (int i=0; i<n; i++)</pre>
34
          for (int j=0; j<n; j++)</pre>
            scanf("%d",&girl[i][j]);
35
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
        memset(cur,0xff,sizeof(cur));
41
        memset(ans,0xff,sizeof(ans));
42
43
        for (int i=0; i<n; i++)</pre>
44
          getMarry(i);
45
        for (int i=0; i<n; i++)
          printf("%d\n",girl[i][ans[i]]);
46
47
48
      return 0;
49
```

Math 5

2

3

4

5

6 7

8 9

10

11

12 13

14

15 16 17

18

19 20

21

22

23

24 25

26

27 28

29

30

31 32

33 34

35

36

37

38

39 40 41

42

1

2

3

4

5

6

im = b;

vir operator +(const vir& b) const {

```
5.1 FFT
   5.1.1 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))
    5:tf(x1,x2) = (tf(x1) + tf(x2), tf(x1))
   | / / Transforms the interval [x, y) in a.
   void transform(int x, int y) {
     if (x == y - 1) {
       return;
     int 12 = (y - x) / 2;
     int z = x + 12;
     transform(x, z);
     transform(z, y);
     for (int i=x; i<z; i++) {</pre>
       int x1 = a[i];
       int x2 = a[i+l2];
       a[i] = (x1 - x2 + MOD) \% MOD;
       a[i+l2] = (x1 + x2) \% MOD;
    // Reverses the transform in
   // the interval [x, y) in a.
   void untransform(int x, int y) {
     if (x == y - 1) {
        return;
     int 12 = (y - x) / 2;
     int z = x + 12;
     for (int i=x; i<z; i++) {
       long long y1 = a[i];
       long long y2 = a[i+l2];
        // x1 - x2 = y1
       // x1 + x2 = y2
        // 2 * x1 = y1 + y2
        // 2 * x2 = y2 - y1
       // In order to solve those equations, we need to divide by 2
        // But we are performing operations modulo 1000000007
        // that needs us to find the modular multiplicative inverse of 2.
       // That is saved in the INV2 variable.
       a[i] = (int)(((y1 + y2)*INV2) % MOD);
       a[i+l2] = (int)(((y2 - y1 + MOD)*INV2) % MOD);
     untransform(x, z);
     untransform(z, y);
43 | }
   5.1.2 Standard
   struct vir {
     long double re, im;
     vir(long double a = 0, long double b = 0) {
       re = a;
```

```
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;
      for (i = 0; i < len; i++) {
19
        t = i;
20
        for (j = k = 0; j < loglen; j++, t >>= 1)
21
22
          k = (k << 1) | (t & 1);
23
        if (k < i) {
          vir wt = x[k];
24
25
          x[k] = x[i];
          x[i] = wt;
26
        }
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;
      for (i = 0; i < loglen; i++, t <<= 1) {
34
35
        s = 0;
        e = s + t;
36
        while (s < len) {</pre>
37
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;
            x[j] = a + b;
42
            x[j + t] = a - b;
43
44
            wn = wn * wo;
45
46
          s = e + t;
47
          e = s + t;
48
49
50
   void dit_fft(vir *x, int len, int loglen) {
51
      int i, j, s, e, t = 1 << loglen;</pre>
52
53
      for (i = 0; i < loglen; i++) {</pre>
54
        t >>= 1;
55
        s = 0;
56
        e = s + t;
57
        while (s < len) {</pre>
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];
            b = (x[j] - x[j + t]) * wn;
61
62
            x[j] = a;
            x[j + t] = b;
63
64
            wn = wn * wo;
65
          }
          s = e + t;
66
          e = s + t;
67
68
        }
69
70
      change(x, len, loglen);
71
      for (i = 0; i < len; i++)
```

```
72
        x[i].re /= len;
73 |}
    5.1.3 Usage
 1 | vir x1[MAXN], x2[MAXN];
    void solve(long long *a, int lena, long long *b, int lenb, long long *ret, int& len)
 2
 3
      int len1 = lena << 1;</pre>
      int len2 = lenb << 1;</pre>
 4
 5
      len = 1;
 6
      int loglen = 0;
      while (len < len1 || len < len2) {</pre>
 7
 8
        len <<= 1;
 9
        loglen++;
10
      for (int i = 0; i < lena; i++)</pre>
11
        x1[i] = vir(a[i], 0);
12
13
      for (int i = lena; i < len; i++)</pre>
14
        x1[i] = vir(0, 0);
15
      for (int i = 0; i < lenb; i++)
16
        x2[i] = vir(b[i], 0);
      for (int i = lenb; i < len; i++)</pre>
17
18
        x2[i] = vir(0, 0);
      fft(x1, len, loglen);
19
      fft(x2, len, loglen);
20
21
      for (int i = 0; i < len; i++)</pre>
22
        x1[i] = x1[i] * x2[i];
      dit_fft(x1, len, loglen);
23
24
      for (int i = 0; i < len; i++)
        ret[i] = (long long)(x1[i].re + 0.5);
25
26
    5.2 Euler function
    int getEuler(int x) {
 1
      getFactor(x);
 2
 3
      int ret=x;
 4
      for (int i=0; i<N; i++)
 5
        ret = ret/fac[i]*(fac[i]-1);
 6
      return ret;
 7
    void getEuler2() {
 8
      memset(euler,0,sizeof(euler));
 9
      euler[1] = 1;
10
      for (int i = 2; i <= 3000000; i++) {</pre>
11
12
        if (!euler[i]) {
13
          for (int j = i; j <= 3000000; j += i) {
             if (!euler[j])
14
               euler[j] = j;
15
             euler[j] = euler[j]/i*(i-1);
16
17
          }
        }
18
19
      }
20
    5.3
        \operatorname{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;
        y = tmp-(a/b)*y;
 6
 7
        return ret;
      } else {
 8
 9
        x = 1;
10
        y = 0;
11
        return a;
12
13
   5.4 Prime
   5.4.1 Get primes
   ∣int N;
 1
   bool isPrime[10001];
 2
   int prime[10000];
   void getPrime(int n) {
 5
      memset(isPrime,1,++n);
      N=0;
 6
 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++) {</pre>
          isPrime[i*prime[j]]=0;
12
          if (i%prime[j]==0)
13
14
            break;
15
        }
16
      }
17
   5.4.2 Get factors
   const int TIME = 8;
 1
   int factor[100], fac_top = -1;
 2
 3
    //GCD of bint
 4
   bint gcd(bint small,bint big) {
 5
      while(small) {
        swap(small,big);
 6
 7
        small%=big;
 8
 9
      return abs(big);
10
11
    //ret = (a*b)%n (n<2^62)
   bint muti_mod(bint a,bint b,bint n) {
12
13
      bint exp = a%n, res = 0;
14
      while(b) {
        if(b&1) {
15
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
   // \text{ 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)
          res=muti_mod(res,exp,m);
30
31
        exp = muti_mod(exp,exp,m);
32
        p>>=1;
33
34
      return muti_mod(res,exp,m);
35
36
    //miller-rabin
   bool miller_rabin(bint n, int times) {
37
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++) {</pre>
        a = rand() \% (n-1) + 1;
48
        x = mod_exp(a, u, n);
49
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
        if( y!=1) return false;
56
57
58
      return true;
59
60
   bint pollard_rho(bint n,int c) {
61
      bint x,y,d,i = 1,k = 2;
      srand(time(0));
62
      x = rand()%(n-1)+1;
63
64
      y = x;
      while(true) {
65
66
        i++;
67
        x = (muti\_mod(x,x,n) + c) % n;
68
        d = gcd(y-x, n);
        if (1 < d && d < n) return d;
69
        if( y == x) return n;
70
        if(i == k) {
71
72
          y = x;
          k <<= 1;
73
74
        }
75
      }
76
77
    void findFactor(bint n,int k) {
78
      if(n==1)return;
      if(miller_rabin(n, TIME)) {
79
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
```

5.5 Simpson

