A.橘猫的ACM DREAM

题目大意：有n支队伍参加比赛，每支队伍有解题数量n和罚时t，给出最终正确的排名。

本来这是一道送个大家温暖的签到题，但是由于数据的问题，且我们对难度考虑不周，让这道题对于新生来说太难了。给所有超时和WA的同学在这里道个歉，没有给你们一个好的比赛体验。下一场比赛我们会根据这一场的情况调整题目的难度，让每一个人都能有一个好的比赛体验，希望大家继续支持软件学院ACM集训队。

这道题我们设计的是只需要按照解题数量递减，如果解题数量相同则罚时递增的排序方法排一下序就过了。排序的方法用algorithm库里的sort函数，或者自己写一个$O(n\log n)$的排序算法都可以，$O(n^2)$是过不了的。

B.橘猫与矩阵

这是一道构造题。

由于可以交换任意次，题意可以转换为：

> 构造一个 $n*n$ 的矩阵，这个矩阵中的元素大小在 1 到 n^2 之间，并且矩阵中的元素两两不会重复。且这个矩阵的每一行的元素和相等。

而如果有解，矩阵的每一行的元素和应为 $\frac{n(1+n^2)}{2}$。

通过观察，我们可以知道，$\frac{n(n^2+1)}{2}$ 是 首项为 1 ，末项为 n^2 ，有 n 项 的等差数列，其公差为 $n+1$。

当 $n=4$ 时这四个元素如下图所示

$$\begin{bmatrix} \underline{1} & 2 & 3 & 4 \\ 5 & \underline{6} & 7 & 8 \\ 9 & 10 & \underline{11} & 12 \\ 13 & 14 & 15 & \underline{16} \end{bmatrix}$$

这四个元素连成了一条斜线。

当这条斜线右移一个单位时，有 3 个元素增加了 1 ，有 1 个元素减少了3 ，其总和不变。

也就是说，当 $n=4$ 时的答案为

$$\begin{bmatrix} 1 & 6 & 11 & 16 \\ 2 & 7 & 12 & 13 \\ 3 & 8 & 9 & 14 \\ 4 & 5 & 10 & 15 \end{bmatrix} \tag{1}$$

同样的，对于 n 取任意值的情况也成立。

记$a[i][j]$为答案矩阵中第i行第j列的元素，那么

$$a[i][j] = (j-1)*n+i+j-1-(i+j-1>n?n:0); \tag{2}$$

时间复杂度 $O(n^2)$

C.橘猫与电梯

这是一道动态规划题。

用 $dp[i][0]$ 表示橘猫到了第 i 层，且橘猫通过爬楼梯从第 $i-1$ 层到第 i 层。

用 $dp[i][1]$ 表示橘猫到了第 i 层，且橘猫通过坐电梯从第 $i-1$ 层到第 i 层。

那么，转移方程为

$$\begin{cases} dp[i+1][0]=min(dp[i][0],dp[i][1])+b[i]; \\ dp[i+1][1]=min(dp[i][0]+c,dp[i][1])+a[i]; \end{cases} \tag{3}$$

答案为 $min(dp[n][0],dp[n][1])$。

时间复杂度 $O(n)$。

D. 橘猫与喵喵字符串

题目大意：

给定一个字符串S,问$[l,r]$范围构成的子串有多少个不同的，长度为3的子序列，恰好能构成CAT这个单词。

考虑一个简单的情况：在整个字符串S里有多少个这样的子序列。我们只需要顺序维护答案，记录C,CA,CAT子序列的总数。每碰到一个C，C的总数加1，每碰到一个A，CA的总数加C,因为每一个前面的C都可以和这个A组成子序列，同理每碰到一个T，CAT的总数就要加CA个。最后的答案就是CAT。

当情况扩展到子串时，不妨考虑相似的维护方法，使用前缀和的想法，维护$C[i]$,$A[i]$,$T[i]$,$CA[i]$,$AT[i]$,$CAT[i]$这几个数组，记录从开始到第i位字符的对应子序列总数，那么$[l,r]$范围的CAT子序列个数就是：

$$\begin{aligned} CAT= & CAT[r]-CAT[l-1]- \\ & C[l-1]*(AT[r]-AT[l-1]-A[l-1](T[r]-T[l-1]))- \\ & CA[l-1]*(T[r]-T[l-1]) \end{aligned} \tag{4--6}$$

这样我们只需要预处理一下前缀和数组，就能$O(1)$回答询问，时间复杂度为$O(len+m)$

这种做法在时间上是最优的，但容易写出锅来，另一个比较好想的方法是使用线段树，维护一段区间的C,A,T,CA,AT,CAT子序列的个数，用l代表左边的区间，r代表右边的区间，区间合并时有：

$$CAT=l.CAT+r.CAT+l.CA*r.T+l.C*r.AT \tag{7}$$
$$CA=l.CA+r.CA+l.C*r.A \tag{8}$$
$$AT=l.AT+r.AT+l.A*r.T \tag{9}$$
$$C=l.C+r.C \tag{10}$$
$$A=l.A+r.A \tag{11}$$
$$T=l.T+r.T \tag{12}$$

处理每次询问的复杂度为$O(\log n)$，总的时间复杂度为$O(len+m\log len)$

Problem E

根据题目，这显然是一个最短路问题，但我们发现只要每个L_i足够小，R_i足够大，那么就可以成为一张完全图。显然直接使用Dijkstra或者SPFA都会得到TLE。但是，我们可以发现，所有边权都是正的。我们从1号点开始看，从1号点出发能够到达的所有点j，对于它们的距离dis_j一定已经最优了。此时，在这一些点中，对于其中某个点u，若它的c_u是最小的，那么由它出发新到达的点v，其dis_v也一定是最优的（因为从1出发一次到达的点中，不会存在点k也能到点v，并且$dis_v > c_1 + c_k$）。以此类推，我们发现，如果令$d_i = dis_i + c_i$，其中dis_i表示1号点到当前点的距离，c_i表示从i点出去耗费的时间，那么每个点都只会被更新一次。利用以上结论，我们可以对每个更新过的点采用简单的数据结构维护，比如像std一样使用并查集，每个点被更新后都连向右边的点（因为从左到右更新），或者使用一个$std :: set$来维护没有被更新过的点，那么可以使用Dijkstra进行计算。此时，由于每个点只会被更新一次，复杂的为$O(nlogn)$