Открытая олимпиада

Разбор задач

Перестроение

Автор задачи: Михаил Густокашин, МГУ

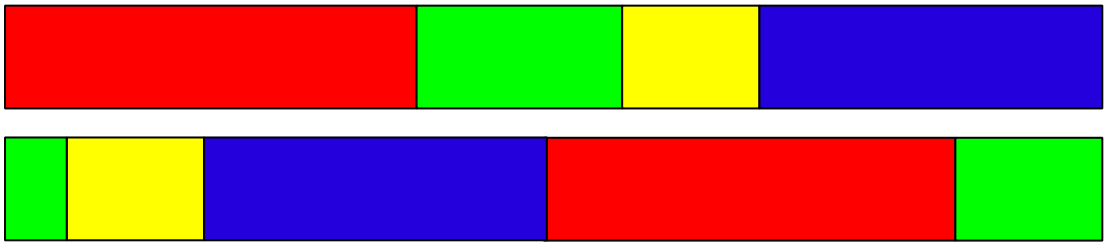
Разработчик: Антон Полднев, МГУ

Частичные решения

- Только буквы a, b, c (30 баллов).
- Перебор (30 баллов → 100 баллов).
- Паросочетания (60 баллов).

Простое полное решение

Предположим, что буквы в строке отсортированы. Тогда можно просто циклически сдвинуть строку на $\left\lfloor \frac{N}{2} \right\rfloor$.



Соответственно, получим решение для произвольной строки: отсортировать буквы, запомнив их первоначальное положение, сделать циклический сдвиг на $\left\lfloor \frac{N}{2} \right\rfloor$, затем восстановить испорченный сортировкой порядок символов.

Берляндолимпстрой

Автор задачи: Михаил Пядёркин, МГУ

Разработчик: Александр Тимин, МФТИ

Постановка задачи

- Есть массив чисел. Он K раз заменяется на массив своих префиксных сумм
- В некоторый ненулевой момент мы можем добавить к какому-нибудь элементу число X
- Необходимо сделать так, чтобы в результате итоговая сумма увеличилась ровно на R

Решение на 30 баллов

- Если мы знаем, в какой момент и где мы добавили величину X, то мы знаем, на сколько изменилась итоговая сумма
- Переберем элемент таблицы и значение X, после этого промоделируем процесс и получим 30 баллов

…	…	…	…	…	…	…	…
…	$+X$						…
…	$+X$	$+X$	$+X$	$+X$	$+X$	$+X$	…
…	$+X$	$+2X$	$+3X$	$+4X$	$+5X$	$+6X$	…
…	$+X$	$+3X$	$+6X$	$+10X$	$+15X$	$+21X$	…

Решение на 60 баллов

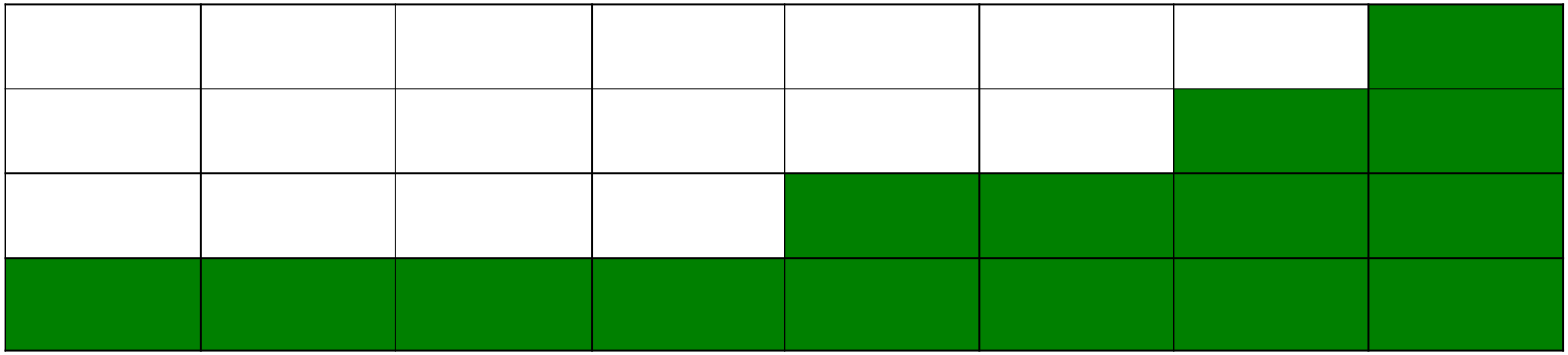
- От исходных чисел в массиве ничего не зависит, а величина R прямо пропорциональна X
- Переберем момент, в который мы изменим число. Пусть при добавлении единицы итоговая сумма изменяется на S. Тогда, если R делится на S, то R/S – кандидат на ответ.
- Среди всех таких кандидатов выбрать минимальный.

Решение на 60 баллов

- Как быстро посчитать изменение итоговой суммы при добавлении единицы?
- В следующий момент времени у всех объектов с не меньшим номером стоимость увеличится на 1
- Тогда если T_{ij} – изменение итоговой суммы, вызванное добавлением единицы в j-тый элемент массива в момент времени i, то $T_{i,j} = \sum_{k \geq j} T_{i+1,k}$
- Посчитать это можно за размер таблицы – $O(NK)$, аналогично префиксным суммам

Решение на 100 баллов

- Заметим, что $T_{i,j}$ быстро (экспоненциально) растут, а для того, чтобы получить кандидата на ответ, то надо, чтобы $T_{i,j}$ было не больше изменения суммы D
- Возможных позиций (i, j), таких, что $T_{i,j} \leq D$, мало.
- Так как $T_{i,j} > T_{i+1,j}$, $T_{i,j} > T_{i,j+1}$, то картинка выглядит так



- Это позволяет нам быстро перебирать: идем снизу-вверх, справа-налево, когда число в строке слишком большое, то заканчиваем обработку строки
- $D_{i,j}$ на самом деле равно $C_{n-j+k-i}^{k-i} = \frac{(n-j+k-i)!}{(k-i)!\,(n-j)!}$

Транспортные потоки

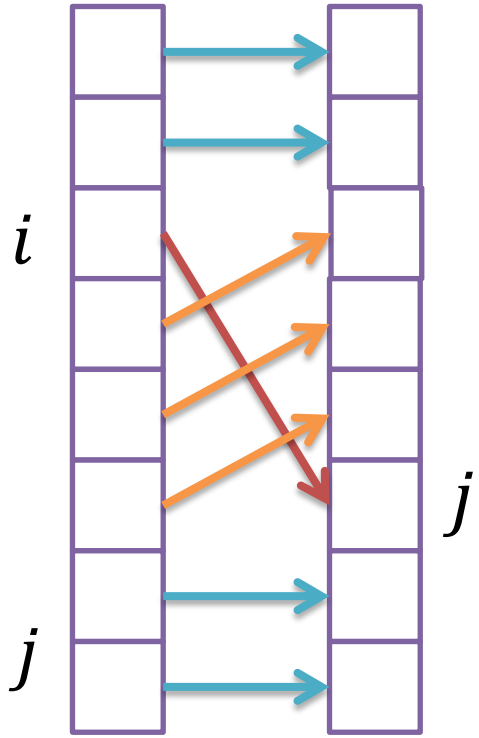
Автор задачи: Ивлев Фёдор, МГУ
Разработчики: Ахмедов Максим, МГУ
Пядёркин Михаил, МГУ

Решение на 20 баллов

- Разберёмся, как может происходить изменение порядка следования автобусов
- Предположим, автобус, отправлявшийся i-ым от первой остановки, приехал на вторую остановку позже, чем надо, например, j-ым, где $i < j$.

Решение на 20 баллов

- Оставшиеся автобусы обязаны соблюсти расписание.
- Первый из оставшихся автобусов приезжает как можно раньше, и ждёт первого времени, указанного на второй остановке.
- Следующий прибывает тоже как можно раньше, и ждёт второго времени, указанного на остановке и так далее.
- Необходимо, чтобы для всех t от $i + 1$ до j было верно, что $A[t] + time \leq B[t - 1]$, условия для остальных автобусов выполняются гарантированно.



Решение на 20 баллов.

- Тем самым, все автобусы от $i + 1$ до j должны успевать приехать на одну позицию раньше своей.
- Найдем для каждой позиции из расписания i максимальное подходящее j – это будет ответ для i-ого автобуса, отходящего от первой остановки, который мы обозначим $F[i]$
- Общая сложность решения – $O(M^2 + Q)$.

Решение на 40 баллов

- Заметим, что массив F можно строить за линейное время.
- Назовём автобус i «тормозящим», если он не успевает приехать на одну позицию раньше по расписанию, т. е. $A[i] + time > B[i-1]$.
- Заметим, что $F[i]$ равняется такому минимальному $j > i$, что автобус $j+1$ – «тормозящий», либо M, если такого нет.
- Находим все «тормозящие автобусы», за один проход строим значения $F[i]$.
- Общая сложность решения – $O(M+Q)$.

