H2, разбор контеста

simagin.mail@yandex.ru

A03.03

A1

Решение. Сохраним значения всех отметок на столбах, а также создадим массив, где в ячейке n будем хранить оптимальный путь до n-й гостиницы. Рассмотрим первую гостиницу, очевидно, что есть только один способ добраться до нее. Пусть теперь мы хотим найти оптимальный путь до гостиницы n, а для всех предыдущих оптимальный путь известен. Для это достаточно перебором выбрать, в какой гостинице наиболее выгодно остановиться до этого (стоит отдельно рассмотреть вариант, когда мы сразу едем в гостиницу n). Таким образом, идя от начала массива к концу, можно заполнить его. Требуемый результат будет храниться в последней ячейке. Сложность алгоритма можно оценить сверху, как $\Theta(n^2)$. Память $\Theta(n)$

A2

Решение. Сохраним расстояния до всех мест, где можно построить ресторан. А вомзожные доходы запишем в $profits$. Также создадим массив $result$, где в ячейке n будем хранить оптимальное решение для первых n мест. Решение для $n = 1$ очевидно. Пусть мы хотим найти оптимальную расстановку для первых n мест, когда известны решения для всех $k < n$. Найдем ближайшее рассмотренное место от n-го, расположенное на допустимом расстоянии. Если такого места нет, то на первых n местах можно построить ровно одну гостиницу, и ее нужно расположить там, где максимальная прибыль. Если же такое место есть, к примеру это место m, то стоит рассмотреть два случая.

1. Мы строим очередную гостиницу на месте n. Тогда доход статнет $result[m] + profit[n]$. В действительности, на всех местах между m и n гостиниц быть не может. А на местах включая m и меньше расстановка гостиниц может быть любая.
2. Не строим гостиницу, тогда доход будет равен $result[n-1]$.

Из этих двух вариантов нужно выбрать оптимальный. Заполнив всю таблицу с результатами, получим ответ в последней ячейке.

Если организоввать поиск ближайшего допустимого места с общей сложностью $\Theta(n)$, а для этого достаточно сохранять предыдущее значение, а затем корректировать его перебором, то сложность решения $\Theta(n)$. Память $\Theta(n)$.

B1

Решение. Для каждой цифры d запомним, множество цифр S_d, куда можно перейти ходом коня. Заведем матрицу $M_{n\times 10}$, где n размер номера. В ячейке $M[l,d]$ будем хранить количество номеров длины l, которые начинаются на цифру d. Заполняя матрицу построчно, получим решение задачи.

$$M[l,d] = \sum_{d_{prev}\in S_d} M[l-1, d_{prev}].$$

Относительно операций сложения сложность алгоритма $\Theta(n)$. Память $\Theta(n)$

Замечение. Для решения задачи необходимо реализовать операцию сложения для длинной арифметики. В качесте разрядности системы на 64-битной машине можно взять 10^{18}. То есть цифры у нас будут пробегать значения от $10^{18}-1$ до 0. Разрядность предложена из соображений удобства реализации и скорости работы.

B2

Решение. Пусть есть массив P, где $P[d]$ – максимальная порция сыра в день d. Заведем матрицу $M_{n\times 100}$, где n количество дней, а в ячейке $M[d,p]$ будем хранить оптимальное количество очков, заработанное за первые d дней, если в последний день было съедено p сыра. Заполняя матрицу построчно, получим решение задачи.

$$M[d,p] = \max_{1\le p_{prev}\le P[d-1]} M[d-1, p_{prev}] + \frac{p}{p_{prev}}$$

Сложность алгоритма $\Theta(n)$. Память $\Theta(n)$.

Замечение. Для восстановления оптимального набора порций сыра достаточно в матрицу M также хранить предыдудущую оптимальную порцию сыра.