Serenity

Contents

1 Template 2

2 Unionfind 4

3 Getid 4

4 SegmentTree 4

5 BIT 5

6 BIT2D 5

7 LIS 5

8 StructVec 6

9 LinePointDist 6

10 SegmentsIntersect 6

11 AreaOfPolygon 6

12 StructLine 7

13 GetMidLine 7

14 ReflectPoint 7

15 ConvexHull 7

16 Bpm 8

17 Hopcroft 8

18 Djikstra-set 9

19 DijkstraPrqueue 9

20 ArticulationPoint 10

21 StronglyConnectedC 10

22 Kruskal 11

23 Maxflow 11

24 MaxflowDinic 12

25 MincostMaxflow 13

26 BigNumInt 14

27 BsearchLE 16

28 BsearchGE 17

29 PermIndex 17

30 Geodistance **17**

31 Matrixmul **17**

32 Heron **18**

33 Genprimes **18**

34 Choose **18**

35 EuclidExtended **18**

36 Isprime2MillerRabin **19**

37 EulerPhi **19**

38 Euler2Phi **19**

39 SGcdCongrModinv **19**

40 Sterling1 **20**

41 Sterling2 **20**

42 Totient **20**

43 SieveBits **21**

44 KmpExplained **21**

1 Template

```
#include <algorithm>
#include <iostream>
#include <cassert>
#include <climits>
#include <utility>
#include <sstream>
#include <iomanip>
#include <cstdlib>
#include <cstring>
#include <complex>
#include <cctype>
#include <cstdio>
#include <bitset>
#include <vector>
#include <string>
#include <queue>
#include <stack>
#include <cmath>
#include <map>
#include <set>
using namespace std;
#define SET(t,v) memset((t), (v), sizeof(t))
typedef long long LL;
typedef long double LD;
#define sz size()
#define mp make_pair
#define pb push_back
#define x first
#define y second
```

```
#define Rep(i, n) for(int i = 0; i < (n); ++i)
#define Repe(i, n) for(int i = 0; i <= (n); ++i)
#define For(i,a,b) for(int i=(a); i<(b); i++)
#define Fore(i,a,b) for(int i=(a); i<=(b); i++)
#define Ford(i,a,b) for(int i=(a); i>=(b); i--)
#define Dhelper(X) " " << #X << "='" << (X) << "'"
#define D(X) {cerr << __LINE__ << ":" << Dhelper(X) << endl;}
#define D2(X,Y) {cerr << __LINE__ << ":" << Dhelper(X) << Dhelper(Y) << endl;}
#define Dv(X) {cerr << __LINE__ << ":  " << #X << " = {"; for(int __i=0;__i<(X).size();__i++) \
    cerr<<" "<<(X)[__i];cerr<<" }"<<endl;}
#define Da(X,n) {cerr << __LINE__ << ":  " << #X << " = {"; for(int __i=0;__i<(n);__i++) \
    cerr<<" "<<(X)[__i];cerr<<" }"<<endl;}
template<class T> inline void checkmin(T &a, const T &b) { if(a > b) a = b; }
template<class T> inline void checkmax(T &a, const T &b) { if(a < b) a = b; }
typedef pair<int,int> PII;
typedef pair<LL,int> PLI;
typedef pair<LL,LL> PLL;
template<class T> inline T gcd(T a,T b) {if(a<0)return gcd(-a,b);if(b<0)return gcd(a,-b);
    return (b==0)?a:gcd(b,a%b);}
template<class T> T lcm(const T &a,const T &b) { return a*(b/gcd(a,b)); }
template<class T> inline T sqr(T X) { return X * X; }
const long double PI=acos(-1.0L);
#define ALL(c) (c).begin(), (c).end()
#define RALL(c) (c).rbegin(), (c).rend()
#define SORT(c) sort(ALL(c))
#define RSORT(c) sort(RALL(c))
#define UNIQUE(c) (c).resize(unique(ALL(c))-(c).begin())
typedef vector<int> VI;
typedef vector<LD> VLD;
typedef vector<LL> VLL;
typedef vector<PII> VPII;
typedef vector<vector<int> > VVI;
typedef vector<string> VS;
#define Foreach(it,X) for(__typeof((X).begin())it=(X).begin();it!=(X).end();++it)
template<class T> inline T iabs(T a) { if(a<0) return -a; return a; }
template<class T> T Dist(T x1,T y1,T x2,T y2) { return sqrt(sqr(x1-x2)+sqr(y1-y2)); }
template<class T> T DistSqr(T x1,T y1,T x2,T y2) { return sqr(x1-x2)+sqr(y1-y2); }
#define iss istringstream
#define oss ostringstream
template<class T> string toString(T n) { oss ost; ost<<n; return ost.str(); }
int toInt(string s){int r=0; iss sin(s); sin>>r; return r; }
LL toLL(string s){ LL r=0; iss sin(s); sin>>r; return r; }
double toDouble(string s){double r=0;iss sin(s);sin>>r;return r;}
LD toLD(string s){LD r=0; iss sin(s); sin>>r; return r; }
#define present(c,X) ((c).find(X) != (c).end())
#define cpresent(c,X) (find(ALL(c),(X)) != (c).end())
#define two(X) (1<<(X))
#define contains(S,X) (((S)&two(X))!=0)
#define twoL(X) (((long long)(1))<<(X))
#define containsL(S,X) ((S&twoL(X))!=0)
#define ones(X) (two(X)-1)
#define onesL(X) (twoL(X)-1)
template<class T> inline int countones(T n) { int c = 0; for(;n;c++) n &= (n-1); return c; }
template<class T> inline T lowbit(T n) { return (n^(n-1))&n; }
#define ForSubset(a,b) for(long long (a) = (b); (a)!=0; (a) = ((b) & ((a)-1)))
template<class T> inline T getmod(T n, T m) {return (n%m+m)%m;}
bool isleap(int year) { return (year%400==0) || ((year%4 == 0) && (year%100 != 0)); }
template<class T> T ipow(const T &m, const T &n) { if(n==0)return 1; T a=ipow(m,n/2);
    return (n&1)?a*a*m:a*a; }
```

```
int main() {
    #ifdef MYCOMP
    freopen("input.txt", "r", stdin);
    #endif
    return 0;
}
```

