Анализ на задача `subway`

Тагове: наблюдения, динамично програмиране, двоично търсене, приоритетна опашка

Решение на втора и трета подзадача - 23 точки

Тук предвиденото решение е симулация на сядането на пътници с номера до 3.10^5. Достатъчно е в приоритетна опашка да поддържаме всички текущи дупки. При сядането на нов пътник премахваме от приоритетната опашка най-лявата максимална дупка, в чийто център сяда пътникът, и разделяме дупката на потенциално две нови дупки, които добавяме в приоритетната опашка. След като сме намерили отговорите за тези пътници директно отговаряме на заявките, като трябва да внимаваме, че може да има въпроси и за първите N начално седнали пътници.

Сложност: $O(M \log_2(N+M))$, където $M = b_Q$.

Решение на четвърта подзадача - 13 точки

Има различни подходи, възползвайки се от допълнителните ограничения в тази подзадача. Може би най-лесният е следният. Забелязваме, че пътниците последователно сядат на седалките с номера: $\lceil K/2 \rceil, \lceil K/4 \rceil, \lceil 3K/4 \rceil, \lceil K/8 \rceil, \lceil 3K/8 \rceil, \lceil 5K/8 \rceil, \lceil 7K/8 \rceil, \ldots$. Това е поради факта, че K е степен на двойката без 1 и рекурсивно винаги дупките са степени на двойката без 1. Можем да опишем процеса като използване на 1 дупка с големина K, използване на 2 дупки с големина $[K/2]$ от ляво надясно, използване на 4 дупки с големина $[K/4]$ от ляво надясно и т.н. По този начин лесно можем да разберем при дадена заявка b_i дупка с каква големина ще използваме и коя по ред от дупките с тази големина ще е тази дупка. Така ако използваме t-тата поредна дупка с големина $[K/2^s]$, то пътникът ще седне на седалка $\lceil (2t-1)K/2^s \rceil$.

Сложност: $O(Q \log_2 K)$.

Решение на пета подзадача - 33 точки (=13+20)

Тук трябва да направим сериозна стъпка към пълното решение, защото вече няма толкова лесни математически съображения като в предната подзадача при специалното K. Донякъде подходът за предната подзадача е правилният начин на мислене - да пробваме първо да открием големината на дупката, в която ще седне съответният пътник. Това може да изглежда като трудно нещо в общия случай, но като разписваме можем да направим следното наблюдение - броят възможни големини на дупки е сравнително малък. По-точно, той винаги е най-много $2\lceil \log_2 K \rceil$. Доказателството се основава на факта, че ако разглеждаме по нива дупките, които се получават, то когато имаме на ниво дупки с големина $2l$ и $2l+1$, то резултатните дупки в следващото ниво ще са с големини $l-1$ и l, т.е. пак две различни големини, които са съседни числа, и освен това 2 пъти по-малки. Аналогично, ако са с големини $2l+1$ и $2l+2$, то на следващото ниво ще имаме дупки с големини l и $l+1$. Затова всъщност нивата ще са $\lceil \log_2 K \rceil$, а на всяко ниво ще имаме по най-много 2 различни големини на дупки.

Вече е лесно да намерим при дадена заявка големината на дупката, която ще разделяме. Достатъчно е просто да симулираме процеса по образуване на дупките и да си пазим за всяка големина колко на брой дупки ще имаме. Трябва да внимаваме в края, защото големините на дупките по нива са различни с изключение на случая: $3, 4 \to 1, 2 \to 1$. След като имаме за всяка големина колко дупки някога ще се получат, обобщаваме тази информация в префиксен масив и с двоично търсене по него можем да намерим големината на дупката за заявката. Допълнително можем и да сметнем коя поред от дупките с тази големина ще е търсената. Нека означим намерената дължина с l.

Последната част е да намерим точно къде ще се намира тази дупка. Отново можем да се възползваме от малкия брой различни дължини и да пресметнем следното динамично: $dp[len] =$

броя дупки с големина l, които ще се получат от дупка с големина len. Базовите случаи са при $len < l$, когато $dp[len] = 0$ и при $len = l$, когато $dp[len] = 1$. В общия случай имаме следната зависимост:

$$dp[len] = \begin{cases} 2.dp[len/2] & \text{, ако } len \text{ е нечетно} \\ dp[len/2 - 1] + dp[len/2] & \text{, ако } len \text{ е четно} \end{cases}$$

След като сме сметнали динамичното е достатъчно да пуснем рекурсия, в която поддържаме интервала на текущата дупка и поредния номер на дупката с големина l, която търсим. Ръководим се от броя дупки с големина l, които ще се получат накрая от лявата и дясната част (спрямо разделянето на текущата дупка). Ако в ляво ще се получат поне толкова дупки, колкото е поредният номер, то трябва да отидем там, а иначе вдясно и намаляваме поредния номер.

Сложност: $O(Q \log_2 K)$.

Пълно решение - 75 точки

Трябва единствено малко да допълним решението, което направихме в предната подзадача. Понеже N вече не е 0, то не започваме с една голяма дупка, а с $N + 1$ на брой в общия случай. Сега всъщност трябва да пуснем алгоритъма за намиране на броя дупки от дадена големина за всяка начална дупка като е важно да запазваме и началния интервал, от който са произлезли дупките. Така ако сортираме получената информация първо по брой и след това по ляв край, можем директно да направим префиксния масив върху бройките и с двоичното търсене да намерим и от кой начален интервал ще е произлязла дупката, която търсим за дадената заявка. Останалата част е абсолютно същата, защото сведохме задачата при дадената заявка от много начални дупки до една начална.

Сложност: $O((N + Q) \log_2 K)$.

Интересно е, че по-лесната версия на задачата от пета подзадача, вече се е срещала в задача bench за група А от 2013 година. Решението от там не може да реши тази задача за 100 точки, защото то се възползва от факта, че няма предварително седнали хора.

Автор: Илиян Йорданов (zaeta)