Разбор задач

третий этап республиканской олимпиады по информатике, январь 2021

Члены жюри Минской области,
студенты БГУИР:

Камеко Валерий,
Северин Клим,
Яценко Станислав

День 1

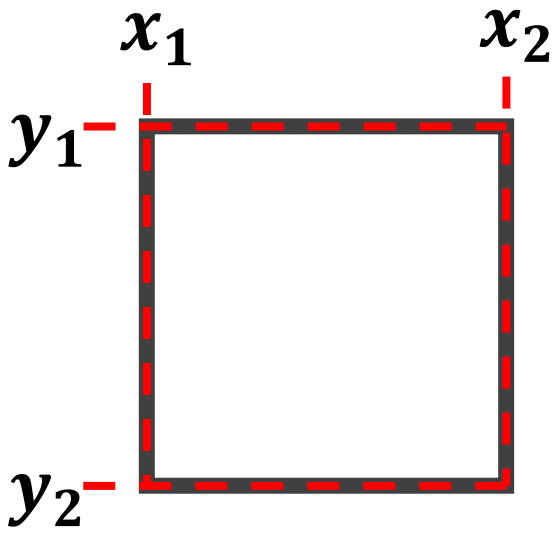
Задача 1. Космодром

Задача 1. Космодром

- Есть матрица, в некоторых клетках которой установлен забор.
- Даны координаты клеток с забором.
- Требуется определить образуют ли все эти клетки с забором квадратную оболочку, если да, то вывести координаты левой верхней клетки и сторону квадрата, иначе вывести -1.

Задача 1. Космодром

- Рассмотрим ситуацию, когда у нас задан квадрат, для которого ответ не -1.
- Заметим, что существуют заборы у которых координаты x совпадают. Также эти заборы задают левую или правую сторону квадрата.
- Аналогично и с координатой y.



Задача 1. Космодром

- Пусть $\boldsymbol{cnt\_up}$ – количество клеток, которые имеют координату $\mathbf{y} = \mathbf{y_1}$;
- $\boldsymbol{cnt\_down}$ – количество клеток, которые имеют координату $\mathbf{y} = \mathbf{y_2}$;
- $\boldsymbol{cnt\_left}$ – количество клеток, которые имеют координату $\mathbf{x} = \mathbf{x_1}$;
- $\boldsymbol{cnt\_right}$ – количество клеток, которые имеют координату $\mathbf{x} = \mathbf{x_2}$;

- Тогда для квадратной области состоящей из всех клеток должны выполняться следующие условия:
  - $\boldsymbol{cnt\_up} = \boldsymbol{cnt\_down} = \boldsymbol{cnt\_left} = \boldsymbol{cnt\_right}$;
  - $\boldsymbol{n + 4} = \boldsymbol{cnt\_up} + \boldsymbol{cnt\_down} + \boldsymbol{cnt\_left} + \boldsymbol{cnt\_right}$ // за исключением когда n = 1

Задача 1. Космодром

- Проверяем условия и если все хорошо, то в ответ можно вывести $\boldsymbol{x_1}$ $\boldsymbol{y_1}$ $\boldsymbol{cnt\_up}$, иначе **-1**.
- Сложность $\boldsymbol{O(n)}$.

Задача 2. Игра с числами

Задача 2. Игра с числами

- Дан массив $\boldsymbol{a}$ состоящий из $\boldsymbol{n}$ элементов.
- Массив $\boldsymbol{b}$ состоящий из $\boldsymbol{k}$ элементов, $\boldsymbol{1 \leq\ b_i \leq 2}$ – определяет кто будет ходить Петя или Petya++.
- На своем ходу игрок может выбрать любой элемент из массива $\boldsymbol{a}$ и изменить его знак на противоположный.
- Задача Пети – максимизировать сумму, Задача Petya++ – минимизировать сумму.
- В ответ вывести сумму элементов массива $\boldsymbol{a}$.

