备战ACM资料

一：知识点

数据结构：

1，单，双链表及循环链表

2，树的表示与存储，二叉树（概念，遍历）二叉树的

应用（二叉排序树，判定树，博弈树，解答树等）

3，文件操作（从文本文件中读入数据并输出到文本文

件中）

4，图（基本概念，存储结构，图的运算）

数学知识

1，离散数学知识的应用（如排列组合、简单的图论，数

理逻辑）

2，数论知识

3，线性代数

4，组合代数

5，计算几何

二 算法

1，排序算法（冒抛法，插入排序，合并排序，快速排

序，堆排序）

2，查找（顺序查找，二分发）

3，回溯算法

4，递归算法

5，分治算法

6，模拟法

7，贪心法

8，简单搜索算法（深度优先，广度优先），搜索中的

剪枝，A\*算法

9，动态规划的思想及基本算法

10，高精度运算

三、ACM竞赛的题型分析

竞赛的程序设计一般只有16种类型，它们分别是：

Dynamic Programming （动态规划）

Greedy （贪心算法）

Complete Search （穷举搜索）

Flood Fill （不知该如何翻译）

Shortest Path （最短路径）

Recursive Search Techniques （回溯搜索技术）

Minimum Spanning Tree （最小生成树）

Knapsack （背包问题）

Computational Geometry (计算几何学)

Network Flow （网络流）

Eulerian Path （欧拉回路）

Two-Dimensional Convex Hull （不知如何翻译）

BigNums （大数问题）

Heuristic Search （启发式搜索）

Approximate Search （近似搜索）

Ad Hoc Problems （杂题）

四 ACM竞赛参考书

《实用算法的分析与程序设计》 （吴文虎，王建德著，电子工业出版社，竞赛类的黑宝书）

《青少年国际和全国信息学（计算机）奥林匹克竞赛指导）――组合数学的算法

和程序设计》（吴文虎，王建德著，清华大学出版社，参加竞赛组合数学必学）

《计算机算法设计与分析》 （王晓东编著，最好的数据结构教材）

《数据结构与算法》 （傅清祥，王晓东编著，我所见过的最好的算法教材）

《信息学奥林匹克竞赛指导――1997-1998竞赛试题解析》（吴文虎，王建德著，清华大学出版社）

《计算机程序设计技巧》 D.E.Kruth著，算法书中最著名的《葵花宝典》，大师的作品，难度大）

《计算几何》周陪德著

《ACM国际大学生程序设计竞赛试题与解析（一）》 （吴文虎著，清华大学出版社）

《数学建模竞赛培训教材》 共三本 叶其孝主编

《数学模型》 第二版 姜启源

《随机规划》

《模糊数学》

《数学建模入门》 徐全智

《计算机算法设计与分析》 国防科大

五 常见的几个网上题库

常用网站：

1）信息学初学者之家：http://oibh.ioiforum.org/

（2）大榕树编程世界：http://www.fjsdfz.org/~drs/program/default.asp

（3）中国教育曙光网：http://www.chinaschool.org/aosai/

（4）福建信息学奥林匹克：http://www.cfcs.com.cn/fjas/index.htm

（5）第20届全国青少年信息学奥林匹克竞赛：http://www.noi2003.org/

（6）第15届国际青少年信息学奥林匹克竞赛：http://www.ioi2003.org/

（7）全美计算机奥林匹克竞赛：http://ace.delos.com/usacogate

（8）美国信息学奥林匹克竞赛官方网站：http://www.usaco.org/

（9）俄罗斯Ural州立大学：http://acm.timus.ru/

（10）西班牙Valladolid大学：http://acm.uva.es/problemset

（11）ACM-ICPC：http://icpc.baylor.edu/icpc/

（12）北京大学：http://acm.pku.edu.cn/JudgeOnline/index.acm

（13）浙江大学：http://acm.zju.edu.cn/

（14）IOI：http://olympiads.win.tue.nl/ioi/

（15）2003年江苏省信息学奥林匹克竞赛夏令营：http://jsoi.czyz.com.cn

（16）http://acm.zju.edu.cn

（17）http://acm.zsu.edu.cn

（18）www.shumo.com

（19）http://www.bepark.com/downldmanag/index.asp

（20）http://www.yh01.com colin\_fox/colin\_fox

五 如何备战ACM/ICPC

1，个人准备（算法书，习题集，网上做题和讨论）

2，1000题=亚洲冠军=世界决赛

3，做好资料收集和整理工作

实验一：递归与分治

1. 二分查找

2. 合并排序

3. 快速排序

实验二：回溯

1. 0-1背包问题

2. 装载问题

3. 堡垒问题（ZOJ1002）

4. \*翻硬币问题

5. 8皇后问题

6. 素数环问题

7. 迷宫问题

8. \*农场灌溉问题（ZOJ2412）

9. \*求图像的周长（ZOJ1047）

10. \*骨牌矩阵

11. \*字母转换（ZOJ1003）

12. \*踩气球（ZOJ1004）

实验三：搜索

1. Floodfill

2. 电子老鼠闯迷宫

3. 跳马

4. 独轮车

5. 皇宫小偷

6. 分酒问题

7. \*找倍数

8. \*8数码难题

实验四：动态规划

1. 最长公共子序列

2. 计算矩阵连乘积

3. 凸多边形的最优三角剖分

4. 防卫导弹

5. \*石子合并

6. \*最小代价子母树

7. \*旅游预算

8. \*皇宫看守

9. \*游戏室问题

10. \*基因问题

11. \*田忌赛马

实验五：贪心与随机算法

1. 背包问题

2. 搬桌子问题

3. \*照亮的山景

4. \*用随即算法求解8皇后问题

5. 素数测试

实验一：递归与分治

实验目的

理解递归算法的思想和递归程序的执行过程，并能熟练编写递归程序。

掌握分治算法的思想，对给定的问题能设计出分治算法予以解决。

实验预习内容

编程实现讲过的例题：二分搜索、合并排序、快速排序。

对本实验中的问题，设计出算法并编程实现。

试验内容和步骤

1． 二分查找

在对线性表的操作中，经常需要查找某一个元素在线性表中的位置。此问题的输入是待查元素x和线性表L，输出为x在L中的位置或者x不在L中的信息。

程序略

2． 合并排序

程序略

3． 快速排序

程序略

实验总结及思考

合并排序的递归程序执行的过程

实验二：回溯算法

实验目的：熟练掌握回溯算法

实验内容：回溯算法的几种形式

a) 用回溯算法搜索子集树的一般模式

void search(int m)

{

if(m>n) //递归结束条件

output(); //相应的处理(输出结果)

else

{

a[m]=0; //设置状态：0表示不要该物品

search(m+1); //递归搜索：继续确定下一个物品

a[m]=1; //设置状态：1表示要该物品

search(m+1); //递归搜索：继续确定下一个物品

}

}

b) 用回溯算法搜索子集树的一般模式

void search(int m)

{

if(m>n) //递归结束条件

output(); //相应的处理(输出结果)

else

for(i=m;i<=n;i++)

{

swap(m,i); //交换a[m]和a[i]

if()

if(canplace(m)) //如果m处可放置

search(m+1); //搜索下一层

swpa(m,i); //交换a[m]和a[i](换回来)

}

}

习题

1． 0-1背包问题

在0 / 1背包问题中，需对容量为c 的背包进行装载。从n 个物品中选取装入背包的物品，每件物品i 的重量为wi ，价值为pi 。对于可行的背包装载，背包中物品的总重量不能超过背包的容量，最佳装载是指所装入的物品价值最高。

程序如下：

#include <stdio.h>

void readdata();

void search(int);

void checkmax();

void printresult();

int c=35, n=10; //c： 背包容量；n：物品数

int w[10], v[10]; //w[i]、v[i]：第i件物品的重量和价值

int a[10], max; //a数组存放当前解各物品选取情况；max：记录最大价值

//a[i]=0表示不选第i件物品，a[i]=1表示选第i件物品

int main()

{

readdata(); //读入数据

search(0); //递归搜索

printresult();

}

void search(int m)

{

if(m>=n)

checkmax(); //检查当前解是否是可行解，若是则把它的价值与max比较

else

{

a[m]=0; //不选第m件物品

search(m+1); //递归搜索下一件物品

a[m]=1; //不选第m件物品

search(m+1); //递归搜索下一件物品

}

}

void checkmax()

{

int i, weight=0, value=0;

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

{

if(a[i]==1) //如果选取了该物品

{

weight = weight + w[i]; //累加重量

value = value + v[i]; //累加价值

}

}

if(weight<=c) //若为可行解

if(value>max) //且价值大于max

max=value; //替换max

}

void readdata()

{

int i;

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

scanf("%d%d",&w[i],&v[i]); //读入第i件物品重量和价值

}

void printresult()

{

printf("%d",max);

}

2． 装载问题

有两艘船，载重量分别是c1、 c2，n个集装箱，重量是wi (i=1…n)，且所有集装箱的总重量不超过c1+c2。确定是否有可能将所有集装箱全部装入两艘船。

提示：求出不超过c1的最大值max，若总重量－max < c2则能装入到两艘船。

3． 堡垒问题（ZOJ1002）

如图城堡是一个4×4的方格，为了保卫城堡，现需要在某些格子里修建一些堡垒。城堡中的某些格子是墙，其余格子都是空格，堡垒只能建在空格里，每个堡垒都可以向上下左右四个方向射击，如果两个堡垒在同一行或同一列，且中间没有墙相隔，则两个堡垒都会把对方打掉。问对于给定的一种状态，最多能够修建几个堡垒。

程序主要部分如下：

int main()

{

readdata(); //读入数据

search(0); //递归搜索

printresult();

}

void search(int m)

{

int row, col;

row=m/n; //求第m个格子的行号

col=m%n; //求第m个格子的列号

if(m>=n\*n)

checkmax(); //检查当前解是否是可行解，若是则把它的价值与max比较

else

{

search(m+1); //该位置不放堡垒递归搜索下一个位置

if(canplace(m)) //判断第m个格子是否能放堡垒

{

place(m); //在第m个格子上放置一个堡垒

search(m+1); //递归搜索下一个位置

takeout(m); //去掉第m个格子上放置的堡垒

}

}

}

4． 翻硬币问题

把硬币摆放成32×9的矩阵，你可以随意翻转矩阵中的某些行和某些列，问正面朝上的硬币最多有多少枚？

提示：（1）任意一行或一列，翻两次等于没有翻；

（2）对于9列的任何一种翻转的情况，每一行翻与不翻相互独立。

5． 8皇后问题

在一个８×８的棋盘里放置８个皇后，要求这8个皇后两两之间互相都不“冲突”。

#include <stdio.h>

#include <math.h>

void search(int);

void printresult(); //打印结果

int canplace(int,int); //判断该位置能否放置皇后

void place(int,int); //在该位置能否放置皇后

void takeout(int,int); //把该位置放置皇后去掉

int a[8]; //a[i]存放第i个皇后的位置

int main()

