## 目录

max flow minimem cost	3
Merge sort reverse	5
prim	6
Kruskal	7
Segment tree	8
Spare Table	10
tarjan	11
Trie	12
4 points on a plane	15
BIT	15
Cantor	16
Dijstra	17
Dinic	
floyed	19
Wythoff	20
hangary	20
josephus	21
KMP	22
Manacher	22
Matrix pow	23
Math	25

## max flow minimem cost

1

```
2
     const int maxn=120;
 3
     const int oo = 0x3f3f3f3f;
4
     struct Edge
5
     {
 6
         int u, v, cap, flow, cost;
7
         Edge(int u, int v, int cap, int f, int cost):u(u), v(v), cap(cap),
8
     flow(f), cost(cost) {}
9
     };
10
11
     struct MCMF
12
13
         int n, m, s, t;
14
         vector<Edge> edge;
15
         vector<int> G[maxn];
16
         int inq[maxn], d[maxn], p[maxn], a[maxn];
17
18
         void init(int n)
19
         {
20
             this->n=n;
21
             for(int i=0; i<=n; i++)G[i].clear();</pre>
22
             edge.clear();
23
24
         void AddEdge(int u, int v, int cap, int cost)
25
26
             edge.push_back(Edge(u, v, cap, ₀, cost));
             edge.push_back(Edge(v, u, 0, 0, -cost));
27
28
             m=edge.size();
29
             G[u].push_back(m-2);
30
             G[v].push back(m-1);
31
32
33
         bool SPFA(int s, int t, int& flow, int& cost)
34
35
             memset(d, 0x3f, sizeof d);
36
             memset(inq, 0, sizeof inq);
37
             d[s]=0, inq[s]=1, p[s]=0, a[s]=oo;
38
39
             queue<int> q;
40
             q.push(s);
41
             while(!q.empty())
42
             {
43
                 int u=q.front();
44
                 q.pop();
45
                 inq[u]=0;
46
                 for(int i=0; i<G[u].size(); i++)</pre>
47
                     Edge& e=edge[G[u][i]];
48
49
                     if(e.cap>e.flow && d[e.v]>d[u]+e.cost)
50
51
                        d[e.v]=d[u]+e.cost;
52
                        p[e.v]=G[u][i];
```

```
53
                           a[e.v]=min(a[u], e.cap-e.flow);
 54
                          if(!inq[e.v])
 55
                          {
 56
                              q.push(e.v);
 57
                              inq[e.v]=true;
 58
                          }
 59
                      }
 60
                  }
 61
              if(d[t]==oo)return false;
 62
 63
              flow+=a[t];
 64
               cost+=d[t]*a[t];
 65
               int u=t;
 66
              while(u!=s)
 67
 68
                   edge[p[u]].flow+=a[t];
 69
                  edge[p[u]^1].flow-=a[t];
 70
                  u=edge[p[u]].u;
 71
 72
              return true;
 73
          }
 74
 75
          int Mincost(int s, int t, int& cost)
 76
 77
               int flow=0;
 78
              while(SPFA(s, t, flow, cost))
 79
 80
              return flow;
 81
 82
      } net;
 83
 84
      int ord[55][55], sto[55][55];
 85
 86
      int main()
 87
 88
          int n, m, k;
          while(~scanf("%d%d%d", &n, &m, &k) && n+m+k)
 89
 90
 91
               for(int i=1; i<=n; i++)</pre>
 92
                   for(int j=1; j<=k; j++)</pre>
                      scanf("%d", &ord[i][j]);
 93
 94
               for(int i=1; i<=m; i++)</pre>
                  for(int j=1; j<=k; j++)</pre>
 95
 96
                      scanf("%d", &sto[i][j]);
 97
               int S=0, T=n+m+2;
 98
               int cost=0;
 99
               for(int p=1; p<=k; p++)</pre>
100
               {
101
                  int sum=0;
102
                  net.init(n+m+10);
103
                  for(int i=1; i<=n; i++)</pre>
104
                  {
105
                      net.AddEdge(i, T, ord[i][p], ∅);
106
                      sum+=ord[i][p];
107
                  }
```

```
108
                  for(int i=1; i<=m; i++)</pre>
109
                      net.AddEdge(S, i+n, sto[i][p], ∅);
110
                  for(int i=1; i<=n; i++)</pre>
111
                      for(int j=1; j<=m; j++)</pre>
112
113
                          int x;
114
                          scanf("%d", &x);
115
                          net.AddEdge(n+j, i, oo, x);
116
                  if(~cost && net.Mincost(S, T, cost)<sum)</pre>
117
118
                      cost=-1;
119
              }
120
              printf("%d\n", cost);
121
          }
122
          return 0;
123
      }
124
      Merge sort reverse
  1
  2
      int arr[1000200], tarr[1000200];
  3
      int cnt;
      void merge(int low, int mid, int high)
  5
      {
  6
           int i, j, k;
           for (i = low, j = mid + 1, k = 0; i <= mid && j <= high;)
  7
  8
 9
               if(arr[i] < arr[j])</pre>
 10
                   tarr[k++] = arr[i++];
 11
               else
 12
 13
                   tarr[k++] = arr[j++];
 14
                   cnt += mid - i + 1;
 15
               }
 16
 17
           while(i <= mid) tarr[k++] = arr[i++];</pre>
 18
           while(j <= high) tarr[k++] = arr[j++];</pre>
 19
 20
           for (k = 0; low \leftarrow high; low++, k++)
 21
               arr[low] = tarr[k];
 22
 23
      void mergesort(int low, int high)
 24
 25
           if(low == high) return;
 26
           int mid = (low + high) / 2;
 27
           mergesort(low, mid);
 28
           mergesort(mid + 1, high);
 29
          merge(low, mid, high);
 30
      int main()
 31
 32
 33
           int n;
 34
           scanf("%d", &n);
```

for (int i = 0; i < n; i++)

```
36
             scanf("%d", &arr[i]);
37
38
         cnt = 0;
39
         mergesort(∅, n-1);
40
         printf("%d\n", cnt);
41
         return 0;
42
     }
     prim
 1
 2
     struct Edge
 3
     {
 4
         int u, v, c;
 5
