Анализ на задача aquarelle

Тагове: динамично програмиране, умно обхождане, наблюдения

Накратко задачата е, че имаме N множества от цветове, започваме с интервал о$[l, r]$ и може да правим стъпка вляво или вдясно само ако има нов цвят, който не сме срещали досега. Целта ни е да стигнем до пълния интервал от множества $[1, N]$. По принцип такъв тип задача може да се разглежда и отзад-напред - започваме с интервала $[1, N]$, махаме всеки път едно ляво или дясно множество само ако има цвят в него, който вече няма да имаме в останалите множества, и искаме да стигнем началния интервал $[l, r]$. В случая такова разглеждане на задачата не ни помага за по-добра идея, но е нещо, което е хубаво да имаме предвид. Нека означим сумата от големините на множествата $K_1 + K_2 + \cdots + K_N$ с K.

Решение на втора подзадача - 11 точки

Предвиденото решение за тази подзадача е пълно изчерпване, написано по най-простия възможен начин, с цел да има всеки какво да направи по задачата, макар че крайното класиране показва, че това не се е получило. Използваме рекурсия, за да разгледаме всяко възможно решение като на всяка стъпка трябва да проверим дали можем да направим съответното движение, като видим има ли цвят във фиксираното множество, който не се е срещал досега.

Сложност: $O(Q2^N K^2)$. (макар тази сложност да изглежда малко голяма, на практика максималния брой нужни операции е значително по-малък)

Решение на трета подзадача - 29 точки

Тази подзадача показва защо е добре да се пишат и лесните решения за малко точки по задачите като това в предната подзадача. Лесно се вижда, че можем да приложим техниката мемоизация за рекурсията на пълното изчерпване. Няма значение колко пъти ще стигнем до интервал от включени множества $[L, R]$, защото всеки път ще направим едно и също. Така можем да дефинираме просто динамично $dp[L][R]$, в което да записваме дали има решение, така че да стигнем до интервала $[1, N]$ накрая. Очевидно в общия случай $dp[L][R] = 1$, ако можем да включим множеството A_{L-1} и $dp[L-1][R] = 1$ или можем да включим A_{R+1} и $dp[L][R+1] = 1$. Иначе $dp[L][R] = 0$.

Възможно е да минем подзадачата и без да оптимизираме намирането на това дали ход вляво или вдясно е валиден, но нека да обсъдим как можем да го забързаме. Лесно можем да поддържаме *unordered_multiset* с цветовете, които имаме досега от множествата и така е достатъчно да минем през фиксираното множество и с константна сложност (в средния случай) да видим всеки цвят дали се е срещал досега. Също предвид, че за стойността на $dp[L][R]$ няма значение от какъв интервал сме тръгнали, то няма нужда всеки път да чистим масива dp.

Сложност: $O(Q + N^2 K)$. (макар тази сложност отново да изглежда страшно, на практика максималния брой нужни операции отново е значително по-малък)

Решение на четвърта подзадача - 53 точки

Решението за тази подзадача не се различава много от предишното, но трябва да направим важна стъпка - да оптимизираме допълнително проверката като сме стигнали до интервал $[L, R]$ дали можем да включим A_{L-1}, или можем да включим A_{R+1}. Нека разглеждаме цвят c от множеството A_{L-1}. Достатъчно е да разберем къ-

де е първото срещане на c вдясно. Нека това срещане е на индекс i. Тогава, ако $L \leq i \leq R$, то c вече се среща в интервала, а ако $R < i$, то този цвят ще е нов. Аналогично, ако разглеждаме цвят c от A_{R+1}, то ще ни трябва първото срещане на c вляво. Ще използваме подхода умно обхождане - ще направим предпроцесване преди да почнем да обработваме заявките, с което да знаем отговорите на тези въпроси. За целта ще обходим множествата от дясно наляво и ще пазим в масива prv за всеки цвят индекса, на който последно се е срещал. Така, като разглеждаме даден цвят c от A_i, просто записваме $prv[c]$ за първото срещане вдясно на този цвят и след това слагаме $prv[c] = i$. Аналогично, правим обхождане и от ляво надясно и допълваме информацията за първото срещане вляво, като ползваме помощен масив nxt. Тук се възползваме от това, че номерата на цветовете са до 10^6, иначе трябваше да направим компресия за този подход.

Авторовото решение за тази подзадача дори прави нещо малко по-добро. Няма нужда да знаем за всички цветове в множествата, първото им срещане надясно и наляво. Нека отново се чудим дали можем да добавим множеството A_{L-1} към интервала $[L, R]$. Достатъчно е да знаем максималния индекс на първото срещане вдясно на цвят от A_{L-1} и да видим дали този индекс е по-голям от R. Така в авторовото решение в допълнителния масив $maxr$ за всяко множество A_i, $maxr[i]$ е максималният индекс на първото срещане вдясно на цвят c от A_i. Допълнително, ни трябва и $minl$, така че в $minl[i]$ да записваме минималния индекс на първото срещане вляво на цвят от A_i.

Като напишем това решение могат да се пробват различни чийтове, които не би трябвало да хващат следващата подзадача. Един такъв е вече споменатото рачешко решение, при което започваме от интервала $[1, N]$ и премахваме множества. По подобен начин можем да намираме dp-то. За да хващаме евентуално следващата подзадача можем вместо да имаме двумерен масив да имаме масив от *unordered_map*-ове и да се надяваме на слаби тестове. Също могат да се прилагат рандомизирани подходи за това дали да ходим вляво или вдясно (когато имаме две възможност) и прочие.