{

search(0); //递归搜索

}

void search(int m)

{

int i;

if(m>=8) //当已经找出一组解时

printresult(); //输出当前结果

else

{

for(i=0;i<8;i++) //对当前行0到7列的每一个位置

{

if(canplace(m,i)) //判断第m个格子是否能放堡垒

{

place(m,i); //在(m,i)格子上放置一个皇后

search(m+1); //递归搜索下一行

takeout(m,i); //把(m,i)格子上的皇后去掉

}

}

}

}

int canplace(int row, int col)

{

int i;

for(i=0;i<row;i++)

if(abs(i-row)==abs(a[i]-col)||a[i]==col)

return(0);

return(1);

}

void place(int row, int col)

{

a[row]=col;

}

void takeout(int row, int col)

{

a[row]=-1;

}

void printresult()

{

int i,j;

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

{

for(j=0;j<8;j++)

if(a[i]==j)

printf(" A ");

else

printf(" . ");

printf("\n");

}

printf("\n");

}

6． 素数环问题

把从1到20这20个数摆成一个环，要求相邻的两个数的和是一个素数。

分析：用回溯算法，考察所有可能的排列。

程序如下：

#include <stdio.h>

#include <math.h>

void search(int);

void init(); //初始化

void printresult(); //打印结果

int isprime(int); //判断该数是否是素数

void swap(int,int); //交换a[m]和a[i]

int a[21]; //a数组存放素数环

int main()

{

init();

search(2); //递归搜索

}

int isprime(int num)

{

int i,k;

k=sqrt(num);

for(i=2;i<=k;i++)

if(num%i==0)

return(0);

return(1);

}

void printresult()

{

int i;

for(i=1;i<=20;i++)

printf("%3d",a[i]);

printf("\n");

}

void search(int m)

{

int i;

if(m>20) //当已经搜索到叶结点时

{

if(isprime(a[1]+a[20])) //如果a[1]+a[20]也是素数

printresult(); //输出当前解

return;

}

else

{

for(i=m;i<=20;i++) //(排列树)

{

swap(m,i); //交换a[m]和a[i]

if(isprime(a[m-1]+a[m])) //判断a[m-1]+a[m]是否是素数

search(m+1); //递归搜索下一个位置

swap(m,i); //把a[m]和a[i]换回来

}

}

}

void swap(int m, int i)

{

int t;

t=a[m];

a[m]=a[i];

a[i]=t;

}

void init()

{

int i;

for(i=0;i<21;i++)

a[i]=i;

}

7． 迷宫问题

给一个20×20的迷宫、起点坐标和终点坐标，问从起点是否能到达终点。

输入数据：’.’表示空格；’X’表示墙。

程序如下：

#include <stdio.h>

#include <math.h>

void search(int,int);

int canplace(int,int);

void readdata(); //读入数据

void printresult(); //打印结果

int a[20][20]; //a数组存放迷宫

int s,t;

int main()

{

int row, col;

readdata();

row=s/20;

col=s%20;

search(row,col); //递归搜索

printresult();

}

void search(int row, int col)

{

int r,c;

a[row][col]=1;

r=row; //左

c=col-1;

if(canplace(r,c)) //判断(r,c)位置是否已经走过

search(r,c); //递归搜索(r,c)

r=row+1; //下

c=col;

if(canplace(r,c)) //判断(r,c)位置是否已经走过

search(r,c); //递归搜索(r,c)

r=row; //右

c=col+1;

if(canplace(r,c)) //判断(r,c)位置是否已经走过

search(r,c); //递归搜索(r,c)

r=row-1; //上

c=col;

if(canplace(r,c)) //判断(r,c)位置是否已经走过

search(r,c); //递归搜索(r,c)

}

void printresult()

{

int i,j;

for(i=0;i<20;i++)

{

for(j=0;j<20;j++)

printf("%3d",a[i][j]);

printf("\n");

}

}

void readdata()

{

int i,j;

for(i=0;i<20;i++)

{

for(j=0;j<20;j++)

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

}

}

int canplace(int row, int col)

{

if(row>=0&&row<20&&col>=0&&col<20&&a[row][col]==0)

return 1;

else

return 0;

}

8． 农场灌溉问题（ZOJ2412）

一农场由图所示的十一种小方块组成，蓝色线条为灌溉渠。若相邻两块的灌溉渠相连则只需一口水井灌溉。给出若干由字母表示的最大不超过50×50具体由(m，n)表示，的农场图，编程求出最小需要打的井数。每个测例的输出占一行。当M=N=-1时结束程序。

Sample Input

2 2

DK

HF

3 3

ADC

FJK

IHE

-1 -1

Sample Output

2

3

提示：参考迷宫问题，实现时关键要解决好各块的表示问题。

9． 求图像的周长（ZOJ1047）

给一个用 . 和X表示的图形，图形在上、下、左、右、左上、左下、右上、右下8个方向都被看作是连通的，并且图像中间不会出现空洞，求这个图形的边长。

输入：首先给出m、n、x、y四个正整数，下面给出m×n的图形，x、y表示点击的位置，全0表示结束。

输出：点击的图形的周长。

Sample Input

2 2 2 2

XX

XX

6 4 2 3

.XXX

.XXX

.XXX

...X

..X.

X...

0 0 0 0

Sample output

8

18

提示：参考迷宫问题，区别在于它是向8个方向填。

10． 骨牌矩阵

多米诺骨牌是一个小正方形方块，每个骨牌都标有一个数字（0~6），现在有28组骨牌，每组两个，各组编号为1~28，每组编号对应的两个骨牌数值如下：

00 01 02 03 04 05 06

11 12 13 14 15 16 22

23 24 25 26 33 34 35

36 44 45 46 55 56 66

现将这28组骨牌排成一个7×8矩阵，此时只能看到每个骨牌上的数字（0~6），而不能知道每组的组号（如左下图所示）。请编程序将每组骨牌分辨出来（如右下图所示）。

7X8骨牌矩阵 骨牌组编号矩阵

66265241 28 28 14 7 17 17 11 11

13201034 10 10 14 7 2 2 21 23

13246654 8 4 16 25 25 13 21 23

10432112 8 4 16 15 15 13 9 9

51360455 12 12 22 22 5 5 26 26

55402603 27 24 24 3 3 18 1 19

60534203 27 6 6 20 20 18 1 19

void search(int n)

{

查找下一个还没放置骨牌的位置(x,y);

若没有，则表示已经找到一个解，输出并且返回；

尝试放置骨牌；

两次尝试都失败，进行回溯；

}

尝试放置骨牌

? 把在(x,y)处的骨牌作为当前骨牌组的一个骨牌；

? 把(x+1,y)处的骨牌作为当前骨牌组的另一个骨牌；

? 判断当前骨牌组是够未被使用，如果未被使用则递归放置下一个骨牌组；

? 把(x,y +1)处的骨牌作为当前骨牌组的另一个骨牌；

? 判断当前骨牌组是否未被使用，如果未被使用则递归放置下一个骨牌组；

11． 字母转换（ZOJ1003）

通过栈交换字母顺序。给定两个字符串，要求所有的进栈和出栈序列（i表示进栈，o表示出栈），使得字符串2在求得的进出栈序列的操作下，变成字符串1。输出结果需满足字典序。例如TROT 到 TORT:

[

i i i i o o o o

i o i i o o i o

]

Sample Input

madam

adamm

bahama

bahama

long

short

eric

rice

Sample Output

[

i i i i o o o i o o

i i i i o o o o i o

i i o i o i o i o o

i i o i o i o o i o

]

[

i o i i i o o i i o o o

i o i i i o o o i o i o

i o i o i o i i i o o o

i o i o i o i o i o i o

]

[

]

[

i i o i o i o o

]

12． 踩气球（ZOJ1004）

六一儿童节，小朋友们做踩气球游戏，气球的编号是1～100，两位小朋友各踩了一些气球，要求他们报出自己所踩气球的编号的乘积。现在需要你编一个程序来判断他们的胜负，判断的规则是这样的：如果两人都说了真话，数字大的人赢；如果两人都说了假话，数字大的人赢；如果报小数字的人说的是真话而报大数字的人说谎，则报小数字的人赢（注意：只要所报的小数字是有可能的，即认为此人说了真话）。

输入为两个数字，0 0表示结束；

输出为获胜的数字。

Sample Input

36 62

49 343

0 0

Sample Output

62

49

实验三：搜索算法

实验目的：熟练掌握搜索算法

实验内容：广度优先搜索

搜索算法的一般模式：

void search()

{

closed表初始化为空;

open表初始化为空;

起点加入到open表;

while( open表非空 )

{

取open表中的一个结点u;

从open表中删除u;

u进入closed表;

for( 对扩展结点u得到的每个新结点vi )

{

if(vi是目标结点)

输出结果并返回;

if vi 的状态与closed表和open表中的结点的状态都不相同

vi进入open表;

}

}

}

搜索算法关键要解决好状态判重的问题，这样可省略closed表，一般模式可改为：

void search()

{

open表初始化为空;

起点加入到open表;

while( open表非空 )

{

取open表中的一个结点u;

从open表中删除u;

for( 对扩展结点u得到的每个新结点vi )

{

if(vi是目标结点)

输出结果并返回;

If(notused(vi))

vi进入open表;

}

}

}

1. Floodfill

给一个20×20的迷宫和一个起点坐标，用广度优先搜索填充所有的可到达的格子。

提示：参考第2题。

2. 电子老鼠闯迷宫

如下图12×12方格图，找出一条自入口（2，9）到出口（11，8）的最短路径。

本题给出完整的程序和一组测试数据。状态：老鼠所在的行、列。程序如下：

#include<stdio.h>

void readdata(); //读入数据

void init(); //初始化

int search(); //广搜，并在每一个可到达的每一个空格出填上最小步数

int emptyopen(); //判栈是否为空：空：1；非空：0。

int takeoutofopen(); //从栈中取出一个元素，并把该元素从栈中删除

int canmoveto(int,int,int\*,int\*,int); //判能否移动到该方向，并带回坐标(r,c)

int isaim(int row, int col); //判断该点是否是目标

int used(int,int); //判断该点是否已经走过

void addtoopen(int,int); //把该点加入到open表

int a[12][12]; //a存放迷宫，0表示空格，-2表示墙。

//广搜时，未找到目标以前到达的空格，填上到达该点的最小步数

int n; //n为迷宫边长，注：若大于12，必须修改一些参数，如a的大小

int open[20],head,tail,openlen=20; //open表

int s,t; //起点和终点

int main()

