**题目解读**

1. 多用STL等系统函数库。
2. 格式控制要求非常严格，末尾不能空格。
3. 一般情况下数据量都很大，接近MAXN值。
4. 要考虑开发成本问题，
   1. 按最大可能性定义数组。
      1. 一般测试数据都很可能接近给定边界
   2. 复用不多的时候不用函数。

**一些技巧**

1. 数据输入输出
   1. ccanf效率比 cin 要高，要习惯用scanf
   2. cin cout 做流处理 比scanf好用
2. 尽量使用**静态数组**
3. **避免**使用**指针**
4. 多用**全局变量**
5. 当输入数据很多时，建议使用scanf()

Huge input data,scanf is recommended.

----- 某题目中给的提示

1. 输出时候可以用scanf
2. 不要混合使用scanf()和cin。
3. 考虑到这是竞赛题，要考虑时间成本
   1. 非特殊情况不要建非常复杂的模型，如果模型十分复杂多半是方向错了。
   2. 简洁的模型一般具有更高的执行效率。
4. 需要考虑特殊情况吗？
   1. 需要，因为题目数据多数都是随机生成的，很可能撞到那些特殊值。
5. 建模就是将问题转化、抽象、简化的过程。

**一些常识**

* + ACM中很多难题都需要穷举
  + P类问题：可简化问题，蚂蚁问题
  + NP问题： 非常难的问题

解决方法：穷举（因为没法化简，只是穷举中有些小技巧）

一些思路

* + 穷举法，**最先考虑**，**最后使用**。
    - 有些问题必须穷举（输出自然数），有些问题有非穷举的简单解。

**一些常用数据结构**

* + 栈 先进后出【逆序输出】
  + 队列 先进先出

**一些算法【常见问题】**

1. **搜索、查找**

* **遍历（枚举）（暴力搜索） O(n)**
* **二分法 O(log n)**

1. **排序**

* **STL sort(),qsort()函数**
* **快速排序法**

1. **搜索问题 【迷宫】**
   * 1. **深度优先**搜索

沿着一种状态一直往下走，走不下去就退回，走通了就结束。

* 针对的问题：一条路走到底就能解决的问题。

例如，两个节点之间有没有一条可行路径。

例如，能不能走出迷宫。

广度优先搜索

* 加边框【墙壁】 减少判断次数

DFS函数：

递归、标记、考虑邻居

1. 广度优先搜索

每次尽可能考虑最多的情况

1. 怎么记录路径？
2. 怎么避免重复？
3. 怎么避免走回头路？

【最短路径】

1. **剪枝**

**基本含义就是： 在枚举过程中，排除一些绝对不可能的 无需计算的情形。**

**一般为了提高程序执行效率，在各种算法中都会或多或少的使用剪枝。**

**例如： 木棍拼接问题**

二元组（剩余棍子数，当前在拼的棍子缺少的长度）

(N,L) -> (0,0)

最小值 max(L1,L2...) 最大值 sum(L1,L2...)/2

棍子排序，从大到小开始拼

不要多次尝试相同长度木棒【相同长度木棒，如果失败了，下一根相同长度舍弃】

(棍子总长 % 当前目标长度) 应该=0

实现： used int[] 记录棍子使用情况。

1. **A\* 算法？**

启发式算法。

BFS算法中，对每个状态n都设定**估价函数f(n)**=g(n)+h(n)

**g(n)** 从初始节到当前节点的步数

**h(n)** 当前节点不符合要求的特征数量

1. **模拟队列**

实质：数组

成员

队头指针 head

队尾指针 tail

操作

头插入、尾插入

头删除、尾删除 【移动头、尾指针】

中间插入【需要移动】

1. **经常用数学知识**
2. **一些常见错误**

**Memory Out of Limit**

超出了内存限制，一般只有某些题才会加这个限制，否则64m内存完全够用了。



**WA**：wrong answer

**RE**：Runtime Error

**常见名词概念**

INF：Infinite 无限，无穷大 用于讨论需要考虑无穷大值的情况

INF不要取到int型的最大值，因为还需要考虑 ∞+∞ = ∞ 的情况 也就是说，INF值不能取2^31-1之类的 但可以是 2^30左右

**solve()** ：一键生成的能解决问题的函数\_

**visited[]**: 图算法常用，记录是否访问过

**DFS**：深度优先

**BFS**： Broad First Search 广度优先

**M,N,K,L,T...** ： 已知条件常用字母

**MAXN**：最大数据量，最大值限制

**学习网站**

**poj.org**

**常识**

1. **只存两种状态 ： 用bool**
2. **If(t!=0) 比 if(t) 更好**
3. **尽量避免指针操作。**
4. **递归程序理解困难，写起来简单**