ACM大量习题题库

ACM大量习题题库

现在网上有许多题库，大多是可以在线评测，所以叫做Online Judge。除了USACO是为IOI准备外，其余几乎全部是大学的ACM竞赛题库。

USACO

http://ace.delos.com/usacogate

美国著名在线题库，专门为信息学竞赛选手准备

TJU

http://acm.tongji.edu.cn/

同济大学在线题库，唯一的中文题库，适合NOIP选手

ZJU

http://acm.zju.edu.cn/

浙江大学在线题库

JLU

http://acm.jlu.edu.cn/

吉林大学在线题库（一直上不去）

PKU

http://acm.pku.edu.cn

北京大学在线题库

URAL

http://acm.timus.ru

俄罗斯乌拉尔大学在线题库

SGU

http://acm.sgu.ru/

俄罗斯圣萨拉托夫州大学在线题库

ELJ

http://acm.mipt.ru/judge/bin/problems.pl?lang=en

俄罗斯莫斯科物理技术学院

SPOJ

https://spoj.sphere.pl/

波兰格但斯克理工大学

UVA

http://acm.uva.es/

西班牙的Universidad de Valladolid在线题

ACM ICPC学习计划

大牛给的计划——

一般要做到50行以内的程序不用调试、100行以内的二分钟内调试成功.acm主要是考算法的,主要时间是花在思考算法上，不是花在写程序与debug上。

下面给个计划你练练：

第一阶段：练经典常用算法，下面的每个算法给我打上十到二十遍，同时自己精简代码，

因为太常用，所以要练到写时不用想，10-15分钟内打完，甚至关掉显示器都可以把程序打

出来.

1.最短路(Floyd、Dijstra,BellmanFord)

2.最小生成树(先写个prim,kruscal要用并查集，不好写)

3.大数（高精度）加减乘除

4.二分查找. (代码可在五行以内)

5.叉乘、判线段相交、然后写个凸包.

6.BFS、DFS,同时熟练hash表(要熟，要灵活,代码要简)

7.数学上的有：辗转相除（两行内），线段交点、多角形面积公式.

8. 调用系统的qsort, 技巧很多，慢慢掌握.

9. 任意进制间的转换

第二阶段：练习复杂一点，但也较常用的算法。

如：

1. 二分图匹配（匈牙利），最小路径覆盖

2. 网络流，最小费用流。

3. 线段树.

4. 并查集。

5. 熟悉动态规划的各个典型：LCS、最长递增子串、三角剖分、记忆化dp

6.博弈类算法。博弈树，二进制法等。

7.最大团，最大独立集。

8.判断点在多边形内。

9. 差分约束系统.

10. 双向广度搜索、A\*算法，最小耗散优先.

第三阶段：

前两个阶段是打基础，第三阶段是锻炼在比赛中可以快速建立模型、想新算法

。这就要平时多做做综合的题型了。

1. 把oibh上的论文看看（大概几百篇的，我只看了一点点，呵呵）。

2. 平时扫扫zoj上的难题啦，别老做那些不用想的题.(中大acm的版主经常说我挑简单的来

做:-P )

3. 多参加网上的比赛，感受一下比赛的气氛，评估自己的实力.

4. 一道题不要过了就算，问一下人，有更好的算法也打一下。

5. 做过的题要记好 :-)

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

ACMer必备知识（这么多呀，慢慢学了……

图论

路径问题

0/1边权最短路径

BFS

非负边权最短路径（Dijkstra）

可以用Dijkstra解决问题的特征

负边权最短路径

Bellman-Ford

Bellman-Ford的Yen-氏优化

差分约束系统

Floyd

广义路径问题

传递闭包

极小极大距离 / 极大极小距离

Euler Path / Tour

圈套圈算法

混合图的 Euler Path / Tour

Hamilton Path / Tour

特殊图的Hamilton Path / Tour 构造

生成树问题

最小生成树

第k小生成树

最优比率生成树

0/1分数规划

度限制生成树

连通性问题

强大的DFS算法

无向图连通性

割点

割边

二连通分支

有向图连通性

强连通分支

2-SAT

最小点基

有向无环图

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

最长公共不下降子序列

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

熟练掌握数据结构、常用算法汇聚

（一）

不可能都完全记住那么多的算法.

常用算法,拿过来就可以写出来

不常用的,拿起书来,看10分钟,就能理解算法(因为以前记过).

对以前没有记过的算法,就不好说了,难的可能要研究好几天.

这样就可以了.

应该熟练掌握的常用的算法应该有:

各种排序算法（插入排序、冒泡排序、选择排序，快速排序，堆排序，归并排序）

线性表(一般的线性表,栈,队列)的插入和删除

二叉树的遍历（前序，中序，后序）

图的遍历（深度优先，广度优先）

二分法查找，排序二叉树，Hash查找（处理冲突的方法）。

（二）

分析一个东西,你可以用不同的眼光去看待,有很多时候,就跟自己生活一样,觉得小时候看待问题很幼稚,现在看问题全面了,而且方式不一样了,为什么,就是成长吧,就跟这个一样的,你对算法,比如写一个程序,可能直接写很简单,可是可以有一些有趣的方式,比如通过什么样来表达,怎么样更高效..等等吧

（三）

于大学里把基本的专业课学扎实就ok，如：数据结构，离散，操作系统等。碰到一些基本的数据结构和算法，如查找排序要根据原理马上能写出相应的代码就行了，我个人是这样理解的，对于更深层次的东西，也是建立在自己熟练的基础之上的吧

（四）

算法与数据结构考验试题精析》第2版 机械工业出版社

如果你想练习的话，这里有N多的题可以来练习，但实际中能用到的比较少，除非搞一些高端的玩意，不过平时也可以在自己的项目中结合使用

（五）

数据结构在平时可能用不上，但数据结构可以培养你程序时如果注意效率的意识，一个学过数据结构的人和一个没有学过数结构的人写出来的程序可能在效率上有差别。

（六）

搞ACM需要的掌握的算法.

要注意,ACM的竞赛性强,因此自己应该和自己的实际应用联系起来.

适合自己的才是好的,有的人不适合搞算法,喜欢系统架构,因此不要看到别人什么就眼红,

发挥自己的长处,这才是重要的.

（八）

搞实际项目的话基本用不着多少。lss说的那点都已经多了。当然，偶个人觉得，判断一个问题是否NPC/NPH还是比较有用的，判是以后就不会把自己的经历浪费在寻找多项式算法上了。这点acm要用，实际项目偶觉得也有用。

acm的话上面贴的那一长串还不够用。所谓不够用，第一，指这些就算都会都不会写错，不会建立dp模型不会建立图论模型的话一样能挂得很惨，这种活的东西不是死做题就能会的。第二，这表还不全。既然图可以扯到最优比率生成树，那博弈的话至少也得扯SG定理，串的话至少也得扯AC自动机（吐槽：不是自动AC机），

（九）补充中。。。。

浙江大学 http://acm.zju.edu.cn 北京大学 http://acm.pku.edu.cn/JudgeOnline

天津大学 http://acm.tju.edu.cn 厦门大学 http://acm.xmu.edu.cn/JudgeOnline

福州大学 http://acm.fzu.edu.cn 华中科技 http://acm.hust.edu.cn/JudgeOnline

宁波理工 http://acm.nit.net.cn 合肥工大 http://acm.tdzl.net:83/JudgeOnline

汕头大学 http://acm.stu.edu.cn 北大内部 http://ai.pku.cn/JudgeOnline

中国科大 http://acm.ustc.edu.cn 暨南大学 http://202.116.24.78/JudgeOnline

浙江工业 http://acm.zjut.edu.cn 中山大学 http://202.116.77.69/sicily

福建师范 http://acm.fjnu.edu.cn 哈工业大 http://acm.hit.edu.cn/ojs/ojs.php

杭电科大 http://acm.hziee.edu.cn 四川大学 http://acm.scu.edu.cn/soj

哈工程大 http://acm.hrbeu.edu.cn 武汉大学 http://acm.whu.edu.cn/noah

同济大学 http://acm.tongji.edu.cn 湖南大学 http://acm.hnu.cn:8080/online/?

上海大学 http://pc.shu.edu.cn/openjudge/problemlist.php

兰州大学 http://acm.sundayclub.cn/JudgeOnline/problemlist

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)

同时由于个人练习的时候可能有些偏向性,可能上面的总结不是很全,还请大家提出和指正,而且由于ACM的题目中专门针对某个算法的题目可能比较少出现,所以上面的分类中的题有可能有多种解法或者是一些算法的综合,这都不会影响大家做题,希望练习的同学能够认真,扎实地训练,做到真正的理解算法,掌握算法.同时在论坛上还有许多前辈的分类,总结,大家也可以按自己的情况采用.注意FTP上有很多的资料,希望大家好好地利用.

如果同学能在明年暑假前能掌握上面大部分算法,那你也基本上达到了训练的目的,到暑假的时候你就可以选择自己比较喜欢的方面进行加深和强化,而且同学们不要觉得看算法的证明是很麻烦的事,这可以加强你的思维能力,这在ACM中也很重要.同时也希望老队员能帮助我整理习题和题目分类.同时ACM的题目是没有范围的,只能在平时中多积累多练习,多比别人多努力一点,你就会比别人多一线希望.

我补充些动态规划、搜索方面的资料吧。

Dp状态设计与方程总结

1.不完全状态记录

<1>青蛙过河问题

<2>利用区间dp

2.背包类问题

<1> 0-1背包，经典问题

<2>无限背包，经典问题

<3>判定性背包问题

<4>带附属关系的背包问题

<5> + -1背包问题

<6>双背包求最优值

<7>构造三角形问题

<8>带上下界限制的背包问题(012背包)

3.线性的动态规划问题

<1>积木游戏问题

<2>决斗（判定性问题）

<3>圆的最大多边形问题

<4>统计单词个数问题

<5>棋盘分割

<6>日程安排问题

<7>最小逼近问题(求出两数之比最接近某数/两数之和等于某数等等)

<8>方块消除游戏(某区间可以连续消去求最大效益)

<9>资源分配问题

<10>数字三角形问题

<11>漂亮的打印

<12>邮局问题与构造答案

<13>最高积木问题

<14>两段连续和最大

<15>2次幂和问题

<16>N个数的最大M段子段和

<17>交叉最大数问题

4.判定性问题的dp(如判定整除、判定可达性等)

<1>模K问题的dp

<2>特殊的模K问题，求最大(最小)模K的数

<3>变换数问题

5.单调性优化的动态规划

<1>1-SUM问题

<2>2-SUM问题

<3>序列划分问题(单调队列优化)

6.剖分问题(多边形剖分/石子合并/圆的剖分/乘积最大)

<1>凸多边形的三角剖分问题

<2>乘积最大问题

<3>多边形游戏(多边形边上是操作符,顶点有权值)

<4>石子合并(N^3/N^2/NLogN各种优化)

7.贪心的动态规划

<1>最优装载问题

<2>部分背包问题

<3>乘船问题

<4>贪心策略

<5>双机调度问题Johnson算法

8.状态dp

<1>牛仔射击问题(博弈类)

<2>哈密顿路径的状态dp

<3>两支点天平平衡问题

<4>一个有向图的最接近二部图

9.树型dp

<1>完美服务器问题(每个节点有3种状态)

