Table of Contents:

[**Precode** 2](#_Toc390104258)

[Bellman Ford 3](#_Toc390104259)

[Floyed Warshall 4](#_Toc390104260)

[BFS 4](#_Toc390104261)

[Topological Sort (Normal) 5](#_Toc390104262)

[Dijkstra (top 2-path weight) 6](#_Toc390104263)

[Dijkstra 8](#_Toc390104264)

[MST 9](#_Toc390104265)

[Stable Marriage 10](#_Toc390104266)

[Bigmod 11](#_Toc390104267)

[Lazy Propagation 12](#_Toc390104268)

[Binary Search 13](#_Toc390104269)

[Infix to postfix then Evaln 13](#_Toc390104270)

[Sudoku Solver 15](#_Toc390104271)

# 

# **Precode:**

//#include <bits/stdc++.h>

//#define \_ ios\_base::sync\_with\_stdio(0);cin.tie(0);

#include <algorithm>

#include <bitset>

#include <cctype>

#include <cmath>

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <fstream>

#include <iostream>

#include <list>

#include <map>

#include <queue>

#include <set>

#include <sstream>

#include <stack>

#include <string>

#include <vector>

using namespace std;

#define all(a,b,c) for(int I=0;I<b;I++) a[I] = c

#define BE(a) a.begin(),a.end()

#define chng(a,b) a^=b^=a^=b;

#define clr(y,z) memset(y,z,sizeof(y))

#define cntbit(mask) \_\_builtin\_popcount(mask)

#define CROSS(a,b,c,d) ((b.x-a.x)\*(d.y-c.y)-(d.x-c.x)\*(b.y-a.y))

#define EQ(a,b) (fabs(a-b)<ERR)

#define ERR 1e-5

#define FORE(i,a) for(typeof((a).begin())i=(a).begin();i!=(a).end();i++)

#define fr(i,a,b) for(i=a;i<=b;i++)

#define fread freopen("input.txt","r",stdin)

#define fri(a,b) for(int i=a;i<=b;i++)

#define frj(a,b) for(int j=a;j<=b;j++)

#define frk(a,b) for(int k=a;k<=b;k++)

#define frl(a,b) for(int l=a;l<=b;l++)

#define frin(a,b) for(int i=a;i>=b;i--)

#define frjn(a,b) for(int j=a;j>=b;j--)

#define frkn(a,b) for(int k=a;k>=b;k--)

#define frln(a,b) for(int l=a;l>=b;l--)

#define frn(i,a,b) for(i=a;i>=b;i--)

#define fwrite freopen("output.txt","w",stdout)

#define inf (1e9)

#define inpow(a,x,y) int i; a=x;fri(2,y) a\*=x

#define makeint(n,s) istringstream(s)>>n

#define mod 1000000007

#define ISS istringstream

#define ll long long

#define oo (1<<30)

#define OSS ostringstream

#define pb push\_back

#define PI 3.141592653589793

#define pi (2\*acos(0))

#define pp pop\_back

#define PRE 1e-8

#define print1(a) cout<<a<<endl

#define print2(a,b) cout<<a<<" "<<b<<endl

#define print3(a,b,c) cout<<a<<" "<<b<<" "<<c<<endl

#define rev(a) reverse(BE(a));

#define round(i,a) i = ( a < 0 ) ? a - 0.5 : a + 0.5;

#define SI set<int>

#define SII set<int>::iterator

#define SIZE(s) ((int)s.size())

#define saja(a) sort(BE(a))

#define sqr(a) ((a)\*(a))

#define SZ 50005

#define SZ1 55

#define typing(j,b) typeof((b).begin()) j=(b).begin();

#define VD vector<double>

#define VI vector<int>

#define VLL vector<long long>

#define VS vector<string>

# Bellman Ford:

#include <cstdio>

#include <queue>

#include <vector>

#define sz 1005

#define pb(a) push\_back(a)

#define pp pop\_back()

#define inf 1e9

using namespace std;

int col[sz], cas=1;

int dist[sz], val[sz];

vector<int> adj[sz],cost[sz];

bool bellman\_ford(int n)

