Разбор задачи «Дела по дому»

Для того, чтобы решить задачу, нужно было перебрать, куда вначале пойдет Майлз: в магазин или на почту, и будет ли он заходить домой между магазином и почтой, чтобы занести груз. Всего получается четыре варианта, для каждого можно вывести формулу, по которой вычисляется время, а после этого взять минимум из четырех получившихся значений. При этом, если числа a, b и c не удовлетворяют неравенству треугольника, нужно было заранее положить $a = min(a, b + c)$, $b = min(b, a + c)$ и $c = min(c, a + b)$.

Разбор задачи «Возвращение домой»

Для начала заметим, что порядок нажатия кнопок значения не имеет из-за ассоциативности операции «логическое или», поэтому достаточно выбирать только множество кнопок, на которые Майлзу нужно нажать.

Для решения первой подзадачи как раз достаточно перебрать это самое множество кнопок за $O(2^n)$.

Для решения второй подзадачи требовалось реализовать более эффективный алгоритм. Таким, например, мог стать поиск в ширину по неявному графу параллельных вселенных: находясь в какой-то текущей вселенной x, попробуем нажать каждую из кнопок a_i и добавим число $x \vee a_i$ в очередь состояний. Такое решение в худшем случае работает за $O(\max(a) \cdot n)$.

Для решения задачи на полный балл требовалось решать задачу отдельно по битам. Для этого можно было использовать следующие утверждения:

- Если в i-м бите текущего номера вселенной стоит 1, а в i-м бите числа b — 0, добраться в b уже не получится;
- Нет смысла нажимать на такую кнопку a_j, которая не добавит единичных битов к текущему номеру вселенной (то есть где $x \vee a_j = x$);
- Если каждое нажатие кнопки на устройстве добавляет хотя бы один единичный бит к двоичной записи номера текущей вселенной, нажатий на кнопки понадобится не больше 30.

Используя утверждения, написанные выше, можно было разработать следующий алгоритм:

- Текущий номер вселенной — x;
- Найти любое такое a_i, что $x \vee a_i > x$ и в $x \vee a_i$ все единичные биты соответствуют единичным битам в b;
- Нажать на кнопку номер i, переместиться во вселенную $x \vee a_i$.

Логично, что тогда мы сделаем не больше 100 (даже не больше 30) перемещений, а также доберемся до вселенной с номером b, если это вообще возможно.

Разбор задачи «Прогулка по Бруклину»

Решение 1. Перебор. Заметим, что траектория движения движения Майлза однозначно задается местами, в которых он пересекал очередную горизонталь города. Таким образом, существует не более $(n + 1)^n$ различных траекторий, и при небольших ограничениях на n мы можем перебрать их все, проверив для каждой, что она не пересекает дома, и выбрать лучшую из подходящих.

Основное решение. Динамическое программирование. Для того, чтобы решить задачу более эффективно решать задачу, применим динамическое программирование. Состоянием будет являться узел решетки, в котором сейчас находится Майлз, направление, в котором он сейчас двигается, а также разность между площадями частей города. Заметим, что эта разность считается только по части города, которую Майлз уже прошел, и она будет меняться в процессе перехода между состояниями динамики. Значением динамики будет булево значение — существует ли такая траектория, которая заканчивается в данном узле, в которой последний шаг был сделан в данном

направлении, а текущая разность площадей частей равна данному значению. Чтобы пересчитывать динамику, нам нужно перебрать, в каком направлении сделать очередной шаг, при этом, если необходимо, пересчитать значение разности площадей. Таким образом, каждый переход выполняется за константное время, а общее время работы алгоритмы составит $\mathcal{O}(n^4)$, так как существует $\mathcal{O}(n^2)$ узлов решетки и $\mathcal{O}(n^2)$ значений разности площадей.

Полное решение. Битовое сжатие Заметим, что значения нашей динамики — это двоичные биты, а пересчет этой динамики по одному измерению происходит так: набор подряд идущих значений динамики сдвигается на определенное значение, а после этого применяется операция ИЛИ в каждом бите к определенному диапазону значений. В этом случае можно применить такую оптимизацию динамики, как битовое сжатие: хранить диапазоны значений динамики блоками по 64 бита в целочисленном типе данных. После этого изменения операцию побитового или с диапазоном значений динамики можно производить в 64 раза быстрее. Также, чтобы не реализовывать битовое сжатие, можно применить структуру `bitset`. Таким образом, с помощью этой оптимизации можно было ускорить решение в 64 раза и получить полное решение.

Разбор задачи «Взлом компьютера»

Давайте поддерживать бор имен файлов. Тогда, чтобы посчитать ответ для файла, нужно спуститься в боре по его имени и посчитать количество вершин на пути, в которых нужно нажать на кнопку. В корне всегда нужно нажать на кнопку. В вершинах, у которых хотя бы два ребенка или в которых заканчивается какое-либо имя файла, нужно нажать на кнопку, соответствующую букве. В вершинах, в которых не нужно нажимать на букву, но в родителях которых нужно нажимать на букву, нужно нажимать на *tab*. Таким образом, мы получили решение за $O(q \cdot L)$, где L — максимальная возможная длина строки.

Теперь заметим, что при добавлении и удалении имени, значение «нужно ли нажимать на кнопку в вершине» изменяется у не более чем $l \cdot \Sigma$ вершин (это вершины, принадлежащие пути, соответствующему имени, а также их дети). Суммарная длина добавляемых и удаляемых имен не превышает $2 \cdot 10^6$, потому что каждое добавляемое имя удаляется не более одного раза, а суммарная длина добавляемых имен не превышает 10^6. В свою очередь, для ответа на третий запрос, нужно посчитать сумму на пути от корня до вершины. Это стандартная задача, которая решается за $O(\log n)$ на запрос с помощью дерева отрезков. Итоговая асимптотика решения получилась $O(q \cdot \log n + n \cdot \Sigma \cdot \log n)$, где n — суммарная длина имен, а Σ — размер алфавита, т.е. 26.