Разбор задач XIII командного чемпионата Самарской области по информатике, математике и программированию.

A. Логика выбора

Задача носит сугубо технический характер: требуется аккуратно прочитать строки, а затем не менее аккуратно выполнить сравнение отношений вида <количество подходящих предметов>/<общее количество предметов> для каждой специальности. Для каждого из абитуриентов нужно найти максимальное отношение среди всех специальностей.

B. Оптимизация нагрузки

Эту задачу можно было решать жадным алгоритмом. Во-первых, получим отрезки вида [<день, в который задано домашнее задание> , <последний день, в который оно может быть выполнено>]. Отсортируем эти отрезки по левой границе (для хранения можно использовать список или массив) и далее поступим следующим образом. Организуем очередь с приоритетами, в которой отрезки будут упорядочены уже по правой границе. Затем будем вычерпывать из списка все домашние задания, которые заданы в некоторый день, и помещать их в очередь (в которой они будут переупорядочены). Из очереди же отрезки будем удалять, ориентируясь на количество пар, которые Николай в этот день свободен. Если мы успеем удалить из очереди все отрезки до того момента, когда их правая граница окажется меньше текущего дня, значит, Николай успеет выполнить все домашние задания. В противном случае успеть ему не удастся.

C. Коллоквиум.

Эта задача может быть решена с помощью ленивой динамики. Введем “состояние” студента – отсортированный набор из количеств вопросов, которые ему известны по каждой теме. Например, если в начале коллоквиума студент знает по два вопроса по двум темам, еще три из другой и по двум темам не знает ни одного вопроса (всего на коллоквиум вынесено пять тем), то его состояние мы можем описать как (0, 0, 2, 2, 3). Каждый раз, когда преподаватель просит его высказаться по какой-либо теме, в соответствующем наборе одно из чисел уменьшается на единицу. Заметим, что удобно (массив же отсортирован) уменьшать на единицу первое среди равных чисел (это позволяет избежать пересортировки массива).

Преподаватель выбирает тему случайным образом. Если он выберет одну из двух тем, по которым студент не знает ничего, то коллоквиум для студента завершится (это может произойти с вероятностью 2/5), в остальных же случаях он будет продолжен: с вероятностью 2/5 студент перейдет в “состояние” (0, 0, 1, 2, 3) и с вероятностью 1/5 – в “состояние” (0, 0, 2, 2, 2). Среднее количество ответов для состояния (0, 0, 2, 2, 3) будет получаться как 2/5 * (количество ответов из стартового “состояния” (0, 0, 1, 2, 3) + 1) + 1/5 * (количество ответов из стартового

“состояния” (0, 0, 2, 2, 2) + 1) (единица прибавляется потому, что на предыдущий вопрос студент успешно ответил).
Понятно, что “выбрасывая” по единице из состояний, мы неизбежно придем к состоянию (0, 0, 0, 0, 0), для которого очевиден ответ 0. В процессе выбрасывания единиц состояния могут повторяться, и, чтобы не перевычислять для них количество ответов, их следует сохранять.

Задача D. Лабораторные работы.
Эта задача требовала симуляции процесса сдачи лабораторных работ с аккуратным учетом приоритетов, согласно которым преподаватель выбирает очередного студента для сдачи лабораторных работ. Ограничения в задаче таковы, что какой-либо оптимизации не требовалось.

Задача E. Допуск к сессии.
Каждого студента можно поместить в вершину дерева (у любой вершины есть единственный предок – обучающий). В корне дерева будет помещен Павел.
Выполним поиск в глубину для каждого поддерева, чтобы определить его высоту, а затем отсортируем поддеревья по этим высотам. Понятно, что чем выше поддерево, тем раньше нужно приступить к его обучению, чтобы оно закончилось раньше.

Задача F. Шпаргалки
Будем решать эту задачу при помощи sqrt-декомпозиции. Разобьем изначальное множество на k групп примерно из N/k элементов.
Внутри каждой группы будем хранить следующую информацию: если из этой группы забрать i первых элементов, сколько получится строк и сколько места останется справа в последней строке. Кроме того, при помощи массива частичных сумм и бинарного поиска научимся определять, сколько первых элементов блока помещается в заданную ширину w <= L.
Теперь ответ мы можем посчитать так. Начнем с 1 строки и свободной ширины L. Последовательно для каждого блока будем спрашивать, сколько при заданной ширине свободного места в начальной строке блока получится строк внутри него и сколько останется в последней строке блока места. Высоты будем складывать. Отдельно надо обработать случай, когда весь блок целиком помещается в свободное место. Это будет сделано примерно за O(k*log(N/k)).
Теперь нам осталось научиться модифицировать блок по запросу. Начнем с расположения, когда участвуют все элементы блока. Будем выкидывать по одному элементы и перестраивать расположение, которое будет получаться после этого, сдвигая позиции переносов строк. Заметим, что к нам никогда не приходит запрос, который забирает элементов больше, чем в изначально построенной первой строке. Кроме того, заметим, что если мы будем останавливаться каждый раз, когда текущая строка не изменилась, то суммарное количество “перемещений”

элементов совпадает с размером блока (действительно, в конечный момент времени каждый элемент находится на одну строку выше, чем был в начальный момент). Таким образом, мы обновим всю необходимую информацию за O(размера блока), т.е. за O(N/k)
Посчитаем суммарную сложность. Это будет O(N + Q*(k*log(N/k)+N/k)). Выбрав константу разбиения порядка sqrt(N), получим O(N + Q * sqrt(N) * log (N)). При удачном подборе константы разбиения (на практике она зависит от скорости работы различных частей программы) и хорошо оптимизированном решении этого вполне достаточно.

Задача G. Сон
Нетрудно получить формулу (K + max(1, N/T) – 1) / max(1, N / T)

Задача H. Перед экзаменом.
Отсортируем вопросы по возрастанию сложности. Можно доказать, что наилучшим с точки зрения задачи распределение вопросов по билетам является следующее. Объединим самый легкий с самым сложным вопросом, второй по сложности с предпоследним и т.д. Никакое иное распределение как минимум не лучше описанного (нам подойдет любое). Теперь остается лишь посчитать требуемую разность и вывести список билетов.

Задача I. Экзаменационные задачи.
Эту задачу можно было решать, например, следующим образом. Отсортируем номера известных задач, после чего попробуем найти “зазор” либо между номерами, либо между границами диапазона и номерами.

Задача J. Елка
Будем поддерживать структуру данных, позволяющую быстро пересчитывать степени каждой вершины после очередного отсоединения очередной ветки. Как только степень вершины становится равной нулю, то ветку, оканчивающуюся ею, можно отсоединять. Все ветки, которые могут быть отсоединены в данный момент (и позже) удобно хранить, например, в упорядоченном множестве. Это позволит легко выбрать ветку с наименьшим номером.

Задача K. В поисках истины.
Воспользуемся следующими соотношениями.
Если x mod a = b и a mod c = 0, то x mod c = b mod c (действительно, если a делится на c, то остаток от деления x на c полностью определяется остатком от деления x на a).
Пусть известно, что
x mod a[i] = b[i]
x mod a[j] = b[j]

Обозначим a = gcd(a[i], a[j]) (наибольший общий делитель). Тогда из вышеперечисленных утверждений следует, что
x mod a = b[i] mod a
x mod a = b[j] mod a
Получаем два равенства с одинаковой левой частью. Очевидно, что правая часть должна быть одинакова. Проверим это для всех пар (i,j), если хотя бы для одной это не выполняется – ответ “NO”. Можно доказать, что иначе ответ “YES”.