```
double Simp(double l,double r) {
 1
 2
      double h = (r-1)/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) {
      double mid = (l+r)/2.0;
 6
      if (abs((Simp(l,r)-Simp(l,mid)-Simp(mid,r)))/15 < eps)
 7
 8
        return Simp(l,r);
 9
      else
10
        return rSimp(l,mid)+rSimp(mid,r);
11
         Chinese remainder theorem
   | int m[10],a[10];//x \mod m_i = a_i
 1
   bool solve(int &m0,int &a0,int m,int a) {
 2
 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
    int MLES() {
14
15
      bool flag=1;
16
      int m0=1,a0=0;
      for (int i=0; i<n; i++)</pre>
17
18
        if (!solve(m0,a0,m[i],a[i])) {
19
          flag=0;
20
          break;
        }
21
      if (flag)
22
23
        return a0;
24
      else
25
        return −1;
26 }
    5.7
         Lucas
 1
    //num[i] = i!
    int comLucus(int n,int m,int p) {
 2
      int ans=1;
 3
 4
      for (; n && m && ans; n/=p,m/=p) {
 5
        if (n%p>=m%p)
          ans = ans*num[n%p]%p*getInv(num[m%p]%p)%p
 6
 7
                 *getInv(num[n%p-m%p])%p;
 8
        else
          ans=0;
 9
10
11
      return ans;
12
```

5.8 Primitive root

```
1
   int getPriRoot(int p) {
 2
     if (p==2) return 1;
 3
     int phi = p - 1;
 4
     getFactor(phi);
     for (int g = 2; g < p; ++g) {</pre>
 5
       bool flag=1;
 6
 7
       for (int i = 0; flag && i < N; ++i)</pre>
 8
         if (power(g, phi/fac[i], p) == 1)
 9
            flag=0;
10
       if (flag)
11
          return g;
12
     }
13
   }
       Inverse element
   void getInv2(int x) {
 1
     inv[1]=1;
 2
     for (int i=2; i<=x; i++)
 3
       inv[i]=(mod-(mod/i)*inv[mod%i]%mod)%mod;
 4
 5 | }
   5.10 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。。)
  #include <iostream>
 2
   #include <cstdio>
 3
   #include <cstring>
 4
   #include <algorithm>
 5
   #include <string>
 6
   using namespace std;
 7
   char s[100000];
 8
 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) {
     if (ch == '*') return 2;
18
19
     return 1;
20
21
22
   int P();
23
   int calc(int a,char op,int b) {
24
     if (op == '+')
25
       return a+b;
26
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) &&
              (get_priority(next_char()) >= p)) {
36
37
        char op = next_char();
38
        cur++;
39
        a = calc(a,op,calc_exp(get_priority(op)+1));
40
      return a;
41
42
43
44
    int totvar,m,var[26],varid[26];
45
46
    int P() {
47
      if (next_char() == '-') {
48
        cur++;
49
        return -P();
      } else if (next_char() == '+') {
50
51
        cur++;
        return P();
52
53
      } else if (next_char() == '(') {
54
        cur++;
55
        int res = calc_exp(0);
        cur++;
56
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) {
        scanf("%d%d",&totvar,&var[0]);
68
69
        if (totvar == 0 && var[0] == 0) break;
        for (int i = 1; i < totvar; i++)</pre>
70
71
          scanf("%d",&var[i]);
        scanf("%d",&m);
72
        scanf("%s",s);
73
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++)</pre>
79
          if (s[i] >= 'a' && s[i] <= 'z') {</pre>
80
            if (id[s[i]-'a'] == -1) {
              id[s[i]-'a'] = minid;
81
82
              minid++;
            }
83
            s[i] = 'a'+id[s[i]-'a'];
84
85
86
        for (int i = 0; i < totvar; i++)
          varid[i] = i;
87
88
        int res = 0;
89
        do {
90
          cur = 0;
91
          int tmp = calc_exp(0);
          if (tmp == m) {
92
93
            res++;
94
            break;
95
        } while (next_permutation(varid,varid+totvar));
96
```

```
97
         //puts(s);
98
         if (res > 0)
           puts("YES");
99
100
           puts("NO");
101
102
103
       return 0;
104
          Linear programming
    #define MAXM 20 //max num of basic varibles
  1
    #define INF 1E200
  2
  3
  4
    double A[MAXM+5][MAXN+MAXM+5];
    double b[MAXM+5],c[MAXN+MAXM+5];
  5
    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
     int fcmp(double a,double b) {
 11
12
       if(fabs(a-b)<1E-7) return 0;</pre>
13
       if(a>b) return 1;
14
       return -1;
15
16
    void Pivot(int l,int e) {
 17
 18
       double t=A[l][e],p=c[e];
       b[l]=b[l]/t;
 19
       for(int i=1; i<=C; i++)</pre>
20
21
         A[l][i]/=t;
22
       V=V-c[e]*b[l];
23
       for(int i=1; i<=R; i++) {</pre>
         if(i==l||fcmp(A[i][e],0.0)==0)
24
25
           continue;
         t=A[i][e];
26
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
       for(int i=1; i<=C; i++)</pre>
31
32
         c[i]=c[i]-p*A[l][i];
       for(int i=1; i<=nCnt; i++) {</pre>
33
 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++)</pre>
 47
           if(fcmp(P[N[i]],mV)==1)
             mV=P[N[i]],e=N[i];
 48
 49
 50
         if(fcmp(mV,0.0)<=0) break;
51
         int l=-1;
         mV=INF;
52
```

```
53
         for(int i=1; i<=bCnt; i++) {</pre>
           if(fcmp(A[i][e],0.0)==1) {
 54
 55
              double t=b[i]/A[i][e];
              if(fcmp(mV,t)==1||(fcmp(mV,t)==0&&(l==-1||B[l]>B[i])))
 56
 57
                mV=t,l=i;
 58
           }
 59
         if(l==-1) return false;
 60
 61
         Pivot(l,e);
 62
 63
       return true;
 64
 65
     bool initSimplex() {
 66
 67
       nCnt=bCnt=0;
 68
       for(int i=1; i<=n; i++)
         N[++nCnt]=i;
 69
 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;
       int p=-1;
 74
 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++;
       for(int i=0; i<=C; i++)</pre>
 82
 83
         A[R][i]=0.0;
 84
       for(int i=1; i<=R; i++)</pre>
 85
         A[i][n+m+1]=-1.0;
 86
       Pivot(p,n+m+1);
       if(!Process(A[R])) return false;
 87
 88
       if(fcmp(b[R],0.0)!=0)
 89
         return false;
 90
 91
       for(int i=1; i<=bCnt&&p==-1; i++)</pre>
 92
         if(B[i]==n+m+1) p=i;
 93
       if(p!=-1) {
 94
         for(int i=1; i<=nCnt; i++) {</pre>
 95
           if(fcmp(A[p][N[i]],0.0)!=0) {
 96
             Pivot(p,N[i]);
 97
             break;
 98
           }
 99
         }
100
       bool f=false;
101
102
       for(int i=1; i<=nCnt; i++) {</pre>
103
         if(N[i]==n+m+1) f=true;
104
         if(f&&i+1<=nCnt)
105
           N[i]=N[i+1];
106
       }
       nCnt—;
107
108
       R---, C---;
109
       return true;
110
111
     //-1: no solution 1: no bound 0: has a solution -V
112
113
     int Simplex() {
114
       if(!initSimplex())
115
         return -1;
       if(!Process(c))
116
```

