

Непалиндромное закливание

Авторы задачи и разработчики: Ильдар Гайнуллин и Андрей Матвеев

Далее под «путём» мы будем понимать **маршрут**: вершины и рёбра разрешается посещать несколько раз.

Основное наблюдение

Обозначим через L максимальную длину маршрута, в символах на ребрах которого встречаются **хотя бы две различные буквы**.

Тогда искомым ответ ans всегда удовлетворяет

$$\text{ans} \in \{L, L - 1\}.$$

Действительно, любой непалиндром содержит хотя бы две разные буквы, значит

$$\text{ans} \leq L.$$

Возьмём маршрут длины L . Если его слово уже не является палиндромом, то $\text{ans} = L$. Иначе это палиндром, но в нём есть хотя бы две разные буквы. Тогда хотя бы один из двух отрезков длины $L - 1$ — префикс или суффикс — не является палиндромом. Иначе оба они были бы палиндромами, откуда легко следует, что все символы исходного слова одинаковы, что невозможно.

Итак:

$$\text{ans} = \begin{cases} -1, & \text{если } L = \infty, \\ 0, & \text{если не существует маршрута, содержащего хотя бы две разные буквы,} \\ L \text{ или } L - 1, & \text{иначе.} \end{cases}$$

Таким образом, задача распадается на две части:

- найти L или понять, что $L = \infty$;
- в конечном случае определить, равен ответ L или $L - 1$.

Группа 1

При $n, m \leq 5$ проходит аккуратный полный перебор, например достаточно перебрать все пути длины $\leq n + 2$, и если есть путь длины $> n$, то ответ бесконечный. А иначе, путей достаточно мало и можно просто взять из них самый длинный, проверяя на палиндромность наивным образом.

Группы 2, 3, 4

Здесь удобно сделать динамику по парам вершин.

Пусть $\text{mid}[u][v]$ — максимальная длина **произвольного** маршрута из u в v . Значение может быть:

- не существует;
- конечное число;
- бесконечность.

Сначала сожмём граф в компоненты сильной связности. Если на каком-то пути из u в v встречается компонента с циклом, то $\text{mid}[u][v] = \infty$. Иначе все маршруты лежат в DAG'е, и максимум считается обычной динамикой по DAG'у.

Теперь определим $\text{base}[u][v]$ — максимальную длину непалиндромного маршрута из u в v , у которого **первая и последняя** буквы уже различны. Тогда

$$\text{base}[u][v] = \max(2 + \text{mid}[x][y]),$$

где максимум берётся по всем рёбрам

$$u \xrightarrow{a} x, \quad y \xrightarrow{b} v, \quad a \neq b.$$

Если какой-то из таких $\text{mid}[x][y] = \infty$, то и $\text{base}[u][v] = \infty$.

Теперь построим граф состояний (u, v) . Переход

$$(u, v) \rightarrow (x, y)$$

существует, если найдётся буква c такая, что есть рёбра

$$u \xrightarrow{c} x, \quad y \xrightarrow{c} v.$$

Смысл такой: если слово начинается и заканчивается одной и той же буквой c , то эти две буквы можно удалить, и мы попадём в состояние (x, y) .

Тогда для $\text{dp}[u][v]$ — максимальной длины непалиндромного маршрута из u в v — верно:

$$\text{dp}[u][v] = \max\left(\text{base}[u][v], 2 + \max \text{dp}[x][y]\right),$$

где максимум берётся по всем переходам $(u, v) \rightarrow (x, y)$.

Бесконечность возникает в двух случаях:

- существует состояние с $\text{base}[u][v] = \infty$;
- в графе состояний (u, v) есть цикл, из которого можно добраться до состояния, где существует подходящая база.

Иначе этот граф можно свести к DAG'у и посчитать dp в порядке топологической сортировки. Этого достаточно для ограничений $n, m \leq 200$. Время работы $\mathcal{O}(n(n + m) + m^2)$.

Группа 5

В этой подзадаче известно, что ответ либо нечётный, либо равен 0, либо равен -1 . Значит достаточно найти только L .

Построим автомат по множеству уже встреченных букв. Для каждой вершины v исходного графа создадим состояния:

- (v, S) — пока не прошли ни одного ребра;
- (v, C_x) — пока встречалась только буква x ;
- (v, M) — уже встретились хотя бы две разные буквы.

Для каждого ребра $u \xrightarrow{c} v$ добавим переходы автомата:

- $(u, S) \rightarrow (v, C_c)$;
- $(u, C_c) \rightarrow (v, C_c)$;
- $(u, C_x) \rightarrow (v, M)$ для всех $x \neq c$;
- $(u, M) \rightarrow (v, M)$.