2 Unionfind

```
struct UnionFind {
    int c;
    vector<int> parent, rank;
    UnionFind(int n) : c(n), parent(n), rank(n)
    { for(int i = 0; i < n; ++i)    parent[i] = i, rank[i] = 0; }
    int root(int x) { if(parent[x] != x) parent[x] = root(parent[x]); return parent[x]; }
    bool join(int a, int b) {
        a = root(a); b = root(b);
        if(a == b) return false;
        if(rank[a] > rank[b]) parent[b] = a;
        else { parent[a] = b; if(rank[a] == rank[b]) rank[b]++; }
        --c;
        return true;
     }
};
```

3 Getid

```
map<string,int> mapid;
int getid(string X) { return present(mapid,X)?mapid[X]:mapid[X]=mapid.size()-1; }
```

4 SegmentTree

```
const int N = 5000;
int A[N];
inline bool comp(int pos1, int pos2) { return A[pos1] < A[pos2]; }
int H[4*N];
void initialize(int n, int i, int j) {
    if(i == j) { H[n] = i; return; }
    initialize(2*n,i,(i+j)/2);
    initialize(2*n+1,(i+j)/2+1, j);
    H[n]=comp( H[2*n] , H[2*n+1] )?H[2*n]:H[2*n+1];
}
int query(int n, int i,int j, int a, int b) {
    if(a<=i && j<=b) return H[n];
    const int mid = (i+j)/2;
    int p1=(a>mid)?-1:query(2*n,i,mid,a,b);
    int p2=(mid>=b)?-1:query(2*n+1,mid+1,j ,a,b);
    if(p1==-1)return p2;
    if(p2==-1)return p1;
    return comp(p1,p2)?p1:p2;
}
void update(int n, int i, int j, int pos, int v) {
    const int mid = (i+j)/2;
    if(pos<=mid && i<mid) update(2*n,i,mid,pos,v);
    if(pos>mid && mid+1<j) update(2*n+1,mid+1,j,pos,v);
    H[n]=comp( H[2*n],H[2*n+1] )?H[2*n]:H[2*n+1];
}
```

5 BIT

```
int bit[M],n,m;
void update(int x, int v) {
    while( x <= n ) {
        bit[x] += v;
        x += x & -x;
    }
}
int sum( int x ) {
    int ret = 0;
    while( x > 0 ){
        ret += bit[x];
        x -= x & -x;
    }
    return ret;
}
```

6 BIT2D

```
int bit[M][M];
int n;
int sum( int x, int y ){
    int ret = 0;
    while( x > 0 ){
        int yy = y;
            while( yy > 0 ) ret += bit[x][yy], yy -= yy & -yy;
        x -= x & -x ;
    }
    return ret ;
}
void update(int x , int y , int val){
    int y1;
    while (x <= n){
        y1 = y;
        while (y1 <= n){
            bit[x][y1] += val;
            y1 += (y1 & -y1);
        }
        x += (x & -x);
    }
}
```

7 LIS

```
void LIS() {
   int n,total, nprob = 0;
   vector< int > table;
   while(scanf("%d", &total)==1){
        if( total == 0 ) break;
        table.clear();
        REP(kkk, total) {
            scanf("%d",&n);
            vector< int >::iterator i = lower_bound( table.begin(), table.end(), n );
            if( i== table.end() ) table.push_back( n );
            else *i <?= n;
        }
        printf("Set %d: %d\n",++nprob,table.size());
```

```
    }
}
```

