POJ 2245【基础】

题目大意：
按升序给出 n 个（6<=n<=13）数字，在里面选出 6 个，按字典序排序输出所有选出的组合。

输入：
有若干组测试数据。每一组测试数据第一行有一个整数 n，接下来有 n 个空格分隔的升序排列的整数。n=0 时测试数据结束。

输出：
每一个组合输出一行，用空格分隔六个整数。
两组测试数据之间有一个空行。

题解：
简单的用深搜求排列组合问题。

POJ 1754【基础】

题目大意：
有一个 4×4 的棋盘，一开始上面每一格都放有一些纸片，纸片非黑即白，而且其反面是另外一种颜色。问最少翻动多少次，才能使得棋盘的颜色统一。

输入：
四行，每一行有四个英文字母，w 为白色，b 为黑色。

输出：
如果可以使棋盘颜色统一，则输出一行，包含一个整数，否则输出一行“Impossible”。

题解：
简单的深搜和回溯，没有涉及到剪枝。

POJ 3256【基础】

题目大意：
有 K 头牛（1<=K<=100），一开始分布在 N 个牧场（1<=N<=1000）中，这些牧场之间有 M 条（1<=M<=10000)有向边连接。问有多少牧场可以让所有的奶牛聚集到一起。

输入：
第一行有三个整数 K、N 和 M。接下来有 K 行，每一行有一个整数，表示一只奶牛在哪个牧场。下面有 M 行，每一行有两个整数，表示一条两个牧场之间的单向边。

输出：
一行，一个整数，即有多少牧场可以让所有的奶牛聚集到一起。

题解：
由于 N^3 的规模太大，所以不能直接用 Floyd。记录下所有有牛的牧场，以这些牧场作为起始点进行深搜，维护 vis 数组防止深搜过程中形成环。搜到的每一个点，在 sum 中加上出发点的牛的数量。这样最后得到的 sum 数组就是每个点有多少牛能够到达了。

POJ 1020【基础】

题目大意：
有一个 s*s（1<=s<=50）的正方形，要用若干个给定宽度的小正方形填充满它，问是否能做得到。

输入：
第一行有一个整数 T，表示有 T 组测试数据。
每一组测试数据一行，有若干个整数。第一个整数是大正方形的边长，第二个整数是小正方形的个数 N（1<=N<=16）。接下来有 N 个整数，每一个代表一个小正方形的宽度，这个宽度的范围在 1～10。

输出：
每一组测试数据输出一行，如果可以用小正方形填充，则输出“KHOOOOB”，否则输出“HUTUTU”。

题解：
这题思路不难，甚至可以说就是贪心的思路：维护一个数组 pos，表示每一行前面多少个格子已经被填满了，然后每一次从被填满格数最少的位置来放。因为所有位置都是要被填满的，如果不从最“低”的位置开始放，那么可能就会因为放进去的方块太大造成有空隙出现。为此，需要在搜索的时候按已有方格从大到小来进行，这是一个剪枝，也是一个保障条件。因为先试大方格，更容易排除掉一些明显不符合的条件，及时回溯。

POJ 1054【基础】

题目大意：
有一个 r*c（1<=r*c<=5000）的矩阵，里面有一些位置被跳过的青蛙踩上了脚印。已知每一只青蛙都是从矩形外面跳入矩形然后再跳出去的，跳的方向和步长不变。问在矩形内跳得最多步数的青蛙可能跳了多少步。

输入：
第一行有两个整数 r 和 c。第二行有一个整数 N（1<=N<=5000），表示有 N 个点被青蛙踩上了脚印。下面有 N 行，每一行有两个整数，表示一个脚印的位置。

输出：
一行，一个整数，即矩形内跳得最多步数的青蛙可能跳了多少步。如果这个值小于 3，那么输出 0。

题解：
肯定是要枚举任意两个点作为最初的两点来求最大值的了。但是可以先排序，按 x 坐标优先 y 坐标第二的优先度升序排序。然后两两枚举，并且有如下剪枝：
1、如果当前枚举的起始位置加上当前的步长乘以当前最佳答案，将超过矩形的长/宽，那么直接换下一个起点来进行枚举；
2、如果当前枚举的起始位置减去当前的步长还在矩形内有脚印，那么换下一个第二步的位置，因为青蛙的起跳位置还在矩形内；
3、如果当前枚举的起始位置起跳，以当前的步长跳当前最佳答案步，超出矩形范围，那么换下一个第二步的位置。

