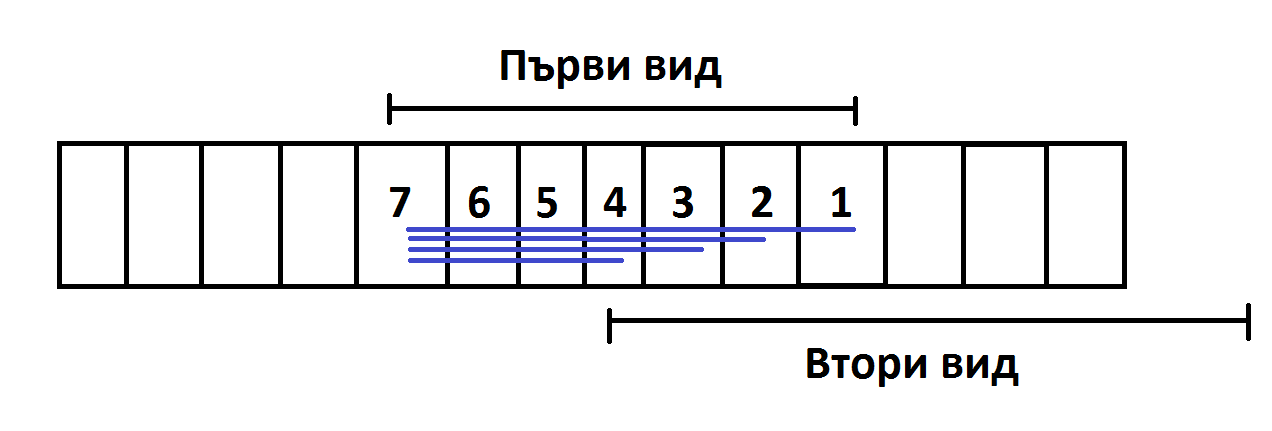
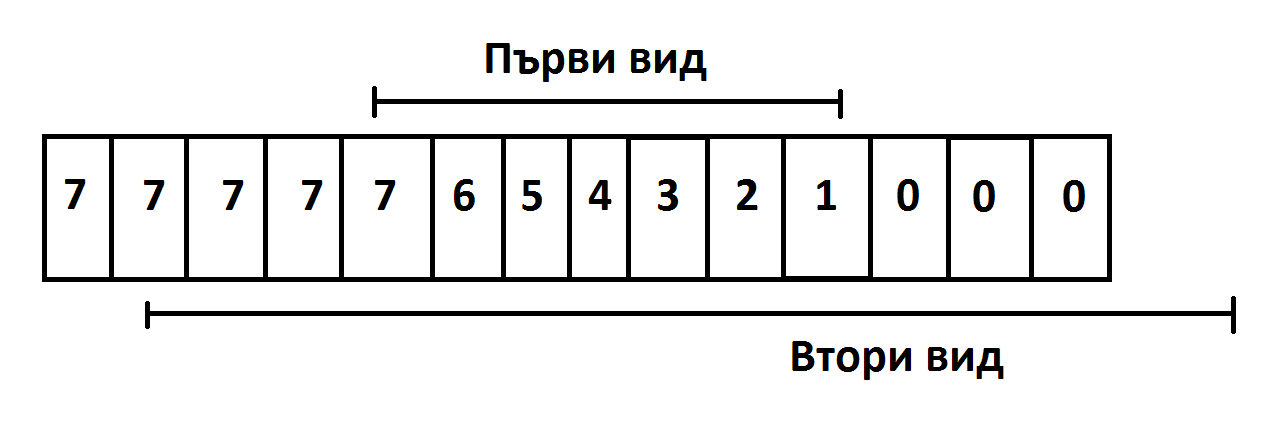
Нека се опитаме да решим задачата с интервално дърво. Какво трябва да пази то? Нека то е построено по координатите на отсечките и пази суми. Какви суми? Ами нека разгледаме някоя координата x. Представяме си, че имаме подотсечка от вид 2, която има ляв край x. Кой е десния й края? Не ни интересува. Сега за листото, отговарящо на x, ще пазим броя подотсечки от вид 1, които ще се пресекат с нашата подотсечка от вид 2 с ляв край x. Тук разбираме, че десния край наистина няма значение, поради това как определяме отсечките от първи и втори вид – можем да си представим, че е безкрайно надясно. Ако още не сме убедени в това, да разгледаме картинка със стойностите в листата на интервалното дърво след като „поставим“ някоя отсечка от първи вид вътре (забележете, че това е цялата отсечка, а не индуцирана подотсечка).

Ето, например, ако x е координатата със стойност 4, то тези 4 подотсечки, които сме преброили, са отсечките във синьо. Тук отново можем да забележим, че десният край на долната отсечка не играе роля.

Сега да видим какво ще се случи, ако левия край на долната подотсечка е вляво от левия край на горната отсечка. За да работи същата логика, трябва да добавим стойности вляво от седмицата, а именно още седмици.



Тук може би забелязахте, че тайно сложихме и нули вдясно. Тяхната обосновка е, че ако долната подотсечка свършва след десния край на горната, то те нямат общи точки.

Защо правихме всичко това? Защото сега, ако получим отсечка от втори вид, то отговора за нея е сумата в интервала, който тя покрива.

Вече сме почти готови да оставим настрана картинките и да се съсредоточим върху имплементацията на интервалното дърво. Последният детайл е, че не винаги можем да игнорираме десния край на долната подотсечката. Частният случай, който досега пропускахме, е че отсечката от втори вид може да е изцяло отляво спрямо отсечката от първи вид и тогава отговорът трябва да е 0. Звучи неприятно, защо просто не се отървем от този случай? Това ще стане като сортираме първите отсечки по ляв край и вторите отсечки по десен край. Така ще можем да обхождаме отсечките от втори вид (за да намерим отговорите за всяка от тях) по такъв начин, че всички досега разгледани отсечки от първи вид да НЕ са изцяло вдясно.

А сега имплементацията на интервалното дърво… То трябва да поддържа добавяне на сума в интервал във вид на аритметична прогресия (вътре в отсечката) и в константен вид (наляво от отсечката). Търсенето отново се осъществява в интервал.

Сложността на алгоритъма е O((N+M)log(N+M) + (N+M)logMAX\_C), където MAX\_C е максималната координата.