单位蒙日矩阵相关算法的介绍

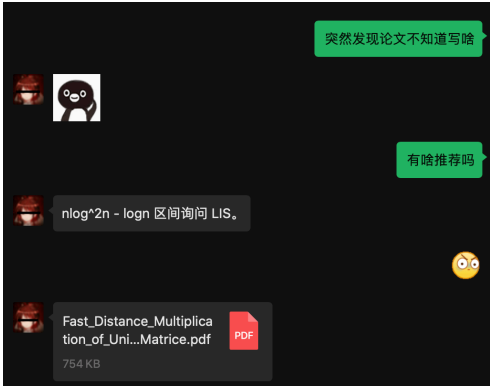
张景行

北京大学附属中学

2022 年 1 月 24 日

目录

为啥



一名不愿意透露姓名的 EI 学长的推荐。(当事人头像已打码)

解决了怎样的问题

$O(n \log^2 n)$ 预处理，$O(\log n)$ 回答区间 LIS 长度询问。
以及一些代价满足单位蒙日性质的 1D/1D 动态规划相关问题的快速求解。
由于时间原因，这里只略微讲一下如何做区间 LIS 长度询问，其它应用见之后的集训队论文。

大概是什么样子的

对于一个矩阵 A，若 A 满足四边形不等式，即对于 $i_1 \le i_2$，$j_1 \le j_2$，$A_{i_1,j_1} + A_{i_2,j_2} \le A_{i_1,j_2} + A_{i_2,j_1}$，那么称其为蒙日（monge）矩阵。

关于满足四边形不等式的 dp 优化是经典问题，但是如果 A 还满足以下性质...

大概是什么样子的

对于一个矩阵 A，若 A 满足四边形不等式，即对于 $i_1 \le i_2$，$j_1 \le j_2$，$A_{i_1,j_1} + A_{i_2,j_2} \le A_{i_1,j_2} + A_{i_2,j_1}$，那么称其为蒙日（monge）矩阵。

关于满足四边形不等式的 dp 优化是经典问题，但是如果 A 还满足以下性质...

$A_{i,0} = A_{n,j} = 0$，元素都是自然数，且相邻两个差不超过 1。

那么它就是一个简单亚单位蒙日矩阵，可以做到用 $O(n)$ 个整数（$O(n \log n)$ 个 bit）存储，并且 $O(n \log n)$ 进行矩阵距离乘法（$C_{i,j} = \min_k A_{i,k} + B_{k,j}$）。

于是就能解决很多问题！

一些定义

i^+, i^- 代表 $i+1/2, i-1/2$。

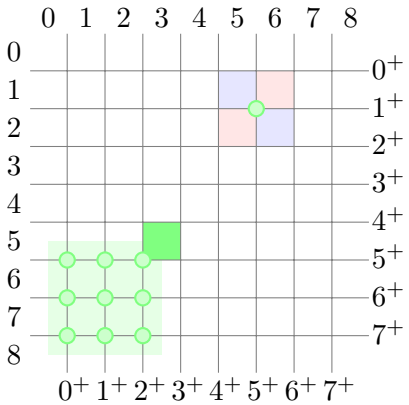
$[a:b], \langle a:b\rangle$ 代表 $\{a, a+1, \cdots, b-1, b\}, \{a+1/2, a+3/2, \cdots, b-1/2\}$。

对于一个 $[i_1:i_2], [j_1:j_2]$ 上的矩阵 A，定义其密度矩阵 A^M 为一个 $\langle i_1, i_2\rangle, \langle j_1:j_2\rangle$ 上的矩阵，其中 $A^M_{i,j} = A_{i^-,j^+} + A_{i^+,j^-} - A_{i^-,j^-} - A_{i^+,j^+}$。

对于一个 $\langle i_1:i_2\rangle, \langle j_1:j_2\rangle$ 上的矩阵 A，定义其分布矩阵 A^D 为一个 $[i_1, i_2], [j_1:j_2]$ 上的矩阵，其中 $A^D_{i,j} = \sum_{i'\in\langle i:i_2\rangle, j'\in\langle j_1:j\rangle} A_{i',j'}$。

