Задача A. Сумма

Посмотрим на выражение $A-B$. Во-первых, его значение обязательно должно быть четным, так как равно $2\cdot(x_1+x_2+\ldots+x_n)$. Во-вторых, так как в этом выражении каждое слагаемое натуральное, его значение должно быть не меньше, чем $2\cdot n$. В случае, когда n положительно, достаточно сделать эти две проверки. В случае, когда n равно нулю, нужно проверить A и B на равенство.

Задача B. Покраска

Обозначим ответ за $f(n,m)$ (для однозачности давайте считать, что $n \geqslant m$, в противном случае аргументы функции меняются местами). Тогда по условию $f(n,m) = f(n-m,m)+1$, а также $f(n,n)=1$. Требовалось написать программу, вычисляющую такую рекуррентную формулу. Можно заметить, что вычисления в исходном виде достаточно объемные, но их можно быстро сжать, если группировать операции до тех пор, пока $n > m$. А именно, формулу пересчета можно переписать как $f(n,m) = f(n \pmod m, m) + \lfloor \frac{n}{m} \rfloor$. Такая формула имеет такую же вычислительную сложность $O(\log n + \log m)$, как и алгоритм Евклида, чего хватает для решения задачи.

Задача C. Операции

Можно доказать, что правильный порядок вытаскивания элементов для Пети — порядок по возрастанию. Таким образом, можно проверить, хватает ли какого-то значения X на позицию x_0 моделированием всей последовательности. Для того, чтобы найти итоговое оптимальное значение, можно воспользоваться бинарным поиском по ответу.

Задача D. Конь

В задаче требовалось реализовать динамическое программирование $dp_{i,j}$— количество способов дойти до клетки (i,j). Начальное значение $dp_{1,1}=1$, переходы $dp_{i,j} = dp_{i-1,j-2} + dp_{i-2,j-1}$. Ответ лежит в значении $dp_{n,m}$.

Задача E. Горы

Назовем прыжком комбинацию переходов $i \to i-a_i \dashrightarrow i-a_i+b_{i-a_i}$.

Введем динамическое программирование dp_i — минимальное количество ходов, нужное для достижения 0, если начать в i.

Мы знаем, что $dp_i = 1 + \min(dp_j + b_j)$, для $i-a_i \leqslant j \leqslant i$. Эту динамику можно считать в обратную сторону. А именно, если из состояния достижим 0, то его dp равна 1. Если 0 не достижим, но достижимо состояние, из которого достижим 0, то dp равно 2, и так далее.

Таким образом, такой пересчет симулирует ни что иное, как поиск в ширину.

Рассмотрим пересчет bfs на стадии, когда все расстояния от 0 до d посчитаны. Возьмем состояние v ($dp_v = d$) и все такие u, что $u+b_u = v$. Тогда для всех j, таких что $j-a_j \leqslant u \leqslant j$ имеем $dp_j = d+1$. Сохраним в дереве отрезков на минимум для каждой позиции i значение $i-a_i$. Тогда надо на суффиксе массива перебрать все элементы со значением меньше u.

Перебирать элементы можно явно, так как использованные элементы второй раз смотреть не нужно — их можно заменить на нейтральный элемент.

Итоговая асимптотика $O(n \log n)$