{

int number;

readdata(); //读取数据

init(); //初始化

number=search(); //广搜并返回最小步数

printf("%d",number); //打印结果

}

int search()

{

int u, row, col, r, c, i, num;

while(!emptyopen()) //当栈非空

{

u=takeoutofopen(); //从栈中取出一个元素，并把该元素从栈中删除

row=u/n; //计算该点的坐标

col=u%n;

num=a[row][col]; //取得该点的步数

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

{

if(canmoveto(row,col,&r,&c,i)) //判能否移动到该方向，并带回坐标(r,c)

{

if(isaim(r,c)) //如果是目标结点

return(num+1); //返回最小步数

if(!used(r,c)) //如果(r,c)还未到达过

{

a[r][c]=num+1; //记录该点的最小步数

addtoopen(r,c); //把该点加入到open表

}

}

}

}

}

int emptyopen()

{

if(head==tail)

return(1);

else

return(0);

}

int takeoutofopen()

{

int u;

if(head==tail)

{

printf("errer: stack is empty");

return(-1);

}

u=open[head++];

head=head%openlen;

return(u);

}

int canmoveto(int row, int col, int \*p, int \*q, int direction)

{

int r,c;

r=row;

c=col;

switch(direction)

{

case 0: c--; //左

break;

case 1: r++; //下

break;

case 2: c++; //右

break;

case 3: r--; //上

}

\*p=r;

\*q=c;

if(r<0||r>=n||c<0||c>=n) //如果越界返回0

return(0);

if(a[r][c]==0) //如果是空格返回1

return(1);

return(0); //其余情况返回0

}

int isaim(int row, int col)

{

if(row\*n+col==t)

return(1);

else

return(0);

}

int used(int row, int col)

{

if(a[row][col]==0) // 0表示空格

return(0);

else

return(1);

}

void addtoopen(int row, int col)

{

int u;

u=row\*n+col;

open[tail++]= u;

tail=tail%openlen;

}

void readdata()

{

int i,j,row,col;

char str[20];

scanf("%d",&n);

scanf("%d%d",&row,&col); //起点坐标

s=row\*n+col;

scanf("%d%d",&row,&col); //终点坐标

t=row\*n+col;

gets(str);

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

{

gets(str);

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

if(str[j]=='.')

a[i][j]=0; //0表示空格

else

a[i][j]=-2; //－2表示墙

}

}

void init()

{

head=0;

tail=1;

open[0]=s;

}

测试数据如下：

12 10 7 1 8

XXXXXXXXXXXX

X......X.XXX

X.X.XX.....X

X.X.XX.XXX.X

X.X.....X..X

X.XXXXXXXXXX

X...X.X....X

X.XXX...XXXX

X.....X....X

XXX.XXXX.X.X

XXXXXXX..XXX

XXXXXXXXXXXX

注：测试数据可在运行时粘贴上去（点击窗口最左上角按钮，在菜单中选则“编辑”/“粘贴”即可）。

想一想：此程序都存在哪些问题，如果openlen太小程序会不会出错，加入代码使程序能自动报出此类错误。

3. 跳马

给一个200×200的棋盘，问国际象棋的马从给定的起点到给定的终点最少需要几步。

Sample Input 0 0 1 1 Sample output 4

状态：马所在的行、列。

程序如下：

#include<stdio.h>

void readdata(); //读入数据

void init(); //初始化

int search(); //广度优先搜索

int emptyopen(); //判栈是否为空：空：1；非空：0。

long takeoutofopen(); //从栈中取出一个元素，并把该元素从栈中删除

int canmoveto(int,int,int\*,int\*,int); //判能否移动到该方向，并带回坐标(r,c)

int isaim(int row, int col); //判断该点是否是目标

int used(int,int); //判断该点是否已经走过

void addtoopen(int,int); //把该点加入到open表

int a[200][200],n=200; //a存放棋盘，n为迷宫边长

long open[2000],head,tail,openlen=2000; //open表1367

long s,t; //起点和终点

int search()

{

long u;

int row, col, r, c, i, num;

while(!emptyopen()) //当栈非空

{

u=takeoutofopen(); //从栈中取出一个元素，并把该元素从栈中删除

row=u/n; //计算该点所在的行

col=u%n; //计算该点所在的列

num=a[row][col]; //取得该点的步数

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

{

if(canmoveto(row,col,&r,&c,i)) //判能否移动到该方向，并带回坐标(r,c)

{

if(isaim(r,c)) //如果是目标结点

return(num+1); //返回最小步数

if(!used(r,c)) //如果(r,c)还未到达过

{

a[r][c]=num+1; //记录该点的最小步数

addtoopen(r,c); //把该点加入到open表

}

}

}

}

return -1;

}

int main() //为了让search()显示在一页内和main函数换了以下

{ //一般的算法程序main函数写在最上面读起来更方便

int number;

readdata(); //读取数据

init(); //初始化

number=search(); //广搜并返回最小步数

printf("%d",number); //打印结果

}

int emptyopen()

{

if(head==tail)

return(1);

else

return(0);

}

long takeoutofopen()

{

long u;

if(head==tail)

{

printf("errer: stack is empty");

return(-1);

}

u=open[head++];

head=head%openlen;

return(u);

}

int used(int row, int col)

{

if(a[row][col]==0)

return(0);

else

return(1);

}

void addtoopen(int row, int col)

{

int u;

if((head-tail)%openlen==1)

printf("open table overflow");

u=row;

u=u\*n+col;

open[tail++]= u;

tail=tail%openlen;

}

void readdata()

{

long row,col;

scanf("%ld%ld",&row,&col); //起点坐标

s=row\*n+col;

scanf("%ld%ld",&row,&col); //终点坐标

t=row\*n+col;

}

void init()

{

head=0;

tail=1;

open[0]=s;

}

int isaim(int row, int col)

{

if(row\*n+col==t)

return(1);

else

return(0);

}

思考：参考第4题，改为用结构体表示状态写此程序。

4. 独轮车

独轮车的轮子上有5种颜色，每走一格颜色变化一次，独轮车只能往前推，也可以在原地旋转，每走一格，需要一个单位的时间，每转90度需要一个单位的时间，转180度需要两个单位的时间。现给定一个20×20的迷宫、一个起点、一个终点和到达终点的颜色，问独轮车最少需要多少时间到达。

状态：独轮车所在的行、列、当前颜色、方向。

另外为了方便在结点中加上到达该点的最小步数。

程序如下：

#include<stdio.h>

struct colornode

{

int row; //该状态的行

int col; // 列

int color; // 颜色

int direction; // 方向

int num; // 最小步数

};

int search(); //广搜返回目标结点的最小步数

void readdata(); //读入数据

void init(); //初始化

struct colornode moveahead(struct colornode u); //返回u向前走一格得到的结点

int used(struct colornode v); //判断该结点是否是到达过的结点

void addtoopen(struct colornode v); //加入到open表

int islegal(struct colornode v); //如果该结点是合法的结点(未越界且是空格)

int isaim(struct colornode v); //判断该结点是否是目标结点

struct colornode takeoutofopen(); //从open表中取出一个结点并把该结点从open表中删除

struct colornode turntoleft(struct colornode u); //u向左转得到新结点v

struct colornode turntoright(struct colornode u); //u向左转得到新结点v

struct colornode s,t; //s:起始结点；t目标结点

struct colornode open[200]; //open表

int head,tail,openlen=200; //open表相关数据

int direct[4][2]={{0,-1},{1,0},{0,1},{-1,0}};//向左、下、右、上四个方向转时，行列的增加值

int a[20][20],n=20; //a数组表示迷宫；n为迷宫边长

int b[20][20][5][4]; //b数组表示搜索时的所有状态(0：未访问；1：已访问)

int main()

{

int number;

readdata();

init();

number=search();

printf("%d\n",number);

}

int search() //广搜返回目标结点的最小步数

{

struct colornode u,v;

while(head!=tail)

{

u=takeoutofopen();

v=moveahead(u); //u向前走一格得到新结点v

if(islegal(v)) //如果该结点是合法的结点(未越界且是空格)

{

if(isaim(v)) //判是否是目标结点

return(v.num);

if(!used(v)) //如果是未到达过的结点

addtoopen(v); //加入到open表

}

v=turntoleft(u); //u向左转得到新结点v

if(!used(v))

addtoopen(v);

v=turntoright(u); //u向右转得到新结点v

if(!used(v))

addtoopen(v);

}

}

int used(struct colornode v) //判断该结点是否是到达过的结点

{

if(b[v.row][v.col][v.color][v.direction]==0)

return(0);

else

return(1);

}

void addtoopen(struct colornode v) //加入到open表

{

open[tail++]=v;

tail=tail%openlen;

b[v.row][v.col][v.color][v.direction]=1;

}

struct colornode takeoutofopen() //从open表中取出一个结点并把该结点从open表中删除

{

struct colornode v;

v=open[head++];

head=head%openlen;

return(v);

}

void init() //初始化

{

int i,j,k,l;

for(i=0;i<n;i++) //所有状态初始化

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

for(k=0;k<5;k++)

for(l=0;l<4;l++)

b[i][j][k][l]=0;

head=0;

tail=0;

addtoopen(s); //把起始点加入到open表

}

void readdata() //读入数据

{

char str[50]; int i,j;

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

{

gets(str);

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

if(str[j]=='.') //读入数据'.'表示空格

a[i][j]=0; //存储时 0：表示空格

else

a[i][j]=1; // 1：表示墙

}

scanf("%d%d%d%d",&s.row,&s.col,&s.color,&s.direction); //读入起始结点信息

scanf("%d%d%d",&t.row,&t.col,&t.color); //读入目标结点信息

}

int isaim(struct colornode v) //判断该结点是否是目标结点

{

if(v.row==t.row&&v.col==t.col&&v.color==t.color)

return 1;

else

return 0;

}

int islegal(struct colornode v) //如果该结点是合法的结点(未越界且是空格)

{

if(v.row<0||v.row>=n||v.col<0||v.col>=n) //若越界

return 0;

if(a[v.row][v.col]==0) //0：表示空格

return 1;

return 0;

}

struct colornode moveahead(struct colornode u) //返回u向前走一格得到的结点

{

struct colornode v;

v.row=u.row+direct[u.direction][0];

v.col=u.col+direct[u.direction][1];

v.color=(u.color+1)%5;

v.direction=u.direction;

v.num=u.num+1;

return(v);

}

struct colornode turntoleft(struct colornode u) //u向左转得到新结点v

{

struct colornode v;

v=u;

v.direction=(v.direction+1)%4;

v.num=v.num+1;

return(v);

}

struct colornode turntoright(struct colornode u) //u向左转得到新结点v

{

struct colornode v;

v=u;

v.direction=(v.direction+3)%4;

v.num=v.num+1;

return(v);

}

测试数据：

XXXXXXXXXXXXXXXXXXXX

.....XXXX......X.XXX

X.........X.XX.....X

X.XXX.XX..X.XX.XXX.X

X.........X.XX.....X

X.XXX.XX..X.XX.XXX.X

.X.....XX.X.....X..X

X....X..X...X.X....X

...XXXX.X.XXX...XXXX

...........X.......X

XXXXXX....XXXXXX..XX

...........X.......X

.X.....XX.X.....X..X

XXXXXX....XXXXXXXX.X

X....X..X...X.X....X

...XXXX.X.XXX...XXXX

...........X.......X

XXX.X.XXXXX.XXXX.X.X

....X.XX....XXX.....