<2>小胖守皇宫问题

<3>网络收费问题

<4>树中漫游问题

<5>树上的博弈

<6>树的最大独立集问题

<7>树的最大平衡值问题

<8>构造树的最小环

先掌握搜索，动态规划，贪心这些思想方法

然后学习各种技巧

ACM基本算法分类

ACM基本算法分类、推荐学习资料和配套pku习题一.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

第01篇 ACM/ICPC竞赛之基础篇

一、ACM/ICPC竞赛的特点

ACM/ICPC（国际大学生程序设计竞赛）是以算法设计为主的程序设计竞赛，并不涉及具体的应用技术。

ACM/ICPC竞赛以组队形式参赛，每个参赛队由三名队员组成，共同使用一台计算机解题。通常每场比赛的试题为6至10题，根据各队的完成题数和罚时进行排名。题目提交通过称为完成，从比赛开始到提交成功所用的时间为题目的基础罚时，另外，一道题目每提交失败一次，将增加20分钟罚时。也就是说，参赛队要尽可能用最快的速度、最少的失败次数，解决最多的题目。

二、输入和输出处理

试题一般采用标准输入和输出方式读取输入和产生输出，在题目中会详细描述输入和输出的格式和值域范围，所写的程序一定要严格遵守题目指定的输入输出格式。

在比赛试题的输入和输出处理上，针对一些常见的情形，有一些常用的方法。

1、多测试用例的输入和输出

有些试题在一次输入中只包含一个测试用例，也就是说，程序每运行一次，只算一道题。也有些试题在一次输入中包含多个测试用例，也就是说，程序每运行一次，要计算多道题。

对多用例输入，通常会先输入要计算的用例的个数，然后依次输入每个测试用例的输入数据，但程序并不需要等到所有的测试用例都计算完后再输出所有测试用例的计算结果，而是可以读入一个测试用例，输出一个结果，再读入一个测试用例，再输出一个结果。因此对多用例输入的试题，可以用这样的输入模式：

以C++为例：

int n;

cin >> n;

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

{

读入测试用例数据

计算

输出计算结果

}

2、单测试用例输入的结束判断

对单用例输入，最主要的问题是如何知道输入什么时候结束。

有些试题会指定某种特殊的输入值作为输入的结束标志，这种情况比较容易处理，只须在读入后，判断一下读入的内容是否为约定结束值即可。

有些试题并不指定特殊的输入值，而是以EOF（文件结束标志）作为结束标志。如果从文件流读入，当读到文件尾时，输入返回EOF。如果从键盘读入时，在Windows的终端中，是以Ctrl+Z表示EOF。对于这种情况，可以用这样的输入模式：

以C++为例：

int m, n; // 假设要连续输入一组整数对

while (cin>>m>>n)

{

处理整数对(m, n)

}

以C语言为例：

int m, n;

while (scanf("%d%d", &m, &n)==2)

{

处理整数对(m, n)

}

三、数据结构的设计

很多试题中已经给出了数据量的上限，因此可以很方便地以数组的方式定义数据结构。但也要注意到有些题目中没有明确指出数据上限时，切不可盲目定义数组大小。

例如在题1070（多项式求和）中，并未说明输入多项式的项数，对这种情况就不宜用数组方式来表示多项式了——除非你的运气足够好，所开辟的数组大小能够经受所有的测试用例的考验。

除了使用一般的数组或链表结构外，对使用C++的选手来说，STL也是一大利器，充分运用可以有效提高编程的效率和正确性。

四、测试用例的考虑

在试题中通常会给出测试用例的样例，这通常会被我们用来测试自己的程序，而且很多选手往往在正确计算出测试用例样例时，会认为自己的程序是正确的。

其实测试用例的样例只是测试用例的个例，实际用于测试的测试用例往往会涵盖各种极限情况和边界情况，而且有时测试用例的数量还会比较大，甚至会重复测试同一个测试用例。因此我们的程序能够通过样例测试，未必能够通过所有的测试用例的测试，一方面要全面考虑所有可能的极限情况和边界情况，一方面程序要有足够的效率。

2008-7-17 01:29 回复

狂晕的迷战士

14位粉丝

2楼

第03篇 ACM/ICPC竞赛之STL--pair

STL的<utility>头文件中描述了一个看上去非常简单的模板类pair，用来表示一个二元组或元素对，并提供了按照字典序对元素对进行大小比较的比较运算符模板函数。

例如，想要定义一个对象表示一个平面坐标点，则可以：

pair<double, double> p1;

cin >> p1.first >> p1.second;

pair模板类需要两个参数：首元素的数据类型和尾元素的数据类型。pair模板类对象有两个成员：first和second，分别表示首元素和尾元素。

在<utility>中已经定义了pair上的六个比较运算符：<、>、<=、>=、==、!=，其规则是先比较first，first相等时再比较second，这符合大多数应用的逻辑。

当然，也可以通过重载这几个运算符来重新指定自己的比较逻辑。

除了直接定义一个pair对象外，如果需要即时生成一个pair对象，也可以调用在<utility>中定义的一个模板函数：make\_pair。make\_pair需要两个参数，

分别为元素对的首元素和尾元素。

在题1067--Ugly Numbers中，就可以用pair来表示推演树上的结点，用first表示结点的值，用second表示结点是由父结点乘以哪一个因子得到的。

#include <iostream>

#include <queue>

using namespace std;

typedef pair<unsigned long, int> node\_type;

main()

{ unsigned long result[1500];

priority\_queue< node\_type, vector<node\_type>, greater<node\_type> > Q;

Q.push( make\_pair(1, 2) );

for (int i=0; i<1500; i++)

{

node\_type node = Q.top();

Q.pop();

switch(node.second)

{ case 2: Q.push( make\_pair(node.first\*2, 2) );

case 3: Q.push( make\_pair(node.first\*3, 3) );

case 5: Q.push( make\_pair(node.first\*5, 5) );

}

result[i] = node.first;

}

int n; cin >> n;

while (n>0)

{

cout << result[n-1] << endl;

cin >> n;

}

return 1;

}

<utility>看上去是很简单的一个头文件，但是<utility>的设计中却浓缩反映了STL设计的基本思想。有意深入了解和研究STL的同学，仔细阅读和体会这个简单的头文件，

不失为一种入门的途径。

2008-7-17 01:31 回复

狂晕的迷战士

14位粉丝

3楼

第04篇 ACM/ICPC竞赛之STL--vector

在STL的<vector>头文件中定义了vector（向量容器模板类），vector容器以连续数组的方式存储元素序列，可以将vector看作是以顺序结构实现的线性表。

当我们在程序中需要使用动态数组时，vector将会是理想的选择，vector可以在使用过程中动态地增长存储空间。

vector模板类需要两个模板参数，第一个参数是存储元素的数据类型，第二个参数是存储分配器的类型，其中第二个参数是可选的，如果不给出第二个参数，

将使用默认的分配器。

下面给出几个常用的定义vector向量对象的方法示例：

vector<int> s;

定义一个空的vector对象，存储的是int类型的元素。

vector<int> s(n);

定义一个含有n个int元素的vector对象。

vector<int> s(first, last);

定义一个vector对象，并从由迭代器first和last定义的序列[first, last)中复制初值。

vector的基本操作有：

s[i]

直接以下标方式访问容器中的元素。

s.front()

返回首元素。

s.back()

返回尾元素。

s.push\_back(x)

向表尾插入元素x。

s.size()

返回表长。

s.empty()

当表空时，返回真，否则返回假。

s.pop\_back()

删除表尾元素。

s.begin()

返回指向首元素的随机存取迭代器。

s.end()

返回指向尾元素的下一个位置的随机存取迭代器。

s.insert(it, x)

向迭代器it指向的元素前插入新元素val。

s.insert(it, n, x)

向迭代器it指向的元素前插入n个x。

s.insert(it, first, last)

将由迭代器first和last所指定的序列[first, last)插入到迭代器it指向的元素前面。

s.erase(it)

删除由迭代器it所指向的元素。

s.erase(first, last)

删除由迭代器first和last所指定的序列[first, last)。

s.reserve(n)

预分配缓冲空间，使存储空间至少可容纳n个元素。

s.resize(n)

改变序列的长度，超出的元素将会被删除，如果序列需要扩展（原空间小于n），元素默认值将填满扩展出的空间。

s.resize(n, val)

改变序列的长度，超出的元素将会被删除，如果序列需要扩展（原空间小于n），将用val填满扩展出的空间。

s.clear()

删除容器中的所有的元素。

s.swap(v)

将s与另一个vector对象v进行交换。

s.assign(first, last)

将序列替换成由迭代器first和last所指定的序列[first, last)。[first, last)不能是原序列中的一部分。

要注意的是，resize操作和clear操作都是对表的有效元素进行的操作，但并不一定会改变缓冲空间的大小。

另外，vector还有其他一些操作如反转、取反等，不再一下列举。

vector上还定义了序列之间的比较操作运算符(>, <, >=, <=, ==, !=)，可以按照字典序比较两个序列。

还是来看一些示例代码。输入个数不定的一组整数，再将这组整数按倒序输出，如下所示：

#include <iostream>

#include <vector>

using namespace std;

main()

{

vector<int> L;

int x;

while (cin>>x) L.push\_back(x);

for (int i=L.size()-1; i>=0; i--) cout << L[i] << " ";

cout << endl;

return 1;

}

2008-7-17 01:31 回复

狂晕的迷战士

14位粉丝

4楼

第04篇 ACM/ICPC竞赛之STL--vector

在STL的<vector>头文件中定义了vector（向量容器模板类），vector容器以连续数组的方式存储元素序列，可以将vector看作是以顺序结构实现的线性表。

当我们在程序中需要使用动态数组时，vector将会是理想的选择，vector可以在使用过程中动态地增长存储空间。

vector模板类需要两个模板参数，第一个参数是存储元素的数据类型，第二个参数是存储分配器的类型，其中第二个参数是可选的，如果不给出第二个参数，

将使用默认的分配器。

下面给出几个常用的定义vector向量对象的方法示例：

vector<int> s;

定义一个空的vector对象，存储的是int类型的元素。

vector<int> s(n);

定义一个含有n个int元素的vector对象。

vector<int> s(first, last);

定义一个vector对象，并从由迭代器first和last定义的序列[first, last)中复制初值。

vector的基本操作有：

s[i]

直接以下标方式访问容器中的元素。

s.front()

返回首元素。

s.back()

返回尾元素。

s.push\_back(x)

向表尾插入元素x。

s.size()

返回表长。

s.empty()

当表空时，返回真，否则返回假。

s.pop\_back()

删除表尾元素。

s.begin()

返回指向首元素的随机存取迭代器。

s.end()

返回指向尾元素的下一个位置的随机存取迭代器。

s.insert(it, x)

向迭代器it指向的元素前插入新元素val。

s.insert(it, n, x)

向迭代器it指向的元素前插入n个x。

s.insert(it, first, last)

将由迭代器first和last所指定的序列[first, last)插入到迭代器it指向的元素前面。

s.erase(it)

删除由迭代器it所指向的元素。

s.erase(first, last)

删除由迭代器first和last所指定的序列[first, last)。

s.reserve(n)

预分配缓冲空间，使存储空间至少可容纳n个元素。

s.resize(n)

改变序列的长度，超出的元素将会被删除，如果序列需要扩展（原空间小于n），元素默认值将填满扩展出的空间。

s.resize(n, val)

改变序列的长度，超出的元素将会被删除，如果序列需要扩展（原空间小于n），将用val填满扩展出的空间。

s.clear()

删除容器中的所有的元素。

s.swap(v)

将s与另一个vector对象v进行交换。

s.assign(first, last)

将序列替换成由迭代器first和last所指定的序列[first, last)。[first, last)不能是原序列中的一部分。

要注意的是，resize操作和clear操作都是对表的有效元素进行的操作，但并不一定会改变缓冲空间的大小。

另外，vector还有其他一些操作如反转、取反等，不再一下列举。

vector上还定义了序列之间的比较操作运算符(>, <, >=, <=, ==, !=)，可以按照字典序比较两个序列。

还是来看一些示例代码。输入个数不定的一组整数，再将这组整数按倒序输出，如下所示：

#include <iostream>

#include <vector>

using namespace std;

main()