Решение на 60 баллов

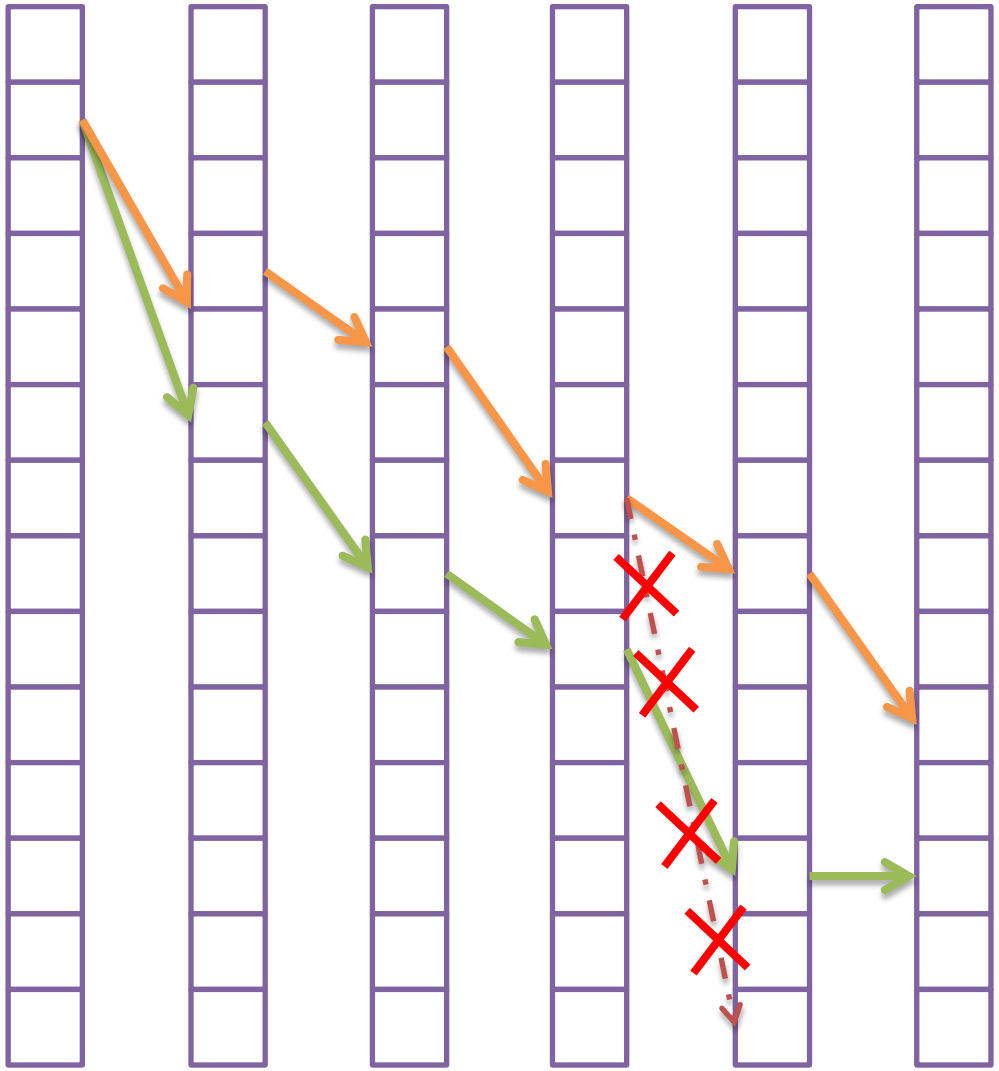
- Назовём для двух соседних остановок массив F, построенный в предыдущих решениях, *отображением.*
- Заметим, что выполняется $F[i] \geq i$.
- Также для $i < j$ верно, что $F[i] \leq F[j]$.
- Обозначим за $F_k[\cdot]$ отображение между k и $k + 1$ остановкой.

Решение на 60 баллов

- Для того, чтобы приехать как можно позже i-ому автобусу с l-ой остановки на r-ую, ему нужно между каждой парой соседних остановок опаздывать по максимуму.
- Иными словами, его позиция x_i на i-ой остановке определяется как $F_{i-1}[x_{i-1}]$.
- Иными словами, на r-ю остановку он может приехать самое позднее $F_{r-1}[F_{r-2}[\dots[F_l[i]]\dots]]$-ым.
- Картинка на следующем слайде.

Решение на 60 баллов

- Верхний путь между соседними остановками каждый раз переходит жадно, по максимальному опозданию.
- В силу того, что для пары остановок отображение F – возрастающее любой нижний путь ни на каком отрезке не может стать выше верхнего.



Решение на 60 баллов

- Построим все отображения за время $O(NM)$, как описано в решении на 40 баллов.
- Ответ на один запрос можно найти за $O(N)$, проходя от l-ой остановки до r-ой, и считая текущую позицию по правилу $x_i = F_{i-1}[x_{i-1}]$.
- Общая сложность решения – $O(NM + NQ)$.

Первое решение на 80 баллов

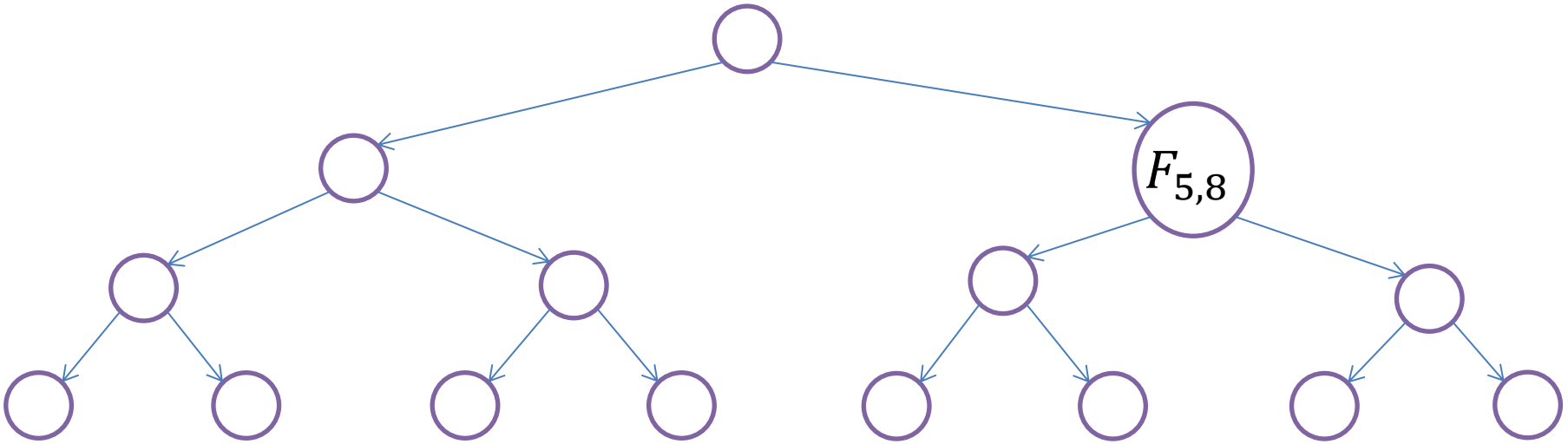
- Задача свелась к тому, чтобы научиться быстро считать значение $F_{l,r}[i]$, где $F_{l,r}$ — это композиция отображений на полуинтервале от l до r.
- Воспользуемся методом двоичных подъёмов. Посчитаем вспомогательные отображения $G_{l,k} = F_{l,l+2^k-1}$ – то есть, для каждой остановки посчитаем отображение из неё в остановку на расстоянии 2^k.
- Это можно сделать по формуле $G_{l,k}[i] = G_{l+2^k-1,k-1}\left[G_{l,k-1}[i]\right]$.

Первое решение на 80 баллов

- Теперь стало возможным отвечать на запрос $F_{l,r}[i]$ за время $\log N$ методом двоичных подъёмов: будем перебирать длину очередного прыжка 2^k сверху вниз. Если на очередном шаге имеется возможность сделать прыжок, и не выйти при этом за r, делаем его.
- Время построения вспомогательных отображений $G - O(NM \log N)$.
- Ответ на один запрос за время $O(\log N)$.
- Общая сложность – $O(NM \log N + Q \log N)$.

Второе решение на 80 баллов

- Воспользуемся идеей дерева отрезков отображений.
- Построим на остановках дерево отрезков.
- Для вершины i, отвечающей за полуинтервал $[l, r)$ построим отображение $G_i = F_{l,r}$, т. е. отображение из левой границы её поддерева в правую границу её поддерева.