Сложност: $O(K + Q + N^2)$.

Решение на пета подзадача - 89 точки

Това е подзадача с най-голям брой точки, защото тук е най-трудната и съществена стъпка да оптимизираме решението. Едно съществено наблюдение е следното. Нека разглеждаме множеството A_i. По някое време трябва да го включим в текущия интервал. Но очевидно тогава интервалът от множества, които сме включили трябва да е станал $[i + 1, R]$, т.е. да се допира до i. Така можем да намерим въобще кое е най-голямото R, при което можем да добавим A_i към $[i + 1, R]$. Лесно можем да забележим, че до някаква стойност е възможно и от някоя стойност нататък не е възможно (а може и до края да е възможно), т.е. е монотонно и би могло да се търси с двоично търсене. За съжаление, въпросът, който трябва да задава двоичното търсене е малко сложен и няма как да се отговаря бързо без по-сложни структури от данни. Един начин това да стане по-лесно е например за всеки цвят да запишем индексите, на които се среща. След което, като сме на фиксирано множество A_i намираме за всеки цвят с двоично търсене в индексите на срещане, първата позиция вдясно от i, на която се среща и всъщност търсеното R е минималната такава позиция -1. Нека тези индекси записваме в масива $maxr$ (забележете, че тези стойности са почти същите като тези в предната подзадача, но намалени с 1). Разбира се, тази информация ще трябва да намерим и за множествата, които се добавят като

разширяваме вдясно - тях ще записваме в масива $minl$.

Нека сега разгледаме общия случай, когато сме стигнали до интервала $[L, R]$, като $1 < L$ и $R < N$. С подобни разсъждения на предната подзадача знаем, че като гледаме дали $R \leq maxr[L-1]$ и $minl[R+1] \leq L$ можем да разберем дали можем да ходим наляво (включвайки лявото множество) или надясно (включвайки дясното множество). Да забележим, че ако можем да ходим само в едната посока, то значи текущия случай няма решение, защото никога няма да можем да ходим в другата посока. Затова да допуснем, че можем да ходим и в двете посоки и трябва да преценим къде да отидем, така че да си гарантираме решение, ако има. За тази цел можем да видим дали например ходенето наляво няма да прецака добавянето на някое от бъдещите множества вдясно $A_{R+1}, A_{R+2}, \ldots, A_N$. Ние вече знаем $minl[R+1], minl[R+2], \ldots, minl[N]$ и проблем ще стане, ако има $minl[j] > L-1$ за $R+1 \leq j \leq N$. Достатъчно е да направим проверката:

$$\max_{R+1 \leq j \leq N} minl[j] > L-1$$

Понеже това са суфиксни максимуми, то можем да ги намерим за линейно време, след като сме намерили масива $minl$. Така, нека имаме $suf[i] = \max_{i \leq j \leq N} minl[j]$ и също така $pref[i] = \min_{i \leq j \leq N} maxr[j]$, за да гледаме дали няма да прецакаме добавянето на някое ляво множество. Сега вече има смисъл да ходим наляво само ако $suf[R+1] \leq L-1$, а надясно само ако $pref[L-1] \geq R+1$. Оказва се, че това е достатъчно да гарантираме, че ако има решение, ще го намерим. Нека да допуснем, че това не е така, т.е. стигаме до интервал $[L, R]$, за който $suf[R+1] > L-1$ и $pref[L-1] < R+1$, но има решение. Тогава интервалът $[L, R]$ се оказва непреодолима пречка. Нека да разгледаме алтернативни движения по-рано, при които да стигнем до интервал $[L', R]$ (все по някое време трябва да стигнем R). Ако $L' < L$, то от $suf[R+1] > L-1 > L'-1 \implies suf[R+1] > L'$, откъдето има някое множество отдясно, чиито цветове вече са в цветовете на интервала. Така единствената ни възможност е $L' > L$, но тогава знаем, че $pref[L'-1] < pref[L-1] < R+1 \implies pref[L'-1] < R$ и от тук има някое множество вляво, чиито цветове са вече в интервала.

Така обобщено първо намираме за всеки цвят на кои индекси се среща. След това с двоични търсения пресмятаме масивите $maxr$ и $minl$ за това докъде най-много надясно и най-много наляво може да е интервала, така че да включим дадено множество. После обобщаваме тази информация с масив suf, който е от суфиксните максимуми на $minl$ и с масива $pref$ - от префиксните минимуми на $maxr$. Накрая за дадена заявка, когато сме стигнали до интервал $[L, R]$ и имаме потенциално движение и в двете посоки, то се придвижваме наляво, когато $suf[R+1] \leq L-1$, а ако това не е така, то пробваме надясно, стига да е изпълнено $pref[L-1] \geq R+1$. Ако и двете не са изпълнени, то нямаме решение. Също трябва да се внимава за случаите $L = 1$ и $R = N$, както и за това, че може още от начало $pref[L-1] < R$ или $suf[R+1] > L$.

Сложност: $O(K \log_2 K + QN)$.

Решение на шеста подзадача - 100 точки

Предната подзадача беше предвидена по-скоро за по-бавни имплементации на описаните идеи. Разбира се, ако използваме линейното намиране на $maxr$ и $minl$, което вече описахме при четвърта подзадача, вместо подхода с двоични търсения от предната подзадача, то ще получим линейно решение.

Сложност: $O(K + QN)$.

Идея: Марин Йорданов
Реализация: Илиян Йорданов