{

vector<int> L;

int x;

while (cin>>x) L.push\_back(x);

for (int i=L.size()-1; i>=0; i--) cout << L[i] << " ";

cout << endl;

return 1;

}

2008-7-17 01:32 回复

狂晕的迷战士

14位粉丝

5楼

第05篇 ACM/ICPC竞赛之STL--iterator简介

iterator(迭代器)是用于访问容器中元素的指示器，从这个意义上说，iterator(迭代器)相当于数据结构中所说的“遍历指针”，也可以把iterator(迭代器)看作是一种泛化的指针。

STL中关于iterator(迭代器)的实现是相当复杂的，这里我们暂时不去详细讨论关于iterator(迭代器)的实现和使用，而只对iterator(迭代器)做一点简单的介绍。

简单地说，STL中有以下几类iterator(迭代器)：

输入iterator(迭代器)，在容器的连续区间内向前移动，可以读取容器内任意值；

输出iterator(迭代器)，把值写进它所指向的容器中；

前向iterator(迭代器)，读取队列中的值，并可以向前移动到下一位置(++p,p++)；

双向iterator(迭代器)，读取队列中的值，并可以向前向后遍历容器；

随机访问iterator(迭代器), 可以直接以下标方式对容器进行访问，vector的iterator(迭代器)就是这种iterator(迭代器)；

流iterator(迭代器)，可以直接输出、输入流中的值；

每种STL容器都有自己的iterator(迭代器)子类，下面先来看一段简单的示例代码：

#include <iostream>

#include <vector>

using namespace std;

main()

{

vector<int> s;

for (int i=0; i<10; i++) s.push\_back(i);

for (vector<int>::iterator it=s.begin(); it!=s.end(); it++)

cout << \*it << " ";

cout << endl;

return 1;

}

vector的begin()和end()方法都会返回一个vector::iterator对象，分别指向vector的首元素位置和尾元素的下一个位置（我们可以称之为结束标志位置）。

对一个iterator(迭代器)对象的使用与一个指针变量的使用极为相似，或者可以这样说，指针就是一个非常标准的iterator(迭代器)。

再来看一段稍微特别一点的代码：

#include <iostream>

#include <vector>

using namespace std;

main()

{

vector<int> s;

s.push\_back(1);

s.push\_back(2);

s.push\_back(3);

copy(s.begin(), s.end(), ostream\_iterator<int>(cout, " "));

cout <<endl;

return 1;

}

这段代码中的copy就是STL中定义的一个模板函数，copy(s.begin(), s.end(), ostream\_iterator<int>(cout, " "));的意思是将由s.begin()至s.end()(不含s.end())所指定的序列复制到标准输出流cout中，用" "作为每个元素的间隔。也就是说，这句话的作用其实就是将表中的所有内容依次输出。

iterator(迭代器)是STL容器和算法之间的“胶合剂”，几乎所有的STL算法都是通过容器的iterator(迭代器)来访问容器内容的。只有通过有效地运用iterator(迭代器)，才能够有效地运用STL强大的算法功能。

2008-7-17 01:32 回复

狂晕的迷战士

14位粉丝

6楼

第05篇 ACM/ICPC竞赛之STL--iterator简介

iterator(迭代器)是用于访问容器中元素的指示器，从这个意义上说，iterator(迭代器)相当于数据结构中所说的“遍历指针”，也可以把iterator(迭代器)看作是一种泛化的指针。

STL中关于iterator(迭代器)的实现是相当复杂的，这里我们暂时不去详细讨论关于iterator(迭代器)的实现和使用，而只对iterator(迭代器)做一点简单的介绍。

简单地说，STL中有以下几类iterator(迭代器)：

输入iterator(迭代器)，在容器的连续区间内向前移动，可以读取容器内任意值；

输出iterator(迭代器)，把值写进它所指向的容器中；

前向iterator(迭代器)，读取队列中的值，并可以向前移动到下一位置(++p,p++)；

双向iterator(迭代器)，读取队列中的值，并可以向前向后遍历容器；

随机访问iterator(迭代器), 可以直接以下标方式对容器进行访问，vector的iterator(迭代器)就是这种iterator(迭代器)；

流iterator(迭代器)，可以直接输出、输入流中的值；

每种STL容器都有自己的iterator(迭代器)子类，下面先来看一段简单的示例代码：

#include <iostream>

#include <vector>

using namespace std;

main()

{

vector<int> s;

for (int i=0; i<10; i++) s.push\_back(i);

for (vector<int>::iterator it=s.begin(); it!=s.end(); it++)

cout << \*it << " ";

cout << endl;

return 1;

}

vector的begin()和end()方法都会返回一个vector::iterator对象，分别指向vector的首元素位置和尾元素的下一个位置（我们可以称之为结束标志位置）。

对一个iterator(迭代器)对象的使用与一个指针变量的使用极为相似，或者可以这样说，指针就是一个非常标准的iterator(迭代器)。

再来看一段稍微特别一点的代码：

#include <iostream>

#include <vector>

using namespace std;

main()

{

vector<int> s;

s.push\_back(1);

s.push\_back(2);

s.push\_back(3);

copy(s.begin(), s.end(), ostream\_iterator<int>(cout, " "));

cout <<endl;

return 1;

}

这段代码中的copy就是STL中定义的一个模板函数，copy(s.begin(), s.end(), ostream\_iterator<int>(cout, " "));的意思是将由s.begin()至s.end()(不含s.end())所指定的序列复制到标准输出流cout中，用" "作为每个元素的间隔。也就是说，这句话的作用其实就是将表中的所有内容依次输出。

iterator(迭代器)是STL容器和算法之间的“胶合剂”，几乎所有的STL算法都是通过容器的iterator(迭代器)来访问容器内容的。只有通过有效地运用iterator(迭代器)，才能够有效地运用STL强大的算法功能。

2008-7-17 01:33 回复

狂晕的迷战士

14位粉丝

7楼

第06篇 ACM/ICPC竞赛之STL--string

字符串是程序中经常要表达和处理的数据，我们通常是采用字符数组或字符指针来表示字符串。STL为我们提供了另一种使用起来更为便捷的字符串的表达方式：string。string类的定义在头文件<string>中。

string类其实可以看作是一个字符的vector，vector上的各种操作都可以适用于string，另外，string类对象还支持字符串的拼合、转换等操作。

下面先来看一个简单的例子：

#include <iostream>

#include <string>

using namespace std;

main()

{

string s = "Hello! ", name;

cin >> name;

s += name;

s += '!';

cout << s << endl;

return 1;

}

再以题1064--Parencoding为例，看一段用string作为容器，实现由P代码还原括号字符串的示例代码片段：

int m;

cin >> m; // P编码的长度

string str; // 用来存放还原出来的括号字符串

int leftpa = 0; // 记录已出现的左括号的总数

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

{

int p;

cin >> p;

for (int k=0; k<p-leftpa; k++) str += '(';

str += ')';

leftpa = p;

}

2008-7-17 01:33 回复

狂晕的迷战士

14位粉丝

9楼

看下面这个简单的示例：

#include <iostream>

#include <queue>

using namespace std;

class T

{

public:

int x, y, z;

T(int a, int b, int c):x(a), y(b), z?

{

}

};

bool operator < (const T &t1, const T &t2)

{

return t1.z < t2.z; // 按照z的顺序来决定t1和t2的顺序

}

main()

{

priority\_queue<T> q;

q.push(T(4,4,3));

q.push(T(2,2,5));

q.push(T(1,5,4));

q.push(T(3,3,6));

while (!q.empty())

{

T t = q.top(); q.pop();

cout << t.x << " " << t.y << " " << t.z << endl;

}

return 1;

}

输出结果为(注意是按照z的顺序从大到小出队的)：

3 3 6

2 2 5

1 5 4

4 4 3

再看一个按照z的顺序从小到大出队的例子：

#include <iostream>

#include <queue>

using namespace std;

class T

{

public:

int x, y, z;

T(int a, int b, int c):x(a), y(b), z?

{

}

};

bool operator > (const T &t1, const T &t2)

{

return t1.z > t2.z;

}

main()

{

priority\_queue<T, vector<T>, greater<T> > q;

q.push(T(4,4,3));

q.push(T(2,2,5));

q.push(T(1,5,4));

q.push(T(3,3,6));

while (!q.empty())

{

T t = q.top(); q.pop();

cout << t.x << " " << t.y << " " << t.z << endl;

}

return 1;

}

输出结果为：

4 4 3

1 5 4

2 2 5

3 3 6

如果我们把第一个例子中的比较运算符重载为：

bool operator < (const T &t1, const T &t2)

{

return t1.z > t2.z; // 按照z的顺序来决定t1和t2的顺序

}

则第一个例子的程序会得到和第二个例子的程序相同的输出结果。

再回顾一下用优先队列实现的题1067--Ugly Numbers的代码：

#include <iostream>

#include <queue>

using namespace std;

typedef pair<unsigned long int, int> node\_type;

main( int argc, char \*argv[] )

{

unsigned long int result[1500];

priority\_queue< node\_type, vector<node\_type>, greater<node\_type> > Q;

Q.push( make\_pair(1, 3) );

for (int i=0; i<1500; i++)

{

node\_type node = Q.top();

Q.pop();

switch(node.second)

{

case 3: Q.push( make\_pair(node.first\*2, 3) );

case 2: Q.push( make\_pair(node.first\*3, 2) );

case 1: Q.push( make\_pair(node.first\*5, 1) );

}

result[i] = node.first;

}

int n;

cin >> n;

while (n>0)

{

cout << result[n-1] << endl;

cin >> n;

}

return 1;

}

2008-7-17 01:34 回复

狂晕的迷战士

14位粉丝

10楼

第09篇 ACM/ICPC竞赛之STL--algorithm

<algorithm>无疑是STL中最大的一个头文件，它是由一大堆模板函数组成的。

下面列举出<algorithm>中的模板函数：

adjacent\_find / binary\_search / copy / copy\_backward / count / count\_if / equal / equal\_range / fill / fill\_n / find / find\_end / find\_first\_of / find\_if / for\_each / generate / generate\_n / includes / inplace\_merge / iter\_swap / lexicographical\_compare / lower\_bound / make\_heap / max / max\_element / merge / min / min\_element / mismatch / next\_permutation / nth\_element / partial\_sort / partial\_sort\_copy / partition / pop\_heap / prev\_permutation / push\_heap / random\_shuffle / remove / remove\_copy / remove\_copy\_if / remove\_if / replace / replace\_copy / replace\_copy\_if / replace\_if / reverse / reverse\_copy / rotate / rotate\_copy / search / search\_n / set\_difference / set\_intersection / set\_symmetric\_difference / set\_union / sort / sort\_heap / stable\_partition / stable\_sort / swap / swap\_ranges / transform / unique / unique\_copy / upper\_bound

如果详细叙述每一个模板函数的使用，足够写一本书的了。还是来看几个简单的示例程序吧。

示例程序之一，for\_each遍历容器：

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int Visit(int v) // 遍历算子函数

{

cout << v << " ";

return 1;

}

class MultInt // 定义遍历算子类

{

private:

int factor;

public:

MultInt(int f) : factor(f)

{

}

void operator()(int &elem) const

{

elem \*= factor;

}

};

main()

{

vector<int> L;

for (int i=0; i<10; i++) L.push\_back(i);

for\_each(L.begin(), L.end(), Visit);

cout << endl;

for\_each(L.begin(), L.end(), MultInt(2));

for\_each(L.begin(), L.end(), Visit);

cout << endl;

return 1;

}

程序的输出结果为：

0 1 2 3 4 5 6 7 8 9

0 2 4 6 8 10 12 14 16 18

示例程序之二，min\_element/max\_element，找出容器中的最小/最大值：

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

main()

