Всероссийская олимпиада школьников по информатике 2017/2018
Региональный этап

Разбор задач

Автор разбора: Андрей Станкевич

Задача 1. Улучшение успеваемости

Автор задачи: Андрей Станкевич
Сложность задачи: доступна широкому кругу участников
Основные темы: целочисленная арифметика, формула, двоичный поиск

Пусть ученик получит x оценок в 5 баллов. Тогда общее число оценок будет равно $(a + b + c + x)$, а средняя оценка будет равна $(2a + 3b + 4c + 5x) / (a + b + c + x)$. По условию задачи должно выполняться условие

$$(2a + 3b + 4c + 5x) / (a + b + c + x) \geq 3.5.$$

Возможны два подхода для поиска минимального x, удовлетворяющего этому неравенству.

Подход 1. Вывод формулы.

Умножив обе части неравенства на положительную величину $2(a + b + c + x)$, получим неравенство

$$4a + 6b + 8c + 10x \geq 7a + 7b + 7c + 7x,$$

перенося x в левую часть неравенства, остальные слагаемые — в правую, получаем неравенство

$$3x \geq 3a + b - c.$$

Таким образом, минимальное подходящее значение x равно $(3a + b - c) / 3$, округленному вверх. Необходимо также учесть, что x не может быть меньше 0. Для вычисления соответствующего значения, например, в языке C++, можно воспользоваться выражением

```
x = max((3 * a + b - c + 2) / 3, 0);
```

Подход 2. Двоичный поиск.

Заметим, что каждая дополнительная оценка в 5 баллов увеличивает среднюю оценку, следовательно для поиска минимального значения x, при котором среднее значение не меньше 3.5 можно применить двоичный поиск.

Для того, чтобы избежать переполнения целочисленного типа при вычислениях в обоих подходах следует использовать 64-битный тип данных.

Решения, которые используют линейный поиск вместо двоичного, подходят для решения подзадач 1, 2, 3 и 4. Для решения подзадач 1 и 2 можно использовать более простые формулы. Для решения подзадачи 3 можно вычислять каждый раз значение средней оценки, суммируя все оценки по одной, вместо использования формулы $(2a + 3b + 4c + 5x) / (a + b + c + x)$.

Задача 2. Квадраты и кубы

Автор задачи: Андрей Станкевич
Сложность задачи: доступна широкому кругу участников
Основные темы: линейный поиск, операции с вещественными числами

Переберем значение y. Заметим, что поскольку $a \leq y^3 \leq b \leq 10^{18}$, достаточно перебирать значение y до 10^6. Убедимся, что $a \leq y^3 \leq b$ и найдем количество подходящих значений x.

Для фиксированного y значение x^2 должно лежать в диапазоне от $L = \max(y^3 - k, a)$ до $R = \min(y^3 + k, b)$: $L \le x^2 \le R$. Извлекая квадратный корень из всех частей неравенства, получаем ограничение на x: $\sqrt{L} \le x \le \sqrt{R}$.

Нас интересует количество целых значений в интервале от $\sqrt{L}$ до $\sqrt{R}$. Для его определения округлим вверх значение $\sqrt{L}$, пусть $l = \lceil \sqrt{L} \rceil$, и округлим вниз значение $\sqrt{R}$, пусть $r = \lfloor \sqrt{R} \rfloor$. Количество подходящих значений x для данного y, таким образом, равно $(r - l + 1)$.

Перебрав значения y и просуммировав подходящие значения x, получим ответ.

Для поиска квадратного корня из числа можно воспользоваться либо функцией для извлечения корня из вещественного числа `sqrt`, с последующим округлением, либо двоичным поиском.

Для решения подзадач 1 и 2, где $k = 0$, можно заметить, что искомые пары $x^2 = y^3$ являются шестыми степенями некоторого числа z, следовательно достаточно перебрать z до 10^3 и посчитать подходящие варианты, для которых $a \le z^6 \le b$. При этом для подзадачи 1 можно, вместо этого, перебирать значения от a до b, проверяя, что оно является шестой степенью числа, либо просто заметить, что $z^6 \le 1000$ выполнено только для $z = 1$, $z = 2$ и $z = 3$.

В подзадаче 3 можно перебрать все потенциальные пары чисел от a до b и проверить, являются ли они x^2 и y^3 для некоторых x и y.

В подзадаче 4 можно перебрать все варианты значений x и y.

В подзадаче 5 можно перебирать y как в полном решении, но проверять все значения от L до R по отдельности на то, является ли оно квадратом.

