A. 双剑合并 （xor字典树）

> 给定两个序列，问从两个序列中各取一个值的异或和最大为多少

把 A序列中的数字看成一个二进制的 01串，然后存到 Trie树里
然后将 B序列的数字同样看成一个 01串，然后在 Trie树上从高位到低位贪心地查找
如果B序列中查找的 01串当前位为 0，则找 Trie树上为 1的子儿子
否则就只能走为 0的右儿子，反之亦然，然后往下走
时间复杂度 O(N)

B. 单词替换 （KMP）

> 给定一个字符串，把其中出现的 A串替换为 B串

KMP入门题，对原串匹配A串，跑一遍KMP
然后匹配到终点的时候替换就好了
最后再输出替换的结果
时间复杂度 O(N)

C. 01的时间 （数位DP）

> 给定一个数，求最小的仅由0和1构成的能被给定数整除的数

暴力 dfs，从高到低，从小到大枚举当前位，然后一直搜到底
最后判断一下余数是否等于 0，如果为 0，则返回
要注意前导 0的情况，即排除全为 0的解

其实就是个数位DP的水题，我没记忆化都让我水过了

D. GooZy的游戏时间 （基于搜索时间的剪枝）

> 给定一个 N*N的拼图，要求重新排列
> 使得一个方块上下左右相邻的方块连接处数字相等

这题写起来比较恶心
我刚开始通过把每一行可行状态存下来，丢到 set里转移，结果MLE了

后来换了种想法，蛇形地去找可连接的方块
然后尝试了若干剪枝，效果都不理想

自己生成了几组随机数据
发现当数字均为 0~3，比较集中时，跑得巨慢
本来 0~3这种可重复的概率比较大
相应地能搜出解的可能也比较大
但是一但无解，他就会重复地去搜，十分浪费时间

基于以上事实，我用了一个玄学剪枝
我设定了一个搜索次数，当 dfs的调用次数过大时，当做无解返回
大概 WA了三四发的样子，尝试出了比较好的参数，72ms就跑过了

后来我找 zw菊苣问了标程，发现标程也是基于上述情况进行的特殊剪枝
不过他的剪枝可比我的科学多了

E. RunningPhoton’s Nightmare （BFS预处理+SPFA）

> 给定一张网格图，其中有一些不可到达点和一些时间重置装置
> RunningPhoton从起点出发，身上有一个定时炸弹，当时间置0时他就会死
> 但是在置0前碰到时间重置装置又能重置时间
> 问 RunningPhoton是否能到达终点
> 若能，则输出最短时间，若不能，则输出 “Poor RunningPhoton”

这题虽然地图是有 600*600，但是有不超过 150个重置装置
普通 bfs搜的话肯定爆炸，因为你要存每个装置是否被用过了
正确解法如下：
因为我们只关心重置装置，起点，终点的位置
所以以每个重置装置及起点，终点作为起点都跑一次 bfs，
计算出两两间的最短距离后构一张图，将距离小于 K的连上边
最后再从起点跑一次 spfa即可

但实际上bfs搜也能过，就看你怎么搜了

F. 表达式 （IDA*）

> 已有 x，求利用除法和乘法算出 x^n的最小步数

IDA* 迭代加深地去搜
如果能在step步得出解，那么大于这个步数的也一定有解
所以从小到大枚举步数，然后暴力去搜
由于限定步数，就可以加一个 A*的估价剪枝
当前最值反复平方也不能在限定步数得出解的时候，则不往下搜
由于N只有1000,而 2^10 > 1000，所以最终步数其实不会很大

G. 神舟的宝藏 （数位DP）

> 求一个最小的 C进制数，使得他满足
> 由给定的 M的数组成，最大长度不超过500，并且能被 N整除

依旧是数位DP裸题
从高到低，从小到大枚举每一位
枚举到底的时候判断余数是否为 0

比较麻烦的是判断前导 0：
由于前导 0不算在给定的数字内
所以先单独计算一次此位为前导 0的，然后再进行枚举

最后要记忆化一下，如果当前 i位，余数为 rem搜不到解
就标记一下以后不搜了

有一个 trick是 N==0的情形，特判一下就好了

H. DNA序列 （状压DP）

> 给定若干个DNA序列，求最短包含所有序列的长度
> 包含不一定是连续包含，可以不是子串

状压DP
依次构造每一位
把每个字符串走到的位置标记一下，压成6进制数
然后每个状态拓展一个字符串
然后同时拓展其他所有下一位与其相同的串
然后把状态丢到队列里转移，当每个串都走到结尾时输出答案
可以保证答案最多不超过40
时间复杂度 $O(ans * len^N)$