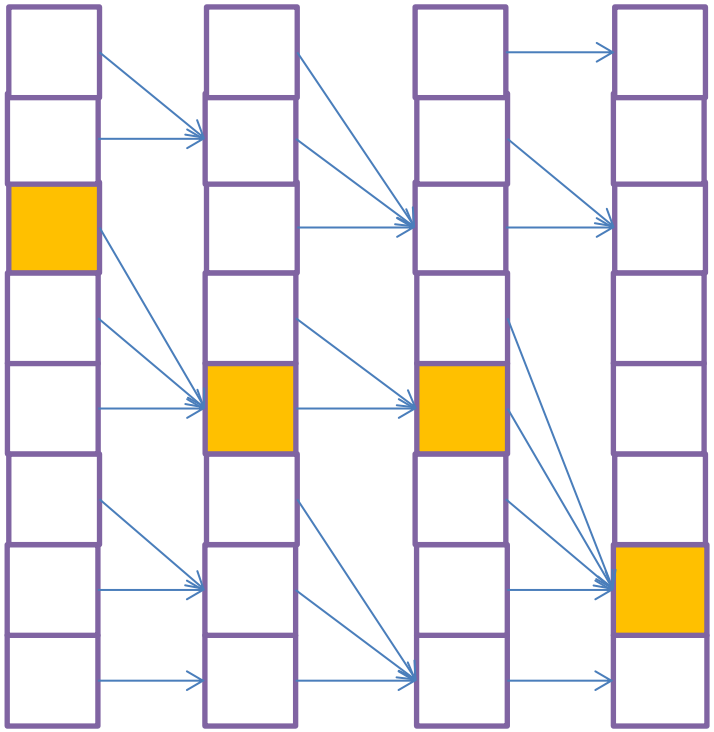


Второе решение на 80 баллов

- Дерево отрезков позволяет разбить отрезок $[l, r)$ на не более $2 \log N$ отрезков, на каждом из которых мы знаем отображение. Пройдём по ним и возьмём результат композиции этих отображений.
- Построение дерева отрезков происходит за $O(NM)$.
- Ответ на один вопрос за $O(\log N)$.
- Общая сложность решения – $O(NM + Q \log N)$.

Полное решение

- Ответим на все запросы offline, т. е. разом.
- Рассмотрим граф, где вершинами являются записи в расписании, т. е. пары (k, i), где k – остановка, а i – номер автобуса. Проведём ребро из вершины (k, i) в вершину $(k + 1, F_k[i])$.
- Заметим, что полученный граф выглядит как лес, где рёбра направлены от потомка к предку.



Полное решение

- Запрос можно переформулировать как «найти для вершины в лесу её $(r - l)$-ого предка».
- Структура запросов в пятой группе такова, что мы сразу знаем, какие запросы будут связаны с каждой вершиной.
- Обойдём граф в глубину, поддерживая стек вершин. Тогда предок глубины k для текущей вершины – это k-ая сверху вершина в стеке.
- Общая сложность решения – $O(NM)$.

Силовое поле

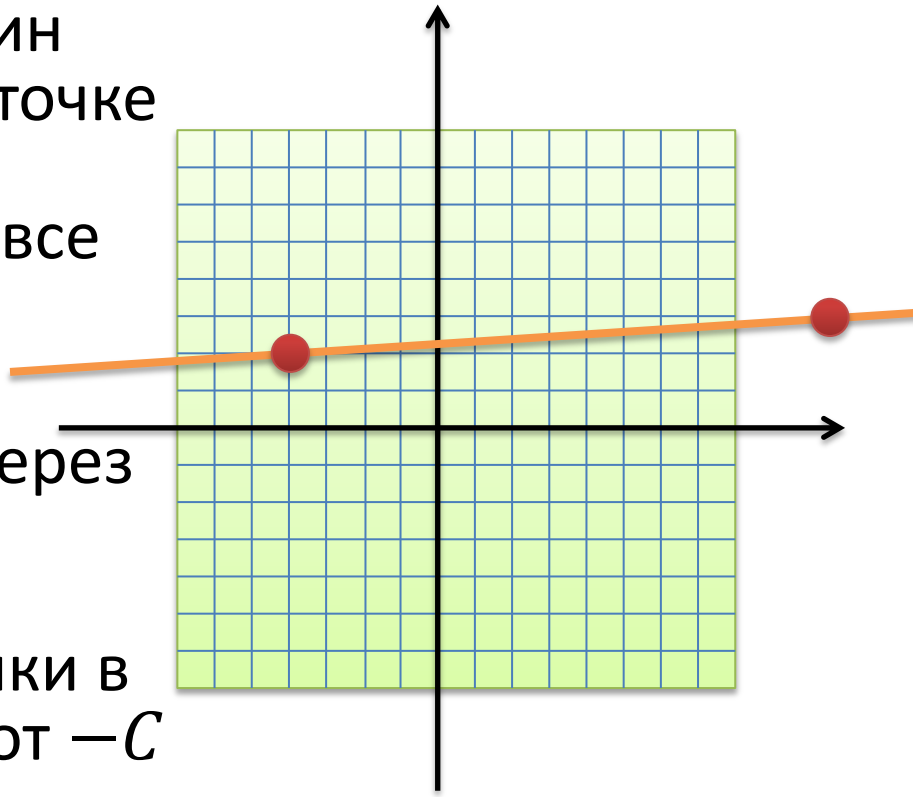
Автор: Шестимеров Андрей, МГУ
Разработчик: Ахмедов Максим, МГУ

Решение на 30 баллов

- Треугольник определяется тремя прямыми, на которых лежат его стороны.
- Если на прямой лежат две точки, то на ней лежит сторона треугольника. Найдём эти прямые.
- Мы можем посчитать количество точек, лежащих на прямой, как $N - l - r$, где l и r – количество точек слева и справа от прямой.
- Переберём все возможные пары точек, в пределах от -10 до 10. Их $\frac{21^2 \cdot (21^2 - 1)}{2} = 97020$, сделаем запрос про каждую прямую и найдем три нужные

Решение на 60 баллов

- Научимся проверять за один запрос, расположена ли в точке станция-ретранслятор. С помощью этого проверим все допустимые точки.
- Проверить, есть ли в точке станция можно, проведя через точку *“косую”* прямую, на которой не лежат никакие другие целочисленные точки в квадрате с координатами от $-C$ до C.



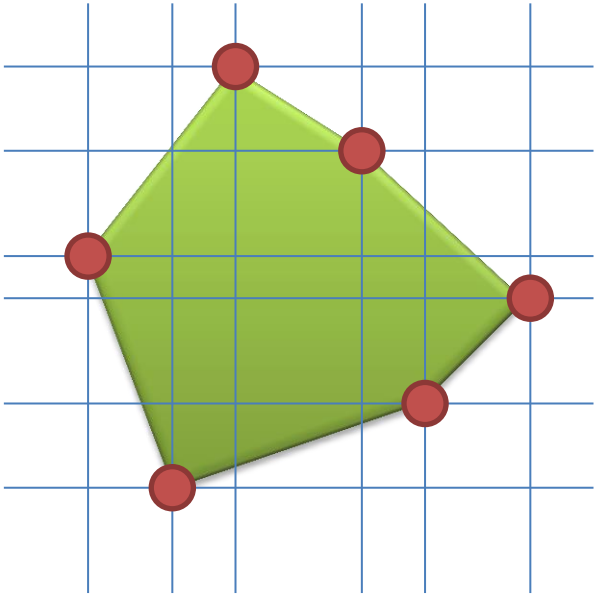
- Например, такой прямой для точки (x, y) является прямая через точки (x, y) и $(x + 3C, y + 1)$.

Важное соображение

- Можно найти x-координаты всех точек.
- Обозначим за $\mathrm{right}(c)$ количество точек справа от вертикальной прямой $x = c$.
- Упорядочим точки по x-координате. Обозначим x-координату i-ой (в нумерации с единицы) точки за x_i.
- Очевидно, функция right – неубывающая.
- Мы можем найти x_i с помощью бинарного поиска по функции $\mathrm{right}(c)$. x_i равняется минимальному c, удовлетворяющему условию $\mathrm{right}(c) \leq n - i$.
- Тем самым, мы можем найти x-координаты всех точек за $N \log C \approx 20\,000$ запросов.