         Edge(){}
 6
         Edge(int u, int v, int c):u(u),v(v),c(c){}
 7
     };
 8
     vector<Edge> G[10020];
 9
     void addedge(int u, int v, int c)
10
         G[u].push_back(Edge(u,v,c));
11
12
         G[v].push_back(Edge(v,u,c));
13
     int n, m;
14
15
     int vis[10020];
16
     int dist[10020];
17
     int prim()
18
     {
19
         int ans = 0;
20
         memset(vis, 0, sizeof(vis));
21
         memset(dist, 0x3f, sizeof(dist));
22
         vis[1] = 1;
23
         int minid, minc;
24
         int now = 1;
25
         for (int t = 1; t < n; t++)
26
         {
27
             for (int i = 0, len = G[now].size(); i < len; i++)
28
29
                 int to = G[now][i].v, c = G[now][i].c;
30
                 if(vis[to] == 1) continue;
31
                 if(dist[to] > c)
32
                     dist[to] = c;
33
34
             minid = -1;
35
             minc = 0x3f3f3f3f;
36
             for (int i = 1; i <= n; i++) if((!vis[i]) && dist[i] < minc)
37
             {
38
                 minid = i;
39
                 minc = dist[i];
40
41
             ans += minc;
42
             vis[minid] = 1;
43
             now = minid;
44
45
         return ans;
```

```
46
47
48
     int main()
49
50
         scanf("%d%d", &n, &m);
51
         int u,v,c;
52
         for (int i = 0; i < m; i++)
53
54
             scanf("%d%d%d", &u,&v,&c);
55
             addedge(u,v,c);
56
57
         printf("%d\n" ,prim());
58
         return 0;
59
     }
60
     Kruskal
 1
 2
 3
     struct Edge
 4
     {
 5
         int u, v, c;
 6
         Edge(){}
 7
         Edge(int u, int v, int c):u(u),v(v),c(c){}
 8
         bool operator < (const Edge &e) const {</pre>
9
             return c < e.c;</pre>
10
         }
11
     };
12
     vector<Edge> ve;
13
     int n, m;
     int R[10020];
14
     // int root(int x)
15
16
     // {
17
     // while(R[x] != x)
18
     //
             x = R[x] = R[R[x]];
     // return R[x];
19
20
     // }
21
     int root(int x)
22
23
         if(R[x] == -1) return x;
         if(R[x] != -1) R[x] = root(R[x]);
24
25
         return R[x];
26
     }
27
     int main()
28
         scanf("%d%d", &n, &m);
29
30
         int u, v, c;
31
         for (int i = 1; i <= m; i++)
32
33
             scanf("%d%d%d", &u,&n,&c);
34
             ve.push_back(Edge(u,n,c));
35
36
         // for (int i = 1; i <= n; i++)
37
             // R[i] = i;
```

```
38
         memset(R, -1, sizeof(R));
39
         sort(ve.begin(), ve.end());
40
         int ans = 0;
41
         int Ru, Rv;
42
         for (int i = 0, len = ve.size(); i < len; i++)</pre>
43
44
              Edge &now = ve[i];
45
             Ru = root(now.u);
46
             Rv = root(now.v);
47
             if(Ru != Rv)
48
             {
49
                  ans += now.c;
50
                  R[Ru] = Rv;
51
              }
52
53
         printf("%d\n", ans);
54
         return 0;
55
     }
     Segment tree
1
2
     #define maxn 100200
3
     #define ll long long
4
     #define lson l, m, rt<<1
5
     #define rson m+1, r, rt<<1|1
6
     using namespace std;
7
     struct SegTree{
8
         11 segsum[maxn<<2];</pre>
9
         11 lazy[maxn<<2];</pre>
10
         void clear()
11
12
             memset(segsum, 0, sizeof(segsum));
13
             memset(lazy, 0, sizeof(lazy));
14
15
         void pushup(int rt)
16
17
             segsum[rt] = segsum[rt << 1] + segsum[rt << 1|1];
18
         void build(int 1, int r, int rt)
19
20
21
             if(1 == r)
22
23
                 scanf("%lld",&segsum[rt]);
24
                 return;
25
             }
26
             int m = (1+r) >> 1;
27
             build(lson);
28
             build(rson);
29
             pushup(rt);
30
         void pushdown(int rt, int m)
31
32
33
             lazy[rt << 1] += lazy[rt];</pre>
34
             lazy[rt << 1 | 1] += lazy[rt];</pre>
```

```
35
             segsum[rt << 1] += lazy[rt] * (m - (m >> 1));
36
             segsum[rt << 1 | 1] += lazy[rt] * (m >> 1);
37
             lazy[rt] = 0;
38
         }
39
         void update(int L, int R, int c, int l, int r, int rt)
40
41
             if(L <= 1 && r <= R)
42
             {
43
                 lazy[rt] += c;
                 segsum[rt] += (r - 1 + 1) * c;
44
45
                 return;
46
             }
47
             if(lazy[rt] != 0)
48
                 pushdown(rt, r - l + 1);
49
             int m = (1 + r) >> 1;
50
             if(L <= m) update(L, R, c, lson);</pre>
51
             if(R > m) update(L, R, c, rson);
52
             pushup(rt);
53
54
         void check(int 1, int r, int rt)
55
56
             printf("l=%d r=%d rt=%d sum=%lld\n", l,r,rt,segsum[rt]);
57
58
             if(1 == r)
59
             {
60
                 return;
61
             }
62
             int m = (1+r)>>1;
63
             check(lson);
64
             check(rson);
65
         }
66
         11 querysum(int L, int R, int l, int r, int rt){
67
             if(L <= 1 && R >= r)
68
             {
69
                 return segsum[rt];
70
             }
71
             if(lazy[rt] != 0)
72
                 pushdown(rt, r - 1 + 1);
73
             int m = (1+r) >> 1;
74
             11 tmp = 0;
75
             if(L <= m) tmp += querysum(L, R, lson);</pre>
             if(R > m) tmp += querysum(L, R, rson);
76
77
             return tmp;
78
79
     }segtree;
80
81
     int main()
82
     {
83
         int n, q;
84
         scanf("%d%d",&n,&q);
85
         segtree.clear();
86
         segtree.build(1, n, 1);
87
         char cmd[2];
88