8 StructVec

```
//vector class, also used as a point
template<int d=2, class T=LD> struct Vec {
    T c[d];
    Vec(){ Rep(i,d) c[i]=0; }
    Vec(T input[]) { Rep(i,d) c[i]=input[i]; } //convert from array
    Vec(vector<T> input) { Rep(i,d) c[i]=input[i]; } //convert from vector
    Vec(T a1, T a2){assert(d==2); c[0]=a1;c[1]=a2; }//2d constructor
    Vec(T a1, T a2, T a3){assert(d==3); c[0]=a1;c[1]=a2;c[2]=a3; }//3d constructor
    T& operator[](int dim) { return c[dim]; } //component
    T operator[](int dim) const { return c[dim]; } //constant component
    Vec operator+(const Vec &a)const{ Vec r; Rep(i,d) r[i]=c[i]+a[i]; return r;}//addition
    Vec operator-(const Vec &a)const{ Vec r; Rep(i,d) r[i]=c[i]-a[i]; return r;}//subtraction
    T operator*(const Vec &a)const{ T r=0; Rep(i,d) r+=a[i]*c[i]; return r;}//dot product
    Vec operator*(T a)const{ Vec r; Rep(i,d) r[i]=c[i]*a; return r;}//scale vector
    T len()const { T r=0; Rep(i,d) r+=c[i]*c[i]; return sqrt(r);}//length of vector
    friend ostream& operator<<(ostream &os, const Vec &a){Rep(i,d){if(i)os<<" ";os<<a[i];}return os;}
    T operator%(const Vec<2> &a)const{assert(d==2);return c[0]*a[1]-c[1]*a[0]; }//2d cross product
    Vec operator%(const Vec<3> &a)const{assert(d==3);return Vec(c[1]*a[2]-c[2]*a[1],
        c[2]*a[0]-c[0]*a[2],c[0]*a[1]-c[1]*a[0]);}//3d cross
    Vec rotate(LD angle) {assert(d==2);return Vec(c[0]*cos(angle)-c[1]*sin(angle),
        c[0]*sin(angle)+c[1]*cos(angle));}
    bool operator<(const Vec &v)const{ Rep(i,d)if(c[i]!=v[i])return c[i]<v[i];return false; }
};
```

9 LinePointDist

```
LD linePointDist(Vec<> A, Vec<> B, Vec<> C, bool isSegment){
    LD dist = ((B-A)%(C-A)) / sqrt((B-A)*(B-A));
    if(isSegment){
        if( (C-B)*(B-A) > eps) return (B-C).len();
        if( (C-A)*(A-B) > eps) return (A-C).len();
    }
    return iabs(dist);
}
```

10 SegmentsIntersect

```
bool segmentsIntersect( Vec<> A, Vec<> B, Vec<> C, Vec<> E ) {
    Vec<> in = Line<>(A,B).intersect(Line<>(C,E));
    return linePointDist(A,B,in,true) < eps && linePointDist(C,E,in,true) < eps;
}
```

11 AreaOfPolygon

```
template<class T> T twoarea(vector< Vec<2,T> > p) {
    T ret=p[p.sz-1]%p[0];
    Rep(i,p.sz-1) ret += p[i]%p[i+1];
    return ret;
}
```

12 StructLine

```
template<class T=LD> struct Line {
    T A,B,C;
    Line(){A=B=C=0;}
    Line(T x1,T y1,T x2,T y2) { A=y2-y1; B=x1-x2; C=A*x1+B*y1; }//construct from 4 values
    T dist(T a0, T a1) const { return iabs( a0*A + a1*B - C ) / sqrt(A*A+B*B); }
    Line rot90(T x,T y)const{Line ret;ret.A=-B;ret.B=A;ret.C=ret.A*x+ret.B*y;return ret;}//rot line
    friend ostream& operator<<(ostream& os,const Line &a){return os<<a.A<<"*x + "<<a.B<<"*y = "<<a.C; }
    //requires Vec
    Line(Vec<2,T>a,Vec<2,T>b){A=b[1]-a[1];B=a[0]-b[0];C=A*a[0]+B*a[1];} //construct from two points
    T dist(Vec<2,T> a) const { return ( a[0]*A + a[1]*B - C ) / sqrt(A*A+B*B); }
    Vec<2,T> intersect(const Line &l) const {T det = A*l.B - l.A*B; if(iabs(det) < eps)det = 0;
            return Vec<2,T>((l.B*C - B*l.C)/det, (A*l.C - l.A*C)/det); }
};
//for segment segment intersection, check additionally
//min(x1,x2) <= x <= max(x1,x2)
```

13 GetMidLine

```
// get a line passing between two points
template<class T>
Line<T> getmidline(Vec<2,T> a, Vec<2,T> b) {
    Vec<2,T> mid(a+b);
    return Line<T>(a,b).rot90(mid[0]/2,mid[1]/2);
}
```

14 ReflectPoint

```
//reflect a point into it's "mirror" with repect to a line
template<class T>
Vec<2,T> reflectPoint(Vec<2,T> p, Line<T> l) {
    Line<T> r = l.rot90(p[0],p[1]);
    Vec<2,T> Y=l.intersect(r);
    return Y - (p-Y);
}
```

