АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА
Monopoly

Задачата Monopoly можеше да бъде формулирана учудващо кратко:

*"Дадена ви е редица от **N** цели (положителни и отрицателни) числа A_i. Намерете колко най-много последователни елементи от редицата можете да вземете, така че сумата им да е не по-голяма от **M**."*

В следствие на това имах известни притеснения, че може подобна (а дали не и същата?) задача да е давана на друго състезание. Поне доколкото ми е известно, обаче, не е (или поне аз не съм я срещал).
Съществуват редица различни начини, по които можем да подходим към задачата.

Наивно
Първият, относително тривиален, е за всяко възможно начало да започнем да удължаваме редицата надясно и на всяка стъпка да проверяваме дали сумата е в дадения ни лимит и ако да – да ъпдейтваме отговора. Това е елементарно за писане, но за съжаление е със сложност $O(N^2)$ и би хванало едва около 20 точки.

Умно наивно
Ако сме малко по-умни, можем да подобрим горното решение, като забележим, че ако досегашният най-добър отговор преди да разгледаме начало в индекс i е бил best, то няма нужда да разглеждаме интервалите [i, i], [i, i+1], …, [i, i+best-1] – те няма как да подобрят отговора. Вместо това можем (ползвайки префиксен масив) директно да започнем от [i, i+best]. Това дребно подобрение не променя сложността на решението, но все пак забележимо го забързва – вече точките, които бихме хванали, стават 40.

Двоично търсене
Някои състезатели сигурно интуитивно биха се насочили към двоично търсене. Наистина, задачата изглежда като такава – ако имаме фиксирана дължина на интервал len, то можем линейно да проверим дали тя е окей (просто правим плъзгащ се прозорец с размер len, като на всяка стъпка добавяме ново число в началото и махаме последното от края). За съжаление, обаче, това решение не е вярно, тъй като функцията не е монотонна (можем да имаме отговор с размер X, да нямаме с X+1, и отново да имаме с X+2. Прост пример би бил (-3, 1, 1, 1, 1, 1, 1, 1, -3) и M = 1. На втората стъпка на двоичното търсене (неправилно) бихме заключили, че отговорът е по-малък от 6 (тъй като всеки интервал с дължина 6 има сума поне 2). Отговор обаче има, при това с всички числа! Все пак, понякога двоичното търсене работи, и това решение би хванало 55 точки (с малко подобрения – дори повече).

Бъкети
Следващото решение, което ще разгледаме, е базирано на бъкети. Нека разбием входните числа на групи от по приблизително sqrt(N) числа. Можем да намерим сумата на всички числа във всеки бъкет и да ги запазим в масив. Всеки валиден отговор ще включва последните няколко (потенциално нула или всички) числа от един бъкет, целите следващи няколко бъкета, и няколко (потенциално нула или всички) от още един бъкет. Тъй като имаме сумите на целите бъкети, можем да сметнем сумата на числата в междинните (със сигурност цели) бъкети много бързо (дори константно, макар и това да не е нужно). Така ни интересува единствено колко числа да вземем от най-левия бъкет и колко от най-десния. Нека направим следното:

За най-левия избран бъкет, от дясно наляво правим стек, в който пазим сумата на всички числа от най-дясното до текущото, както и индекса, в който се намираме. Ако сумата стане по-малка от върха на стека, махаме от него, докато или се изпразни, или върхът му стане с по-малка сума. Така в стека получаваме различни под-интервали, свършващи в последния елемент на най-левия избран бъкет, сортирани по сума. Лесно можем да видим, че интервалите, които сме извадили от стека, не ни интересуват (те са по-неоптимални, щом сме ги извадили).
Правим нещо подобно и за най-десния избран бъкет, само че от ляво надясно, като под-интервалите започват от най-лявото число в този бъкет и свършват в различни позиции.
Така, след като имаме тези два стека (впрочем, имплементационно е по-удобно да ползваме вектори или двустранни опашки) в началото включваме най-дългия под-интервал от левия стек и най-късия от десния. След това започваме да разглеждаме все по-къси под-интервали от левия стек и все по-дълги от десния. Реално, разглеждаме под-интервали от началната редица, с все по-десни начала и краища.
Както строенето на тези стекове, така и последвалото им обхождане, са операции със сложност O(sqrt(N)) – тъй като такъв е размерът на бъкетите. Това ни се налага да правим за всяка двойка бъкети. Тъй като броят бъкети е също sqrt(N), то цялата сложност на това решение е $O(sqrt^3(N))$, или O(N * sqrt(N)). То би хванало около 80 точки.

Умно решение
Все пак можем да сме дори по-умни, използвайки горната идея със стековете. Вместо да ползваме бъкети, обаче, ще ползваме следното наблюдение. Ако знаем индекса на един (произволен) елемент от оптималната подредица (отговора), то можем да го намерим за O(N). Реално ще построим пак тези два стека, само че започвайки от елемента, който знаем, че е включен в редицата. Вече нямаме бъкети, така че стековете могат да станат с по (до) N елемента, откъдето идва и линейната сложност.

Окей де, но ние НЕ знаем елемент от отговора.
Все пак можем хитро да намерим такъв! Нека допуснем, че отговорът е с дължина между N / 2 и N. Тогава елементът с индекс N / 2 със сигурност ще участва в отговора! Следователно, започвайки от него, можем да го намерим. Ако, започвайки от него, най-добрият отговор, който намерим, е с размер по-малък от N / 2, то тогава би могло елементът на индекс N / 2 да не участва в глобалния отговор. Затова, в този случай, ще пробваме индексите N * 1 / 4 и N * 3 / 4, като търсим отговор с размер между N / 4 и N / 2. Ако намерим такъв – супер. Ако не, намаляме още размера и пробваме начални индекси N * 1 / 8, N * 3 / 8, N * 5 / 8, и N * 7 / 8, като търсим отговор с размер между N / 8 и N / 4. Продължавайки в същия дух рано или късно ще стигнем до размера на отговора и ще го намерим.
Каква е сложността на това решение? На първата стъпка правим N операции за 1 елемент – тоест O(N). На втората стъпка правим N / 2 операции за 2 елемента, тоест отново O(N). На третата стъпка правим N / 4 операции за 4 елемента – отново O(N) (и така нататък). Тъй като на всяка стъпка намаляме размера на половина, то имаме най-много логаритъм на брой стъпки, всяка със сложност O(N). Така сложността на цялото решение става O(N * logN). Това решение хваща 100 точки.
Други решения?
Подозирам, че може да има и други решения със сложност O(N * logN), а дори и линейни. Поне за момента, обаче, не съм се сетил за такива.

Автор: Александър Георгиев