         int x, y ,z;
89
         while(q--){
```

```
90
              scanf("%s", cmd);
 91
              if(cmd[0] == '0')
 92
              {
 93
                 scanf("%d%d",&x,&y);
 94
                 printf("%lld\n", segtree.querysum(x, y, 1, n, 1));
 95
              }else{
 96
                 scanf("%d%d%d",&x,&y,&z);
 97
                 segtree.update(x, y, z, 1, n, 1);
 98
              }
99
100
          return 0;
101
      }
      Spare Table
  1
  2
      using namespace std;
  3
      struct ST
  4
      {
 5
          int high[maxn][33],low[maxn][33], a[maxn];
 6
          int n;
 7
          int depth;
 8
          void clear()
 9
          {
 10
              n = 0;
 11
              depth = 1;
 12
              memset(high, 0, sizeof(high));
 13
              memset(low, 0, sizeof(low));
 14
              memset(a, ∅, sizeof(a));
 15
 16
          void rmq()
 17
 18
              for(int j = 1; j <= 20; j++)
 19
              for(int i = 1; i <= n; i++)if(i + (1 << j) - 1 <= n){
 20
                 high[i][j] = max(high[i][j - 1], high[i + (1 << (j - 1))][j - 1]);
 21
                 low[i][j] = min(low[i][j - 1], low[i + (1 << (j - 1))][j - 1]);
 22
              }
 23
 24
 25
          void init()
 26
 27
              for(int i = 1; i <= n; i++){
 28
                  scanf("%d",&a[i]);
 29
                 high[i][0] = low[i][0] = a[i];
 30
              }
 31
          }
 32
          int query(int s, int e)
 33
 34
              int k = log2(e - s + 1);
 35
                printf("max : %d , min : %d\n", max(high[s][k], high[e - (1 << k) +
 36
      1][k]), min(low[s][k], low[e - (1 << k) + 1][k]));
 37
              return \max(high[s][k], high[e - (1 << k) + 1][k]) - \min(low[s][k],
 38
      low[e - (1 << k) + 1][k]);
 39
 40
          void check()
```

```
41
         {
42
             for(int j = 0; j <= 3; j++)
43
                 for(int i = 1; i <= n; i++)
44
                    printf("%d%c", low[i][j], i == n? '\n':' ');
45
             printf("\n\n");
46
         }
47
     }st;
48
     int main()
49
50
         st.clear();
51
         int m;
52
         scanf("%d%d", &st.n, &m);
53
         st.init();
54
         st.rmq();
55
           st.check();
     //
56
         int s, e;
57
         while(m--)
58
         {
59
             scanf("%d%d",&s,&e);
60
             printf("%d\n",st.query(s, e));
61
62
         return 0;
63
     }
     tarjan
1
2
     using namespace std;
     vector<int>v[maxn];
3
4
     bool instack[maxn];
5
     int dfn[maxn], low[maxn];
6
     int n, m;
7
     int depth, strongcnt;
8
     int belong[maxn], strongsize[maxn];
9
     int stk[maxn], top;
10
     int inde[maxn], outde[maxn];
11
12
13
     void clear(){
14
         memset(dfn, 0, sizeof(dfn));
15
         memset(low, 0, sizeof(low));
16
         memset(instack, 0, sizeof(instack));
         memset(belong, 0, sizeof(belong));
17
18
         memset(strongsize, 0, sizeof(strongsize));
19
         memset(inde, 0, sizeof(inde));
20
         memset(outde, 0, sizeof(outde));
21
         depth = 0;
22
         strongcnt = 0;
23
         for (int i = 1; i <= n; ++i)
24
25
             v[i].clear();
26
         }
27
28
29
     void tarjan(int u){
```

```
30
         dfn[u] = low[u] = ++depth;
         instack[u] = true;
31
32
         stk[++top] = u;
33
         int to;
34
         for (int i = 0; i < v[u].size(); ++i)</pre>
35
36
              to = v[u][i];
37
              if(!dfn[to]){
38
                  tarjan(to);
39
                  low[u] = min(low[to], low[u]);
40
              }else if (instack[to])
41
42
                  //dfn[i] not low[i];
43
                  low[u] = min(low[u], dfn[to]);
44
              }
45
46
         if (dfn[u] == low[u])
47
48
49
              strongcnt++;
50
              do
51
              {
52
                  to = stk[top--];
53
                  instack[to] = false;
54
                  belong[to] = strongcnt;
55
                  strongsize[strongcnt]++;
56
              } while (to != u);
57
         }
58
59
     int main()
60
61
         for (int i = 1; i <= n; ++i)
62
63
              if (!dfn[i])
64
              {
65
                  tarjan(i);
66
              }
67
         }
68
         int to;
69
         for(int i = 1; i <= n; i++)
70
71
              for (int j = 0; j < v[i].size(); ++j)</pre>
72
                  to = v[i][j];
73
74
                  if(belong[i] != belong[to]){
75
                      outde[belong[i]]++;
76
                      inde[belong[to]]++;
77
                  }
78
              }
79
         }
80
         return 0;
81
     }
```

## Trie

1

```
2
     struct Trie
3
     {
4
         struct Node
5
6
             bool end;
7
             int id;
8
             Node *next[26];
9
10
         Node *head;
11
         void clear()
12
         {
13
             head = new Node();
14
         }
15
16
         void insert(char *s, int id)
17
         {
18
             int len = strlen(s);
19
             Node *now = head;
20
             for (int i = 0; i < len; i++)
21
22
                 int x = s[i] - 'a';
23
                 if (now->next[x] == NULL)
24
25
                     now->next[x] = new Node();
26
                     now->next[x]->end = false;
                     memset(now->next[x]->next, 0, sizeof(now->next[x]->next));
27
28
                 }
29
                 now = now->next[x];
                 if (i == len - 1)
30
31
32
                     now->end = true;
33
                     now->id = id;
