**★★★★☆**

**題組：Problem Set Archive** **with Online Judge**

**題號：10067: Playing with Wheels**

**解題者：林琮皓**

**解題日期：2023年5月11日**

**題意：**

一開始題目先給N個case，每個case裡會輸入一組做為起點的四個數字及一組作為終點的四個數字，接著輸入x組禁止的數字，求最快到終點需要調整幾次，若不能到達就輸出-1。

**題意範例：**

2

 8 0 5 6 0 0 0 0

 6 5 0 8 5 3 1 7

 5 8

 8 0 5 7 0 0 0 1

8 0 4 7 🡺 14 0 0 0 9 🡺 -1

 5 5 0 8 0 0 1 0

 7 5 0 8 0 0 9 0

6 4 0 8 0 1 0 0

 0 9 0 0

 1 0 0 0

 9 0 0 0

**解法：**

這題可以想像成是一個0000-9999的迷宮，從起點到終點避開禁止區域用BFS找最短路徑。

**解法範例：**

創一個struct分別存步數及數字，再用一個queue去存已經走到的位子，每次將queue的第一項取出尋找其所有可以走到的位子再放入queue中，第一個找到終點的步數即為答案；若queue空了則表示沒有找到，輸出-1。

**討論：**

若是起點剛好是被ban的位子的話還是可以移動，若是起點跟終點一樣或終點被ban可以分別輸出0和-1後直接跳出節省時間。

 注意0跟9的處理

**程式：**

#include<iostream>

#include<deque>

#include<cstring>

using namespace std;

int wheel[10000]={0};

typedef struct \_point{

 int num;

 int count;

}Point;

deque<Point> queue;

int main(){

 int CASE\_NUM,FORBIDDEN\_NUM,S1,S2,S3,S4,T1,T2,T3,T4,F1,F2,F3,F4;

 int start,target,ban;

 Point now,next;

 bool found=false;

 cin>>CASE\_NUM;

 int answer[CASE\_NUM];

 for(int i=0;i<CASE\_NUM;i++){

 memset(wheel,0,sizeof(wheel));

 cin>>S1>>S2>>S3>>S4;

 start=S1\*1000+S2\*100+S3\*10+S4;

 cin>>T1>>T2>>T3>>T4;

 target=T1\*1000+T2\*100+T3\*10+T4;

 cin>>FORBIDDEN\_NUM;

 for(int j=0;j<FORBIDDEN\_NUM;j++){

 cin>>F1>>F2>>F3>>F4;

 ban=F1\*1000+F2\*100+F3\*10+F4;

 if(ban==start){

 }

 else{

 wheel[ban]=-1;

 }

 }

 queue.clear();

 now.num=start;

 now.count=0;

 queue.push\_back(now);

 while(queue.size()>0){

 now=queue.front();

 queue.pop\_front();

 if(now.num==target){

 found=true;

 break;

 }

 if(wheel[now.num]==0){

 if(now.num%10==9){

 next.num=now.num-9;

 if(wheel[next.num]==0){

 next.count=now.count+1;

 queue.push\_back(next);

 }

 }

 else{

 next.num=now.num+1;

 if(wheel[next.num]==0){

 next.count=now.count+1;

 queue.push\_back(next);

 }

 }

 if((now.num%100/10)==9){

 next.num=now.num-90;

 if(wheel[next.num]==0){

 next.count=now.count+1;

 queue.push\_back(next);

 }

 }

 else{

 next.num=now.num+10;

 if(wheel[next.num]==0){

 next.count=now.count+1;

 queue.push\_back(next);

 }

 }

 if((now.num%1000/100)==9){

 next.num=now.num-900;

 if(wheel[next.num]==0){

 next.count=now.count+1;

 queue.push\_back(next);

 }

 }

 else{

 next.num=now.num+100;

 if(wheel[next.num]==0){

 next.count=now.count+1;

 queue.push\_back(next);

 }

 }

 if(now.num/1000==9){

 next.num=now.num-9000;

 if(wheel[next.num]==0){

 next.count=now.count+1;

 queue.push\_back(next);

 }

 }

 else{

 next.num=now.num+1000;

 if(wheel[next.num]==0){

 next.count=now.count+1;

 queue.push\_back(next);

 }

 }

 if(now.num%10==0){

 next.num=now.num+9;

 if(wheel[next.num]==0){

 next.count=now.count+1;

 queue.push\_back(next);

 }

 }

 else{

 next.num=now.num-1;

 if(wheel[next.num]==0){

 next.count=now.count+1;

 queue.push\_back(next);

 }

 }

 if((now.num%100/10)==0){

 next.num=now.num+90;

 if(wheel[next.num]==0){

 next.count=now.count+1;

 queue.push\_back(next);

 }

 }

 else{

 next.num=now.num-10;

 if(wheel[next.num]==0){

 next.count=now.count+1;

 queue.push\_back(next);

 }

 }

 if((now.num%1000/100)==0){

 next.num=now.num+900;

 if(wheel[next.num]==0){

 next.count=now.count+1;

 queue.push\_back(next);

 }

 }

 else{

 next.num=now.num-100;

 if(wheel[next.num]==0){

 next.count=now.count+1;

 queue.push\_back(next);

 }

 }

 if(now.num/1000==0){

 next.num=now.num+9000;

 if(wheel[next.num]==0){

 next.count=now.count+1;

 queue.push\_back(next);

 }

 }

 else{

 next.num=now.num-1000;

 if(wheel[next.num]==0){

 next.count=now.count+1;

 queue.push\_back(next);

 }

 }

 }

 wheel[now.num]=1;

 }

 if(found){

 found=false;

 answer[i]=now.count;

 }

 else{

 answer[i]=-1;

 }

 }

 for(int i=0;i<CASE\_NUM;i++){

 cout<<answer[i]<<endl;

 }

 return 0;

}