{

//initialize

for (int i = 1; i<n; i++)

dist[i]=inf;

//relaxation of paths

for(int k=0; k<n-1; k++)

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

for (int j = 0; j<adj[i].size(); j++)

if(dist[i]+cost[i][j]<dist[adj[i][j]])

dist[adj[i][j]] = dist[i]+cost[i][j];

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

val[i]=dist[i];

bool flag=false;

queue<int>q;

//checking negative-cycle

for(int k=0; k<n-1; k++)

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

for (int j = 0; j<adj[i].size(); j++)

if(val[i]+cost[i][j]<val[adj[i][j]]) q.push(i),flag=true;

int x,len;

while(!q.empty())

{

x = q.front();

col[ x ]=cas;

q.pop();

len = adj[x].size();

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

if(col[ adj[x][i] ]!=cas) col[ adj[x][i] ]=cas,q.push(adj[x][i]);

}

return flag;

}

void init(int n)

{

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

adj[i].clear(),cost[i].clear();

}

int main()

{

int t, n, m, x, y, z;

scanf("%d", &t);

while(t--)

{

scanf("%d %d", &n, &m);

init(n);

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

{

scanf("%d %d %d", &x, &y, &z);

adj[y].pb(x);

cost[y].pb(z);

}

printf("Case %d:",cas);

bool flag = false;

if(bellman\_ford(n))

{

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

if(col[i]==cas) printf(" %d", i);

puts("");

}

else printf(" impossible\n");

cas++;

}

return 0;

}

# Floyed Warshall:

#include <stdio.h>

#include <algorithm>

#define sz 105

#define fread freopen("input.txt","r",stdin)

#define fwrite freopen("output.txt","w",stdout)

#define inf (1e8)

int adj[sz][sz];

void floyed\_warshall(int n)

{

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

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

{

if(i==j) continue;

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

{

if(i==k || j==k) continue;

adj[j][k] = std::min(adj[j][k], adj[j][i]+adj[i][k]);

}

}

return;

}

void init(int n)

{

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

{

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

adj[i][j] = inf;

adj[i][i] = 0;

}

return;

}

# BFS:

#include <cstdio>

#include <cstring>

#include <queue>

#include <vector>

#define sz 20010

#define pb(a) push\_back(a)

#define clr(abc,z) memset(abc,z,sizeof(abc))

using namespace std;

vector<int>adj[sz];

bool col[sz];

int m[sz];

int cnt, zero[2], vis[sz];

void bfs(int x)

{

int now,c = 0, len;

queue<int>q;

q.push(x);

vis[x] = 0;

col[x] = true;

zero[ c ]++;

while(!q.empty())

{

x = q.front();

c = (1^vis[x]);

len = adj[x].size();

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

{

now = adj[x][i];

if(col[now]) continue;

col[now] = true;

vis[now] = c;

zero[ c ]++;

q.push(now);

}

q.pop();

}

return;

}

# Topological Sort (Normal):

#include <cstdio>

#include <cstring>

#include <queue>

#include <vector>

#include <algorithm>

#include <map>

#include <string>

#define sz 20001

#define pb(a) push\_back(a)

#define clr(abc,z) memset(abc,z,sizeof(abc))

using namespace std;

int cnt,indeg[sz];

vector<int>adj[sz];

map<string,int>mp;

int incoming(int i, int j)

{

return binary\_search(adj[i].begin(),adj[i].end(),j);

}

bool topsort()

{

queue<int>q;

int deg=0;

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

if(!indeg[i])q.push(i),deg++;

int n,len;

while(!q.empty())

{

n = q.front();

len = adj[n].size();

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

if(--indeg[ adj[n][i] ]==0) q.push(adj[n][i]),deg++;

q.pop();

}

return cnt==deg;

}

int main()

{

int m,t,cas=1,x1,x2,n;

char n1[20],n2[20];

scanf("%d", &t);

while(t--)

{

scanf("%d", &m);

mp.clear();

cnt=0;

clr(indeg,0);

n = (m<<1);