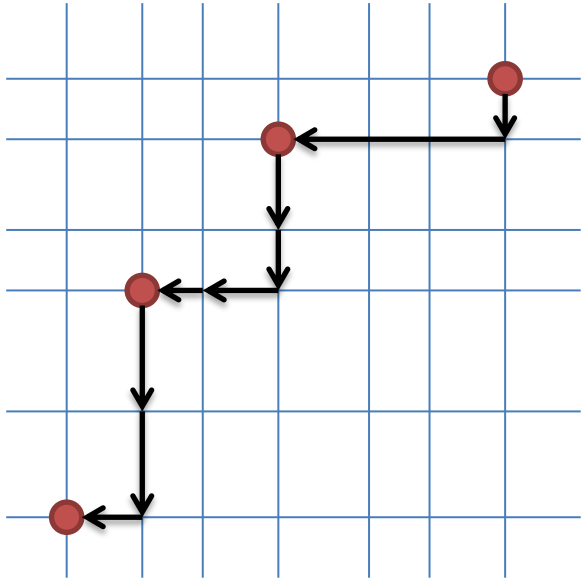
Полное решение №1

- Найдём x-координаты и y-координаты всех точек. Тем самым по определим N^2 точек, среди которых находятся станции-ретрансляторы. Они образуют «сетку» на плоскости.
- Станции-ретрансляторы на этой сетке образуют выпуклый многоугольник. Можно попытаться его «обойти» за $O(N)$, найдя при этом все вершины.



Полное решение №1

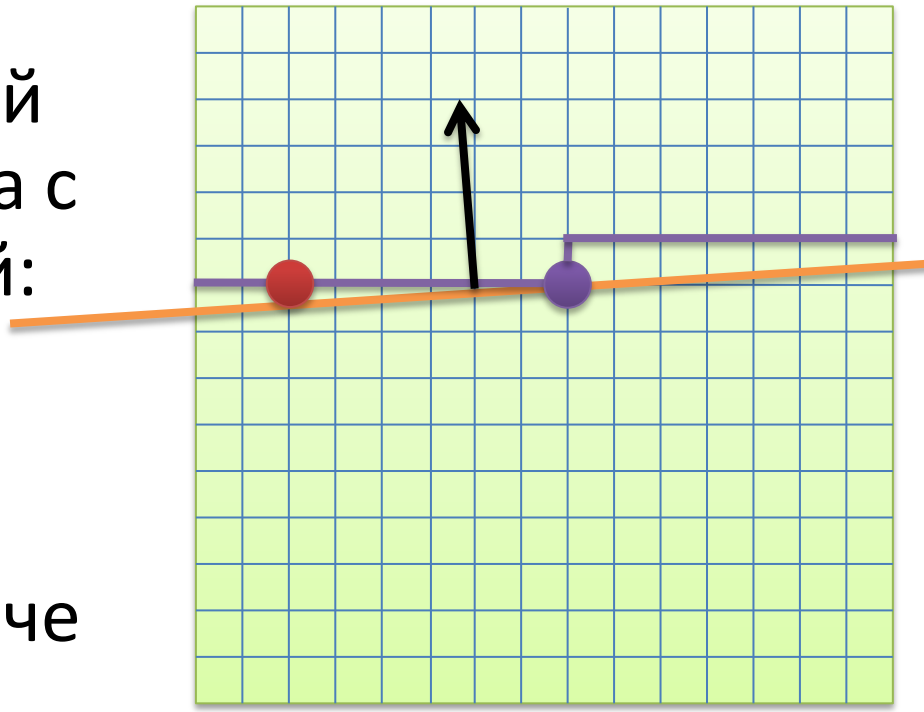
- Будем обходить точки многоугольника против часовой стрелки.
- Найдём с помощью *косой* прямой самую верхнюю, самую нижнюю, самую левую и самую правую вершину многоугольника. Они разбивают его на четыре дуги.



- Стартуем из самой верхней вершины. Вплоть до самой левой вершины многоугольника x-координата каждой очередной вершины будет всё меньше и меньше

Полное решение №1

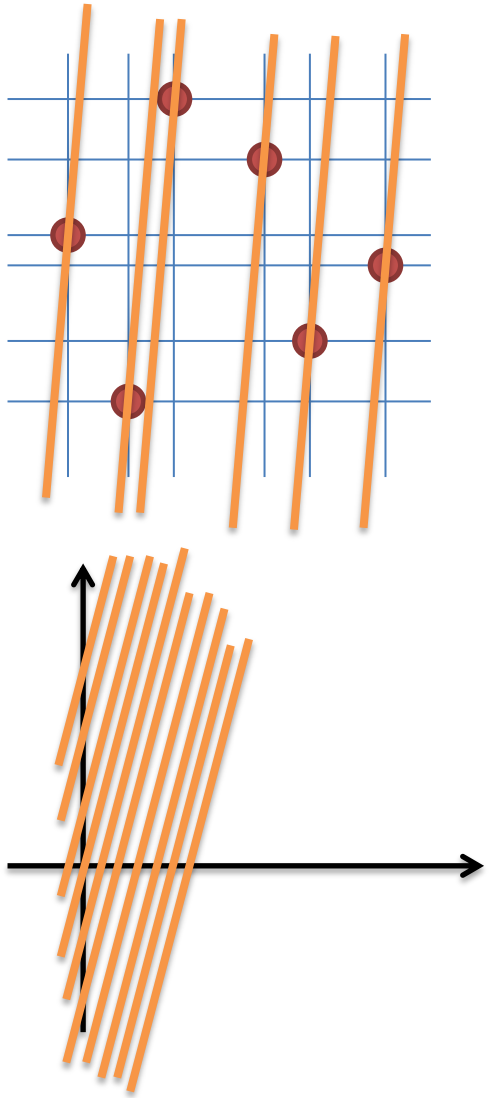
- Понимаем, есть ли на текущей горизонтальной прямой ещё точки слева с помощью *косой* прямой: пока над *косой* прямой больше точек, чем над текущей горизонталью, сдвигаемся вправо, иначе – вниз.



- Аналогично обойдём оставшиеся три дуги.
- Итоговое количество запросов – $2N \log C + k \cdot N$ (в решении жюри $k = 6$), что не превосходит $50\,000$ запросов.

Полное решение №2

- Мы умеем находить x-координаты точек, или, что то же самое, проводить через каждую точку вертикальную прямую.
- Заметим, что то же самое можно сделать и с *косыми* прямыми, параллельными вектору $(1, 3C)$
- По направлению перпендикулярно вертикальной прямой точки идут в таком же порядке, как и по направлению перпендикулярно *косой* прямой.
- Значит можно найти i-ую точку многоугольника как пересечение i-ой найденной вертикальной прямой с i-ой найденной *косой* прямой.



Полное решение №2

- Для бинарного поиска необходимо уметь брать $right(t)$ – количество точек справа от t-ой по порядку прямой.
- Заметим, что раз уравнение такой прямой имеет вид $y = 3Cx + t$, прямая однозначно задаётся значением t, то есть точкой пересечения с координатной осью OY
- Бинпоиск по *косым* прямым фактически будет бинпоиском по свободному коэффициенту t уравнения косой прямой. Зафиксировав очередную прямую надо не забыть выбрать на ней две точки, с координатами от $-5C$ до $5C$, для этого нужно положить $x_1 = \left[\frac{t}{3C}\right], x_2 = x_1 + 1$. Тогда $|y_1|, |y_2| \leq 3C$.

Полное решение №2

- Оценим количество запросов.
- Количество косых прямых, пересекающих квадрат допустимых точек – $O(C^2)$, так как каждая прямая задаётся своим свободным коэффициентом t в уравнении $y = 3Cx + t$, а крайние положения прямой – при $x = \pm C$, где t принимает значение порядка $\pm 3C^2$.
- Тем самым общая сложность решения – $N \log C + N \log(6C^2) \approx 3N \log C$.

Небольшое улучшение решения №2

- Можно заметить, что на i-ом шаге нас интересуют только *косые* прямые, пересекающие i-ую вертикальную прямую, а их – $O(C)$ штук. Бинпоиск по *косым* прямым можно пускать только среди них.
- Получилось решение за $N \log C + N \log C = 2N \log C$, которое при данных ограничениях на N и C делает $40\,000$ запросов.

