АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА МАКСИМАЛЕН СБОР ОТ ЦИФРИ

Задачата може да се атакува с набор от стандартни алгоритми, които трябва да са познати: обхождане на последователни числа (с вложен цикъл!), отделяне на цифрите на число и сумирането им, сумиране на сумите (и двете чрез суматори), поддържане на максимална стойност. За правилна реализация на комбинацията от познати алгоритми са предвидени 30% от точките. Примерна реализация може да се види във файла sumdig30.cpp. Алгоритъмът е със сложност O(k.(b-a).lg(b)) (логаритъмът е заради работата с цифрите в най-вътрешния цикъл).

За по-големите обхвати трябват някакви подобрения в изчислителния процес. Най-съществено е досещането, че сумата на цифри за следващата k-орка всъщност се получава от предишната, като се извади сумата от цифрите на най-малкото число в нея и се добави сумата от цифрите на новото (следващо) число, което идва. Тази идея всъщност има динамичен привкус (макар и в най-прост вид) и състезател в Е група, за когото тя е естествена, има сериозно разбиране на идеите в информатиката. За такова постижение са предвидени 70% от точките. Примерна реализация – sumdig70.cpp. Сложността пада на O((b-a).lg(b)).

По-нататъшно подобрение на изчислителния процес се получава чрез вмъкване на някои идеи, които за тази група са „висш пилотаж“. Те са на естествена за възрастта математическа основа, но алгоритъмът, макар и прост, е твърде специфичен и не се изучава в училище в явен вид – предполага се досещане. Става дума за това, че сумата от цифрите X на едно число N лесно може да се получи от сумата на цифрите S на предишното (N-1). Алгоритъмът е много естествен и следва непосредствено от познатия алгоритъм за увеличаване на цяло число в десетичен запис с 1. А именно:

- Присвояваме на X стойността на S;
- Докато последната цифра на (N-1) е 9, намаляваме X с 9 и я „изтриваме“ от (N-1);
- Увеличаваме X с едно.

Този подход на практика премахва множителя lg(b) от сложността на задачата. Съчетаването на тези идеи трябва да реши задачата докрай. За тази възрастова група не бива да изискваме по-задълбочен анализ, но в задачата се съдържат още идеи, съвсем различни от разгледаните (които са по-скоро „очакваните“). Няма да е чудно, ако някой състезател просто „умозрително“ се хване за тях, което може да донесе разнообразни точки, в зависимост от точността на съчинения алгоритъм.

Става дума за естественото „лакомо“ досещане, че вероятно в търсената k-орка има място число, което се записва „с най-големи цифри“ или, близко до това, „има най-голяма сума от цифри“. Идеята е да съставим такова число Q, което все пак е в рамките на интервала [a, b], и да се опитаме да го допълним с още k-1 числа „около него“, като образуваме желаната k-орка.

Как можем да съставим такова „ключово“ число? Ако b има за цифри само деветки, полагаме просто $Q = b$. Нека броят цифри на b е n, а числото a се записва с цифрите $\overline{a_p a_{p-1} a_{p-2} \cdots a_1}$. Тъй като $a < b$, може да си мислим, че, ако a е с по-малко от n цифри, то е допълнено с необходимия брой (по-точно n-p) водещи нули. Полученото можем да считаме за началното $Q = \overline{\underbrace{00\cdots0}_{n-p}\, a_p a_{p-1} a_{p-2} \cdots a_1}$. Започвайки от цифрата на единиците на Q, увеличаваме всяка цифра (най-много до 9), до първия момент, в който Q **надмине** крайната стойност b за интервала, който разглеждаме. Тогава предишното състояние е търсеният ключ Q. След това можем да разгледаме всички k-орки, в които участва Q, и да върнем максимума на сумата от цифрите, който е пресметнат за участващите числа в някоя от тях. Реализацията на алгоритъм като този крие доста подводни камъни, но сложността му може да бъде сведена до O(k). Примерна реализация – sumdigAlt.cpp. Тестовете са подбрани така, че при намиране на ключа и на първата k-орка, в която той участва „възможно най-надясно“, правилната реализация на тази творческа идея да донесе до 65% от точките.

Автор: Павлин Пеев