{

vector<int> L;

for (int i=0; i<10; i++) L.push\_back(i);

vector<int>::iterator min\_it = min\_element(L.begin(), L.end());

vector<int>::iterator max\_it = max\_element(L.begin(), L.end());

cout << "Min is " << \*min\_it << endl;

cout << "Max is " << \*max\_it << endl;

return 1;

}

程序的输出结果为：

Min is 0

Max is 9

示例程序之三，sort对容器进行排序：

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

void Print(vector<int> &L)

{

for (vector<int>::iterator it=L.begin(); it!=L.end(); it++)

cout << \*it << " ";

cout << endl;

}

main()

{

vector<int> L;

for (int i=0; i<5; i++) L.push\_back(i);

for (int i=9; i>=5; i--) L.push\_back(i);

Print(L);

sort(L.begin(), L.end());

Print(L);

sort(L.begin(), L.end(), greater<int>()); // 按降序排序

Print(L);

return 1;

}

程序的输出结果为：

0 1 2 3 4 9 8 7 6 5

0 1 2 3 4 5 6 7 8 9

9 8 7 6 5 4 3 2 1 0

示例程序之四，copy在容器间复制元素：

#include <vector>

#include <algorithm>

#include <iostream>

using namespace std;

main( )

{

// 先初始化两个向量v1和v2

vector <int> v1, v2;

for (int i=0; i<=5; i++) v1.push\_back(10\*i);

for (int i=0; i<=10; i++) v2.push\_back(3\*i);

cout << "v1 = ( " ;

for (vector <int>::iterator it=v1.begin(); it!=v1.end(); it++)

cout << \*it << " ";

cout << ")" << endl;

cout << "v2 = ( " ;

for (vector <int>::iterator it=v2.begin(); it!=v2.end(); it++)

cout << \*it << " ";

cout << ")" << endl;

// 将v1的前三个元素复制到v2的中间

copy(v1.begin(), v1.begin()+3, v2.begin()+4);

cout << "v2 with v1 insert = ( " ;

for (vector <int>::iterator it=v2.begin(); it!=v2.end(); it++)

cout << \*it << " ";

cout << ")" << endl;

// 在v2内部进行复制，注意参数2表示结束位置，结束位置不参与复制

copy(v2.begin()+4, v2.begin()+7, v2.begin()+2);

cout << "v2 with shifted insert = ( " ;

for (vector <int>::iterator it=v2.begin(); it!=v2.end(); it++)

cout << \*it << " ";

cout << ")" << endl;

return 1;

}

程序的输出结果为：

v1 = ( 0 10 20 30 40 50 )

v2 = ( 0 3 6 9 12 15 18 21 24 27 30 )

v2 with v1 insert = ( 0 3 6 9 0 10 20 21 24 27 30 )

v2 with shifted insert = ( 0 3 0 10 20 10 20 21 24 27 30 )

2008-7-17 01:37 回复

狂晕的迷战士

14位粉丝

11楼

第10篇 ACM/ICPC竞赛之算法策略

ACM/ICPC竞赛其实就是算法设计和编码的竞赛，熟悉各种常用算法和算法设计策略并能灵活运用是非常必要的。

这里对几种在竞赛中经常用到的算法设计策略做一简单的介绍。

1、穷举法

穷举法是最基本的算法设计策略，其思想是列举出问题所有的可能解，逐一进行判别，找出满足条件的解。

穷举法的运用关键在于解决两个问题：

如何列举所有的可能解；

如何判别可能解是否满足条件；

在运用穷举法时，容易出现的问题是可能解过多，导致算法效率很低，这就需要对列举可能解的方法进行优化。

以题1041--纯素数问题为例，从1000到9999都可以看作是可能解，可以通过对所有这些可能解逐一进行判别，找出其中的纯素数，但只要稍作分析，就会发现其实可以大幅度地降低可能解的范围。根据题意易知，个位只可能是3、5、7，再根据题意可知，可以在3、5、7的基础上，先找出所有的二位纯素数，再在二位纯素数基础上找出三位纯素数，最后在三位纯素数的基础上找出所有的四位纯素数。

2、分治法

分治法也是应用非常广泛的一种算法设计策略，其思想是将问题分解为若干子问题，从而可以递归地求解各子问题，再综合出问题的解。

分治法的运用关键在于解决三个问题：

确定分治规则，即如何分解问题。

确定终结条件，即问题分解到什么状态时可以直接求解。

确定归纳方法，即如何由子问题的解得到原问题的解。这一步并不总是需要的，因为对某些问题来说，并不需要对子问题的解进行复杂的归纳。

我们熟知的如汉诺塔问题、折半查找算法、快速排序算法等都是分治法运用的典型案例。

以题1045--Square Coins为例，先对题意进行分析，可设一个函数f(m, n)等于用面值不超过n2的货币构成总值为m的方案数，则容易推导出：

f(m, n) = f(m-0\*n\*n, n-1)+f(m-1\*n\*n, n-1)+f(m-2\*n\*n, n-1)+...+f(m-k\*n\*n, n-1)

这里的k是币值为n2的货币最多可以用多少枚，即k=m/(n\*n)。

也很容易分析出，f(m, 1) = f(1, n) = 1

对于这样的题目，一旦分析出了递推公式，程序就非常好写了。所以在动手开始写程序之前，分析工作做得越彻底，逻辑描述越准确、简洁，写起程序来就会越容易。

3、动态规划法

动态规划法多用来计算最优问题，动态规划法与分治法的基本思想是一致的，但处理的手法不同。动态规划法在运用时，要先对问题的分治规律进行分析，找出终结子问题，以及子问题向父问题归纳的规则，而算法则直接从终结子问题开始求解，逐层向上归纳，直到归纳出原问题的解。

动态规划法多用于在分治过程中，子问题可能重复出现的情况，在这种情况下，如果按照常规的分治法，自上向下分治求解，则重复出现的子问题就会被重复地求解，从而增大了冗余计算量，降低了求解效率。而采用动态规划法，自底向上求解，每个子问题只计算一次，就可以避免这种重复的求解了。

动态规划法还有另外一种实现形式，即备忘录法。备忘录的基本思想是设立一个称为备忘录的容器，记录已经求得解的子问题及其解。仍然采用与分治法相同的自上向下分治求解的策略，只是对每一个分解出的子问题，先在备忘录中查找该子问题，如果备忘录中已经存在该子问题，则不须再求解，可以从备忘录中直接得到解，否则，对子问题递归求解，且每求得一个子问题的解，都将子问题及解存入备忘录中。

例如，在题1045--Square Coins中，可以采用分治法求解，也可以采用动态规划法求解，即从f(m, 1)和f(1, n)出发，逐层向上计算，直到求得f(m, n)。

在竞赛中，动态规划和备忘录的思想还可以有另一种用法。有些题目中的可能问题数是有限的，而在一次运行中可能需要计算多个测试用例，可以采用备忘录的方法，预先将所有的问题的解记录下来，然后输入一个测试用例，就查备忘录，直接找到答案输出。这在各问题之间存在父子关系的情况下，会更有效。例如，在题1045--Square Coins中，题目中已经指出了最大的目标币值不超过300，也就是说问题数只有300个，而且各问题的计算中存在重叠的子问题，可以采用动态规划法，将所有问题的解先全部计算出来，再依次输入测试用例数据，并直接输出答案。

4、回溯法

回溯法是基于问题状态树搜索的求解法，其可适用范围很广。从某种角度上说，可以把回溯法看作是优化了的穷举法。回溯法的基本思想是逐步构造问题的可能解，一边构造，一边用约束条件进行判别，一旦发现已经不可能构造出满足条件的解了，则退回上一步构造过程，重新进行构造。这个退回的过程，就称之为“回溯”。

回溯法在运用时，要解决的关键问题在于：

如何描述局部解。

如何扩展局部解和回溯局部解。

如何判别局部解。

回溯法的经典案例也很多，例如全排列问题、N后问题等。

5、贪心法

贪心法也是求解最优问题的常用算法策略，利用贪心法策略所设计的算法，通常效率较高，算法简单。贪心法的基本思想是对问题做出目前看来最好的选择，即贪心选择，并使问题转化为规模更小的子问题。如此迭代，直到子问题可以直接求解。

基于贪心法的经典算法例如：哈夫曼算法、最小生成树算法、最短路径算法等。

但是，贪心法的运用是有条件的，必须能够证明贪心选择能够导出最优解，且转化出的子问题与原问题是同性质的问题，才能使用贪心法求解。

一个比较经典的贪心法求解的问题就是找硬币问题：有1、2、5、10、20、50、100七种面值的硬币，要支付指定的金额，问怎么支付所用的硬币个数最少。这是一个非常日常化的问题，凭直觉我们会想到，尽可能先用大面值的硬币，这就是“贪心选择”，而在这个问题上，这个贪心选择也是正确的。

6、限界剪枝法

限界剪枝法是求解较复杂最优问题的一种算法策略，与回溯法类似的是，限界剪枝法也是在问题状态空间树上进行搜索，但回溯法是搜索一般解，而限界剪枝法则是搜索最优解。限界剪枝法的基本思想是通过找出权值函数的上下界函数，以下界函数来指导搜索的方向，以上界函数来帮助剪除一些不可能含有最优解的分枝。

关于算法和算法策略的讨论是一个非常庞大的话题，几乎每个问题点都能扩展出一大堆可讨论的内容和案例。我实在不知道该怎样用简短的几篇文字就能够把这个话题说透，这里只能蜻蜓点水地对竞赛中经常用到的几种策略做一极为简略的介绍。

也许我们可以在以后的文章中，针对具体的题目进行算法和策略的分析，效果可能会更好。

2008-7-17 01:37 回复

狂晕的迷战士

14位粉丝

12楼

第11篇 ACM/ICPC竞赛之调试

在写程序时，调试程序也是一个重要的环节。怎样才能够更有效地调试程序，发现并修正错误呢？

1、调试中的输入输出

为了调试程序，我们可能需要反复执行程序，也就需要反复输入相同或不相同的测试数据。如果每次调试运行时都是以手工的方式输入测试数据，相信很多人都会觉得不胜其烦。其实我们可以用一些辅助的手段来简化这个过程。

方法一：使用剪贴板

可以将输入数据预先写好（用记事本、开发环境的编辑器或随便什么能够录入的东西），再将输入数据复制到剪贴板上（也就是说我们通常所说的复制操作）。在调试运行时，就可以直接将输入数据粘贴上去，不需要手工输入，这对于反复调试同一组测试数据尤其方便。

方法二：使用重定向

使用剪贴板对于多组测试数据或者比较长的测试数据就会显得不那么好用了。而使用输入输出的重定向则会更方便。

输入输出重定向是在终端窗口下的一种命令行功能，在命令行上可以用“<”表示输入重定向，在“<”后跟随输入文件名，则程序将从指定的输入文件中获取输入数据，而不再从键盘读入数据。也可以用“>”表示输出重定向。在“>”后跟输出文件名，则程序产生的标准输出将写入指定的输出文件中，而不是显示在屏幕上。

我们可以预先将输入数据存到文本文件中（如果有多组测试数据，可以存成多个文件），用重定向指定准备使用的输入数据。

例如，程序名为myprog，输入数据已经存到文件test.txt中，则在命令行下可以这样执行：

C:>myprog < test.txt

则程序会直接从test.txt中读取输入。如果想把输出结果也存到文件中（这在输出结果比较多的时候尤其有用，因为直接输出到屏幕上可能会来不及看到输出，或看不全所有的输出），例如，可以这样执行：

C:>myprog > test.out

这样我们就可以在执行后，用一个文本编辑器打开输出文件，慢慢阅读和分析输出结果。

如果把输入和输出的重定向结合起来，也可以这样执行：

C:>myprog < test.txt > test.out

2、输出调试信息

在调试时，很多同学往往首先想到的是使用开发环境所提供的调试功能：设置断点、单步执行、查看和修改变量，甚至改变程序的流程。不可否认，使用开发环境所提供的调试功能的确很方便，但当你过分依赖于这些集成工具时，你可能忽略了很多更有效的手段：仔细地分析、充分的信息。

当我们发现程序没有按照自己预期得那样工作时，不要急于跟踪甚至修改程序，而是应该首先仔细对程序的逻辑、语句、表达式进行检查和分析，尽可能使程序在表达上更简洁、更干净。如果实在难以发现问题所在，也不必急于借助于集成工具去跟踪程序的运行。早期的程序员在调试程序时经常会在程序中加入输出调试信息的语句或过程，用以观察程序的运行过程，分析程序的运行逻辑，这种调试手段即使在今天也仍然是非常有效的。

输出的调试信息要尽量容易阅读，格式清楚，在必要的时候，可以借助工具程序或自己编写的程序对输出信息进行处理，以帮助分析问题。

3、发现线索

调试的目的就是要分析错误发生的原因，寻找线索。盲目的调试只会浪费时间。

调试中的技巧很多，这里提出几条基本原则：

首先是要使错误可重现，要设法保证能够使错误按照自己的意愿重复出现。对于不知道什么时候会冒出来的错误，分析起来会困难得多！

缩小导致错误的输入，设法构造出最小的又能保证错误出现的输入，这样可以减少变化的可能性，使分析范围更集中。经常可以采用二分选择的方法来选择输入，就是舍掉一半输入，看看错误是否会出现，如果不出现，则选择另一半输入，如此反复，并不断缩小导致错误的输入。

4、构造测试数据和测试程序

在题目中所给出的测试样例只是一小组测试数据，这虽然通常是我们用来测试程序的第一组数据，但却是远远不够的。我们应该根据题意自行构造更多的测试数据，尤其是一些边界状态的测试数据（数据极大、数据极小、数据量极多、数据量极少、预期出现极端结果等情况）。

边界测试数据可以用于检查程序中是否存在边界错误，设计有缺陷的程序，在处理边界测试数据时往往容易暴露出错误。但如果没有发生明显的运行错误，就需要对结果的正确性进行验证。

有些测试数据可以通过手工计算求出结果，再与程序的计算结果相对比，而也有些问题，可以通过构造测试程序来进行验证。

测试程序通常是用确定可靠的算法编写的解题程序，但不须考虑时间和空间的消耗，用测试程序对测试数据进行求解，用计算结果与待测试程序的计算结果进行对比。

以题1041--纯素数问题为例，我们可以用最简单的穷举法进行求解，也许这样的解法是不被接受的，因为效率太低，但这个解法却可以用作我们的测试程序，甚至——有同学索性在本地先用这个程序把结果算出来，再写一个程序直接输出结果——居然也被接受了！

写得有点疲了，脑子也有点麻木了~~~~先这样吧，等缓一缓再说吧。

2008-7-17 01:40 回复

狂晕的迷战士

14位粉丝

13楼

第02篇 ACM/ICPC竞赛之STL简介

一、关于STL

STL(Standard Template Library，标准模板库）是C++语言标准中的重要组成部分。STL以模板类和模板函数的形式为程序员提供了各种数据结构和算法的精巧实现，程序员如果能够充分地利用STL，可以在代码空间、执行时间和编码效率上获得极大的好处。

STL大致可以分为三大类：算法(algorithm)、容器(container)、迭代器(iterator)。

STL容器是一些模板类，提供了多种组织数据的常用方法，例如vector(向量，类似于数组)、list(列表，类似于链表)、deque(双向队列)、set(集合)、map(映象)、stack(栈)、queue(队列)、priority\_queue(优先队列)等，通过模板的参数我们可以指定容器中的元素类型。

STL算法是一些模板函数，提供了相当多的有用算法和操作，从简单如for\_each（遍历）到复杂如stable\_sort（稳定排序）。

STL迭代器是对C中的指针的一般化，用来将算法和容器联系起来。几乎所有的STL算法都是通过迭代器来存取元素序列进行工作的，而STL中的每一个容器也都定义了其本身所专有的迭代器，用以存取容器中的元素。有趣的是，普通的指针也可以像迭代器一样工作。

熟悉了STL后，你会发现，很多功能只需要用短短的几行就可以实现了。通过STL，我们可以构造出优雅而且高效的代码，甚至比你自己手工实现的代码效果还要好。

STL的另外一个特点是，它是以源码方式免费提供的，程序员不仅可以自由地使用这些代码，也可以学习其源码，甚至按照自己的需要去修改它。

下面是用STL写的题1067--Ugly Numbers的代码：

#include <iostream>#include <queue>using namespace std;typedef pair<unsigned long, int> node\_type;main(){ unsigned long result[1500]; priority\_queue< node\_type, vector<node\_type>, greater<node\_type> > Q; Q.push( make\_pair(1, 2) ); for (int i=0; i<1500; i++) { node\_type node = Q.top(); Q.pop(); switch(node.second) { case 2: Q.push( make\_pair(node.first\*2, 2) ); case 3: Q.push( make\_pair(node.first\*3, 3) ); case 5: Q.push( make\_pair(node.first\*5, 5) ); } result[i] = node.first; } int n; cin >> n; while (n>0) { cout << result[n-1] << endl; cin >> n; } return 1;} 在ACM竞赛中，熟练掌握和运用STL对快速编写实现代码会有极大的帮助。

二、使用STL

在C++标准中，STL被组织为以下的一组头文件（注意，是没有.h后缀的！）：

algorithm / deque / functional / iterator / list / map

memory / numeric / queue / set / stack / utility / vector

当我们需要使用STL的某个功能时，需要嵌入相应的头文件。但要注意的是，在C++标准中，STL是被定义在std命名空间中的。如下例所示：

#include <stack>

main()

{

std::stack<int> s;

s.push(0);

...

return 1;

}

如果希望在程序中直接引用STL，也可以在嵌入头文件后，用using namespace语句将std命名空间导入。如下例所示：

#include <stack>

using namespace std;

main()

{

stack<int> s;

s.push(0);

...

return 1;

}

STL是C++语言机制运用的一个典范，通过学习STL可以更深刻地理解C++语言的思想和方法。在本系列的文章中不打算对STL做深入的剖析，而只是想介绍一些STL的基本应用。

有兴趣的同学，建议可以在有了一些STL的使用经验后，认真阅读一下《C++ STL》这本书（电力出版社有该书的中文版）。

2008-7-17 01:54 回复

狂晕的迷战士

14位粉丝

14楼

第08篇 ACM/ICPC竞赛之STL--map

在STL的头文件<map>中定义了模板类map和multimap，用有序二叉树来存贮类型为pair<const Key, T>的元素对序列。序列中的元素以const Key部分作为标识，map中所有元素的Key值都必须是唯一的，multimap则允许有重复的Key值。

可以将map看作是由Key标识元素的元素集合，这类容器也被称为“关联容器”，可以通过一个Key值来快速确定一个元素，因此非常适合于需要按照Key值查找元素的容器。

map模板类需要四个模板参数，第一个是键值类型，第二个是元素类型，第三个是比较算子，第四个是分配器类型。其中键值类型和元素类型是必要的。

map的基本操作有：

1、定义map对象，例如：

map<string, int> m;

2、向map中插入元素对，有多种方法，例如：

m[key] = value;

[key]操作是map很有特色的操作，如果在map中存在键值为key的元素对，则返回该元素对的值域部分，否则将会创建一个键值为key的元素对，值域为默认值。所以可以用该操作向map中插入元素对或修改已经存在的元素对的值域部分。

m.insert( make\_pair(key, value) );

也可以直接调用insert方法插入元素对，insert操作会返回一个pair，当map中没有与key相匹配的键值时，其first是指向插入元素对的迭代器，其second为true；若map中已经存在与key相等的键值时，其first是指向该元素对的迭代器，second为false。

3、查找元素对，例如：

int i = m[key];

要注意的是，当与该键值相匹配的元素对不存在时，会创建键值为key的元素对。

map<string, int>::iterator it = m.find(key);

如果map中存在与key相匹配的键值时，find操作将返回指向该元素对的迭代器，否则，返回的迭代器等于map的end()（参见vector中提到的begin和end操作）。

4、删除元素对，例如：

m.erase(key);

删除与指定key键值相匹配的元素对，并返回被删除的元素的个数。

m.erase(it);

删除由迭代器it所指定的元素对，并返回指向下一个元素对的迭代器。

看一段简单的示例代码：

#include<map>#include<iostream> using namespace std; typedef map<int, string, less<int> > M\_TYPE;typedef M\_TYPE::iterator M\_IT;typedef M\_TYPE::const\_iterator M\_CIT; int main(){ M\_TYPE MyTestMap; MyTestMap[3] = "No.3"; MyTestMap[5] = "No.5"; MyTestMap[1] = "No.1"; MyTestMap[2] = "No.2"; MyTestMap[4] = "No.4"; M\_IT it\_stop = MyTestMap.find(2); cout << "MyTestMap[2] = " << it\_stop->second << endl; it\_stop->second = "No.2 After modification"; cout << "MyTestMap[2] = " << it\_stop->second << endl; cout << "Map contents : " << endl; for(M\_CIT it = MyTestMap.begin(); it != MyTestMap.end(); it++) { cout << it->second << endl; } return 0;}

程序执行的输出结果为：

MyTestMap[2] = No.2

MyTestMap[2] = No.2 After modification

Map contents :

No.1

No.2 After modification

No.3

No.4

No.5

再看一段简单的示例代码：

#include <iostream>

#include <map>

using namespace std;

main()

{

map<string, int> m;

m["one"] = 1;

m["two"] = 2;

// 几种不同的insert调用方法

m.insert(make\_pair("three", 3));

m.insert(map<string, int>::value\_type("four", 4));

m.insert(pair<string, int>("five", 5));

string key;

while (cin>>key)

{

map<string, int>::iterator it = m.find(key);

if (it==m.end())

{

cout << "No such key!" << endl;

}

else

{

cout << key << " is " << it->second << endl;

cout << "Erased " << m.erase(key) << endl;

}

}

return 1;

}

主流算法：

1.搜索　//回溯

2.DP（动态规划）

3.贪心

4.图论　//Dijkstra、最小生成树、网络流

5.数论　//解模线性方程

6.计算几何　//凸壳、同等安置矩形的并的面积与周长

7.组合数学　//Polya定理

8.模拟

9.数据结构　//并查集、堆

10.博弈论

1、 排序

1423, 1694, 1723, 1727, 1763, 1788, 1828, 1838, 1840, 2201, 2376, 2377, 2380, 1318, 1877,

1928, 1971, 1974, 1990, 2001, 2002, 2092, 2379,

1002（需要字符处理，排序用快排即可） 1007（稳定的排序） 2159（题意较难懂） 2231 2371（简单排

序） 2388（顺序统计算法） 2418（二叉排序树）

2、 搜索、回溯、遍历

1022 1111 1118 1129 1190 1562 1564 1573 1655 2184 2225 2243 2312 2362 2378 2386

1010,1011,1018,1020,1054,1062,1256,1321,1363,1501，

1650,1659,1664,1753,2078,2083,2303,2310,2329

简单：1128, 1166, 1176, 1231, 1256, 1270, 1321, 1543, 1606, 1664, 1731, 1742, 1745, 1847,

1915, 1950, 2038, 2157, 2182, 2183, 2381, 2386, 2426,

不易：1024, 1054, 1117, 1167, 1708, 1746, 1775, 1878, 1903, 1966, 2046, 2197, 2349,

推荐：1011, 1190, 1191, 1416, 1579, 1632, 1639, 1659, 1680, 1683, 1691, 1709, 1714, 1753,