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

adj[i].clear();

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

{

scanf("%s %s", n1,n2);

if(mp.find(n1)==mp.end())

{

x1=cnt;

mp[n1]=cnt++;

}

else x1=mp[n1];

if(mp.find(n2)==mp.end())

{

x2=cnt;

mp[n2]=cnt++;

}

else x2=mp[n2];

adj[x1].pb(x2);

indeg[x2]++;

}

if(topsort())printf("Case %d: Yes\n",cas++);

else printf("Case %d: No\n",cas++);

}

return 0;

}

/\*

1

3

soda wine

water wine

wine water

2

2

soda wine

water wine

3

soda wine

water wine

wine water

\*/

# Dijkstra(top 2-path weight):

#include <cstdio>

#include <queue>

#include <vector>

#define sz 5005

#define pb(a) push\_back(a)

#define inf (1e9)

using namespace std;

vector<int>adj[sz], cost[sz];

int nodecost[2][sz];

struct node{

int n, w;

node(){}

node(int x, int y)

{

n = x;

w = y;

}

bool operator < (const node &p) const

{

return w > p.w;

}

};

priority\_queue<node>q;

void dijkstra()

{

nodecost[0][0] = 0;

q.push(node(0,0));

node now;

int then, c;

int len;

while(!q.empty())

{

now = q.top();

q.pop();

len = adj[now.n].size();

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

{

then = adj[now.n][i];

c = cost[now.n][i];

if(now.w+c<nodecost[0][then])

{

nodecost[1][then]=nodecost[0][then];

nodecost[0][then]= now.w+c;

q.push(node(then,nodecost[0][then]));

}

else if(now.w+c<nodecost[1][then] && now.w+c!= nodecost[0][then])

{

nodecost[1][then]= now.w+c;

q.push(node(then,nodecost[1][then]));

}

}

}

return ;

}

int main()

{

int t, n, m, cas=1, u, v, w;

scanf("%d", &t);

while(t--)

{

scanf("%d %d", &n, &m);

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

{

adj[i].clear();

cost[i].clear();

nodecost[0][i] = inf;

nodecost[1][i] = inf;

}

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

{

scanf("%d %d %d", &u, &v, &w);

adj[u-1].pb(v-1);

adj[v-1].pb(u-1);

cost[u-1].pb(w);

cost[v-1].pb(w);

}

dijkstra();

printf("Case %d: %d\n", cas++, nodecost[1][n-1]);

}

return 0;

}

/\*

2

3 3

1 2 100

2 3 200

1 3 50

4 4

1 2 100

2 4 200

2 3 250

3 4 100

\*/

# Dijkstra:

#include <cstdio>

#include <cstring>

#include <queue>

#include <vector>

#include <algorithm>

#define sz 155

#define pb(a) push\_back(a)

#define inf (1e9)

using namespace std;

vector<int>adj[sz],cost[sz];

int node[sz];

struct junc{

int u, w;

junc(){}

junc(int a, int c)

{

u = a;

w = c;

}

bool operator < (const junc &p) const

{

return w > p.w;

}

};

priority\_queue<junc>data;

int dijkstra(int s, int e)

{

node[s] = 0;

data.push(junc(s,0));

junc p;

while(!data.empty())

{

p = data.top();

for (int i = 0; i<adj[ p.u ].size(); i++)

{

if(node[p.u]+ cost[p.u][i]< node[ adj[p.u][i] ])

{

node[ adj[p.u][i] ] = node[p.u] + cost[p.u][i];

data.push(junc(adj[p.u][i], node[ adj[p.u][i] ]));

}

}

data.pop();

}

return node[e];

}

int main()

{

int t, n, m, cas=1,x,y,w;

scanf("%d", &t);

while(t--)

{

scanf("%d %d", &n, &m);

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

adj[i].clear(), cost[i].clear(), node[i]=inf;

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

{

scanf("%d %d %d", &x, &y, &w);

adj[y-1].pb(x-1);

adj[x-1].pb(y-1);

cost[y-1].pb(w);

cost[x-1].pb(w);