34
             }
35
36
37
         int query(char *s)
38
39
             int len = strlen(s);
             Node *now = head;
40
41
             for (int i = 0; i < len; i++)
42
43
                 int x = s[i] - 'a';
44
                 if (now->next[x] == NULL)
45
                     return false;
46
                 now = now->next[x];
47
                 if (i == len - 1){
48
                     if(now->end) return now->id;
49
                     else return 0;
50
                 }
51
             }
52
             return 0;
53
         }
54
     };
55
56
     const int maxm = 10000000;
```

```
57
      struct Trie
 58
      {
 59
          struct Node
 60
 61
              bool end;
 62
              int id;
 63
              int next[26];
 64
          }node[maxm];
 65
          int head, tot;
 66
          void clear()
 67
          {
 68
              head = 0;
 69
              memset(node[head].next, -1, sizeof(node[head].next));
 70
              tot = 0;
 71
 72
          void insert(char *s, int id)
 73
 74
              int len = strlen(s);
 75
              int nowid = head;
 76
              for (int i = 0; i < len; i++)
 77
 78
                  Node& now = node[nowid];
 79
                  int x = s[i] - 'a';
 80
                  if (now.next[x] == -1)
 81
                  {
                      now.next[x] = ++tot;
 82
 83
                      node[tot].end = false;
                      memset(node[tot].next, -1, sizeof(node[tot].next));
 84
 85
                  }
 86
                  nowid = now.next[x];
 87
                  if (i == len - 1)
 88
 89
                      node[nowid].end = true;
 90
                      node[nowid].id = id;
 91
                  }
 92
              }
 93
 94
          int query(char *s)
 95
 96
              int len = strlen(s);
 97
              int nowid = head;
 98
              for (int i = 0; i < len; i++)
99
100
                  Node& now = node[nowid];
101
                  int x = s[i] - 'a';
102
                  if (now.next[x] == -1)
                      return false;
103
104
                  nowid = now.next[x];
105
                  if (i == len - 1){
106
                      if(node[nowid].end) return node[nowid].id;
107
                      else return 0;
108
                  }
109
              }
110
              return 0;
111
          }
```

```
4 points on a plane
 1
 2
     using namespace std;
     struct Point3 {
 3
 4
         double x, y, z;
 5
         Point3 operator - ( Point3 & p ) {
 6
             Point3 ans;
 7
             ans.x = this -> x - p.x;
             ans.y = this->y - p.y;
 8
9
             ans.z = this \rightarrow z - p.z;
10
             return ans;
11
         }
12
     };
13
     Point3 operator * ( const Point3 & a, const Point3 & b ) {
14
         Point3 ans;
15
         ans.x = a.y * b.z - a.z * b.y;
16
         ans.y = a.z * b.x - a.x * b.z;
17
         ans.z = a.x * b.y - a.y * b.x;
18
         return ans;
19
20
     double dot( const Point3 & a, const Point3 & b ) {
21
         return a.x * b.x + a.y * b.y + a.z * b.z;
22
     }
23
     int main() {
24
         Point3 p[4];
25
         int T;
26
         cin >> T;
27
         while(T--)
28
29
             for( int i = 0; i < 4; ++i ) scanf( "%lf%lf%lf", &p[i].x, &p[i].y,
30
     &p[i].z);
             puts( dot( p[3] - p[0], (p[2] - p[0])*(p[1] - p[0])) == 0.0 ? "Yes" :
31
     "No" );
32
33
         }
34
         return 0;
35
     BIT
 1
 2
     struct BIT{
 3
         int c[maxn];
 4
         int n;
 5
         void clear(int n){
 6
             memset(c, 0, sizeof(c));
 7
             this->n = n;
 8
         }
9
         inline int lowbit(int x){
10
             return x \& (-x);
11
12
         void add(int pos, int delta){
13
             printf("n = %d\n", n);
14
             while(pos < maxn){</pre>
```

**}**;

```
15
                c[pos] += delta;
16
                pos += lowbit(pos);
17
             }
18
         }
19
         int getsum(int pos){
20
             int ans = 0;
21
             while(pos > ∅){
22
                ans += c[pos];
23
                pos -= lowbit(pos);
24
             }
25
             return ans;
26
27
     }bit;
     Cantor
1
2
     /*
3
      *
        康拓展开
4
        元素个数 len
5
      * 0-based count
6
      * last edit : 2015/9/25
7
      */
8
9
     int fact[10] = {1,1,2,6,24,120,720,5040,40320,362880};
10
     int cantor(int* a,int len)
11
12
         int ret = 0;
13
         for(int i = 0; i < len; i++)
14
15
             int tmp = 0;
16
             for(int j = i+1; j < len; j++)if(a[i] > a[j]) tmp++;
17
             ret += tmp * fact[len-i-1];
18
19
         return ret;
20
     }
21
22
     void cantorrev(int* a,int d, int len)
23
24
         int vis[10] = \{0\}, tmp, tt;
25
         for(int i = 0; i < len; i++)</pre>
26
27
             tmp = d / fact[len-i-1];
28
             d %= fact[len-i-1];
29
             //the min
30
             tt = 1;
31
             while(tmp || vis[tt])
32
33
                 if(vis[tt] == 0)
34
                     tmp--;
35
                 tt++;
36
             }
37
             vis[tt] = 1;
38
             a[i] = tt;
39
         }
```

```
40 }
```

Dijstra

```
2
     //v: node id
     //1: length from start
3
4
     //c: mincost
5
6
         int v, 1, c;
7
         Node(){}
8
         Node(int v, int 1, int c):v(v),1(1),c(c){}
9
         bool operator < (const Node &a) const
10
         //priority queue 的优先级和 < 相反
11
         {
12
