АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА ДРЕВНИ ЯЙЦА

Подход 1: Динамично оптимиране

Първо да забележим, че ако $K \geqslant \log_2(N)$ тогава можем да решим проблема с двоично търсене в етажите и отговорът винаги е $\lceil \log_2(N) \rceil$. В случая, когато $K < \log_2(N)$, прилагаме динамично оптимиране. Изчисляваме двумерната таблица A където $A[N][K]$ съдържа най-малкия брой опити, с който можем да определим в N етажна сграда с K яйца къде яйцата започват да се чупят.

Инициализация: Очевидно $A[i][1] = i$ и $A[1][i] = 1$.

Рекурента зависимост: За да разберем рекурентната зависимост, нека вземем първият ни опит. Ако хвърлим първото яйце от етаж I, имаме два варианта:

- Ако се счупи, тогава най-ниският "чуплив" етаж е някъде между 1 и I и ни остават K-1 яйца. След като знаем, че яйцето се чупи в I, остава да определим дали се чупи някъде между 1 и I-1 със K-1 яйца. Това можем да направим най-малко с $A[I - 1][K - 1]$ опита.
- Ако яйцето не се счупи, тогава трябва да проверим дали се чупи някъде между етаж I+1 и N, но все още имаме K яйца. Това можем да направим най-малко с $A[N-I][K]$ опита.

В най-лошия случай, ако хвърлим първото яйце от I-тия етаж, ще ни трябват $1 + \max(A[I - 1][K - 1], A[N - I][K])$ опита за да определим от кой етаж се чупи яйцето. Тогава стойността на $A[N][K]$ е:

$$A[N][K] = \min_{I=1..N} (1 + max(A[I-1][K-1], A[N-I][K]))$$

Използвайки тази рекурентна зависимост, можем да попълним A и да намерим отговора в позиция $A[N][K]$. Сложността на това решение е $O(N^2K)$, но тъй като $K<\log_2(N)$ на практика сложността е $O(N^2\log_2(N))$. За подробности вижте авторския код.

Подход 2: Извеждане на формула

Идеята е на проф. Роналд ван Люйк, който по стечение на обстоятелствата се оказа "на точното място в подходящото време" и съдейства на авторите.

Възможно е да изведем следната формула за най-високата сграда която можем да решим с K яйца и T опита: $h_K(T) = \sum_1^k \binom{T}{i}$.

Използвайки тази формула, можем да намерим най-малкото T за което $h_k(T) \geqslant N$.

Тъй като $\binom{T+1}{I} = \frac{\binom{T}{I} * (T+1)}{(T-I+1)}$, можем да изчислим $h_k(T+1)$ от $h_k(T)$ за време $O(K)$. По този начин цялата задача може да бъде решена в време $O(N K)$.

Отново, тъй като прибягваме до формулата само когато $K<\log_2(N)$, сложността на това решение става $O(N \log_2(N))$.

Преди да прочетете доказателството, предлагаме на по-любознателните ученици да го направят сами, използвайки следния план:

1. Покажете че $h_K(T) = \sum_0^{T-1} h_{K-1}(i) + 1$.

2. Отбележете, че $h_1(T) = \binom{T}{1} = \sum_1^1 \binom{T}{i}$.

3. Използвайки стъпка (1) и формулата $\binom{N}{K} + \binom{N}{K+1} = \binom{N+1}{K+1}$, докажете формулата по индукция.

Подробно доказателство:

1. Доказване, че $h_K(T) = \sum_0^{T-1} h_{K-1}(i) + 1$

Нека x е най-високият етаж, от който можем да хвърлим първото яйце. Ако то се счупи остават k-1 яйца и T-1 опита, и с тях трябва да определим на кой етаж между 1 и x-1 се чупи яйцето. Тогава $x \leqslant h_{k-1}(T\text{-}1)+1$. Тъй като искаме да максимизираме x, избираме $x = h_{k-1}(T-1)+1$. По същата логика хвърляме следващото яйце от етаж $h_{k-1}(T-1)+1+h_{k-1}(T-2)+1$ и т. н. Продължавайки по този начин, получаваме сумата $h_k(T)=\sum_0^{T-1} h_{k-1}(i)+1$.

2**.** Тривиално.

