

动态 DP 入门

翁思源

信息学奥赛

wensiyuan40@gmail.com

2025 年 10 月 19 日

① 广义矩阵乘法

② 动态 DP

③ 例题

① 广义矩阵乘法

② 动态 DP

③ 例题

定义

假设有矩阵 $A: p \times m$ 和矩阵 $B: m \times q$, 定义广义矩阵乘法 $A \times B = C$ 的结果是一个 $p \times q$ 的矩阵 C 满足:

$$C_{i,j} = (A_{i,1} \otimes B_{1,j}) \oplus (A_{i,2} \otimes B_{2,j}) \oplus \dots \oplus (A_{i,p} \otimes B_{p,j})$$

其中 $\oplus$ 和 $\otimes$ 为两种运算。

定义

考察这种广义矩阵乘法是否满足结合律：

$$\begin{aligned}((AB)C)_{i,j} &= \bigoplus_{k=1}^p (AB)_{i,k} \otimes C_{k,j} \\&= \bigoplus_{k=1}^p \left(\bigoplus_{t=1}^q A_{i,t} \otimes B_{t,k} \right) \otimes C_{k,j} \\(A(BC))_{i,j} &= \bigoplus_{t=1}^q A_{i,t} \otimes (BC)_{t,j} \\&= \bigoplus_{t=1}^q A_{i,t} \otimes \left(\bigoplus_{k=1}^p B_{t,k} \otimes C_{k,j} \right)\end{aligned}$$

观察上式可知，当 $\oplus$ 运算满足交换律， $\otimes$ 运算满足结合律，且 $\otimes$ 对 $\oplus$ 存在分配律，那么广义矩阵乘法满足结合律。可以矩阵快速幂。

一些广义矩阵乘法的例子

ID	A	$\oplus$	0	$\otimes$	1
1	$\mathbb{R}$	min	$+\infty$	max	$-\infty$
2	$\mathbb{R}$	max	$-\infty$	min	$+\infty$
3	$\mathbb{R} \cup \{-\infty\}$	max	$-\infty$	+ 或 -	0
4	$\mathbb{R} \cup \{+\infty\}$	min	$+\infty$	+ 或 -	0
5	$\forall n \in \mathbb{N}^+, [0, 2^n) \cap \mathbb{N}$	OR	0	AND	$2^n - 1$
6	$\forall n \in \mathbb{N}^+, [0, 2^n) \cap \mathbb{N}$	AND	$2^n - 1$	OR	0
7	$\forall n \in \mathbb{N}^+, [0, 2^n) \cap \mathbb{N}$	XOR	0	AND	$2^n - 1$

提示

使用广义矩阵乘法时，初始矩阵和单位矩阵对应的 0 和 1 应当使用广义的 0 和 1。

① 广义矩阵乘法

② 动态 DP 最大子段和

③ 例题

什么是动态 DP ?

当我们能将 DP 状态转移表示为 矩阵形式，且这种矩阵乘法在某种“加法、乘法”下满足结合律时，我们就能用线段树或其他数据结构来维护这些矩阵，实现带修改的 DP。

典型应用：

- 序列 DP 的区间查询与单点修改；
- 树上带点权/边权修改的 DP；
- 本质上是一种“广义矩阵乘法”维护方法。

① 广义矩阵乘法

② 动态 DP
最大子段和

③ 例题

问题定义

题意

给定序列 $a_1 \dots a_n$ ，共 q 次操作：

- 1 x y: 查询区间 $[x, y]$ 的最大子段和；
- 2 x y: 将 a_x 修改为 y 。

数据范围： $n, q \leq 10^5$ 。

经典 DP 复盘

最大子段和的经典 DP:

$$f_i = \max(f_{i-1} + a_i, a_i)$$

其中 f_i 表示以 i 结尾的最大子段和。

再定义 g_i 为前 i 个位置的最大子段和:

$$g_i = \max(g_{i-1}, f_i)$$

两者结合就是:

$$g_i = \max(g_{i-1}, f_{i-1} + a_i, a_i)$$

引入矩阵思想

我们希望找到一个矩阵 M_i , 使得

$$M_i \times v_{i-1}^{\rightarrow} = v_i^{\rightarrow}$$

其中

$$v_i^{\rightarrow} = \begin{bmatrix} f_i \\ g_i \\ 0 \end{bmatrix}$$

于是, 我们要构造 M_i 使其同时实现

$$f_i = \max(f_{i-1} + a_i, a_i), \quad g_i = \max(g_{i-1}, f_{i-1} + a_i, a_i)$$

广义矩阵乘法

在“广义矩阵乘法”中：

$$\otimes = +, \quad \oplus = \max$$

并定义乘法规则：

$$C_{i,j} = \max_k (A_{i,k} \otimes B_{k,j})$$

因为 $\max$ 与 $+$ 都满足结合律和分配律，因此这一结构合法。

构造转移矩阵

满足上式的 M_i 可以构造为:

$$M_i = \begin{bmatrix} a_i & -\infty & a_i \\ a_i & 0 & a_i \\ -\infty & -\infty & 0 \end{bmatrix}$$