Задача 2. Игра с числами

- Воспользуемся «жадным» подходом.
- Заметим, что если наша задача увеличить сумму, то имеет смысл изменять знак только у отрицательных чисел. Если же мы хотим уменьшить сумму, то следует менять знак у положительного числа.
- При этом нам выгоднее брать числа, модуль которых больше, так как они сильнее повлияют на сумму.
- Тогда:
  - Если нам нужно максимизировать сумму, то нам будет выгодно за свой ход взять самое минимальное **отрицательное** число и изменить его знак.
  - Если нам нужно минимизировать сумму, то нам будет выгодно взять самое большое **положительное** число и изменить его знак.

Задача 2. Игра с числами (44 балла)

- Проходимся по массиву $\boldsymbol{b}$ и если текущий ход Пети, то ищем за $O(\boldsymbol{n})$ минимальный элемент. Если он отрицательный, то меняем его знак. Если же текущий ход Petya++, то ищем максимальный элемент и, если он положительный, то меняем его знак.
- В конце считаем сумму массива $\boldsymbol{a}$.
- Сложность $O(\boldsymbol{n^2})$.

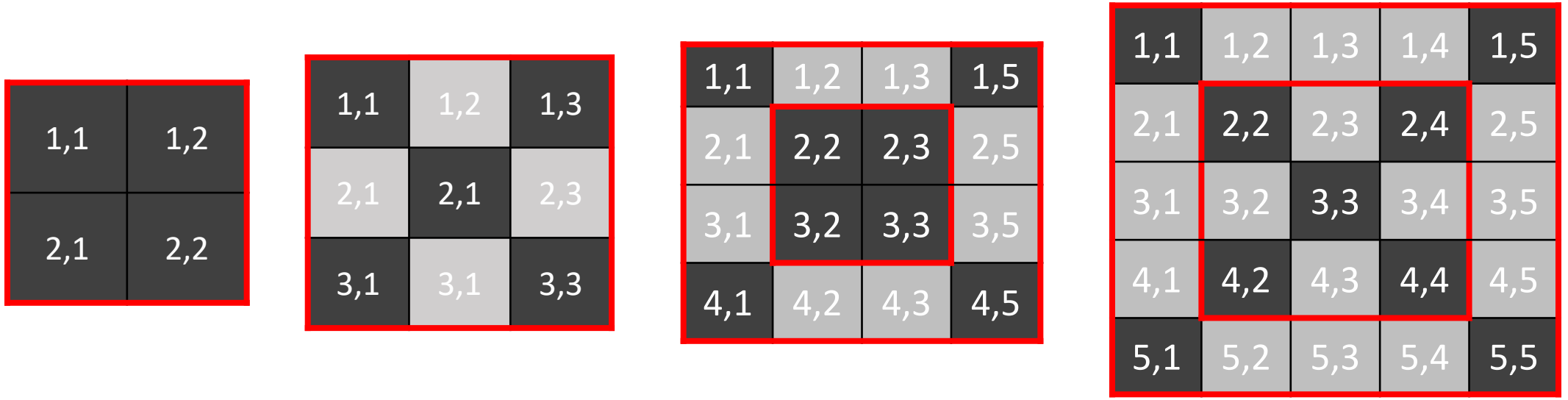
Задача 2. Игра с числами (100 баллов)

- Ход решения точно такой же.
- Для нахождения минимума и максимума будем использовать какую-нибудь структуру данных, которая будет позволять делать это за $\boldsymbol{O(\log n)}$. Например АВЛ-дерево.
- Сложность $\boldsymbol{O(n * \log n)}$.

Задача 3. Вражеские шпионы

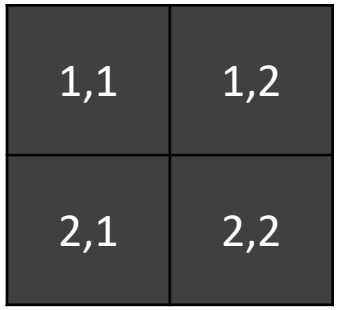
Задача 3. Вражеские шпионы

- На вход подается прямоугольная матрица $n * m$ с закрашенными «#» и не закрашенными «.» клетками.
- Требуется посчитать количество *подозрительных* подматриц.
- Подозрительными являются подматрицы вида:

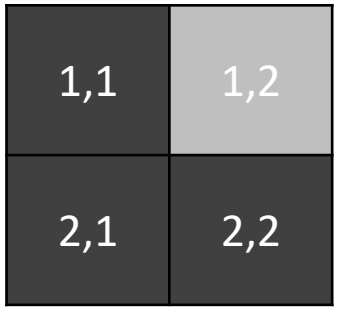


Задача 3. Вражеские шпионы (3 балла)

- Решение задачи для ограничений $n, m \leq 2$.
- Необходимо проверить является ли матрица размера (2x2) подозрительной.
- Проверка четырех значений массива.