其实就是二维前缀和以及二维差分，几乎互为逆运算。

一些定义



方格为整数，格线为半奇数。
格子内形成下标是整数的矩阵，也就是分布矩阵。
格线上（注意不包括外圈）形成下标为半奇数的矩阵，也就是密度矩阵。
绿色位置的分布矩阵是浅绿色区域里的和。
绿色位置的密度矩阵是红蓝色位置的和之差。
注意二维差分的时候丢失了边界上的信息，故并不是完全的逆运算，$A^{MD} = A$ 当且仅当第一列和最后一行全零。

一些定义

我们定义蒙日矩阵 A 是其密度矩阵 $A^M \geq 0$ 的矩阵。
一个矩阵是蒙日矩阵当且仅当满足四边形不等式，也就是 $\forall i_1 \leq i_2, j_1 \leq j_2, A_{i_1,j_1} + A_{i_2,j_2} \leq A_{i_1,j_2} + A_{i_2,j_1}$。
这是显然的，回忆上一页的图示，四边形不等式的一项就是 A^M 一个矩形的和，根据定义 ≥ 0。
定义简单蒙日矩阵是最后一行和第一列全零的蒙日矩阵，也就是上面说过的 $A^{MD} = A$ 的矩阵。

还是一些定义

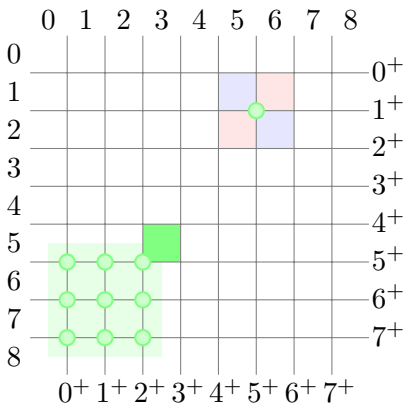
一般蒙日矩阵算法大家之前可能已经比较熟悉了，其实就是决策单调性优化 dp 那一套。

所以我们研究特殊的蒙日矩阵。

如果一个矩阵除了每行每列恰好一个 1，剩下的全 0，那么是置换矩阵。如果 A 是简单蒙日矩阵且 A^M 是置换矩阵，那么 A 是简单单位蒙日矩阵。

如果一个矩阵除了每行每列最多一个 1，剩下的全 0，那么是亚置换矩阵。如果 A 是简单蒙日矩阵且 A^M 是亚置换矩阵，那么 A 是简单亚单位蒙日矩阵。

简单单位蒙日矩阵



继续看这个图，切记这个密度和分布矩阵就是二维前缀和，那么很容易得到以下结论。

一个矩阵是简单单位蒙日矩阵当且仅当是元素是整数的简单蒙日矩阵，且第一行每一个元素比上一个多 1，最后一列每一个元素比上一个少 1。

一个矩阵是简单亚单位蒙日矩阵当且仅当是元素是整数的简单蒙日矩阵，且第一行每一个元素比上一个至多多 1，最后一列每一个元素比上一个至多少 1。

简单单位蒙日矩阵

有什么东西满足简单亚单位蒙日矩阵的条件呢？

对于一个数组，$A_{l,r}$ 代表 $[l,r)$ 内的最长上升子序列长度，那么

$$B_{l,r}=\begin{cases}r-l-A_{l,r} & r>l\\ 0 & r\le l\end{cases}$$ 是简单亚单位蒙日的。

对于一个数组，$A_{l,r}$ 代表 $[l,r)$ 内的不同元素个数，那么

$$B_{l,r}=\begin{cases}r-l-A_{l,r} & r>l\\ 0 & r\le l\end{cases}$$ 是简单亚单位蒙日的。

其它例子还有很多，可以发现只需满足它是蒙日的（满足四边形不等式），而且相邻两个位置最多差 1 即可。

距离乘法

定义 A 和 B 的距离乘法 $AB = C$，$C_{i,j} = \min_k A_{i,k} + B_{k,j}$。
接下来说的所有矩阵乘法都指的是矩阵的距离乘法，而不是一般定义的矩阵乘法。
可以证明，两个蒙日矩阵的乘积还是蒙日矩阵。两个简单单位蒙日矩阵的乘积还是简单单位蒙日矩阵。两个简单亚单位蒙日矩阵的乘积还是简单亚单位蒙日矩阵。

