9.149
- w_i - произвольная перестановка
- $y(w_i)$ - количество неподвижных точек в ней
- $p(w_i) = \frac{1}{n!}$ - вероятность случайной перестановки
- $E(y) = \sum\limits_i y(w_i)p(w_i) = \frac{1}{n!}\sum\limits_i y(w_i)$
- Будем считать общее количество неподвижных точек во всех перестановках по рядам.
- Мы нарисуем двудольный граф, где слева все точки, а справа все перестановки, где ребро есть между точкой и перестановкой только,
  когда точка неподвижна в данной перестановке. Количество ребер равно необходимому количеству. Кол-во ребер можно посчитать используя только левые точки.
- Зафиксируем a_i, возможных перестановок, где a_i будет неподвижной точкой $(n-1)!$, включая те, где больше одной неподвижной точки.
- $n(n-1)!$ - необходимое количество $\implies E(y) = 1$

10.152
- ```
  for i in range(n):
      for j in range(k):
          dp[i + j + 1] = min(dp[i + j + 1], dp[i] + a[i + j + 1])
  ```

10.153
- На i-том шаге поддерживаем очередь на минимум из предыдущих k элементов
- Очередь на двух стеках
  - Элемент очереди: (элемент, минимальный элемент в этом стеке среди элементов добавленных ранее)
  - Добавление: (элемент, q1.front || элемент)
- При переходе добавляем элемент в очередь и удаляем q.front
- Очередь работает за $\mathcal{O}(1)$ так как каждый элемент добавляется и удаляется только 1 раз, в итоге 3n операций

10.154
- 1. $dp[i-1] > dp[i-2] \implies dp_{path}[i] = dp_{path}[i-2]$
- 2. $dp[i-1] < dp[i-2] \implies dp_{path}[i] = dp_{path}[i-1]$
- 3. $dp[i-1] == dp[i-2] \implies dp_{path}[i] = dp_{path}[i-1] + dp_{path}[i-2]$

10.155
- 1. $dp[i-1] > dp[i-2] \implies dp_{path}[i] = dp_{path}[i-2]$
- 2. $dp[i-1] < dp[i-2] \implies dp_{path}[i] = dp_{path}[i-1]$
- 3. $dp[i-1] == dp[i-2] \implies dp_{path}[i] = dp_{path}[i-1] + dp_{path}[i-2]$

10.156
- На i-том шаге ищем лексиграфически минимальный префикс среди $[0, i-1]$
- $dp[i] = findMinPrefix(0, i-1) + a_i$

10.157
- Поддерживаем dp_{even}, dp_{odd}, каждый элемент которых представляет максимальную сумму в данном элементе.
- a_i - odd $\implies$

$$dp_{even}[i][j] = max(dp_{odd}[i-1][j], dp_{odd}[i][j-1]) + a_i$$
$$dp_{odd}[i][j] = max(dp_{even}[i-1][j], dp_{even}[i][j-1]) + a_i$$

- Аналогично с a_i - even
- Сумма максимальна так как мы всегда прибавляем к максимальной
  ```