POJ 2362【中等】

题目大意：
给出 N（4<=N<=20）根长度不一的木棍，求问能不能摆成一个正方形。

输入：
文件第一行有一个整数，表示有多少组测试数据。
每一组测试数据包含一行，第一个整数 N 表示有 N 根棍子。接下来有 N 个整数，表示每一根木棍的长度。

输出：
对每一组测试数据，输出一行，如果能构成一个正方形，则输出“yes”，否则输出“no”。

题解：
很经典的一道剪枝题，有三个最基本的剪枝：
1、总长度必须是 4 的倍数；
2、最长边必须小于等于边长；
3、对边进行排序，从长度长的开始搜索，这样速度快一点；
（*非必要）4、相同的长度只搜索一次。

POJ 1691【中等】

题目大意：
有一个矩形框由 n 个小的矩形拼成，现在要把每个小矩形涂上一种颜色 c（可相同可不同）。由于颜料没干时会流下来，所以每一个小矩形必须等它正上方相邻的小矩形都涂完了颜料才可以涂。但同一种颜色的话就没有关系，可以一次过涂两个或多个小矩形（只要下面的小矩形正对上方的小矩形都是同一种颜色）。涂不同的颜色需要换不同的刷子，同一种颜色先后涂也要换刷子，问最少需要多少把刷子。

输入：
第一行有一个整数 t，表示有 t 组测试数据。
每一组测试数据第一行有一个整数 n（1<=n<=15）。接下来有 n 行，每一行有五个整数 y1 x1 y2 x2 c，前四个表示一个矩形的左上角和右下角坐标。最小的左上角坐标是(0,0)，最大的右下角坐标不会超过(99,99)。c 是颜色编号。

输出：
每一组测试数据输出一行，包含一个整数，即最少需要多少把刷子。

题解：
这题有点拓扑排序的味道。由题意，可以根据上下关系建立一个有向边表和入度表，每涂一个小矩形就把这个点和出边拿掉，每一次涂的小矩形都是入度为0的。然后就是深搜了，每一次枚举一个入度为0的小矩形，搜索下去即可。

POJ 3411【中等偏上】

题目大意：
有n座城市和m条单向路(1<=n，m<=10)。现在要从城市1到城市n。有些路是要收费的，从a城市到b城市，如果之前到过c城市，那么只要付p的钱，如果没有去过就付r的钱。求最少要花多少钱。

输入：
第一行有两个整数n和m。下面有m行，每一行有五个整数a b c p r，描述了一条道路的信息。

输出：
一行，如果能够到达目的地，输出最小费用，否则输出impossible。

题解：
网上有说用SPFA的，但是我觉得深搜写起来更简明。这题的关键在于，有些路可能需要重复走，才能做到最省钱。那么就不能像以前一样用vis数组来标记哪些走过哪些没走过然后简单地判断不走已经走过的地方。这里需要升级一下，把vis数组改成记录每个点进入了多少次，超过一定的次数就可以肯定是不行的，肯定要回溯。那么这次次数，网上的说法是“闸数”，要怎么求出来呢？搜了一下网上的题解，都没有给出明确的结果。下面我简单说明一下：
首先，如果一个点被重复走过，一定是因为经过了一个环路回来。构造这么一种情况：起始点被重复经过，起始点到终点的路径上某条边初始费用非常大，经过某个环路上的点后这条边费用非常小，那么就必须要经过环路了。
要确定重复经过最多的情况，那么就要构造最多的环路，环路内除了被重复经过的点以外的点的数量最少。那么如果是1个点，来回需要两条边，去终点需要一条边，我们最多可以有10/3=3个环路。因此阈值设为3。根据网上的测试，由于数据比较弱，可以把阈值减少到2，也还是可以通过的。我测了一下，阈值为3时用了47ms，阈值为2时直接0ms。

POJ 1011【中等偏上】

题目大意、输入输出见原题，有中文。

题解：
这题是典型的考深搜剪枝，算是 POJ2362 的加强版。我给出了四个剪枝条件，在代码注释里有写。其中第四个我一开始没有想出来，导致了 TLE，不查都不知道。唉，看来自己深搜还是太弱了啊。

