# IOI2020 国家集训队第一阶段作业第一部分解题报告

北京大学附属中学　李白天

2019 年 11 月 15 日

## 1　Codeforces 559E Gerald and Path

### 1.1　题目大意

有 $n$ 组有序数对 $(a_i, l_i)$，对于每个 $i$ 有两种选择，将数轴上的 $[a_i - l_i, a_i]$ 这条线段染色或者将 $[a_i, a_i + l_i]$ 这条线段染色。求最大化的最终被染色总线段长度。

### 1.2　数据范围

输入均为整数，$1 \leq n \leq 100, 0 \leq a_i \leq 10^8, 1 \leq l_i \leq 10^8$，保证 $a_i$ 互不相同。

### 1.3　解题过程

本题官方给出的是一个 $\Theta(n^4)$ 的做法，但实际上本题可以在 $\Theta(n^2)$ 的时间内解决。

考虑在第一步枚举最优解中被染色的最右端点来自哪个 $i$，由于我们已经枚举了最右端点，那么在 $(a_i, a_i + l_i)$ 区间内的 $a_j$ 都不必要选取向右延伸，因为在这种枚举方式下已经假设这段区间的右端不可改进，故要求它们均向左延伸。进一步地，如果在这段区间内的 $a_j - l_j$ 之最小值 $< a_i$，则说明区间的左端可以得到扩展，扩展后新的被包含的 $a_j$ 同理要被如此考虑，直至区间无法进一步扩展。

于是在规划剩余的 $i$ 时，还需考虑到与之前原有线段可能有交。我们考虑动态规划。$f(x)$ 表示假设之前已经规划的部分左端点为 $x$，接下来规划所有 $a_i < x$

的 $i$ 对应的方案，总线段与 $(-\infty, x]$ 的交长度的最大值。$f(+\infty)$ 即为答案。关键点只有所有的 $a_i, a_i \pm l_i$，因此总共只有 $\Theta(n)$ 种，可以将状态离散化。

对于之前的扩展部分，我们首先可以预处理每个区间内 $a_i - l_i$ 的最小值，通过区间最小值递推出每个区间最终向左扩展出的区间。这一部分的时间复杂度为 $\Theta(n^2)$。

动态规划总共有 $\Theta(n)$ 个状态，每个状态有 $\Theta(n)$ 种转移方式，通过上述预处理可以将转移结果 $\Theta(1)$ 算出。

综上所述，本题可以在 $\Theta(n^2)$ 时间复杂度内解决。

# 2 AGC 037E Reversing and Concatenating

## 2.1 题目大意

给一个长为 $n$ 的字符串 $S$，进行 $k$ 次操作，每次首先将 $S$ 的翻转 $T$ 与 $S$ 拼接成一个长为 $2n$ 的字符串 $ST$，然后选取一个长为 $n$ 的子串作为新的 $S$，要求最小化 $k$ 次操作后的字符串，要求字典序最小。

## 2.2 数据范围

$1 \leq n \leq 5000, 1 \leq k \leq 10^9$。

## 2.3 解题过程

首先容易注意到 $k$ 只要超过 $\log n$ 的量级就没有意义，因为设字符串的最小字符为 $c$，可能生成的字符串不难得到一个下界，即一个长为 $n$ 的，每个字符都是 $c$ 的字符串。这可以在第一次操作的时候使 $c$ 作为末尾出现，之后每次操作都会由 $ST$ 中间部分使得 $c$ 的连续段倍长。因此设这一连续段长度为 $L$，只需 $L2^{k-1} \geq n$ 即可让最后整段均为 $c$。

接下来考虑 $L2^{k-1} < n$ 的情况，这启发我们同样从最小字符的角度考虑。为了使得字典序最小，我们首先需要保证得到串开头的 $c$ 前缀尽量长，考虑最初的 $ST$ 中最长的 $c$ 连续段为 $L$，我们之后每次操作必然可以使长度倍增，而剩下的其余 $c$ 段（设长度 $l$）由于没有在中间的位置，故不可能倍增，只有 $l \leq L < 2L$，因此在接下来选定了倍长哪一段后，没有长度大于等于它的，接下来的操作必定是将其一直倍长。考虑最终状态，假设原字符串中这一连续段所占据位置为 $S_j = S_{j+1} = \cdots = S_{j+L-1} = c$，则终态 $S'$ 在最后截取长度为 $n$ 的段前，是一个长度为 $2n$ 的回文串，原本连续段起点之前的前缀部分 $\ldots, S_{j-2}, S_{j-1}$ 的一个后缀得以保留，因此最后的 $S'$ 是形如 $L2^{k-1}$ 个 $c$ 再加上 $S_{j-1}, S_{j-2}, \ldots$。