15 ConvexHull

```
// Returns a list of points on the convex hull in counter-clockwise order.
// Note: the last point in the returned list is NOT the same as the first one. Because of (k-1)
template<class T>
vector< Vec<2,T> > convexHull(vector<Vec<2,T> > P) {
    int n = P.size(), k = 0;
    vector< Vec<2,T> > H(2*n);
    sort(P.begin(), P.end());
    // Build lower hull
    for (int i = 0; i < n; i++) {
        while (k >= 2 && (H[k-2] - H[k-1]) % (H[k-2] - P[i]) <= 0) k--;
        H[k++] = P[i];
    }
    // Build upper hull
    for (int i = n-2, t = k+1; i >= 0; i--) {
        while (k >= t && (H[k-2] - H[k-1]) % (H[k-2] - P[i]) <= 0) k--;
        H[k++] = P[i];
    }
    H.resize( k-1 );//k-1 to remove last point (duplicate)
    return H;
}
```

16 Bpm

```
#define M 1010
int grid[M][M];
int l[M], r[M], seen[M];
int rows, cols;
bool dfs(int x) {
    if( seen[x] ) return false;
    seen[x] = true;
    Rep(i,cols) if( grid[x][i] ) if( r[i] == -1 || dfs( r[i] ) ) {
        r[i] = x, l[x] = i;
        return true;
    }
    return false;
}
int bpm() {
    SET( l, -1 );
    SET( r, -1 );
    int ret = 0;
    Rep(i,rows) {
        SET( seen, 0 );
        if( dfs( i ) ) ret ++;
    }
    return ret;
}
bool lT[M], rT[M];
void konigdfs(int x) {
    if( !lT[x] ) return; lT[x] = 0;
    Rep(i,cols) if(grid[x][i] && i != l[x]) {
        rT[i] = true;
        if( r[i] != -1) konigdfs(r[i]);
    }
}
int konig() {
    SET(lT, 1);
    SET(rT, 0);
    Rep(i,rows) if(l[i] == -1) konigdfs(i);
}
```

17 Hopcroft

```
#define M 1000
vector< int > edge[ M ];
int nr, nc;
int n, m;
int l[M], r[M], seen[M];
bool dfs( int x ) {
    if( seen[x] ) return false;
    seen[x] = true;
    REP(i,edge[x].sz) {
        int v = edge[x][i];
        if( r[v] == -1 || dfs( r[v] ) ) {
            r[v] = x,  l[x] = v;
            return true;
        }
    }
    return false;
}
int match() {
    int ret = 0;
    REP(i,n) l[i] = -1; REP(i,m) r[i] = -1;
    bool done;
    do {
        done = true;
        REP(i,n) seen[i] = 0;
        REP(i,n) if( l[i] == -1 && dfs( i ) ) done = false;
    }while( !done );
    REP(i,n) ret += ( l[i] != -1 );
    return ret;
}
```

18 Djikstra-set

```
double d[51][51];
double a[51][4001];
typedef pair<double, pii> st;
Rep(i,n) Rep(j,m) a[i][j] = 1e100;
S.clear();
/*add*/a[0][0] = 0.0; S.insert( make_pair(0.0, pii(0, 0) ) );
while (!S.empty()) {
    st curr = *S.begin(), next;
    S.erase(S.begin());
    i = curr.y.x;
    k = curr.y.y;
    for (j = 0; j < n; j++) {
        next.x = curr.x + d[i][j];
        next.y.x = j;
        next.y.y = curr.y.y + get(d[i][j], maxD);
        if (next.x <= maxT && next.y.y < m && next.x < a[next.y.x][next.y.y]) {
            S.erase( st(a[next.y.x][next.y.y], next.y) );
            /*add*/a[next.y.x][next.y.y] = next.x; S.insert(next);
        }
    }
}
```

19 DijkstraPrqueue

```
VI edge[Z], cost[Z];
int d[Z];
struct data {
    int u, c;
    data() {}
    data( int uu, int cc ) : u( uu ), c( cc ) {}
    bool operator < ( const data& p ) const {
        return c > p.c;
    }
};
int dijkstra( int na, int nb ) {
    priority_queue< data > q;
    q.push( data( na, 0 ) );
    FOR(i,1,ncity) d[i] = -1;
    d[na] = 0;
    while( !q.empty() ) {
        data u = q.top(); q.pop();
        if( d[ u.u ] < u.c ) continue;
        if( u.u == nb ) break;
        REP(i,edge[u.u].sz) {
            int v = edge[u.u][i];
            if( d[v] == -1 || d[v] > d[u.u] + cost[u.u][i] ) {
                d[v] = d[u.u] + cost[u.u][i];
                q.push( data( v, d[v] ) );
            }
        }
    }
    return d[ nb ];
}
```

20 ArticulationPoint

```
//articulation, root is a special case
DFS_Visit(v) { color[v]=GREY;time=time+1;d[v] = time;
    low[v]= d[v];
    for each w in Adj[v]{
        if(color[w] == WHITE){
            prev[w]=u;
            DFS_Visit(w);
            if low[w] >= d[v]
                record that vertex v is an articulation
                    if (low[w] < low[v]) low[v] := low[w];
        }
        else if w is not the parent of v then
            //--- (v,w) is a BACK edge
            if (d[w] < low[v]) low[v] := d[w];
    }
    color[v] = BLACK;  time = time+1;   f[v] = time;
}
```