```
117
                                   return 1;
                          for(int i=1; i<=nCnt; i++)</pre>
118
119
                                  X[N[i]]=0.0;
120
                           for(int i=1; i<=bCnt; i++)</pre>
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;
                           //b[1] = 5.0;
130
131
                           //Simplex();
132
                           //printf("V = %.3f\n",V);
133
                          while(scanf("%d",&v1[1]) == 1) {
134
135
                                  for(int i = 2; i<=6; i++)
                                           scanf("%d",&v1[i]);
136
137
                                  n = 4;
                                  m = 6;
138
139
                                   for(int i = 0 ; i<=m+1; i++)
140
                                           for(int j=0; j<=n+m+2; j++)</pre>
141
                                                  A[i][j] = c[j] = 0;
142
                                  memset(b,0,sizeof(b));
143
                                  V = 0.0;
                                   /*
144
                                  n 为未知数个数
145
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
                                   */
                                  b[1] = v1[1];
151
152
                                  A[1][1] = 1;
                                  A[1][4] = 1;
153
                                  b[2] = v1[2];
154
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;
                                  b[4] = v1[4];
160
                                  A[4][2] = 1;
161
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;
                                  A[6][2] = 1;
168
                                  c[1] = 1;
169
170
                                  c[2] = 1;
171
                                  c[3] = 1;
172
                                  c[4] = 1;
173
                                   Simplex();
174
                                  //printf("V = %.3f\n",V);
175
                                  printf("%.3f_{\perp}%.3f_{\perp}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{\parallel}%.3f_{
176
177
178
                           return 0;
179
```

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

```
|#include <stdio.h>
 1
   #include <string.h>
 2
 3
   #include <stdlib.h>
 4
   int p,expp,A,B,aa,ans,tt;
 5
   long long M;
    long long exp(int a,int b,long long mod) {
 6
 7
      long long ans=1,num=a;
      while (b!=0) {
 8
        if (b&1) {
 9
10
          ans=((ans%mod)*(num%mod))%mod;
11
12
        num=((num%mod)*(num%mod))%mod;
        b>>=1;
13
      }
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
        Α,Β;
22
        while (1) {
23
          aa=rand()%p;
24
          if (aa==0) continue;
          A=exp(aa,expp,p);
25
          ans=(((long long)A%p)*((long long)A%p))%p;
26
          if (ans==p-1) break;
27
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;
            A=B;
35
            B=tt;
36
37
          }
38
          tt=A;
          A=B;
39
40
          B=tt\%B;
41
          M=((long long) A*(long long) A
42
             +(long long)B*(long long)B)/p;
43
        if (B<=A) {
44
45
          x=B;
46
          y=A;
47
        } else {
48
          x=A;
          y=B;
49
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
      return 0;
59
   }
```

5.13 Partition ways of an integer

```
O(n\sqrt{n})
 1 | #include <cstdio>
   #include <cmath>
 2
   #include <cstring>
 4
   #include <map>
   #include <algorithm>
 5
 6
   using namespace std;
 7
   bool check(int x) {
      for (int i=2; i*i<=x; i++)</pre>
 8
 9
        if (x%i==0)
10
          return 0;
11
      return 1;
12
    int p[100000];
13
14
    inline int calc(int x) {
15
      return x*(x*3-1)/2;
16
17
    int main() {
18
      p[0]=1;
      for (int i=1; i<100000; i++) {</pre>
19
        for (int j=1,k=1; calc(j)<=i; j++,k*=-1) {</pre>
20
          p[i]+=k*p[i-calc(j)];
21
22
          if (p[i]<0)
23
            p[i]+=1000000;
          if (p[i]>=1000000)
24
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)
            p[i]-=1000000;
31
32
33
        if (!p[i])
          printf("%d\n",i);
34
35
36
      return 0;
37
    5.14 Pell's equation
   import java.math.BigInteger;
 1
    import java.util.*;
 2
 3
   public class Main {
 4
      public static class Fraction {
 5
        public BigInteger num,den;
        public Fraction() {
 6
 7
          num=BigInteger.ZERO;
 8
          den=BigInteger.ONE;
 9
        public Fraction(int _num,int _den) {
10
11
          num=BigInteger.valueOf(_num);
12
          den=BigInteger.valueOf(_den);
13
        public Fraction(BigInteger _num, BigInteger _den) {
14
15
          num=_num;
16
          den=_den;
17
        public Fraction gen() {
18
19
          BigInteger g=num.gcd(den);
```

