Разбор задачи «Заплати за воду»

Автор идеи и разработка задачи: Николай Ведерников

Задача решается формулой: $60.48 * n + 25.14 * m + 17.95 * (n + m)$.

Разбор задачи «Счастливые билеты-палиндромы»

Автор идеи и разработка задачи: Николай Ведерников

Заметим, что любой палиндром чётной длины, является счастливым числом (так как половины совпадают при развороте, то и сумма цифр будут равны). Значит нам надо посчитать количество палиндромов длиной N.

Заметим, что первая половина однозначно задаёт палиндром. А всего различных первых половин $10^{\frac{N}{2}}$. Но при вычислении не на языке Python, могло возникнуть переполнение, поэтому можно было вывести одну единицу и $\frac{N}{2}$ нулей.

Разбор задачи «Конкатенируй»

Разработка задачи: Демид Кучеренко

Необходимо было проверить две возможные конкатенации чисел на то, являются ли они полным квадратом числа. Проще всего было считать два числа как строки, построить их конкатенации (просто через операцию +), затем перевести конкатенацию обратно в число и проверить уже число.

Чтобы проверить является ли число x полным квадратом, можно воспользоваться одним из способов:

- Проверить разницу, между вещественным число $\sqrt{x}$ и его округлением до ближайшего целого. Если разница меньше, чем некоторое маленькое число *eps*, то это полный квадрат. Такой метод работает за константное время.
- Завести счетчик i, и увеличивать его, пока $i^2 < x$. Если в конце $i^2 = x$, значит это полный квадрат. Работает за время порядка корня из x.

Разбор задачи «Концентрация»

Разработка задачи: Демид Кучеренко

Две клетки, принадлежащие одинаковым фигурам: (9, 18), (18, 14). Можно было выводить и другие клетки этой пары фигур.

Разбор задачи «Сбор морковок»

Разработка задачи: Демид Кучеренко

Можно было перебрать номер столбца, в котором будет совершен шаг вниз. Тогда, перебирая его слева направо, мы будем знать весь путь: по первой строке мы пройдем элементы с 1 по i, а во второй с i по n. Пусть мы знаем ответ для некоторого i (для $i = 1$ посчитаем его "в лоб", пройдя по всей нижней строке). При переходе к $i + 1$ нам нужно к ответу добавить $a_{1,i+1}$ и вычесть $a_{2,i}$. В остальном наш путь не изменится. Сравним это с достигнутом ранее максимумом и продолжим перебор.

Задача является частным случаем задачи о черепашке на динамическое программирование (пункт 2.3 http://kuimova.ucoz.ru/modul_8-dinamicheskoe_programmirovanie.pdf).

Разбор задачи «Простой прямоугольник»

Автор идеи: Демид Кучеренко Разработка задачи: Николай Ведерников

Для начала, давайте каждое простое число заменим на единицу, а каждое непростое на ноль, $a[i][j] = isPrime(a[i][j])$. Надо уметь проверять на простоту, за корень от числа. Теперь мы получили матрицу из нулей и единиц.

Далее, на полученной матрице из нулей и единиц, давайте посчитаем префиксные суммы, $sum[i+1][j+1] = sum[i][j+1] + sum[i+1][j] - sum[i][j] + a[i][j]$. С помощью префиксных сумм мы сможем отвечать на количество единиц в прямоугольнике $(x1, y1), (x2, y2)$, $sum[x2+1][y2+1] + sum[x1][y1] - sum[x2+1][y1] - sum[x1][y2+1]$.

Последним шагом, давайте переберем левый верхний угол прямоугольник и правый нижний. Если количество единиц в этой матрице равен площади прямоугольника, то этот прямоугольник кандидат на ответ. Среди всех таких прямоугольников, выберем прямоугольник с наибольшей площадью.

Время работы: $O(n^2 * sqrt(10^6)$ — чтобы проверить каждое число на простоту, $O(n^2)$ — на построение префиксных сумм, а так же $O(n^4)$ — перебор углов прямоугольника. Итоговое время работы при $n = 100$ составляет $O(n^4)$.

