自动机上的 DP

清华大学　任舍予

2024 年 8 月 9 日

- 对于一类 DP 问题：

- 对于一类 DP 问题：
  - 状态与转移构成了一个 DAG 的形式；

- 对于一类 DP 问题：
  - 状态与转移构成了一个 DAG 的形式；
  - 每次发生转移时，转移到的状态仅由当前的状态和加入的新参数决定。

- 对于一类 DP 问题：
  - 状态与转移构成了一个 DAG 的形式；
  - 每次发生转移时，转移到的状态仅由当前的状态和加入的新参数决定。
- 例如：最大独立集问题不关心先前的每个数的具体值，只关心先前的 DP 结果与当前的值。

- 对于一类 DP 问题：
  - 状态与转移构成了一个 DAG 的形式；
  - 每次发生转移时，转移到的状态仅由当前的状态和加入的新参数决定。
- 例如：最大独立集问题不关心先前的每个数的具体值，只关心先前的 DP 结果与当前的值。
- 这类问题均属于在一个**自动机**上的转移过程。

- 自动机：接受某些特定模式的字符串的模型。

- 自动机：接受某些特定模式的字符串的模型。
- 形式化定义：
  - 字符集 Σ：转移时输入的参数；
  - 状态集合 Q：DAG 上的结点，即可能转移到的状态；
  - 起始状态 $q_{start} \in Q$：初始状态；
  - 接受状态集合 $F \subseteq Q$：转移结束后接受的状态集合；
  - 转移函数 $\delta : Q \times \Sigma \to Q$：由当前状态与输入参数决定的转移到的状态。

- 自动机：接受某些特定模式的字符串的模型。
- 形式化定义：
  - 字符集 Σ：转移时输入的参数；
  - 状态集合 Q：DAG 上的结点，即可能转移到的状态；
  - 起始状态 $q_{start} \in Q$：初始状态；
  - 接受状态集合 $F \subseteq Q$：转移结束后接受的状态集合；
  - 转移函数 $\delta : Q \times \Sigma \to Q$：由当前状态与输入参数决定的转移到的状态。
- 常见的自动机：Trie，KMP 自动机，AC 自动机。

- 由于 DP 的转移过程对应 DAG 上的一条路径，因而产生了一类对路径计数的问题，即问有多少种可能的合法序列。

- 由于 DP 的转移过程对应 DAG 上的一条路径，因而产生了一类对路径计数的问题，即问有多少种可能的合法序列。
- 这类问题的一般处理手段：首先描述清楚给定序列下的做法，然后按照内层状态设计 DP，同时对状态数进行优化。

问题

给定序列 $[a_1, \ldots, a_m]$，求有多少个值域在 $[1, c]$ 范围内的序列 $[b_1, \ldots, b_n]$，满足序列 a 在序列 b 中出现恰好 r 次。

数据范围：$1 \le m \le n \le 2000$，$1 \le c \le 10^9$，$1 \le r \le 10$。

- 考虑 KMP 的匹配过程：每次从当前状态不断跳失配指针，直到找到第一个可以匹配的位置。

- 考虑 KMP 的匹配过程：每次从当前状态不断跳失配指针，直到找到第一个可以匹配的位置。
- 这一过程即为在自动机上的转移过程，找到的位置仅有当前匹配的长度与下一个需要匹配的字符相关。

- 考虑 KMP 的匹配过程：每次从当前状态不断跳失配指针，直到找到第一个可以匹配的位置。
- 这一过程即为在自动机上的转移过程，找到的位置仅有当前匹配的长度与下一个需要匹配的字符相关。
- 设计如下 DP：$f_{i,j,k}$ 表示 $[b_1, \ldots, b_i]$ 在 KMP 自动机上转移到结点 j，且目前已经出现了 k 次序列 a 的方案数。