```
20
          return new Fraction(num.divide(g),den.divide(g));
        }
21
22
        public Fraction add(Fraction x) {
          return new Fraction(x.num.multiply(den).add(num.multiply(x.den)),x.den.multiply
23
              (den)).gen();
24
25
        public Fraction reciprocal() {
26
          return new Fraction(den,num);
27
28
        public void out() {
          System.out.println(num+"/"+den);
29
30
      }
31
      public static BigInteger sqrt(BigInteger a) {
32
33
        BigInteger b=a;
34
        while (a.compareTo(b.multiply(b))<0)</pre>
          b=b.multiply(b).add(a).divide(b.multiply(BigInteger.valueOf(2)));
35
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];
      static int l;
42
43
      public static void main(String[] args) {
44
        BigInteger ans=BigInteger.ZERO;
45
        int idx=0;
        for (int n=2,r=2; n<=1000; n++) {</pre>
46
          if (n==r*r) {
47
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;
          while (true) {
54
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) {</pre>
                ans=x.num;
61
62
                idx=n;
63
64
              break;
65
            }
66
            a=-a;
            b=(n-a*a)/b;
67
68
          }
69
70
        System.out.println(idx);
71
      private static Fraction getFrac() {
72
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)
            return (int)i-1;
81
```

5.15 Polya

设 $G \in \mathcal{P}$ 个对象的一个置换群,用 k 种颜色去染这 \mathcal{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:

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

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:

$$C(f_i) = qcd(n, i)$$
, i 表示一次转过 i 颗宝石, $i = 0$ 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: $C(f) = \frac{n}{2} + 1$

拉格朗日插值法 5.16

已知 $y = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$ 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。

$$y = y_1 \frac{(x - x_2)(x - x_3) \cdots (x - x_n)}{(x_1 - x_2)(x_1 - x_3) \cdots (x_1 - x_n)}$$

$$+ y_2 \frac{(x - x_1)(x - x_3) \cdots (x - x_n)}{(x_2 - x_1)(x_2 - x_3) \cdots (x_2 - x_n)}$$

$$+ \cdots$$

$$+ y_n \frac{(x - x_1)(x - x_2) \cdots (x - x_{n-1})}{(x_n - x_1)(x_n - x_2) \cdots (x_n - x_{n-1})}$$

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:

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

这个组合系数可以通过高斯消元暴出来,前提是要猜到它满足递推关系。

5.17 正多面体顶点着色

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

正四面体:
$$N = \frac{12}{12}$$
 正六面体: $N = \frac{(n^8 + 17 \times n^4 + 6 \times n^2)}{24}$

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

下十二面体:
$$N = \frac{(n^{20}+15\times n^{10}+20\times n^8+24\times n^4)}{(n^{20}+15\times n^{10}+20\times n^8+24\times n^4)}$$

正六面体:
$$N = \frac{(n^{4+1} \times n^{4} + 12 \times n^{4})}{24}$$
 正八面体: $N = \frac{(n^{6} + 3 \times n^{4} + 12 \times n^{3} + 8 \times n^{2})}{24}$ 正十二面体: $N = \frac{(n^{20} + 15 \times n^{10} + 20 \times n^{8} + 24 \times n^{4})}{60}$ 正二十面体: $N = \frac{(n^{12} + 15 \times n^{6} + 44 \times n^{4})}{60}$

5.18 求和公式

$$\sum_{k=0}^{\infty} k = \frac{n \times (n+1)}{2}$$

$$\sum_{k=0}^{\infty} 2k - 1 = n^{2}$$

$$\sum_{k=0}^{\infty} k^{2} = \frac{n \times (n+1) \times (2n+1)}{6}$$

$$\sum_{k=0}^{\infty} (2k-1)^{2} = \frac{n \times (4n^{2}-1)}{3}$$

$$\sum_{k=0}^{\infty} k^{3} = (\frac{n \times (n+1)}{2})^{2}$$

$$\sum_{k=0}^{\infty} (2k-1)^{3} = n^{2} \times (2n^{2}-1)$$

$$\sum_{k=0}^{\infty} k^{4} = \frac{n \times (n+1) \times (2n+1) \times (3n^{2}+3n-1)}{30}$$

$$\sum_{k=0}^{\infty} k^{5} = \frac{n^{2} \times (n+1)^{2} \times (2n^{2}+2n-1)}{12}$$

$$\sum_{k=0}^{\infty} k \times (k+1) = \frac{n \times (n+1) \times (n+2)}{3}$$

$$\sum_{k=0}^{\infty} k \times (k+1) \times (k+2) = \frac{n \times (n+1) \times (n+2) \times (n+3)}{4}$$

$$\sum_{k=0}^{\infty} k \times (k+1) \times (k+2) \times (k+3) = \frac{n \times (n+1) \times (n+2) \times (n+3) \times (n+4)}{5}$$

5.19 几何公式

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: $V = \frac{2\pi r^2 h}{3}$

5.20 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 | int S[MaxM],O[MaxN];
```

```
5
      bool hasans;
 6
      void init() {
 7
        h = 0;
 8
        hasans = false;
 9
        tot = m+n;
        for (int i = 0; i <= m; i++) {</pre>
10
          D[i] = U[i] = Col[i] = i;
11
          Row[i] = S[i] = 0;
12
13
          L[i] = (i+m)\%(m+1);
          R[i] = (i+1)\%(m+1);
14
15
        for (int i = 1; i <= n; i++) {
16
          R[i+m] = L[i+m] = i+m;
17
          Row[i+m] = i;
18
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;
        while (true) {
39
          rowPos = R[rowPos];
40
          if (rowPos == x+m || Col[rowPos] > y)
41
                                                      break;
42
43
        rowPos = L[rowPos];
44
        if (Col[rowPos] == y)
                                  return;
45
        L[tot] = rowPos;
46
        R[tot] = R[rowPos];
        L[R[tot]] = R[L[tot]] = tot;
47
48
      void print(int deep) {
49
50
        for (int i = 0; i < deep; i++)</pre>
          printf("%d<sub>□</sub>", 0[i]);
51
52
        printf("\n");
53
      void cover(int col) {
54
        L[R[col]] = L[col];
55
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) {
              U[D[j]] = U[j];
60
61
              D[U[j]] = D[j];
              S[Col[j]]--;
62
63
            }
64
65
      void resume(int col) {
        for (int i = U[col]; i != col; i = U[i])
66
67
          for (int j = L[i]; j != i; j = L[j])
            if (Col[j] != col) {
68
```

```
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() {
         for (int i = 1; i <= n; i++) {</pre>
77
           L[R[i+m]] = L[i+m];
78
 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;
         cover(tc);
92
93
         for (int i = D[tc]; i != tc; i = D[i]) {
94
           int temp = O[deep];
95
           O[deep] = Row[i];
           for (int j = R[i]; j != i; j = R[j])
96
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
    DLX g;
 1
    g.n = ROW_SIZE;
    g.m = COL_SIZE;
    g.init();
    g.insert(ROW, COL);
    g.initDFS();
 7 g.DFS(0);
    6.2
         Dancing links (A-star)
    |namespace DLX {
 1
    const int MAXN = 1000;
 2
    const int MAXM = 400;
    const int INF = 0x3f3f3f3f;
 4
    int D[MAXN * MAXM], U[MAXN * MAXM], L[MAXN * MAXM], R[MAXN * MAXM], COL[MAXN * MAXM],
         ROW[MAXN * MAXM];
    int CNT, BEG[MAXN * MAXM], END[MAXN * MAXM], ANS, USE[MAXM], _USE[MAXM];
 6
    int SUM[MAXM];
 7
    bool vis[MAXM];
 8
 9
    void init(int n) {
 10
      memset(BEG, 0xff, sizeof(BEG));
```

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

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

7.1.2 Pointer version

```
const int CHAR=26;
const int TOTLEN=500000;
const int MAXLEN=1000000;
struct Vertex {
Vertex *fail,*next[CHAR];
```

```
6
      Vertex() {}
      Vertex(bool flag) { //为什么要这样写?
 7
 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
   void add(Vertex *pos,int cha) {
18
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];
      for (int i=0; i<1; i++) {</pre>
25
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];
      for (int i=0; i<CHAR; i++)</pre>
34
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;
      while (!que.empty()) {
41
42
        u=que.front();
        que.pop();
43
        for (int i=0; i<CHAR; i++)</pre>
44
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

```
//Self match
 1
 2
   int j;
 3
    p[0] = j = -1;
   for ( int i = 1; i < lb; i++) {</pre>
 4
 5
      while (j \ge 0 \&\& b[j + 1] != b[i]) j = p[j];
      if (b[j + 1] == b[i]) j ++;
 6
 7
      p[i] = j;
 8
   }
 9
    //Match
10
   j = -1;
   for ( int i = 0; i < la; i++) {
11
      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 $A[i \cdots]$ and B