1771, 1826, 1855, 1856, 1890, 1924, 1935, 1948, 1979, 1980, 2170, 2288, 2331, 2339,

2340,1979（和迷宫类似） 1980（对剪枝要求较高）

3、 历法

1008 2080 （这种题要小心）

4、 枚举

1012，1046， 1387， 1411， 2245， 2326， 2363， 2381，1054（剪枝要求较高），1650 （小数的精

度问题）

5、 数据结构的典型算法

容易：1182, 1656, 2021, 2023, 2051, 2153, 2227, 2236, 2247, 2352, 2395,

不易：1145, 1177, 1195, 1227, 1661, 1834,

推荐：1330, 1338, 1451, 1470, 1634, 1689, 1693, 1703, 1724, 1988, 2004, 2010, 2119, 2274,

1125(弗洛伊德算法) ，2421（图的最小生成树）

6、 动态规划

1037 A decorative fence、

1050 To the Max、

1088 滑雪、

1125 Stockbroker Grapevine、

1141 Brackets Sequence、

1159 Palindrome、

1160 Post Office、

1163 The Triangle、

1458 Common Subsequence、

1579 Function Run Fun、

1887 Testing the CATCHER、

1953 World Cup Noise、

2386 Lake Counting

7、 贪心

1042, 1065, 1230, 1323, 1477, 1716, 1784,1328 1755（或用单纯形方法），2054，1017， 1328，

1862， 1922 ，2054， 2209， 2313， 2325， 2370。

8、 模拟

容易：1006, 1008, 1013, 1016, 1017, 1169, 1298, 1326, 1350, 1363, 1676, 1786, 1791, 1835,

1970, 2317, 2325, 2390,

不易：1012, 1082, 1099, 1114, 1642, 1677, 1684, 1886,1281 1928 2083 2141 2015

9、 递归

1664

10、字符串处理

1488, 1598, 1686, 1706, 1747, 1748, 1750, 1760, 1782, 1790, 1866, 1888, 1896, 1951, 2003,

2121, 2141, 2145, 2159, 2337, 2359, 2372, 2406, 2408, 1016 1051 1126 1318 1572 1917 1936

2039 2083 2136 2271 2317 2330，2121 2403

11、数论

1006,1014,1023,1061,1152,1183,1730,2262

12、几何有关的题目

凸包：1113, 1228, 1794, 2007, 2187,1113 wall，2187 beauty contest

容易：1319, 1654, 1673, 1675, 1836, 2074, 2137, 2318,

不易：1685, 1687, 1696, 1873, 1901, 2172, 2333,

13、任意精度运算、数字游戏、高精度计算

1001 1023 1047 1060 1079 1131 1140 1142 1207 1220 1284 1289 1306 1316 1338 1405 1454 1503

1504 1519 1565 1650 1969 2000 2006 2081 2247 2262 2305 2316 2389

1001, 1220, 1405, 1503,1001（高精度乘法） 2413(高精度加法，还有二分查找)

14、概率统计

1037,1050

15、小费用最大流、最大流

2195 going home，2400 supervisor, supervisee，1087 a plug for UNIX，1149 PIGS，1273 drainage

ditches，1274 the perfect stall，1325 machine schedule，1459 power network，2239 selecting

courses

16、压缩存储的DP

1038 bugs integrated inc，1185 炮兵阵地，2430 lazy cow

17、最长公共子串（LCS）

1080 human gene functions，1159 palindrome，1458 common subsequence，2192 zipper

18、图论及组合数学

2421 Constructing Roads、

2369 Permutations、

2234 Matches Game、

2243 Knight Moves、

2249 Binomial Showdown、

2255 Tree Recovery、

2084 Game of Connections、

1906 Three powers、

1833 排列、

1850 Code、

1562 Oil Deposits、

1496 Word Index、

1306 Combinations、

1125 Stockbroker Grapevine、

1129 Channel Allocation、

1146 ID Codes、

1095 Trees Made to Order、找规律

2247 Humble Numbers、

2309 BST、

2346 Lucky tickets、

2370 Democracy in danger、

2365 Rope、

2101 Honey and Milk Land

2028 When Can We Meet?、

2084 Game of Connections、

1915 Knight Moves、

1922 Ride to School、

1941 The Sierpinski Fractal、

1953 World Cup Noise、

1958 Strange Towers of Hanoi、

1969 Count on Canton、

1806 Manhattan 2025、

1809 Regetni、

1844 Sum、

1870 Bee Breeding、

1702 Eva\'s Balance、

1728 A flea on a chessboard、

1604 Just the Facts、

1642 Stacking Cubes、

1656 Counting Black、

1657 Distance on Chessboard、

1662 CoIns、

1663 Number Steps、

1313 Booklet Printing、

1316 Self Numbers、

1320 Street Numbers、

1323 Game Prediction、

1338 Ugly Numbers、

1244 Slots of Fun、

1250 Tanning Salon、

1102 LC-Display、

1147 Binary codes、

1013 Counterfeit Dollar、

19、博弈类

1067 取石子游戏、

1740 A New Stone Game、

2234 Matches Game、

1082 Calendar Game 、

2348 Euclid\'s Game、

2413 How many Fibs?、

2419 Forest

20、简单、模拟题

1001 Exponentiation 、

1002 487-3279、

1003 Hangover 、

1701 Dissatisfying Lift、

2301 Beat the Spread!、

2304 Combination Lock、

2328 Guessing Game、

2403 Hay Points 、

2406 Power Strings、

2339 Rock, Scissors, Paper、

2350 Above Average、

2218 Does This Make Me Look Fat?、

2260 Error Correction、

2262 Goldbach\'s Conjecture、

2272 Bullseye、

2136 Vertical Histogram、

2174 Decoding Task、

2183 Bovine Math Geniuses、

2000 Gold Coins、

2014 Flow Layout、

2051 Argus、

2081 Calendar、

1918 Ranking List、

1922 Ride to School、

1970 The Game、

1972 Dice Stacking、

1974 The Happy Worm、

1978 Hanafuda Shuffle、

1979 Red and Black、

1617 Crypto Columns、

1666 Candy Sharing Game、

1674 Sorting by Swapping、

1503 Integer Inquiry、

1504 Adding Reversed Numbers、

1528 Perfection、

1546 Basically Speaking、

1547 Clay Bully、

1573 Robot Motion、

1575 Easier Done Than Said?、

1581 A Contesting Decision、

1590 Palindromes、

1454 Factorial Frequencies、

1363 Rails、

1218 THE DRUNK JAILER、

1281 MANAGER、

1132 Border、

1028 Web Navigation、

21、初等数学

1003 Hangover、

1045 Bode Plot、

1254 Hansel and Grethel、

1269 Intersecting Lines、

1401 Factorial、

1410 Intersection、

2363 Blocks 、

2365 Rope、

2242 The Circumference of the Circle、

2291 Rotten Ropes、

2295 A DP Problem、

2126 Factoring a Polynomial、

2191 Mersenne Composite Numbers、

2196 Specialized Four-Digit Numbers、

1914 Cramer\'s Rule、

1835 宇航员、

1799 Yeehaa!、

1607 Deck、

1244 Slots of Fun、

1269 Intersecting Lines、

1299 Polar Explorer、

1183 反正切函数的应用、

22、匹配

1274, 1422, 1469, 1719, 2060, 2239,

-------------------------------------------------------------------------------------------

经典

1011（搜索好题）

1012（学会打表）

1013

1019（它体现了很多此类问题的特点）

1050（绝对经典的dp）

1088（dp好题）

1157（花店，经典的dp）

1163（怎么经典的dp那么多呀？？？）

1328（贪心）

1458（最长公共子序列）

1647（很好的真题，考临场分析准确和下手迅速）

1654（学会多边形面积的三角形求法）

1655（一类无根树的dp问题）

1804（逆序对）

2084（经典组合数学问题）

2187（用凸包求最远点对，求出凸包后应该有O(N)的求法，可我就是调不出来）

2195（二分图的最佳匹配）

2242（计算几何经典）

2295（等式处理）

2353（dp，但要记录最佳路径）

2354（立体解析几何）

2362（搜索好题）

2410（读懂题是关键）

2411（经典dp）

趣味

1067（很难的数学，但仔细研究，是一片广阔的领域）

1147（有O(n)的算法，需要思考）

1240（直到一棵树的先序和后序遍历，那么有几种中序遍历呢？dp）

1426（是数论吗？错，是图论！）

1648（别用计算几何，用整点这个特点绕过精度的障碍吧）

1833（找规律）

1844（貌似dp或是搜索，其实是道有趣的数学题）

1922（贪心，哈哈）

2231

2305（不需要高精度噢）

2328（要仔细噢）

2356（数论知识）

2359（约瑟夫问题变种）

2392（有趣的问题）

很繁的题

1001

1008

1087（构图很烦，还有二分图的最大匹配）

1128（USACO）

1245

1329

1550（考的是读题和理解能力）

1649（dp）

2200（字符串处理+枚举）

2358（枚举和避免重复都很烦）

2361（仔细仔细再仔细）

难题

1014（数学证明比较难，但有那种想法更重要）

1037（比较难的dp）

1405（高精度算法也分有等级之分，不断改进吧）

2002（不知道有没有比O(n^2\*logn)更有的算法？）

2054（极难，很强的思考能力）

2085（组合数学）

2414（dp，但要剪枝）

2415（搜索）

2423（计算几何+统计）

多解题

1002（可以用排序，也可以用统计的方法）

1338（搜索和dp都可以）

1664（搜索和dp都练一练吧）

2082（这可是我讲的题噢）

2352（桶排和二叉树都行）

Note:

1011: 很经典的剪支

1014: 难在数学上

1017: 严格的数学证明貌似不容易

1021: 有点繁,考察对图形进行各种旋转的处理

1083: 巧妙的思考角度

1150: 分奇偶讨论,lg(n)算法

1218: 三行就够了,虽然简单,但也有优劣之别

1505: 二分加贪心

1654: 做法也许很多吧,本人用有向面积做的

1674: 计算圈的个数(算是graph 吧)

1700: 数学证明不容易

1742: O(m\*n)的算法

1863: 要耐心地慢慢写…^\_^

1988: 并查集

2051: 堆

2078: 不难，但剪支可以做到很好

2082::O(n),你想到了吗？

2084: 卡特兰数

2182: 线段树

2195: 最小费用最大流

2234: 经典博弈算法

2236: 并查集

2299: 二分思想

2395: Kruskal 最小生成树的拓展

2406: KMP

2411: 用二进制串来表示状态

1、 排序

1423, 1694, 1723, 1727, 1763, 1788, 1828, 1838, 1840, 2201, 2376, 2377, 2380, 1318, 1877, 1928, 1971, 1974, 1990, 2001, 2002, 2092, 2379,

1002（需要字符处理，排序用快排即可） 1007（稳定的排序） 2159（题意较难懂） 2231

2371（简单排序） 2388（顺序统计算法） 2418（二\*排序树）

2、 搜索、回溯、遍历

1022,1111,1118,1129,1190,1562,1564,1573,1655,2184,2225,2243,2312,2362,2378,2386,1010,1011,1018,1020,1054,1062,1256,1321,1363,1501,1650,1659,1664,1753,2078,208

3,2303,2310,2329

简单：1128, 1166, 1176, 1231, 1256, 1270, 1321, 1543, 1606, 1664, 1731, 1742, 1745, 1847, 1915, 1950, 2038, 2157, 2182, 2183, 2381, 2386, 2426,

不易：1024, 1054, 1117, 1167, 1708, 1746, 1775, 1878, 1903, 1966, 2046, 2197, 2349,

推荐：1011, 1190, 1191, 1416, 1579, 1632, 1639, 1659, 1680, 1683, 1691, 1709, 1714, 1753, 1771, 1826, 1855, 1856, 1890, 1924, 1935, 1948, 1979, 1980, 2170, 2288, 2331, 2339, 2340,1979（和迷宫类似） 1980（对剪枝要求较高）

