АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА ЗАПЛАТИ

Във всяка итерация, която прави директорът има две основни стъпки: изчисляване на новите заплати и проверка дали се е променил статусът на служителите по отношение на заплащането. Разбира се, трябва да се провери и дали новата итерация е донесла нещо ново в заплатите, но това е лесно – изчислявайки новата заплата на всеки следващ служител, проверяваме дали тя се различава от старата и, ако има поне една разлика, то значи се е получило ново множество от заплати.

Първо да се заемем с изчисляването на новите заплати. Наивното решение е за всеки служител от списъка да преглеждаме последователно заплатите на колегите му с по-големи номера докато срещнем такъв с по-голяма заплата или стигнем до края на списъка, след което да пресметнем новата му заплата. Това решение е със сложност $O(N^2)$ и е твърде бавно при ограничение $N \leq 100000$.

За да получим по-бързо решение, първото нещо, което трябва да съобразим е, че е по-удачно да преглеждаме служителите в списъка ***от края към началото***. Нека масивът със заплатите е *sal*. Нека разглеждаме служител с номер *i*. Първо неговата заплата трябва да бъде сравнена с тази на съседа му отдясно, т.е. със заплатата на служител с номер *i+1*. Ако $sal[i+1] > sal[i]$, то изчисляваме новата заплата на служител *i* (запомняме я в друг масив, а не променяме стойността на $sal[i]$) и преминаваме към служител с номер *i-1*. Ако $sal[i+1] \leq sal[i]$, то трябва да продължим и да сравним $sal[i]$ със $sal[i+2]$. Ако, обаче, $sal[i+1] \geq sal[i+2]$, то няма никакъв смисъл да сравняваме $sal[i]$ със $sal[i+2]$. Следващото сравнение трябва да е между $sal[i]$ и $sal[j_1]$, където j_1 е първият индекс, по-голям от $i+1$, за който $sal[i+1] < sal[j_1]$. Ако $sal[i] < sal[j_1]$, то изчисляваме новата заплата на служител *i* и преминаваме към служител с номер *i-1*. Ако $sal[i] \geq sal[j_1]$, то, прилагайки същите разсъждения, търсим следващ индекс $j_2 > j_1$, такъв, че $sal[j_1] < sal[j2]$. И така нататък, докато не намерим такъв индекс j_k, за който $sal[i] < sal[j_k]$ или не свърши масивът. Въпросът е, как да поддържаме списък само на ония заплати, които ни трябват, когато разглеждаме служител с номер *i*, т.е. на $sal[i+1] < sal[j_1] < \ldots sal[j_k]$, и да преглеждаме само този списък. Ще използваме допълнителен масив *B*, който ще попълваме отзад напред. При разглеждането на служител *i* и неговата заплата $sal[i]$, в масива *B* ще се намират точно ония $sal[i+1] < sal[j_1] < \ldots sal[j_k]$, които ни трябват. Прилага се следният алгоритъм:

Нека в момента, в който разглеждаме $sal[i]$, масивът *B* е попълнен от индекс *N* до индекс *k*. Започвайки от индекс *k* надясно, в масива *B* се търси първият елемент $B[j]$, за който е изпълнено $B[j] > sal[i]$. Ако такъв елемент съществува, то се изчислява новата заплата на служител *i* по формулата $(sal[i]+B[j])/2$ (новата заплата се запомня в друг масив, а не се променя масивът *sal*), прави се $k=j-1$ и $B[k]=sal[i]$. Ако такъв елемент не съществува, то заплатата на служител *i* не се променя, прави се $k=N$ и $B[k]=sal[i]$.

Този алгоритъм е линеен. При всяко сравнение на $sal\ [i]$ с $B[j]$ или $sal\ [i]$ се вкарва в масива *B*, или $B[j]$ се премахва от *B*. По този начин всеки елемент от масива *sal* е съпътстван от едно сравнение, при което той се вкарва в масива *B*, и най-много едно,

при което се вади от масива *B*. Това значи, че сравненията са не повече от *2*N*, а сложността на тази част от алгоритъма е *O(N)*.

Проверката за запазване на статуса също може да се прави линейно по *N*. За целта да си представим, че сме образували групи с номера на служители, такива че във всяка група заплатите на служителите са равни. Запазването на статуса след преизчисляване на заплатите означава, че и при новите заплати групите остават същите (защото двама служители, които са имали равни заплати, продължават да имат равни заплати). Освен това отношението (< или >) между заплатите в групите се запазва. Тогава да използваме размерите на заплатите като индекси в масив от вектори. Във вектор с номер *j* записваме във възходящ ред номерата на служителите, които имат заплата с размер *j*. Тъй като заплатите са цели положителни числа и не надвишават 100000, а, освен това, във всичките вектори има записани общо *N* номера на служители, то това не води до използването на много памет. В един такъв масив от вектори записваме групите служители, съответстващи на началните заплати. След всяко преизчисляване, при което се е получило ново множество от заплати, формираме нов масив от вектори, съответстващ на групите от нови заплати. Линейно по *N* проверяваме дали групите са се запазили и дали са подредени в същия ред по размера на заплатите.

Максималният брой итерации може да бъде много голям (проиграйте следния пример с 11 служители със следните заплати: 310, 308, 306, 304, 302, 300, 302, 304, 306, 308, 310 и ще видите как се променят заплатите на служителите на всяка итерация. След това си представете порядъка на броя итерации, които ще трябват за тестов пример, в който заплатите са следните: 100000, 99998, 99996,....., 302, 300, 302,...., 99996, 99998, 100000). Но, тъй като директорът е много зает и малко суеверен, то ще се наложи да се направят най-много 12 итерации, което при линейния алгоритъм за всяка итерация с константа, приблизително равна на 4, прави решението с максимална сложност *O(N)* с константа, приблизително равна на 48.

Автор: Руско Шиков