Двумерная разреженная таблица

Двумерная разреженная таблица (англ. *2D Sparse Table*) — структура данных, которая позволяет решать задачу online static RMQ.

Задача:

Дан двумерный массив $A[1\ldots N][1\ldots M]$ *целых чисел. Поступают запросы вида* (x_1, y_1, x_2, y_2) *такие, что* $x_1 \leqslant x_2$ *и* $y_1 \leqslant y_2$ *, для каждого из которых требуется найти минимум среди* $A[i][j], x_1 \leqslant i \leqslant x_2$ *и* $y_1 \leqslant j \leqslant y_2$.

Содержание

Структура 2D Sparse Table

В целом структура 2D Sparse Table схожа со структурой обычной разреженной таблицы.

Разреженная таблица представляет собой четырехмерный массив: $ST[N][M][LOGN][LOGM]$, где $LOGN = \log(N), LOGM = \log(M)$.

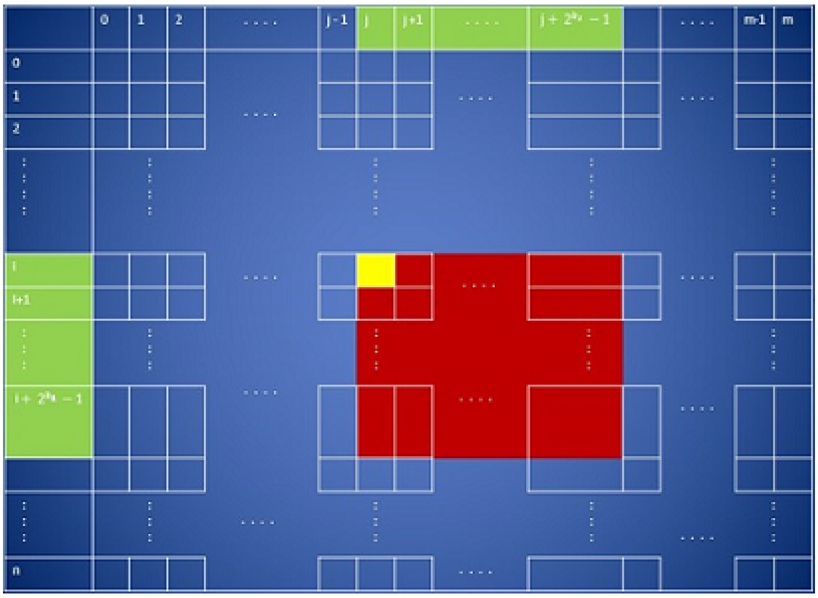
$$ST[i][j][k_1][k_2] = \min_{r=i,\ldots,i+2^{k_1}-1, c=j,\ldots,j+2^{k_2}-1} A[r][c], r < N, c < M$$

То есть в ячейке структуры мы храним минимум для подматрицы, длины сторон которого являются некоторыми степенями двойки.

Рассмотрим иллюстрации.

Слева изображен общий случай. Пусть $n+1$ — количество строк, $m+1$ — количество столбцов массива A. Рассмотрим элемент на позиции (i, j), который выделен желтым цветом. Данный элемент является левым верхним углом прямоугольника, стороны которого равны 2^{k_1} и 2^{k_2}. Прямоугольна выделен красным, а проекции его сторон - зеленым. Тогда в $ST[i][j][k_1][k_2]$ будет храниться минимум из всех элементов, которые входят в красную и желтую область.