21 StronglyConnectedC

```
// Inputs (populate these).
int deg[NN]; int adj[NN][NN];
// Union-Find.
int uf[NN];
int FIND( int x ) { return uf[x] == x ? x : uf[x] = FIND( uf[x] ); }
void UNION( int x, int y ) { uf[FIND( x )] = FIND( y ); }
// dfsn[u] is the DFS number of vertex u.
int dfsn[NN], dfsnext;
// mindfsn[u] is the smallest DFS number reachable from u.
int mindfsn[NN];
// The O(1)-membership stack containing the vertices of the current component.
int comp[NN], ncomp;
bool incomp[NN];
void dfs( int n, int u ) {
  dfsn[u] = mindfsn[u] = dfsnext++;
  incomp[comp[ncomp++] = u] = true;
  for( int i = 0, v; v = adj[u][i], i < deg[u]; i++ ) {
    if( !dfsn[v] ) dfs( n, v );
    if( incomp[v] ) mindfsn[u] <?= mindfsn[v];
  }
  if( dfsn[u] == mindfsn[u] ) {
    // u is the root of a connected component. Unify and forget it.
    do {
      UNION( u, comp[--ncomp] );
      incomp[comp[ncomp]] = false;
    } while( comp[ncomp] != u );
  }
}
void scc( int n ) {
  // Init union-find and DFS numbers.
  for( int i = 0; i < n; i++ ) dfsn[uf[i] = i] = ncomp = incomp[i] = 0;
  dfsnext = 1;
  for( int i = 0; i < n; i++ ) if( !dfsn[i] ) dfs( n, i );
}
```

22 Kruskal

```
sort( s, s+nedge, comp );
unionfind u;
int cost = 0;
REP(i,nedge) {
    if( u.find( s[i].a ) != u.find( s[i].b ) ) {
        cost += s[i].c;
        u.Union( s[i].a , s[i].b );
    }
}
```

23 Maxflow

```
#define MM 1211
VI edges[MM];
int visited[MM];
int cap[MM][MM];
int dfs(int src, int end, int fl) {
    if(visited[src])return 0;
    if(src==end)return fl;
    visited[src]=1;
    Foreach(v,edges[src]) {
        int x = min( fl, cap[src][*v] );
        if(x>0) {
            x = dfs(*v,end,x);
            if(x==0)continue;
            cap[src][*v] -= x;
            cap[*v][src] += x;
            return x;
        }
    }
    return 0;
}
int flow(int src, int sink) {
    int ret = 0;
    while(1) {
        SET(visited,0);
        int d = dfs(src,sink,INT_MAX);
        if(d==0)break;
        ret += d;
    }
    return ret;
}
void addEdge(int from, int to, int value) {
    cap[ from ][to]=value;
    edges[from].pb(to);
    edges[to].pb(from);
}
```

24 MaxflowDinic

```
#include<limits>
#define NN 5010
const long long INF = numeric_limits<long long>::max();
struct edge { int point, next; LL flow, capa; };
vector<edge> edges;
int head[NN], dist[NN], Q[NN], work[NN];
int node, src, dest;
void flowinit(int _node,int _src,int _dest) {
    node=_node;
    src=_src;
    dest=_dest;
    SET(head,-1);
    edges.clear();
    edges.reserve( 60010 );
}
bool dinic_bfs() {
    SET(dist,-1); dist[src]=0;
    int sizeQ=0;
    Q[sizeQ++]=src;
    for (int cl=0;cl<sizeQ;cl++)
        for (int k=Q[cl],i=head[k];i>=0;i=edges[i].next)
            if (edges[i].flow<edges[i].capa && dist[edges[i].point]<0)
                dist[edges[i].point]=dist[k]+1, Q[sizeQ++]=edges[i].point;
    return dist[dest]>=0;
}
LL dinic_dfs(int x, LL exp) {
    if (x==dest) return exp;
    for (int &i=work[x];i>=0;i=edges[i].next) {
        int v=edges[i].point;
        LL tmp;
        if (edges[i].flow<edges[i].capa && dist[v]==dist[x]+1
                && (tmp=dinic_dfs(v,min(exp,edges[i].capa-edges[i].flow)))>0)
            return edges[i].flow+=tmp,edges[i^1].flow-=tmp, tmp;
    }
    return 0;
}
LL dinic_flow() {
    LL result=0;
    while (dinic_bfs()) {
        Rep(i,node) work[i]=head[i];
        for(LL delta; delta=dinic_dfs(src,INF); result+=delta);
    }
    return result;
}
void addEdge(int u,int v, LL c1=1, LL c2=0) {
//  cout << "add edge " << u+1 << " to " << v+1 << endl;
    edges.pb( (edge) { v,head[u],0,c1 } ); head[u] = edges.sz-1;
    edges.pb( (edge) { u,head[v],0,c2 } ); head[v] = edges.sz-1;
}
```