3、 历法

1008 2080 （这种题要小心）

4、 枚举

1012，1046， 1387， 1411， 2245， 2326， 2363， 2381，1054（剪枝要求较高），1650(小数的精度问题）

5、 数据结构的典型算法

容易：1182, 1656, 2021, 2023, 2051, 2153, 2227, 2236, 2247, 2352, 2395,

不易：1145, 1177, 1195, 1227, 1661, 1834,

推荐：1330, 1338, 1451, 1470, 1634, 1689, 1693, 1703, 1724, 1988, 2004, 2010, 2119, 2274, 1125(弗洛伊德算法) ，2421（图的最小生成树）

6、 动态规划

1037 A decorative fence、

1050 To the Max、

1088 滑雪、

1125 Stockbroker Grapevine、

1141 Brackets Sequence、

1159 Palindrome、

1160 Post Office、

1163 The Triangle、

1458 Common Subsequence、

1579 Function Run Fun、

1887 Testing the CATCHER、

1953 World Cup Noise、

2386 Lake Counting

动态规划

容易：

1018, 1050, 1083, 1088, 1125, 1143, 1157, 1163, 1178, 1179, 1189, 1208, 1276, 1322, 1414, 1456, 1458, 1609, 1644, 1664, 1690, 1699, 1740, 1742, 1887, 1926, 1936, 1952, 1953, 1958, 1959, 1962, 1975, 1989, 2018, 2029, 2033, 2063, 2081, 2082, 2181, 2184, 2192, 2231, 2279, 2329, 2336, 2346, 2353, 2355, 2356, 2385, 2392, 2424,

不易：

1019, 1037, 1080, 1112, 1141, 1170, 1192, 1239, 1655, 1695, 1707, 1733, 1737, 1837, 1850, 1920, 1934, 1937, 1964, 2039, 2138, 2151, 2161, 2178,

推荐：

1015, 1635, 1636, 1671, 1682, 1692, 1704, 1717, 1722, 1726, 1732, 1770, 1821, 1853, 1949, 2019, 2127, 2176, 2228, 2287, 2342, 2374, 2378, 2384, 2411,

7、 贪心

1042, 1065, 1230, 1784,1328 1755（或用单纯形方法），2054，1017， 1328，1862， 1922 ，2054， 2209， 2313， 2325， 2370。

8、 模拟

容易：1006, 1008, 1013, 1016, 1017, 1169, 1298, 1326, 1350, 1363, 1676, 1786, 1791, 1835, 1970, 2317, 2325, 2390,

不易：1012, 1082, 1099, 1114, 1642, 1677, 1684, 1886,1281 1928 2083 2141 2015

9、 递归

1664

10、字符串处理

1488, 1598, 1686, 1706, 1747, 1748, 1750, 1760, 1782, 1790, 1866, 1888, 1896, 1951, 2003, 2121, 2141, 2145, 2159, 2337, 2359, 2372, 2406, 2408, 1016 1051 1126 1318 1572 1917 1936 2039 2083 2136 2271 2317 2330，2121 2403

11、数论

1006,1014,1023,1061,1152,1183,1730,2262

12、几何有关的题目

凸包：1113, 1228, 1794, 2007, 2187,1113 wall，2187 beauty contest

容易：1319, 1654, 1673, 1675, 1836, 2074, 2137, 2318,

不易：1685, 1687, 1696, 1873, 1901, 2172, 2333,

13、任意精度运算、数字游戏、高精度计算

1001 1023 1047 1060 1079 1131 1140 1142 1207 1220 1284 1289 1306 1316 1338 1405 14541503 1504 1519 1565 1650 1969 2000 2006 2081 2247 2262 2305 2316 2389

1001, 1220, 1405, 1503,1001（高精度乘法） 2413(高精度加法，还有二分查找)

14、概率统计

1037,1050

15、小费用最大流、最大流

2195 going home，2400 supervisor, supervisee，1087 a plug for UNIX，1149 PIGS，1273 drainage ditches，1274 the perfect stall，1325 machine schedule，1459 power network，2239 selecting courses

16、压缩存储的DP

1038 bugs integrated inc，1185 炮兵阵地，2430 lazy cow

17、最长公共子串（LCS）

1080 human gene functions，1159 palindrome，1458 common subsequence，2192 zipper

18、图论及组合数学

2421 Constructing Roads、

2369 Permutations、

2234 Matches Game、

2243 Knight Moves、

2249 Binomial Showdown、

2255 Tree Recovery、

2084 Game of Connections、

1906 Three powers、

1833 排列、

1850 Code、

1562 Oil Deposits、

1496 Word Index、

1306 Combinations、

1125 Stockbroker Grapevine、

1129 Channel Allocation、

1146 ID Codes、

1095 Trees Made to Order、找规律

2247 Humble Numbers、

2309 BST、

2346 Lucky tickets、

2370 Democracy in danger、

2365 Rope、

2101 Honey and Milk Land

2028 When Can We Meet?、

2084 Game of Connections、

1915 Knight Moves、

1922 Ride to School、

1941 The Sierpinski Fractal、

1953 World Cup Noise、

1958 Strange Towers of Hanoi、

1969 Count on Canton、

1806 Manhattan 2025、

1809 Regetni、

1844 Sum、

1870 Bee Breeding、

1702 Eva\'s Balance、

1728 A flea on a chessboard、

1604 Just the Facts、

1642 Stacking Cubes、

1656 Counting Black、

1657 Distance on Chessboard、

1662 CoIns、

1663 Number Steps、

1313 Booklet Printing、

1316 Self Numbers、

1320 Street Numbers、

1323 Game Prediction、

1338 Ugly Numbers、

1244 Slots of Fun、

1250 Tanning Salon、

1102 LC-Display、

1147 Binary codes、

1013 Counterfeit Dollar、

19、博弈类

1067 取石子游戏、

1740 A New Stone Game、

2234 Matches Game、

1082 Calendar Game 、

2348 Euclid\'s Game、

2413 How many Fibs?、

2419 Forest

20、简单、模拟题

1001 Exponentiation 、

1002 487-3279、

1003 Hangover 、

1701 Dissatisfying Lift、

2301 Beat the Spread!、

2304 Combination Lock、

2328 Guessing Game、

2403 Hay Points 、

2406 Power Strings、

2339 Rock, Scissors, Paper、

2350 Above Average、

2218 Does This Make Me Look Fat?、

2260 Error Correction、

2262 Goldbach\'s Conjecture、

2272 Bullseye、

2136 Vertical Histogram、

2174 Decoding Task、

2183 Bovine Math Geniuses、

2000 Gold Coins、

2014 Flow Layout、

2051 Argus、

2081 Calendar、

1918 Ranking List、

1922 Ride to School、

1970 The Game、

1972 Dice Stacking、

1974 The Happy Worm、

1978 Hanafuda Shuffle、

1979 Red and Black、

1617 Crypto Columns、

1666 Candy Sharing Game、

1674 Sorting by Swapping、

1503 Integer Inquiry、

1504 Adding Reversed Numbers、

1528 Perfection、

1546 Basically Speaking、

1547 Clay Bully、

1573 Robot Motion、

1575 Easier Done Than Said?、

1581 A Contesting Decision、

1590 Palindromes、

1454 Factorial Frequencies、

1363 Rails、

1218 THE DRUNK JAILER、

1281 MANAGER、

1132 Border、

1028 Web Navigation、

21、初等数学

1003 Hangover、

1045 Bode Plot、

1254 Hansel and Grethel、

1269 Intersecting Lines、

1401 Factorial、

1410 Intersection、

2363 Blocks 、

2365 Rope、

2242 The Circumference of the Circle、

2291 Rotten Ropes、

2295 A DP Problem、

2126 Factoring a Polynomial、

2191 Mersenne Composite Numbers、

2196 Specialized Four-Digit Numbers、

1914 Cramer\'s Rule、

1835 宇航员、

1799 Yeehaa!、

1607 Deck、

1244 Slots of Fun、

1269 Intersecting Lines、

1299 Polar Explorer、

1183 反正切函数的应用、

22、匹配

1274, 1422, 1469, 1719, 2060, 2239

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

经典

1011（搜索好题）

1012（学会打表）

1013

1019（它体现了很多此类问题的特点）

1050（绝对经典的dp）

1088（dp好题）

1157（花店，经典的dp）

1163（怎么经典的dp那么多呀？？？）

1328（贪心）

1458（最长公共子序列）

1647（很好的真题，考临场分析准确和下手迅速）

1654（学会多边形面积的三角形求法）

1655（一类无根树的dp问题）

1804（逆序对）

2084（经典组合数学问题）

2187（用凸包求最远点对，求出凸包后应该有O(N)的求法，可我就是调不出来）

2195（二分图的最佳匹配）

2242（计算几何经典）

2295（等式处理）

2353（dp，但要记录最佳路径）

2354（立体解析几何）

2362（搜索好题）

2410（读懂题是关键）

2411（经典dp）

趣味

1067（很难的数学，但仔细研究，是一片广阔的领域）

1147（有O(n)的算法，需要思考）

1240（直到一棵树的先序和后序遍历，那么有几种中序遍历呢？dp）

1426（是数论吗？错，是图论！）

1648（别用计算几何，用整点这个特点绕过精度的障碍吧）

1833（找规律）

1844（貌似dp或是搜索，其实是道有趣的数学题）

1922（贪心，哈哈）

2231

2305（不需要高精度噢）

2328（要仔细噢）

2356（数论知识）

2359（约瑟夫问题变种）

2392（有趣的问题）

很繁的题

1001

1008

1087（构图很烦，还有二分图的最大匹配）

1128（USACO）

1245

1329

1550（考的是读题和理解能力）

1649（dp）

2200（字符串处理+枚举）

2358（枚举和避免重复都很烦）

2361（仔细仔细再仔细）

难题

1014（数学证明比较难，但有那种想法更重要）

1037（比较难的dp）

1405（高精度算法也分有等级之分，不断改进吧）

2002（不知道有没有比O(n^2\*logn)更有的算法？）

2054（极难，很强的思考能力）

2085（组合数学）

2414（dp，但要剪枝）

2415（搜索）

2423（计算几何+统计）

多解题

1002（可以用排序，也可以用统计的方法）

1338（搜索和dp都可以）

1664（搜索和dp都练一练吧）

2082（这可是我讲的题噢）

2352（桶排和二\*树都行）

Note:

1011: 很经典的剪支

1014: 难在数学上

1017: 严格的数学证明貌似不容易

1021: 有点繁,考察对图形进行各种旋转的处理

1083: 巧妙的思考角度

1150: 分奇偶讨论,lg(n)算法

1218: 三行就够了,虽然简单,但也有优劣之别

1505: 二分加贪心

1654: 做法也许很多吧,本人用有向面积做的

1674: 计算圈的个数(算是graph 吧)

1700: 数学证明不容易

1742: O(m\*n)的算法

1863: 要耐心地慢慢写…^\_^

1988: 并查集

2051: 堆

2078: 不难，但剪支可以做到很好

2082::O(n),你想到了吗？

2084: 卡特兰数

2182: 线段树

2195: 最小费用最大流

2234: 经典博弈算法

2236: 并查集

2299: 二分思想

2395: Kruskal 最小生成树的拓展

2406: KMP

2411: 用二进制串来表示状态

主流算法：

1.搜索　//回溯

2.DP（动态规划）

3.贪心

4.图论　//Dijkstra、最小生成树、网络流

5.数论　//解模线性方程

6.计算几何　//凸壳、同等安置矩形的并的面积与周长

7.组合数学　//Polya定理

8.模拟

9.数据结构　//并查集、堆

10.博弈论

1、 排序

1423, 1694, 1723, 1727, 1763, 1788, 1828, 1838, 1840, 2201, 2376, 2377, 2380, 1318, 1877,