验证:

$$\begin{bmatrix} a_i & -\infty & a_i \\ a_i & 0 & a_i \\ -\infty & -\infty & 0 \end{bmatrix} \times \begin{bmatrix} f_{i-1} \\ g_{i-1} \\ 0 \end{bmatrix} = \begin{bmatrix} f_i \\ g_i \\ 0 \end{bmatrix}$$

例如:

$$f_i = \max(a_i + f_{i-1}, -\infty + g_{i-1}, a_i + 0)$$

区间乘积与线段树维护

整个序列可表示为矩阵积：

$$M = M_n \times M_{n-1} \times \cdots \times M_1$$

则：

$$M \times \begin{bmatrix} -\infty \\ -\infty \\ 0 \end{bmatrix} = \begin{bmatrix} f_n \\ g_n \\ 0 \end{bmatrix}$$

查询区间 $[l, r]$ 时，只需取该区间的矩阵乘积。若某个 a_i 被修改，只需更新对应叶子和其祖先节点。

因此使用线段树维护矩阵乘积即可，单次查询/修改复杂度： $O(3^3 \log n)$ 。

实现注意

- 当状态以列向量形式表示时，状态转移为

$$\mathbf{f}_{i+1} = M_i \mathbf{f}_i$$

因此多个区间合并时，应按物理顺序「右在前、左在后」进行矩阵乘法：

$$M_{[l,r]} = M_r \times M_{r-1} \times \cdots \times M_l$$

即线段树上合并时应写作：

$$\text{info}[p] = \text{info}[\text{right}] * \text{info}[\text{left}]$$

实现注意

- 若状态以行向量形式表示，则转移为

$$\mathbf{f}_{i+1} = \mathbf{f}_i M_i$$

此时合并顺序是「左在前、右在后」:

$$M_{[l,r]} = M_l \times M_{l+1} \times \cdots \times M_r$$

对应线段树写作:

$$\text{info}[p] = \text{info}[\text{left}] * \text{info}[\text{right}]$$

总结

表示方式	状态转移	线段树合并顺序
列向量	$\mathbf{f}' = M\mathbf{f}$	right $\times$ left
行向量	$\mathbf{f}' = \mathbf{f}M$	left $\times$ right

① 广义矩阵乘法

② 动态 DP

③ 例题

[ABC246Ex] 01? Queries

CF1609E William The Oblivious

① 广义矩阵乘法

② 动态 DP

③ 例题

[ABC246Ex] 01? Queries

CF1609E William The Oblivious

[ABC246Ex] 01? Queries

题意

给定一个长度为 n 的字符串 s ，每个字符为 0、1 或 ?。

有 q 次操作，每次将第 x 个字符修改为 c ，问修改后：

将所有 ? 替换为 0/1 后，不同的非空子序列有多少种？

数据范围： $n, q \leq 10^5$ 。

状态定义与转移

不考虑修改，设：

$f_{i,0}, f_{i,1}$ = 前 i 位中，以 0/1 结尾的方案数

初始化 $f_{0,0} = f_{0,1} = 0$ 。

根据 s_i 的取值可列出转移：

$$s_i = 1 : \quad f_{i,0} = f_{i-1,0}, \quad f_{i,1} = f_{i-1,0} + f_{i-1,1} + 1$$

$$s_i = 0 : \quad f_{i,0} = f_{i-1,0} + f_{i-1,1} + 1, \quad f_{i,1} = f_{i-1,0}$$

$$s_i = ? : \quad f_{i,0} = f_{i,1} = f_{i-1,0} + f_{i-1,1} + 1$$

统一转移方程

可以将三种情况统一表示为：

$$f_{i,0} = f_{i-1,0} + [s_i \neq 1] \cdot (f_{i-1,1} + 1)$$

$$f_{i,1} = f_{i-1,0} + [s_i \neq 0] \cdot (f_{i-1,1} + 1)$$

构造列向量：

$$v_{i-1}^{\rightarrow} = \begin{bmatrix} f_{i-1,0} \\ f_{i-1,1} \\ 1 \end{bmatrix}, \quad \vec{v}_i = M_i \times v_{i-1}^{\rightarrow}$$

我们需要构造出转移矩阵 M_i 。

构造转移矩阵

根据上式，有：

$$M_i = \begin{bmatrix} 1 & [s_i \neq 1] & [s_i \neq 1] \\ [s_i \neq 0] & 1 & [s_i \neq 0] \\ 0 & 0 & 1 \end{bmatrix}$$