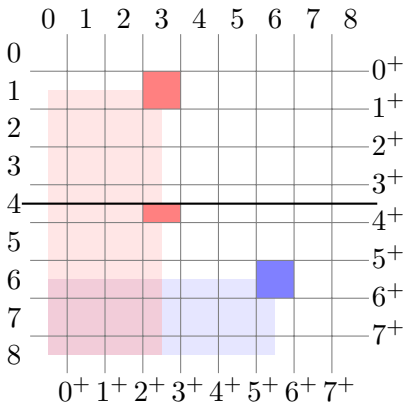
表示

首先研究简单单位蒙日矩阵的乘法。
注意到其密度矩阵只有 n 个 1，只需存储其密度矩阵的所有 1 的位置，可以用 $O(n)$ 个整数表示一个 $O(n^2)$ 的矩阵。一般是用其每行的 1 所在列的排列来表示的。
能 $o(n^2)$ 做大小为 n^2 的乘法的基础。

分治

若对于 A, B，我们想计算 $D = (A^D B^D)^M$。
考虑使用分治算法，对于 $n = 1$，只有唯一一个可能的矩阵，直接返回即可。
而若 $n > 1$，令 $C = A^D B^D$，观察到距离乘法的定义 $C_{i,j} = \min_k A^D_{i,k} + B^D_{k,j}$，可以考虑把 k 分成前后两部分，即令 $m = n/2$，分成 $[0 : m], [m : n]$ 两部分。
$C^L_{i,j} = \min_{k\in[0:m]} A^D_{i,k} + B^D_{k,j}, C^R_{i,j} = \min_{k\in[m:n]} A^D_{i,k} + B^D_{k,j}$。
令 A^{DL}, A^{DR} 为 A^D 的前 m 列和剩下的列的矩阵；令 B^{DL}, B^{DR} 为 B^D 的前 m 行和剩下的行的矩阵；那么上面的定义可以写成 $C^L = A^{DL} B^{DL}, C^R = A^{DR} B^{DR}$。
但还是不能显式的得到 A^D, B^D, C，但令 A^L, A^R 为 A 的前 m 列和剩下的列的矩阵，B^L, B^R 为 B 的前 m 行和剩下的行的矩阵。注意到 A^L, A^R, B^L, B^R 这些都只有 $O(n)$ 个 1，可以线性得到。
然后使用某种方式分治，得到 $D^L = (A^{LD} B^{LD})^M$，$D^R = (A^{RD} B^{RD})^M$。

表示 C^L, C^R



然而求出了 D^L, D^R 看上去并没有卵用。不过可以惊奇地发现，D^L, D^R 可以表示出 C^L, C^R 来。首先根据前缀和的定义。$A^{DL}_{i,j} = A^{LD}_{i,j}$，$B^{DR}_{i,j} = B^{RD}_{i,j}$，$A^{DR}_{i,j} = A^{RD}_{i,j} + A^{LD}_{i,m}$，$B^{DL}_{i,j} = B^{LD}_{i,j} + B^{RD}_{m,j}$。左图是 B^{DL}, B^{DR} 的解释。