Стартовыми считаем все состояния (v, S) , а финишными — все состояния (v, M) .

Теперь длина маршрута в автомате равна длине маршрута в исходном графе, а маршрут заканчивается в состоянии M тогда и только тогда, когда в его слове встретились хотя бы две разные буквы.

Оставим только те вершины автомата, которые достижимы из старта и из которых достигим финиш.

- Если в этом подграфе есть цикл, то $L = \infty$, а значит ответ равен -1 .
- Иначе это DAG, и L ищется как максимальный путь в DAG'е.

После этого ответ восстанавливается сразу:

- если маршрута с двумя разными буквами не существует, то ответ 0;
- если L нечётно, то ответ L ;
- если L чётно, то ответ $L - 1$.

Группа 6

Здесь предполагается более аккуратно написанное решение прошлых подзадач.

Группа 7

Здесь для всех рёбер выполняется $u_i < v_i$, то есть исходный граф уже является DAG'ом.

Отсюда сразу следует, что бесконечного ответа быть не может. Автомат из предыдущего пункта тоже будет DAG'ом, так как каждое ребро по-прежнему идёт в вершину с большим номером.

Дальше:

- находим L в автомате;
- оставляем только рёбра, лежащие на каком-то максимальном пути длины L ;
- проверяем, существует ли среди этих путей непалиндром.

Проверка делается через середину маршрута; тот же приём будет использован и в полном решении ниже.

Группа 8

В этой подзадаче алфавит бинарный, то есть встречаются только буквы 'a' и 'b'.

Поэтому автомат сильно упрощается: для каждой вершины достаточно всего 4 состояния:

- S — ещё не прошли ни одного ребра;
- A — пока встречали только 'a';
- B — пока встречали только 'b';
- AB — уже встретили и 'a', и 'b'.

Дальше всё полностью совпадает с полным решением: находим L , а затем решаем, равен ли ответ L или $L - 1$.

Полное решение

Распишем по шагам все необходимые идеи:

1. Автомат по множеству встреченных букв

Для каждой вершины v исходного графа создадим 28 состояний:

- (v, S) — ещё не прошли ни одного ребра;
- (v, C_x) для каждой буквы x — пока встречалась только буква x ;
- (v, M) — уже встретились хотя бы две разные буквы.

Для каждого ребра $u \xrightarrow{c} v$ добавим переходы:

- $(u, S) \rightarrow (v, C_c)$;
- $(u, C_c) \rightarrow (v, C_c)$;
- $(u, C_x) \rightarrow (v, M)$ для всех $x \neq c$;
- $(u, M) \rightarrow (v, M)$.

Количество вершин автомата равно $28n$, количество рёбер — $28m$.

Стартовые состояния — все (v, S) , финишные — все (v, M) .

2. Находим L и проверяем бесконечность

Сделаем два обхода:

- из всех стартовых состояний;
- по обратным рёбрам из всех финишных состояний.

Оставим только вершины, которые достижимы из старта и из которых достигим финиш.

- Если в этом подграфе есть цикл, то можно сколько угодно раз обходить этот цикл и затем дойти до финиша. Значит $L = \infty$, а ответ равен -1 .
- Иначе это DAG.

На этом DAG'е посчитаем:

- $d_1[v]$ — максимальную длину пути из старта в вершину v ;
- $d_2[v]$ — максимальную длину пути из вершины v в финиш.

Тогда

$$L = \max d_1[v]$$

по всем финишным вершинам.

Если ни один финиш недостижим, то ответа вообще нет, и нужно вывести 0.

3. Оставляем только максимальные пути

Теперь нас интересуют только маршруты длины ровно L .

Оставим только те рёбра $u \rightarrow v$, для которых

$$d_1[u] + 1 + d_2[v] = L.$$

Именно эти рёбра лежат хотя бы на одном максимальном пути длины L .

После этого получаем новый DAG со свойством:

- все пути из старта в финиш имеют длину ровно L ;
- для каждого ребра $u \rightarrow v$ выполняется $d_1[v] = d_1[u] + 1$.

То есть граф становится **слоистым**.

4. Что нужно знать о префиксах и суффиксах

Для каждой вершины v этого DAG'а хотим знать:

- какие строки можно получить на путях из старта в v ;
- какие строки можно получить на путях из v в финиш, если читать их в обратном порядке.

Но хранить множество строк целиком не нужно. Достаточно трёх состояний:

- **NONE** — путей нет;
- **ONE(id)** — все такие строки одинаковы;
- **MANY** — существует хотя бы две разные строки.