}

w=dijkstra(0,n-1);

if(w<inf) printf("Case %d: %d\n",cas++,w);

else printf("Case %d: Impossible\n",cas++);

}

return 0;

}

# MST:

#include <stdio.h>

#include <vector>

#include <string>

#include <map>

#include <algorithm>

#define sz 55

#define pb(a) push\_back(a)

#define inf (1e9)

using namespace std;

struct edge

{

int u, v, w;

edge() {}

edge(int a, int b, int c)

{

u = a;

v = b;

w = c;

}

};

vector<edge>e;

int par[sz];

int find\_par(int n)

{

return par[n] = (par[n]==n?n:find\_par(par[n]));

}

void init(int n)

{

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

par[i] = i;

return;

}

bool comp(edge a, edge b)

{

return a.w<b.w;

}

int mst(int n)

{

sort(e.begin(), e.end(), comp);

int len = e.size(), x, y, ret=0;

vector<int>k;

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

{

x = find\_par(e[i].u);

y = find\_par(e[i].v);

if(x!=y)

{

par[x] = y;

k.pb(i);

ret+=e[i].w;

}

}

if(k.size()<n-1) return -1;

else return ret;

}

map<string, int>mp;

int main()

{

int t, n,m, cas=1, c;

char line[51], line1[51];

scanf("%d", &t);

while(t--)

{

e.clear();

n =0;

mp.clear();

scanf("%d", &m);

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

{

scanf("%s %s %d", line, line1, &c);

if(mp.find(line)==mp.end()) mp[line]=n++;

if(mp.find(line1)==mp.end())mp[line1]=n++;

e.pb(edge(mp[line],mp[line1],c));

}

init(n);

c = mst(n);

if(c!=-1)printf("Case %d: %d\n", cas++, c);

else printf("Case %d: Impossible\n", cas++);

}

return 0;

}

# Stable Marriage:

#include <cstdio>

#include <stack>

#define sz 200

using namespace std;

int main()

{

int n,t,cas=1,x,p,q,r;

int cand[sz][sz],comp[sz][sz];

stack<int>qq;

int conn[sz];

scanf("%d", &t);

while(t--)

{

scanf("%d", &n);

for (int i = n-1; i>=0; i--)

{

qq.push(i);

conn[i]=-1;

}

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

{

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

{

scanf("%d", &cand[i][j]);

cand[i][j]-=(n+1);

}

}

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

{

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

{

scanf("%d", &comp[i][j]);

comp[i][j]--;

}

}

while(!qq.empty())

{

x = qq.top();

qq.pop();

int k = 0;

while(cand[x][k]==-1) k++;

p = cand[x][k];

if(conn[p]==-1) conn[p] = x;

else

{

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

if(comp[p][i]==conn[p])

{

q = i;

break;

}

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

if(comp[p][i]==x)

{

r = i;

break;

}

if(r<q)

{

qq.push(conn[p]);

conn[p] = x;

}

else qq.push(x);

}

cand[x][k]=-1;

}

printf("Case %d:",cas++);

for (int i = 0; i<n; i++) printf(" (%d %d)", conn[i]+1,n+i+1);

printf("\n");

}

return 0;

}

/\*

1

3

4 5 6

4 5 6

4 5 6

1 2 3

1 2 3

1 2 3

\*/

# Bigmod:

ll bigmod(ll B,ll P,ll M)

{

ll R=1;

while(P>0)

{

if(P%2==1)

{

R=(R\*B)%M;

}

P/=2;

B=(B\*B)%M;

}

return R;

}

# Lazy Propagation:

#include <bits/stdc++.h>

#define \_ ios\_base::sync\_with\_stdio(0);cin.tie(0);

#define sz 100010

#define ll long long

#define clr(abc,z) memset(abc,z,sizeof(abc))

using namespace std;

ll stree[(sz<<2)], scale[(sz<<2)];

bool upd[(sz<<2)];

void push\_down(ll ind, ll LB, ll UB)

{

upd[ind] = false;

stree[ind]+=(scale[ind]\*(UB-LB+1));

if(UB!=LB)