表示 C^L, C^R

然后根据 C^L 的定义，

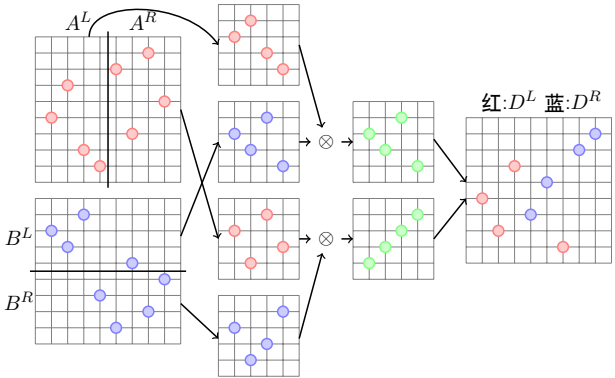
$$\begin{aligned} C^L_{i,j} &= \min_{k\in[0:m]} A^{DL}_{i,k} + B^{DL}_{k,j} \\ &= \min_{k\in[0:m]} A^{LD}_{i,k} + B^{LD}_{k,j} + B^{RD}_{m,j} \\ &= \left(\min_{k\in[0:m]} A^{LD}_{i,k} + B^{LD}_{k,j}\right) + B^{RD}_{m,j} \\ &= D^{LD}_{i,j} + B^{RD}_{m,j} \\ &= D^{LD}_{i,j} + D^{RD}_{0,j} \end{aligned}$$

其中最后一步，是由于 $D^{RD}_{0,j} = \min_{k\in[m:n]} A^{RD}_{0,k} + B^{RD}_{k,j} = \min_{k\in[m:n]} k - m + B^{RD}_{k,j}$，显然在 $k = m$ 取得最小。

表示 C^L, C^R

同理，$C^R = D^{RD}_{i,j} + D^{LD}_{i,n}$，而已经求出 $C^L = D^{LD}_{i,j} + D^{RD}_{0,j}$，而 $C_{i,j} = \min(C^L_{i,j}, C^R_{i,j})$。故我们可以用分治得到的这个 D^L, D^R 来表示 C 了。

分



要求 $D^L = (A^{LD}B^{LD})^M, D^R = (A^{RD}B^{RD})^M$。注意这样一件事，对于 A^L 的一个空行，它在 D^L 中也一定是空的，由于二维前缀和和下面一行相同，那么距离乘法也自然和下面一行相同。所以空行可以先删去再做完乘法后加回来，这样就转化成了两个规模减半的子问题了。

注意 D^L, D^R 的并是置换矩阵。

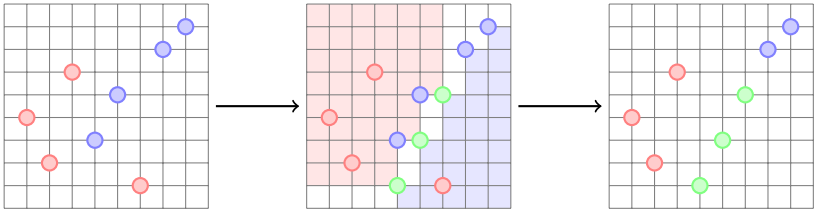
计算 $D = C^M$

求出了 D^L, D^R，$C^L = D^{LD}_{i,j} + D^{RD}_{0,j}, C^R = D^{RD}_{i,j} + D^{LD}_{i,n}$。现在就可以根据 $C_{i,j} = \min(C^L_{i,j}, C^R_{i,j})$ 求出 $D = C^M$ 了。

$C_{i,j} = \min(C^L_{i,j}, C^R_{i,j})$ 取在哪一边取决于 $\delta_{i,j} = C^L_{i,j} - C^R_{i,j} = D^{LD}_{i,j} + D^{RD}_{0,j} - D^{RD}_{i,j} - D^{LD}_{i,n} = (D^{RD}_{0,j} - D^{RD}_{i,j}) - (D^{LD}_{i,n} - D^{LD}_{i,j})$ 的符号。

我们发现 $\delta_{i,j} = (D^{RD}_{0,j} - D^{RD}_{i,j}) - (D^{LD}_{i,n} - D^{LD}_{i,j})$，其实就是左上角 D^R 的和减去右下角 D^L 的和。即 $\delta_{i,j} = \sum_{i' \in \langle 0:i \rangle, j' \in \langle 0:j \rangle} D^R_{i',j'} - \sum_{i' \in \langle i:n \rangle, j' \in \langle j:n \rangle} D^L_{i',j'}$。

