Задача 1. Ошибка в расписании

`Максимальная оценка за задачу: 100 баллов.`

Для решения задачи необходимо для каждой последовательной пары уроков проверить все необходимые условия. Для упрощения сравнения время начала каждого урока можно перевести в количество минут, прошедших относительно времени `00:00`.

Задача 2. Ворд ли?

`Максимальная оценка за задачу: 100 баллов.`

Задача является оптимизационной. В общем случае, полного решения для этой задачи не существует. Предполагалось, что участники постараются придумать какие-то свои эвристики, которые помогут в решении этой задачи.

Опишем решение жюри, которое набирает 74 балла.

Для начала для каждой пары слов из словаря предподсчитаем паттерн из символов 0, 1, 2, который мы получим, если введем второе слово при загаданном первом.

На каждом ходу будем поддерживать два набора слов: те, которые могут являться правильным ответом с учетом предыдущих попыток, и те, которые мы можем ввести в качестве следующей попытки (первый набор является подмножеством второго). Перед первым ходом оба набора равны словарю.

Перед каждым ходом будем стараться выбрать такое слово из второго набора, которое как можно больше сократит первый набор. Опишем алгоритм, как выбрать такое слово.

Пусть в первом наборе осталось n слов, во втором наборе — m слов ($n \leqslant m$).

Рассмотрим k-е слово из второго набора, и посчитаем среднее количество слов, которое останется в первом наборе после его ввода в качестве следующей попытки с учетом того, что правильным ответом может быть с равной вероятностью любое слово из первого набора. Другими словами, посчитаем математическое ожидание количества слов, которые останутся в первом наборе.

После ввода слова нам может вернуться один из 3^5 возможных паттернов. Присвоим каждому паттерну номер i от 1 до 3^5. Пусть p_{ik} — количество слов из первого набора, которые подходят под i-й паттерн при введенном k-м слове, то есть количество слов, которые останутся в первом наборе при выпадении паттерна i. Вероятность выпадения i-го паттерна при вводе k-го слова равна $\frac{p_{ik}}{n}$.

Математическое ожидание количества слов, которые останутся в первом наборе после ввода k-го слова равно $\sum\limits_{i=1}^{3^5} \frac{p_{ik}}{n} \cdot p_{ik} = \sum\limits_{i=1}^{3^5} \frac{p_{ik}^2}{n}$.

Номер оптимального слова на текущем ходу рассчитывается как $\min\limits_k \sum\limits_{i=1}^{3^5} \frac{p_{ik}^2}{n}$.

Задача 3. Простые сложения

`Максимальная оценка за задачу: 100 баллов.`

Для решения первой подзадачи заметим, что $(a + x, x) = (a, x)$, а значит достаточно добавить к числу любое взаимно простое с ним число.

Для решения второй задачи, рассмотрим 2 случая: если $(a_1, a_2) = 1$, то достаточно одной операции. Покажем, что в противном случае достаточно четырех действий: прибавим такое число x к a_1, чтобы выполнялось $(a_1 + x, a_2)$ был равен 1. Сделать это можно, например, если мы возьмём число x равным $p * a_2 + 1 - a_1$, где p — достаточно большое число. Несложно показать, что при данном x будет равен 1. В итоге остается одно число и задача сводится к предыдущей подзадаче.

Для решения третьей подзадачи покажем, что трех действий всегда хватит: если $(A_1, A_2) = 1$ — очевидно. Несложно показать, что можно подобрать x, такой, что можно стереть обе пары.

Для решения четвертой и пятой подзадач можно заметить, что при данных ограничениях можно явно перебирать вычитаемые из A_i числа.

Для решения шестой подзадачи можно воспользоваться гипотезой Гольдбаха, доказанной для чисел в диапазоне задачи, и вычитать всегда простые нечетные числа.

Для решения седьмой подзадачи заметим, что из каждой степени 2, кроме последней, если их нечетное число, можно последовательно вычитать 3 и 5. Можно показать, что весь ряд тогда будет вычеркнут.

