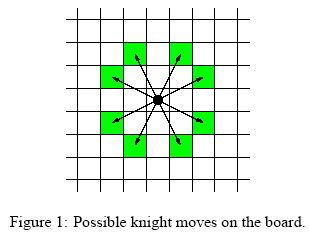
## poj 1915 Knight Moves

**题意：已知国际象棋中骑士的走法（具体看原图）。现给定棋盘大小与起始位置，问一个骑士从起点走到终点，至少要走几步。**

**思路：BFS。由于问的是最少步数，而且又是棋盘遍历题，所以优先考虑BFS。对于一个点，扫描其8个方向，若该方向上的下一个点未走过，则将该点入队，标记。最后就是检验一下终点的步数即可。由于是广度优先，所以第一次找到终点时的步数一定是最小的。**

****

**#include <iostream>**

**#include <queue>**

**#include<cstdio>**

**using namespace std;**

**typedef struct point{**

**int x,y;**

**int moves;**

**}point;**

**queue<point>Q;**

**int e\_x,e\_y,l;**

**int map[305][305];**

**int dir[8][2]={{-2,-1},{-2,1},{-1,2},{1,2},{2,1},{2,-1},{1,-2},{-1,-2}};**

**int main()**

**{**

**int bfs(point s);**

**int n,i,j,ans;**

**int s\_x,s\_y;**

**point start;**

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

**while(n--)**

**{**

**scanf("%d",&l);**

**scanf("%d%d%d%d",&s\_x,&s\_y,&e\_x,&e\_y);**

**for(i=0;i<l;i++)**

**for(j=0;j<l;j++)**

**map[i][j]=0;**

**while(!Q.empty())**

**Q.pop ();**

**start.x=s\_x;**

**start.y=s\_y;**

**start.moves=0;**

**ans=bfs(start);**

**printf("%d\n",ans);**

**}**

**return 0;**

**}**

**int bfs(point s)**

**{**

**int i;**

**map[s.x][s.y]=1;**

**point head,t;**

**Q.push(s);**

**while(!Q.empty ())**

**{**

**head=Q.front ();**

**Q.pop ();**

**if(head.x==e\_x&&head.y==e\_y) return head.moves;**

**for(i=0;i<8;i++)**

**{**

**int x=head.x+dir[i][0];**

**int y=head.y+dir[i][1];**

**if(x>=0&&x<l&&y>=0&&y<l&&!map[x][y])**

**{**

**map[x][y]=1;**

**t.x=x;**

**t.y =y;**

**t.moves =head.moves+1;**

**Q.push(t);**

**}**

**}**

**}**

**}**

## HDU2717-Catch That Cow(POJ3278)

**题意描述：一个人站在n点，只能向左走一步，或者向右走一步，或者直接走n的2倍步，要求走最少的步数抓住站在k点的牛。**

**题意分析：一看就是广搜的经典题目。一般要求最少步数用广搜。**

**#include<stdio.h>**

**#include<string.h>**

**#include<queue>**

**using namespace std;**

**#define MAX 200005**

**int v[MAX];**

**int cnt;**

**void BFS( int n, int k ){**

**queue<int>Q;**

**Q.push( n );**

**Q.push( cnt );**

**v[n] = 1;**

**while( !Q.empty() ){**

**n = Q.front();**

**Q.pop();**

**cnt = Q.front();**

**Q.pop();**

**if( n == k ){**

**return ;**

**}**

**if( n-1>=0 && !v[n-1] ){**

**Q.push( n-1 );**

**Q.push( cnt+1 );**

**v[n-1] = 1;**

**}**

**if( n+1<MAX && !v[n+1] ){**

**Q.push( n+1 );**

**Q.push( cnt+1 );**

**v[n+1] = 1;**

**}**

**if( 2\*n<MAX && !v[2\*n] ){**

**Q.push( 2\*n );**

**Q.push( cnt+1 );**

**v[2\*n] = 1;**

**}**

**}**

**}**

**int main(){**

**int n,k;**

**while( scanf( "%d%d", &n,&k ) != EOF ){**

**if( n >= k ){**

**printf( "%d\n",n-k );**

**continue;**

**}**

**cnt = 0;**

**memset( v, 0, sizeof(v) );**

**BFS( n, k );**

**printf( "%d\n", cnt );**

**}**

**return 0;**

**}**

## 非常可乐

**Problem Description**

**大家一定觉的运动以后喝可乐是一件很惬意的事情，但是seeyou却不这么认为。因为每次当seeyou买了可乐以后，阿牛就要求和seeyou一起分享这一瓶可乐，而且一定要喝的和seeyou一样多。但seeyou的手中只有两个杯子，它们的容量分别是N 毫升和M 毫升 可乐的体积为S （S<101）毫升　(正好装满一瓶) ，它们三个之间可以相互倒可乐 (都是没有刻度的，且 S==N+M，101＞S＞0，N＞0，M＞0) 。聪明的ACMER你们说他们能平分吗？如果能请输出倒可乐的最少的次数，如果不能输出"NO"。**