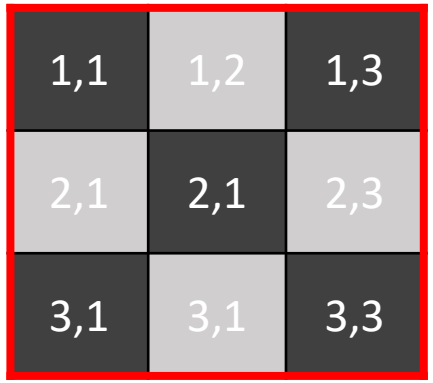
Подозрительная матрица



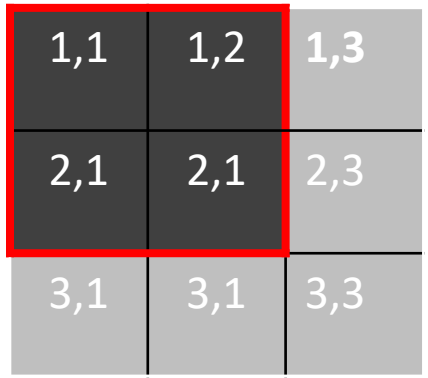
Не подозрительная матрица

Задача 3. Вражеские шпионы (4 балла)

- Решение задачи для ограничений $n, m \leq 3$.
- Необходимо проверить является ли матрица размера 3x3 подозрительной или одна из четырех матриц 2x2.
- Необходимо написать 5 условий.



Одна подозрительная подматрица



Одна подозрительная подматрица

Две подозрительные подматрица

Не подозрительная матрица

Задача 3. Вражеские шпионы (18 баллов)

- Частично решить задачу можно организовав полный перебор и проверку всех подматриц матрицы.
- За $O(n^2)$ переберем все клетки матрицы – левый верхний угол “креста”.
- За $O(n)$ перебираем все размеры “крестов”.
- За $O(n^2)$ проверяем чтобы диагонали “крестов” были закрашены, а остальные клетки подматрицы чистыми.
- Для “крестов” четного и нечетного размера будет отличатся формула для проверки диагоналей.
- Итоговая асимптотика $O(n^5)$.

Задача 3. Вражеские шпионы (100 баллов)

- Полное решение является оптимизацией частичного решения 3.
- Для решения задачи необходимо научится быстро (за $O(1)$) проверять подматрицу на подозрительность, для этого необходимо проверить главную и побочную диагонали, и отсутствие закрашенных клеток вне диагоналей.

Задача 3. Вражеские шпионы (100 баллов)

- Последним условием подозрительной подматрицы, необходимо проверить отсутствие закрашенных клеток кроме диагоналей.
- Для этого необходимо посчитать количество закрашенных клеток в подматрице и сравнить с длинами двух диагоналей.
- Сделать это можно с помощью префиксной суммы на матрице.

Префиксные суммы

- Идея префиксных сумм проста донельзя. Запишем сумму на отрезке [L; R] в таком виде:

$$\sum_{i=L}^{R} a_i = \sum_{i=0}^{R} a_i - \sum_{i=0}^{L-1} a_i$$

- Пример:

1	2	3	4	5	6	7	8	9	10
1	3	6	10	15	21	28	36	45	55

- Сумму элементов 5-7 можно посчитать как сумму элементов 1-7 минус элементы 1-4. (28 – 10 = 5 + 6 +7)
- Подробнее про одномерные и двумерные префиксные суммы https://brestprog.by/topics/prefixsums/

Задача 3. Вражеские шпионы (100 баллов)

- Для проверки диагоналей посчитаем префиксную сумму, это позволит находить число закрашенных клеток на диагонали за $O(1)$.
- Для префиксной суммы основных диагоналей формула:

$$d_1[i][j] = d_1[i-1][j-1] + matrix[i][j]$$

  - Где d_1 матрица префиксных сумм основных диагоналей, $matrix$ – матрица закрашенных клеток.

