АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА
ZERO

Задачата Zero трябваше да бъде интересна не толкова заради алгоритъма, който я решава (динамично оптимиране) а предимно заради свеждането на задачата до него. Най-очевидното решение беше да пазим числата на всяко от петте момичета и да имитираме възможните стъпки. Тъй като числата могат да са сравнително големи (до 30,000) това изисква ужасно много памет и би било напълно невъзможно освен за много малки тестове, каквито почти нямаше. Състезателите можеха да се досетят, че таблицата е изключително рядка, тъй като сравнително често делим числата на две. Поради това си свойство броят запълнени клетки е логично да бъде от рода на log(30,000)^5. Това би било напълно вярно ако имахме САМО деления, но не и изваждания. Тук е момента да съобразим, че изважданията са разрешени само ако и двете числа са нечетни, тоест биха могли да настъпват само всяка втора стъпка (след изваждане, числото задължително е останало четно и ще бъде разделено на следващата стъпка). От тук броят запълнени клетки на таблицата става O((log(30,000) * 2)^5). Това не е твърде много, тъй като log(30,000) = 15, а 30^5 = 24,300,000 (а всъщност много по-малко). Един съобразителен състезател, който не се досети за истинското решение, би могъл да "cheat"-не, като ползва map за да пази state-а на динамичното. За съжаление на съобразителните състезатели, обаче, и аз се досетих за това решение и тестовете са така направени, че тези решения да хващат около 50 точки.

Въпреки бавнотата на STL-ския map, горното решение е вярно, стига да има някакъв по-добър начин да пазим state-а, тоест числата на момичетата. Наблюдението, че след изваждане задължително следва събиране е изключително важно за това. Нека разгледаме случая, когато сме задължени да делим четно число и сме задължени да изваждаме единица от нечетно (какъвто, всъщност, беше оригиналният замисъл на задачата). Например нека имаме числото 35. На първа стъпка вадим и получаваме 34. На втора стъпка делим, и получаваме 17. На трета стъпка вадим и получаваме 16. На следващите четири стъпки сме задължени да делим, а на последната – отново да извадим. Ако разгледаме числото в двоичен запис, то 35(10) = 100011. Гледайки числото отзад напред можем да намерим известна връзка между изважданията и единиците в двоичен запис – където имаме единица, трябва да вадим и след това да делим, а където имаме нула – само да делим. Нека представим всяко от числата на момичетата като последвателност от вадения и деления по гореописания алгоритъм. Тези последвателности, както казахме, са не по-дълги от log(30,000) * 2, тоест около 30 (всъщност 28 е най-дългата възможна такава редица). Редицата за 35 би била "-/-////-", а за 42 би била „/-//-//-". По индекса в тези редици можем еднозначно да определим какви са текущите числа на момичетата. Тоест, ако числото е нечетно (текущият знак от редицата е `-`) можем да извадим едно (и да увеличим индекса с единица), или да разделим числото на две (тоест да извадим едно и да разделим на две, като увеличаваме индекса с 2). Горното е вярно, тъй като, както казахме, след всяко вадене делим, а целочисленото деление прави

точно това – премахва остатъка (който ние изваждаме). Ако пък числото е четно, тоест текущият знак в редицата е `/' единственият ни избор е да разделим числото, като увеличаваме индекса в редицата с единица. Какво печелим от тези редици? Ами с тях можем еднозначно да определяме числата на момичетата в текущия state на динамичното. Тъй като редиците са с максимална дължина, 28 ни стига да пазим 5 индекса между нула и 27, включително, които указват до кой индекс сме стигнали в редицата на всяко от момичетата. Необходимата памет за това е 28^5 = 17210368 клетки, което е достатъчно малко за да запазим в int масив (този масив заема малко по-малко от 70 мегабайта). Изборът дали ще ползваме петмерен масив (int()()()()()) или ще кодираме state-а в едно единствено число е изцяло наш. Дори този съкратен масив е значително рядък, едва около десета от него е запълнена с реални клетки, но те са разположени хаотично в него и няма много лесен начин да спечелим заслужаващо си количество памет от неизползваната. Тъй като работата с масив е значително по-бърза от работата с мап, това решение е достатъчно бързо да хване 100 точки.

Каква е сложността на горното решение? За всяко състояние имаме един единствен цикъл до 5, който определя кои две момичета ще пият. Тоест общата сложност на алгоритъма е O((log(30,000)*2)^K * K), където K е броят момичета (в случая K = 5). Макар и експоненциална, фиксираният малък брой момичета позволява това решение да е достатъчно бързо.

Автор: Александър Георгиев