I. 小冰和小娜 （BFS暴力）

> 给定一个网格图，有一辆独轮车
> 每走一个换一个颜色，一共五个颜色循环
> 然后可以向前，左转，右转，花费时间相同
> 刚开始面朝北边，绿色朝下
> 问最终到达终点，且绿色向下的最短时间是多少

没啥好说的，把位置，方向，时间，朝下颜色全记下来
然后全丢到队列里，直接 bfs暴力跑即可
适当调整上下左右的顺序，可以减少代码量

J. TooEasy Or TooDifficult （Manacher+xor-Trie）

> 题意太长就不说了

板子题，依题意描述分为三个步骤
1. 求每个位置为中心的回文串，以及最大回文串
2. 求每个回文串的长度的异或前缀和
3. 求两个异或前缀和异或的最大值 （xor-Trie）

然后用快速幂算出 JD，再和 FJD比大小即可

注意一下 Trie上要先插入一个 0

K. 奶牛合影（最小表示法）

> 给定一个循环串，问从哪个位置剖分能使得字典序最小

最小表示法裸题，后缀数组裸题
然而后缀数组我还不太会构造
所以转而学习了一下最小表示法

朴素算法：
将原数组复制一遍
枚举两个串的开头p_1, p_2，依次比较两个串的每一位
每当 $S[p_1 + k] \neq S[p_2 + k]$ 时，字典序较大的头指针向后移一位
时间复杂度 $O(N^2)$

最小表示法：
与朴素算法大致相同
就是在发现 $S[p_1 + k] \neq S[p_2 + k]$时，字典序较大的头指针向后移动 k位

证明如下：

1. 不妨设 $S[p_1 + k] > S[p_2 + k]$，则 $\forall i \in [p - 1, k)$ 都不可能成为最小串的头
   因为如果其为最小串的头，那么可以在第二个串中对应找一个位置

使得 $S_{i_1,i_1+k} > S_{i_2,i_2+k}$
所以 $\forall i \in [p-1, k)$ 都不可能成为最小串的头

2. 同理 $S[p_1+k] < S[p_2+k]$， $\forall i \in [p-2, k)$ 都不可能成为最小串的头
   所以在发生失配时，头指针直接向后移动$k+1$位

3. $S[p_1+k] = S[p_2+k]$，则 k就自增，不断比较，直到失配
   或者当$k == N$时，此时就得到了最小串
   $p_1 \neq p_2$ 不妨设 $p_2 > p_1$
   由 1和 2可知，$p_1 - > p_2$的字符都不可能是最小串的开头
   由于 $S_1 == S_2$，所以相同结论可以映射到 S_2上
   所以就可得出 p_1即最小串的开头

最后 p_1，p_2中较小的那个就是最小串的头指针
如果 p_1，p_2都小于N，最小串不唯一，任意一个都行
如果最小串唯一，那么有一个势必滑动出 N了，所以取最小的
显然尾指针最多移动 4*N次，所以算法时间复杂度 $O(N)$

M. 奶牛硬盘（模拟）

> 求由于硬盘容量进制不同造成的容量损失率

按题意模拟就好，其实这题跟前面的数字无关，只需要关注单位

N. 奶牛情书（AC自动机）

> 求给定长度的文本串，使得每个模式串至少出现过一次
> 求这样的模式串的个数

AC自动机禁止模式串的裸题
这题求的是一个串至少出现过一次

只要求一次都没出现过的方案总数
再拿所有方案总数相减即可得到答案

然后就拿所有模式串构造 Trie树，然后在上面跑 AC自动机
跑到单词结尾的时候不转移，这样就能求出答案