XXXX.....XX.........

1 1 1 1 4 8 1

5. 皇宫小偷

有一个小偷要到皇宫里偷取宝物，经过仔细的侦察，他发现皇宫实际上是一个20×20的迷宫，并且有一名卫兵巡逻，卫兵巡逻的路线是固定的，每单位时间移动一格，每4个单位时间循环一次。小偷每单位时间移动一格或在原地不动。任何时刻，小偷和卫兵在同一格，或卫兵能看到小偷，小偷将会被卫兵杀死。现在小偷已经得到了皇宫的地图，卫兵巡逻的起点，卫兵连续四次移动的方向和宝物的位置，请你设计一个程序判断小偷能否偷得宝物。

提示：参考第3题、第4题

6. 分酒问题

有一酒瓶装有8斤酒，没有量器，只有分别装5斤和3斤的空酒瓶。设计一程序将8斤酒分成两个4斤，并以最少的步骤给出答案。

7. \*找倍数

对于每个输入的数字（如：2），则要求 给出一个由1，0构成的十进制整数，且该整数为输入数字的某个倍数（如2对应的10，6的倍数100100100100100100）。

8. \*8数码难题

由左图变到右图最少需要几步，并演示移动过程。

实验四：动态规划

实验目的：理解动态规划的基本思想，理解动态规划算法的两个基本要素最优子结构性质和子问题的重叠性质。熟练掌握典型的动态规划问题。掌握动态规划思想分析问题的一般方法，对较简单的问题能正确分析，设计出动态规划算法，并能快速编程实现。

实验内容：编程实现讲过的例题：最长公共子序列问题、矩阵连乘问题、凸多边形最优三角剖分问题、电路布线问题等。本实验中的问题，设计出算法并编程实现。

习题

1． 最长公共子序列

一个给定序列的子序列是在该序列中删去若干元素后得到的序列。确切地说，若给定序列X=<x1, x2,…, xm>，则另一序列Z=<z1, z2,…, zk>是X的子序列是指存在一个严格递增的下标序列 <i1, i2,…, ik>，使得对于所有j=1,2,…,k有

解答如下：

a) 最长公共子序列的结构

若用穷举搜索法，耗时太长，算法需要指数时间。

易证最长公共子序列问题也有最优子结构性质

设序列X=<x1, x2, …, xm>和Y=<y1, y2, …, yn>的一个最长公共子序列Z=<z1, z2, …, zk>，则：

i. 若xm=yn，则zk=xm=yn且Zk-1是Xm-1和Yn-1的最长公共子序列；

ii. 若xm≠yn且zk≠xm ，则Z是Xm-1和Y的最长公共子序列；

iii. 若xm≠yn且zk≠yn ，则Z是X和Yn-1的最长公共子序列。

其中Xm-1=<x1, x2, …, xm-1>，Yn-1=<y1, y2, …, yn-1>，Zk-1=<z1, z2, …, zk-1>。

最长公共子序列问题具有最优子结构性质。

b) 子问题的递归结构

由最长公共子序列问题的最优子结构性质可知，要找出X=<x1, x2, …, xm>和Y=<y1, y2, …, yn>的最长公共子序列，可按以下方式递归地进行：当xm=yn时，找出Xm-1和Yn-1的最长公共子序列，然后在其尾部加上xm(=yn)即可得X和Y的一个最长公共子序列。当xm≠yn时，必须解两个子问题，即找出Xm-1和Y的一个最长公共子序列及X和Yn-1的一个最长公共子序列。这两个公共子序列中较长者即为X和Y的一个最长公共子序列。

由此递归结构容易看到最长公共子序列问题具有子问题重叠性质。例如，在计算X和Y的最长公共子序列时，可能要计算出X和Yn-1及Xm-1和Y的最长公共子序列。而这两个子问题都包含一个公共子问题，即计算Xm-1和Yn-1的最长公共子序列。

我们来建立子问题的最优值的递归关系。用c[i,j]记录序列Xi和Yj的最长公共子序列的长度。其中Xi=<x1, x2, …, xi>，Yj=<y1, y2, …, yj>。当i=0或j=0时，空序列是Xi和Yj的最长公共子序列，故c[i,j]=0。建立递归关系如下：

c) 计算最优值

由于在所考虑的子问题空间中，总共只有θ(m\*n)个不同的子问题，因此，用动态规划算法自底向上地计算最优值能提高算法的效率。

计算最长公共子序列长度的动态规划算法LCS\_LENGTH(X,Y)以序列X=<x1, x2, …, xm>和Y=<y1, y2, …, yn>作为输入。输出两个数组c[0..m ,0..n]和b[1..m ,1..n]。其中c[i,j]存储Xi与Yj的最长公共子序列的长度，b[i,j]记录指示c[i,j]的值是由哪一个子问题的解达到的，这在构造最长公共子序列时要用到。最后，X和Y的最长公共子序列的长度记录于c[m,n]中。

程序如下：

#include<stdio.h>

#include<string.h>

int lcs\_length(char x[], char y[]);

int main()

{

char x[100],y[100];

int len;

while(1)

{

scanf("%s%s",x,y);

if(x[0]=='0') //约定第一个字符串以‘0’开始表示结束

break;

len=lcs\_length(x,y);

printf("%d\n",len);

}

}

int lcs\_length(char x[], char y[] )

{

int m,n,i,j,l[100][100];

m=strlen(x);

n=strlen(y);

for(i=0;i<m+1;i++)

l[i][0]=0;

for(j=0;j<n+1;j++)

l[0][j]=0;

for(i=1;i<=m;i++)

for(j=1;j<=n;j++)

if(x[i-1]==y[j-1]) //i,j从1开始，但字符串是从0开始

l[i][j]=l[i-1][j-1]+1;

else if(l[i][j-1]>l[i-1][j])

l[i][j]=l[i][j-1];

else

l[i][j]=l[i-1][j];

return l[m][n];

}

由于每个数组单元的计算耗费Ο(1)时间，算法lcs\_length耗时Ο(mn)。

思考：空间能节约吗？

2． 计算矩阵连乘积

在科学计算中经常要计算矩阵的乘积。矩阵A和B可乘的条件是矩阵A的列数等于矩阵B的行数。若A是一个p×q的矩阵，B是一个q×r的矩阵，则其乘积C=AB是一个p×r的矩阵。由该公式知计算C=AB总共需要pqr次的数乘。其标准计算公式为：

现在的问题是，给定n个矩阵{A1,A2,…,An}。其中Ai与Ai+1是可乘的，i=1,2,…,n-1。要求计算出这n个矩阵的连乘积A1A2…An。

递归公式：

程序如下：

#include<stdio.h>

int main()

{

int p[101],i,j,k,r,t,n;

int m[101][101]; //为了跟讲解时保持一致数组从1开始

int s[101][101]; //记录从第i到第j个矩阵连乘的断开位置

scanf("%d",&n);

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

scanf("%d",&p[i]); //读入p[i]的值(注意：p[0]到p[n]共n+1项)

for(i=1;i<=n;i++) //初始化m[i][i]=0

m[i][i]=0;

for(r=1;r<n;r++) //r为i、j相差的值

for(i=1;i<n;i++) //i为行

{

j=i+r; //j为列

m[i][j]=m[i+1][j]+p[i-1]\*p[i]\*p[j]; //给m[i][j]赋初值

s[i][j]=i;

for(k=i+1;k<j;k++)

{

t=m[i][k]+m[k+1][j]+p[i-1]\*p[k]\*p[j];

if(t<m[i][j])

{

m[i][j]=t; //m[i][j]取最小值

s[i][j]=k;

}

}

}

printf("%d",m[1][n]);

}

3． 凸多边形的最优三角剖分

多边形是平面上一条分段线性的闭曲线。也就是说，多边形是由一系列首尾相接的直线段组成的。组成多边形的各直线段称为该多边形的边。多边形相接两条边的连接点称为多边形的顶点。若多边形的边之间除了连接顶点外没有别的公共点，则称该多边形为简单多边形。一个简单多边形将平面分为3个部分：被包围在多边形内的所有点构成了多边形的内部；多边形本身构成多边形的边界；而平面上其余的点构成了多边形的外部。当一个简单多边形及其内部构成一个闭凸集时，称该简单多边形为凸多边形。也就是说凸多边形边界上或内部的任意两点所连成的直线段上所有的点均在该凸多边形的内部或边界上。

通常，用多边形顶点的逆时针序列来表示一个凸多边形，即P=<v0 ,v1 ,… ,vn-1>表示具有n条边v0v1，v1v2，… ,vn-1vn的一个凸多边形，其中，约定v0=vn 。

若vi与vj是多边形上不相邻的两个顶点，则线段vivj称为多边形的一条弦。弦 将多边形分割成凸的两个子多边形<vi ,vi+1 ,… ,vj>和<vj ,vj+1 ,… ,vi>。多边形的三角剖分是一个将多边形分割成互不重迭的三角形的弦的集合T。如图是一个凸多边形的两个不同的三角剖分。