1928, 1971, 1974, 1990, 2001, 2002, 2092, 2379,

1002（需要字符处理，排序用快排即可） 1007（稳定的排序） 2159（题意较难懂）

2231 2371（简单排序） 2388（顺序统计算法） 2418（二叉排序树）

2、 搜索、回溯、遍历

1022 1111 1118 1129 1190 1562 1564 1573 1655 2184 2225 2243 2312 2362 2378 2386

1010,1011,1018,1020,1054,1062,1256,1321,1363,1501,1650,1659,1664,1753,2078,2083,2303,2310,2329

简单：1128, 1166, 1176, 1231, 1256, 1270, 1321, 1543, 1606, 1664, 1731, 1742, 1745, 1847,

1915, 1950, 2038, 2157, 2182, 2183, 2381, 2386, 2426,

不易：1024, 1054, 1117, 1167, 1708, 1746, 1775, 1878, 1903, 1966, 2046, 2197, 2349,

推荐：1011, 1190, 1191, 1416, 1579, 1632, 1639, 1659, 1680, 1683, 1691, 1709, 1714, 1753,

1771, 1826, 1855, 1856, 1890, 1924, 1935, 1948, 1979, 1980, 2170, 2288, 2331, 2339,

2340,1979（和迷宫类似） 1980（对剪枝要求较高）

3、 历法

1008 2080 （这种题要小心）

4、 枚举

1012，1046， 1387， 1411， 2245， 2326， 2363， 2381，

1054（剪枝要求较高），1650 （小数的精度问题）

5、 数据结构的典型算法

容易：1182, 1656, 2021, 2023, 2051, 2153, 2227, 2236, 2247, 2352, 2395,

不易：1145, 1177, 1195, 1227, 1661, 1834,

推荐：1330, 1338, 1451, 1470, 1634, 1689, 1693, 1703, 1724, 1988, 2004, 2010, 2119, 2274,

1125(弗洛伊德算法) ，2421（图的最小生成树）

6、 动态规划

1037 A decorative fence、

1050 To the Max、

1088 滑雪、

1125 Stockbroker Grapevine、

1141 Brackets Sequence、

1159 Palindrome、

1160 Post Office、

1163 The Triangle、

1458 Common Subsequence、

1579 Function Run Fun、

1887 Testing the CATCHER、

1953 World Cup Noise、

2386 Lake Counting

7、 贪心

1042, 1065, 1230, 1323, 1477, 1716, 1784,1328 1755（或用单纯形方法），2054，1017， 1328，

1862， 1922 ，2054， 2209， 2313， 2325， 2370。

8、 模拟

容易：1006, 1008, 1013, 1016, 1017, 1169, 1298, 1326, 1350, 1363, 1676, 1786, 1791, 1835,

1970, 2317, 2325, 2390

不易：1012, 1082, 1099, 1114, 1642, 1677, 1684, 1886,1281 1928 2083 2141 2015

9、 递归

1664

10、字符串处理

1488, 1598, 1686, 1706, 1747, 1748, 1750, 1760, 1782, 1790, 1866, 1888, 1896, 1951, 2003,

2121, 2141, 2145, 2159, 2337, 2359, 2372, 2406, 2408, 1016 1051 1126 1318 1572 1917 1936

2039 2083 2136 2271 2317 2330，2121 2403

11、数论

1006,1014,1023,1061,1152,1183,1730,2262

12、几何有关的题目

凸包：1113, 1228, 1794, 2007, 2187,1113 wall，2187 beauty contest

容易：1319, 1654, 1673, 1675, 1836, 2074, 2137, 2318,

不易：1685, 1687, 1696, 1873, 1901, 2172, 2333,

13、任意精度运算、数字游戏、高精度计算

1001 1023 1047 1060 1079 1131 1140 1142 1207 1220 1284 1289 1306 1316 1338 1405 1454 1503

1504 1519 1565 1650 1969 2000 2006 2081 2247 2262 2305 2316 2389

1001, 1220, 1405, 1503,1001（高精度乘法） 2413(高精度加法，还有二分查找)

14、概率统计

1037,1050

15、小费用最大流、最大流

2195 going home，2400 supervisor, supervisee，1087 a plug for UNIX，1149 PIGS，

1273 drainage ditches，1274 the perfect stall，1325 machine schedule，

1459 power network，2239 selecting courses

16、压缩存储的DP

1038 bugs integrated inc，1185 炮兵阵地，2430 lazy cow

17、最长公共子串（LCS）

1080 human gene functions，1159 palindrome，1458 common subsequence，2192 zipper

18、图论及组合数学

2421 Constructing Roads、

2369 Permutations、

2234 Matches Game、

2243 Knight Moves、

2249 Binomial Showdown、

2255 Tree Recovery、

2084 Game of Connections、

1906 Three powers、

1833 排列、

1850 Code、

1562 Oil Deposits、

1496 Word Index、

1306 Combinations、

1125 Stockbroker Grapevine、

1129 Channel Allocation、

1146 ID Codes、

1095 Trees Made to Order、找规律

2247 Humble Numbers、

2309 BST、

2346 Lucky tickets、

2370 Democracy in danger、

2365 Rope、

2101 Honey and Milk Land

2028 When Can We Meet?、

2084 Game of Connections、

1915 Knight Moves、

1922 Ride to School、

1941 The Sierpinski Fractal、

1953 World Cup Noise、

1958 Strange Towers of Hanoi、

1969 Count on Canton、

1806 Manhattan 2025、

1809 Regetni、

1844 Sum、

1870 Bee Breeding、

1702 Eva\'s Balance、

1728 A flea on a chessboard、

1604 Just the Facts、

1642 Stacking Cubes、

1656 Counting Black、

1657 Distance on Chessboard、

1662 CoIns、

1663 Number Steps、

1313 Booklet Printing、

1316 Self Numbers、

1320 Street Numbers、

1323 Game Prediction、

1338 Ugly Numbers、

1244 Slots of Fun、

1250 Tanning Salon、

1102 LC-Display、

1147 Binary codes、

1013 Counterfeit Dollar、

19、博弈类

1067 取石子游戏、

1740 A New Stone Game、

2234 Matches Game、

1082 Calendar Game 、

2348 Euclid\'s Game、

2413 How many Fibs?、

2419 Forest

20、简单、模拟题

1001 Exponentiation 、

1002 487-3279、

1003 Hangover 、

1701 Dissatisfying Lift、

2301 Beat the Spread!、

2304 Combination Lock、

2328 Guessing Game、

2403 Hay Points 、

2406 Power Strings、

2339 Rock, Scissors, Paper、

2350 Above Average、

2218 Does This Make Me Look Fat?、

2260 Error Correction、

2262 Goldbach\'s Conjecture、

2272 Bullseye、

2136 Vertical Histogram、

2174 Decoding Task、

2183 Bovine Math Geniuses、

2000 Gold Coins、

2014 Flow Layout、

2051 Argus、

2081 Calendar、

1918 Ranking List、

1922 Ride to School、

1970 The Game、

1972 Dice Stacking、

1974 The Happy Worm、

1978 Hanafuda Shuffle、

1979 Red and Black、

1617 Crypto Columns、

1666 Candy Sharing Game、

1674 Sorting by Swapping、

1503 Integer Inquiry、

1504 Adding Reversed Numbers、

1528 Perfection、

1546 Basically Speaking、

1547 Clay Bully、

1573 Robot Motion、

1575 Easier Done Than Said?、

1581 A Contesting Decision、

1590 Palindromes、

1454 Factorial Frequencies、

1363 Rails、

1218 THE DRUNK JAILER、

1281 MANAGER、

1132 Border、

1028 Web Navigation、

21、初等数学

1003 Hangover、

1045 Bode Plot、

1254 Hansel and Grethel、

1269 Intersecting Lines、

1401 Factorial、

1410 Intersection、

2363 Blocks 、

2365 Rope、

2242 The Circumference of the Circle、

2291 Rotten Ropes、

2295 A DP Problem、

2126 Factoring a Polynomial、

2191 Mersenne Composite Numbers、

2196 Specialized Four-Digit Numbers、

1914 Cramer\'s Rule、

1835 宇航员、

1799 Yeehaa!、

1607 Deck、

1244 Slots of Fun、

1269 Intersecting Lines、

1299 Polar Explorer、

1183 反正切函数的应用、

22、匹配

1274, 1422, 1469, 1719, 2060, 2239,

-------------------------------------------------------------------------------------------

经典

1011（搜索好题）

1012（学会打表）

1013

1019（它体现了很多此类问题的特点）

1050（绝对经典的dp）

1088（dp好题）

1157（花店，经典的dp）

1163（怎么经典的dp那么多呀？？？）

1328（贪心）

1458（最长公共子序列）

1647（很好的真题，考临场分析准确和下手迅速）

1654（学会多边形面积的三角形求法）

1655（一类无根树的dp问题）

1804（逆序对）

2084（经典组合数学问题）

2187（用凸包求最远点对，求出凸包后应该有O(N)的求法，可我就是调不出来）

2195（二分图的最佳匹配）

2242（计算几何经典）

2295（等式处理）

2353（dp，但要记录最佳路径）

2354（立体解析几何）

2362（搜索好题）

2410（读懂题是关键）

2411（经典dp）

趣味

1067（很难的数学，但仔细研究，是一片广阔的领域）

1147（有O(n)的算法，需要思考）

1240（直到一棵树的先序和后序遍历，那么有几种中序遍历呢？dp）

1426（是数论吗？错，是图论！）

1648（别用计算几何，用整点这个特点绕过精度的障碍吧）

1833（找规律）

1844（貌似dp或是搜索，其实是道有趣的数学题）

1922（贪心，哈哈）

2231

2305（不需要高精度噢）

2328（要仔细噢）

2356（数论知识）

2359（约瑟夫问题变种）

2392（有趣的问题）

很繁的题

1001

1008

1087（构图很烦，还有二分图的最大匹配）

1128（USACO）

1245

1329

1550（考的是读题和理解能力）

1649（dp）

2200（字符串处理+枚举）

2358（枚举和避免重复都很烦）

2361（仔细仔细再仔细）

难题

1014（数学证明比较难，但有那种想法更重要）

1037（比较难的dp）

1405（高精度算法也分有等级之分，不断改进吧）

2002（不知道有没有比O(n^2\*logn)更有的算法？）

2054（极难，很强的思考能力）

2085（组合数学）

2414（dp，但要剪枝）

2415（搜索）

2423（计算几何+统计）

多解题

1002（可以用排序，也可以用统计的方法）

1338（搜索和dp都可以）

1664（搜索和dp都练一练吧）

2082（这可是我讲的题噢）

2352（桶排和二叉树都行）

Note:

1011: 很经典的剪支

1014: 难在数学上

1017: 严格的数学证明貌似不容易

1021: 有点繁,考察对图形进行各种旋转的处理

1083: 巧妙的思考角度

1150: 分奇偶讨论,lg(n)算法

1218: 三行就够了,虽然简单,但也有优劣之别

1505: 二分加贪心

1654: 做法也许很多吧,本人用有向面积做的

1674: 计算圈的个数(算是graph 吧)

1700: 数学证明不容易

1742: O(m\*n)的算法

1863: 要耐心地慢慢写…^\_^

1988: 并查集

2051: 堆

2078: 不难，但剪支可以做到很好

2082::O(n),你想到了吗？

2084: 卡特兰数

2182: 线段树

2195: 最小费用最大流

2234: 经典博弈算法

2236: 并查集

2299: 二分思想

2395: Kruskal 最小生成树的拓展

2406: KMP

2411: 用二进制串来表示状态