δ 的性质

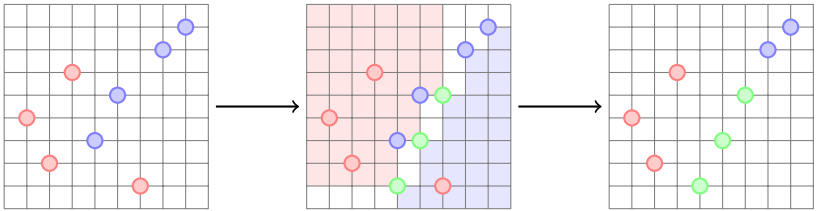


上面是从 D^L, D^R 获得答案 D 的过程，$D_{i,j} = C_{i^-,j^+} + C_{i^+,j^-} - C_{i^-,j^-} - C_{i^+,j^+}$。注意到 $\delta_{i,j}$ 单调且相邻两项最多差 1。对于 $D_{i,j}$，若 $\delta_{i^-,j^-} \geq 0$，那么这四个点的 δ 均 ≥ 0，最小值均在 C^R 取到，此时 $D_{i,j} = D^R_{i,j}$。

$\delta_{i^+,j^+} \leq 0$，那么这四个点的 δ 均 ≤ 0，最小值均在 C^L 取到，此时 $D_{i,j} = D^L_{i,j}$。

所以说，上图和红色（$\delta_{i,j} < 0$）区域有接触的红色（D^L 中的 1 的位置）的点都保持不变。同理，与蓝色区域有接触的蓝色的点都保持不变。

δ 的性质



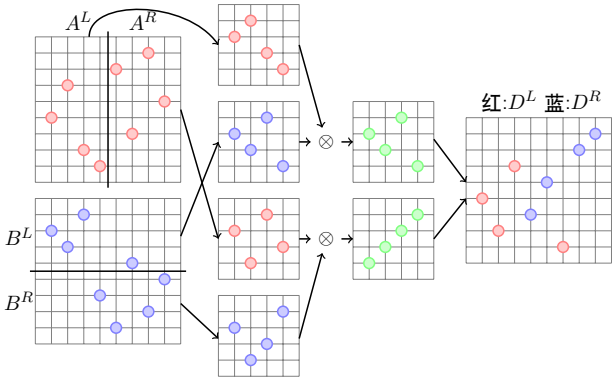
可是那些绿色的点呢？注意唯一不属于上面的情况的点是 $\delta_{i^-,j^-} < 0$，$\delta_{i^+,j^+} > 0$，的情况，由于相邻两项最多差 1，所以这种情况一定有

$\delta_{i^-,j^-} = -1, \delta_{i^+,j^+} = 1, \delta_{i^+,j^-} = \delta_{i^-,j^+} = 0$。

$D_{i,j} = C^L_{i^-,j^+} + C^L_{i^+,j^-} - C^L_{i^-,j^-} - (C^L_{i^+,j^+} - 1) = D^L_{i,j} + 1$

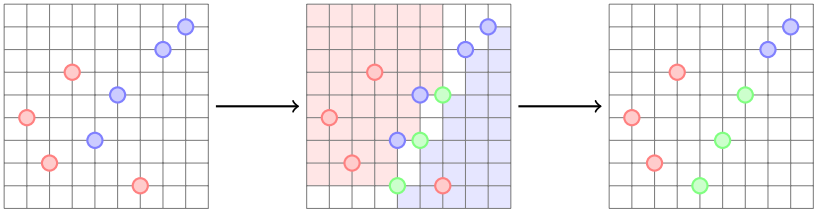
由于 $D^L_{i,j} \geq 0, 1 \leq D_{i,j} \leq 1$，这种情况下 $D_{i,j}$ 一定为 1。

总结



先把 A, B 分开，去除间隔，分治，然后再加回缺少的间隔，得到 D^L, D^R，注意他们的并一定是个置换矩阵。

总结



然后使用双指针求出 $\delta_{i,j} > 0$，$\delta_{i,j} < 0$ 的上下边界，两边界的交点处加入新的 1，然后不再对应颜色区域内的 1 去掉，得到最终的 D。