Находить в матрице максимальный по площади прямоугольник из единиц можно и за время $O(n^2)$, но это не требовалось в этой задаче `https://e-maxx.ru/algo/maximum_zero_submatrix`

Разбор задачи «Понимание»

Автор идеи и разработка задачи: Демид Кучеренко

Задача была на аккуратную реализацию условия. Будем выводить ответ построчно. Сначала для каждой строки нам нужно определить, сколько слов она будет содержать. Поскольку нам не важно, из каких букв состоит слово, будем рассматривать только его длину. Будем "набирать" слова в строку. Пусть мы уже набрали слов на x символов. Чтобы проверить, можно ли добавить еще слово длиной y, мы должны удостовериться, что $x + 1 + y \leqslant w$ (так как нам еще нужно поставить минимум один пробел перед новым словом). Если слово "влазит", то пересчитаем текущую длину как $x = x + y + 1$.

Теперь мы определили множество слов, которые мы хотим вывести в очередной строке. Пусть суммарная длина слов с одним пробелом между ними по-прежнему x, значит нам нужно распределить оставшиеся $w - x$ пробелов между $s - 1$ промежутком, где s это количество слов в строке. Обозначим за m остаток от деления $w - x$ на $s - 1$. Тогда у нас должно быть m блоков пробелов "подлиннее" и $s - 1 - m$ блок пробелов "покороче". Короткие блоки у нас будут длины $1 + \lfloor \frac{w-x}{s-1} \rfloor$. Теперь мы знаем, что после первых $s - 1 - m$ слов мы должны поставить $1 + \lfloor \frac{w-x}{s-1} \rfloor$ пробел, а после следующих $2 + \lfloor \frac{w-x}{s-1} \rfloor$ пробел. И не забудем, что если в строке одно слово, то его нужно вывести без пробелов.

Разбор задачи «Заселение в общежитие»

Автор идеи и разработка задачи: Андрей Захаров

Переберем количество двушек, которые займут мальчики (пусть это число b_2). Тогда мальчикам еще нужно $b_3 = \lceil \frac{b-b_2}{3} \rceil$ трёшек. Проверим что $b_3 \leqslant n$. Теперь получается, что девочкам осталось $g_2 = n - b_2$ двушек и $g_3 = n - b_3$ трешек. Проверим, что они в них помещаются, и если да, аккуратно выведем ответ из имеющихся b_2, b_3, g_2 и g_3.

Разбор задачи «Учтивость»

Разработка задачи: Демид Кучеренко и Николай Ведерников

Поскольку мы не можем влиять на жребий, мы должны составить программу, которая гарантированно победит каждого соперника.

Давайте представим, что играем против всех противников одновременно. Как мы выбираем первый шаг нашей программы? Мы должны сыграть против каждого оппонента, поэтому мы рассмотрим набор их первых ходов. Если он включает в себя все три возможных хода, мы обречены; независимо от того, что мы выберем, по крайней мере один противник победит нас. Если он включает в себя только один ход (например, каждый противник начинает с R), то мы можем выбрать ход, который побеждает этот ход (в данном случае, P) и мгновенно победить всех. В противном случае

набор включает два возможных хода, и мы должны выбрать ход, который не проиграет одному и выиграет у другого. Например, если все противники ведут с S или P, мы должны выбрать S.

Исключая всех побежденных противников и переходя к следующему ходу этого комбинированного "матча", мы можем применить ту же стратегию, но с учетом набора вторых ходов оставшихся противников (циклически просматривая их программы по мере необходимости), и так далее. Мы будем устранять по крайней мере одного противника с каждым ходом, поэтому после n ходов у нас будет либо наша программа выигрыша, либо мы поймем, что ответ не существует.

Исключать программы можно или с помощью множества в вашем языке, или просто отмечая в массиве, что эта программа уже побеждена.