Чтобы состояние **ONE(id)** было точным, а не вероятностным, построим бор всех возникающих строк.

- Пустой строке дадим идентификатор 0.
- Операция **extend(id, c)** возвращает идентификатор строки, полученной приписыванием буквы c справа.

Тогда две строки равны тогда и только тогда, когда равны их идентификаторы.

Слияние двух состояний определяется так:

- $\text{merge}(\text{NONE}, X) = X$;
- $\text{merge}(\text{MANY}, X) = \text{MANY}$;
- $\text{merge}(\text{ONE}(a), \text{ONE}(b)) = \text{ONE}(a)$, если $a = b$;
- $\text{merge}(\text{ONE}(a), \text{ONE}(b)) = \text{MANY}$, если $a \neq b$.

5. Считаем префиксы и суффиксы

Пусть $\text{pref}[v]$ — информация обо всех строках на путях из старта в v .

- Для стартовых вершин $\text{pref} = \text{ONE}(0)$.
- Дальше идём по слоям слева направо.
- Если есть ребро $u \xrightarrow{c} v$, то из состояния $\text{pref}[u]$ получаем кандидат для $\text{pref}[v]$ приписыванием буквы c .

Аналогично считаем $\text{suf}[v]$ — информацию обо всех **перевёрнутых** строках на путях из v в финиш:

- для финишных вершин $\text{suf} = \text{ONE}(0)$;
- идём по слоям справа налево;
- по ребру $u \xrightarrow{c} v$ переносим информацию из $\text{suf}[v]$ в $\text{suf}[u]$, тоже через $\text{extend}(\dots, c)$.

6. Проверяем середину

Теперь нужно понять, существует ли среди максимальных путей длины L непалиндром.

Случай 1: $L = 2k$. Тогда середина — это вершина слоя k .

Для такой вершины любой максимальный путь через неё имеет вид:

$$P + Q,$$

где $|P| = |Q| = k$.

Такой путь является палиндромом тогда и только тогда, когда строка P совпадает с перевёрнутой строкой Q .

Значит:

- если существует вершина v слоя k , для которой $\text{pref}[v]$ и $\text{suf}[v]$ **не являются одним и тем же** состоянием **ONE(id)**, то можно выбрать левую и правую половины так, чтобы получить непалиндром длины L ;
- иначе все максимальные пути являются палиндромами, и ответ равен $L - 1$.

Случай 2: $L = 2k + 1$. Тогда середина — это ребро $u \rightarrow v$, где $d_1[u] = k$.

Полностью аналогично:

- если существует такое среднее ребро, что $\text{pref}[u]$ и $\text{suf}[v]$ **не являются одним и тем же** состоянием $\text{ONE}(\text{id})$, то существует непалиндром длины L ;
- иначе все максимальные пути палиндромны, и ответ равен $L - 1$.

7. Почему проверка середины корректна

После фильтрации на максимальные пути граф стал слоистым, а все пути из старта в финиш имеют одинаковую длину L .

Значит палиндромность полностью определяется тем, совпадает ли левая половина пути с перевёрнутой правой половиной.

Если в некоторой середине с одной стороны возможны две разные строки, либо единственные возможные строки слева и справа различаются, то можно выбрать несовпадающие половины и получить непалиндром.

Если же в каждой середине возможна ровно одна строка слева и ровно одна строка справа, причём они совпадают, то любой максимальный путь будет палиндромом.

Время работы

Пусть $K = 28$ для полной задачи и $K = 4$ для бинарного алфавита. Тогда:

- число вершин автомата — $\mathcal{O}(Kn)$;
- число рёбер автомата — $\mathcal{O}(Km)$;
- все обходы, топологическая сортировка и динамики работают за $\mathcal{O}(K(n + m))$;
- бор строк тоже строится за линейное время от числа обработанных рёбер.

Итоговая сложность: $\mathcal{O}(K(n + m))$ по времени и памяти. Для полной задачи это $\mathcal{O}(28(n + m))$, что укладывается в ограничения.

Бонус

Эту задачу можно решить даже в случае, если ребра могут иметь произвольные веса, не ограничиваясь 26 буквами латинского алфавита, например натуральные числа от 1 до m . Для этого достаточно применить одну достаточно классическую идею: давайте построим коллекцию из не очень большого числа множеств $S_1, S_2, \dots, S_k$, что для каждой пары чисел $1 \leq x < y \leq n$, найдется k , что $x \in S_k, y \notin S_k$. Для этого достаточно взять как очередное множество все такие натуральные числа, что бит 2^k отсутствует в двоичном разбиении этого числа.

Таким образом мы можем $\log m$ раз запустить решение для двоичного алфавита.