АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА
СПЕЦИАЛНИ РАЗХОДКИ

Решение със сложност $O(Q * MN * (M + N))$

Идеята на това решение е да отговорим на всяка заявка, като следваме маршрута на съответната специална разходка, без да прескачаме клетки (включително и тези, в които не променяме посоката на движение). Тъй като всяка разходка има най-много $O(MN)$ клетки, в които променяме посоката, и в най-лошия случай за намирането на две последователни такива ще имаме нужда от $O(M + N)$ придвижвания по таблицата, общата сложност за една заявка е $O(MN * (M + N))$.

Решение със сложност $O(Q * MN + MN * (M + N))$

Една оптимизация на предходния подход е да запазим предварително за всяка клетка от таблицата, коя е следващата клетка с по-малка стойност във всяка от четирите посоки. Така ще може да намираме следващата клетка в разходката със сложност $O(1)$. Разбира се, самият прекъмпют отнема $O(MN * (M + N))$, защото за всяка клетка, трябва да направим обхождане на реда и на колоната, в която се намира.

Решение със сложност $O(Q * MN)$

Промяната в това решение, спрямо предишното, е в бързината на прекъмпюта. Може да забележим, че вместо да смятаме за всяка клетка от един и същи ред коя е най-близката клетка с по-малка стойност отляво на тях поотделно, можем да поддържаме монотонно намаляващ стек по този ред и така да намерим търсените клетки за $O(N)$. Основата на тази идея е в наблюдението, че ако клетката на колона j_1 има по-малка стойност от клетката на колона j_2 и $j_2 < j_1$, то никога за някоя колона $j_3 > j_1$, клетката на колона j_2 няма да е следваща в разходката – т.е. тя се явява безполезна за всички колони след j_1 и можем да я премахнем от стека. Така, когато приключим с премахването на клетките с по-големи или равни стойности от върха на стека, ще намерим следващата клетка наляво от клетката, до която сме стигнали. Тъй като добавяме всяка клетка точно веднъж в стека и я изваждаме най-много веднъж, то сложността е линейна. Така сложността на целия прекъмпют за четирите посоки (като използваме аналогична идея за останалите три) става $O(MN)$.

Пълно решение

Следващата стъпка в решението е да оптимизираме намирането на клетката след l-тия голям скок. Очевидно разглеждането на клетките в спираловидните разходки последователно за всяка поотделно е неефективно. Забелязваме обаче, че графът, който се получава, ако за всяка клетка от таблицата създадем по четири върха (по един във всяка посока) и свържем последователните клетки с ребро в обратна посока (от следващия връх, към който бихме скочили при спираловидна разходка, към текущия) е гора от коренови дървета с корени – върховете, съответстващи на двойки от клетка и посока, от които не можем да продължим разходката. Така реално можем да сведем задачата до намирането на l-ти предшественик в кореново дърво, но се изисква още малко съобразителност.

Целта ни е с едно обхождане на целия граф да намерим отговорите на всички заявки. Затова е удобно във всяка клетка от таблицата да си запазим кои заявки се отнасят до нея. Реално всяка заявка се отнася върха в графа, създаден за таблицата и посока нагоре (защото това е началната посока на всички разходки). Обхождаме графа в дълбочина, започвайки от различните корени. Така, когато достигнем до връх, в който имаме заявка, знаем, че DFS-ът е преминал точно през тези върхове, през които ще премине и разходката, започваща от съответната клетка, но в обратен ред.

АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА
СПЕЦИАЛНИ РАЗХОДКИ

Сега трябва да преценим, каква допълнителна информация ни е необходима, за да намерим отговора на една заявка. Тъй като не знаем предварително стойността на l_i за съответната заявка, ни е необходимо да поддържаме всички върхове, в които ще се озовем след голям скок. Но това е сравнително лесно – поддържаме един стек (реално вектор, за да имаме достъп до всички елементи в него) и когато преминем по ребро, свързващо две клетки с разлика в стойностите поне K, добавяме първия връх в стека. Когато обхождането се върне, премахваме този връх от върха на стека, преди да преминем към следващото ребро (когато евентуално може да го добавим пак). Така, когато попаднем на връх, в който има заявка, е достатъчно да проверим кой е l_i-тия пореден връх от върха на стека и това е отговорът. Ако стекът има по-малка големина, тогава отговорът е $(0, 0)$.

Решение със сложност $O((Q + MN) * \log(MN))$

Предходното решение се възползва от факта, че можем да отговаряме на заявките офлайн. Но, ако такова отговаряне не е позволено, т.е. трябва да намерим отговора на текущата заявка, преди да получим следващата, то това решение не е възможно. Вместо това, може (и видях, че някои състезатели са оувъркилнали с такова решение, което (за тяхна жалост и мое щастие) не хваща последната подзадача) да конструираме sparse table по графа (запазвайки само ребрата с големина поне K) и да отговаряме на заявките за l-ти предшественик с binary lifting.

Автор: Добрин Башев