Задача A. Прямое попадание

Пройдёмся по массиву порогов прохождения стёкол, сравнивая текущую интенсивность и текущий порог стёкол. Если интенсивность больше либо равна порогу прохождения для текущего стекла, идём дальше, снижая интенсивность на значение, равное порогу. Иначе останавливаемся и выводим порядковый номер текущего стекла. Если интенсивность ни разу не стала меньше порога стекла, то свет прошёл все стёкла, и мы выводим «-1».

Задача B. Парковка у отеля

Давайте будем поддерживать упорядоченный массив, в котором будут содержаться суммы, которые готов заплатить каждый постоялец, находящийся на платной парковке. Также нам понадобится переменная sum, в которой будем хранить сумму, которую платят все вместе взятые владельцы, чьи машины стоят на платной парковке. Для каждого нового постояльца проверяем: если мы поставим его машину на платную парковку, увеличит ли это доход за один час; если да, то у нас два случая:

- если на парковке есть свободные места, то мы добавляем транспорт в контейнер и прибавляем сумму, которую готов платить постоялец за 1 час, к sum;
- если минимальная цена в контейнере меньше, чем цена, которую готов заплатить вновь прибывший постоялец, мы убираем с платной парковки транспорт с минимальной ценой и ставим транспорт вновь прибывшего постояльца. В противном случае транспорт вновь прибывшего постояльца отправляется на бесплатную парковку.

Задача C. Волшебные камни

Будем решать задачу методом динамического программирования. Состояния динамики: номер текущей комнаты и цвет камня, который мы хотим взять в этой комнате.

Рекуррентное соотношение: $dp[cur\_room][j] = c_j + max_{i \neq j, i=1..4} dp[cur\_room - 1][i]$, где cur_room — номер комнаты, j — тип камня, c_j — энергия камня типа j.

Начальные условия: $dp[1][j] = c_j$.

Ответом будет значение $max_{j=1..4} dp[n][j]$, где n — количество комнат, j — типы камней.

Задача D. Битые сохранения

Поскольку длина массива может достигать 10^{18}, то его хранение в памяти не представляется возможным. Поэтому нужно сразу формировать ответ к задаче. Считываем очередной индекс устанавливаемого бита и определяем, какую позицию он займет в восьмиразрядном числе. Для этого находим остаток от деления этого индекса на 8, затем полученное число вычитаем из 7, поскольку нумерация битов в массиве дана слева направо, а в двоичной записи чисел она ведется справа налево. Допустим, мы получили индекс i. В полученной позиции i результата устанавливаем бит, если до этого он был сброшен, и сбрасываем в противном случае.

Задача E. Король Молока

Создадим структуру данных, чтобы хранить информацию о коровах. Добавим коров в массив. Напишем компаратор — функцию, которая сравнивает эти структуры. В компараторе сначала сравниваем $a \cdot b$, если они равны, то сравниваем количество символов «a» в s, если и эти значения равны, то сравниваем значения c. Сортируем массив коров при помощи этого компаратора. Сортировать можно как встроенной функцией языка, так и самостоятельно любым алгоритмом быстрой сортировки (с асимптотикой не хуже $O(n\ log\ n)$). Проходим по отсортированному массиву, выводим имена.