为了确认最初的决策点，我们需要将每个长度达到最大值 $L$ 的 $c$ 的连续段，其前面部分的前缀 $pre(j-L)$ 的字符串翻转的结果中找到字典序最小的，截取前 $n - L2^{k-1}$ 位，与连续段拼接后就是答案。

如果我们暴力进行字符串比较，则可以做到 $\Theta(n^2)$，可以通过本题。而找到这些前缀的翻转中最小的那一个，还可以通过各种后缀排序的算法解决，因此本

题可以做到 $\Theta(n \log n)$ 或者 $\Theta(n)$。

# 3 AGC 020F Arcs on a Circle

## 3.1 题目大意

给一个周长为 $C$ 的圆以及有 $n$ 条圆弧分别长为 $l_i$，要求计算这 $n$ 条圆弧均匀随机的方式放置在圆上时，整个圆每个点都被覆盖的概率。

## 3.2 数据范围

$2 \leq n \leq 6, 1 \leq l_i < C \leq 50$，输入均为整数。

## 3.3 解题过程

首先考虑到旋转对称性，可以将某一个弧先固定假设为不动的，这样我们就可以讨论其他弧的坐标。假设选择的那条弧的逆时针端点是 0，其他弧的左端点坐标 $x_i$ 定义为该点逆时针走到 0 的距离。此时可以发现选取最长的那条弧是最合适的，因为我们可以将判据变为：这些弧的对应的区间 $[x_i, x_i + l_i]$ 的并包含了 $[0, C]$，这是因为当我们选取的弧是最长的时候，没有弧可以包含它，因此超出一圈的部分必然没有意义。

接下来考虑如何计算转化后的模型后的概率，我们考虑弧的坐标被表示为 $x_i = n_i + \alpha_i (n_i \in \mathbb{Z}, \alpha_i \in [0, 1))$，即整数部分和小数部分。我们不必考虑有 $\alpha_i = \alpha_j (i \neq j)$ 的情况，因为这种情况下发生概率已经是 0，不需要纳入计算。我们先考虑一个暴力：先枚举每个点的坐标整数位置 $0 \leq x_i < C$，然后考虑任何一个排列 $p$ 表示一个不等关系 $\alpha_{p_i} < \alpha_{p_{i+1}} (1 \leq i \leq n-2)$。注意到此时所有区间的相对大小关系也必然确定。我们之前的要求覆盖的条件可以转化为若干个不等式，即每个弧的坐标要被之前某个弧覆盖，我们假设放在原点的弧其坐标为 $x_0$，长度为 $l_i$，那么就是对于每个 $1 \leq i \leq n-1$，有

$$\begin{aligned} x_i &\leq \max_{0 \leq j < i} x_j + l_j \\ &= \max_{0 \leq j < i} (n_j + l_j) + \alpha_j \end{aligned}$$

由于 $n_j + l_j$ 部分我们已知，所以这一部分的数值大小已经被确定，由于这

一部分是整数而 $a_j$ 部分是小于 1 的，我们可以在整数值取到最大的部分中通过比较 $\alpha_j$ 来得到最大的数对应的 $j$。我们接下来可以得到：

$$x_i \le x_j + l_j$$
$$n_i + \alpha_i \le n_j + \alpha_j + l_j$$
$$\alpha_i - \alpha_j \le n_j + l_j - n_i$$