В подзадаче 6, перебирая y как в полном решении, можно использовать для поиска подходящих значений x метод двух указателей.

Задача 3. Лифт

Автор задачи: Георгий Корнеев
Сложность задачи: средняя, многие подзадачи доступны широкому кругу участников
Основные темы: моделирование, структуры данных, обработка событий

Основная трудность при решении этой задачи — правильная организация хранения данных в программе.

Будем использовать алгоритм обработки событий. Создадим события для каждого факта «сотрудник подошёл к лифту», для каждого такого события запомним его время и номер сотрудника. Будем хранить события в структуре данных, поддерживающей добавление и извлечение элемента с минимальным ключом, в качестве ключа будем использовать время события. Подойдет, например, структура данных из стандартной библиотеки C++ std::set, либо самостоятельно реализованная структура данных «приоритетная очередь».

Кроме событий «сотрудник подошёл к лифту» нам потребуются события «лифт прибыл на интересный этаж». Интересным при движении вверх будет только этаж, на котором сделан активный вызов, а при движении вниз — каждый этаж ниже текущего положения лифта, на котором находится хотя бы один ожидающий лифта сотрудник, а также первый этаж. Для каждого этажа будем хранить множество ожидающих лифта сотрудников.

Лифт может находиться в трех состояниях:

- на первом этаже в ожидании вызова,
- движется вверх к активному вызову,
- движется вниз на первый этаж.

Будем рассматривать события в порядке возрастания их времени, при необходимости добавляя новые события. Рассмотрим действия при обработке очередного события для каждого

состояния лифта. Для любого состояния лифта, когда сотрудник подходит к лифту, будем добавлять этого сотрудника в множество ожидающих лифт на соответствующем этаже.

Если лифт находится на первом этаже и ожидает вызова, то единственный возможный тип события — сотрудник подошел к лифту. Пусть это происходит на этаже s. При обработке этого события данный вызов становится активным, а лифт переходит в состояние «движется вверх к активному вызову». Если текущее время — максимум из времени события и времени, когда лифт последний раз освободился на первом этаже, — равно t, то добавим событие «лифт прибывает на этаж s» с временем $t + s$.

Если лифт движется вверх к активному вызову, то при обработке события «лифт прибыл на интересный этаж» необходимо сделать следующее: добавим в очередь события «лифт прибыл на интересный этаж» для всех этажей, где находятся сотрудники, ожидающие лифт, которые находятся ниже текущего положения лифта. Если время текущего события t, а этаж, на котором находится лифт — s, то время для события «лифт прибыл на интересный этаж a» равно $t + s – a$. Добавим также событие для первого этажа. Затем переместим всех находящихся на текущем этаже сотрудников из множества ожидающих лифт на этом этаже в множество перемещающихся на лифте, и перейдем к обработке следующего события.

Если лифт движется вниз, то обработка события «сотрудник подошёл к лифту на этаже a», на котором в этот момент нет других сотрудников, происходит следующим образом. Если лифт в этот момент находится не ниже этажа a, то необходимо добавить в очередь событие «лифт прибыл на интересный этаж a». При обработке события «лифт прибыл на интересный этаж a», если $a > 1$, то все ожидающие лифт сотрудники на этом этаже перемещаются с этажа в лифт, а если $a = 1$, то все сотрудники из лифта помечаются как обработанные, а время их прибытия на первый этаж равно времени рассматриваемого события.

В заключение описания алгоритма отметим, что при сравнении событий с одинаковым временем события «лифт прибыл на интересный этаж» должны идти после событий «сотрудник подошёл к лифту», а события типа «сотрудник подошёл к лифту» следует обрабатывать в порядке возрастания номера сотрудника.

Для решения подзадачи 1 можно применить простую формулу: ответ для первого и единственного сотрудника равен $t_1 + 2(a_1 – 1)$.

Для решения остальных подзадач необходимо с различной эффективностью промоделировать описанный алгоритм обработки событий.

В подзадаче 2 возможно пошаговое моделирование каждой секунды описанного в условии процесса. При этом можно каждую моделируемую секунду произвольным разумным образом обрабатывать входные данные, например, просматривая всех сотрудников и проверяя, не находится ли он на этаже, где находится лифт.

В подзадаче 3 событий мало, поэтому каждый раз для обработки события можно выбирать очередное событие проходом по всем имеющимся событиям, использование быстрых структур данных не требуется.

