**Анализ**

*Тагове: псевдо-интерактивна задача, двоично търсене, две показалки, поддържане на минимум с дек*

Задачата е от така наречените псевдо-интерактивни задачи. Интерактивна е, защото трябва да се пишат функции, а не цяла програма. А е псевдо, защото това е направено единствено поради причината да може времето за вход и изход да не се включват в цялото време за измерване, т.е. по същество не е истинска интерактивна задача. По принцип до такъв подход се прибягва и за това заявките да се поддържат онлайн, но е ясно, че за тази задача заявките не са централна част. Освен това, за да е по-бързо изчисляването и да няма големи входни файлове, грейдърът генерира тестовите данни по време на изпълнение, ползвайки зададени параметри от входните файлове. След като се направят няколко ключови наблюдения се получава доста интересно решение, което бидейки линейно, не може да се различи добре от „ен-лог“ решение, ако се включва и четенето на входа.

Първата подзадача е за 11 точки. Предвидена е за пълното изчерпване. Все пак трябва да направим някои прости наблюдения. Първото е, че оптималният маршрут се състои от ходене до някоя стая вляво, връщане в стаята за квесторуване (*k*) и отиване до някоя стая вдясно и отново връщане до началната стая. Евентуално лявата или дясната част може да ги няма за оптимален маршрут. Това означава, че пътят ни винаги е с четна дължина, така че нямаме дробни числа при смятане на крайния отговор. Всъщност дори отговорът, който трябва да върнем, така ще се получи директно като намерим най-лявата и най-дясната точка от оптималния маршрут. Това е достатъчно за тази подзадача. Можем да разгледаме всички възможности за левия и десен край (разбира се, които съдържат началната стая *k*) и да видим за кои от тях, броя деца, през които ще се мине, е ***M*** или повече. От тях трябва да подберем най-краткия път. Ясно е, че ако например с един цикъл фиксираме левия край, то с другия можем просто да увеличаваме десния край и да изчисляваме съответните суми. Сложността на такова решение е .

Втората подзадача е за 17 точки. Сега ще оптимизираме по стандартен начин горното решение. Наличието на заявки не трябва да ни притеснява, понеже те са много малко. Нека отново фиксираме ляв край в един цикъл. Лесно наблюдение е, че понеже искаме да минимизираме цялото разстояние, то бихме искали десния край да е възможно най-близко. Това означава, че ще искаме да изберем най-близкия десен край, където сумата по децата е ≥ ***M***. Понеже имаме само положителни числа при фиксиран ляв край, можем да направим двоично търсене за намиране на десния край. Ще вземем за десен край, първото място (ако има), където сумата по децата става ≥ ***M***. Така може да разгледаме всички възможности за ляв край ≤ *k* и съответно за всяка да получим по една възможност за десен край. От тези ще намерим минималната сума на разстояние между двата края. Това решение има един проблем, който не е много очевиден. Той важи и за следващите решения. Възможно е да пропуснем оптималния откъм разстояние маршрут по тази схема. Ясно е, че ако намаляваме последователно левия край, то ще стигнем случай, където потенциалния десен край вече ще е преди *k*-та позиция. Ако прекъснем търсенето може да изпуснем една възможност. Това е оптималния маршрут да ходи само наляво точно до тази позиция (където е ясно, че сумата по децата ще е ≥ ***M***). Сложността тук е доста по-добра от преди е .

Третата подзадача е за 13 точки. Понеже броя заявки в ограниченията не са се увеличили, ясно е, че трябва да си подобрим още предното решение, т.е. да го направим линейно. За добрите състезатели това не би трябвало да е голям проблем. Лесно се забелязва, че при намаляване на фиксирания ляв край, десния също се движи малко по-малко наляво. Така можем да използваме техниката с две показалки. По този начин няма да следим десния край с двоично търсене, а просто ще намаляваме старата позиция, докато сумата на децата е все още ≥ ***M***. Отново имаме същия частен случай като в предната подзадача. Очакваната сложност за тази подзадача е .