3**.** Стъпка (2) е базата на нашата индукция. Да допуснем, че за някое k вече имаме $h_{k-1}(T)=\sum_1^{k-1}\binom{T}{i}$. Тогава:

$$h_k(T)=\sum_0^{T-1} h_{k-1}(i)+1=\sum_{i=0}^{T-1}\left[\sum_{j=1}^{k-1}\binom{i}{j}+1\right]=\sum_{i=0}^{T-1}\sum_{j=1}^{k-1}\binom{i}{j}+T=\sum_{j=1}^{k-1}\sum_{i=0}^{T-1}\binom{i}{j}+T=$$

$$\sum_{j=1}^{k-1}\left[\binom{0}{j}+\binom{1}{j}+\binom{2}{j}+\binom{2}{j}+\ldots+\binom{T-1}{j}\right]+T=\sum_{j=1}^{k-1}\left[\binom{1}{j}+\binom{2}{j}+\binom{2}{j}+\ldots+\binom{T-1}{j}\right]+T$$

Във вътрешната сума може да добавим $\binom{1}{j+1}=0$. Така получаваме:

$$\sum_{j=1}^{k-1}\left[\binom{1}{j+1}+\binom{1}{j}+\binom{2}{j}+\binom{2}{j}+\ldots+\binom{T-1}{j}\right]+T=\sum_{j=1}^{k-1}\left[\binom{2}{j+1}+\binom{2}{j}+\binom{2}{j}+\ldots+\binom{T-1}{j}\right]+T=\ldots$$

$$\ldots=\sum_{j=1}^{k-1}\binom{T}{j+1}+T=\sum_{j=2}^{k}\binom{T}{j}+\binom{T}{1}=\sum_{j=1}^{k}\binom{T}{j}$$

Подход 3: Почти като формулата, но малко по-лесно

Вглеждайки се по-внимателно в подход 2, се оказва, че стъпвайки на доказаната зависимост $h_k(T) = \sum_0^{T-1}(h_{k-1}(i) + 1)$, можем да решим задачата без да извеждаме и използваме комбинаторната формула. Прилагайки тази зависимост, можем да получим:

$$h_k(T) = \sum_0^{T-1}(h_{k-1}(i) + 1)$$

$$= \sum_0^{T-2}(h_{k-1}(i) + 1) + h_{k-1}(T-1) + 1 = h_k(T-1) + h_{k-1}(T-1) + 1$$

Бележка: Поразсъждавайте и ще откриете, че последната зависимост може да бъде изведена пряко чрез разсъждения и без да стъпвате на доказаната зависимост $h_k(T) = \sum_0^{T-1}(h_{k-1}(i) + 1)$.

Както знаем $h_1(T) = T$. Тогава можем да построим следната таблица, в която номерата на редовете са броя яйца, а номерата на редовете – броя опити. В елемента на таблицата, намиращ се на ред i и стълб j се намира максималната височина на сграда, за която можем да решим задачата с i яйца и j опита.

T / K	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	1	3	6	10	15	21	28	36	45
3	1	3	7	14	25	41	63	92	129
4	1	3	14	22	37	63	105	169	262

Запълването на таблицата става много лесно – по редове, като всеки ред се запълва отляво надясно. Първият ред е ясен – в него стоят числата 1, 2, 3,.....Първият стълб е изцяло запълнен с единици. Тогава, започвайки от втория ред и втория стълб, всеки следващ елемент се получава като сума от предхождащия го на реда елемент, елемента който е горе, вляво от него и 1. Това запълване на всеки ред трябва да продължава, докато в него се

получи височина на сградата, по-голяма или равна на N (можете да забележите, че редовете ще стават все „по-къси“). Когато в ред с номер K достигнем до число, по-голямо или равно на N, номерът на стълба на това число дава търсения минимален брой опити.
Това, което е важно е, че на нас въобще не ни е нужна цялата таблица – в сметките участват само последните два реда. Това ни позволява да работим с масив p[2][N], като алтернативно сменяме редовете му, докато не стигнем до K яйца. Тъй като N може да е много голямо (до 10^9), то не можем да си позволим да използваме толкова голям масив. За целта е достатъчно да се съобрази и пресметне, че още на втория ред от таблицата, за достигане до височина на сградата, по-голяма от 10^9, трябва да се пресметнат по-малко от 45000 елемента. Така че случаят K=1 може да се обработи отделно (извежда се направо стойността на N) и първият ред на масива да се запълни направо за K=2, използвайки формулата p[0][j]=p[0][j-1]+(j-1)+1=p[0][j]+j (p[0][1]=1). По този начин масивът, който ни трябва е с максимална размерност [2][45000].
На пръв поглед решението е със сложност $O(N*K)$ или, тъй като разглеждаме $K<log_2N$, $O(N*log_2N)$. Ако това беше така, то нямаше да е много добре, като се има предвид, че N може да е до 10^9. Горните разсъждения показват, че при K=1 решението е със сложност O(1) (извежда се направо N), а при K>1, броят на елементите в редовете рязко намалява със всеки следващ ред, така че всъщност сложността е доста по-малка – определено по-малка от $O(45000*K)$.

Реализациите са във файловете ***egg_dyn.cpp, egg_formula.cpp*** и ***eggR.cpp***.

Идея на задачата: Димитър Бунов, Павел Петров
Реализация: Димитър Бунов, Руско Шиков
Анализ: Димитър Бунов, Руско Шиков