Анализ

Първото важно наблюдение е, че за избрано множество от хора, които ще седнат, оптималното им разположение е да се оставят празните седалки на последователни позиции и да се поставят двамата с най-голямо Б в двата края. Разположението на останалите няма значение.

(P, P, P, maxB1, __, __, __, __, maxB2, P, P, P ; където P=пътник, __=празно място, maxB1/2 = пътници с първо/второ максимално B)

Последователните празни места ще наричам „дупка“.

1. **Частично решение за *O(N*N*L)***

   Нека си фиксираме кои ще са двамата, които ще сложим в двата края на „дупката“ и каква ще е големината ѝ. Тогава ни остава само да избираме хората с най-голямо A докато запълним останалите места. За да стане това по-бързо можем да сортираме всички пътници по A и след *O(N)* преизчисление ще можем да намираме сумата на A-тата на последните *K* пътници константно.

   И в останалите решения се сортират пътниците по нарастване на A, затова ще приемам, че в началото на алгоритъма това е направено.

2. **Частично решение за *O(N*N)***

   Нека сме си фиксирали колко пътници ще седнат и нека този брой е *K*. Тогава има 3 възможни конфигурации:

   A) Ще изберем *K* пътника с най-големи A-та и ще поставим тези с най-големи B-та в края на „дупката“.

   Б) Ще изберем (*K-1*) пътника с най-големи A-та и ще изберем от останалите пътници този, който ще увеличи крайната сума възможно най-много (този с максимално A+ B*(*L-K*)). Тогава в двата края на „дупката“ ще са този с максимално B от първите (*K-1*) и този с максимално A+B*(*L-K*) от останалите.

   В) Ще поставим (*K-2*) пътници с най-големи A-та и ще изберем 2-ма от останалите, които ще сложим в края на „дупката“. Аналогично на Б), търсим пътниците които дават максимално A+B*(*L-K*).

   Пътниците, които ще слагаме в краищата на „дупката“ във всеки от тези случаи можем да намерим с обикновено линейно обхождане на масива с пътници.

   Пълно решение *O(N*logN + N*20)*

   Отново както в 2. ще обхождаме всички стойности на *K* – броя на седналите пътници.

   Сега обаче искаме да намираме по-бързо кои са 2-мата пътника, които ще сложим в края на „дупката“. Можем да намираме сумата на останалите (*K-2*)-ма константно, тъй като те ще са някои от първите *K* (сортирани по нарастване на A). Тогава ако знаем сумата на A-тата на първите *K* и знаем кои точно ще са в краищата на „дупката“, можем лесно да намерим сумата на A-тата на тези, които няма да са в краищата на „дупката“. За да стане изчислението константно, в началото можем да сметнем сумата на A-тата на първите *I* пътника, за всяко *I*.

A) Тук избираме пътници с максимално B. Търсенето им може да се оптимизира като още в началото сметнем кои са 2-мата пътника с максимално B от първите *I* пътника, за всяко *I*. Така намирането им става константно

И в Б) и във В) търсим пътници, които дават максимално A+B*(*L-K*). Забелязваме, че това прилича на уравнение на права, затова ще използваме Upper/Lower Envelope (известно още като Convex Hull Trick).

Upper/Lower Envelope работи по следният начин:

Нека имаме *N* прави дефинирани с уравнение $y = B_i * x + A_i$ (ъгловият коефициент е означен с B_i, за да е със същите означения като в оригиналната задача). Искаме да намерим списък от прави, които за някое *X* са с максимален *Y*. Очевидно за някоя права тези *X*-ове, на които тя е с максимално *Y*, са в затворен интервал. Искаме също така списъка от прави да е подреден по нарастване на координатите на интервалите, в които те са максимални. (нека права *I* е максимална в интервала *Pi, Qi*, а права *J* –в интервала *Pj, Qj*. Тогава *I* < *J* ⇔ (*Pi, Qi*) < (*Pj, Qj*))

Нека подредим правите по намаляване на свободния член. Ясно е, че първата права ще участва в търсения списък и също така ще е на първо място там. Ще итерираме правите последователно и ще ги слагаме в списъка и евентуално ще премахваме правите в края на списъка. Нека последните 2 прави в списъка означим с *L1*, *L2*, а сега разглежданата права с P. Нека X*(A,B)* е функция, която връща *X*-координатата на пресечната точка на А и Б. Ще махнем последната права в списъка(*L2*), ако X(L1, P) < X(L1, L2). С това неравенство разбираме, че правата L2 не би била максимална в нито един момент, защото правите P и L1 ще са над нея. Премахваме L2 от списъка и продължаваме да правим същата проверка за последните 2 прави от променения списък. Когато приключим с премахването на прави добавяме правата P в края на списъка. Сега, когато вече сме конструирали списъка, можем по-бързо да отговаряме на запитвания от вида „Колко е най-големият Y на някоя права за конкретен X“.

Да се върнем към нашата задача.

За Б) ни трябваше пътника с максимално A + B*(L-K). Тук (L-K) ни се явява X-а в геометричната репрезентация. Ако разглеждаме K-тата в намаляващ ред, и, преди всяко търсене за фиксирано K, добавяме правата(пътника) с номер K, тогава ще имаме построен Upper Envelop-а на последните (N-K) пътника. Сега можем с двоично търсене да намерим коя е правата, с максимално Y в координата (L-K). За В) нещата стават малко по-сложни, защото тогава ни трябват 2-те прави с максимални У в дадена координата X. За целта ще строим 2 Upper Envelop-а едновременно и всеки пътник ще добавяме в произволен от 2-та. За 2-та пътника в края на „дупката“ ще взимаме максималните прави в 2-та Envelop-а. Шансът при такова разпределение да не намерим оптималните 2-ма пътници е 1:2 – това ще се случи само ако и двамата са в един и същи Envelope. Ако изпълним този алгоритъм P пъти, шансът да не сме намерили оптималното решение е (1/2)^P. При P около 20, този шанс е достатъчно малък.

Относно търсенето на оптималния пътник в Envelop-а – ако веднъж сме намерили къде е той за някое X, то следващото търсене ще е за X+1. Тогава не е необходимо да се прави двоично търсене, за да се намери отговора, а може да се тръгне от последния намерен и да започнем „да се изкачваме“ (отиваме в посоката, в която се увеличават У-ците). В крайна сметка ще сме минали през всяка права по веднъж => O(N).

И тъй като ще изпълним този алгоритъм 20-тина пъти => O(20*N).

Автор: Виктор Терзиев