Четвъртата подзадача е за 18 точки. Тя е съществена стъпка към пълното решение. Идеята от сега нататък ще е да преизчислим за всяка възможна стая, от която можем да започнем, какъв е оптималният отговор. Трябва да погледнем предното решение от друг ъгъл. Общо взето, почти не разглеждахме стаята, от която тръгваме. Ако имаме фиксиран ляв край, то съответния му десен се определя само от стойността на ***M***. Това означава, че разстоянието за този интервал е потенциален отговор за всички стаи, които се включват в интервала. Освен това за всяка стая оптималния маршрут ще е някакъв интервал, получен по този начин, с малко изключение – частния случай от предните подзадачи, когато стаята, от която тръгваме се явява десен край при оптимален отговор. Този частен случай лесно може да се обработи, като за всяка стая се намери доколко наляво трябва да ходим за да получим сума по децата ≥ ***M***, например с двоично търсене. Така остана да изясним трудната част – как ще отчитаме потенциалните отговори, които идват от интервалите. Начинът, по който можем да намираме самите интервали, е чрез техниката с две показалки, по подобен начин като в миналата подзадача (за тази подзадача би трябвало да се хваща, ако се прави и с двоично търсене). Най-стандартният начин е може би следният. Пазим в един *multiset* потенциалните отговори. Когато намерим нов интервал, добавяме стойността в *multiset*-а и вземаме минимума за текущата позиция. Понеже тези стойности са активни до някакво време, то най-лесно е да пазим за всяка позиция кои стойности трябва да се махнат и така преди да минем към следваща стъпка да изчистим стойностите, които спират да бъдат активни. Сложността на такова решение е . Ползване на *multiset* придава голяма константа на решението, затова то може да хване само тази и първа подзадача и решението за следващата подзадача, макар със същата сложност е доста по-бързо.

Петата подзадача е за 12 точки. Друг стандартен начин, за да отчитаме отговорите е чрез сегментно дърво. Можем да построим сегментно дърво върху масива от стаите и всяка заявка да приемаме за ъпдейт в сегментното дърво. Няма нужда от lazy propagation. Достатъчно е просто да забиваме даденото число като потенциален минимум за върховете на сегментното дърво спрямо заявката (съответно ако за някой връх се получат няколко стойности, вземаме минималната от тях). След като намерим всички интервали и направим съответните заявки в дървото е достатъчно да обходим цялото дърво, като си пазим минимума по пътя, за да може като достигнем до листата (съответстващи на стаи) да го отчетем като потенциален отговор за съответната стая. Другата стойност, която трябва да гледаме за всяка стая, е от частния случай. По този начин сравнително лесно можем да намерим нужното за всяка стая и да отговаряме константно на всяка заявка. Така сложността става .

Последната подзадача е за 29 точки. Тук ще оптимизираме преизчислението, за да стане с линейна сложност. Едно от нещата, които лесно можем да подобрим е при частния случай. Ясно е, че тези отговори можем да ги намираме отново с подхода с двете показалки, като тръгнем отдясно-наляво. Другата част е по-интересна. Разбира се, ще се отървем от сегментното дърво. Всъщност, решението няма да използва някоя сложна техника. Имаме следната ситуация – търсим минимум за всяка стая, като потенциалните отговори идват за цели интервали, а самите интервали са получени вследствие от техниката с показалките. Това означава, че те са специфични – няма интервал, който се намира вътре в друг интервал. Съответно те ни задават че някой потенциален отговор започва да действа от някоя позиция и спира да действа в някоя позиция. Това означава, че стойностите, които са активни, могат да се поддържат със сравнително известната техника за поддържане на минимуми с дек. Малко обосновка защо можем да го направим. Първо, когато намерим нов потенциален отговор, то можем да махнем някои стари, които са по-големи, защото те никога няма да бъдат използвани вече. Освен това, понеже стойностите стават неактивни в нарастващ ред на времето на постъпване, то можем да махаме неактивни стойности от върха на дека. Затова трябва да използваме и дек – имаме добавяне в края и махане от двата края. Окончателната сложност за задачата е .

*Начална идея: Велислав Гърков*  
*Автор: Илиян Йорданов*