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

7.4 Manacher

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

```
18
        else
19
          Mp[i] = 1;
20
        for (; Ma[i-Mp[i]] == Ma[i+Mp[i]]; Mp[i]++);
        if (i+Mp[i] > pnow) {
21
22
          pnow = i+Mp[i];
23
          pid = i;
24
25
      }
26
    /*
27
28
    abaaba
29
                            ,
7
                        2
30
           2
              1
                  4
                     1
                               2
                                  1
    0
        1
   */
31
         Suffix array
    7.5
    const int maxn = 200010;
 1
 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];</pre>
 6
    void da(int str[],int sa[],int rank[],int height[],int n,int m) {
 7
 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;</pre>
12
      for(i=0; i<n; i++)wss[x[i]=s[i]]++;
      for(i=1; i<m; i++)wss[i]+=wss[i-1];</pre>
13
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) {</pre>
16
        for(i=n-j,p=0; i<n; i++)y[p++]=i;</pre>
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]];</pre>
19
        for(i=0; i<m; i++)wss[i]=0;</pre>
20
        for(i=0; i<n; i++)wss[wv[i]]++;</pre>
        for(i=1; i<m; i++)wss[i]+=wss[i-1];</pre>
21
        for(i=n-1; i>=0; i--)sa[--wss[wv[i]]]=y[i];
22
        for(t=x,x=y,y=t,p=1,i=1,x[sa[0]]=0; i<n; i++)</pre>
23
24
          x[sa[i]] = cmp(y,n,sa[i-1],sa[i],j)?p-1:p++;
25
      for(int i=0; i<n; i++) rank[sa[i]]=i;</pre>
26
27
      for(int i=0,j=0,k=0; i<n; height[rank[i++]]=k)
        if(rank[i]>0)
28
29
          for(k?k--:0,j=sa[rank[i]-1];
               i+k < n && j+k < n && str[i+k]==str[j+k];
30
31
               k++);
32
    7.5.1 Longest common prefix
    int lcp(int x,int y) {
 1
 2
      if (x > y) swap(x,y);
 3
      if (x == y)
 4
        return len-sa[x];//NOTICE!
 5
      x++;
      int k = lent[y-x+1];
 6
      return min(f[x][k],f[y-(1<<k)+1][k]);
 7
 8
   //Interval
```

```
void getinterval(int pos,int comlen,int& pl,int& pr) {
10
11
      int l,r,mid,cp;
12
      l = 0;
13
      r = pos;
      while (l < r) {
14
        mid = l+r>>1;
15
        cp = lcp(mid,pos);
16
        if (cp < comlen)</pre>
17
          l = mid+1;
18
19
        else
20
          r = mid;
21
      }
      pl = l;
22
23
      l = pos;
      r = len-1;
24
25
      while (l < r) {
        mid = l+r+1>>1;
26
        cp = lcp(pos,mid);
27
28
        if (cp < comlen)</pre>
          r = mid-1;
29
30
        else
          l = mid;
31
32
33
      pr = l;
34
    7.6 Smallest represention
    int Gao(char a[],int len) {
 1
 2
      int i = 0, j = 1, k = 0;
      while (i < len && j < len && k < len) {</pre>
 3
        int cmp = a[(j+k)\%len]-a[(i+k)\%len];
 4
 5
        if (cmp == 0)
 6
          k++;
 7
        else {
 8
          if (cmp > 0)
 9
            j += k+1;
10
          else
            i += k+1;
11
          if (i == j) j++;
12
13
          k = 0;
14
        }
15
16
      return min(i,j);
17
   8
        Tool
        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) {
      return sta / bit[pos + 1] * bit[pos + 1] + val * bit[pos] + sta % bit[pos];
 6
 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) {
      return ((sta >> (pos + 1)) << (pos + 1)) | (val << pos) | (sta & ((1 << pos) - 1));
13
14 | }
   8.2 Hash map
   struct hash_map {
 1
      int head[MOD];
 2
      struct hash_tables {
 3
        int key1, key2;
 4
 5
        long long val;
 6
        int next;
 7
      } ele[ELE];
 8
     int N;
 9
     int getHash(int key1, int key2) {
        return (key1 * 1000000 + key2) % MOD;
10
11
     void init() {
12
13
        memset(head, -1, sizeof(head));
14
        N = 0;
15
     void clear() {
16
17
        for (int i = 0; i < N; i++)
18
          head[getHash(ele[i].key1, ele[i].key2)] = -1;
19
        N = 0;
20
     int fint(int key1, int key2) {
21
        for (int i = head[getHash(key1, key2)]; i != -1; i = ele[i].next) {
22
23
          if (ele[i].key1 == key1 && ele[i].key2 == key2)
24
            return i;
25
        return -1;
26
27
     void insert(int key1, int key2) {
28
29
        int tmp = getHash(key1, key2);
        ele[N].key1 = key1;
30
31
        ele[N].key2 = key2;
32
        ele[N].val = 0;
33
        ele[N].next = head[tmp];
34
        head[tmp] = N++;
35
     long long get(int key1, int key2) {
36
37
        int tmp = fint(key1, key2);
        if (tmp == -1) {
38
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
     void add(int key1, int key2, long long val) {
52
        int tmp = fint(key1, key2);
53
        if (tmp == -1) {
54
55
          insert(key1, key2);
56
          ele[N - 1].val += val;
57
        } else
```

```
ele[tmp].val += val;
58
59
      }
60 | };
        120 bit integer
   struct integer {
 1
      long long pa, pb;
 2
 3
      integer() {}
 4
      integer(long long _pa, long long _pb) {
 5
        pa = _pa;
        pb = _pb;
 6
 7
 8
      integer negate() {
 9
        if (pa == 0 && pb == 0)
          return integer(pa, pb);
10
        else if (pa == 0)
11
          return integer(pa, -pb);
12
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);
        if (ret.pb >= MOD) {
18
19
          ret.pa += 1;
20
          ret.pb -= MOD;
        }
21
22
        return ret;
23
24
      bool operator <(const integer& b) const {</pre>
25
        if (pa == b.pa)
26
          return pb < b.pb;</pre>
27
        return pa < b.pa;</pre>
28
29 | };
   8.4 Bash script
   while true; do
 1
 2
      ./gen > input
      ./sol < input > output.sol
 3
 4
      ./bf < input > output.bf
 5
 6
      diff output.sol output.bf
      if [ $? -ne 0 ] ; then break; fi
 7
   done
   8.5 Codeblocks settings
 1 | gnome—terminal —t $TITLE —x
   8.6 Bit operation
   8.6.1 基本操作
   注意括号
```

功能	示例	位运算
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	$(1101101 \to 101)$	x and 7
取末 <i>k</i> 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	$(1001011111 \rightarrow 100100000)$	x and $(x+1)$
把右起第一个 0 变成 1	$(1001011111 \rightarrow 1001111111)$	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	$(1001011111 \rightarrow 1111)$	(x xor (x+1)) shr 1
去掉右起第一个 1 的左边	$(100101000 \to 1000)$	x and x xor x -1)

8.6.2 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 | }
```

8.7 vimrc

```
syntax on
 1
 2
   set backspace=start,indent,eol
 3
    set showmode
   set showcmd
 5
   set hlsearch
 6
 7
   set nowrap
   set smarttab
9
   set autoindent
10
   set tabstop=4
    set softtabstop=4
11
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()
      exec "silent∟w"
18
      exec "silent<sub>□</sub>!rm<sub>□</sub>-f<sub>□</sub>'%:p.mzry1992'"
19
20
      exec "silent∟make"
      exec "cw"
21
22 endfunction
```

```
23
   function! Run()
24
     call Gao()
     let execFile = expand("%:p").".mzry"
25
     if filereadable(execFile)
26
27
        exec "silentu!gnome-terminalu-tu'%:p.mzry'u--working-directory='%:p:h'u-xu/usr/
           bin/cb_console_runner_'%:p.mzry'"
     endif
28
29
   endfunction
30
31
   colorscheme slate
32
   set gfn=Monospace\ 14
33
34
   map <C-F9> :call Gao()<Enter>
   imap <C-F9> <Esc>:call Gao()<Enter>
35
36
   map <F9> :call Run()<Enter>
37
   imap <F9> <Esc>:call Run()<Enter>
38
   map <C-c> :s!^!//<Enter>:noh<Enter>
39
40
   |imap <C-c> <Esc>:s!^!//<Enter>:noh<Enter>
   map <C-x> :s!//!<Enter>:noh<Enter>
41
42 | imap <C-x> <Esc>:s!//!<Enter>:noh<Enter>
```

9 Appendix

9.1 Template by elfness

9.1.1 AC machine