             if(1 == a.1) return c > a.c;
13
             return 1 > a.l;
14
15
     };
16
     vector<Edge>G[maxn];
17
     priority_queue<Node>pq;
18
     int dist[maxn],cost[maxn],vis[maxn],tot;
19
     void add_edge(int u, int v, int l, int c)
20
     {
21
         G[u].push_back(Edge(u, v, 1, c));
22
23
     PII dijstra(int s, int d)
24
     //start s, dest d
25
26
         memset(dist, INF, sizeof(dist));
27
         memset(cost, INF, sizeof(cost));
28
         memset(vis, ∅, sizeof(vis));
29
         while(!pq.empty()) pq.pop();
30
         pq.push(Node(s, 0, 0));
31
         while(!pq.empty())
32
         {
33
             const Node nd = pq.top();
34
             pq.pop();
35
             if(vis[nd.v]) continue;
36
             vis[nd.v] = true;
37
             dist[nd.v] = nd.l;
             cost[nd.v] = nd.c;
38
39
             if(nd.v == d) return make_pair(dist[d], cost[d]);
40
             for(int i = 0, len = G[nd.v].size(); i < len; i++)</pre>
41
                 Edge& e = G[nd.v][i];
42
43
                 if(!vis[e.v])
44
                 {
45
                     pq.push(Node(e.v, nd.1 + e.1, nd.c+e.c));
46
                 }
47
             }
48
49
         //dist[d]: shortest distance
50
         //cost[d]: mincost
51
         return make_pair(dist[d], cost[d]);
```

```
52 }
```

## Dinic

```
2
     #define maxn 320
     using namespace std;
 4
     int G[maxn][maxn], layer[maxn];
 5
     int m, n;
     bool vis[maxn];
7
     bool countLayer()
8
9
         queue<int>q;
10
         memset(layer, 0xff, sizeof(layer));
11
         layer[1] = 0;q.push(1);
12
13
         while(!q.empty())
14
             int v = q.front();q.pop();
15
16
             for(int j = 1; j <= n; j++)</pre>
17
                 if(G[v][j] > 0 && layer[j] == -1)
18
                 {
19
                     layer[j] = layer[v] + 1;
20
                     if(j == n) return true;
21
                     else q.push(j);
22
                 }
23
24
         return false;
25
     }
26
     int Dinic()
27
28
         int i;
29
         int maxflow = 0;
30
         deque<int> q;
31
         while(countLayer())
32
         {
33
             q.push_back(1);
34
             memset(vis, ∅, sizeof(vis));
35
             vis[1] = true;
36
             while(!q.empty())
37
38
                 int nd = q.back();
39
                 if(nd == n)
40
                 {
41
                     int minc = 1000000000;
42
                     int minstart;
43
                     for(i = 1; i < q.size();i++)</pre>
44
45
                         int vs = q[i-1];
46
                         int ve = q[i];
47
                         if(G[vs][ve] > 0 && minc > G[vs][ve])
48
49
                              minc = G[vs][ve];
50
                             minstart = vs;
51
                          }
```

```
52
                     }
53
                     maxflow += minc;
54
                      for(i = 1; i < q.size(); i++)</pre>
55
56
                          int vs = q[i-1];
57
                          int ve = q[i];
58
                         G[vs][ve] -= minc;
59
                         G[ve][vs] += minc;
60
                     while(!q.empty() && q.back() != minstart)
61
62
63
                         vis[q.back()] = false;
64
                          q.pop_back();
65
                      }
66
                 }else{
67
                      for(i = 1; i <= n; i++)
68
                          if(G[nd][i] > 0 && layer[i] == layer[nd] + 1 && !vis[i])
69
70
                              vis[i] = true;
71
                              q.push_back(i);
72
                              break;
73
74
                      if(i > n) q.pop_back();
75
                 }
             }
76
77
78
         }
79
80
         return maxflow;
81
     }
82
83
84
     int main()
85
86
         while(scanf("%d%d", &m, &n) != EOF)
87
         {
88
             int s, e, c;
             memset(G, 0, sizeof(G));
89
90
             for(int i = 0; i < m; i++)</pre>
91
92
                 scanf("%d%d%d",&s,&e,&c);
93
                 G[s][e] += c;
94
95
             printf("%d\n", Dinic());
96
97
         return 0;
98
     }
     floyed
 1
 2
     const int INF=10000000;
     int dist[maxn][maxn], G[maxn][maxn];
 3
 4
     int n, m, num, minc;
 5
     void floyd()
```

```
6
     {
7
         minc=INF;
8
         // 求最小环
9
         for(int k=1; k<=n; k++)</pre>
10
             for(int i=1; i<k; i++)</pre>
11
12
                for(int j=i+1; j<k; j++)</pre>
13
14
                    int ans=dist[i][j]+G[i][k]+G[k][j];
15
                    if(ans<minc) //找到最优解
16
17
                        minc=ans;
18
19
20
             for(int i=1; i<=n; i++)</pre>
21
                for(int j=1; j<=n; j++)</pre>
22
23
                    if(dist[i][j]>dist[i][k]+dist[k][j])
24
25
                        dist[i][j]=dist[i][k]+dist[k][j];
26
27
                }
28
         }
29
     }
     Wythoff
1
2
     //Wythoff Game
     //A first
3
     //B second
5
     //当 n 过大时需要用高精度处理,和精确的黄金比例数
6
     int main()
7
     {
8
         int T;
9
         scanf("%d", &T);
10
11
         while(T--)
12
13
             int a, b;
             scanf("%d%d", &a, &b);
14
15
             if(a > b) swap(a, b);
16
17
             int k = b - a;
18
             if(a == (int)((k)*(1+sqrt(5.0))/2.0)) cout << "B" << endl;
19
             else cout << "A" << endl;</pre>
20
21
22
         return 0;
23
     }
     hangary
2
     struct Edge
3
     {
```

```
4
        int from, to, weight;
5