Для полного решения задачи разобьем все числа на пары. Явно сотрем пары взаимно простых чисел. В итоге получаем несколько пар чисел, которые не являются взаимно простыми, и, возможно, ещё одно число. Покажем теперь, что можно избавиться от них, проведя не более $\frac{m}{2}$ раза, где m — количество пар. Рассмотрим случай, когда N — чётный. Для этого из второй пары можно вычесть какое-нибудь большое число и свести задачу к подзадаче 3. Случай нечётным N задача решается схожим образом.

Задача 4. Шахты

`Максимальная оценка за задачу: 100 баллов.`

Шахту удобно представить в виде дерева, вершинами будут являться комнаты шахты, а рёбрами — тоннели. Каждая вершина будет иметь два параметра — объём volume и количество золота worth.

Целью по задаче будет выбрать такое множество непересекающихся поддеревьев начального дерева, что их суммарный объём будет больше заданного L, но суммарное количество потерянного золота в этих поддеревьях будет минимальным из всех возможных.

Как можно заметить, нужно минимизировать один параметр и при этом максимизировать другой параметр. Для решения это задачи поможет динамическое программирование, а конкретнее решение задачи о рюкзаке.

Если упорядочить вершины определённым образом, то тогда результат для n вершин будет тривиально выводиться из результата для $n-1$ вершин.

Для этого отлично подойдёт нумерация по постфиксному обходу. При заходе в вершину, кроме нумерации, можно запомнить количество вершин P_{in}, которые были отмечены до захода, и количество вершин P_{out}, которые было после обхода всех детей этой вершины. P_{out} и будет порядковым номером — тот шаг динамики, когда будет просмотрена эта вершина.

Для оптимизации по памяти достаточно иметь матрицу $N \times L$, отдельно обрабатывая случаи с суммарным объёмом больше L. В поле этой матрицы на l-й позиции n-й строки будет указано минимальное количество золота, которое будет потеряно с комбинацией поддеревьев с корнями, чьи индексы меньше n, и чей суммарный объём равен l.

После этого динамика строится просто. На каждом шагу просматриваются все объёмы $0 \leqslant l < L$:

1. Если значение матрицы по координатам $(i-1, l)$ установлено, то следует обновить значение матрицы по координатам (i, l).

2. Если значение матрицы по координатам (P_{in}, l) установлено, то следует обновить значение матрицы по координатам $(i, l + volume_i)$ или обновить ответ, если $l + volume_i \geqslant L$, чтобы корректно обработать выход за границы.

Асимптотика алгоритма — $O(N \cdot L)$.

Задача 5. Торт

`Максимальная оценка за задачу: 100 баллов.`

Для полного решения задачи зафиксируем одну вершину многоугольника, пусть это будет первая вершина во входных данных — вершина A, далее посчитаем префиксную сумму треугольников, образованных диагоналями, проведенными из вершины A во все остальные. Это выполнимо, так как вершины многоугольника задаются в порядке обхода по часовой стрелке.

Далее рассмотрим вершины, которые находятся в той же полуплоскости относительно прямой разреза r, что и вершина A. Для нахождения площади части многоугольника, необходимо найти вершины C и D, расстояния от которых до прямой r минимально, и которые находятся по разные стороны относительно прямой, перпендикулярной r и проходящей через A. Пусть вершина D следует после вершины C в порядке обхода. Найти вершины C и D можно, предварительно найдя любую точку, находящуюся по другую сторону от вершины A, относительно прямой r, назовем эту точку

— O. Найти такую точку можно также за время $O(log_2 n)$, используя бинарный поиск, так как расстояние до прямой AB должно уменьшаться на каждом шаге.

После этого вершины C и D находятся в разных полуплоскостях относительно AO, находим каждую из них, используя бинарный поиск — расстояние до r должно уменьшаться. Имея точки и D, находим точки пересечения r и отрезков, краями которых являются C и D, назовем эти точки пересечения C' и D'.

Тогда площадь одной из частей многоугольника вычисляется из следующих составляющих: префиксной суммы треугольников от A до C, префиксной суммы треугольников от A до D (разность площади многоугольника и префиксной суммы от A до D), площади треугольника ACD и площади четырехугольника $CDC'D'$. Площадь второй части — разность площади всего многоугольника и первой части.

Ответ на задачу — максимум из первой и второй площади.