# Есть n стульев...

*Автор задачи и разработчик: Владислав Власов*

Заметим, что любой набор выгодно рассматривать в отсортированном по высоте порядке, иначе разница высот некоторой пары соседних стульев будет больше, чем если бы стулья были отсортированы. Отсортируем стулья по высоте, теперь нас интересует некоторая подпоследовательность полученного массива.

Заметим, что на самом деле нас интересует не просто подпоследовательность, а отрезок массива. Если бы оптимальным ответом была подпоследовательность не идущих подряд стульев, можно было бы добавить все «пропущенные», и суммарная ширина бы увеличилась, а максимальная разность высот между соседними — уменьшилась бы.

Для решения **первой подгруппы** достаточно перебирать все возможные отрезки и проверять каждый за линейное время. Такое решение работает за $\mathcal{O}(n^3)$.

Решение **второй подгруппы** похоже на решение первой, но для этой подгруппы нужно быстро обрабатывать каждый отрезок. Чтобы находить суммарную ширину стульев на отрезке, можно воспользоваться префиксными суммами, а чтобы находить максимальную разность соседних по высоте — либо реализовать операцию «минимум на отрезке» с поддерживающих ее структур данных, либо просто перебирать отрезки по левой границе, а при фиксированной левой границе — по возрастанию правой, одновременно обновляя максимум при расширении отрезка. Время работы такого решения — $\mathcal{O}(n^2)$.

В **третьей подгруппе** нам достаточно проверять отрезки длины ровно $H$, поскольку именно столько необходимо, чтобы Влад поместился, а расширение отрезка не может уменьшить его неудобность. Таким образом решение сводится к задаче «максимум в окне», которую можно решать за $\mathcal{O}(n \log n)$ с помощью `std::set` или же за $\mathcal{O}(n)$ с помощью очереди с поддержкой минимума.

В **четвертой подгруппе** ответ не превосходит 30. Тогда можно перебрать ответ и проверить, можно ли получить ответ не хуже такого, за $\mathcal{O}(n)$. Для проверки будем поддерживать текущий отрезок и идти слева-направо, жадно набирая элементы. Как только какой-то элемент не получается добавить из-за большой разности с предыдущим, начнем новый отрезок с него.

Есть несколько вариантов **полного решения**. Можно воспользоваться идеей четвертой подгруппы, заменив перебор ответа бинарным поиском по ответу. Альтернативно можно воспользоваться идеей второй и третьей подгрупп, но сократить перебор отрезков до $\mathcal{O}(n)$ с помощью техники двух указателей, поддерживая для фиксированной левой границы ближайшую правую, суммарная ширина стульев между которыми достаточна. Максимальную разность на отрезке можно поддерживать тем же `std::set`.