- Для проверки закрашенных клеток на диагонали подматрицы:

$$Size = d_1[y_1][x_1] - d_1[y_0 - 1][x_0 - 1]$$

  - x_1, y_1 – координаты правого нижнего угла проверяемой подматрицы;
  - x_0, y_0 – координаты левого верхнего угла проверяемой подматрицы;
  - $Size$ – размер проверяемой подматрицы(который равен длине диагонали).

1,1	**1,2**	**1,3**	**1,4**
2,1	2,2	2,3	2,4
3,1	3,2	3,3	3,4
4,1	4,2	4,3	4,4

Задача 3. Вражеские шпионы (100 баллов)

- Для префиксной суммы побочных диагоналей формула:

$$d_2[i][j] = d_2[i - 1][j + 1] + matrix[i][j]$$

  - d_2 матрица префиксных сумм побочных диагоналей;
  - $matrix$ – матрица закрашенных клеток.
- Обход по матрице для подсчета префиксной суммы побочных диагоналей необходимо делать в направлении сверху вниз, справа налево.
- Для проверки закрашенных клеток на побочной диагонали подматрицы:

$$Size = d_2[y_1][x_0] - d_1[y_0 - 1][x_1 + 1]$$

  - x_1, y_1 – координаты правого нижнего угла проверяемой подматрицы;
  - x_0, y_0 – координаты левого верхнего угла проверяемой подматрицы;
  - $Size$ – размер проверяемой подматрицы (который равен длине диагонали).

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4
3,1	3,2	3,3	3,4
4,1	4,2	4,3	4,4

Задача 3. Вражеские шпионы (100 баллов)

- Исходя из вышеперечисленного мы научились быстро отвечать на запрос является ли подматрица с координатами i, j, размера $size$ подозрительной.
- Перебирая координаты центра (или верхнего левого угла) исследуемой подматрицы и ее размер мы можем ответить подозрительная ли она.
- Подозрительная матрица размера $size$ (5x5) будет включать $\lfloor size \ / \ 2 \rfloor$ (2) подозрительных подматриц (5x5 и 3x3).
- Таким образом нам необходимо найти максимальный размер подматрицы и она будет включать подматрицы меньших размеров.

Задача 3. Вражеские шпионы (100 баллов)

- При подборе “крестов” четных и нечетны размеров, формула для проверки диагоналей будет отличаться.
- Итоговая асимптотика алгоритма составит $O(n^3)$, однако алгоритм будет работать за время $O(n^2)$ так как подбор размера “креста” будет сразу прекращаться если мы не находимся в центре “креста”.

Задача 4. Мощный процессор

Задача 4. Мощный процессор

- Дан массив $\boldsymbol{a}$ состоящий из $\boldsymbol{n}$ элементов, при этом $0 \leq \boldsymbol{a_i} \leq \boldsymbol{7}$.
- Требуется выполнить $\boldsymbol{q}$ запросов двух типов:
  - $\boldsymbol{xor\ l_j\ r_j\ v_j}$ – заменить элементы $\boldsymbol{a_i}$ на $\boldsymbol{a_i} \otimes \boldsymbol{v_j}$ для на отрезке от $\boldsymbol{l_j}$ до $\boldsymbol{r_j}$.
  - $\boldsymbol{seq\ l_j\ r_j}$ – вывести длину наибольшей возрастающей (НВП) последовательности на отрезке от $\boldsymbol{l_j}$ до $\boldsymbol{r_j}$.