25 MincostMaxflow

```
#define N 705
int n, nE;
int d[N], pre[N];
struct edge {
    int to, cost, cap;
    int back;
};
edge E[N*N];
vector< int > e[N];
int mincost( int s, int t, int lim ) {
    int flow = 0, ret = 0;
    while( flow < lim ) {
        SET( d, -1 ); SET( pre, -1 );
        d[s] = 0;
        // bellman ford
        int jump = n-1;
        bool done = 0;
        while( !done && --jump >= 0) {
            done = 1;
            REP(i,n) if( d[i] != -1 ) REP(j,e[i].sz) {
                edge& x = E[ e[i][j] ];
                int v = x.to;
                if( x.cap > 0 && ( d[v] == -1 || d[v] > d[i] + x.cost )) {
                    d[v] = d[i] + x.cost;
                    pre[v] = x.back;
                    done = 0;
                    //cout<<v<<" "<<d[v]<<endl;
                }
            }
            if( done ) break;
        }
        if( d[t] == -1 ) break;
        // traverse back
        int x = t, cflow = 1<<30;
        while( x != s ) {
            edge& ed = E[ pre[x] ];
            cflow = min( cflow, E[ ed.back ].cap );
            x = ed.to;
        }
        if( !cflow ) break;
        int take = min( lim - flow, cflow );
        ret += d[t] * take;
        flow += take;
        x = t;
        while( x != s ) {
            edge& back = E[ pre[x] ];
            edge& forw = E[ back.back ];
            back.cap += take;
            forw.cap -= take;
            x = back.to;
        }
    }
    if( flow < lim ) return -1;
    return ret;
}
// remember to add -cost in the opposite direction
void add( int u, int v, int uv, int vu, int fuv, int fvu ) {
    int a = nE, b = nE+1;
    nE += 2;
    E[ a ].to = v, E[ a ].cost = uv, E[ a ].cap = fuv, E[ a ].back = b;
    E[ b ].to = u, E[ b ].cost = vu, E[ b ].cap = fvu, E[ b ].back = a;
    e[ u ].pb( a ), e[ v ].pb( b );
}
```

26 BigNumInt

```
const int B = 10;// If you change this value, then you
                 // will also have to change operator<<.
class num : public vector<char>
{public:
    num(){}
    num(int n) {
        for(;n;n/=B) push_back(n%B);
    }
    num(string n) {
        int down=0,sign=1;
        if(n[down]=='-') {
            down++;
            sign=-1;
        }
        for(int i=n.size()-1;i>=0;i--)
            push_back(sign*(n[i]-'0'));
        norm();
    }
    void norm() {
        while(!empty()&&!back())
            pop_back();
        int r = 0;
        for(int i=0;i<size();i++) {
            at(i) += r;
            r = at(i) / B;
            at(i) %= B;
        }
        for(;r;r/=B) push_back(r);
    }
};
ostream& operator<<(ostream &s, const num &a) {
    if(!a.size()) return s << '0';
    if(a.back()<0)s << '-';
    for(int i = a.size()-1;i>=0;i--)
        s << char(iabs(a[i])+'0');
    return s;
}
bool operator==(const num &a, const num &b) {
    if(a.size()!=b.size())return false;
    for(int i=0;i<a.size();i++)
        if(a[i]!=b[i])return false;
    return true;
}
bool operator!=(const num &a, const num &b) {
    return !(a==b);
}
bool operator<(const num &a, const num &b) {
    if(!a.size() && !b.size())return false;
    if(!a.size()) return 0 < b.back();
    if(!b.size()) return a.back() < 0;
    if(a.back()*b.back()<0)return a.back()<0;
    if(a.size()!=b.size())return (a.size()<b.size())^(a.back()<0);
    for(int i=a.size()-1;i>=0;i--)
        if(a[i]!=b[i])return a[i]<b[i];
    return false;
}
bool operator<=(const num &a, const num &b) {
/*    if(!a.size() && !b.size())return true;
    if(!a.size()) return 0 <= b.back();
    if(!b.size()) return a.back() <= 0;
    if(a.back()*b.back()<0)return a.back()<0;
    if(a.size()!=b.size())return (a.size()<b.size())^(a.back()<0);
    for(int i=a.size()-1;i>=0;i--)
        if(a[i]!=b[i])return a[i]<=b[i];
    return true;*/
    return a < b || a == b; //inefficient?
}
num operator-(const num &a) {
    num b(a);
    for(int i=0;i<b.size();i++) b[i]=-b[i];
    return b;
}
num operator-(const num &a, const num &b);
num operator+(const num &a, const num &b) {
    if(!a.size())return b;
    if(!b.size())return a;
    if(a.back()*b.back()<0)return a -(-b);
    num c(a);
    int s = min(a.size(),b.size());
    for(int i=0;i<s;i++)
        c[i]+=b[i];
    while(s<b.size())
        c.push_back(b[s++]);
    c.norm();
    return c;
}
num operator-(const num &a, const num &b) {
    if(!a.size())return -b;
    if(!b.size())return a;
    if(a.back()*b.back()<0)return a +(-b);
    num up=iabs(a),down=iabs(b);
    if(up<down)swap(up,down);
    for(int i=0;i<down.size();i++)
        up[i]-=down[i];
    for(int i=0;i<up.size();i++)
        for(;up[i]<0;up[i]+=B)
            up[i+1]--;
    up.norm();
    return (a<b)?-up:up;
}
num operator*(const num &a, const num &b) {
    num c;
    c.resize(a.size()+b.size());
    fill(c.begin(),c.end(),0);
    for(int i=0;i<a.size();i++)
        for(int j=0;j<b.size();j++) {
            int r = c[i+j] + int(a[i])*b[j];
            c[i+j+1] += r/B;
            c[i+j] = r%B;
        }
    c.norm();
    return c;
}
num divByInt(const num &a, int b) {
    num c(a);
    int r=0;
    for(int i=c.size()-1;i>=0;i--) {
        c[i] += r*B;
        r = c[i] % b;
        c[i] /= b;
    }
    c.norm();
    return c;
}
num operator/(const num &a, const num &b) {
    if(a==0 || b==0) return 0;
    num c;
    c.resize(a.size());
    num row,bAbs=iabs(b),aAbs=iabs(a);
    for(int i=a.size()-1;i>=0;i--) {
        row.insert(row.begin(),1,aAbs[i]);
        row.norm();
        for(c[i]=0;bAbs <= row;c[i]++)
            row = row - bAbs;
    }
    c.norm();
    return (a.back()*b.back()<0)?-c:c;
}
num operator%(const num &a, const num &b) {
    if(a==0 || b==0) return 0;
    num row,bAbs=iabs(b),aAbs=iabs(a);
    for(int i=a.size()-1;i>=0;i--) {
        row.insert(row.begin(),1,aAbs[i]);
        row.norm();
        while(bAbs <= row)
            row = row - bAbs;
    }
    return (a.back()<0)?-row:row;
    //return a - (a/b)*b;//unefficient?
}
num power(const num &m, const num &n) {
    if(n.empty())return 1;
    num a=power(m,n/2);
    return (n[0]%2)?a*a*m:a*a;
}
```

