**所有的代码均已上传到GitHub：**

**https://github.com/MrsSun/JiaShi.git**

1.请用java代码解决以下问题。

   一位老农带着猫、狗、鱼过河，河边有一条船，

\* 每次老农只能带一只动物过河。当老农不和猫狗鱼在一起时，  
\* 狗会咬猫，猫会吃鱼，当老农和猫狗鱼在一起时，  
\* 则不会发生这种问题。编程解决猫狗鱼过河问题。

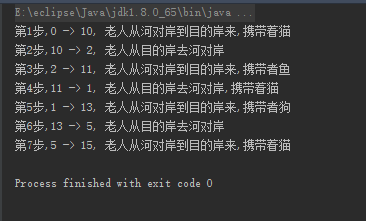
**解答：**

**老人过河问题，首先可以用0或1表示各者的状态，0表示在目的岸边，1表示在对岸。**

**其次，老人每次过河其实都只有四种选择，用状态来表示的话，分别为1000（只用老人自己），1100（携带着狗），1010（携带着猫），1001（携带着鱼）。**

**很明显这就是一个搜索问题，这里用广度搜索更合适。广度搜索得借助队列来实现，计算出下一个状态存入队列中，依次出队判断是否安全，如果安全，计算出此状态的下一个状态存入队列，直至队列为空，或者达到1111安全状态。**

**代码如下（仅一部分）：**

**/\*\*  
 \* Created by chaoge on 2017/6/12.  
 \*/  
import animal.\*;  
import com.sun.jmx.remote.internal.ArrayQueue;  
  
import java.util.ArrayList;  
import java.util.LinkedList;  
import java.util.Queue;  
  
/\*  
 用四位二进制表示岸边的状态，从最高位起，每一位分别表示老人，狗，猫，鱼  
 以目的岸边为基准，即从0000开始，找到一个安全序列，使得能够达到1111  
 算法设计：采用广度优先算法，借助队列存储状态实现  
 \*/  
public class RiverCross {  
 //定义一个静态常量表示状态的个数  
 private static final int statusLen = 16;  
 //用来表示各个状态是否已经出现过  
 private int[] visted = new int[statusLen];//默认0为未曾出现过  
 //每次过河都有4种选择，安不安全合不合适交给isSafe函数去判断  
 private int choice[] = {8,9,10,12};  
 //用队列来存储路径，因为一共就16中状态，所以队列容量为16足以  
 private ArrayQueue<Integer> queue = new ArrayQueue<Integer>(statusLen);  
 //存储路径  
 private int route[] = new int[statusLen];  
 private animal[] animals = {new man(),new dog(),new cat(),new fish()};  
  
 public void print(){  
 int []result = RiverCross();  
 int j = 0;  
 int count = 0;  
 while (j < 16 ){  
 System.out.print("第"+ ++count + "步,");  
 System.out.print(j + " -> " + result[j]+", ");  
 for (animal a:animals) {  
 if(a instanceof man)  
 a.getDescribe(result[j]);  
 else  
 a.getDescribe((result[j]^j));  
 }  
 System.out.println();  
 j = result[j];  
 if(j == 15)  
 break;  
 }  
 }  
 //初始化相关数组队列等  
 public void Init(){  
 for(int i = 0; i < statusLen; i++){  
 route[i] = -1;  
 visted[i] = 0;  
 }  
 queue.add(0);//初始化队列，表示目的岸边的状态为0  
 visted[0] = 1;  
 }  
 //过河  
 public int[] RiverCross(){  
 Init();  
 int choiceLen = choice.length;  
 int nextStatus;  
 int result[] = new int[statusLen];  
 while (!queue.isEmpty()){//当队列非空  
 //取出对头元素，这里的remove中的参数只能为0  
 int status = queue.remove(0);  
 for(int i = 0; i < choiceLen; i++){  
 //用异或就能保证过河的次序是来还是去  
 nextStatus = status ^ choice[i];  
 if(visted[nextStatus]==0 && isSafe(nextStatus)){  
 visted[nextStatus] = 1;  
 route[nextStatus] = status;  
 if(nextStatus == 15)  
 break;  
 queue.add(nextStatus);  
 }  
 }  
 }  
 if(visted[statusLen-1] == 0)  
 return null;  
 //逆序遍历，找出路径存入result中  
 int j = 15;  
 while (j != -1 && route[j]!=-1){  
// System.out.print("j == "+j+" temp[j]== "+temp[j]+", ");  
 result[route[j]] = j;  
 j = route[j];  
 }  
 return result;  
  
 }  
  
 //判断下一个状态是否安全  
 public boolean isSafe(int status){  
 /\*  
 经过归纳总结可以发现，狗和猫如果状态一样的话，那么跟老人的状态也必须一样，  
 否则两岸肯定会有一边处于不安全的状态。同理可知猫和鱼。  
 \*/  
 if(animals[1].isExists(status)==animals[2].isExists(status) &&  
 animals[1].isExists(status) != animals[0].isExists(status))  
 return false;  
 if(animals[2].isExists(status)==animals[3].isExists(status) &&  
 animals[2].isExists(status) != animals[0].isExists(status))  
 return false;  
 return true;  
 }  
  