        Edge(int f, int t, int w):from(f), to(t), weight(w) {}
6
     };
7
     vector<Edge> G[__maxNodes]; /* G[i] 存储项点 i 出发的边的编号 */
8
     int matching[ maxNodes]; /* 存储求解结果 */
9
     int check[__maxNodes];
10
     int n, m, sum;
11
     /*DFS*/
12
     bool dfs(int u)
13
14
        for (int i = 0; i < G[u].size(); i++) {
15
            int v = G[u][i].to;
16
                               // 要求不在交替路中
            if (!check[v]) {
17
               check[v] = true; // 放入交替路
18
               if (matching[v] == -1 || dfs(matching[v])) {
19
                   // 如果是未盖点,说明交替路为增广路,则交换路径,并返回成功
20
                   matching[v] = u;
21
                   matching[u] = v;
22
                   return true;
23
               }
24
            }
25
26
        return false; // 不存在增广路,返回失败
27
28
     int hungarian()
29
30
        int ans = 0;
31
        memset(matching, -1, sizeof(matching));
32
        for (int u=1; u <= n; ++u) {
33
            if (matching[u] == -1) {
34
               memset(check, 0, sizeof(check));
35
               if (dfs(u))
36
                   ++ans;
37
            }
38
        }
39
        return ans;
40
     }
    josephus
1
2
     //編號從0開始,也就是說如果編號從1開始結果要加1
3
     int josephus(int n, int k) { //非遞回版本
4
        int s = 0;
5
        for (int i = 2; i <= n; i++)
6
            s = (s + k) \% i;
7
        return s;
8
9
     int josephus_recursion(int n, int k) { //遞回版本
10
        return n > 1? (josephus recursion(n - 1, k) + k) % n : 0;
11
12
     int main() {
13
        for (int i = 1; i <= 100; i++)
14
            cout << i << ' ' << josephus(i, 5) << endl;</pre>
```

```
15
         return 0;
16
     }
     KMP
1
2
     char src[maxn],substring[maxn];
3
     int nxt[maxn];
4
     void get_nxt(char* substring)
5
6
         int substring_len = strlen(substring);
7
         memset(nxt, 0, sizeof(nxt));
8
         nxt[0] = -1;
9
         int j = -1;
10
         for(int i = 1; i < substring_len; i++)</pre>
11
12
             while(j > -1 && substring[i] != substring[j + 1])
13
                 j = nxt[j];
14
             if(substring[j+1] == substring[i])
15
                 j = j + 1;
16
             nxt[i] = j;
17
         }
18
     }
19
20
     //process src & substring to get the position
21
     int kmp(char* src, char* substring)
22
     {
23
         int j = -1;
24
         int ans = 0;
25
         int substring_len = strlen(substring);
26
         int src_len = strlen(src);
27
         for(int i = 0; i < src_len; i++)</pre>
28
29
             while(j > -1 \&\& src[i] != substring[j + 1])
30
                 j = nxt[j];
31
             if(src[i] == substring[j + 1])
32
                 j++;
33
             if(j == substring_len -1)
34
35
                 ans ++;
36
                 printf("From position %d to position %d\n", i + 2 - substring_len,
37
     i+1);
38
                 j = nxt[j];
39
             }
40
41
         return ans;
42
     }
43
     Manacher
1
2
     const int maxn = 2100000;
3
4
     /*
5
      * 求最长回文字串
```

```
* 0(n);
6
      */
7
8
9
     char Ma[maxn*2];
10
     int Mp[maxn*2];
11
     char s[maxn];
12
13
     void manacher(int len)
14
15
         int 1 = 0;
         Ma[1++] = '$';
16
17
         Ma[1++] = '#';
         for(int i = 0; i <len; i++)</pre>
18
19
20
             Ma[l++] = s[i];
21
             Ma[1++] = '#';
22
23
         Ma[1] = 0;
24
         int mx = 0, id = 0;
25
         for(int i = 0; i < 1; i++)</pre>
26
27
             Mp[i] = mx>i? min(Mp[2*id-i], mx-i):1;
28
             while(Ma[i+Mp[i]] == Ma[i-Mp[i]]) Mp[i]++;
29
             if(i+Mp[i]>mx)
30
31
                 mx = i + Mp[i];
32
                 id = i;
33
34
         }
35
36
     int main()
37
         while(scanf("%s", s) != EOF)
38
39
         {
40
             scanf("%s", s);
41
42
             int len = strlen(s);
43
             manacher(len);
             int ans = 0;
44
45
             for(int i = 0; i < len*2+2; i++)</pre>
46
47
                 ans = max(ans, Mp[i]-1);
                 // printf("%d ", Mp[i]);
48
49
50
             printf("%d\n", ans);
51
52
         return 0;
53
     }
     Matrix pow
1
2
     #define maxn 30
3
     using namespace std;
     typedef long long LL;
```

```
5
6
     struct Matrix{
7
         LL m[maxn][maxn];
8
         Matrix(){memset(m, 0, sizeof(m));}
9
     };
10
     typedef Matrix matrix;
11
     LL Mod;
12
     int n;
13
     matrix operator* (matrix A, matrix B)
14
15
         matrix C;
16
         for(int i = 0; i < n; i++)
17
             for(int j = 0; j < n; j++)
18
19
                 C.m[i][j] = OLL;
20
                 for(int k = 0; k < n; k++)
21
                     C.m[i][j] += A.m[i][k]*B.m[k][j];
22
                 C.m[i][j] \% = Mod;
23
24
         return C;
25
26
     matrix operator+ (matrix A, matrix B)
27
28
         for(int i = 0; i < n; i++)</pre>
29
             for(int j = 0; j < n; j++)
30
                 A.m[i][j] = (A.m[i][j] + B.m[i][j]) % Mod;
31
         return A;
32
33
     matrix operator% (matrix A, LL m)
34
35
         for(int i = 0; i < n; i++)</pre>
36
             for(int j = 0; j < n; j++)
37
                 A.m[i][j] %= m;
38
         return A;
39
     }
40
     matrix matrix pow(int k, matrix M)
41
42
         if(k == 1) return M;
         matrix ans;
43
44
         memset(ans.m, 0, sizeof(ans.m));
45
         for(int i = 0; i < n; i++)
46
             ans.m[i][i] = 1LL;
47
         while(k)
48
         {
49
             if(k&1)
50
             {
51
                 ans = ans * M;
52