```
1 | #include < cstdio >
   #include<cstring>
 2
 3
   #include<cstdlib>
   #include<cmath>
 4
   #include<algorithm>
 6
   #include<iostream>
   using namespace std;
 7
   typedef long long LL;
 8
 9
   struct tree {
10
      tree *ne[26],*fail;
11
      int ct;
    } tr[500100], VD, *root, *Q[500100];
12
    int tn;
13
14
    void init() {
15
      tr[tn=0]=VD;
16
      root=tr+(tn++);
17
18
   char s[1000100];
19
   void build() {
20
      tree *p=root;
      for(int i=0; s[i]; i++) {
21
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];
          root->ne[i]->fail=root;
38
39
        } else root->ne[i]=root;
40
      while(top<tail) {</pre>
41
        p=Q[++top];
        for(i=0; i<26; i++)</pre>
42
43
          if(p->ne[i]!=NULL) {
44
            q=p->ne[i];
45
            Q[++tail]=q;
            q->fail=p->fail->ne[i];
46
47
            if(q->fail==NULL)q->fail=root;
          } else p->ne[i]=p->fail->ne[i];
48
      }
49
50
51
    int doit() {
      int ret=0;
52
      tree *p=root,*q;
53
      for(int i=0; s[i]; i++) {
54
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
   int i,n,_;
65
66
    int main() {
      for(i=0; i<26; i++)VD.ne[i]=NULL;</pre>
67
68
      VD.ct=0;
69
      scanf("%d",&_);
      while(_--) {
70
        scanf("%d",&n);
71
        init();
72
73
        for(i=0; i<n; i++) {</pre>
74
          scanf("%s",s);
75
          build();
76
        }
77
        pre();
        scanf("%s",s);
78
        printf("%d\n",doit());
79
80
81
   }
   9.1.2 E-KMP
   |#include<cstdio>
   #include<cstring>
 2
   #include<cstdlib>
 4
   #include<cmath>
 5
   #include<algorithm>
 6
   #include<iostream>
 7
   using namespace std;
   typedef long long LL;
 8
    void e_kmp(char *s,char *t,int *has,int *e_has) {
 9
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 ) {
          for(tn=mx-p; s[mx]==t[tn]; tn++)mx++;
13
```

```
14
          has[sp=p]=mx-p;
15
          if(mx==p)sp=++mx;
16
        } else has[p]=e_has[p-sp];
17
18
   const int V=1001000;
19
20
    char t[V],s[V];
21
    int e_has[V],has[V],tn;
22
    int main() {
      scanf("%s%s",s,t);
23
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)
   #include<cstdio>
 1
 2
   #include<cstring>
    #include<cstdlib>
 4
   #include<cmath>
   #include<algorithm>
 5
 6
   using namespace std;
 7
   const int V=1200;
   const int En=21000;
 8
   const int oo=10000000000;
 9
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;
      e[K].w=z;
16
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;
      for(int i=p[u]; i!=-1; i=e[i].ne) {
24
25
        int v=e[i].num;
26
        if(!sy[v]&&lx[u]+ly[v]==e[i].w) {
27
          sy[v]=true;
          if(mat[v]==-1||path(mat[v])) {
28
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++) {</pre>
40
        lx[i]=-oo;
41
        for(j=p[i]; j!=-1; j=e[j].ne)
          lx[i]=max(lx[i],e[j].w);
42
43
44
      for(i=0; i<N; i++)ly[i]=0,mat[i]=-1;</pre>
```

```
45
      for(int u=0; u<N; u++)
        while(1) {
46
47
           for(i=0; i<N; i++)sx[i]=0,sy[i]=0;</pre>
48
          if(path(u))break;
49
          int dx=oo;
50
           for(i=0; i<N; i++)if(sx[i])</pre>
               for(j=p[i]; j!=-1; j=e[j].ne)
51
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;</pre>
56
          for(i=0; i<N; i++)if(sy[i])ly[i]+=dx;</pre>
57
        }
58
      int ret=0;
59
      for(i=0; i<N; i++)ret+=lx[i]+ly[i];</pre>
60
      return -ret;
61
    int _,ca,n,m,i,x,y,z,te;
62
63
    int main() {
      scanf("%d",&_);
64
65
      ca=0;
      while(_--) {
66
67
        ca++;
        scanf("%d%d",&n,&m);
68
69
        N=n;
70
        for(i=0; i<n; i++)p[i]=-1;</pre>
71
        K=0;
        for(i=0; i<m; i++) {</pre>
72
           scanf("%d%d%d",&x,&y,&z);
73
74
75
          y--;
76
          add(x,y,-z);
          add(y,x,-z);
77
78
79
        te=KM();
        printf("Case<sub>□</sub>%d:<sub>□</sub>",ca);
80
        if(te==-1)puts("NO");
81
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
    #include <cstdio>
 8
    #include <cstring>
 9
10
    #include <cstdlib>
11
    #include <cmath>
    #include <algorithm>
12
13
14
   using namespace std;
15
   const int MaxN = 120000;
16
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
      Point operator—(const Point& p) const {
26
27
        return Point(x - p.x, y - p.y);
28
29
      double norm() const {
30
        return hypot(x, y);
31
32
      void init() {
        scanf("%lf%lf", &x, &y);
33
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;
      if (r - l <= 2) {
49
        for (int i = l; i < r; i++)</pre>
50
51
          for (int j = i + 1; j <= r; j++)
52
            ret = min(ret, (p[x[i]] - p[x[j]]).norm());
53
        return ret;
54
      }
55
56
      int mid = (l + r) >> 1;
      int lp = l, rp = mid + 1;
57
      for (int i = l; i <= r; i++)
58
        vst[x[i]] = i <= mid;</pre>
59
      for (int i = l; i <= r; i++)</pre>
60
61
        if (vst[y[k][i]])
62
          y[k + 1][lp++] = y[k][i];
63
        else
          y[k + 1][rp++] = y[k][i];
64
      double lhs = dfs(k + 1, l, mid);
65
      double rhs = dfs(k + 1, mid + 1, r);
66
67
      double mx = (p[x[mid + 1]].x + p[x[mid]].x) / 2.0;
68
      ret = min(lhs, rhs);
69
70
      lp = 0;
      for (int i = l; i <= r; i++)</pre>
71
72
        if (fabs(mx - p[y[k][i]].x) < ret)
73
          tmp[lp++] = y[k][i];
74
      for (int i = 0; i < lp; i++)</pre>
75
        for (int j = 1; j < 8 && i + j < lp && (p[tmp[i + j]].y - p[tmp[i]].y) < ret; j</pre>
76
           ++)
77
          ret = min(ret, (p[tmp[i]] - p[tmp[i + j]]).norm());
78
79
      return ret;
80
81
82
   int main() {
83 |#ifdef __FISH__
```