Справа изображен частный случай, когда N = 11, M = 11. Посмотрим, что будет храниться в $ST[2][1][2][3]$: Клетка (2, 1), которая выделена желтым цветом, является левым верхним углом красного прямоугольника, длины сторон которого 2^2 и 2^3 (проекции этих сторон выделены зеленым

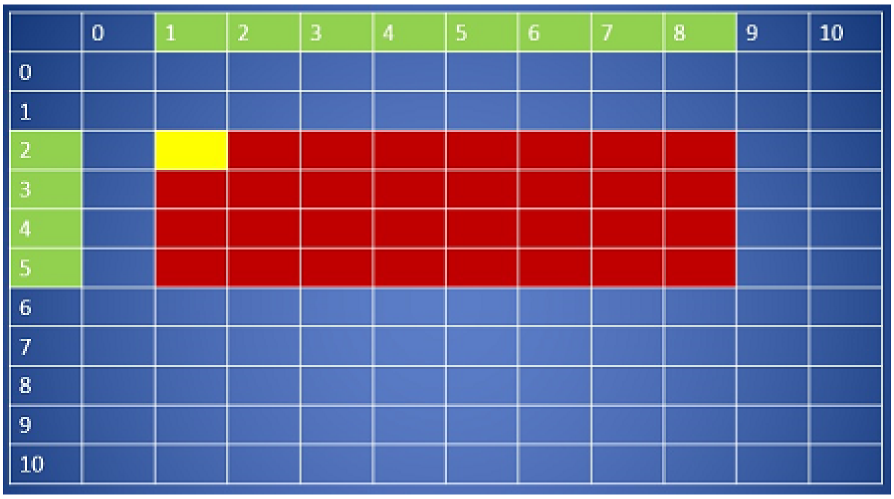


цветом). И по определению выше, $ST[2][1][3][2]$ хранит минимум из элементов красной области.

Реализация 2D Sparse Table

Построение

Изначально заполним таблицу следующим образом:



$$ST[i][j][k_1][k_2] = \begin{cases} \infty, & \text{если } k_1 \neq 0 \vee k_2 \neq 0\ ; \\ A[i][j], & \text{если } k_1 = 0 \wedge k_2 = 0\ ; \end{cases}$$

Далее мы считаем одномерную разреженную таблицу для каждого столбца:

```
for i = 0 to N
    for j = 0 to M
        for lg = 1 to log(M)
            ST[i][j][0][lg] = min(ST[i][j][0][lg − 1], ST[i][j + 2^(lg−1)][0][lg − 1])
```

Следующим шагом мы можем обновить значения для подматриц, так как для всех столбцов ответы уже известны. Будем это делать по аналогичному алгоритму для 1D Sparse Table.

```
for k1 = 1 to log(N)
    for i = 0 to N
        for k2 = 0 to log(M)
            for j = 0 to M
                ST[i][j][k1][k2] = min(ST[i][j][k1 − 1][k2], ST[i + 2^(k1−1)][j][k1 − 1][k2])
```