Соревнование по распилу

Автор задачи: Глеб Евстропов, МГУ

Разработчик: Александр Тимин, МФТИ

Постановка задачи

- Пусть N_i обозначает общее число участников в i-ом туре, N_1 задано
- Нужно выбрать такие $N_1 > N_2 > \cdots > N_p$, что p – максимально и для каждой пары соседних чисел N_i, N_{i+1} можно выбрать такие $A,\ B,\ C$, что $N_i = AB$, $N_{i+1} = AC$, $A, B \geq 2, C < B$

Некоторое наблюдение

- $N_i = AB, N_{i+1} = AC$
  $N_i - N_{i+1} = A(B - C) \geq A$
- A – делитель N_i
- То есть количество участников уменьшается как минимум на минимальный делитель N_i
- Мы хотим, чтобы было проведено как можно больше туров
- То есть нам надо постараться, чтобы участники выбывали как можно медленней

Алгоритм

- Если $N = 1$, то мы можем провести один тур
- Пусть у нас есть $N > 1$ участников и $D > 1$ это минимальный нетривиальный делитель N
- Тогда проведем тур следующим образом: D подгрупп по N/D человек, в каждой подгруппе в следующий тур проходят N/D – 1
- Выбыло D, меньше человек в первом туре выбыть не может

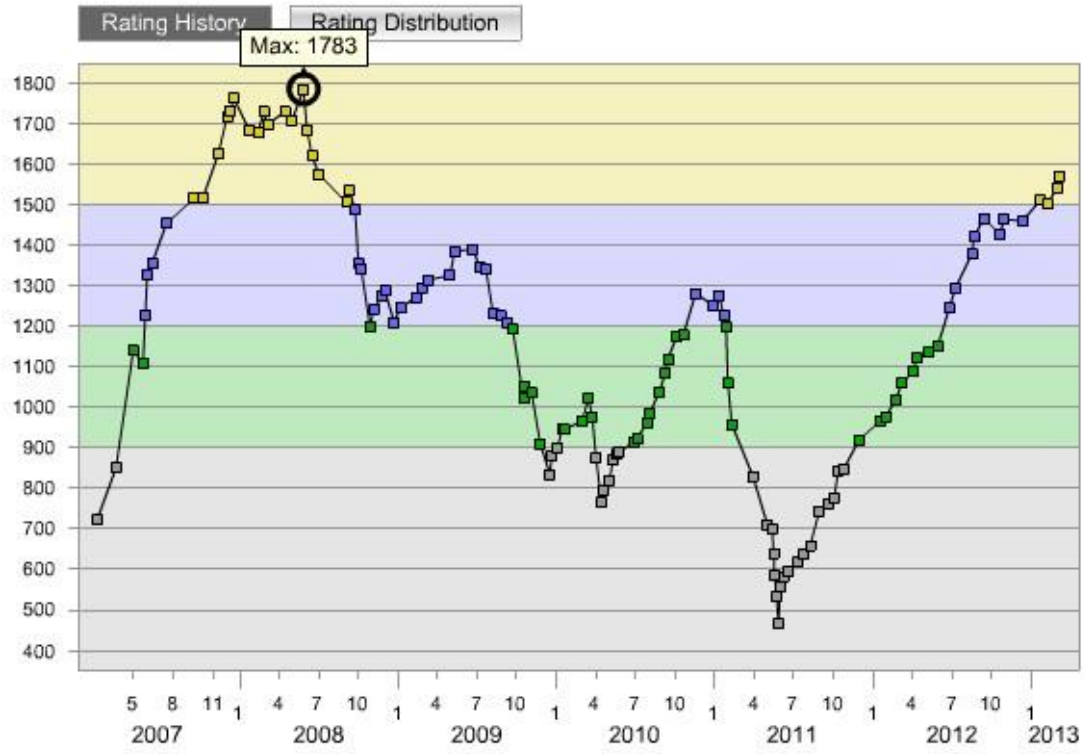
Доказательство корректности в случае, если N – четное

- Пусть N четное. Тогда его минимальный делитель – 2.
- То есть в следующий тур проходят $N - 2$ человека – четное количество.
- Общим числом туров (ответом задачи) в этом случае будет являться число $N / 2$. Больше быть не может, так как в каждом туре выбывает как минимум два человека.

Доказательство корректности в случае, если N – нечетное

- Пусть N нечетное. Тогда его минимальный делитель D – нечетное число.
- Тогда в следующий тур пройдет $N - D$ человек – четное количество
- Дальше количество участников всегда четное
- Ответ: $(N - D) / 2 + 1$
- На 100 баллов поиск минимального делителя требуется производить за корень из N

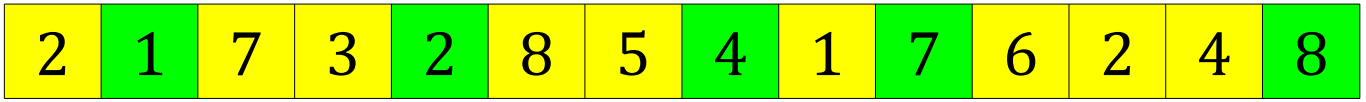
Несостоявшийся программист



Автор задачи: Александр Тимин, МФТИ

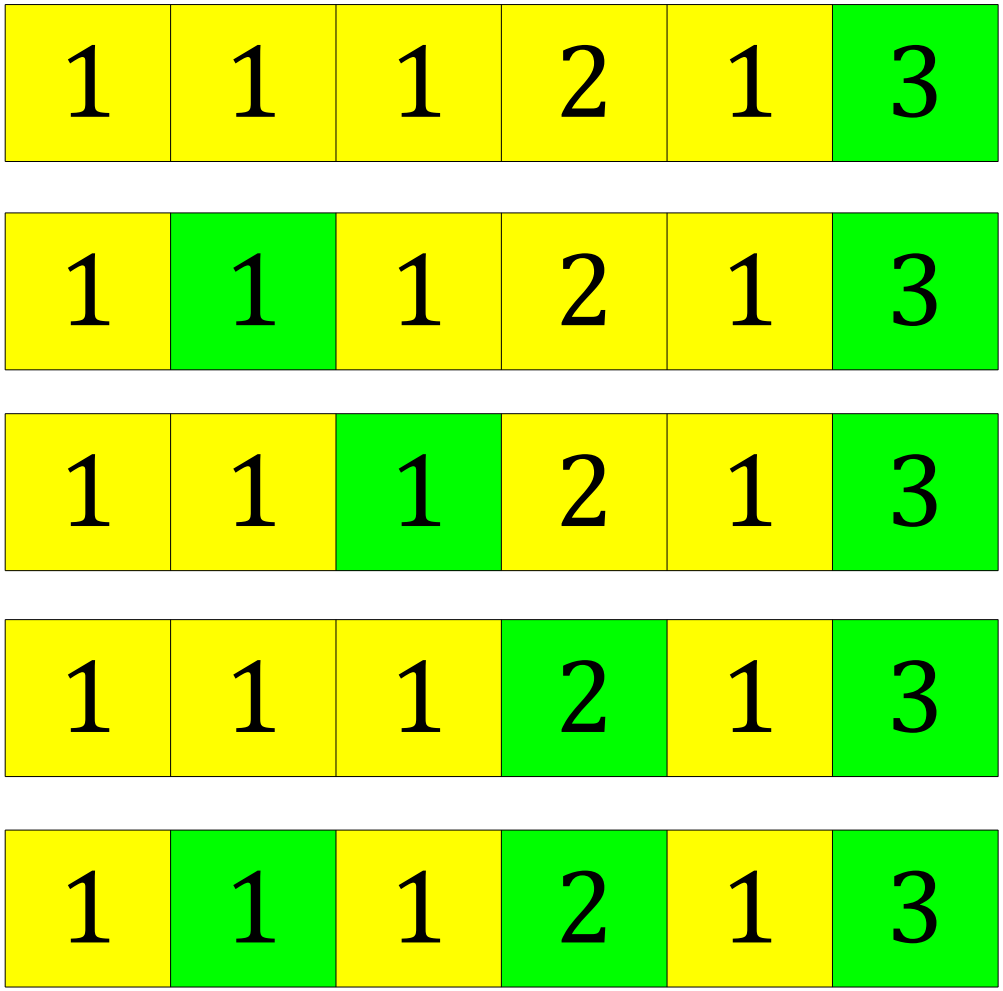
Разработчик: Роман Андреев, СПбГУ

Каждому набору $(v_1, \ldots, v_8)$ соответствует подпоследовательность цифр числа N, обладающая определёнными свойствами. Назовём такую подпоследовательность *хорошей.*



1. Состоит только из цифр от 1 до 8.
2. Возрастает.
3. Заканчивается в последнем символе.
4. Начинается не в первом символе.
5. Между любыми соседними элементами есть хотя бы одна цифра.
6. После каждого элемента стоит не 0.
7. Число N не начинается с нуля.

Пример из условия



$L \leq 1\,000$ — 60 баллов

Если s_1 равно нулю, то ответ – ноль. Иначе,
пусть cnt_i $(i = 1 \ldots L)$ — количество хороших последовательностей, заканчивающихся в i-м элементе.
$cnt_1 = 0$, $cnt_2 = 1$, если s_2 от 1 до 8, и 0 иначе.
Найдём остальные cnt_i, зная предыдущие.

```
for i = 1..L:
  x = s[i]
  if x ≠ 0 and x ≠ 9 and (i = L or s[i + 1] ≠ 0):
    cnt[i] = (i > 0 and s[1] ≠ 0) ? 1 : 0
    for j = 1..i-2:
      if s[j] < x:
        cnt[i] = (cnt[i] + cnt[j]) mod M
```

$L \leq 1\,000\,000$ — 100 баллов

Пусть $before_{i,j}$ — общее количество хороших последовательностей, заканчивающихся на позиции i или раньше, причём на цифру j или меньшую и таких, после которых идёт не 0.

Тогда при $i > 2$ и s_i от 1 до 8 имеем:

$$cnt_i = before_{i-2,s_i-1}$$

И, возможно, +1, если $i > 0$ и $s_1 \neq 0$.