凸多边形最优三角剖分的问题是：给定一个凸多边形P=<v0 ,v1 ,… ,vn-1>以及定义在由多边形的边和弦组成的三角形上的权函数ω。要求确定该凸多边形的一个三角剖分，使得该三角剖分对应的权即剖分中诸三角形上的权之和为最小。

可以定义三角形上各种各样的权函数W。例如：定义ω(△vivjvk)=|vivj|+|vivk|+|vkvj|，其中，|vivj|是点vi到vj的欧氏距离。相应于此权函数的最优三角剖分即为最小弦长三角剖分。(注意：解决此问题的算法必须适用于任意的权函数)

4． 防卫导弹

一种新型的防卫导弹可截击多个攻击导弹。它可以向前飞行，也可以用很快的速度向下飞行，可以毫无损伤地截击进攻导弹，但不可以向后或向上飞行。但有一个缺点，尽管它发射时可以达到任意高度，但它只能截击比它上次截击导弹时所处高度低或者高度相同的导弹。现对这种新型防卫导弹进行测试，在每一次测试中，发射一系列的测试导弹（这些导弹发射的间隔时间固定，飞行速度相同），该防卫导弹所能获得的信息包括各进攻导弹的高度，以及它们发射次序。现要求编一程序，求在每次测试中，该防卫导弹最多能截击的进攻导弹数量，一个导弹能被截击应满足下列两个条件之一：

a)它是该次测试中第一个被防卫导弹截击的导弹；

b)它是在上一次被截击导弹的发射后发射，且高度不大于上一次被截击导弹的高度的导弹。

输入数据：第一行是一个整数n，以后的n各有一个整数表示导弹的高度。

输出数据：截击导弹的最大数目。

分析：定义l[i]为选择截击第i个导弹，从这个导弹开始最多能截击的导弹数目。

由于选择了第i枚导弹，所以下一个要截击的导弹j的高度要小于等于它的高度，所以l[i]应该等于从i＋1到n的每一个j，满足h[j]<=h[i]的j中l[j]的最大值。

程序如下：

#include<stdio.h>

int main()

{

int i,j,n,max,h[100],l[100];

scanf("%d",&n);

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

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

l[n-1]=1;

for(i=n-2;i>=0;i--)

{

max=0;

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

if(h[i]>h[j]&&max<l[j])

max=l[j];

l[i]=max+1;

}

printf("%d",l[0]);

}

5． 石子合并

在一个圆形操场的四周摆放着n堆石子(n<= 100)，现要将石子有次序地合并成一堆。规定每次只能选取相邻的两堆合并成新的一堆,并将新的一堆的石子数,记为该次合并的得分。编一程序，由文件读入堆栈数n及每堆栈的石子数(<=20)。

选择一种合并石子的方案,使得做n－1次合并,得分的总和最小；

输入数据：

第一行为石子堆数n；

第二行为每堆的石子数,每两个数之间用一个空格分隔。

输出数据：

从第一至第n行为得分最小的合并方案。第n+1行是空行.从第n+2行到第2n+1行是得分最大合并方案。每种合并方案用n行表示，其中第i行(1<=i<=n)表示第i次合并前各堆的石子数(依顺时针次序输出，哪一堆先输出均可)。要求将待合并的两堆石子数以相应的负数表示。

Sample Input

4

4 5 9 4

Sample Output

－4 5 9 －4

－8 －5 9

－13 －9

22 4 －5 －9 4

4 －14 －4

－4 －18

22

6． 最小代价子母树

设有一排数，共ｎ个，例如：22 14 7 13 26 15 11。任意2个相邻的数可以进行归并，归并的代价为该两个数的和,经过不断的归并，最后归为一堆，而全部归并代价的和称为总代价，给出一种归并算法，使总代价为最小。

输入、输出数据格式与“石子合并”相同。

Sample Input

4

12 5 16 4

Sample Output

－１２ －５ １６ ４

１７ －１６ －４

－１７ －２０

３７

7． 商店购物

某商店中每种商品都有一个价格。例如，一朵花的价格是2 ICU(ICU 是信息学竞赛的货币的单位）；一个花瓶的价格是5 ICU。为了吸引更多的顾客，商店提供了特殊优惠价。特殊优惠商品是把一种或几种商品分成一组。并降价销售。例如：3朵花的价格不是6而是5 ICU；2个花瓶加1朵花是10 ICU不是12 ICU。

编一个程序，计算某个顾客所购商品应付的费用。要充分利用优惠价以使顾客付款最小。请注意，你不能变更顾客所购商品的种类及数量，即使增加某些商品会使付款总数减小也不允许你作出任何变更。假定各种商品价格用优惠价如上所述，并且某顾客购买物品为：3朵花和2个花瓶。那么顾客应付款为14 ICU因为：

1朵花加2个花瓶优惠价：10 ICU

2朵花正常价：4 ICU

输入数据：第一个文件INPUT．TXT描述顾客所购物品（放在购物筐中）;第二个文件描述商店提供的优惠商品及价格（文件名为OFF ER．TXT）。 两个文件中都只用整数。

第一个文件INPUT．TXT的格式为:第一行是一个数字B（0≤B≤5），表示所购商品种类数。下面共B行，每行中含3个数C，K，P。 C 代表商品的编码（每种商品有一个唯一的编码），1≤C≤999。K代表该种商品购买总数，1≤K≤5。P 是该种商品的正常单价（每件商品的价格），1≤P≤999。请注意，购物筐中最多可放5\*5＝25件商品。

第二个文件OFFER．TXT的格式为:第一行是一个数字S（0≤S≤9 9），表示共有S 种优惠。下面共S行，每一行描述一种优惠商品的组合中商品的种类。下面接着是几个数字对（C，K），其中C代表商品编码，1≤C≤9 99。K代表该种商品在此组合中的数量，1≤K≤5。本行最后一个数字P（1≤ P≤9999）代表此商品组合的优惠价。当然， 优惠价要低于该组合中商品正常价之总和。

输出数据：在输出文件OUTPUT．TXT中写 一个数字（占一行），该数字表示顾客所购商品（输入文件指明所购商品）应付的最低货款。

8． 旅游预算

一个旅行社需要估算乘汽车从某城市到另一城市的最小费用，沿路有若干加油站，每个加油站收费不一定相同。旅游预算有如下规则：

若油箱的油过半，不停车加油，除非油箱中的油不可支持到下一站；每次加油时都加满；在一个加油站加油时，司机要花费2元买东西吃；司机不必为其他意外情况而准备额外的油；汽车开出时在起点加满油箱；计算精确到分（1元=100分）。编写程序估计实际行驶在某路线所需的最小费用。

输入数据：从当前目录下的文本文件“route.dat”读入数据。按以下格式输入若干旅行路线的情况：

第一行为起点到终点的距离（实数）

第二行为三个实数，后跟一个整数，每两个数据间用一个空格隔开。其中第一个数为汽车油箱的容量（升），第二个数是每升汽油行驶的公里数，第三个数是在起点加满油箱的费用，第四个数是加油站的数量。（〈=50）。接下去的每行包括两个实数，每个数据之间用一个空格分隔，其中第一个数是该加油站离起点的距离，第二个数是该加油站每升汽油的价格（元/升）。加油站按它们与起点的距离升序排列。所有的输入都有一定有解。

输出数据：答案输出到当前目录下的文本文件“route.out”中。该文件包括两行。第一行为一个实数和一个整数，实数为旅行的最小费用，以元为单位，精确到分，整数表示途中加油的站的N。第二行是N个整数，表示N个加油的站的编号，按升序排列。数据间用一个空格分隔，此外没有多余的空格。

Sample Input

516.3 38.09 1

15.7 22.1 20.87 3 2

125.4 1.259

297.9 1.129

345.2 0.999

Sample Output

38.09 1

2

9． 皇宫看守

太平王世子事件后，陆小凤成了皇上特聘的御前一品侍卫。皇宫以午门为起点，直到后宫嫔妃们的寝宫，呈一棵树的形状；某些宫殿间可以互相望见。大内保卫森严，三步一岗，五步一哨，每个宫殿都要有人全天候看守，在不同的宫殿安排看守所需的费用不同。可是陆小凤手上的经费不足，无论如何也没法在每个宫殿都安置留守侍卫。

请你编程计算帮助陆小凤布置侍卫，在看守全部宫殿的前提下，使得花费的经费最少。

输入数据：输入数据由文件名为intput.txt的文本文件提供。输入文件中数据表示一棵树，描述如下：

第1行 n，表示树中结点的数目。

第2行至第n+1行，每行描述每个宫殿结点信息，依次为：该宫殿结点标号i（0<i<=n），在该宫殿安置侍卫所需的经费k，该边的儿子数m，接下来m个数，分别是这个节点的m个儿子的标号r1，r2，...，rm。

对于一个n（0 < n <= 1500）个结点的树，结点标号在1到n之间，且标号不重复。

输出数据：输出到output.txt文件中。输出文件仅包含一个数，为所求的最少的经费。

如右图的输入数据示例：

Sample Input

6

1 30 3 2 3 4

2 16 2 5 6

3 5 0

4 4 0

5 11 0

6 5 0

Sample Output

25

10． 游戏室问题

有一个游戏室里有多个游戏室，并且这每个游戏室里还有多个游戏室，每个游戏室里面还有游戏室，依此类推。进入每个游戏室都可得到一定的快乐，每个游戏室的门票价格都大于等于0，都有一个快乐值，并且只有进入了一个游戏室，才可以进入它内部的游戏室，小明现在有n元钱，问最大能得到多少的快乐。

11． \*基因问题

已知两个基因序列如s：AGTAGT；t：ATTAG。现要你给序列中增加一些空格后，首先使得两个序列的长度相等，其次两个串对应符号匹配得到的值最大。基因只有四种分别用A、G、C、T表示，匹配中不允许两个空格相对应，任意两符号的匹配值由下表给出：

A G C T ︺

A 5 -2 -1 -2 -4

G -2 5 -4 -3 -2

C -1 -4 5 -5 -1

T -2 -3 -5 5 -2

︺ -4 -2 -1 -2

提示：定义问题l[i][j]为取第一个序列的前i项，和第二个序列的前j项，这两个序列加空格匹配的最大值。它的最优值与三个子问题有关，l[i-1][j-1]、l[i][j-1]、l[i-1][j]。