Задача 4. Мощный процессор (22 балла)

- Ограничения на группу – $\boldsymbol{n, q \leq 1000}$.
- Для каждого запроса типа $\boldsymbol{xor}$ изменяем проходим по заданному отрезку $\boldsymbol{l_j\ r_j}$ и изменяем значения элементов $\boldsymbol{a_i}$ на $\boldsymbol{a_i \otimes v_j}$.
- При поступлении запроса на подсчет НВП проходим по отрезку поддерживая массив $\boldsymbol{cnt_k}$ – текущие длины НВП которые заканчиваются на число $\boldsymbol{k}$, при этом обновляя по формуле $\boldsymbol{cnt_k = max(cnt_k, cnt_h + 1)}$, для $\boldsymbol{h < k}$. Ответ на запрос – $\boldsymbol{max(cnt_k)}$.
- Сложность $O(\boldsymbol{nq})$.

Задача 4. Мощный процессор (34 балла)

- Ограничения на группу – $\boldsymbol{a_i \leq 1}$.
- Заметим, что НВП может быть либо длины 1, либо длины 2 и при этом равна последовательности $\boldsymbol{0, 1}$.
- Построим дерево отрезков (ДО) над заданным массивом в котором будем хранить минимальный индекс 0 и максимальный индекс 1 на отрезке.
- Для ответа на запрос $\boldsymbol{seq}$ выполним запрос в ДО, и если индекс 0 на отрезке меньше индекса 1, то можно собрать НВП длины 2 – иначе 1.
- Для выполнения запроса $\boldsymbol{xor}$ выполним групповую модификацию на заданном отрезке.
- Сложность $O(\boldsymbol{nq * log(n)})$.

Задача 4. Мощный процессор (100 балла)

- Построим ДО на заданном массиве, в котором будем хранить минимальный индекс каждого числа на отрезке.
- Для ответа на запрос $\boldsymbol{seq}$ вычислим значения $\boldsymbol{dp_{i,k}}$ – минимальная длина префикса отрезка, на котором можно построить НВП длины $\boldsymbol{i}$ и которая заканчивается на число $\boldsymbol{k}$.
- Для просчета динамического программирования будем использовать формулу: $\boldsymbol{dp_{i+1,h}} = \max\left(\boldsymbol{dp_{i+1,h}}, \boldsymbol{idx_{h,dp_{i,k}+1}}\right)$, где $\boldsymbol{idx_{h,t}}$ – индекс первого вхождения $\boldsymbol{h}$ в отрезок от $\boldsymbol{t}$ до $\boldsymbol{r}$.

Задача 4. Мощный процессор (100 балла)

- Массив $\boldsymbol{idx_{h,t}}$ можно получить с помощью запроса в ДО на отрезке от $\boldsymbol{t}$ до $\boldsymbol{r}$.
- Запрос $\boldsymbol{xor}$ обрабатывается с помощью групповой модификации ДО на отрезке $\boldsymbol{l_j}$ до $\boldsymbol{r_j}$.
- Сложность $O(\boldsymbol{cq} * \boldsymbol{log}(\boldsymbol{n}))$, где $\boldsymbol{c = 64}$.

День 2

Задача 1. Футбол

Задача 1. Футбол

- Дан набор команд участвующих в соревновании.
- Последовательность матчей - названий двух команд участвующих в соревновании и количества голов забитых командами соответственно.
- Если чемпионат еще не закончился требуется вывести количество игр до завершения.
- Если чемпионат закончился, то вывести победителя.

Задача 1. Футбол

- Первым делом проверим завершились ли все матчи: количество матчей среди n команд можно найти по формуле $n * (n - 1)$.
- Если это значение не совпадает с m выводим “NO” и количество оставшихся матчей по формуле $n * (n - 1) - m$.
- В случае когда все матчи закончились: читаем названия команд и выполняем их сортировку в лексикографическом порядке.
- Далее читаем m строк - информацию о матчах.
- Если победила первая команда, то выполняем поиск имени первой команды в массиве имен, и прибавляем 3 в массив результата по индексу – позиции имени в массиве имен.

$$results[find(name)\ in\ names]\ +=\ 3$$

- Аналогично для случая победы второй команды.

Задача 1. Футбол

- После обработки всех матчей выполняем поиск максимума в массиве результатов.
- И выводим имя победившей команды.
- Данную задачу можно решить с использованием ассоциативного контейнера типа map храня пару (название команды, результат).