- 考虑 KMP 的匹配过程：每次从当前状态不断跳失配指针，直到找到第一个可以匹配的位置。
- 这一过程即为在自动机上的转移过程，找到的位置仅有当前匹配的长度与下一个需要匹配的字符相关。
- 设计如下 DP：$f_{i,j,k}$ 表示 $[b_1, \ldots, b_i]$ 在 KMP 自动机上转移到结点 j，且目前已经出现了 k 次序列 a 的方案数。
- 转移时只需要枚举所有自动机上的出边转移，由 KMP 的性质可知转移边不超过 $2m$ 条。

- 考虑 KMP 的匹配过程：每次从当前状态不断跳失配指针，直到找到第一个可以匹配的位置。
- 这一过程即为在自动机上的转移过程，找到的位置仅有当前匹配的长度与下一个需要匹配的字符相关。
- 设计如下 DP：$f_{i,j,k}$ 表示 $[b_1, \ldots, b_i]$ 在 KMP 自动机上转移到结点 j，且目前已经出现了 k 次序列 a 的方案数。
- 转移时只需要枚举所有自动机上的出边转移，由 KMP 的性质可知转移边不超过 $2m$ 条。
- 时间复杂度 $O(nmr)$，空间复杂度 $O(mr)$。

问题 (洛谷 P4590)

令字符集 $\Sigma = \{\mathtt{N}, \mathtt{O}, \mathtt{I}\}$。给定长度为 k 的字符串 S，对于 $i = 0, 1, \ldots, k$，求有多少个长度为 n 的字符串 T 满足：

- 不含有子串 $\mathtt{NOI}$；
- 与 S 的最长公共子序列长度为 i。

数据范围：$1 \le n \le 1000$，$1 \le k \le 15$。

- 求解最长公共子序列的 DP：
  $f_{i,j} = \max(f_{i-1,j}, f_{i,j-1}, f_{i-1,j-1} + [T_i = S_j])$。

- 求解最长公共子序列的 DP：
  $f_{i,j} = \max(f_{i-1,j}, f_{i,j-1}, f_{i-1,j-1} + [T_i = S_j])$。
- 考虑 f_i 一整层的 DP 状态，只与 f_{i-1} 一整层与 T_i 有关。

- 求解最长公共子序列的 DP：
  $f_{i,j} = \max(f_{i-1,j}, f_{i,j-1}, f_{i-1,j-1} + [T_i = S_j])$。
- 考虑 f_i 一整层的 DP 状态，只与 f_{i-1} 一整层与 T_i 有关。
- 设计如下 DP：$g_{i,w_1,w_2,\ldots,w_k}$ 表示长度为 i，DP 值为 $f_{i,j} = w_j$ 的字符串数。

- 求解最长公共子序列的 DP：
  $f_{i,j} = \max(f_{i-1,j}, f_{i,j-1}, f_{i-1,j-1} + [T_i = S_j])$。
- 考虑 f_i 一整层的 DP 状态，只与 f_{i-1} 一整层与 T_i 有关。
- 设计如下 DP：$g_{i,w_1,w_2,\ldots,w_k}$ 表示长度为 i，DP 值为 $f_{i,j} = w_j$ 的字符串数。
- 最长公共子序列的性质保证了 $w_i - w_{i-1} \in \{0,1\}$，于是状态只有 $n2^k$ 个。

- 求解最长公共子序列的 DP：
  $f_{i,j} = \max(f_{i-1,j}, f_{i,j-1}, f_{i-1,j-1} + [T_i = S_j])$。
- 考虑 f_i 一整层的 DP 状态，只与 f_{i-1} 一整层与 T_i 有关。
- 设计如下 DP：$g_{i,w_1,w_2,\dots,w_k}$ 表示长度为 i，DP 值为 $f_{i,j} = w_j$ 的字符串数。
- 最长公共子序列的性质保证了 $w_i - w_{i-1} \in \{0, 1\}$，于是状态只有 $n2^k$ 个。
- 对于不含有指定串的限制，同时维护对应 KMP 自动机上的转移即可。