{

ll c = (ind<<1);

upd[c] = upd[c+1] = true;

scale[c]+= scale[ind];

scale[c+1]+= scale[ind];

}

scale[ind] = 0;

return;

}

void push\_up(ll ind, ll LB, ll UB)

{

stree[ind] = stree[(ind<<1)] + stree[(ind<<1)+1];

return;

}

void update(ll ind, ll LB, ll UB, ll P, ll Q, ll val)

{

if(upd[ind]) push\_down(ind, LB, UB);

if(P<=LB&&Q>=UB)

{

scale[ind]+= val;

push\_down(ind,LB,UB);

return;

}

if(UB<P||LB>Q) return;

ll mid = ((UB+LB)>>1);

update((ind<<1), LB, mid, P,Q,val);

update((ind<<1)+1, mid+1,UB, P,Q,val);

push\_up(ind,LB,UB);

return;

}

ll query(ll ind, ll LB, ll UB, ll P, ll Q)

{

if(upd[ind]) update(ind, LB, UB,P,Q,0);

if(LB>Q||UB<P) return 0L;

if(LB>=P&&UB<=Q) return stree[ind];

ll mid = ((UB+LB)>>1);

return (query((ind<<1), LB, mid, P, Q)+query((ind<<1)+1,mid+1, UB, P,Q));

}

int main()

{

\_

ll t, n, q, x, y, v, w,cas=1;

scanf("%lld", &t);

while(t--)

{

clr(stree,0);

clr(upd,0);

clr(scale,0);

scanf("%lld %lld", &n, &q);

printf("Case %lld:\n", cas++);

while(q--)

{

scanf("%lld", &w);

if(w)

{

scanf("%lld %lld", &x, &y);

printf("%lld\n", query(1,0,n-1,x,y));

}

else

{

scanf("%lld %lld %lld", &x, &y, &v);

update(1,0,n-1,x,y,v);

}

}

}

return 0;

}

/\*

2

10 5

0 0 9 10

1 1 6

0 3 7 2

0 4 5 1

1 5 5

20 3

0 10 12 1

1 11 12

1 19 19

\*/

# Binary Search:

int bg=1,en=1000000011,ans;

while(bg<=en)

{

int mid = ((bg+en)>>1);

if(check(mid,n,m)) en = mid-1, ans = mid;

else bg = mid+1;

}

# Infix to postfix then Evaln:

#include <bits/stdc++.h>

using namespace std;

int pres[200]; ///presedence of operators

string infix\_to\_postfix(string P)

{

stack<char>s;

string Q;

int i =0;

char element;

while(i<P.size())

{

element = P[i++];

if(isalpha(element)) Q=Q+element; ///operand

else if(element == '(' || s.empty()) s.push(element); ///parenthesis start or nothing in stack

else if(element == ')') ///parenthesis end

{

while(s.top()!='(')

{

Q=Q+s.top();

s.pop();

}

s.pop(); ///poping up the first parenthesis

}

else

{

while(!s.empty() && pres[s.top()]>=pres[element]) ///wating for lower presedence or stack to be empty

{

Q=Q+s.top();

s.pop();

}

s.push(element);

}

}

while(!s.empty()) ///rest

{

Q=Q+s.top();

s.pop();

}

return Q;

}

int eval\_postfix(string &exp, int val[])

{

stack<int>s;

int len = exp.size();

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

{

if(isalpha(exp[i])) s.push(val[exp[i]]);

else if(exp[i]=='!')

{

int a = s.top();

s.pop();

s.push(!a);

}

else if(exp[i]=='&')

{

int a = s.top();

s.pop();

int b = s.top();

s.pop();

s.push(a&b);

}

else if(exp[i]=='|')

{

int a = s.top();

s.pop();

int b = s.top();

s.pop();

s.push(a|b);

}

}

return s.top();

}

string simplify(string s)

{

string f;

int len=s.size();

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

{

if(s[i]=='!')

