Для начала поймем, когда нельзя построить перестановку по данным ограничениям. Для этого, построим граф, где вершинами будут индексы элементов перестановки, а ребра построим по слежующему правилу: для каждого ограничения из каждого элемента из отрезка $[a_i; b_i]$ проведем ориентированное ребро в каждый элемент из отрезка $[c_i; d_i]$. У нас получился полный граф сравнений. Заметим, что если в данном графе есть цикл, то получается, что существует какой-то элемент x который меньше самого себя. Тогда по данным ограничениям построить перестановку невозможно.

Иначе, так как наш граф не имеет циклов, то мы можем построить топологическую сортировку и расставить числа в порядке, обратном топологической сортировке. Данное решение работает за $\mathcal{O}(n + L^2)$

Теперь для каждого индекса перестановки будем поддерживать количество еще необработанных ограничений, где данный индекс лежит в отрезке $[c_i; d_i]$. Также будем поддерживать очередь из индексов, в которых данная величина(для удобства назовем её cnt_i) достигла 0. Также для каждого ограничения будем поддерживать величину $cntseg_i$ — количество элементов из отрезка $[a_i; b_i]$, которые еще не обработаны. Теперь рассмотрим очередной элемент x из очереди. Поставим на позицию x минимальное непоставленное число. Теперь рассмотрим все ограничения, где x лежит в отрезке $[a_i; b_i]$. У каждого такого отрезка уменьшим $cntseg_i$ на 1, и, если $cntseg_i$ стало равно 0, то у каждого элемента y из отрезка $[c_i; d_i]$ вычтем 1 из cnt_y. Если cnt_y стало равно 0, то добавим y в очередь.

Заметим, что если после конца работы алгоритма остались необработанные элементы, то в графе сравнений есть цикл. Также несложно заметить, что мы рассматриваем элементы в порядке, обратном топологичской сортировке.

Теперь оценим асимптотику: каждый элемент мы добавим и удалим из очереди ровно 1 раз. Для каждого элемента мы обновляем $cntseg$ за количество отрезков, в которых он находится в отрезке $[a_i; b_i]$, т.е. суммарно мы обновим $cntseg$ не больше, чем $\mathcal{O}(L)$ раз. Также cnt мы обновим не больше, чем $\mathcal{O}(L)$ раз из тех же соображений. Следовательно алгоритм будет работать за $(O)(n + L)$.