В конце каждого шага насчитываем для всех j от 1 до 8

$$before_{i,j} = \begin{cases} before_{i-1,j} + cnt_i, \text{если } j \geq s_i \text{ и } s_{i+1} \neq 0 \\ before_{i-1,j} \qquad \text{иначе} \end{cases}$$

Две башни

Автор задачи: Андрей Шестимеров, МГУ

Разработчик: Антон Полднев, МГУ

Насколько большим может быть ответ?

Каждая башня отнимает у каждого монстра хотя бы одну жизнь. Следовательно, все монстры с 1 и 2 жизнями умрут, зато каждый монстр с хотя бы 3 жизнями продвинется дальше предыдущего. Значит, оценка сверху на номер первого прорвавшегося монстра — $K + K + D$. Тогда оценка на ответ — $2K + 2D = O(K + D)$.

Например, на тесте

```
1000 0 500 501 1000
```

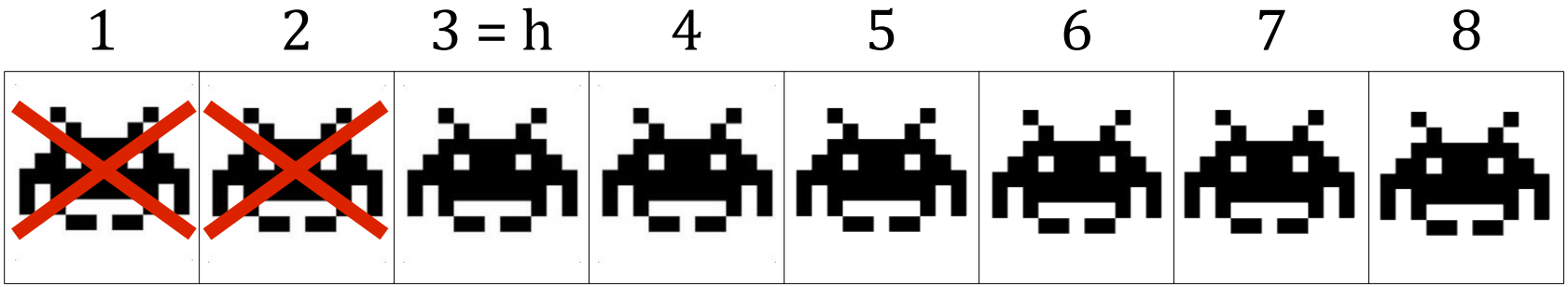
ответ — 3501.

$K, D \leq 1\,000$ — 30 баллов

Самое простое моделирование.
Заведём массив от 0 до $D + 1$, в каждой ячейке которого будем хранить количество жизней монстра.
На t-м шаге за $O(D)$ сдвинем всю армию вправо, в нулевую ячейку пишем $\left\lceil \frac{t}{K} \right\rceil$.
Если в $(D + 1)$-й ячейке > 0, то t — это ответ.
Иначе для каждой башни за $O(D)$ ищем самого правого живого монстра и убавляем ему жизнь.
Время работы — $O((K + D) * D)$.

$K, D \le 100\,000$ — 60 баллов

Другой по смыслу массив: в t-й момент времени в нём будет храниться количество жизней 1-го, 2-го,…, t-го монстра.
В произвольный момент первые несколько монстров мертвы, остальные живы. Пусть h – номер первого живого монстра

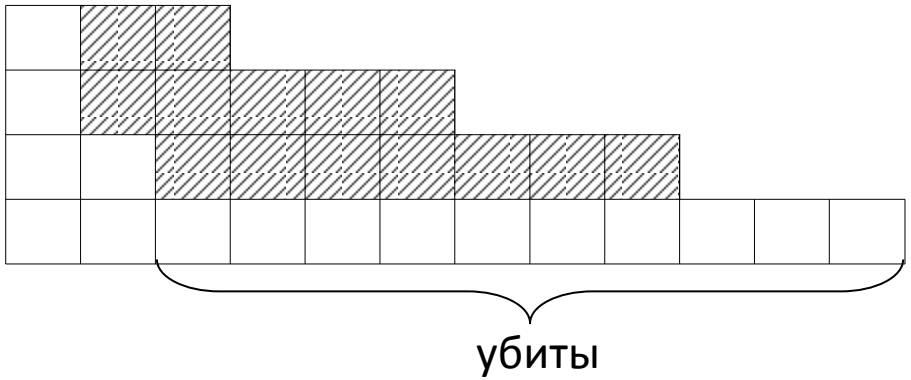


Не монстры будут двигаться относительно башен, а башни относительно монстров. Исходные координаты башен: −A, −B, −C, −D.

На t-м шаге в конец массива дописываем монстра с $\left\lceil \frac{t}{K} \right\rceil$. жизнями, сдвигаем башни вправо, h < D ⇒ конец, каждая башня стреляет по монстру, обновляем h при необходимости.

$K, D \leq 1\,000\,000\,000$ — 100 баллов

Моделируем поведение башни длины. Если монстры не умирают, есть L выстрелов до прорыва. Если же тот, по кому мы выстрелили, умер, мы получаем «бонусный» выстрел.



Как найти номер первого выжившего? Если у него P жизней, то $K + 2K + \ldots + (P-2)K < L \leq K + 2K + \ldots + (P-1)K \Leftrightarrow (P-2)(P-1) < 2L/K \leq (P-1)P$, т. е. $P = \max x\colon (x-2)(x-1) < 2L/K$.

Это P можно найти как простым перебором за $O(\sqrt{L})$, так и двоичным поиском за логарифм. Зная P, можем простыми вычислениями найти номер первого прорвавшегося.

Теперь вернёмся к исходной задаче, в которой у нас две башни.
Проблема в том, что к моменту подхода к первой башне армия имеет более сложный вид, чем изначально, и описывается не только числом K, но и ещё тремя числами V_0, V_1, K_1: количеством жизней у самого правого монстра, количеством жизней у следующей группы монстров и количеством монстров с V_1 жизнями.

