Solution

栈 (stack)

题目来源

UOJ#174 新年的破栈 弱化版（原题不是排列，只保证任意两项不同，然而没什么卵用...）

10分算法

显然操作次数是 $2n$ 的，我们暴力枚举每次操作是什么，判断合法性并统计答案即可。

时间复杂度 $O(3^{2n})$

30分算法

上面那个做法显然太暴力了，我们换个好一点的：

由于要求字典序最小，因此考虑每个数的可能取值，只能是三种情况：剩余序列中、栈顶、栈底。

因此比较这三种情况哪种最小，如果是在剩余序列中的数则一直压栈直到将这个数压到栈顶然后弹栈，否则直接在相应位置弹栈。

剩余序列中的最小值查询使用暴力，时间复杂度 $O(n^2)$ 。

100分算法

作为省选级别选手不可能不会优化掉其中找剩余序列中数的 n 吧...

使用后缀最小值等类似方法维护一下即可。

时间复杂度 $O(n)$ 。

串 (string)

以下使用 len 代指 $Max\{strlen(B_i)\}$ 。

题目来源

BZOJ2121 字符串游戏 加强版

20分算法

直接暴力模拟即可，匹配时不需要KMP等。

时间复杂度 $O((n\cdot m\cdot len)^k)$ ，可以通过前两类数据点。

45分算法

这一类数据点保证 $k=n$ ，表明可以无限制使用魔法，是BZOJ2121的原题。

我们考虑：每次删除连续的一段，对应到原串上即：删除 $[l,r]$ 中所有未被删除的字符。其中 l,r 都未被删除。

这样就相当于选择若干区间来删除。

注意到选择的任意两个区间要么包含要么不相交（相离），对于相邻的相离的也可以看作是包含（右区间左端点看作是左区间左端点，即一个空位置），因此只有包含关系。

那么如下图：

先选择 $[b,c]$ 的串 S ，再选择 $[a,d]$ 的串 T ，可以看作是处理出 $[a,b)$ 能够匹配到 T 的中间位置，$[b,c]$ 能够匹配到 S 的结束位置（即删除掉），进而推知 $[a,c]$ 能够匹配到 T 的中间位置，再向右匹配得知 $[a,d]$ 能够匹配到 T 的结束位置。

考虑区间dp。设 $f[l][r]$ 表示 $[l,r]$ 是否可以全部删掉，再设 $g[l][r][i][j]$ 表示 $[l,r]$ 是否能够删成第 i 个字符串的前 j 个字符。

那么考虑区间 $[l,r]$ ，如果进行匹配的话转移为 $g[l][r][i][j]=g[l][r-1][i][j-1]$ ，前提条件 $str[r]==w[i][j]$ ，即区间右端点和第 i 个串的第 j 个字符相同。

如果不进行匹配的话，r 一定在某个 $[k,r]$ 中被消掉，因此枚举 $k\in[l,r]$ ，转移为 $g[l][r][i][j]=g[l][k-1][i][j]\&\&f[k][r]$ 。

根据 f 的定义有转移 $f[l][r]=g[l][r][i][len[i]]$ 。

这样我们就能够推出 f 和 g 。

再考虑答案：设 $h[i]$ 表示前 i 个字符的答案，那么 $h[i]=h[i-1]+1$ ；如果某个 j 满足 $f[j][i]=1$ ，即 $[j,i]$ 能删掉，则还有 $h[i]=h[j-1]$ 。

最终答案就是 $h[n]$ 。

时间复杂度 $O(n^3 \cdot m \cdot len)$，综合以上算法可以得到45分。

70分算法

这一部分测试点的 n 和 k 相比极限数据小一些，可以借鉴45分算法。设 $f[t][l][r]$ 表示能否使用 t 次删除删掉 $[l,r]$，$g[t][l][r][i][j]$ 表示能否使用 t 次删除将 $[l,r]$ 删成第 i 个字符串的前 j 个字符。转移时对第一维进行背包合并。

时间复杂度 $O(k^2 \cdot n^3 \cdot m \cdot len)$，综合以上算法可以得到70分。

100分算法

相信对于省选级别的选手来说，如果想到了45分算法，相信也一定能够想到100分算法。

用bool设状态实在是太浪费了，我们考虑使用int设状态。

设 $f[l][r]$ 表示将 $[l,r]$ 删掉需要的最小次数，$g[l][r][i][j]$ 表示将 $[l,r]$ 删成第 i 个字符串的前 j 个字符需要的最小次数。

如果不能够做到，则dp值视为 $+\infty$ 。

同理转移即可。

时间复杂度 $O(n^3 \cdot m \cdot len)$，可以通过全部测试点。

向量 (vector)

题目来源

BZOJ3533[Sdoi2014]向量集 ~~弱化~~ 版

讲道理这也不能算弱化...原题 x 和 y 有负数，只是多维护一个下凸壳而已...这道题为了方便只需要维护一个

10分算法

暴力不解释。

时间复杂度 $O(n^2)$ 。

60分算法

除了暴力以外，我们要分析一下点积的本质。

本质是什么呢：给出 x_0 和 y_0 ，求满足条件的 x,y 使得 x_0x+y_0y 最大。

我们不妨设这个值是 c ，则要最大化 c 。

给这个式子 $c=x_0x+y_0y$ 变形，得：$y=-\frac{x_0}{y_0}x+\frac{c}{y_0}$ 。想要 c 最大，即要 $\frac{c}{y_0}$ 最大。

相当于是一条斜率为 $-\frac{x_0}{y_0}$ 的直线，要使纵截距最大，答案一定在上凸壳上，且所选点与答案形成单峰函数。

因此维护上凸壳，在凸壳上 三分/二分 即可。

然而我们要求的是一段区间内的答案，不能直接维护上凸壳。

考虑分块，对于整块我们维护上凸壳，对于其余部分暴力处理。当块大小为 $\sqrt{n\log n}$ 时取得最优时间复杂度 $O(n\sqrt{n\log n})$ ，可以得到60分。

75分算法

既然想到60分算法也就不难想到75分算法。

由于询问在修改之后，因此可以先拿出所有点，建一棵线段树，每个节点维护这个区间所有点形成的上凸壳。

对于询问，在线段树的对应区间上修改即可。

时间复杂度 $O(n\log^2 n)$ ，综合以上算法可以得到75分。

100分算法

既然想到75分算法也就不难想到100分算法。

由于强制在线，因此不能提前求凸壳。但我们可以发现：如果线段树的一个节点没有被全部加入（即没加全），那么这个点一定不会出现在询问区间中。

因此我们在线段树上插入，当且仅当 $now==r$ 时，对该节点建立上凸壳即可。

时间复杂度 $O(n\log^2 n)$ 。