27 BsearchLE

```
int down = start;
int up = end+1;
while(down+1 < up) {
    int mid = (down+up)/2;
    if(f1(mid)) down = mid; else up = mid;
}
assert(down == L);
```

28 BsearchGE

```
int down = start-1;
int up = end;
while(down+1 < up) {
    int mid = (down+up)/2;
    if(f2(mid)) up = mid; else down = mid;
}
assert(down+1 == L);
```

29 PermIndex

```
int perm_index (char pit[], int size) {
    int i;
    register int j, ball;
    int index = 0;
    for (i = 1; i < size; i++) {
        ball = pit[i-1];
        for (j = i; j < size; j++) {
            if (ball > pit[j])
            index ++;
        }
        index *= size - i;
    }
    return index;
}
```

30 Geodistance

```
double gdist
( double a_lat, double b_lat, double a_long, double b_long ) {
    return acos(cos(a_lat) * cos(b_lat) * cos(a_long - b_long) + sin(a_lat) * sin(b_lat));
} // don't forget to multiply radius with it! ;)
```

31 Matrixmul

```
i64 mod;
struct matrix {
    int m,n;
    i64 a[10][10];
};
void print( matrix& a ) {
    REP(i,a.m) {
        REP(j,a.n) cout <<" " << a.a[i][j];
        cout << endl;
    }
}
matrix operator * (  matrix& x,  matrix& y ) {
    matrix ret;
    ret.m = x.m;
    ret.n = y.n;
    REP(i,ret.m) REP(j,ret.n) {
        ret.a[i][j] = 0;
        REP(k, x.n) {
            ret.a[i][j] += (x.a[i][k] * y.a[k][j]);
            ret.a[i][j] %= mod;
        }
    }
    return ret;
}
matrix power( matrix& x , int n ) {
    if( n == 1 ) return x;
    matrix ret = power( x, n/2 );
    ret = ret * ret;
    if( n % 2 == 0 ) return ret;
    ret = ret * x ;
    return ret;
}
```

32 Heron

```
double area( double x, double y, double z ) {
    double s = ( x + y + z ) / 2.;
    double X = s * ( s - x ) * ( s - y ) * ( s - z );
    return sqrt( X );
}
```

33 Genprimes

```
#define MAXP 1000000
bool primes[MAXP];
vector<int> plist;
void genprimes() {
    plist.clear();
    int m = (int)sqrt(int(MAXP));
    primes[0]=primes[1]=0;
    for(int i=2;i<MAXP;i++) primes[i] = 1;
    for(int i=2;i<=m;i++)
        if (primes[i])
            for(int j=i*i;j<MAXP;j+=i) primes[j] = 0;
    for(int i=2;i<MAXP;i++)if(primes[i])plist.push_back(i);
}
```

34 Choose

```
LL choose[50][50];
For(i,1,50) {
    choose[i][0]=choose[i][i]=1;
    For(j,1,i) {
        choose[i][j] = choose[i-1][j]+choose[i-1][j-1];
    }
}
```

35 EuclidExtended

```
template<class T> inline T euclid(T a,T b,T &X,T &Y) {
    if(a<0) { T d=euclid(-a,b,X,Y); X=-X; return d; }
    if(b<0) { T d=euclid(a,-b,X,Y); Y=-Y; return d; }
    if(b==0) { X=1; Y=0; return a; }
    else{
        T d=euclid(b,a%b,X,Y); T t=X;
        X=Y; Y=t-(a/b)*Y;
        return d;
    }
}
```