                 k--;
53
             }
54
             else
55
56
                 k /= 2;
57
                 M = M * M;
58
             }
59
         }
```

```
60
         return ans;
61
62
     matrix sum(matrix ma, int k)
63
64
         matrix ret;
65
         if(k == 1) return ma;
66
         if(k&1)
67
68
             matrix tmp = sum(ma, k/2) % Mod, tmp1 = matrix pow(k/2+1, ma) % Mod;
             ret = (tmp + tmp1 + tmp * tmp1) % Mod;
69
70
         }
71
         else
72
         {
73
             matrix tmp = sum(ma, k/2) % Mod, tmp1 = matrix_pow(k/2, ma) % Mod;
74
             ret = (tmp + tmp * tmp1) % Mod;
75
76
         return ret;
77
78
     int main()
79
     {
         int k;
80
81
         matrix A;
82
         scanf("%d%d%1ld", &n, &k, &Mod);
         for(int i = 0; i < n; i++)</pre>
83
             for(int j = 0; j < n; j++)
84
85
                 scanf("%11d", &A.m[i][j]);
86
         A = sum(A, k);
87
         for(int i = 0; i < n; i++)</pre>
88
             for(int j = 0; j < n; j++)
89
90
                 printf("%11d%c", A.m[i][j],(j == n-1)? '\n':' ');
91
             }
92
         return 0;
93
     }
     Math
1
2
3
4
         math templates
5
         created by poore : 2015/09/14
6
         last edit : 2015/10/19
7
8
         Contents:
9
10
      * GCD
         ext GCD
11
      * 筛法求素数
12
      * slow_mul
13
14
         linear_mod_equation 一元线性方程组求解
15
         pow mod
16
      * Lucas Lehmer 判定梅森素数
17
      * miller robbin 素数判定
18
      * pollard rho 返回一个随机的约数
```

```
* calc 寻找最小的约数
19
     * mega mod(n)解 n 个一元线性同于方程组
20
21
      * CRT() 中国剩余定理
      * 欧拉函数
22
23
      * 整数拆分
24
      * Stirling's approximation
25
26
27
28
     #include <cstdio>
29
     #include <iostream>
30
     #include <cmath>
31
     #include <cstring>
32
     #include <cstdlib>
33
     #define INF 0x3f3f3f3f
34
     typedef long long LL;
35
36
37
     using namespace std;
38
39
     const int MOD = 1e9+7;
40
41
42
     //GCD
43
     LL GCD(LL a, LL b)
44
     //递归
45
46
         if(a > b) swap(a, b);
47
         LL r = a \% b;
48
         if(r == 0) return b;
49
         return GCD(b, r);
50
     }
51
     LL gcd(LL M, LL N)
52
53
     //非递归
54
     {
55
         LL Rem;
56
        while(N > 0)
57
58
            Rem = M \% N;
            M = N;
59
60
            N = Rem;
61
         }
62
         return M;
63
     }
64
     void EXT_GCD(LL a, LL b, LL &d, LL &x, LL &y)
65
66
     //a , b 任意
67
     {
68
         if(!b) {d = a, x = 1, y = 0;}
69
         else {EXT_GCD(b, a % b, d, y, x), y -= x * (a / b);}
70
71
72
     //递归求逆元
```

```
73
      //p, x 互质
 74
      LL inv(LL x, LL m)
 75
 76
          if (x == 1) return x;
 77
          return inv(m \% x, m)*(m - m / x) \% m;
 78
      }
 79
 80
 81
      ll inv(LL a, LL c)
 82
      // 用扩展欧几里得求逆元
83
      // 要求 a, c 互质
      // 如果没有逆元返回 -1
 84
 85
 86
          LL d, x, y;
 87
          EXT_GCD(a, c, d, x, y);
 88
          return d == 1 ? (x + c) % c : -1;
 89
 90
      LL ext gcd(LL a, LL b, LL& x, LL& y)
 91
      // a >= 0, b > 0
 92
 93
          LL x1=0LL, y1=1LL, x0=1LL, y0=0LL;
94
          LL r = (a\%b + b) \% b;
 95
          LL q = (a-r) / b;
          x = 0LL, y = 1LL;
96
 97
          while(r)
98
99
              x=x0-q*x1;y=y0-q*y1;
100
              x0=x1;y0=y1;
101
              x1=x;y1=y;
102
              a=b;b=r;
103
              r=a%b;
104
              q=(a-r)/b;
105
106
          return b;
107
108
109
      const int maxn = 100020;
110
      bool isprime[maxn];
111
      LL prime[maxn];
112
      int doprime(LL N)
113
      //prime[] 储存质数。1-based index;
114
115
          int nprime = 0;
116
          memset(isprime, true, sizeof(isprime));
117
          isprime[1] = false;
          for(LL i = 2; i <= N; i++)
118
119
120
              if(isprime[i])
121
122
                  prime[++nprime] = i;
123
                  for(LL j = i*i; j <= N; j+=i)</pre>
124
                      isprime[j] = false;
125
              }
126
          }
```

```
127
          return nprime;
128
      }
129
130
131
      LL slow_mul(LL a, LL b, LL p)
132
          // cout << a << " " << b << endl;
133
134
          LL ret = ∅;
          while(b) {
135
              if(b & 1) ret = (ret + a) % p;
136
137
              a = (a + a) \% p;
138
              b >>= 1;
139
140
          return ret % p;
141
      }
142
143
      LL pow_mod(LL a, LL b, LL p)
144
      //快速幂
145
      {
146
          LL ret = 1;
147
          while(b) {
148
              if(b & 1) ret = (ret*a)%p;
149
              a = (a*a)%p;
150
              b >>= 1;
151
152
          return ret%p;
153
      }
154
155
      //判断Mp = 2^p-1 是否为梅森素数
156
157
      bool lucas_lehmer(int p)
158
          if(p == 2) return true;
159
160
          LL m = (1LL << p)-1LL, tmp = 4LL;
          for(int i = 0; i < p-2; i++)
161
162
          {
              tmp = (slow mul(tmp, tmp, m) - 2 + m) % m;
163
164
          if(tmp == 0LL) return true;
165
          return false;
166
167
      }
168
169
      LL witness(LL a, LL b, LL c)
170
171
          if(b==0)return 1;
172
          LL x,y,t=0;
173
          while((b\&1)==0)
174
              b>>=1, t++;
175
          y=x=pow_mod(a,b,c);
176
          //二次探测
177
          while(t--)
178
          {
179
              y=slow_mul(x,x,c);
180
              if(y==1 \&\& x!=1 \&\& x!=c-1)
181
                  return false;
```

```
182
              x=y;
183
184
          return y==1;
185
186
      bool miller_rabin(LL n)
187
      //..质数为true, 非质数为false...
