**Анализ**

*Тагове: сума от брой делители, амортизационен анализ, предпроцесване, RMQ*

Първата подзадача е за 13 точки. Тя е за наивното решение. Единственото, което може да спре състезателите да я вземат, е да се уплашат от по-високите ограничения. Но в действителност, нека анализираме каква е сложността, с която смятаме сумите за всяко зайче за едно предположение. Трябва да минем през всички числа за сумата на първото зайче, след това през тези на четни позиции за второто зайче и т.н. Броят числа, през които трябва да минем общо е . Сумата от обикновените дроби, известна още като сума на хармоничния ред, е много изучавана и всъщност е приблизително , така че броят числа за смятане на сумите на зайчетата е около – наистина не много голям. Този факт можем да видим и по друг начин – известно е, че сложността на решето на Ератостен без оптимизациите е , а на нас ни трябва точно това. С други думи за тази подзадача достатъчно е за всяко предположение да намираме сумите по наивния начин и от тях да вземаме максималната. Сложността е .

Втората подзадача е за 21 точки. Тук просто трябва да оптимизираме малко горното решение. Независимите предположения всъщност не променят много нещата. Единствено част от сумите се изменят с *s*. По-точно единствено сумите на зайчетата с номера, които са делители на *p*, се променят. Лесно може да видим, че не е нужно да смятаме сумите всеки път. Може да направим един предпроцесинг, в който да ги сметнем предварително. Така за всяко предположение просто обхождаме сметнатите суми и намираме максималната, като вземем предвид, че сумите на зайчетата с номера делители на *p* се изменят с *s* за предположение *p s*. Очакваната сложност е .

Третата подзадача е за 25 точки. Вече се доближаваме до пълното решение. Всъщност задачата е главно за подходящо предпроцесване, съответно в тази подзадача ще направим още една стъпка в тази посока. В миналия абзац казахме, че предварително знаем промяната за дадена позиция *p* върху кои суми ще се отрази. На нас ни трябва максимум, така че най-разумно е след като сметнем сумите да пресметнем за всяка позиция *p* коя е максималната сума, която би се променила. Ще означаваме тази стойност с *max[p]*. За да направим това бързо, когато смятаме сумите, можем да намерим делителите на всяко число. Това е едно от класическите разширения на решето на Ератостен. Отново, използвайки началните разсъждения, се вижда, че това не е бавно и сумарната работа е най-много . Допълнително, нека сметнем началния максимум на всички суми *max*. Сега да се върнем на предположенията. Ще се възползваме от това, че промените са неотрицателни. Да помислим какъв е търсеният максимум. Единият вариант е старият максимум *max* да се увеличи, защото е за зайче с номер, който дели *p*. Другият вариант е старият максимум да не се промени, но тогава е възможно някоя от нарастващите суми да стане по-голяма от него. Така всъщност е достатъчно единствено да намерим по-голямото от *max[p]* + *s* и *max*. Сложността тук е .

Четвъртата подзадача е за 30 точки. Тази подзадача е и за по-бавни пълни решения. Ясно е, че горният подход не работи, ако промените в сладостта могат да са отрицателни. Проблемът е в случая, когато максимумът на всички суми е за зайче с номер, който дели *p*, защото ако намалее тази сума не е ясно тогава коя ще е максималната. Освен сумите за *max[p]*, при предположение за позиция *p* ни трябва максимумът от сумите за номера, които не делят *p* (не участват в намирането на *max[p]*). Те вече са голям брой и трябва да сметнем този максимум малко по-умно. Тук вече има различни начини. Най-лесният вариант е да използваме *multiset* за всички суми, от който да махнем сумите от *max[p]*, за да намерим максимума на останалите. Този подход обаче е доста по-бавен от другите идеи и не е предвидено да хваща тази подзадача. Може би най-директният начин е да използваме *RMQ*. Нека означим сумите на зайчетата със *sums* и за дадена позиция означим делителите с . Тогава максимумът на сумите на зайчетата с номера, които не са делителите, всъщност е най-големият максимум от тези на следните подмасиви: . Такава задача директно се решава с *RMQ*, като ако се възползваме от това, че знаем сумите предварително и те реално не се променят (защото предположенията са независими), то е достатъчно да реализираме статично *RMQ* със *sparse table* и така заявката за максимум в интервал да е константна. Ще отбележим, че по този начин намирането на отговора при предположение може да става и онлайн, което е предвиденото за тази подзадача. Сложността на такова решение е . (реално броят делители на число ***n*** е ограничен отгоре от , но всъщност за числа до 1018 важи и указаната оценка)

Петата подзадача е за 11 точки. Единият начин, е който описахме, като трябва всичко да извършим като предпроцесинг. Друг хитър начин за намирането на максимума на останалите суми е следният. Нека сортираме всички суми в намаляващ ред. Тогава просто започваме да ги обхождаме и когато намерим сума, която не е за номер, който дели *p*, то сме намерили търсеният максимум. Защо това не е бавно? Ами в най-лошия случай ще обходим всички суми за делители на *p*, докато намерим исканото. Така сумарната работа за намирането на максимумите за всяко *p* ще е . Крайната сложност е .

*Идея и условие: Антон Чернев  
Реализация: Илиян Йорданов*