XLII НАЦИОНАЛНА ОЛИМПИАДА ПО ИНФОРМАТИКА
Общински кръг, 20 декември 2025 г.
Група С, 7 – 8 клас

Задача С2. ЛАМПИ (Анализ на решението)

⏳ 0.5 сек. 💾 16 MB

Нека с A_n означим минималния брой ходове, с които можем да включим първите n лампи, ако първоначално всички те са изключени. Важно наблюдение е, че възможността да променим дадена лампа зависи само от лампите с по-малки номера.

Първа подзадача

За $N \leq 20$ можем направо да симулираме ходовете. Нека функцията `light(n)` променя състоянието на лампа n. Преди това трябва да направим лампа $n-1$ включена, а лампите преди нея - изключени. Обхождаме ги отдясно наляво и за всяка лампа, чието състояние не е желаното, рекурсивно извикваме същата функция. Накрая превключваме лампа n и увеличаваме отговора с единица.

За да включим всички лампи, отново ги обхождаме отдясно наляво. Ако текущата лампа е изключена, извикваме `light` за нея. Така вече обработените лампи не се променят. Оптималността на този подход ще следва и от разсъжденията в следващата подзадача.

Постигната сложност: $O(2^N)$ време и $O(N)$ памет.
Имплементация: `lights_kiril_30p.cpp`

Втора подзадача

Тук $N \leq 10^6$, така че не можем да извършваме всеки ход поотделно. Нека първо решим помощна задача: с колко хода от изцяло изключени лампи можем да стигнем до състояние, в което единствено лампа n е включена? Нека този брой е D_n.

В оптимално решение лампа n се превключва точно веднъж. Наистина, ако я превключваме повече пъти, можем да премахнем две нейни превключвания, без това да повлияе на възможността да променяме по-малките лампи. Преди единственото превключване трябва само лампа $n-1$ да е включена. След него трябва да изключим и нея. Двата прехода са обратни един на друг и изискват по D_{n-1} хода. Следователно

$$D_0 = 0, \qquad D_n = 2D_{n-1} + 1 = 2^n - 1.$$

Да се върнем към първоначалната задача. За да включим лампа n, първо правим D_{n-1} хода и после още един за самата лампа. В този момент лампите $n-1$ и n са включени, а всички преди тях - изключени. Остава да включим първите $n-2$ лампи. Няма нужда отново да променяме последните две: евентуалните им двойки превключвания могат да се премахнат по същата причина. Получаваме

$$A_0 = 0, \qquad A_1 = 1, \qquad A_n = D_{n-1} + 1 + A_{n-2} = 2^{n-1} + A_{n-2}.$$

Авторовата програма предварително изчислява стойностите D_n в масива `dp`. След това събира $D_{N-1}+1$, $D_{N-3}+1$ и така нататък, като при нечетно N добавя и A_1. Всички изчисления се правят по модул 10^9+7.

Постигната сложност: $O(N)$ време и $O(N)$ памет в дадената имплементация. Паметта може да се намали до $O(1)$.
Имплементация: `lights_kiril_75p.cpp`

Трета подзадача

Остава да пресметнем същата сума по-бързо. От рекурентната зависимост се вижда, че събираме степени на двойката, чиито показатели намаляват с две. Това е геометрична прогресия:

$$A_N = 2^{N-1} + 2^{N-3} + \cdots = \begin{cases} \dfrac{2^{N+1}-2}{3}, & \text{при четно } N, \\ \dfrac{2^{N+1}-1}{3}, & \text{при нечетно } N. \end{cases}$$

Степента 2^{N+1} намираме с бързо степенуване. Трябва да внимаваме с делението: след вземане на остатък не можем просто да разделим на 3. Вместо това умножаваме по обратния на 3 елемент по модул $p = 10^9 + 7$. Тук той се намира непосредствено:

$$3^{-1} \equiv \frac{p+1}{3} = 333333336 \pmod{p}.$$

Във формулата изваждаме 1 или 2 според четността на N, а след това умножаваме по тази константа и вземаме остатък. При реализация на модулно изваждане е удобно да прибавим p, преди отново да вземем остатък. За произведенията използваме 64-битов целочислен тип.

Постигната сложност: $O(\log N)$ време. Рекурсивното степенуване в авторовото решение използва $O(\log N)$ памет; итеративният вариант използва $O(1)$.
Имплементация: `lights_kiril_100p.cpp`, `author.cpp`

Автор: Кирил Зулямски