Задача «Сквозь вселенные»

Темы: комбинаторика, арифметика остатков

Первым делом требовалось заметить, что ответ — количество перестановок из n элементов, а это просто $n!$ (произведение всех чисел от 1 до n), так как у нас есть n способов выбрать первую вселенную, $n-1$ вторую и так далее.

В случае, если $n \geq m$, то в числе $n!$ присутствует множитель m, а значит остаток от деления $n!$ на m равен нулю. Если же $n < m$, то достаточно промоделировать процесс, беря остаток от деления на m при каждом умножении на следующее число.

Задача «Лепрекон»

Темы: битовые операции, арифметика

Для случаев $n \leq 100$ можно для каждого номинала считать «$\widehat{\text{XOR}}$» всех номиналов кроме него при каждом запросе и сравнивать результат с запросом.

Добиться работы при $n \leq 10000$ можно было подсчитав «$\widehat{\text{XOR}}$» для всех отрезков (например динамическим деревом отрезков или декартовым деревом) и затем проверять для каждого номинала за $O(log(n))$ интересен ли он. К сожалению такое решение никак не помогало продвинуться к полному.

Обозначим «$\widehat{\text{XOR}}$» всех элементов массива за y. «$\widehat{\text{XOR}}$» всех номиналов кроме монеты номиналом x, равен y «$\widehat{\text{XOR}}$» x. Утверждается, что если $y = 0$, то все монеты счастливые. Действительно, какую бы монету с номиналом x мы не взяли, x «$\widehat{\text{XOR}}$» 0 («$\widehat{\text{XOR}}$» всех монет, кроме x) будет равен x, т.е. любая монета будет счастливой.

Таким образом, для полного решения было достаточно поддерживать «$\widehat{\text{XOR}}$» всех элементов массива. При изменении номинала монеты необходимо было применить операцию «$\widehat{\text{XOR}}$» к сумме и старому значению номинала монеты, а затем к этой же сумме применить операцию «$\widehat{\text{XOR}}$» и новое значение номинала монеты.

Задача «Даша и сериалы»

Темы: деревья, динамическое программирование

Заметим, что интересные моменты образуют дерево, подвешенное за первую вершину. Для небольших n можно перебрать все варианты просмотренных моментов и выбрать минимальный по количеству моментов среди удовлетворяющих условию задачу. Для проверки удобно для каждой вершины дерева сохранить её предка и для каждой интересной вершины проверить, что все её предки, вплоть до корня дерева, просмотрены.

В случае $a_i = i$ события развиваются линейно, а значит необходимо просмотреть все моменты, вплоть до момента с наибольшим номером (найти самую глубокую вершину в вырожденном дереве).

В полном решении подниматься по дереву от каждой интересной Даше вершины, прекращая подъём в случае, если мы попали в уже посещенную вершину. Ответом будет являться количество посещенных вершин. Действительно, в дереве существует единственный путь от корня до любой из вершин и для каждой из интересных Даше моментов в ответ должен быть включен весь путь от корня, до вершины соответствующей этому моменту. В случае, если при подъёме мы попадаем в уже посещенную вершину, то подниматься выше не нужно, т.к. все вершины, лежащие выше, уже помечены проходом из другого интересного момента.

Задача «Король Ночи»

Темы: эффективная сортировка, сортировка слиянием, сор-

тировка подсчетом, два указателя, двухпроходные алгоритмы

Задача могла быть решена на полный балл одним из двух способов.

Первый из них представляет собой модификацию сортировки слиянием, где вместо двух сливаемых последовательностей используется n. Сначала возьмем из каждого королевства самого низкого человека и оценим сумму разностей в росте (она же равна разнице между самым высоким и самым низким человеком в наборе). Чтобы получить следующий потенциально подходящий набор людей, необходимо в королевстве, от которого представлен самый низкий человек среди всех n людей, взять следующего по росту человека. Доказательство корректности алгоритма аналогично доказательству алгоритма сортировки слиянием. Для восстановления ответа необходимо реализовать двухпроходный алгоритм. На первом проходе запоминается минимальная разность в росте, а на втором проходе при первом же возникновении такой же разности в росте необходимо вывести рост всех людей в текущем наборе и прекратить работу. Сложность такого алгоритма равна $O(n \times m \log m + n \times m)$.

Второй способ предполагает, что мы добавим всех людей в один массив и отсортируем их по росту (запоминая номер королевства для каждого из людей). После этого можно воспользоваться методом двух указателей. Нам необходимо, чтобы между двумя указателями L и R находилось хотя бы по одному представителю от каждого королевства. Установим L на начало массива, а R будем двигать до тех пор, пока между L и R не будет представителей каждого королевства. Чтобы осуществлять такую проверку быстро можно воспользоваться, например, модифицированной сортировкой подсчетом, где для каждого королевства будет считаться количество представителей этого королевства между двумя указателями. Необходимо поддерживать счетчик количества нулевых значений на таком массиве сортировки подсчётом. После того, как правый указатель сдвинут на нужную позицию, мы можем оценить разницу в росте между самым высоким и самым низким человеком между указателями и обновить минимальную разность. После этого необходимо передвинуть левый указатель на один элемент вправо (пересчитав массив для сортировки подсчетом и количество нулевых элементов в нём) и двигать правый указатель до тех пор, пока счетчик нулевых элементов не станет равным нулю. Такой способ гарантирует нам, что на позиции R будет находится единственный представитель какого-либо королевства. Зафиксированное значение R не будет меняться до тех пор, пока L не достигнет также единственного представителя своего королевства. Эти два человека определяют разницу в росте, а представители всех других королевств гарантированно находятся между указателями. Восстановление ответа также делается за два прохода, аналогично предыдущему варианту. Сложность такого решения составляет $O(n \times m \times \log nm + n \times m)$.

Частичные баллы получали решения, использующие квадратичную сортировку или однопроходное сохранение ответа (когда текущий набор людей копировался в ответ при каждом обновлении минимальной разности).

Задача «Щелчок»

Темы: дерево отрезков, корневая декомпозиция

Частичным решением задачи является простое моделирование, сложность такого решения равна $O(n \times m)$.

В полном решении необходимо реализовать две эффективные структуры данных, позволяющие осуществлять операции на отрезке.

Будем хранить две структуры для четных и нечетных индексов. На каждой из двух

структур требуется выполнять два вида операций: инвертирование элементов на отрезке и запрос количества единиц на отрезке. Для этого пригодны многие структуры: корневая декомпозиция, bitset, дерево отрезков и т.п., в зависимости от реализации они набирают разное количество баллов. Для определения того, в какой структуре необходимо производить модификацию, необходимо определить четность левой границы запроса инвертирования. Запрос суммы на отрезке выполняется на каждой из структур (с аккуратно подсчитанными индексами), результаты запросов суммируются.