```
freopen("data.in", "r", stdin);
84
85
      freopen("nlogn.out", "w", stdout);
86
87
     while (scanf("%d", &n) == 1 && n) {
88
        for (int i = 0; i < n; i++) {</pre>
          p[i].init();
89
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
   #include<cstdio>
 2
   #include<cstring>
   #include<cstdlib>
 3
 4
   #include<cmath>
   #include<algorithm>
 5
 6
   #include<iostream>
   #include<vector>
 7
   #include<string>
 8
   using namespace std;
 9
10
   typedef long long LL;
11
   const int N=100100;
   char s[N]; /// 长度+1, 对于非字符串, 加一个小于最小值的元素,
12
   int sa[N]; /// 倍增算法, 结果 下标 1-n, 第
13
                                                   i 大的是 sa[i]
   int rk[N]; /// 第 i 位开始的后缀, 的排名为
15
   int wa[N],wb[N],wv[N],rmq[20][N];
16
   int sn, to[N];
   bool cmp(int *y,int a,int b,int L) {
17
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;
      for(i=0; i<len; i++)rk[x[i]=s[i]]++;</pre>
26
27
      for(i=0; i<dn; i++)rk[i+1]+=rk[i];</pre>
28
      for(i=len-1; i>=0; i--)sa[--rk[x[i]]]=i;
      for(j=1,p=1; p<len; j*=2,dn=p) {</pre>
29
        for(p=0; p<j; p++)y[p]=len-j+p;</pre>
30
        for(i=0; i<len; i++)if(sa[i]>=j)y[p++]=sa[i]-j;
31
        for(i=0; i<len; i++)wv[i]=x[y[i]];</pre>
32
        for(i=0; i<dn; i++)rk[i]=0;</pre>
33
        for(i=0; i<len; i++)rk[wv[i]]++;</pre>
34
35
        for(i=0; i<dn; i++)rk[i+1]+=rk[i];</pre>
        for(i=len-1; i>=0; i--)sa[--rk[wv[i]]]=y[i];
36
37
        swap(x,y);
38
        x[sa[0]]=0;
        for(p=i=1; i<len; i++) {</pre>
39
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++)</pre>
49
         rk[sa[i]] = i;
       for(i=0; i<len; i++) {</pre>
50
51
         if(k>0)k--;
         j=sa[rk[i]-1];
52
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;
       int rn=(int)floor(log(n*2.0)/log(2.0));
59
60
       for(i=1; i<rn; i++)
61
         for(j=0; j<n+2-(1<<(i-1)); j++)
           rmq[i][j]=min(rmq[i-1][j],rmq[i-1][j+(1<<(i-1))]);</pre>
62
63
     int askRMQ(int a, int b) { /// [a,b]闭区间
64
65
       int rq=to[b-a];
       return min(rmq[rq][a],rmq[rq][b+1-(1<<rq)]);</pre>
66
67
    void PT(char *s,int *sa) {
68
69
       int i,sn;
70
       sn=strlen(s);
71
       for(i=0; i<sn; i++)</pre>
72
         puts(s+sa[i+1]);
       puts("");
73
       for(i=0; i<sn; i++)</pre>
74
         printf("rank_%d_=_%d\n",i,rk[i]);
75
76
     int lcp(int a,int b,int len) {
77
78
       if(a==b)
79
         return len-a;
80
       a=rk[a];
       b=rk[b];
81
82
       if(a>b)swap(a,b);
83
       return askRMQ(a+1,b);
84
85
    void pre_log() {
86
       int i;
       to[0]=to[1]=0;
87
       for(i=1; i*2<N; i++)</pre>
88
89
         to[i*2]=to[i*2+1]=to[i]+1;
90
     int main() {
91
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);
         scanf("%d",&T);
100
101
         while (T--) {
102
           int a,b;
103
           scanf("%d%d",&a,&b);
           a--,b-;/// 求原串的
                                   a b 开始的后缀的公共前缀
104
           printf("lcp_{\sqcup}=_{\sqcup}%d\n",lcp(a,b,sn));
105
         }
106
107
108
       return 0;
```

9.1.6 SAP

```
#include<cstdio>
 1
 2
   #include<cstring>
   #include<cstdlib>
 3
 4
   #include<cmath>
 5
   #include<algorithm>
 6
   using namespace std;
 7
   const int V=220;
   const int En=200000;
 8
   const int oo=0x3f3f3f3f;
 9
10
   struct Edge {
11
      int num,ne,c;
12
   } e[En];
   int d[V],p[V],pre[V],low[V];
13
   int gap[V],cur[V];
15
    int N,K,st,ed;
   void add(int x,int y,int c) {
16
      e[K].num=y;
17
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() {
      int ret=0;
27
      bool fail;
28
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;
      gap[0]=N;
34
35
      int u=st;
36
      while(d[st]<N) {</pre>
37
        fail=true;
        for(int i=cur[u]; i!=-1; i=e[i].ne) {
38
39
          int v=e[i].num;
          cur[u]=i;
40
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]]--;
          if(!gap[d[u]])return ret;
59
```

```
60
         d[u]=N;
         for(int i=p[u]; i!=-1; i=e[i].ne)
61
62
           if(e[i].c)d[u]=min(d[u],d[e[i].num]+1);
63
         gap[d[u]]++;
64
         cur[u]=p[u];
         if(u!=st)u=e[pre[u]^1].num;
65
66
67
68
     return ret;
69
   9.1.7 一般图最大匹配
   #include <stdio.h>
 1
 2
   #include <string.h>
   #include <algorithm>
 3
   #include <vector>
   #define maxn 300
 6
   #define maxm 90010
 7
 8
   using namespace std;
 9
10
   int match[maxn];
                                //标记是否匹配
   int st[maxn],aim[maxm],nxt[maxm],ln; //边表
11
12
   int q[maxn];
                             //bfs队列
   int level[maxn];
                                //离根深度的奇偶性
13
                                  //存每个点到根的路径
   vector<int> ar[maxn];
14
                                //找到的一条增广路
15
   vector<int> a;
16
   int n;
17
   void init() {
18
     for(int i=0; i<n; i++)st[i]=-1;
     ln=0;
19
20
21
   void in_edge(int x,int y) {
22
     aim[ln]=y;
23
     nxt[ln]=st[x];
24
     st[x]=ln++;
25
                                    //求p和q的最近公共祖先
26
   int lca(int p,int q) {
27
     int ret=0;
28
     while (ret<ar[p].size() && ret<ar[q].size() && ar[p][ret]==ar[q][ret]) ret++;</pre>
29
     return ret-1;
30
   int FindAlterRoad(int sp) {
31
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++) {</pre>
38
       int x=q[p];
        for (int i=st[x]; i!=-1; i=nxt[i]) {
39
         int u=aim[i];
40
41
         if (match[u]==u) continue;
                                  //u是未访问的点
42
         if (level[u]==-1) {
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;
           } else {
                              //u 是已匹配的点
47
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
                                           //u和x同为偶点.形成花
           } else if (level[u]==1) {
 57
 58
             int root=lca(u,x);
59
             vector<int> tmp=ar[x];
             for (int i=ar[u].size()-1; i>root; i---) {
60
61
               int y=ar[u][i];
               tmp.push_back(y);
62
               if (level[y]==0) {
63
                 level[y]=1;
64
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---) {
               int y=ar[x][i];
72
 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
       memset(match,-1,sizeof(match));
88
89
       for (int i=0; i<n; i++)</pre>
90
         if (match[i]==-1)
           if (FindAlterRoad(i)) {
91
92
             for (int i=0; i<a.size(); i+=2) {</pre>
93
               int u=a[i],v=a[i+1];
94
               match[u]=v;
               match[v]=u;
95
96
             }
97
             ret++;
98
           } else match[i]=i;
99
       return ret;
100
    9.1.8 上下界最大流
  1
    Author: elfness@UESTC
  2
  3
    */
  4
    #include<cstdio>
    #include<cstring>
  5
    #include<cstdlib>
  6
  7
    #include<cmath>
    #include<algorithm>
  8
  9
    #include<iostream>
 10 | #include < vector >
```