建立递归公式如下：

其中m[0][t[j]表示表中空格和t[j]匹配的对应值。

思考：本问题的初始化。

12． \*田忌赛马

田忌与齐王赛马，双方各有n匹马参赛（n<=100），每场比赛赌注为1两黄金，现已知齐王与田忌的每匹马的速度，并且齐王肯定是按马的速度从快到慢出场，现要你写一个程序帮助田忌计算他最好的结果是赢多少两黄金（输用负数表示）。

分析：先排序，齐王的马的速度放在数组a中，田忌的马的速度放在数组b中。本问题应用的算法是动态规划和贪心算法相结合解决的。从两人的最弱的马入手：

若田忌的马快，就让这两匹马比赛；

若田忌的马慢，干脆就让他对付齐王最快的马；

若两匹马的速度相等，这时有两种选择方案，或者它俩比赛，或者对付齐王最快的马。

定义子问题：l(i，j)为齐王的从第i匹马开始的j匹马与田忌的最快的j匹马比赛，田忌所获得的最大收益。

则：

程序具体实现时，为了适合c数据从0开始，稍加变动，定义子问题：l(i，j)为齐王的从第i匹马开始到第i＋j匹马共j+1匹马与田忌的最快的j+1匹马比赛，田忌所获得的最大收益。初始化时：l[i][0]表示齐王的第i匹马与田忌最快的马比赛的结果。

程序如下：

#include<stdio.h>

void readdata();

void init();

int n,a[100],b[100],l[100][100];

int main()

{

int i,j;

readdata();

init();

for(i=n-2;i>=0;i--)

for(j=1;j<n-i;j++)

if(a[i+j]<b[j])

l[i][j]=l[i][j-1]+1;

else if(a[i+j]>b[j])

l[i][j]=l[i+1][j-1]-1;

else if(l[i+1][j-1]-1>l[i][j-1])

l[i][j]=l[i+1][j-1]-1;

else

l[i][j]=l[i][j-1];

printf("%d",l[0][n-1]);

}

void readdata()

{

int i;

scanf("%d",&n);

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

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

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

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

}

void init()

{

int i;

// qsort(a,n); //略

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

{

if(a[i]<b[0])

l[i][0]=1;

else if(a[i]==b[0])

l[i][0]=0;

else

l[i][0]=-1;

}

}

实验五：贪心算法和随机算法

1. 背包问题

有一个背包，背包容量是M=150。有7个物品，物品可以分割成任意大小。

要求尽可能让装入背包中的物品总价值最大，但不能超过总容量。

物品 A B C D E F G

重量 35 30 60 50 40 10 25

价值 10 40 30 50 35 40 30

2. 照亮的山景

在一片山的上空，高度为T处有N个处于不同位置的灯泡，如图。如果山的边界上某一点于某灯i的连线不经过山的其它点，我们称灯i可以照亮该点。开尽量少的灯，使得整个山景都被照亮。山被表示成有m个转折点的折线。

提示：照亮整个山景相当于照亮每一个转折点。

3. 搬桌子问题

某教学大楼一层有n个教室，从左到右依次编号为1、2、…、n。现在要把一些课桌从某些教室搬到另外一些教室，每张桌子都是从编号较小的教室搬到编号较大的教室，每一趟，都是从左到右走，搬完一张课桌后，可以继续从当前位置或往右走搬另一张桌子。输入数据：先输入n、m，然后紧接着m行输入这m张要搬课桌的起始教室和目标教室。输出数据：最少需要跑几趟。

Sample Input

10 5

1 3

3 9

4 6

6 10

7 8

Sample Output

3

分析：贪心算法，把课桌按起点从小到大排序，每次都是搬离当前位置最近的课桌。

程序如下：

#include<stdio.h>

int main()

{

struct

{

int start;

int end;

}a[100];

int i,j;

int n,m,min,num,temp,used[100]={0};

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

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

scanf("%d%d",&a[i].start,&a[i].end);

// sort(a,n); //按start从小到大对数组a排序

min=0;

num=0;

while(num<n)

{

temp=0;

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

if(used[i]==0&&a[i].start>=temp)

{

temp=a[i].end;

used[i]=1;

num++;

}

min++;

}

printf("%d",min);

}

4. 用随即算法求解8皇后问题

5. 素数测试

附录：逻辑推理问题

对于较难的逻辑推理问题，看上去难于解决，不知道该从哪里下手时，认真的读题从最简单最特殊的情况入手

1． 月薪30K的面试题

小明和小强都是张老师的学生，张老师的生日是m月n日，2人都知道张老师的生日是下列10组中的一天，张老师把m值告诉了小明，把n值告诉了小强，张老师问他们知道他的生日是那一天吗？

3月4日 3月5日 3月8日

6月4日 6月7日

9月1日 9月5日

12月1日 12月2日 12月8日

小明说：如果我不知道的话，小强肯定也不知道

小强说：本来我也不知道，但是现在我知道了

小明说：哦，那我也知道了

提示：做西服有的需要几分钟，有的需要几百道工序。只要认真就能做好。

此题做对的人很少，不是因为题目太难，而是因为不够认真。

2． 微软面试题

一个大院子里住了50户人家，每家都养了一条狗，有一天他们接到通知说院子里有狗生病了，并要求所有主人在发现自己家狗生病的当天就要把狗枪杀掉。然而所有主人和他们的狗都不能够离开自己的房子，主人与主人之间也不能通过任何方式进行沟通，他们能做的只是通过窗户观察别人家的狗是否生病从而判断自己的狗病否。（就是说，每个主人只能看出其他 49家的狗是不是生病，单独看自己的狗是看不出来的）第一天没有枪声，第二天还是没有枪声，第三天传出一阵枪声，问有多少条狗被枪杀。

提示：上面的大字

3． 海盗分金块问题

海盗，大家听说过吧。这是一帮亡命之徒，在海上抢人钱财，夺人性 命，干的是刀头上舔血的营生。在我们的印象中，他们一般都瞎一只 眼，用条黑布或者讲究点的用个黑皮眼罩把坏眼遮上。他们还有在地 下埋宝的好习惯，而且总要画上一张藏宝图，以方便后人掘取。不过 大家是否知道，他们是世界上最民主的团体。参加海盗的都是桀骜不 驯的汉子，是不愿听人命令的，船上平时一切事都由投票解决。船长 的唯一特权，是有自己的一套餐具--可是在他不用时，其他海盗是 可以借来用的。船上的唯一惩罚，就是被丢到海里去喂鱼。

现在船上有若干个海盗，要分抢来的若干枚金币。自然，这样的问题 他们是由投票来解决的。投票的规则如下：先由最凶猛的海盗来提出 分配方案，然后大家一人一票表决，如果有50%或以上的海盗同意这个 方案，那么就以此方案分配，如果少于50%的海盗同意，那么这个提出 方案的海盗就将被丢到海里去喂鱼，然后由剩下的海盗中最凶猛的那 个海盗提出方案，依此类推。

我们先要对海盗们作一些假设。

1)每个海盗的凶猛性都不同，而且所有海盗都知道别人的凶猛性，也就是说，每个海盗都知道自己和别人在这个提出方案的序列中的位置。另外，每个海盗的数学和逻辑都很好，而且很理智。最后，海盗间私底下的交易是不存在的，因为海盗除了自己谁都不相信。

2)一枚金币是不能被分割的，不可以你半枚我半枚。

3)每个海盗当然不愿意自己被丢到海里去喂鱼，这是最重要的。

4)每个海盗当然希望自己能得到尽可能多的金币。

5)每个海盗都是现实主义者，如果在一个方案中他得到了1枚金币，而下一个方案中，他有两种可能，一种得到许多金币，一种得不到金币，他会同意目前这个方案，而不会有侥幸心理。总而言之，他们相信二 鸟在林，不如一鸟在手。

6)最后，每个海盗都很喜欢其他海盗被丢到海里去喂鱼。在不损害自己利益的前提下，他会尽可能投票让自己的同伴喂鱼。

现在，如果有10个海盗要分100枚金币，将会怎样？

提示：同上

4． 囚犯放风问题

有100个无期徒刑囚徒，被关在100个独立的小房间，互相无法通信。每天会有一个囚徒被随机地抽出来放风，随机就是说可能被抽到多次，也可能一次抽不到。

放风的地方有一盏灯，囚徒可以打开或者关上，除囚徒外，没有别人会去动这个灯。每个人除非出来放风，是看不到这个灯的。一天，全体囚徒大会，国王大赦，给大家一个机会：

如果某一天，某个囚徒能够明确表示，所有的囚徒都已经被放过风了，而且的确如此，那么所有囚徒释放；如果仍有囚徒未被放过风，那么所有的囚徒一起处死！囚徒大会后给大家20分钟时间讨论，囚徒们能找到方法么？除了那个灯以外，囚徒们不能以其他方式联系

提示：说过了

5． 天盟笔试的最后一道题目

一段文章，中有逗号若干，现求一算法，搜寻某一逗号位置。要求：

1、只搜索文章一遍

2、搜索过程中只能存储一个逗号的位置

3、对于每个逗号，被搜寻到的几率是相等的

提示：书读百遍其意自见

6． 现有100个黑球和100个白球，每次从中任意取出两个球，若颜色相同，则给这堆球中放入一个黑球，若颜色不同，则给这堆球中放入一个白球，这样当这堆球中只剩下一个球时这个球是什么颜色，请证明你的结论。

提示：多读仔细分析就能抓住关键。

7． 下面用数学归纳法证明“所有的马的颜色都相同”的证明是否正确，如不正确指出错误的原因。

（1）基础：当n=1时显然正确。

（2）归纳：假设n=k时命题成立，当n=k+1时，从中间取出一匹马，这是只有k匹马，根据假设，这k匹马的颜色是相同的，再从中取出一匹马，把刚才这匹马放回，这是又是k匹马，根据假设，这k匹马的颜色是相同的，所以这k+1匹马的颜色是相同的。

由以上两步可知，所有的马的颜色都是相同的。

提示：不需要什么，除了知识。

8． 现在小明一家过一座桥，过桥时候是黑夜，所以必须有灯。现在小明过桥要１秒，小明的弟弟要３秒，小明的爸爸要６秒，小明的妈妈要８秒，小明的爷爷要１２秒。每次此桥最多可过两人，而过桥的速度依过桥最慢者而定，而且灯在点燃后３０秒就会熄灭。问小明一家如何过桥？（再加一点，如果是5个人，6个人呢？）

9． 小明早晨6：00从山脚下开始爬山，中午12：00到达山顶；第二天早晨6：00从山顶开始下山，中午12：00到达山脚下。上山的路只有一条没有任何岔路，你能否证明：小明在同一个时间经过了同一个点。