{

int cnt = 0;

while(i<len)

{

if(s[i]=='!') cnt++;

else break;

i++;

}

if(cnt%2) f=f+"!";

if(i!=len)i--;

}

else f=f+s[i];

}

return f;

}

int main()

{

int t, n, m, cas=1, val[200];

char line[200];

string s1,s2;

pres['|'] = 1;

pres['&'] = 2;

pres['!'] = 3;

scanf("%d", &t);

while(t--)

{

scanf("%s", line);

s1 = line;

s1 = simplify(s1);

s1 = infix\_to\_postfix(s1);

scanf("%s", line);

s2 = line;

s2 = simplify(s2);

s2 = infix\_to\_postfix(s2);

set<char>ss;

int len = s1.size();

for (int i = 0; i<len; i++) if(isalpha(s1[i]))ss.insert(s1[i]);

len = s2.size();

for (int i = 0; i<len; i++) if(isalpha(s2[i]))ss.insert(s2[i]);

vector<char>v(ss.begin(), ss.end());

bool flag = true;

len = (1<<v.size());

int n = v.size();

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

{

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

if(i&(1<<j)) val[ v[j] ] = 1;

else val[ v[j] ] = 0;

if(eval\_postfix(s1,val)!=eval\_postfix(s2,val))

{

flag = false;

break;

}

}

if(flag) printf("Case %d: Equivalent\n", cas++);

else printf("Case %d: Not Equivalent\n", cas++);

}

return 0;

}

# Sudoku Solver:

#include <cstdio>

#include <cstring>

#include <vector>

#include <algorithm>

#define pb(a) push\_back(a)

#define inf (1e9)

#define clr(abc,z) memset(abc,z,sizeof(abc))

using namespace std;

char grid[9][10], ans[9][10];

int limit=9,up;

int sq[9];

int column[9], row[9];

bool flag;

int quad[9][9];

struct point{

int x, y;

};

vector<point>v;

void rec(int z)

{

if(!flag) return;

if(z==up)

{

flag = false;

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

strcpy(ans[i],grid[i]);

return;

}

int a=0, b, c=inf;

point p;

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

{

p = v[i];

if(grid[p.x][p.y]!='.') continue;

b=0;

for (int j = 1; j<=limit; j++)

if(!(sq[ quad[p.x][p.y] ]&(1<<j)) && !(column[p.y]&(1<<j)) && !(row[p.x]&(1<<j)) ) b++;

if(b<c)

{

c = b;

a = i;

}

}

if(c==0) return;

p = v[a];

for (int j = 1; j<=limit; j++)

{

if(!(sq[ quad[p.x][p.y] ]&(1<<j)) && !(column[p.y]&(1<<j)) && !(row[p.x]&(1<<j)) )

{

sq[ quad[p.x][p.y] ]|=(1<<j);

column[p.y]|=(1<<j);

row[p.x]|=(1<<j);

grid[p.x][p.y] = j+'0';

if(flag) rec(z+1);

sq[ quad[p.x][p.y] ]^=(1<<j);

column[p.y]^=(1<<j);

row[p.x]^=(1<<j);

grid[p.x][p.y] = '.';

}

}

return;

}

int main()

{

int t, n=3, m, cas=1,len, cnt;

vector<int>q;

scanf("%d", &t);

getchar();

point p;

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

for (int j = 0; j<limit; j++)

quad[i][j] = (i/n)\*n+(j/n);

while(t--)

{

gets(grid[0]);

clr(column,0);

clr(row,0);

clr(sq,0);

v.clear();

flag = true;

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

{

gets(grid[i]);

for (int j = 0; j<limit; j++)

{

if(grid[i][j]!='.')

{

m = grid[i][j] - '0';

sq[ quad[i][j] ]|= (1<<m);

column[j]|= (1<<m);

row[i]|= (1<<m);

}

else

{

p.x = i;

p.y = j;

v.pb(p);

}

}

}

up = v.size();

rec(0);

printf("Case %d:\n",cas++);

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

printf("%s\n", ans[i]);

}

return 0;

}

/\*

1

.46...9..

.3.1.....

.2..6..85

...87....

6...3...4

....14...

79..5..3.

.....2.4.

..2...61.

\*/