Анализ на задача chess

Подзадача 1 и 3

Това са подзадачите, които предполагат наивни подходи. Заявките от тип 1 ще решаваме с for-цикъл, а тези от тип 2 като ръчно се спускаме в интервала, зададен от тях, чрез рекурсия.

Подзадача 2

Продължаваме да нямаме упражнения от тип 2. Понеже n е с нормалната си дължина тук обаче наивния подход няма да е достатъчен.

Ако погледнем на задачата под контекста на задача за интервали, можем да се убедим, че идея, базирана на sweepline, звучи смислено. Наистина, ако обикаляме позициите на редицата, която ще изведем финално отляво надясно, можем да различим два типа събития: *създаване* на интервал и *унищожаване* на интервал. Можем да отбележим тези събития или чрез елементи във `std::vector`, които по-късно да сортираме по позицията, която афектират (и съответно да получим сложност $O(n \log n)$) или в броячен масив да отбележим началото с $+1$ и края с -1 (за да излязат сметките в реалност ще отбележим първата позиция *след* края). Същинската стойност за всяка позиция в резултатната редица всъщност е префиксна сума в този броячен масив. Финално имаме $O(n)$ времева сложност.

Този блог влиза в повече детайли и предлага алтернативна интуиция за второто решение.

Подзадача 4

Главното препятствие в задачата ще са именно упражненията от тип 2 – с тези от тип 1 още в подзадача 2 се справяме достатъчно ефикасно.

Ако визуално си представим списъка от упражнения, то тип 2 всъщност се "пресяга" назад към миналите упражнения и ги активира още веднъж. Момент на досещане е да обхождаме упражненията в обратен на зададения ред – така всъщност тип 2 ни казва "Някакви бъдещи упражнения ще се извършат v допълнителен брой пъти." Когато стигнем до тези упражнения можем вместо "добави 1 в интервал" да направим "добави v в интервал". Така всъщност се осмисля идеята за всяко упражнение да пазим брояч колко пъти ще бъде изпълнено то – при тип 1 това ще афектира с колко променяме стойностите в интервалите, а при тип 2 това ще промени колко пъти бъдещите упражнения се изпълняват (навързани упражнения от тип 2). Тук е достатъчно наивно поддържане на броячите.

Алтернативно решение (благодарност на Виктор Лазаров): Друго решение се уповава на наблюдението, че състоянието на редицата след изпълнението на заявките в интервала $[l..r]$ всъщност може да се представи чрез състоянието на редицата след изпълнението на заявките $[1..r]$ "минус" същото след заявките в $[1..(l-1)]$. Тук разликата на две редици разбираме като на всяка позиция слагаме разликата от стойностите и в двете редици. Ако запазим състоянието на редицата за всеки префикс от заявки $[1..i]$, можем да отговаряме на заявките за $O(n)$ време и да получим обща времева сложност $O(nq)$. Забележете, че сложността по памет също се вдига тук до $O(nq)$ – тоест ако подзадачата беше за $n, q \leq 10^4$, това решение нямаше да работи. (Упражнение за читателя: защо сложността на заявките от тип 1 също се вдигна на $O(n)$, а не остана $O(1)$?)

Подзадача 5

Броячите, които въведохме в миналата подзадача се променят чрез заявки “броячите в интервал $[l, r]$ се увеличават с v”. Но това е именно заявката, която решихме доста ефикасно в подзадача 2. Независимо, че накрая имахме една стъпка на изчисляване на префиксни суми, тя всъщност може да се поддържа успоредно на обхождането в обратен ред на заявките, което е основния момент от подзадача 4.

Анализ: Иван Лупов(заета)