Задача 2. Странные антинейтрино

Задача 2. Странные антинейтрино

- Дан массив $\boldsymbol{a}$ состоящий из $\boldsymbol{n}$ элементов.
- Есть некоторая функция:

$$S(a) = \max(a_1 + a_2, a_2 + a_3, \dots, a_{n-1} + a_n)$$

- Требуется минимизировать значение этой функции для массива $\boldsymbol{a}$ путем перестановки элементов.
- В ответ вывести минимальное значение функции и последовательность, дающая это значение.

Задача 2. Странные антинейтрино

- Заметим, что на значение функции влияют соседние два числа из массива $\boldsymbol{a}$.
- Значит, нам нужно постараться переставить элементы в массиве, так чтобы минимизировать $\boldsymbol{a_i + a_{i+1}}$.
- Для этого удобно располагать последовательно максимальное и минимальное число из оставшегося набора чисел. Поэтому отсортируем массив по возрастанию.
- Тогда итоговая последовательность будет выглядеть так:

$$a_n\ a_1\ a_{n-1} a_2\ a_{n-2}\ a_3\ a_{n-3}\ a_4\ \ldots$$

Задача 2. Странные антинейтрино

- Если использовать обычную сортировку, то можно получить 52 балла.
- Сложность $O(n^2)$.

- Используя любую из быстрых сортировок над массивом $\boldsymbol{a}$ мы получим решение на 100 баллов.
- Сложность $O(n * \log n)$.

Задача 3. Быстрые вычисления

Задача 3. Быстрые вычисления

- Дан отрезок $[\boldsymbol{l}, \boldsymbol{r}]$ и число $\mathbf{s}$.
- Требуется найти количество таких подотрезков $[\boldsymbol{i}, \boldsymbol{j}] \in [\boldsymbol{l}, \boldsymbol{r}]$, для которых соблюдается условие:

$$\mathrm{F}(i, j) = \mathrm{i} \otimes (\mathrm{i} + 1) \otimes \cdots \otimes j = \bigotimes_{k=i}^{j} k = s$$

- где $\otimes$ – означает операцию исключающего или (xor).

Задача 3. Быстрые вычисления (21 балл)

- Переберем концы подотрезка $[\boldsymbol{i}, \boldsymbol{j}]$
- В процессе перебора будем поддерживать $\boldsymbol{xor}$ чисел отрезка. Каждый раз, когда $\boldsymbol{xor}$ равен $\boldsymbol{s}$ будем обновлять ответ.
- Сложность $O(\boldsymbol{n^2})$, где $\boldsymbol{n} = \boldsymbol{r} - \boldsymbol{l}$.

Задача 3. Быстрые вычисления (34 балла)

- Заметим, что $\boldsymbol{xor}$ отрезка $[\boldsymbol{i}, \boldsymbol{j}]$ равен:

$$\mathrm{i} \otimes (\mathrm{i}+1) \otimes \cdots \otimes j = \bigotimes_{k=i}^{j} k =$$

$$= \left(\bigotimes_{k=0}^{j} k \right) \otimes \left(\bigotimes_{k=0}^{i-1} k \right) = F(0, j) \otimes F(0, i-1)$$

- Тогда используя это свойство можно предпросчитать префикс-суммы $\boldsymbol{f_i = F(0, i)}$.

Задача 3. Быстрые вычисления (34 балла)

- Переберем правую границу подотрезка $\boldsymbol{j}$.
- Чтобы найти количество левых границ $\boldsymbol{i}$, таких что отрезок $[\boldsymbol{i}, \boldsymbol{j}]$ удовлетворяет требуемому условию. Нужно для $\mathbf{f_j}$ посчитать количество таких $\mathbf{f}_{\boldsymbol{i}} = \boldsymbol{f}_{\boldsymbol{j}} \otimes \boldsymbol{s}$.
- Для этого будем поддерживать статистику количества вхождений $\boldsymbol{f}_{\boldsymbol{i}}$ с помощью любого ассоциативного контейнера.
- После чего на каждой итерации будем прибавлять к ответу количество $\mathbf{f}_{\boldsymbol{i}} = \boldsymbol{f}_{\boldsymbol{j}} \otimes \boldsymbol{s}$.
- Сложность $O(\boldsymbol{n} * \boldsymbol{log}(\boldsymbol{n}))$, где $\boldsymbol{n} = \boldsymbol{r} - \boldsymbol{l}$.