由于左边必然是 $(-1, 1)$ 内的数，右边必然是整数，不难得到几类情况：

- 若 $n_j + l_j - n_i \ge 1$，这个不等式总是成立。
- 若 $n_j + l_j - n_i \le -1$，这个不等式不可能成立。
- 若 $n_j + l_j - n_i = 0$，考虑到已经假设了 $\alpha_i \ne \alpha_j$，我们得到了 $\alpha_i < \alpha_j$，这要么被蕴含于我们枚举的排列中，要么与排列得到的关系违背推出无解。

因此，我们可以对于每组 $\{n_i\}$ 和 $\{p_i\}$，要么满足该小数部分关系的均使得圆被覆盖，要么均不能，这一部分被算入答案的贡献应该为 $\frac{1}{C^{n-1}}p$，其中 $p$ 是 $n-1$ 个小数满足排列 $p_i$ 的大小关系之概率。由于任何一个排列是对称的，而所有排列对应的发生概率之和应当为 1，可得 $p = \frac{1}{(n-1)!}$。因此假设我们所有可行的 $(\{n_i\}, \{p_i\})$ 总共有 $m$ 种，那么答案为 $\frac{m}{C^{n-1}(n-1)!}$。

接下来我们考虑动态规划计算出总共可行的正整数放置位置和排列方案数，记 $f(j, S, kn + p)$ 表示当前已经将整数部分 $\le j$ 的弧放置完毕，已经放置的集合为 $S$，当前弧最远延伸到的位置是整数 $k$ 加上以前加的数中第 $p$ 大的位置。因此 $0 \le j \le C, S \subseteq \{1, 2, \ldots, n-1\}, j \le k \le C, 0 \le p \le |S|, 0 \le kn + p \le Cn$，这时我们当 $k \ge C$ 时只令 $p = 0$，因此时 $p$ 的具体值已经无意义。这里我们假设 $p = 0$ 的时候表示是原点位置的弧延伸到的位置。

这个动态规划的计算可以通过容斥进行优化，我们考虑通过每个 $j$ 递推至 $j + 1$，先考虑忽略每个弧的坐标要被之前某个弧覆盖这一条件。我们可以枚举每个数是否在这个位置加入来更新这一 DP 值，设 $g_i(S, kn + p)$ 是在已经考虑了编号 $< i$ 的弧之后的状态。可以得到对于 $i \notin S$，我们考虑新的数插在大小第 $q(1 \le q \le |S| + 1)$ 大的地方，有贡献：

- 对于 $q \le p$,

$$g_i(S, kn+p) \to g_{i+1}(S \cup \{i\}, \max(kn+p+1, (j+l_i)n+q))$$

- 对于 $q > p$,

$$g_i(S, kn+p) \to g_{i+1}(S \cup \{i\}, \max(kn+p, (j+l_i)n+q))$$

接下来考虑容斥掉不符合要求的部分，首先这在 $k=j$ 的时候发生，我们将每个不同的 $p=p_0$ 分开计算，此时我们考虑的是所有插入的数都比原本第 $p_0$ 个数大。这时能得到的方案来自递推时的转移时的限制:

- 对于 $p_0 < q \le p$,

$$h_{p_0,i}(S, kn+p) \to h_{p_0,i+1}(S \cup \{i\}, \max(kn+p+1, (j+l_i)n+q))$$

- 对于 $q > p$,

$$h_{p_0,i}(S, kn+p) \to h_{p_0,i+1}(S \cup \{i\}, \max(kn+p, (j+l_i)n+q))$$

最终我们得到的 $f(j+1, S, kn+p) = g_{n-1}(j+1, S, kn+p) - \sum_{p_0=0}^{p} h_{p_0,n-1}(j+1, S, kn+p)$。

综上，我们最后的答案就是

$$\frac{f(C, \{1, 2, \ldots, n-1\}, Cn)}{C^{n-1}(n-1)!}$$

上述 DP 式子计算一个 $g/h$ 状态的复杂度是 $O(n)$, 一组 $h_{p_0,i}/g_i$ 有 $O(2^n nC)$ 个状态, 整个算法中需要计算的 $h$ 部分是瓶颈, 需要计算 $O(nC)$ 组, 综上所述, 我们得到了一个 $\Theta(2^n n^3 C^2)$ 复杂度的算法，相较于官方给出的 $\Theta((n-1)!(Cn)^2 2^n)$ 更加优秀。