**Input**

**三个整数 : S 可乐的体积 , N 和 M是两个杯子的容量，以"0 0 0"结束。**

**Output**

**如果能平分的话请输出最少要倒的次数，否则输出"NO"。**

**Sample Input**

**7 4 3 4 1 3 0 0 0**

**Sample Output**

**NO 3**

**思路：预处理m < n < s，以后处理方便点**

**初始状态，m,n杯中可乐体积为0，s杯中体积为s;**

**然后分六种情况:**

**1, s 倒 m**

**2, s 倒 n**

**3, m 倒 n**

**4, m 倒 s**

**5, n 倒 m**

**6, n 倒 s**

**直到n,s杯中的可乐能等分（此时m杯中体积为0）为止，若不能等分，则输出 NO**

**#include<iostream>**

**#include<algorithm>**

**#include<queue>**

**using namespace std;**

**#define maxn 101**

**bool visited[maxn][maxn];**

**int m,n,s,si,sj;**

**struct node**

**{**

**int x,y,all,t; //x,y,all分别表示m,n,s杯中可乐的体积，t表示倒了多少次**

**};**

**void BFS()**

**{**

**queue<node> que;**

**memset(visited,false,sizeof(visited));**

**node p,q;**

**p.x = 0,p.y = 0,p.t = 0,p.all = s;**

**que.push(p);**

**visited[p.x][p.y] = true;**

**while(!que.empty())**

**{**

**p = que.front();**

**que.pop();**

**if(p.y == p.all && p.y == s/2)**

**{**

**printf("%d\n",p.t);**

**return;**

**}**

**if(p.all+p.x>m) //s倒m**

**{**

**q.x = m,q.y= p.y,q.all =p.all+p.x-m,q.t = p.t+1;**

**if(!visited[q.x][q.y])**

**que.push(q),visited[q.x][q.y] = true;**

**}**

**else**

**{**

**q.x = p.all+p.x,q.y = p.y,q.all = 0,q.t = p.t+1;**

**if(!visited[q.x][q.y])**

**que.push(q),visited[q.x][q.y] = true;**

**}**

**if(p.all+p.y>n) //s倒n**

**{**

**q.x= p.x, q.y = n, q.all = p.all+p.y-n,q.t = p.t+1;**

**if(!visited[q.x][q.y])**

**que.push(q),visited[q.x][q.y] = true;**

**}**

**else**

**{**

**q.x = p.x,q.y = p.all+p.y,q.all = 0,q.t = p.t+1;**

**if(!visited[q.x][q.y])**

**que.push(q),visited[q.x][q.y] = true;**

**}**

**if(p.x+p.y>n) //m倒n**

**{**

**q.x = p.x+p.y-n,q.y = n,q.all = p.all,q.t = p.t+1;**

**if(!visited[q.x][q.y])**

**que.push(q),visited[q.x][q.y] = true;**

**}**

**else**

**{**

**q.x = 0,q.y = p.x+p.y,q.all = p.all,q.t = p.t+1;**

**if(!visited[q.x][q.y])**

**que.push(q),visited[q.x][q.y] = true;**

**}**

**q.all =p.all+p.x,q.x = 0,q.y = p.y,q.t = p.t+1; //m倒s**

**if(!visited[q.x][q.y])**

**que.push(q),visited[q.x][q.y] = true;**

**if(p.x+p.y > m)**

**{**

**q.y=p.y+p.x-m,q.x=m,q.all=p.all,q.t = p.t+1;//n倒m**

**if(!visited[q.x][q.y])**

**que.push(q),visited[q.x][q.y] = true;**

**}**

**else**

**{**

**q.x = p.x+p.y,q.y = 0,q.all = p.all,q.t = p.t+1;**

**if(!visited[q.x][q.y])**

**que.push(q),visited[q.x][q.y] = true;**

**}**

**q.all = p.all+p.y,q.x = p.x,q.y = 0,q.t = p.t+1; //n倒s**

**if(!visited[q.x][q.y])**

**que.push(q),visited[q.x][q.y] = true;**

**}**

**printf("NO\n");**

**}**

**int main()**

**{**

**//freopen("1013.txt","r",stdin);**

**while(scanf("%d%d%d",&s,&m,&n) && (s||m||n))**

**{**

**if(s%2)**

**{**

**printf("NO\n");**

**continue;**

**}**

**if(m > n) swap(m,n);**

**BFS();**

**}**

**return 0;**

**}**

## 3126 Prime Path

**题意： 给定两个素数（四位数），求第一个数经过几次转换能够得到第二个素数。转换方式：是变换数中某一位的数字（第一位不能为零，其他的变换数字是0~~9），变换之后的数也为素数。**

**#include <iostream>**

**#include <cstdio>**

**#include <cstring>**

**#include <cstdlib>**

**#include <cmath>**

**#include <queue>**

**#define N 10005**

**using namespace std;**

**int isPrime[N];**

**int v[N];**

**typedef struct**

**{**

**int bit[5];**

**int num;**

**int s;**

**} Node;**

**void Solve()**

**{**

**int i,j;**

**for(i=0;i<N;i++)**

**isPrime[i]=1;**

**isPrime[0]=isPrime[1]=0;**

**for(i=2;i<N;i++)**

**{**

**if(isPrime[i]==1)**

**{**

**for(j=i+i;j<N;j+=i)**

**{**

**isPrime[j]=0;**

**}**

**}**

**}**

**}**

**int getNum(int a[],int b,int c)**

**{**

**int ans=0;**

**for(int i=0;i<4;i++)**

**{**

**if(i==b)**

**ans = ans\*10+c;**

**else**

**ans = ans\*10+a[i];**

**}**

**return ans;**

**}**

**void bfs(int first,int second)**

**{**

**queue< Node > ac;**

**int i,j,k;**

**memset(v,0,sizeof(v));**

**Node front ,rear;**

**front.num = first;**

**front.bit[3] = first%10;**

**front.bit[2] = (first/10)%10;**

**front.bit[1] = (first/100)%10;**

**front.bit[0] = (first/1000)%10;**

**front.s=0;**

**ac.push(front);**

**while(!ac.empty())**

**{**

**rear = ac.front();**

**ac.pop();**

**if(rear.num == second)**

**{**

**printf("%d\n",rear.s);**

**break;**

**}**

**for(i=0;i<4;i++)**

**{**

**for(j=0;j<=9;j++)**

**{**

**if(j==0&&i==0) continue;**

**if(rear.bit[i] == j)**

**continue;**

**else**

**{**

**int inter = getNum(rear.bit,i,j);**

**if(isPrime[inter] 1&& !v[inter] )**

**{**

**v[inter]=1;**

**for(k=0;k<4;k++)**

**{**

**front.bit[k]=rear.bit[k];**

**}**

**front.bit[i]=j;**

**front.num = inter;**

**front.s = rear.s+1;**

**ac.push(front);**

**}**

**}**

**}**

**}**

**}**

**}**

**int main()**

**{**

**int ncase;**

**int i;**

**int first,second;**

**//freopen("in.txt","r",stdin);**

**Solve();**

**scanf("%d",&ncase);**

**while(ncase--)**

**{**

**scanf("%d%d",&first,&second);**

**bfs(first,second);**

**}**

**return 0;**

**}**