Tiling the Plane – Solution

试题来源

ACM/ICPC World Finals 2005 G

简要题意

给定一个每个角都是直角的多边形，问是否能通过它不断复制、平移（不能旋转或翻转）来不重不漏的覆盖无限的二维平面。

考察算法

枚举、字符串匹配

题解

注意到题面上提供的有关“棋盘覆盖”和“蜂巢覆盖”的信息，我们不难把此题转化成一道字符串匹配题。

首先，我们将原描述串展开（将一个字母重复多次的表达方式展开为字符串序列，如“S 2 E 2 N 2 W 2”展开得到“SSEENNWW”），设展开串的长度为n，在之后的说明中，我们都将使用展开串来描述多边形或其边界上的一段。

假设我们现在检测该字符串是否满足“棋盘覆盖”，即检测是否能依次找到分割点 A,B,C,D，使得 A->B 与 D->C 匹配，B->C 与 A->D 匹配。注意到需要匹配的两个串方向相反，因此我们需要将其中一串倒置并将其中的每个方向替换为相反的方向（N 换为 S，S 换为 N，E 换为 W，W 换为 E）再进行匹配。假如我们现在检测“SENE”与“WSWN”是否匹配，我们先把“WSWN”反向得到“NWSW”，再将每个方向反向得到“SENE”，我们发现这与另一匹配串“SENE”是相等的，则说明“SENE”与“WSWN”是匹配的。接下来就是枚举并检查匹配。暴力枚举 A,B,C,D 点并暴力匹配的时间复杂度达到了O(n^5)，这是我们所不能接受的。但稍作考虑我们即可发现，A->C 和 C->A 一定是长度相等的两个串，并且 A->B 与 D->C 长度相等，B->C 与 A->D 长度相等。这样我们在枚举时可以先枚举将序列分割成长度相等的两段的分割方案，这样便确定了 A、C 的位置，再枚举 A->B（D->C）的长度l，就能确定 B、D 的位置。这样就把枚举量减小到O(n^2)，再乘上匹配所需的O(n)，总时间复杂度降为了O(n^3)。

$$\underbrace{S\quad S}_{l}\quad E\quad E\ \Big|\ \underbrace{N\quad N}_{l}\quad W\quad W$$

图：枚举分割方案后枚举 A->B 串长度

图中两个红色部分互相匹配，两个蓝色部分互相匹配，该多边形可以铺满平面

我们仍然可以继续优化这个算法。设原串为 A，我们先处理出原串整个串倒置并将每个方向反向的字符串 B，这样检查原串中两段是否能匹配的问题就转化为了检查 A 的某个子串和 B 的某个子串是否相等的问题了。我们预处理$f(i,j)$表示从 A 的第 i 位开始，B 的第 j 位开始的最长匹配长度，即$\mathrm{A}[i \dots n]$与$\mathrm{B}[j \dots n]$的最长公共前缀，它可以利用以下递推式在$\mathrm{O}(n^2)$时间内求出：

$$f(i,j)=\begin{cases} f(i+1,j+1)+1, & \mathrm{A}[i]=B[j] \\ 0, & \mathrm{A}[i]\neq B[j] \end{cases}$$

这样我们就可以在$\mathrm{O}(1)$时间内检查 A、B 的两个子串是否相等，检测该字符串是否满足"棋盘覆盖"便能够在$\mathrm{O}(n^2)$时间内完成。

接下来考虑"蜂巢覆盖"的情况。这种情况与"棋盘覆盖"的情况实质上没有太大区别。我们同样首先枚举将序列分割成等长两段的分割方案，即确定了 A、D 的位置、之后我们发现，它与棋盘覆盖唯一的不同只是有 A→B 与 E→D、B→C 与 F→E、C→D 与 A→F 三段需要匹配。我们先枚举 A→B 的长度$l1$，即确定了 B、E 的位置，再枚举 B→C 的长度$l2$，即确定了 C、F 的位置，最后用与"棋盘覆盖"同样的方法验证是否匹配即可。这样总枚举量是$\mathrm{O}(n^3)$，总时间复杂度也是$\mathrm{O}(n^3)$。由于大多数情况下，在枚举完$l1$之后已经发现不匹配，不会再进行对$l2$的枚举，因此这个算法的实际效果较为不错。

有没有与周长无关只与顶点数有关的算法呢？答案是肯定的。我们发现顶点的本质是展开序列中，字母发生变化的位置，也就是说一个顶点到下一个顶点之后的一系列字母都是相同的。注意到我们枚举$l1$实际上就是枚举 A 的对应点 E 的位置，因此我们可以只枚举 E 之后第一个顶点的位置，由于 E 到该顶点之间的字母都是相同的，我们只要看 A 之后有多少与之相同的字母即可确定 E 的位置。枚举$l2$时同理。设多边形的顶点数为m，这样对于每个分割方案的枚举量就减小为$\mathrm{O}(m^2)$。而枚举分割方案时，容易发现分割方案一定过某个顶点，这样分割方案数为$\mathrm{O}(m)$，总枚举量便降为$\mathrm{O}(m^3)$。如果我们预处理匹配结果，使检查匹配能在$\mathrm{O}(1)$时间内完成的话，整个算法的时间复杂度就为$\mathrm{O}(m^3)$。