Задача 3. Быстрые вычисления (50 балла)

- Ограничения – $\boldsymbol{s = 0}$.
- Заметим, что если $\boldsymbol{t = 4k}$, то $\boldsymbol{t \otimes (t+1) \otimes (t+2) \otimes (t+3) = 0}$. Значит $\boldsymbol{f_i}$ равно:
  - $\boldsymbol{i}$ для $\boldsymbol{i = 4k}$
  - $\boldsymbol{1}$ для $\boldsymbol{i = 4k+1}$
  - $\boldsymbol{i+1}$ для $\boldsymbol{i = 4k+2}$
  - $\boldsymbol{0}$ для $\boldsymbol{i = 4k+3}$
- Так как $\boldsymbol{F(i,j) = 0}$ только, при $\boldsymbol{f_j = f_{i-1}}$, а учитывая предыдущее свойство это может быть только если $\boldsymbol{f_j = f_{i-1}}$ равно $\boldsymbol{0}$ или $\boldsymbol{1}$.
- Поэтому можно посчитать количества $\boldsymbol{0}$ и $\boldsymbol{1}$ на отрезке $\boldsymbol{\mathrm{cnt}_0}$ и $\boldsymbol{\mathrm{cnt}_1}$. Послу чего ответ будет равен $\boldsymbol{\frac{\mathrm{cnt}_0(\mathrm{cnt}_0+1)}{2} + \frac{\mathrm{cnt}_1(\mathrm{cnt}_1+1)}{2}}$.

Задача 3. Быстрые вычисления (100 баллов)

- Посчитаем ДП $\boldsymbol{dp}$, со следующими измерениями:
  - $\boldsymbol{bit\_index}$ – индекс префикса который уже собран;
  - $\boldsymbol{i\_suffix}$ – суффикс (последние 2 бита) левой границы $\boldsymbol{i-1}$;
  - $\boldsymbol{j\_suffix}$ – суффикс (последние 2 бита) левой границы $\boldsymbol{j}$;
  - $\boldsymbol{i\_eq\_j}$ – равен ли собранный префикс $\boldsymbol{i-1}$ префиксу $\boldsymbol{j}$;
  - $\boldsymbol{i\_eq\_l}$ – равен ли собранный префикс $\boldsymbol{i-1}$ префиксу $\boldsymbol{l}$;
  - j$\boldsymbol{_eq_r}$ – равен ли собранный префикс $\boldsymbol{j}$ префиксу $\boldsymbol{r}$;
- Для упрощения просчета состояний можно использовать DFS с мемоизацией, запоминая уже просчитанные состояния в массиве.

Задача 3. Быстрые вычисления (100 баллов)

- Для просчета текущего состояния ДП нужно перебрать бит $\boldsymbol{i}$ и бит $\boldsymbol{i}$ чтобы увечить проверяемый префикс.
- После чего нужно проверить возможен переход, а именно:
  - Идут ли $\boldsymbol{l}, \boldsymbol{i}, \boldsymbol{j}, \boldsymbol{r}$ в возрастающем порядке
  - Равен ли $\boldsymbol{bit\_index}$–й бит $\mathbf{F}(\boldsymbol{i}, \boldsymbol{j})$ $\boldsymbol{bit\_index}$–ому биту $\boldsymbol{s}$.
- Значением текущего состояния ДП является сумма значений по переходам ДП.
- После можно вычислить ответ как сумма ДП по всем возможным суффиксам $\boldsymbol{i\_suffix}$ и $\boldsymbol{j\_suffix}$.
- Сложность $\boldsymbol{O(C * log(n))}$, где $\boldsymbol{n = r - l}$, $\boldsymbol{C}$ – константа, не зависящая от $\boldsymbol{n}$.