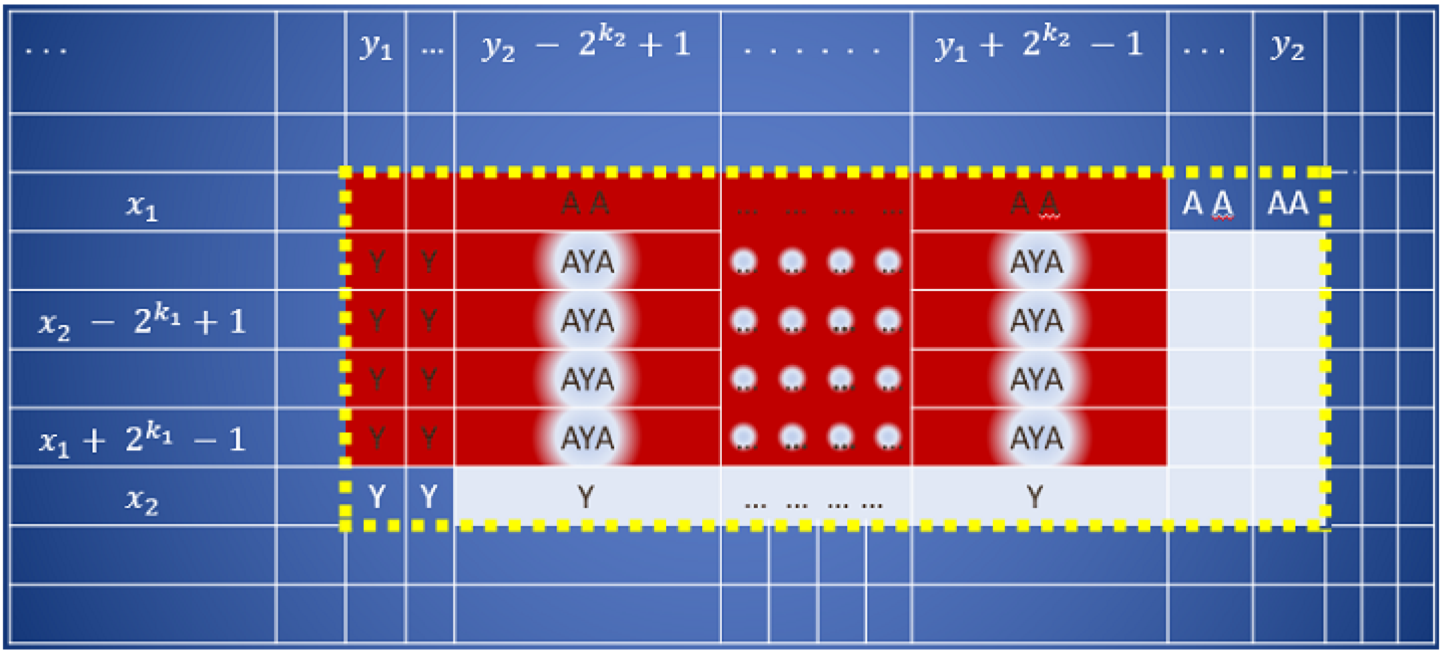
Таким образом мы получили двумерную разреженную таблицу за $O(NM\log(N)\log(M))$

Ответы на запросы

Для ответа на запрос $\mathrm{RMQ(l, r)}$ в 1D Sparse Table использовалось пересечение отрезков $[l, l + 2^k]$ и $[r - 2^k + 1, r]$, где $k = \lfloor \log(SZ) \rfloor$, SZ — размера массива для одномерной задачи.

Тут мы будем пользоваться аналогичным утверждением для матриц. Таким образов минимум на подматрице (x_1, y_1, x_2, y_2) будет высчитываться следующим образом:

$k_1 = \lfloor \log(x_2 - x_1 + 1) \rfloor$
$k_2 = \lfloor \log(y_2 - y_1 + 1) \rfloor$
$ans_1 = ST[x_1][y_1][k_1][k_2]$ `// красный прямоугольник`
$ans_2 = ST[x_2 - 2^{k_1} + 1][y_1][k_1][k_2]$ `// Y-прямоугольник`
$ans_3 = ST[x_1][y_2 - 2^{k_2} + 1][k_1][k_2]$ `// AA-прямоугольник`
$ans_4 = ST[x_2 - 2^{k_1} + 1][y_2 - 2^{k_2} + 1][k_1][k_2]$ `// белый прямоугольник`
$ans = \min(ans_1, ans_2, ans_3, ans_4)$



Таким образом мы получаем ответ на запрос $\mathrm{RMQ(x_1, y_1, x_2, y_2)}$ за $O(1)$, если предпосчитать логарифмы двоек, например так:

$lg[1] = 0$
for $i = 2$ **to** $\max(N, M)$
    $lg[i] = lg[i/2] + 1$

Обобщение на большие размерности

Можно заметить, что возможно реализовать и D-мерную разреженную таблицу за $O((N\log(n))^D)$ памяти и $O((N\log(n))^D)$ времени на построение, где ответ на запрос, например, $\mathrm{RMQ(l, r)}$ будет выполняться за $O(2^D)$.

Способ построения: Если в данном случае, для того, чтобы построить двумерную структуру мы сначала должны были построить одномерную, то также и в случае с D-мерной структуре - сначала нужно построить (D-1)-мерную, а из нее получить D-мерную.

Ответ на запрос: Абсолютно аналогичен рассмотренному здесь, только обобщается до размерности D.

См. также

- Решение RMQ с помощью разреженной таблицы