10． 一男孩和一女孩分别在离家2千米和1千米且方向相反的两所学校上学每天同时以4千米/时和2千米/时的速度步行上学一小狗以6千米/时的速度从男孩奔向女孩又从女孩处奔向男孩，如此往返当两人到达学校时小狗在何处？

然后一步一步分析

ACM竞赛须掌握的知识

图论

拓扑排序

有向无环图与动态规划的关系

二分图匹配问题

一般图问题与二分图问题的转换思路

最大匹配

有向图的最小路径覆盖

0 / 1矩阵的最小覆盖

完备匹配

最优匹配

稳定婚姻

网络流问题

网络流模型的简单特征和与线性规划的关系

最大流最小割定理

最大流问题

有上下界的最大流问题

循环流

最小费用最大流 / 最大费用最大流

弦图的性质和判定

组合数学

解决组合数学问题时常用的思想

逼近

递推 / 动态规划

概率问题

Polya定理

计算几何 / 解析几何

计算几何的核心：叉积 / 面积

解析几何的主力：复数

基本形

点

直线，线段

多边形

凸多边形 / 凸包

凸包算法的引进，卷包裹法

Graham扫描法

水平序的引进，共线凸包的补丁

完美凸包算法

相关判定

两直线相交

两线段相交

点在任意多边形内的判定

点在凸多边形内的判定

经典问题

最小外接圆

近似O(n)的最小外接圆算法

点集直径

旋转卡壳，对踵点

多边形的三角剖分

数学 / 数论

最大公约数

Euclid算法

扩展的Euclid算法

同余方程 / 二元一次不定方程

同余方程组

线性方程组

高斯消元法

解mod 2域上的线性方程组

整系数方程组的精确解法

矩阵

行列式的计算

利用矩阵乘法快速计算递推关系

分数

分数树

连分数逼近

数论计算

求N的约数个数

求phi(N)

求约数和

快速数论变换

……

素数问题

概率判素算法

概率因子分解

数据结构

组织结构

二叉堆

左偏树

二项树

胜者树

跳跃表

样式图标

斜堆

reap

统计结构

树状数组

虚二叉树

线段树

矩形面积并

圆形面积并

关系结构

Hash表

并查集

路径压缩思想的应用

STL中的数据结构

vector

deque

set / map

动态规划 / 记忆化搜索

动态规划和记忆化搜索在思考方式上的区别

最长子序列系列问题

最长不下降子序列

最长公共子序列

一类NP问题的动态规划解法

树型动态规划

背包问题

动态规划的优化

四边形不等式

函数的凸凹性

状态设计

规划方向

线性规划

常用思想

二分

最小表示法

串

KMP

Trie结构

后缀树/后缀数组

LCA/RMQ

有限状态自动机理论

排序

选择/冒泡

快速排序

堆排序

归并排序

基数排序

拓扑排序

排序网络

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*以下为POJ推荐\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

题目均为POJ上的

http://acm.pku.edu.cn

个别题目的分类并不准确

======================================

OJ上的一些水题(可用来练手和增加自信)

(poj3299,poj2159,poj2739

,poj1083,poj2262,poj1503,poj3006,poj2255,poj3094)

初期:

一.基本算法:

(1)枚举. (poj1753,poj2965)

(2)贪心(poj1328,poj2109,poj2586)

(3)递归和分治法.

(4)递推.

(5)构造法.(poj3295)

(6)模拟法.(poj1068,poj2632,poj1573,poj2993,poj2996)

二.图算法:

(1)图的深度优先遍历和广度优先遍历.

(2)最短路径算法(dijkstra,bellman-ford,floyd,heap+dijkstra)

(poj1860,poj3259,poj1062,poj2253,poj1125,poj2240)

(3)最小生成树算法(prim,kruskal)

(poj1789,poj2485,poj1258,poj3026)

(4)拓扑排序 (poj1094)

(5)二分图的最大匹配 (匈牙利算法) (poj3041,poj3020)

(6)最大流的增广路算法(KM算法). (poj1459,poj3436)

三.数据结构.

(1)串 (poj1035,poj3080,poj1936)

(2)排序(快排、归并排(与逆序数有关)、堆排) (poj2388,poj2299)

(3)简单并查集的应用.

(4)哈希表和二分查找等高效查找法(数的Hash,串的Hash)

(poj3349,poj3274,POJ2151,poj1840,poj2002,poj2503)

(5)哈夫曼树(poj3253)

(6)堆

(7)trie树(静态建树、动态建树) (poj2513)

四.简单搜索

(1)深度优先搜索 (poj2488,poj3083,poj3009,poj1321,poj2251)

(2)广度优先搜索(poj3278,poj1426,poj3126,poj3087.poj3414)

(3)简单搜索技巧和剪枝(poj2531,poj1416,poj2676,1129)

五.动态规划

(1)背包问题. (poj1837,poj1276)

(2)型如下表的简单DP(可参考lrj的书 page149):

1.E[j]=opt{D[i]+w(i,j)} (poj3267,poj1836,poj1260,poj2533)

2.E[i,j]=opt{D[i-1,j]+xi,D[i,j-1]+yj,D[i-1][j-1]+zij} (最长公共子序列)

(poj3176,poj1080,poj1159)

3.C[i,j]=w[i,j]+opt{C[i,k-1]+C[k,j]}.(最优二分检索树问题)

六.数学

(1)组合数学:

1.加法原理和乘法原理.

2.排列组合.

3.递推关系.

(POJ3252,poj1850,poj1019,poj1942)

(2)数论.

1.素数与整除问题

2.进制位.

3.同余模运算.

(poj2635, poj3292,poj1845,poj2115)

(3)计算方法.

1.二分法求解单调函数相关知识.(poj3273,poj3258,poj1905,poj3122)

七.计算几何学.

(1)几何公式.

(2)叉积和点积的运用(如线段相交的判定,点到线段的距离等). (poj2031,poj1039)

(3)多边型的简单算法(求面积)和相关判定(点在多边型内,多边型是否相交)

(poj1408,poj1584)

(4)凸包. (poj2187,poj1113)

中级:

一.基本算法:

(1)C++的标准模版库的应用. (poj3096,poj3007)

(2)较为复杂的模拟题的训练(poj3393,poj1472,poj3371,poj1027,poj2706)

二.图算法:

(1)差分约束系统的建立和求解. (poj1201,poj2983)

(2)最小费用最大流(poj2516,poj2516,poj2195)

(3)双连通分量(poj2942)

(4)强连通分支及其缩点.(poj2186)

(5)图的割边和割点(poj3352)

(6)最小割模型、网络流规约(poj3308, )

三.数据结构 .

(1)线段树. (poj2528,poj2828,poj2777,poj2886,poj2750)

(2)静态二叉检索树. (poj2482,poj2352)

(3)树状树组(poj1195,poj3321)

(4)RMQ. (poj3264,poj3368)

(5)并查集的高级应用. (poj1703,2492)

(6)KMP算法. (poj1961,poj2406)

四.搜索

(1)最优化剪枝和可行性剪枝

(2)搜索的技巧和优化 (poj3411,poj1724)

(3)记忆化搜索(poj3373,poj1691)

五.动态规划

(1)较为复杂的动态规划(如动态规划解特别的施行商问题等)

(poj1191,poj1054,poj3280,poj2029,poj2948,poj1925,poj3034)

(2)记录状态的动态规划. (POJ3254,poj2411,poj1185)

(3)树型动态规划(poj2057,poj1947,poj2486,poj3140)

六.数学

(1)组合数学:

1.容斥原理.

2.抽屉原理.

3.置换群与Polya定理(poj1286,poj2409,poj3270,poj1026).

4.递推关系和母函数.

(2)数学.

1.高斯消元法(poj2947,poj1487, poj2065,poj1166,poj1222)

2.概率问题. (poj3071,poj3440)

3.GCD、扩展的欧几里德(中国剩余定理) (poj3101)

(3)计算方法.

1.0 /1分数规划. (poj2976)

2.三分法求解单峰(单谷)的极值.

3.矩阵法(poj3150,poj3422,poj3070)

4.迭代逼近(poj3301)

(4)随机化算法(poj3318,poj2454)

(5)杂题.

(poj1870,poj3296,poj3286,poj1095)

七.计算几何学.

(1)坐标离散化.

(2)扫描线算法(例如求矩形的面积和周长并,常和线段树或堆一起使用).

(poj1765,poj1177,poj1151,poj3277,poj2280,poj3004)

(3)多边形的内核(半平面交)(poj3130,poj3335)

(4)几何工具的综合应用.(poj1819,poj1066,poj2043,poj3227,poj2165 ,poj3429)

高级:

一.基本算法要求:

(1)代码快速写成,精简但不失风格

(poj2525,poj1684,poj1421,poj1048,poj2050,poj3306)

(2)保证正确性和高效性. poj3434

二.图算法:

(1)度限制最小生成树和第K最短路. (poj1639)

(2)最短路,最小生成树,二分图,最大流问题的相关理论(主要是模型建立和求解)

(poj3155, poj2112,poj1966,poj3281,poj1087,poj2289,poj3216,poj2446

(3)最优比率生成树. (poj2728)

(4)最小树形图(poj3164)

(5)次小生成树.

(6)无向图、有向图的最小环

三.数据结构.

(1)trie图的建立和应用. (poj2778)

(2)LCA和RMQ问题(LCA(最近公共祖先问题) 有离线算法(并查集+dfs) 和 在线算法

(RMQ+dfs)).(poj1330)

(3)双端队列和它的应用(维护一个单调的队列,常常在动态规划中起到优化状态转移的

目的). (poj2823)

(4)左偏树(可合并堆).

(5)后缀树(非常有用的数据结构,也是赛区考题的热点).

(poj3415,poj3294)

四.搜索

(1)较麻烦的搜索题目训练(poj1069,poj3322,poj1475,poj1924,poj2049,poj3426)

(2)广搜的状态优化:利用M进制数存储状态、转化为串用hash表判重、按位压缩存储状态、双向广搜、A\*算法. (poj1768,poj1184,poj1872,poj1324,poj2046,poj1482)

(3)深搜的优化:尽量用位运算、一定要加剪枝、函数参数尽可能少、层数不易过大、可以考虑双向搜索或者是轮换搜索、IDA\*算法. (poj3131,poj2870,poj2286)

五.动态规划

(1)需要用数据结构优化的动态规划.

(poj2754,poj3378,poj3017)

(2)四边形不等式理论.

(3)较难的状态DP(poj3133)

六.数学

(1)组合数学.

