**Анализ на задача „Кучешка храна”**

Първата идея, с която може да се атакува задачата, е за всяко въведено **xi** да пресмятаме всички полиноми и да броим тези от тях, които имат положителна стойност. При дадените ограничения обаче може да се предположи, че това не е особено оптимално – всъщност това е решението, което хваща 20-те процента от тестовете, в които F, Q ≤ 1000.

Ключов момент в задачата е, че функциите, които ни се задават, не са произволни – те са полиноми от най-много трета степен. При анализ на поведението им може да се забележи, че всяка от тях е положителна в 0, 1 или най-много 2 интервала от реални числа. Всеки от тези интервали е ограничен от корените на полинома, или от . Следователно ако намерим корените на всеки от полиномите, можем да се абстрахираме от самите полиноми и да разглеждаме задачата като задача, боравеща с интервали – имаме определен брой интервали, и определен брой запитвания от вида „на колко интервала принадлежи даден x”.

А как се решава задачата с интервали? Има няколко различни ефективни начина – някои по-лесни за имплементация, други – по-трудни.

1. Интервално/индексно дърво – ако не се обмисли добре каква функционалност се изисква за решаването на задачата, при думата „интервал” някои хора биха се втурнали направо към тежката артилерия – двоичните дървета. Наистина, това би решило задачата без особени затруднения. Единственото по-необичайно нещо при конструирането на дървото е работата с реални числа. Този проблем обаче има лесно решение: след като знаем началото и края на всеки от интервалите, можем просто на всяко начало/край да съпоставим естествено число, съответстващо на това колко от останалите краища са по-малки от него. Тъй като нямаме динамична промяна на функциите, това е напълно достатъчно. Следва стандартна имплементация на двоично дърво, която е извън рамките на този анализ.

Сложност за отговор на единична заявка – **О(log F)**.

1. Двоично търсене – сортираме всички събития (като под събитие разбираме начало или край на интервал – т.е. кога един полином започва да бъде положителен и кога спира да бъде) по големина, като за всяко събитие пазим информация за това колко интервала са отворени непосредствено след момента, в което то става активно – сложност **O(F \* log F)** за сортиране, **O(F)** за събиране на нужната информация. След това за всяко запитване можем да направим двоично търсене, с което да намерим най-големият елемент, който е по-малък от дадения – броят интервали, които са отворени за него, ще ни даде и броя полиноми, които приемат положителна стойност при текущата заявка. За да намери този елемент, авторското решение използва стандартната функция lower\_bound (която всъщност е реализирана чрез двоично търсене) – тя намира най-малкия елемент, по-голям от дадения, т.е. елемента след този, който търсим.

Сложност за отговор на единична заявка – **O(log F)**.

1. Sweep line – тъй като задачата не изисква моментален отговор за всяка заявка, можем подобно на предния подход да сортираме всички събития, но този път да приемем за събития и **x**-овете от заявките. По този начин когато събираме нужната информация, автоматично ще съберем и информацията за заявките. Това е може би най-оптималният подход за целите на задачата. Но тъй като времето, необходимо за изчисляване на корените на полиномите, е значително по-голямо от времето, необходимо за отговори на запитвания, използването на този метод не е наложително. Сложност – **O((F + Q) \* log(F + Q))** за сортиране, **O(F + Q)** за събиране на информацията.

Сега остава само да намерим интервалите, в които полиномите са положителни – което всъщност е еквивалентно на задачата да намерим корените им. За полиномите от първа и втора степен това е тривиално:

Ако

Ако няма корени

По-интересно е намирането на корените на полиномите от трета степен. Тъй като то не е толкова тривиално, са предвидени 20-те процента от тестовете без такива полиноми.

Съществуват формули, които намират всички корени на полином от трета степен, но те не са особено приятни за помнене и не се очаква да се знаят от участници в състезание по информатика, така че най-вероятно трябва да потърсим друг подход.

Не е особено трудно да се стигне до идеята, че ако имаме интервал, в единия край на който даден полином приема положителна стойност, а в другия – отрицателна, то в този интервал полиномът има поне един корен – това е изпълнено, защото полиномът е непрекъсната функция и няма как функционалните му стойности да „прескочат” от положителни в отрицателни числа. Във висшата математика това е известно като теорема на Болцано, или в студентските среди – като „теоремата за спагетите”. Този корен може да се намери, използвайки модификация на двоично търсене по следния начин:

Разглеждаме интервал , такъв че , където f е полиномът, който разглеждаме в момента (това условие всъщност е еквивалентно на твърдението, че f е с различни знаци в двата края на интервала). В зависимост от стойността на предприемаме следните действия:

1. Ако (всъщност достатъчно е или , където е достатъчно малко положително число), то вече сме намерили корена и можем да терминираме търсенето.
2. В противен случай избираме един от двата интервала и , така че да запазим инвариантата, че стойностите на полинома в двата края на интервала са с различни знаци. Това ни позволява да продължим търсенето аналогично в новия интервал, който е двойно по-малък от първоначалния.

Сега остава само да си изберем подходящ интервал. Един възможен начин е да намерим екстремумите на полинома (т.е. локалните минимуми и максимуми) и да търсим корени между тях. Това обаче би изисквало намиране на производна на функция, което също не е необходимо за доброто представяне на състезание по информатика.

Тук е моментът за следващото наблюдение – за всеки полином от трета степен знакът, който стойностите на полинома приемат в , е различен от този, който приемат в . Нестрого обяснение на този факт е горе-долу следното: колкото по-голямо става , толкова по-голямо става в сравнение с . По тази причина знакът на полинома при агрумент, клонящ към безкрайност (т.е. към достатъчно голямо по абсолютна стойност число), се определя единствено то знака на , който е различен в + и . Всъщност това наблюдение, допълнено с предишното, ни гарантира, че всеки полином от трета степен (и изобщо от нечетна такава) има поне един реален корен. За неговото намиране можем да използваме описаното по-горе двоично търсене.

Остава само да намерим другите два корена (ако има такива). За целта можем да се възползваме от факта, че ако дадено число е корен на полином , то , където е полином от степен с 1 по-малка от степента на . В нашия случай степента на е трета, така че ако намерим , той ще бъде от втора степен и ще можем да намерим корените му, като просто решим уравнение от втора степен по вече известния ни начин. За да намерим коефициентите на , можем да използваме два основни метода:

1. Деление на полиноми – делим на и частното е . Можем спокойно да приемем, че остатъкът е нула, тъй като точно дели .
2. Схема на Хорнер – схемата на Хорнер всъщност е алгоритъм за деление на полином с линеен полином – т.е. в случая, деление на с . Изучава се в училищната математика, и то точно с такава цел – по един намерен корен да получим уравнение, в което да търсим останалите.

И по двата начина (очаквано) се получават еднакви резултати. Трябва да се отбележи, че имплементирането на някой от тези методи в програма е напълно ненужно – тъй като ги използваме винаги за деление на полином от трета степен на полином от вида , можем просто да пресметнем коефициентите и да спестим няколко реда код (и няколко минути мислене и дебъгване). Решението чрез схема на Хорнер е горе-долу следното:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |

Полученото в последната колона – – е точно стойността на f в точката (това твърдение е известно в математиката като малка теорема на Безу), която, както знаем, е 0. Т.е. единственото, което ни остава, за да намерим корените на уравнение от трета степен, за което знаем, че е корен, е да решим следното квадратно уравнение: