CF1107题解

A

贪心的想法，只分两段，第一段只有一位数，第二段是剩下的所有数。因此n>2时必有解，n=2时要比较两位数的大小，n<2直接无解。

B

说白了就是模9为x的第k个，直接9\*(k-1)+x即可。

C

贪心，如果有一段连续的字母数量超过k个，就挑这一段里最大的k个即可。这个可以用优先队列实现，而其它字母都要即可。复杂度O(n log(n))。

D

首先考虑一行内的压缩幅度。一行里面显然是由很多个0段和1段构成的。比方说0001110001110010由4个0段和3个1段构成。对于一行来说，最大的压缩幅度是所有01段的长度的最大公约数。

再来看行与行之间的压缩幅度。只要记录一下相等的行的数量，取最大公约数就可以了。最后只要取行内的最大压缩幅度，和行间最大压缩幅度的最大公约数就是答案了。

E

区间 DP 。

很明显是区间 DP，所以一开始有一个很不成熟的想法是设 f[i][j] 表示消除区间 [i,j] 的答案，但是有可能存在消掉 j 右边一段的决策。

这个时候类似于 UVa10559，设 f[i][j][k] 表示消除区间 [i,j] 及 j 右边恰好 k 个与位置 j 相同字符的答案，那么很明显有两种转移：

1、直接消掉 j 及右边 k 个。

2、枚举一个与 j 同字符的位置 l，通过消掉 [l+1,j+1] 将这两个区间合并起来。

所以说就得到有 f[i][j][k]=max(f[i][j−1][0]+a[k+1],f[l+1][j−1][0]+f[i][l][k+1])，直接转移是 O(n4) 的，可以通过。

F

我们固定住买车的时间，那么这 n 种贷款可能有以下 3 种可能性：

1、不使用

2、使用，在买车很久很久以前就到期了

3、使用，并且在买车的时候已经用了 j 天。

对于第三种情况，根据贪心的思想，j 肯定要越小越好，所以假设我们已经有了 t−1 个正在使用的贷款，准备付第 t+1 个的时候很明显要将它排在买车前第 t−1 天开始使用。

然后就可以 01 背包了，dp[i][j] 表示在前 i 个贷款中选了 j 个正在使用的方案数，那么还是从以下三种情况转移来：

1、不使用，dp[i−1][j]

2、使用，但是到期了，dp[i−1][j]+a[i]−b[i]×k[i]。

3、使用，但是没到期。根据上面的推论，将它排在买车前的第 j−1 天开始付，因此就是 dp[i−1][j−1]+a[i]−b[i]×min(k[i],j−1)。

但是我们背包的时候，没有考虑具体选择贷款的顺序，需要定一个顺序。在贷款顺序的选择上，需要按b排序，这个可以用贪心来证明，否则样例都过不了。

最终答案：max{dp(n,i)}。

G

记sum[x]为c[i]的前缀和，那堆平方和为f[l][r]，设选[l,r]时答案为ans[l][r]，则我们推式子：ans[l][r]=(r-l+1)\*a-sum[r]+sum[l-1]-f[l][r]，

ans[l][r]+l\*a-a-sum[l-1]=r\*a-sum[r]-f[l][r]

ans[l][r]+l\*a-a-sum[l-1]=F[r]-max{(d[i+1]-d[i])^2,i=l,l+1,...,r-1}

发现左边只和l有关，右边只有一个max较为麻烦。考虑max在左端点一定时单调上升，所以左端点每次往左移一格，只会对一小部分r造成影响，将他们的max弄成一样的。既然一样了，那当然可以记录一个max[Fr]，然后把它们并在一起。

因此可以用一个单调栈记录r，每次l往左移一位，就把一堆r缩在一起。

栈里每个元素也要存自己下面的右侧最大值，用来统计答案。

时间复杂度O(n)，因为左边也可以考虑用单调栈来搞。

如果用线段树，就是O(n logn)。