1.MoBius反演(poj2888,poj2154)

2.偏序关系理论.

(2)博奕论.

1.极大极小过程(poj3317,poj1085)

2.Nim问题.

七.计算几何学.

(1)半平面求交(poj3384,poj2540)

(2)可视图的建立(poj2966)

(3)点集最小圆覆盖.

(4)对踵点(poj2079)

八.综合题.

(poj3109,poj1478,poj1462,poj2729,poj2048,poj3336,poj3315,poj2148,poj1263)

=================================

下面是另一版本POJ推荐，基本都比较难，很多题目与黑书配套

=================================

推荐一些题目，希望对参与ICPC竞赛的同学有所帮助。

POJ上一些题目在

http://162.105.81.202/course/problemSolving/

可以找到解题报告。

《算法艺术与信息学竞赛》的习题提示在网上可搜到

一.动态规划

参考资料：

刘汝佳《算法艺术与信息学竞赛》

《算法导论》

推荐题目：

http://acm.pku.edu.cn/JudgeOnline/problem?id=1141

简单

http://acm.pku.edu.cn/JudgeOnline/problem?id=2288

中等，经典TSP问题

http://acm.pku.edu.cn/JudgeOnline/problem?id=2411

中等，状态压缩DP

http://acm.pku.edu.cn/JudgeOnline/problem?id=1112

中等

http://acm.pku.edu.cn/JudgeOnline/problem?id=1848

中等，树形DP。

可参考《算法艺术与信息学竞赛》动态规划一节的树状模型

http://acm.zju.edu.cn/show\_problem.php?pid=1234

中等，《算法艺术与信息学竞赛》中的习题

http://acm.pku.edu.cn/JudgeOnline/problem?id=1947

中等，《算法艺术与信息学竞赛》中的习题

http://acm.pku.edu.cn/JudgeOnline/problem?id=1946

中等，《算法艺术与信息学竞赛》中的习题

http://acm.pku.edu.cn/JudgeOnline/problem?id=1737

中等，递推

http://acm.pku.edu.cn/JudgeOnline/problem?id=1821

中等，需要减少冗余计算

http://acm.zju.edu.cn/show\_problem.php?pid=2561

中等，四边形不等式的简单应用

http://acm.pku.edu.cn/JudgeOnline/problem?id=1038

较难，状态压缩DP，《算法艺术与信息学竞赛》中有解答

http://acm.pku.edu.cn/JudgeOnline/problem?id=1390

较难，《算法艺术与信息学竞赛》中有解答

http://acm.pku.edu.cn/JudgeOnline/problem?id=3017

较难，需要配合数据结构优化（我的题目^\_^）

http://acm.pku.edu.cn/JudgeOnline/problem?id=1682

较难，写起来比较麻烦

http://acm.pku.edu.cn/JudgeOnline/problem?id=2047

较难

http://acm.pku.edu.cn/JudgeOnline/problem?id=2152

难，树形DP

http://acm.pku.edu.cn/JudgeOnline/problem?id=3028

难，状态压缩DP，题目很有意思

http://acm.pku.edu.cn/JudgeOnline/problem?id=3124

难

http://acm.pku.edu.cn/JudgeOnline/problem?id=2915

非常难

二.搜索

参考资料：

刘汝佳《算法艺术与信息学竞赛》

推荐题目：

http://acm.pku.edu.cn/JudgeOnline/problem?id=1011

简单，深搜入门题

http://acm.pku.edu.cn/JudgeOnline/problem?id=1324

中等，广搜

http://acm.pku.edu.cn/JudgeOnline/problem?id=2044

中等，广搜

http://acm.pku.edu.cn/JudgeOnline/problem?id=2286

较难，广搜

http://acm.pku.edu.cn/JudgeOnline/problem?id=1945

难，IDA\*，迭代加深搜索，需要较好的启发函数

http://acm.pku.edu.cn/JudgeOnline/problem?id=2449

难，可重复K最短路，A\*。

可参考解题报告:

http://acm.pku.edu.cn/JudgeOnline/showcontest?contest\_id=1144

http://acm.pku.edu.cn/JudgeOnline/problem?id=1190

难，深搜剪枝，《算法艺术与信息学竞赛》中有解答

http://acm.pku.edu.cn/JudgeOnline/problem?id=1084

难，《算法艺术与信息学竞赛》习题

http://acm.pku.edu.cn/JudgeOnline/problem?id=2989

难，深搜

http://acm.pku.edu.cn/JudgeOnline/problem?id=1167

较难，《算法艺术与信息学竞赛》中有解答

http://acm.pku.edu.cn/JudgeOnline/problem?id=1069

很难

三. 常用数据结构

参考资料：

刘汝佳《算法艺术与信息学竞赛》

《算法导论》

线段树资料：

http://home.ustc.edu.cn/~zhuhcheng/ACM/segment\_tree.pdf

树状数组资料

http://home.ustc.edu.cn/~zhuhcheng/ACM/tree.ppt

关于线段树和树状数组更多相关内容可在网上搜到

后缀数组资料

http://home.ustc.edu.cn/~zhuhcheng/ACM/suffix\_array.pdf

http://home.ustc.edu.cn/~zhuhcheng/ACM/linear\_suffix.pdf

推荐题目

http://acm.pku.edu.cn/JudgeOnline/problem?id=2482

较难，线段树应用，《算法艺术与信息学竞赛》中有解答

http://acm.pku.edu.cn/JudgeOnline/problem?id=1151

简单，线段树应用矩形面积并，《算法艺术与信息学竞赛》中有解答

http://acm.pku.edu.cn/JudgeOnline/problem?id=3225

较难，线段树应用，可参考解题报告

http://acm.pku.edu.cn/JudgeOnline/showcontest?contest\_id=1233

http://acm.pku.edu.cn/JudgeOnline/problem?id=2155

难，二维树状数组。

http://acm.pku.edu.cn/JudgeOnline/problem?id=2777

中等，线段树应用。

http://acm.pku.edu.cn/JudgeOnline/problem?id=2274

难，堆的应用，《算法艺术与信息学竞赛》中有解答

http://acm.zju.edu.cn/show\_problem.php?pid=2334

中等，左偏树，二项式堆或其他可合并堆的应用。

左偏树参考http://www.nist.gov/dads/HTML/leftisttree.html

二项式堆参见《算法导论》相关章节

http://acm.pku.edu.cn/JudgeOnline/problem?id=1182

中等，并查集

http://acm.pku.edu.cn/JudgeOnline/problem?id=1816

中等，字典树

http://acm.pku.edu.cn/JudgeOnline/problem?id=2778

较难，多串匹配树

参考：http://home.ustc.edu.cn/~zhuhcheng/ACM/zzy2004.pdf

http://acm.pku.edu.cn/JudgeOnline/problem?id=1743

难，后缀数组

http://acm.pku.edu.cn/JudgeOnline/problem?id=2774

较难，最长公共子串，经典问题，后缀数组

http://acm.pku.edu.cn/JudgeOnline/problem?id=2758

很难，后缀数组

可参考解题报告

http://acm.pku.edu.cn/JudgeOnline/showcontest?contest\_id=1178

http://acm.pku.edu.cn/JudgeOnline/problem?id=2448

很难，数据结构综合运用

四.图论基础

参考资料：

刘汝佳《算法艺术与信息学竞赛》

《算法导论》

《网络算法与复杂性理论》谢政

推荐题目:

http://acm.pku.edu.cn/JudgeOnline/problem?id=2337

简单，欧拉路

http://acm.pku.edu.cn/JudgeOnline/problem?id=3177

中等，无向图割边

http://acm.pku.edu.cn/JudgeOnline/problem?id=2942

较难，无向图双连通分支

http://acm.pku.edu.cn/JudgeOnline/problem?id=1639

中等，最小度限制生成树，《算法艺术与信息学竞赛》中有解答

http://acm.pku.edu.cn/JudgeOnline/problem?id=2728

中等，最小比率生成树，《算法艺术与信息学竞赛》中有解答

http://acm.pku.edu.cn/JudgeOnline/problem?id=3013

简单，最短路问题

http://acm.pku.edu.cn/JudgeOnline/problem?id=1275

中等，差分约束系统，Bellman-Ford求解，《算法艺术与信息学竞赛》中有解答

http://acm.pku.edu.cn/JudgeOnline/problem?id=1252

简单，Bellman-Ford

http://acm.pku.edu.cn/JudgeOnline/problem?id=1459

中等，网络流

http://acm.pku.edu.cn/JudgeOnline/problem?id=2391

较难，网络流

http://acm.pku.edu.cn/JudgeOnline/problem?id=1325

中等，二部图最大匹配

http://acm.pku.edu.cn/JudgeOnline/problem?id=2226

较难，二部图最大匹配

http://acm.pku.edu.cn/JudgeOnline/problem?id=2195

中等，二部图最大权匹配

KM算法参考《网络算法与复杂性理论》

http://acm.pku.edu.cn/JudgeOnline/problem?id=2516

较难，二部图最大权匹配

http://acm.pku.edu.cn/JudgeOnline/problem?id=1986

中等，LCA（最近公共祖先）问题

参考Tarjan's LCA algorithm 《算法导论》第21章习题

http://acm.pku.edu.cn/JudgeOnline/problem?id=2723

较难，2-SAT问题

参考：http://home.ustc.edu.cn/~zhuhcheng/ACM/2-SAT.PPT

http://acm.pku.edu.cn/JudgeOnline/problem?id=2749

较难，2-SAT问题

http://acm.pku.edu.cn/JudgeOnline/problem?id=3164

较难，最小树形图

参考《网络算法与复杂性理论》中朱-刘算法

五.数论及组合计数基础

http://acm.pku.edu.cn/JudgeOnline/problem?id=1811

简单，素数判定，大数分解

参考算法导论相关章节

http://acm.pku.edu.cn/JudgeOnline/problem?id=2888

较难，Burnside引理

http://acm.pku.edu.cn/JudgeOnline/problem?id=2891

中等，解模方程组

http://acm.pku.edu.cn/JudgeOnline/problem?id=2154

中等，经典问题，波利亚定理

http://cs.scu.edu.cn/soj/problem.action?id=2703

难，极好的题目，Burnside引理+模线性方程组

http://acm.pku.edu.cn/JudgeOnline/problem?id=2764

较难，需要数学方法，该方法在《具体数学》第七章有讲

http://acm.pku.edu.cn/JudgeOnline/problem?id=1977

简单，矩阵快速乘法

/\*