36 Isprime2MillerRabin

```
bool isprime2(long long p, int iteration){
    if(p<2)
        return false;
    if(p!=2 && p%2==0)
        return false;
    long long s=p-1;
    while(s%2==0)
        s/=2;
    for(int i=0;i<iteration;i++){
        long long a=rand()%(p-1)+1,temp=s;
        long long mod=powermod(a,temp,p);
        while(temp!=p-1 && mod!=1 && mod!=p-1){
            mod=multiplymod(mod,mod,p);
            temp *= 2;
        }
        if(mod!=p-1 && temp%2==0)
            return false;
    }
    return true;
}
```

37 EulerPhi

```
template<class T> T euler(T n) {
    T result = n;
    for(int i=2;i*i <= n;i++) {
        if (n % i == 0) result -= result / i;
        while (n % i == 0) n /= i;
    }
    if (n > 1) result -= result / n;
    return result;
}
```

38 Euler2Phi

```
template<class T> inline T euler2(T n) {
    vector<pair<T,int> > R=factorize(n);
    T r=n;
    for (int i=0;i<R.size();i++)
        r=r/R[i].first*(R[i].first-1);
    return r;
}
```

39 SGcdCongrModinv

```
LL gcd( LL a, LL b ) {
    return !b ? a : gcd( b, a%b );
}
PII egcd( LL a, LL b ) {  // returns x,y | ax + by = gcd(a,b)
    if( b == 0 ) return mp( 1, 0 );
    else {
        PII d = egcd( b, a % b );
        return mp( d.y, d.x - d.y * ( a / b ) );
    }
}
LL congruence( int a, int b, int n ) { // finds ax = b(mod n)
    int d = gcd( a, n );
    if( b % d != 0 ) return 1<<30; // no solution
    PII ans = egcd( a, n );
    LL ret = ans.x * ( b/d + 0LL ), mul = n/d;
    ret %= mul;
    if( ret < 0 ) ret += mul;
    return ret;
}
LL m_inverse( LL num, LL mod ) {
    PII p = egcd( num, mod );
    int ret = p.x % mod;
    if( ret < 0 ) ret += mod;
    return ret;
}
```

40 Sterling1

```
LL S[110][110]; LL fact[110];
fact[0] = 1; For(i,1,110) fact[i] = (fact[i-1] * i) % M;
SET(S,0);
For(n,1,110) {
    S[n][n] = 1; S[n][1] = fact[n-1];
    if(n%2 == 0) S[n][1] *= -1;
    For(k,2,n) {
        S[n][k] = ( - (n-1)*S[n-1][k] + S[n-1][k-1] )%M;
    }
}
if(K==0)
    cout << (N==0) << endl;
else
    cout << iabs(S[N][K]%M) << endl;
```

41 Sterling2

```
LL S[110][110]; LL fact[110];
fact[0] = 1; For(i,1,110) fact[i] = (fact[i-1] * i) % M;
SET(S,0);
For(n,1,110) {
    S[n][n] = 1; S[n][1] = 1;
    For(k,2,n) {
        S[n][k] = ( (k)*S[n-1][k] + S[n-1][k-1] )%M;
    }
}
if(K==0)
    cout << (N==0) << endl;
else
    cout << iabs(S[N][K]%M) << endl;
```

42 Totient

```
FOR(i,1,M) f[i] = i;
FOR(n,2,M) if( f[n] == n ) for(int k=n; k<=M; k+=n) f[k] *= n-1, f[k] /= n;
```

43 SieveBits

```
const int MAX = 100000000;
int p[ MAX/64 + 2 ];
int np = 0;
#define on(x) ( p[x/64] & (1<<( (x%64)/2 ) ) )
#define turn(x)  p[x/64] |= (1<<( (x%64)/2 ) )
void sieve() {
    for(int i=3;i*i<MAX; i+=2) {
        if( !on(i) ) {
            int ii = i*i;
            int i2 = i+i;
            for(int j=ii; j<MAX; j+=i2) turn(j);
        }
    }
}
inline bool prime( int num ) {
    return num > 1 && ( num == 2 || ( (num&1) && !on( num ) ) );
}
```

44 KmpExplained

```
f[ len ] = the longest suffix that we can still use for matching if its mismatched at len
f[0] = f[1] = 0;
FOR(i,2,len) {
    int j = f[i-1]; // think as a recursion - last best matching
    while( true ) {
        if( s[j] == s[i-1] ) { // we got a way to maximize
            f[i] = j + 1;
            break;
        }else if( !j ) { // suicide
            f[i] = 0;
            break;
        }else j = f[j]; // think recursively, we couldn't use s[i-1] to improve the match for j
        //now we'll go for the best suffix of j and check if we can improve that by using s[i-1]
    }
}
i = j = 0;
while( true ) {
    if( i == len ) break;
    if( text[i] == s[j] ) { // we got an improve
        i++, j++;
        if( j == slen ) // match found
    }else if( j > 0 ) j = f[j]; // then we'll check again if we can improve at i
    else i++;
}
```