188
189
          if(n==2)return true;
190
          if(n<2 || (n&1)==0)return false;</pre>
191
          for(int i=0;i<3;i++)</pre>
              if(witness(rand()%(n-2)+2,n-1,n)!=1)
192
193
                  return false;
194
          return true;
195
      }
196
197
      LL ans = INF;
198
199
      LL pollard rho(LL n,LL c)
200
      //..随机返回一个 n 的约数..
201
202
          if(n\%2==0)return 2;
203
          LL i=1, k=2, x=rand()%n, y=x, d;
204
          while(1){
205
              i++;
206
              x=(slow_mul(x,x,n)+c)%n;
207
              d=gcd(y-x,n);
208
              if(d==n)return n;
209
              if(d!=n && d>1)return d;
210
              if(i==k) y=x,k<<=1;
211
          }
212
      void calc(LL n,LL c=240)
213
      //寻找最小的约数...
214
215
216
          if(n==1)return;
217
          if(miller_rabin(n)){
218
              ans=min(ans,n);
219
              return;
220
          LL k=n;
221
          while(k==n)k=pollard_rho(n,c--);
222
223
          calc(k,c),calc(n/k,c);
224
      }
225
226
      vector<LL> linear_mod_equation(LL a, LL b, LL n)
227
228
      //线性方程求解
229
      //ax = b \pmod{n}
230
231
          LL x, y, d;
232
          vector<LL> sol;
233
          sol.clear();
234
          EXT\_GCD(a, n, d, x, y);
235
          if( b%d ) d = 0;
```

```
236
          else
237
          {
238
              sol.push back(x * (b/d) % n);
239
              for (int i = 1; i < d; i++)
240
                  sol.push_back((sol[i-1] + n/d + n) % n);
241
242
          return sol;
243
244
      LL mega mod(int n)
245
      //解 n 个一元线性同于方程组
246
      //x \equiv r \pmod{a}
247
      //求x
248
249
          LL a1, a2, r1, r2, d, c, x, y, x0,s;
250
          bool flag = true;
251
          scanf("%lld%lld", &a1, &r1);
252
          for(int i = 1; i < n; i++)</pre>
253
          {
254
              scanf("%11d%11d", &a2, &r2);
255
              if(!flag) continue;
256
              c = r2 - r1;
257
              EXT_GCD(a1, a2, d, x, y);
258
              if(c%d!=0)
259
260
                  flag = false;
261
                  continue;
              }
262
263
              x0 = x*c/d;
264
              s = a2/d;
265
              x0 = (x0\%s+s)\%s;
266
              r1=r1+x0*a1;
267
              a1=a1*a2/d;
268
269
          if(flag) return r1;
270
          else return -1LL;
271
272
273
      LL CRT(LL *a, LL *m, int n)
      //中国剩余定理
274
275
      //x \equiv a[i] \pmod{m[i]}
276
      //m[i] is coprime
277
278
          LL M = 1, Mi, x0, y0, d, ret = 0;
279
          for(int i = 0; i < n; i++)
              M *= m[i];
280
281
          for(int i = 0; i < n; i++)</pre>
282
283
              Mi = M/m[i];
284
              EXT_GCD(Mi, m[i], d, x0, y0);
285
              ret = (ret+Mi*x0*a[i]) % M;
286
287
          if(ret < 0)
288
              ret += M;
289
          return ret;
290
      }
```

```
291
292
      //欧拉函数
293
      LL calphi(LL n)
294
295
          LL res = n;
296
          for(LL i = 2; i*i <= n; i++)if(n%i==0)</pre>
297
298
              res -= res/i;
299
              while(n%i==0) n/=i;
300
301
          if(n > 1)
302
              res -= res/n;
303
          return res;
304
305
306
      //欧拉函数预处理
307
      int phi[maxn];
308
      void getpthi(int n)
309
      {
310
          memset(phi, 0, sizeof(phi));
311
          phi[1] = 1;
312
          for(int i = 2; i <= n; i++)if(!phi[i])</pre>
313
          {
314
              for(int j = i; j <= n; j+=i)</pre>
315
              {
316
                  if(!phi[j])
317
                      phi[j] = j;
318
                  phi[j] = phi[j]/i*(i-1);
319
              }
320
          }
321
      }
322
323
324
325
      //把整数 n 拆分成几个数相加的形式, 问有多少种拆分方法
326
      int dp[maxn];
327
      void splitint()
328
329
          memset(dp, ∅, sizeof(dp));
330
          dp[0]=1;
331
          for(int i = 1; i <= maxn; i++)</pre>
332
333
              for(int j = 1, r = 1; i - (3*j*j-j)/2 >= 0; j++, r*=-1)
334
335
                  dp[i] += dp[i-(3*j*j-j)/2]*r;
336
                  dp[i] %= MOD;
337
                  dp[i] = (dp[i] + MOD) \% MOD;
338
                  if(i-(3*j*j+j)/2 >= 0)
339
                      dp[i] += dp[i-(3*j*j+j)/2] *r;
340
341
                      dp[i] %= MOD;
342
                      dp[i] = (dp[i] + MOD)%MOD;
343
                  }
344
              }
345
          }
```

```
346
347
348
      //Stirling N的阶乘的长度
      const double PI=3.1415926;
349
350
      int main()
351
352
         int t,n,a;
         while(scanf("%d",&n)!=EOF)
353
354
355
             a=(int)((0.5*log(2*PI*n)+n*log(n)-n)/log(10));
356
             printf("%d\n",a+1);
357
358
         return 0;
359
      }
360
361
362
363
      /*
364
365
      Something Tasteless
366
367
      1.素数个数估算
368
          设PI(x) 为小于 x 的素数的个数
          当 x 足够大时, PI(x) = x/lnx;
369
370
      2.n! 的素因子分解中的素数 p 的次数 为
371
         [n/p] + [n/(p^2)] + [n/(p^3)] + ... +
372
373
374
375
      3.
376
377
      */
378
```