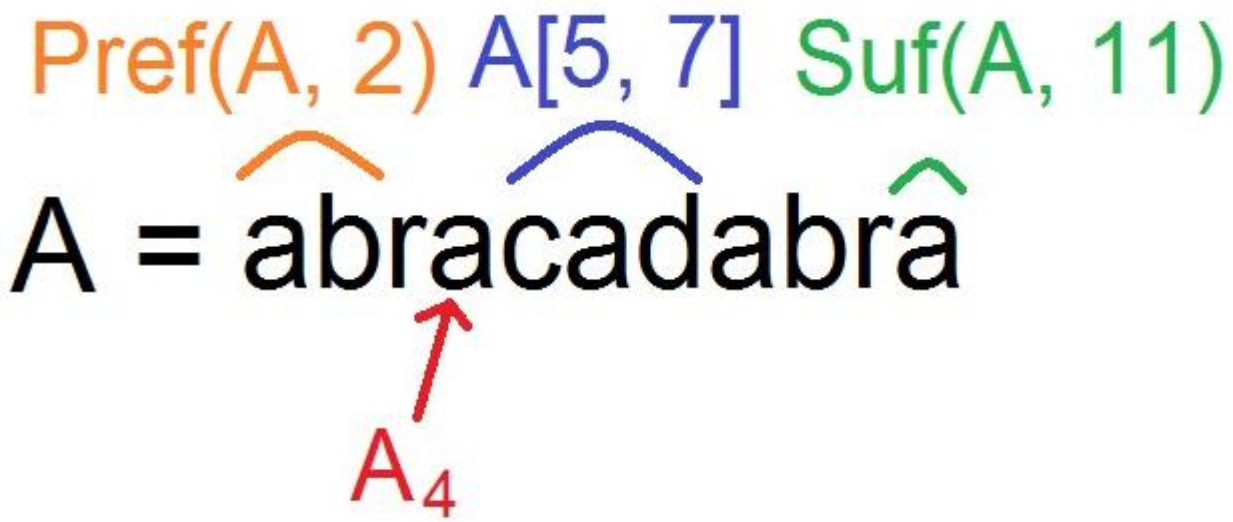
Три строки

Автор задачи и разработчик: Глеб Евстропов, МГУ

Обозначения

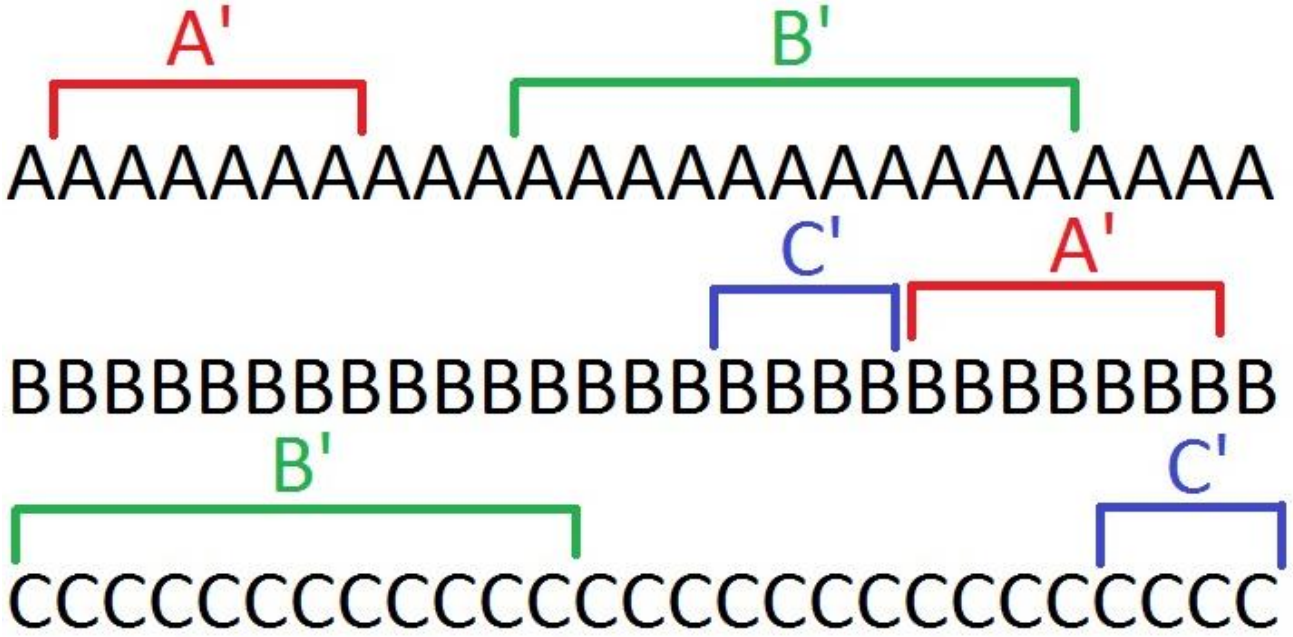
Пусть A - некоторая строка, будем обозначать:

- $|A|$ - длина строки.
- $A[i, j]$ - подстрока с i-ого по j-ый символ включительно.
- $Pref(A, i) = A[1, i]$ – i-ый префикс строки A.
- $Suf(A, i) = A[i, |A|]$ – i-ый суффикс строки A.
- $LCS(A, B)$ - длина наибольшей общей подстроки строк A и B.



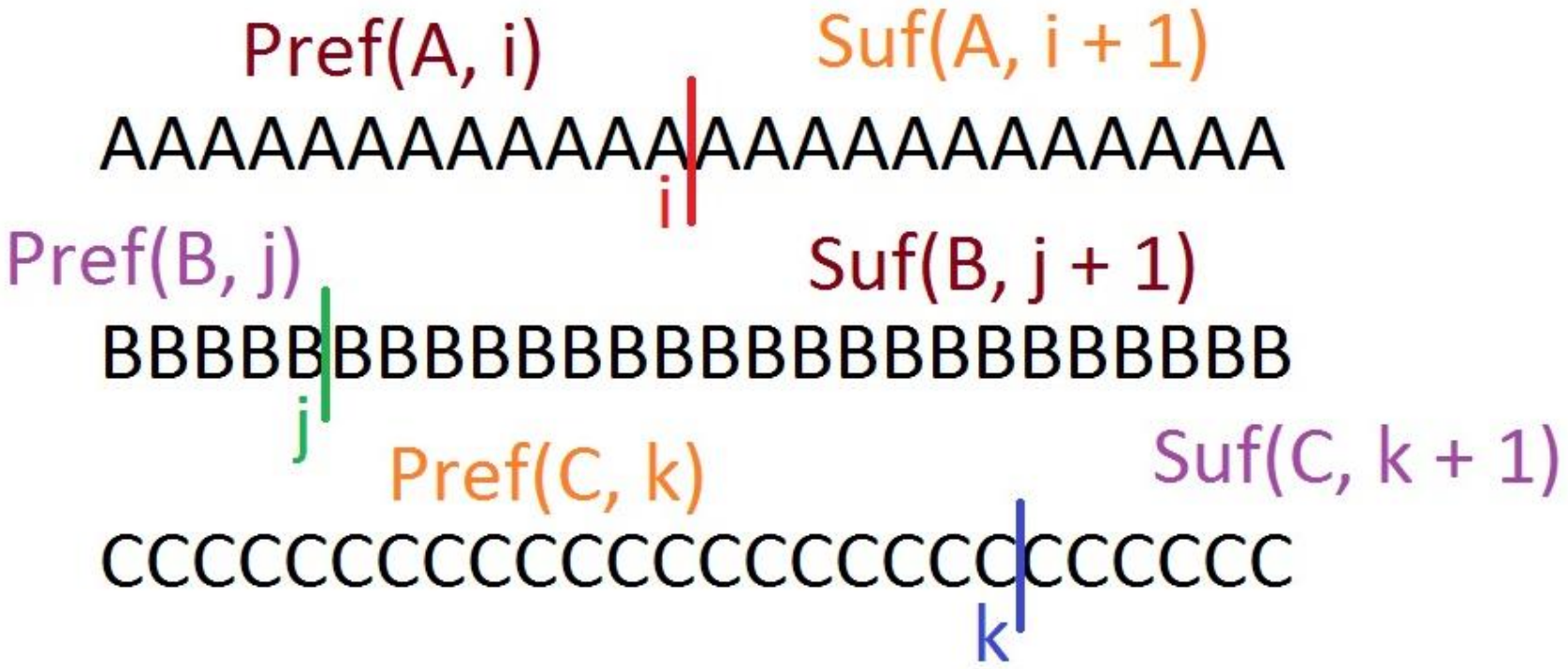
Формулировка задачи

Даны три строки A, B, C. Требуется выбрать строки A', B' и C' (возможно пустые, возможно совпадающие), так чтобы строка A удовлетворяла шаблону $*A'*B'*$, строка B удовлетворяла шаблону $*C'*A'*$, строка C удовлетворяла шаблону $*B'*C'*$ и величина $|\boldsymbol{A'}| + |\boldsymbol{B'}| + |\boldsymbol{C'}|$ **максимальна.**



Решение за $O(N^5)$

- Будем вычислять $LCS(A, B)$ за сложность $O(|A| \cdot |B|)$
- Переберем три границы i, j, k
- Для фиксированных i, j, k кандидатом на ответ является значение
$$LCS\big(Pref(A, i), Suf(B, j + 1)\big) + LCS\big(Pref(B, j), Suf(C, k + 1)\big) + LCS\big(Pref(C, k), Suf(A, i + 1)\big)$$
- Если написать аккуратно, то можно получить 30 баллов!



Решение за $O(N^4)$

- LCS всегда вычисляется от какой-то пары префикс-суффикс
- Вычисление LCS вызывается $O(N^3)$ раз, но количество различных пар префикс-суффикс есть $O(N^2)$
- Предподсчитаем $AB_{i,j} = LCS(Pref(A,i), Suf(B,j))$.

  Аналогично определим и вычислим $BC_{i,j}$ и $CA_{i,j}$.
- Сложность вычисления AB, BC и $CA - O(N^4)$
- После этого для фиксированных i, j, k ответ вычисляется за $O(1)$
- 30 баллов – гарантированно!

Вычисление AB[i][j] за $O(N^3)$

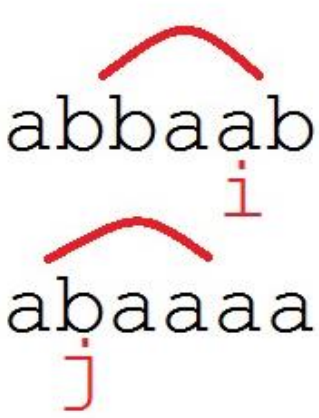
- p[i][j] - максимальное целое число, такое что

  A[i - p[i][j] + 1, i] = B[j, j + p[i][j] - 1]

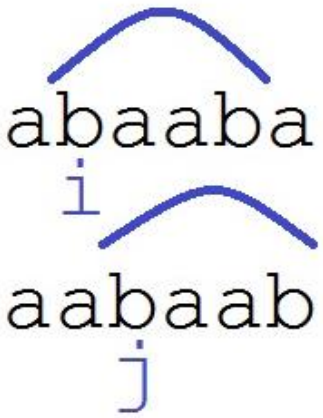
- AB[i][j] = max(AB[i - 1][j], AB[i][j + 1], p[i][j])
- d[i][j] - максимальное целое число, такое что

  A[i, i + d[i][j] - 1] = B[j, j + d[i][j] - 1]

