Разбор задачи «Микросхемы и макросхемы»

Ответ на задачу $= 3a + 5b$. Для прохождения последней группы тестов нужно учесть переполнение и воспользоваться 64-битным типом данных.

Разбор задачи «Шаттл до Рутнока»

Заметим, что можно добиться того, чтобы в шаттле не было нескольких пустых капсул подряд. Если после нескольких пустых капсул есть капсулы, занятые повстанцами, то можно сдвинуть этих повстанцев в капсулы с меньшими номерами. Если далее нет занятых капсул, то можно разместить в свободные капсулы еще как минимум одного повстанца.

Также можно добиться того, чтобы первая капсула была занята. Если она свободна, можно сдвинуть всех повстанцев на одну капсулу влево, и это не ухудшит ответ.

Два этих замечания позволяют написать простое линейное решение, которое пройдет группы тестов с первой по пятую. Поместим повстанца в первую капсулу, а дальше в цикле будем идти с шагом через одну капсулу и помещать повстанцев туда, пока не дойдем до капсулы n или $n - 1$.

Заметим, что каждая вторая капсула в нашем линейном решении оказывается свободной. Если n четно, тогда и свободных, и занятых капсул будет по $\frac{n}{2}$. При нечетном n мы получим $\frac{n+1}{2}$ занятых и $\frac{n-1}{2}$ свободных капсул. Запрограммировав эти формулы, можно пройти шестую и седьмую группы тестов соответственно.

В последней группе тестов число n не помещается в 32-битный целый тип, поэтому для ее прохождения нужно использовать 64-битный целый тип.

Разбор задачи «Башни из контейнеров»

В первой группе тестов исходная высота единственной башни равна 1, и понадобится $m - 1$ контейнер, чтобы сделать ее высоту равной m. Во второй группе тестов $m \geqslant h_1$, поэтому понадобится $m - h_1$ контейнеров.

Посчитаем суммарное количество контейнеров в башнях: $S = h_1 + \cdots + h_n$. Если высота каждой башни не менее m, то в сумме в них не менее nm контейнеров. Так как мы можем распределять контейнеры как угодно, то при $nm \leqslant S$ нам не нужны дополнительные контейнеры (ответ 0). Иначе ответ $= nm - S$.

Для прохождения последней группы тестов нужно учесть переполнение и воспользоваться 64-битным типом данных.

Разбор задачи «Нежданные гости»

Решение, моделирующее с шагом в одну секунду поведение объекта по описанному в условии алгоритму, проходило первые три группы тестов.

Для того чтобы пройти четвертую группу тестов, можно было на шагах 1 и 3 сдвигать объект сразу на w квадратов и прибавлять к текущему времени w секунд, до тех пор пока не будет превышено время t. Если на очередном шаге 1 и 3 время t оказалось превышено, нужно откатить последний шаг и смоделировать последние секунды движения простой эмуляцией (как в первом решении). Такое решение работает за время $O(\frac{t}{w} + w)$.

Для полного балла нужно было реализовать решение, работающее за время $O(1)$. Заметим, что за $2w+2$ секунды объект проходит один полный цикл траектории, то есть все четыре шага описанного в условии алгоритма. Пусть $d = \lfloor \frac{t}{2w+2} \rfloor$, а $r = t \mod (2w + 2)$. d — количество полных циклов траектории движения объекта, за каждый цикл он сдвигался на два квадрата в сторону увеличения номеров столбцов, значит, после d циклов он оказался в столбце с номером $2d$. После этого объекту осталось пройти еще r секунд. Разберем два случая:

1. Если $0 \leqslant r \leqslant w$, то объект окажется в строке r и столбце $2d$
2. Если $w + 1 \leqslant r \leqslant 2w + 1$, то объект окажется в строке $2w + 1 - r$ и столбце $2d + 1$

Разбор задачи «Круговая по рукам»

В первой группе тестов тиif забирает единственную монету из первого мешка, и количество ЗЕТов в остальных двух мешках не меняется. Ответ: «0 1 1»

Во второй группе тестов во всех мешках по одному ЗЕТу. Положим x_i равным 1 для всех i от 1 до n и будем обнулять x_i, встречая i в запросах. Тогда после всех запросов x_i будет количеством ЗЕТов в i-м мешке.

Для прохождения третьей группы тестов нужно проэмулировать описанный в условии процесс: забирать из q_i-го мешка один ЗЕТ, а остальные ЗЕТы раскладывать в $(q_i + 1)$-й, $(q_i + 2)$-й и так далее мешки. Каждый запрос мы обрабатываем за $O(x_{q_i})$, где x_{q_i} — количество ЗЕТов в q_i-м мешке к моменту этого запроса.

Для прохождения четвертой группы тестов нужно было заметить, что в процесе эмуляции не обязательно раскладывать все ЗЕТы из мешка по одному: если в мешке осталось x_{q_i} ЗЕТов, то мы можем добавить $\lfloor \frac{x_{q_i}}{n} \rfloor$ во все мешки, а оставшиеся $x_{q_i} \mod n$ разложить в мешки с номерами $q_i + 1$, $q_i + 2$, $\ldots$, $((q_i - 1 + x_{q_i} \mod n) \mod n) + 1$. Так каждый запрос мы обрабатываем за $O(n)$.

Заметим, что мы совершаем операции трех типов: обнуляем количество ЗЕТ-ов в одном мешке, прибавляем ко всем мешкам одно и то же число ЗЕТ-ов или прибавляем один ЗЕТ к какому-то отрезку мешков (если происходит переход от n-го мешка к первому, то нужно прибавить один ЗЕТ к двум отрезкам). Вторую и третью операции можно выполнять эффективнее, чем за $O(n)$, если воспользоваться какой-нибудь древовидной структурой данных. Например, дерево отрезков умеет выполнять все три операции за $O(\log n)$. Так можно было пройти пятую группу тестов.

Разбор задачи «Ящик Пандоры»

Наивное решение, которое для каждого k_i проходит по массиву кнопок с шагом k_i и меняет состояние каждой встреченной кнопки на противоположное, проходит первые четыре группы тестов.

Перейдем к полному решению задачи. Для начала заметим, что ответ не зависит от порядка k_i. Кроме того, если два ученых высказали предположения с одинаковым k_i, то применение этих предположений по очереди дважды меняет состояние некоторых кнопок, что равносильно тому, что состояние никаких кнопок не меняется. Тогда можно сгруппировать предположения с одинаковым значением k_i — с помощью сортировки или структуры данных map / dictionary — оба способа работают за $O(n \cdot \log(n))$. После этого останется применить по одному разу лишь те предположения, которые были высказаны нечетное количество раз.

Теперь обработаем оставшиеся предположения k_i наивным способом, описанным выше. Один проход по массиву кнопок совершит $\lfloor \frac{n}{k_i} \rfloor$ шагов. Поскольку все k_i теперь различны, суммарное количество шагов такого алгоритма можно оценить сверху величиной $\sum_{i=1}^{n} \lfloor \frac{n}{i} \rfloor$. Известный факт, что такая сумма имеет порядок $O(n \cdot \log(n))$ — вы можете найти его доказательство в Интернете в статьях про гармонический ряд. Такая оценка позволяет пройти последнюю группу тестов.