}**

2.sql题（交付时需包含完整建表语句，插入语句，查询语句）

    把这样一个表儿

year   month amount  
1991   1     1.1  
1991   2     1.2  
1991   3     1.3  
1991   4     1.4  
1992   1     2.1  
1992   2     2.2  
1992   3     2.3  
1992   4     2.4  
查成这样一个结果  
year m1    m2   m3   m4  
1991 1.1  1.2  1.3  1.4  
1992 2.1  2.2  2.3  2.4

**解答：**

**1.1首先建表**

use WebTest4

go

create proc CreateMyTable

as

create table MyData(

year int not null,

month int ,

amount float

)

insert into MyData values

(1991,1,1.1),

(1991,2,1.2),

(1991,3,1.3),

(1991,4,1.4),

(1992,1,1.1),

(1992,2,1.2),

(1992,3,1.3),

(1992,4,1.4)

go

**1.2查询**

**两种方法，可以用自连接的形式，亦可以用case判断**

use WebTest4

go

--用存储过程创建表格

--exec CreateMyTable

--select year,

--case month

-- when 1 then 'm1' end as m1

--select year as year,

-- (select SUM(amount) from MyData b where a.year = b.year and b.month = 1)as m1,

-- (select SUM(amount) from MyData b where a.year = b.year and b.month = 2)as m2,

-- (select SUM(amount) from MyData b where a.year = b.year and b.month = 3)as m3,

-- (select SUM(amount) from MyData b where a.year = b.year and b.month = 4)as m4,

-- (select SUM(amount) from MyData b where a.year = b.year and b.month = 5)as m5

-- from MyData a group by year

select year as year,

SUM(case when month =1 then amount else 0 end) as m1,

SUM(case when month =2 then amount else 0 end) as m2,

SUM(case when month =3 then amount else 0 end) as m3,

SUM(case when month =4 then amount else 0 end) as m4,

SUM(case when month =5 then amount else 0 end) as m5,

SUM(case when month =6 then amount else 0 end) as m6,

SUM(case when month =7 then amount else 0 end) as m7,

SUM(case when month =8 then amount else 0 end) as m8,

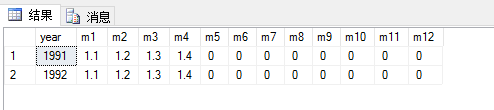
SUM(case when month =9 then amount else 0 end) as m9,

SUM(case when month =10 then amount else 0 end) as m10,

SUM(case when month =11 then amount else 0 end) as m11,

SUM(case when month =12 then amount else 0 end) as m12

from MyData group by year



1. 给定10亿个不重复的unsigned int的整数，没排过序的，然后随机给一个数，快速判断这个数是否在那10亿个数当中 。给出伪代码

**解答：**

**看到这个题目，想了很多策略。首先想到排序，但就算就稳定的归并排序，时间复杂度以Nlog2N 的复杂度来讲还是很耗费时间，而且额外开销很大。**

**接着又想到了散列表，根据散列函数计算出给出数的hashcode，再在对应的hashcode的链表中进行查找，还是耗内存。**

**问题关键是内存问题，所以首先要解决的便是它的内存消耗问题。**

**可以这么想，可以用类似位图的形式存储这些数，每一位0或1代表着对应的数是否存在。本来是需要消耗10 ^ 9 \* 4 字节 = 4G，现在只需要2^32 / 8字节 = 512M，（注意内存的计算是用int的取值范围来算，而不是用题目中给出数字的个数）可见内存消耗极大的降低。**

**以下便是伪代码：**

**/\*\*  
 \* Created by chaoge on 2017/6/14.  
 \*/  
public class Bit {  
 //定义一个二维数组用来存储数的位置  
 private int size = 2^32/32;  
 private Bit position[32][size];  
  
 //计算数的位置，然后存储在二维数组中  
 public void stroPosition(int num){  
 row = num >> 5; //相当于除以32  
 col = num & 31; //相当于对32求余  
 position[row][col] = 1;  
 }  
 public boolean isExists(int num){  
 row = num >> 5; //相当于除以32  
 col = num & 31; //相当于对32求余  
 position[row][col] == 1 ? return true : return false;  
 }  
 //初始化  
 public void init(){  
 while(true){  
 int num = input(); //依次输入是十亿个数  
 stroPosition(num);//存储是否出现的信息  
 }  
 }  
 public static void main(String [] args){  
 Bit bit = new Bit();  
 bit.init();  
 int num = bit.input();  
 if(bit.isExists(num))  
 System.out.println("存在这个数");  
 else  
 System.out.println("不存在这个数");  
 }  
}**

1. 给出一个能设计到最全面的权限系统的ER模型。

抽象出权限系统中的各个实体，以及各自的属性，实体之间的联系。。。

直接上图：