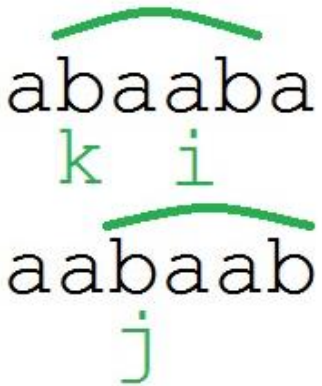
- Научимся вычислять p[i][j] с помощью d[i][j]



p[i][j] = 3



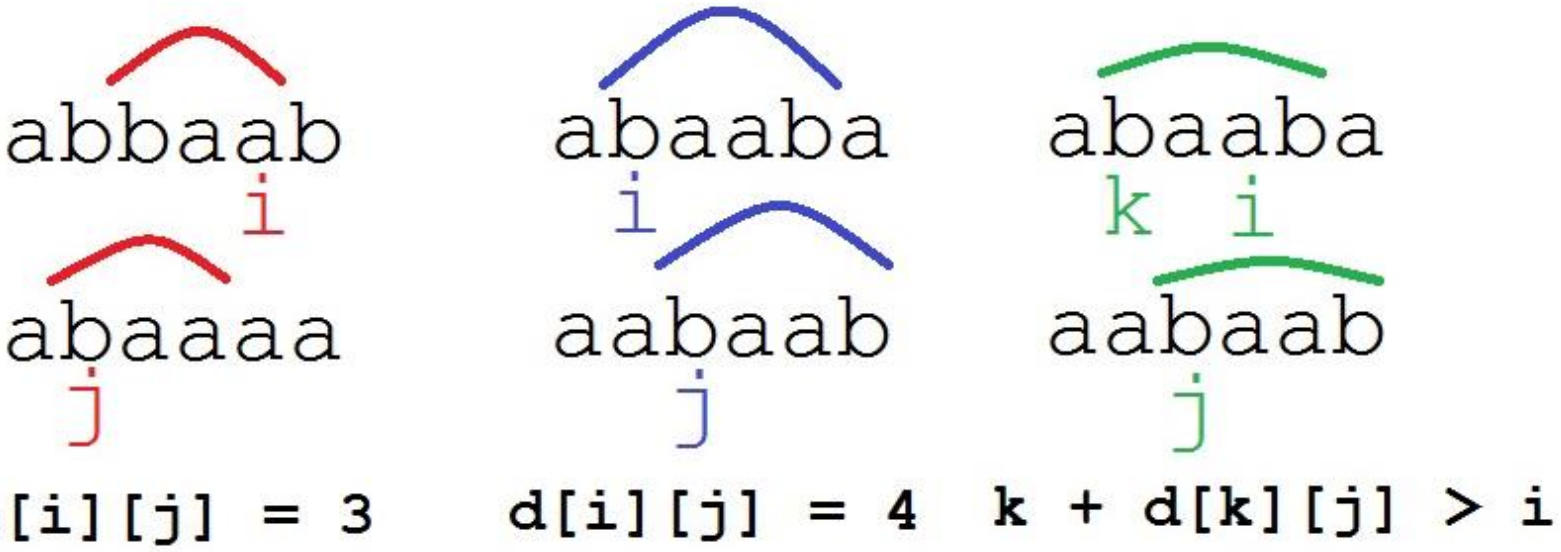
d[i][j] = 4



k + d[k][j] > i

Вычисление AB[i][j] за $O(N^3)$

- p[i][j] = i - k + 1, где k - минимальный индекс, такой что k ≤ i и k + d[k][j] > i. Если нет ни одного подходящего k, то p[i][j] = 0
- d[i][j] вычисляется за $O(N^2)$
- p[i][j] считается по d[i][j] за $O(N^3)$
- Итоговая сложность $O(N^3)$, что получает уже 60 баллов!



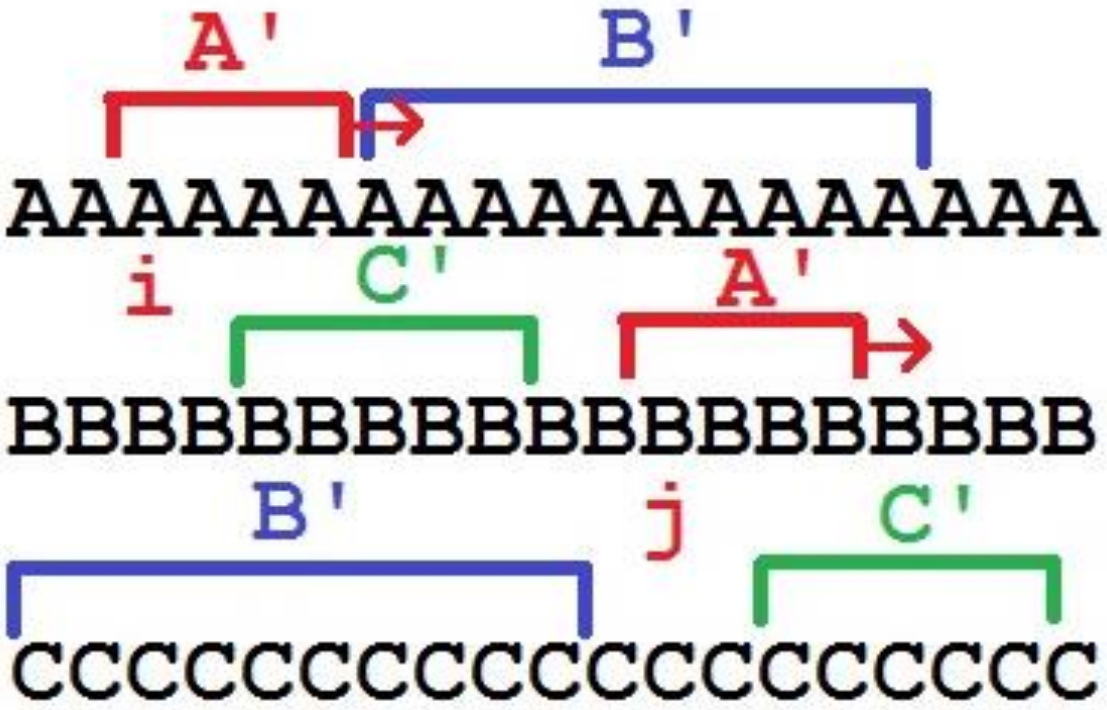
Вычисление AB за $O(N^2)$

- Пусть j фиксировано. k[i] - минимальный индекс, такой что k[i] + d[k[i]][j] > i (аналогично k с предыдущего слайда)
- k[i + 1] >= k[i], так как k[i + 1] + d[k[i + 1]][j] > i + 1 > i
- Если k[i] + d[k[i]][j] > i + 1, то k[i + 1] = k[i]
- В противном случае увеличиваем k[i + 1] пока неравенство не будет выполнено или не нарушится условие k[i + 1] ≤ i
- Суммарное время подсчета для фиксированного j теперь $O(N)$
- Таким образом мы построим p[i][j] по d[i][j] за $O(N^2)$
- Итог - AB[i][j] вычисляется за $O(N^2)$
- Сложность всего решения попрежнему $O(N^3)$, так как мы перебираем все тройки границ i, j, k

Решение за $O(N^2)$

- При фиксированной A' всегда имеет смысл брать ее самое левое вхождение в строку A и самое правое вхождение в строку B

- При фиксированных A', B' и C' и фиксированных позициях их вхождения в A, B и C, если $A_{i+|A'|} = B_{j+|A'|}$ то A' можно расширить вправо на один символ и ответ не уменьшиться. Действительно, в худшем случае при расширении A' вправо мы затронем вхождение B' в A, тогда мы можем просто сузить B' на один влево, что не ухудшит итоговую сумму.



Решение за $O(N^2)$

- Построим бор содержащий все суффиксы строк A и B
- Каждая вершина бора содержащая в поддереве концы обеих строк соответствует какой то общей подстроке A и B
- Каждой такой вершине можно назначить наиболее оптимальное вхождение в строку A и в строку B
- В силу соображения с предыдущего слайда имеет смысл рассматривать только те вершины бора, степень которых больше единицы.
- Вершины для которых выполнены оба условия назовем хорошими
- Для каждой хорошей вершины однозначно получаем вхождения соответстующей ей строки A' в строки A и B. Обозначим за i начало самого левого вхождения в A и за j начало самого правого

Решение за $O(N^2)$

- Теперь переберем границу в строке C и попробуем обновить максимум величиной $|A'| + BC[j - 1][k + 1] + CA[k][i + |A'|]$
- Проверка одной хорошей вершины бора занимает $O(N)$. Но такой бор есть не что иное как объединение суффиксных деревьев строк A и B, и, следовательно, содержит не больше вершин степени больше либо равной 2 чем сумма количеств таких вершин в исходных суфдеревьях. Но количество таких вершин в суффиксном дереве одной строки есть $O(N)$
- Таким образом мы получили только $O(N^2)$ границ которые достаточно проверить, чтобы гарантированно найти ответ. Это решение получает 100 баллов!