**Подзадача 1:**

За тази подзадача е предвидено решение със сложност с пълно изчерпване, за всяка точка пробваме двата варианта: да е в или да е в . За всяко разпределение смятаме търсената сума и връщаме минимума от откритите възможности. Както и във всички по-нататъшни решения, трябва да внимаваме и , и да са непразни.

**Подзадача 2:**

За тази подзадача е предвидено решение със сложност . Нужно е да направим наблюдението, че винаги има оптимално решение, в което множеството се състои от непрекъсната редица точки (когато те са сортирани по , както е във входа), а се състои от всички точки преди тази редица и всички точки след нея, т.е. от даден префикс и даден суфикс от точки. Има възможни начина да фиксираме двата края на и за всеки от тях линейно смятаме стойността на търсената сума.

**Подзадача 3:**

За тази подзадача е предвидено решение със сложност . Трябва да видим, че можем да смятаме широчината на за дадени лява и дясна граница в , като акумилираме минимума и максимума в докато местим дясната граница (а ги инициализираме, когато преместим лявата граница). В същото време можем да акумилираме минимума и максимума в префиксната част на докато местим лявата граница, а за суфиксната му част можем да прекомпютнем всички суфиксни минимуми и максимуми на -ци от всяка позиция нататък. Така за всички леви и десни граница смятаме стойността на търсената сума в .

**Подзадача 4:**

За тази подзадача са предвидени всякакви решения със сложност . Такива са разгледани на идейно ниво, но тъй като никое реално не е по-лесно от решението със сложност , няма имплементирано такова авторово решение.

**Подзадача 5:**

За тази подзадача е предвидено решение със сложност . Изисква се да забележим, че за всяка комбинация от лява и дясна граница сме в един от четири случая: 1) и минимума, и максимума на са в суфикса; 2) минимума е в префикса, а максимума в суфикса; 3) минимума е в суфикса, а максимума в префикса; 4) и минимума, и максимума са в префикса. Сега нека разглеждаме фиксирана дясна граница . Лесно можем с двоично търсене да открием, за коя стойност на лявата граница се сменя къде е минимума и за коя ѝ стойност се сменя къде е максимума. Нека това са и и нека приемем, че . Т.е. за леви граници в сме в случай 1), за в сме в случай 2), а за в сме в случай 4). (Ако средният случай става случай 3).) Сега за всеки от четирите възможни случая, можем да разделим стойността на търсената сума на два компонента: членове зависещи само от и членове зависещи само . За фиксирано първият компонент е константен, а вторият е променлив. Т.е. за всеки от трите интервала, търсим това в него, което минимизира вторият компонент на съответния случай. Това всъщност е RMQ (Range Minimum Query) и за тази задача най-удачно е да го правим със сегментно дърво. (Всъщност лесно се вижда, че за случай 1) не ни трябва сегментно дърво, защото отговорът винаги е най-десният елемент в интервала.)

**Подзадача 6:**

За тази подзадача е предвидено решение със сложност . Основната идея за решението е същата както това за предната подзадача. Нека първо видим как да намираме и в за дадено . Всъщност можем да видим, че суфиксният минимум намалява като намалява, а префиксният максимум намалява като се увеличава, т.е. можем да използваме two pointers: започваме с най-вдясно и го местим наляво, като на всяка стъпка местим наляво докато префиксният минимум не достигне суфиксния такъв. Еквивалентно и с . Естествено , т.е. трябва да ги намаляваме с по едно на всяка стъпка ако са достигнали .

Сега да видим как да отговаряме на RMQ-тата в . Естествено, на теория можем да използваме подход през LCA с Cartesian Trees, но това е доста бавно като константа и доста неприятно за писане. Вместо това предлагаме два по-приятни подхода. Първият се базира на това, че заявките са „горе-долу монотонни“ надясно, т.е. в общи линии и левите им краища, и десните им краища се движат само надясно. Ако това винаги беше вярно, можеше да използваме трик за монотнно RMQ с deque. Тъй като обаче и и започват да се движат наляво, когато достигнат , първо ще си прекомпютнем стойностите на и за всяка стойност на и ще видим кога първо се срещат. Нека тези точки са и и нека и . Сега можем да забележим, че RMQ-тата за случаи 2) и 3) са монотонни отляво на , а тези за случай 4) – отляво на . От друга левите краища на нито едно трите вида RMQ-та никога не надвишават и съответно. Това ни позволява да разделим всяко куери на две подкуерита: едно отляво на съответното и едно отдясно на него. Лявото подкуери си е точно монотонно RMQ, на което отговаряме с deque, а дясното е просто минимум от съответното до съответното , т.е. префиксен минимум от -то, всички от които можем да си прекомпютнем. Така отговаряме на всяко от шестте подкуерита в , комбинираме ги по двойки в отговори на трите желани куерита и от там процедираме както в решението на предната подзадача.

Друг подход със сложност се възползва от това, че на RMQ-тата не е нужно да бъде отговаряно онлайн, т.е. можем първо да си генерираме всички куерита по случаи (като за всяко запазваме какво трябва да направим с резултата му) и после да им отговорим в друг, по-удобен ред. Един много лесен алгоритъм за RMQ, който се възползва от това, е такъв базиран на Union-Find/Disjoint-Set Union. Детайлите му може да намерите онлайн или в авторовото решение.

Автор: Емил Инджев

Начална идея: Радостин Чонев