- 求解最长公共子序列的 DP：
  $f_{i,j} = \max(f_{i-1,j}, f_{i,j-1}, f_{i-1,j-1} + [T_i = S_j])$。
- 考虑 f_i 一整层的 DP 状态，只与 f_{i-1} 一整层与 T_i 有关。
- 设计如下 DP：$g_{i,w_1,w_2,\ldots,w_k}$ 表示长度为 i，DP 值为 $f_{i,j} = w_j$ 的字符串数。
- 最长公共子序列的性质保证了 $w_i - w_{i-1} \in \{0, 1\}$，于是状态只有 $n2^k$ 个。
- 对于不含有指定串的限制，同时维护对应 KMP 自动机上的转移即可。
- 时间复杂度 $O(nk2^k)$，空间复杂度 $O(2^k)$。

问题 (洛谷 P8352)

给定一棵 n 个点的树，每个点权值属于 $[1, k]$，求最大独立集分别为 $1, \ldots, nk$ 的方案数。

数据范围：$1 \leq n \leq 1000$，$1 \leq k \leq 5$。

- **求解最大独立集的 DP：**

$$
\begin{cases}
f_{u,0} = \sum_v \max(f_{v,0}, f_{v,1}) \\
f_{u,1} = \max\left(f_{u,0}, a_u + \sum_v f_{v,0}\right)
\end{cases}
.
$$

- 求解最大独立集的 DP：

$$\begin{cases} f_{u,0} = \sum_v \max(f_{v,0}, f_{v,1}) \\ f_{u,1} = \max\left(f_{u,0}, a_u + \sum_v f_{v,0}\right) \end{cases}.$$

- 设计 DP 如下：$g_{u,x,y}$ 表示子树 u 内 $f_{u,0} = x$，$f_{u,1} = y$ 的方案数。

- 求解最大独立集的 DP：

$$\begin{cases} f_{u,0} = \sum_v \max(f_{v,0}, f_{v,1}) \\ f_{u,1} = \max\left(f_{u,0}, a_u + \sum_v f_{v,0}\right) \end{cases}.$$

- 设计 DP 如下：$g_{u,x,y}$ 表示子树 u 内 $f_{u,0} = x$，$f_{u,1} = y$ 的方案数。
- 考虑优化状态：由最大独立集的性质，$f_{u,0} \le f_{u,1} \le f_{u,0} + k$，于是状态数是 $O(n^2k^2)$ 的。

- 求解最大独立集的 DP：

$$\begin{cases} f_{u,0} = \sum_v \max(f_{v,0}, f_{v,1}) \\ f_{u,1} = \max\left(f_{u,0}, a_u + \sum_v f_{v,0}\right) \end{cases} .$$

- 设计 DP 如下：$g_{u,x,y}$ 表示子树 u 内 $f_{u,0} = x$，$f_{u,1} = y$ 的方案数。
- 考虑优化状态：由最大独立集的性质，$f_{u,0} \le f_{u,1} \le f_{u,0} + k$，于是状态数是 $O(n^2k^2)$ 的。
- 转移是树上背包的形式，复杂度能够保证。

- 求解最大独立集的 DP：

$$\begin{cases} f_{u,0} = \sum_v \max(f_{v,0}, f_{v,1}) \\ f_{u,1} = \max\left(f_{u,0}, a_u + \sum_v f_{v,0}\right) \end{cases}.$$

- 设计 DP 如下：$g_{u,x,y}$ 表示子树 u 内 $f_{u,0} = x$，$f_{u,1} = y$ 的方案数。
- 考虑优化状态：由最大独立集的性质，$f_{u,0} \le f_{u,1} \le f_{u,0} + k$，于是状态数是 $O(n^2k^2)$ 的。
- 转移是树上背包的形式，复杂度能够保证。
- 时间复杂度 $O(n^2k^4)$，空间复杂度 $O(n^2k^2)$。