POJ 1190【中等偏上】

题目大意、输入输出见原题，有中文。

题解：
还是很典型的一题考深搜剪枝的，而且剪枝有三个方向：表面积、体积和搜索方向，分别体现对搜索结果的分析剪枝、对搜索过程中不符合要求的情况剪枝和搜索技巧上的剪枝。对表面积的剪枝是代码注释里的剪枝 2，对体积的剪枝是注释里的剪枝 1、3，搜索技巧上的剪枝是注释里的剪枝 4。其中剪枝 1、2 是我最早想到的；剪枝 3 一直没有想到，要上网查；剪枝 4 一开始写错了。我测试了一下，这四个剪枝去掉任何一个都会 TLE。另外，MinS 和 MinV 表示上面 x 层的最小体积和表面积，这个是可以打表出来的，因为取最小的条件是固定的。

POJ 2286【中等偏上】

题目大意：
有一个“井”字形的图案，其中的两横两竖都有七个格子，在第 3、5 个格子处交叉（强烈建议看原题图）。有八种操作（A~H），每一种操作可以令其中的一横或者一竖向某个方向循环移位一格（还是建议看原题图）。所有格子内的数字只可能是 1、2 或者 3。给出一个初始状态，问最少需要多少步，中间的八个格子都是一样的数字。

输入：
有若干行测试数据。如果一行测试数据第一个数字是 0，那么测试数据结束。否则，一共有 24 个数字，分别是从左上到右下的第几个数字的值。

输出：

每一组测试数据输出两行。第一行包含一个字符串，每一个字符表示一个操作。第二行包含一个数字，即中间 8 个格子的数字是什么。如果有多解，那么输出操作字典序最小的一个。

题解：
这题学习控制深度的 DFS 搜索。所谓控制深度的 DFS，一般用于那些可以无限深搜，但是广搜的话空间复杂度又太大的情况。限定深搜的深度从 1 开始增加，每一次搜索如果搜到了指定深度还没有结果就返回。这样虽然会导致前面的一些节点被重复搜索，但是避免了纯粹 DFS 的盲目性。
这一题我只用了一个很小的剪枝，即如果当前搜索深度+中心区域到目标的最少步数>最大深度就放弃。中心区域到目标的最少步数即为，中心区域中最多的那个数，比 8 小多少。因为每一次操作最多只能增加 1 个数，所以 8-最多的数的个数就是最少步数。

POJ 3373【难】

题目大意：
给出 2 个整数 n（n<10^100）和 k（k<10000），求满足以下条件的整数 m
1、m 与 n 位数相同
2、m 能被 k 整除
3、满足以上两点时，数字不同的位数最少
4、满足以上三点时，m 值最小

输入：
有若干组测试数据，处理到文件末尾。每一组测试数据有两行，第一行有一个整数 n，第二行有一个整数 k。

输出：
每一组测试数据输出一行，包含一个整数，即满足要求的 m。

题解：
不知道为何这题被分到了记忆化搜索里，但是我看了一下网上的题解，基本上都是强剪枝。这一题仅有一个剪枝，但是不加上这个剪枝则肯定超时，我一开始写出了朴素的方法来做没有加剪枝就超时了。朴素的方法很好想：枚举改变的位数，然后先找比 n 小的，再找比 n 大的：找比 n 小的时候，从最高位开始改变，然后看看改变的后的数是否可以 mod k=0 即可；找比 n 大的时候，从最低位开始改变，然后检测一下是否符合。
剪枝就在于，如果当前的搜索参数（搜索到的位置，剩余可以改变的位数）搜索失败了，那么记录 flag[当前搜索到的位置][当前搜索的数的 mod k]=当前剩

余可改变的位数，下一次进入相同的搜索位置和 mod k 值时，如果剩余可改变的位数小于已有记录，那么肯定不是最优解，直接返回。
判断 mod k=0 我从网上学了一种方法，可以避免每一次都做大整数的求余，原理是模同余。用 mod[i][j]表示 j*(10^i) mod k，那么由模同余可以得到 mod[i+1][j]=(mod[i][j]*10)%k。那么在改变某一位的时候，设当前的数 mod k 的结果为 mod_k，那么第 i 位从 n[i]变为 j 得到的新 mod k 值为 new_mod_k=(mod_k-(mod[i][n[i]]-mod[i][j])+k)%k，可以很快得出。
关键还是在于剪枝啊，如果没有那个剪枝就完全过不了了。