```
11
   #include<string>
12
   using namespace std;
13
   typedef long long LL;
   const int V=1500;
14
15
   const int En=900000;
   const int inf=0x3f3f3f3f;
16
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
    int d[V],pre[V],pree[V],gap[V],cur[V];
32
33
    int N,st,ed;
34
    int low[V];
35
    int sap() {
36
      int ret=0;
37
      bool fail;
      for(int i=0; i<=N; i++) {</pre>
38
39
        d[i]=0;
40
        gap[i]=0;
        cur[i]=p[i];
41
42
        low[i]=0;
43
44
      low[st]=inf;
45
      gap[0]=N;
46
      int u=st;
47
      while(d[st]<N) {</pre>
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;
            low[v]=min(low[u],e[i].c);
55
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=pre[u];
               } while(u!=st);
62
63
               ret+=low[ed];
64
            }
65
            fail=false;
            break;
66
          }
67
68
69
        if(fail) {
70
          gap[d[u]]--;
          if(!gap[d[u]])return ret;
71
72
          d[u]=N;
73
          for(int i=p[u]; i!=-1; i=e[i].ne)
            if(e[i].c)d[u]=min(d[u],d[e[i].num]+1);
74
```

```
75
           gap[d[u]]++;
 76
           cur[u]=p[u];
 77
           if(u!=st)u=pre[u];
 78
 79
       }
 80
       return ret;
 81
 82
     int n,m,s,t;
 83
     struct Elf {
       int u,v,lo,up;
 84
 85
     } b[12000];
 86
     int lb[12000];
 87
     int doit() {
 88
       int i;
 89
       N=n+2;
 90
       st=n;
 91
       ed=n+1;
       for(i=0; i<N; i++)p[i]=-1;</pre>
 92
 93
 94
       for(i=0; i<n; i++)lb[i]=0;</pre>
 95
       for(i=0; i<m; i++) {</pre>
 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++) {</pre>
101
         if(lb[i]>0)add(st,i,lb[i]);
102
         else add(i,ed,-lb[i]);
103
       add(t,s,inf);
104
       int te=sap();
105
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
          上下界最小流
     9.1.9
    |#include<cstdio>
  1
     #include<cstdlib>
  2
     #include<cstring>
  3
     #include<cmath>
  5
     #include<algorithm>
  6
    using namespace std;
     const int V=600;
  7
     const int En=50000;
     const int oo=0x3f3f3f3f;
  9
 10
     struct Edge {
 11
       int num,ne,c;
     } e[En];
 12
 13
     int p[V],K;
     void add(int x,int y,int c) {
 14
 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

22

e[K].ne=p[y];

p[y]=K++;

```
23
   int d[V],cur[V],low[V],pre[V],gap[V],pree[V];
24
   int st,ed,N;
25
   int sap() {
26
27
      int ret=0;
      bool fail;
28
      memset(gap,0,sizeof(gap));
29
30
      memset(low,0,sizeof(low));
31
      memset(d,0,sizeof(d));
32
      for(int i=0; i<N; i++)cur[i]=p[i];</pre>
33
      gap[0]=N;
34
      low[st]=oo;
      int u=st;
35
36
      while(d[st]<N) {</pre>
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;
          if(e[i].c&&d[u]==d[v]+1) {
41
42
            pre[v]=u;
43
            pree[v]=i;
            low[v]=min(low[u],e[i].c);
44
45
            u=v;
            if(u==ed) {
46
              do {
47
48
                 e[pree[u]].c-=low[ed];
49
                 e[pree[u]^1].c+=low[ed];
50
                 u=pre[u];
51
               } while(u!=st);
52
               ret+=low[ed];
53
            }
54
            fail=false;
55
            break;
          }
56
57
58
        if(fail) {
59
          gap[d[u]]--;
          if(!gap[d[u]])return ret;
60
          d[u]=N;
61
62
          for(int i=p[u]; i!=-1; i=e[i].ne)
            if(e[i].c)d[u]=min(d[u],d[e[i].num]+1);
63
64
          gap[d[u]]++;
65
          cur[u]=p[u];
          if(u!=st)u=pre[u];
66
        }
67
68
69
      return ret;
70
71
   struct ELF {
72
      int u,v,lo;
73
    } b[En];
   int n,m,lb[V],ts,tt;
74
75
    void solve() {
      N=n+4;
76
77
      ts=0;
78
      tt=n+1;
79
      st=n+2;
80
      ed=n+3;
81
      memset(lb,0,sizeof(lb));
      int i,u,v;
82
83
      for(i=0; i<N; i++)p[i]=-1;
84
      K=0;
85
      for(i=0; i<m; i++) {</pre>
        u=b[i].u;
86
```

```
87
         v=b[i].v;
         lb[v]+=b[i].lo;
 88
 89
         lb[u]-=b[i].lo;
 90
         add(u,v,oo-b[i].lo);
 91
 92
       for(i=1; i<=n; i++) {</pre>
 93
         add(ts,i,oo);
 94
         add(i,tt,oo);
 95
       for(i=0; i<n+2; i++) {</pre>
 96
 97
         if(lb[i]>0)add(st,i,lb[i]);
 98
         else add(i,ed,-lb[i]);
 99
       }
       int ans=sap();
100
101
       add(tt,ts,oo);
102
       printf("%d\n",sap());
103
104
     int _,ca,i;
105
     int main() {
       scanf("%d",&_);
106
107
       ca=0;
       while(_--) {
108
109
         ca++;
         scanf("%d%d",&n,&m);
110
111
         for(i=0; i<m; i++) {</pre>
112
            scanf("%d%d%d",&b[i].u,&b[i].v,&b[i].lo);
113
         printf("Case<sub>□</sub>#%d:<sub>□</sub>",ca);
114
115
         solve();
116
117
     9.1.10
            全局最小割
    using namespace std;
  1
     #define inf 100000000
  2
     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));
       for(int i=0; i<N; i++) {</pre>
  9
         int Num=0,Max=—inf;
 10
         for(int j=0; j<V; j++)</pre>
 11
 12
            if(!com[j]&&!visit[j]&&W[j]>Max) {
 13
              Max=W[j];
 14
              Num=j;
           }
 15
         visit[Num]=true;
 16
 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
     int stoer(int V) {
 25
 26
       int Mincut=inf;
 27
       int N=V;
 28
       memset(com,0,sizeof(com));
```

for(int i=0; i<V-1; i++) {

29

```
30
        int Cut;
31
        s=0, t=0;
32
        Cut=maxadj(N,V);
33
34
        if(Cut<Mincut)Mincut=Cut;</pre>
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
   #include<cstdio>
   #include<cstring>
   #include<cstdlib>
 3
 4
   #include<cmath>
 5
   #include<algorithm>
 6
   using namespace std;
 7
   const int V=1200;
 8
   const int En=2100000;
 9
   struct Elf {
      int u,v,len;
10
   } b[En];
11
12
   const int oo=1000000000;
   int ret;
13
   int N,M,Root;//点数, 边数, 根, 默认从0开始
14
15
   int id[V],pre[V],cnt,vis[V];
    int in[V];
17
   bool TreeMST() {
18
      ret=0;
19
      int i,u,v;
20
      while(1) {
21
        for(i=0; i<N; i++)</pre>
22
          in[i]=oo;
23
        memset(pre,-1,sizeof(pre));
24
        for(i=0; i<M; i++) {</pre>
25
          u=b[i].u;
          v=b[i].v;
26
27
          if(b[i].len<in[v]&&u!=v) {
28
            pre[v]=u;
29
            in[v]=b[i].len;
          }
30
31
        for(i=0; i<N; i++) {</pre>
32
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));
        for(i=0; i<N; i++) {</pre>
40
          ret+=in[i];
41
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
        if(cnt==0)return true;
53
54
        for(i=0; i<N; i++)</pre>
55
          if(id[i]==-1)id[i]=cnt++;
56
        for(i=0; i<M; i++) {</pre>
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
        N=cnt;
63
64
        Root=id[Root];
65
66
      return true;
67 }
```