除了分治的部分都可以线性，所以复杂度是 $O(n \log n)$。代码并不长，只有 40 行左右，如果实现的好一些常数不大，1s 大约可以跑 10^6。

亚单位乘法

上面只讨论了单位乘法，对于亚单位的，其密度矩阵是亚置换的，也就是可能有空行空列。

令亚置换矩阵 A, B，我们要求 $D = (A^D B^D)^M$。

注意到还是，A 的全零行可以去掉，B 全零列可以去掉。但注意去掉后 A 可以有全零列，B 还可能有全零行。

一个做法是向 A 上面随便补一些行，B 右边随便补一些列，补成置换矩阵。由于根据二维前缀和是从左下到右上，补的这些行列对我们求的乘积的左下角没有影响，所以可以直接随便补，然后把乘积的左下角拿出来就好了。

就是 $\begin{bmatrix} * \\ A \end{bmatrix}^D \begin{bmatrix} B & * \end{bmatrix}^D = \begin{bmatrix} * & * \\ D & * \end{bmatrix}^D$，然后直接套用上述的乘法算法即可。

处理询问

回想起之前说的，可以证明，对于一个数组，$A_{l,r}$ 代表 $[l,r)$ 内的最长上升子序列长度，那么 $B_{l,r}=\begin{cases}r-l-A_{l,r} & r>l\\ 0 & r\le l\end{cases}$ 是简单亚单位蒙日的。

所以 B^M 里最多有 n 个 1，那么求 B 的一项的值可以二维数点。

所以只需要求出 B^M 就好了。不像大多数题目需要使用各种数据结构解决，从某种意义上，我们这里将直接计算这一大小为 $(n+1)^2$ 的矩阵，只不过这个矩阵可以用 $O(n)$ 个数表示。

分治

考虑分治，把序列分成上下两部分，也就是对于一个阈值 x，分成值 $> x$ 和 $\leq x$ 的。$B_{l,r}$ 的意义就是从 l 到 r 走上升的路，最少没经过多少点。

那么枚举这个上升的路何时跨过 x。也就是定义一个 $B^L_{l,r}$ 代表只走 $\leq x$ 的上升路最少不经过多少，和最多不经过多少，$B^R_{l,r}$ 则是只走 $> x$。可以发现 $B = B^L B^R$，也就是如果我们可以分治求出 B^{LM}, B^{RM} 就可以使用刚才说的乘法直接求出 B^M。

分治

那么唯一的细节是如何把计算 B^L, B^R 转化为一个子问题。以 B^L 为例。
自然的，我们试图删去那些 $> x$ 的下标，因为它们永远不可能在序列里。求出后，再插入对应的空行空列。
但注意到，这时我们删去的东西的代价并不是 0 而是 1，所以还需要在插入的空行空列的对角线上补上 1 即可。这样一个区间就会多出其中删去空行的代价。
这个过程显然可以 $O(n)$。

复杂度

分治每轮需要一个 $O(n \log n)$ 的乘法，且子问题规模减半，复杂度 $T(n) = 2T(n/2) + O(n \log n) = O(n \log^2 n)$。

然后询问离线二维数点，只需要 $O((n+q) \log n)$。

总复杂度 $O(n \log^2 n + q \log n)$。

复杂度

分治每轮需要一个 $O(n \log n)$ 的乘法，且子问题规模减半，复杂度 $T(n) = 2T(n/2) + O(n \log n) = O(n \log^2 n)$。

然后询问离线二维数点，只需要 $O((n+q) \log n)$。

总复杂度 $O(n \log^2 n + q \log n)$。

如果对算法细节有疑问的可以上 library checker （judge.yosupo.jp）的 Static Range LIS Query 一题中查看 dengyaotriangle 的提交。

总结

这里只讲了单位蒙日矩阵的一个应用，但已经足以说明其潜力，感兴趣的可以关注集训队论文。
总之，所有满足这个简单亚单位蒙日性质的距离，代价之类的计算都可以快速进行，很多问题能做到以前意想不到的复杂度。
如何将该算法引入 OI，以及如何利用它命制优秀的题目仍是一个很大的问题，而这套算法背后蕴含着很多复杂的东西，我也没有完全理解，希望我的这次交流和集训队论文能起到抛砖引玉的作用，希望有兴趣有能力的人可以把这套理论发扬光大。

感谢聆听