Студент: Маргарита Чудова
Группа: 2
Дата: 27 апреля 2022 г.

3. Сделаем все то же самое, что в НВП, но будем искать максимум суммы и проверять заданные условия на индексы.
   Пусть $d[i]$ — наибольшая сумма последовательности, удовлетворяющей условиям задачи и заканчивающейся в элементе a_i, $d[0] = a_0$: $p[i]$ — массив, который хранит номер предыдущего элемента последовательности. Псевдокод:

```
for i in range(n):
    for j in range(i-1):
        if l <= |i - j| <= r:
            if d[i] < dp[j] + a[i]:
                d[i] = dp[j] + a[i]
                p[i] = j
            #иначе последовательность никак не изменилась
            #значит не нужно ничего делать с d[i], p[i]
```

   Как восстановить последовательность? Найдем номер максимального элемента в $d[i]$, назовем его t. $a[t]$ — это последний элемент нужной нам последовательности, а в $p[t]$ лежит номер предыдущего элемента последовательности, можем дальше переходить к нему. Псевдокод:

```
seq = []
while t >= 0:
    seq.append(a[t])
    t = p[t]
```

4. Заведем массив $d[i][j]$, который будет хранить длину наибольшего общего префикса $s[i:]$ и $s[j:]$. Ясно, что $d[n-1][n-1] = 1$, также можем посчитать значения $d[i][n-1]$ и $d[n-1][i]$ — это ноль, если $s[i] \neq s[n-1]$ и единица, если они равны.
   Если мы хотим добавить по одному элементу к каждому префиксу, то нам нужно будет сравнить эти элементы и добавить единицу к длине исходного общего суффикса, если же они не равны, то общего суффикса нет в приципе, его длина 0.

   Псевдокод по этим рассуждениям:

```
d[n][n] = 1
for i in range(n):
    if s[i] == s[n-1]:
        d[i][n-1] = 1
        d[n-1][i] = 1
    else:
        d[i][n-1] = 0
        d[n-1][i] = 0

for i in range(n-2, -1, -1):
```

```
        for j in range(n-2, -1, -1):
            if s[i] == s[j]:
                d[i][j] = 1 + d[i+1][j+1]
            else:
                d[i][j] = 0
```

5. Пусть $d[i][j]$ — вероятность выбросить i орлов при условии, что подброшено j первых монет. Зададим граничные условия. Понятно, что

$$d[0][j] = \prod_{k=1}^{j}(1 - p_k),\ \forall j$$

$$d[0][0] = 1,\ d[i][0] = 0$$

Рассмотрим некоторое $d[i][j]$. У нас может быть 2 варианта: либо при подбрасывании $j-1$ предыдущей монеты мы получили нужное количество орлов — тогда при подбрасывании j-й монеты нам нужно, чтобы орел не выпал, либо до подбрасывания этой монеты было выброшено $i-1$ орлов — тогда нужно, чтобы орел выпал. В вероятности нам нужно учесть оба исхода. Тогда:

$$d[i][j] = (1 - p_i) \cdot d[i][j-1] + p_i \cdot d[i-1][j-1]$$

Таким образом сможем заполнить весь массив за $\mathcal{O}(nk)$. Ответом будет $d[n][k]$.

6. не вышло :(

7. Пусть $d[i][j]$ — количество i-значных чисел с суммой j. Зададим граничные условия:

$$\forall i : d[i][0] = 1, \forall j \neq 0 : d[0][j] = 0$$

Как получить количество чисел $d[i][j]$ через предыдущие $i-1$? Чтобы получить сумму j, можем прибавлять к предыдущим суммам числа от 0 до 9, будем брать соответствующие d. Максимальная сумма равна $9n$. Таким образом, получаем псевдокод:

```
for i in range(1, n+1):
    for j in range(1, 9n+1):
        for prev in range(max(0, j-9), j):
                d[i][j] += d[i-1][prev]
```

Итого в последнем столбце получим количества чисел для всех возможных сумм. Половинки билета из n чисел независимы друг от друга, таким образом, всего вариантов для каждой суммы j будет $d[n][j] \cdot d[n][j]$, все это надо сложить. Псевдокод:

```
s = 0
for k in range(9n+1):
    s += d[k][n]*d[k][n]
```

S и будет нашим ответом.