**Анализ на задача Стена**

**Първа подзадача**. Когато камъните са дължина **1**, е достатъчно да намерим общото лице на смъкната стена. За удобство ще наричаме **N**-те нападения заявки. За всяка от тях обхождаме секторите в съответния интервал и добавяме към отговора общо смъкнатите метри.

Сложност – **O( N × S ).**

**Втора подзадача**. Щом стане дума за пресичане на интервали, няма как да не се замислим за техниката на помитащата права. Разделяме всяка заявка на две части – начало, намиращо се на позиция , и край, намиращ се на позиция . Във един масив от вектори **v[MAXS]** добавяме всички начала и краища и започваме да обхождаме векторите отляво надясно. В една променлива **h** поддържаме сринатите метри от текущия сектор. Обхождаме векторите за дадена позиция и съответно за всяко начало добавяме към **h**, а за всеки край вадим от **h**.

Сложност – **O( S ).**

**Трета подзадача**. Можем да подобрим идеята от предната подзадача. Вместо да имаме масив от вектори, ще имаме само един вектор, като всеки негов елемент съдържа три числа – **pos, h, type** – съответно позицията, сринатите метри и типа – начало/край. Сортираме го спрямо **pos** и отново го обхождаме отляво надясно, като този път вместо на всяка стъпка да добавяме към отговора **h**, добавяме **h×(pos - last\_pos)**, където **last\_pos** е позицията на предния елемент.

Сложност – **O( N × log(N) ).**

**Четвърта подзадача**. Ограниченията са достатъчно малки, за да попълним една таблица **S×H**, представяща стената. За всеки ред от нея намираме интервалите от празни пространства и добавяме необходимия брой камъни за всеки от тях към отговора.

Ако с **L** отбележим дължината на един от празните интервали, то броят необходими камъни за него би бил , където с **a%b** означаваме остатъка при деление на **a** с **b**. Ще го бележим с **F(L)** за по-кратко.

Сложност – **O( S × H ).**

**Пета подзадача**. Тук идва най-сложната част от задачата. За всеки сектор ще намерим следното: най-близките сектори отляво и отдясно със строго по-голяма височина (за удобство ще приемем, че секторите на позиции **0** и **S+1** са с височина **H**) и количеството камъни, необходимо да запълним нивото до по-ниския от двата най-близки по-високи сектора, ако приемем, че нивото до височината на текущия сектор вече е запълнено.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Да разгледаме петия сектор. Най-близкият сектор отляво по-голям от него е **1**, а отдясно – **7**. Приемаме, че сме запълнили пространството до височина **2** и се фокусираме до това на височина **min(height[1],height[7])=4**. Следователно сметката за текущия сектор ще е **F(7-1-1)×(4-2)**. Ако означим позициите на двата най-близки по-високи сектора с **L** и **R**, а височината на текущия с **h**, то в общия случай формулата изглежда така: **F(R-L-1)×(min(height[L],height[R])-h).**

Така един вид на всеки сектор съпоставяме участък от стената и сумата от отговорите би трябвало да е търсения брой камъни. Проблем биха създали секторите с еднаква височина, между които няма по-високи сектори (като **9**, **11** и **14**). И на трите съответства означения регион и така го броим три пъти. Лесно можем да се справим с проблема, ако за всеки сектор пазим и дали има такъв с височина равна на неговата след най-близкия му ляв по-висок сектор. В случай, че има такъв, просто го пропускаме.

Сложност – **O( N × S + S2 ).**

**Шеста подзадача**. Бавното в пета подзадача беше намирането на предишен и следващ по-висок сектор. Можем да ги намерим само с две обхождания с помощта на стек. За повече яснота може да погледнете задачата **report** от С група ЕТИ 2019. Тъй като заявките се отнасят само до един сектор, можем да ги считаме за константни.

Сложност – **O( N + S ).**

**Седма подзадача**. Комбинираме идеите от втора и шеста подзадача.

Сложност – **O( N + S ).**

**Осма подзадача**. Комбинираме идеите от трета и шеста подзадача.

Сложност – **O( N × log(N) ).**

Автор: Александър Гатев