В подзадаче 4, как и в подзадаче 2, возможно пошаговое моделирование процесса, но необходимо аккуратное хранение информации о сотрудниках, чтобы суммарное время работы было $O(T + n)$ или $O(T + n \log n)$, где T — максимальное время, в которое может произойти событие (порядка 10^8 в этой подзадаче).

Наконец для решения последней пятой подзадачи необходима эффективная реализация моделирования, описанного в разборе, время работы $O(n \log n)$.

Задача 4. Мониторинг труб

Автор задачи: Николай Будин

Сложность задачи: высокая, первые подзадачи доступны широкому кругу участников

Основные темы: деревья, алгоритмы на графах, динамическое программирование

Переформулировав задачу математически, получим следующее условие: задано корневое дерево, на каждом ребре которого написана буква. Требуется покрыть все ребра дерева словами из заданного множества, причем слово покрывает путь вниз по дереву, буквы на котором при прочтении вдоль пути образуют это слово.

У этой задачи есть два принципиально разных решения: первое использует алгоритм поиска остовного дерева минимального веса в ориентированном графе, а второе основано на методе динамического программирования на поддеревьях. Рассмотрим оба решения.

Решение 1. Построим вспомогательный ориентированный граф. Вершины графа будут совпадать с узлами заданного во входном файле дерева, а ориентированные взвешенные ребра построим следующим образом. Если, начав от вершины u можно пройти вниз по дереву по буквам слова s_i и попасть в вершину v, то добавим в граф ребро uv с весом w_i. Также для каждой вершины u добавим ребро веса 0 в ее родителя в исходном дереве p_u.

Легко убедиться, что минимальная стоимость проверки всех труб равна стоимости минимального ориентированного остова в получившемся графе, с корнем в корне исходного дерева. Для поиска минимального остова можно использовать алгоритм Чу Йонджина и Лю Цзенхонга, известный в русскоязычной литературе как «алгоритм двух китайцев». Его время работы в самой простой реализации составляет $O(n^3)$, что достаточно для этой задачи.

Отметим, что построение графа в подзадачах 1, 3 и 4 можно выполнить наивно, попытавшись отложить каждое слово от каждой вершины, получив сложность $O(n^2m)$. В остальных подзадачах требуется использовать структуру данных «бор». А именно, сложим в бор все заданные слова и при обходе дерева от каждой вершины будем помнить, в какой вершине бора мы находимся. При таком подходе время работы этой части решения составляет $O(\sum \text{len}(s_i) + n^2)$.

Решение 2. Как и в предыдущем предыдущем решении, найдем все пути, которые можно покрыть каким-либо словом.

Теперь воспользуемся методом динамического программирования. Рассмотрим какой-нибудь ответ на задачу. Это множество путей, которое покрывает все ребра дерева. Возьмем некоторое поддерево и рассмотрим подмножество путей из ответа, нижние вершины которых лежат в этом поддереве. Заметим, что такое подмножество путей покрывает все ребра поддерева и также некоторый путь выше по дереву до корня поддерева. Значит, за состояние алгоритма можно принять пару вершин (u, v), одна из которых находится выше другой: нижняя обозначает корень поддерева, которое покрыто целиком, а верхняя обозначает вершину в которой заканчивается покрытый путь из корня. Обозначим как $dp[u][v]$ минимальную стоимость покрыть поддерево вершины v так, чтобы одновременно покрыть путь от u до v.

Будем считать значения одновременно для всех состояний с фиксированным v. Для листа $dp[u][v]$ равно минимальной стоимости слова, которым можно покрыть путь от u до v, если такое есть. Для внутренней вершины найдем сначала значения для детей. Для каждого ребенка

Итоговое время работы $O(n^2)$.

Для решения подзадачи 1 заметим, что каждое ребро необходимо проверять независимо, а значит можно просто для каждой буквы выбрать минимальную стоимость, для которой слово совпадает с этой буквой, и просуммировать соответствующие стоимости для всех труб.

В подзадаче 2 возможны различные подходы, основанные на динамическом программировании на прямой, также можно, построив граф как указано в разборе, вместо минимального остовного дерева найти кратчайший путь от корня до листа.

В подзадаче 3 помимо описанных в разборе решений возможно решение перебором или динамическим программированием с состоянием «множество покрытых ребер».

Наконец, учитывая технические сложности восстановления ответа в обоих подходах к решению, решения, которые только находят минимальную стоимость покрытия проходят все подзадачи, кроме подзадачи 6.