验证：

$$\begin{bmatrix} 1 & [s_i \neq 1] & [s_i \neq 1] \\ [s_i \neq 0] & 1 & [s_i \neq 0] \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_{i-1,0} \\ f_{i-1,1} \\ 1 \end{bmatrix} = \begin{bmatrix} f_{i,0} \\ f_{i,1} \\ 1 \end{bmatrix}$$

整体结构与线段树维护

整个字符串的转移过程可表示为矩阵积：

$$M = M_n \times M_{n-1} \times \cdots \times M_1$$

最终状态：

$$M \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} f_{n,0} \\ f_{n,1} \\ 1 \end{bmatrix}$$

所有非空子序列的个数为 $f_{n,0} + f_{n,1}$ 。

当字符串有修改操作时，仅会影响对应位置的 M_i ，用线段树维护矩阵乘积，每次修改/查询均为 $O(3^3 \log n)$ 。

小结

- 本题的 DP 可线性推导，但由于存在多次修改，需引入 矩阵乘积思想。
- 每个字符对应一个 3×3 矩阵，表示“局部转移”。
- 矩阵乘积满足结合律 $\rightarrow$ 可用线段树维护区间积。
- 单点修改、区间查询复杂度： $O(\log n)$ 。

关键思想：将复杂的 DP 状态转移抽象为矩阵形式，从而实现“动态 DP”。

① 广义矩阵乘法

② 动态 DP

③ 例题

[ABC246Ex] 01? Queries

CF1609E William The Oblivious

CF1609E William The Oblivious

题意

给定一个长度为 n 的字符串 s ，由字符 a, b, c 组成。有 q 次修改操作，每次操作将第 x 个字符改为 $c \in \{a, b, c\}$ 。

每次修改后，求：最少替换多少个字符，使得字符串中不再出现子序列 abc 。

数据范围： $n, q \leq 10^5$ 。

DP 状态设计

对于单个字符串（不考虑修改），考虑按位置动态转移。

定义：

$$f_{i,0/1/2} = \begin{cases} 0: & \text{前 } i \text{ 个字符中无 } a \\ 1: & \text{已有 } a \text{ 但无 } ab \\ 2: & \text{已有 } ab \text{ 但无 } abc \end{cases}$$

表示当前状态下的最少修改次数。

答案： $\min(f_{n,0}, f_{n,1}, f_{n,2})$ ，即“构造出一个无 abc 的字符串”。

状态转移推导

由字符 s_i 决定的状态转移如下：

$$f_{i,0} = f_{i-1,0} + [s_i \neq a]$$

$$f_{i,1} = \min(f_{i-1,0} + [s_i = b], f_{i-1,1} + [s_i \neq b])$$

$$f_{i,2} = \min(f_{i-1,1} + [s_i = c], f_{i-1,2} + [s_i \neq c])$$

每个位置仅与上一个位置的三个状态有关，因此可以用矩阵描述。这是一个典型的“**min +**”形式的动态规划。

矩阵形式表示转移

由于转移由 $\min$ 和 $+$ 构成，该问题可在 $(\min, +)$ 半环下建立广义矩阵乘法：

$$C_{i,j} = \min_k (A_{i,k} + B_{k,j})$$

每个字符对应一个 3×3 的转移矩阵 M_i ，用于实现从 f_{i-1} 到 f_i 的状态转移。

构造矩阵

构造列向量 $v_{i-1}^{\rightarrow} = \begin{bmatrix} f_{i-1,0} \\ f_{i-1,1} \\ f_{i-1,2} \end{bmatrix}$ 通过广义矩阵乘法 ($\otimes = \min$, $\oplus = +$) 使得 $M \times v_{i-1}^{\rightarrow} = \vec{v}_i$,
易得

$$M = \begin{bmatrix} s_i = a & \infty & \infty \\ s_i \neq a & s_i = b & \infty \\ \infty & s_i \neq b & s_i \neq c \end{bmatrix}$$

线段树维护矩阵乘积

整个字符串可表示为矩阵积：

$$M = M_n \times M_{n-1} \times \cdots \times M_1$$

初始向量：

$$\vec{v}_0 = \begin{bmatrix} 0 \\ \infty \\ \infty \end{bmatrix}$$

则：

$$\vec{v}_n = M \times \vec{v}_0$$

每次修改仅影响单个字符对应的矩阵 M_i ，用线段树维护区间矩阵乘积即可。

时间复杂度： $O(3^3 \log n)$ 。

小结

- 将字符串 DP 抽象为 矩阵链乘问题；
- 运算规则换为 $(\min, +)$ ，形成新的半环；
- 每次单点修改仅影响一个矩阵；
- 用线段树维护矩阵乘积即可实现 “动态 DP”；
- 复杂度 $O(\log n)$ 每次操作。

核心思想：

“把 DP 状态转移写成矩阵形式 $\rightarrow$ 用数据结构维护矩阵积”

GOOD BYE!