

Добрый эксперимент

Автор задачи: Антон Вдовин, разработчики: Даниил Орешников и Антон Вдовин

Сначала заметим важный факт, который будет использоваться почти везде: для любой открывающей скобки в ПСП существует отрезок, начинающийся в этой позиции и являющийся ПСП, а для любой закрывающей скобки такого отрезка не существует, потому что никакая ПСП не может начинаться с символа ')'. При внимательном рассмотрении первый факт становится очевидным: для каждой открывающей скобки справа есть «парная» закрывающая, отрезок между ними также является ПСП.

Дальше разберём решения по подгруппам, постепенно улучшая их.

Группа 1

Здесь $n = 6$, а число правильных скобочных последовательностей длины 6 равно 5 (Числа Каталана). Поэтому можно просто перебрать все возможные ПСП длины 6, построив их, например, вручную, и вбив к код как константы.

После этого можно проверить каждого «кандидата»: спросить ответы для всех возможных отрезков. Таких отрезков ровно $\frac{n(n+1)}{2} = 21$. Возьмем каждого кандидата, сравним каждый его подотрезок на соответствие ответу интерактора на этом подотрезке, и если ответы расходятся, то эта строка не является ответом. Итого нам нужно проверить 21 отрезков в 4 из 5 кандидатах (если ни один из четырех не подошел, то ответ пятый), суммарно выходит 84 запроса.

Группа 2

Вторая группа решается так же, как и первая, за исключением того, что нужно придумать как сгенерировать все ПСП длины 10. Это делается стандартным рекурсивным перебором: то, что обозначено в условии как «глубина» ПСП, стандартно называется *балансом*. Альтернативное определение ПСП заключается в том, что суммарный баланс равен 0 (у каждой скобки есть парная) и минимальный баланс тоже равен 0 (ни на каком префиксе не может быть больше открывающих скобок, чем закрывающих). Поэтому в рекурсивный генератор достаточно передавать префикс и текущий баланс, а затем перебирать, какую скобку ставить следующей.

Группа 3

Здесь $n = 22$, $q = 59000$. Перебор всё ещё проходит. А количество ПСП длины 22 равно числу Каталана $C_{11} = 58786$.

Заметим, что число различных ПСП чуть меньше допустимого количества запросов. Давайте научимся за один запрос отбрасывать одного кандидата на ответ. Среди кандидатов выберем две любые строки, затем найдем такой подотрезок $[l; r]$, что в одной строке этот подотрезок будет ПСП, а в другой — нет. Такой всегда существует, причем находить его можно как за $\mathcal{O}(n^2)$, так и за $\mathcal{O}(n)$. Таким образом, сделав запрос на этом отрезке, мы узнаем, какую из этих строк нужно убрать из кандидатов.

Группа 4

В этой группе гарантируется, что глубина (максимальный баланс) последовательности не превосходит 2. Это значит, что текущий баланс при просмотре строки слева направо может принимать только значения 0, 1 и 2.

Рассмотрим, как восстанавливать строку слева направо.

- Если текущий баланс равен 0, то следующий символ обязан быть '(', потому что ПСП не может начинаться с закрывающей скобки (баланс не может опускаться ниже нуля).
- Если текущий баланс равен 2, то следующий символ обязан быть ')', потому что баланс больше двух запрещен.

- Единственный нетривиальный случай — когда текущий баланс равен 1. Но тогда следующий символ можно определить одним запросом к соседней паре.

Действительно, если к позиции $i - 1$ текущий баланс равен 1, то сделаем запрос $[1, i]$. Если следующий символ — закрывающая скобка, баланс станет равен нулю, и интерактор сообщит YES. Иначе, интерактор сообщит NO. Так мы восстанавливаем всю строку слева направо за линейное число запросов.

Группа 5

Здесь $n \leq 500$, а $q = 2 \cdot 10^5$. Этого уже достаточно, чтобы спросить ответы для всех отрезков, потому что их всего $\frac{n(n+1)}{2} \leq 125250$.

После этого строку можно восстановить напрямую. Действительно, любая ПСП начинается на открывающую скобку и заканчивается закрывающей. Таким образом, если на какой-то запрос $[l, r]$ мы получили YES, то $s_l = '('$ и $s_r = ')'$. Утверждается, что если сделать запросы про каждый возможный отрезок, мы так восстановим всю строку.

Действительно, рассмотрим произвольную позицию i . Если на ней стоит открывающая скобка, у нее есть парная закрывающая справа: нам подойдет отрезок между ними, при запросе в него s_i будет выставлено. Аналогично и симметрично для закрывающей. Это уже даёт полное решение пятой группы за $O(n^2)$ запросов и времени.

Группа 6

Заметим, что первый символ любой непустой ПСП обязан быть '(', поэтому достаточно восстановить позиции от 2 до n . Будем восстанавливать позиции по очереди и предположим, что префикс $s_1 s_2 \dots s_{i-1}$ уже восстановлен. Среди найденных открывающих скобок будем отмечать те, которые уже получили себе пару внутри этого префикса.

Далее здесь подойдет любое решение, заимствующее идею полного решения ниже, но менее эффективно реализующее ее: менее эффективный поиск пары к последней открывающей скобке, другой порядок обхода скобок, любое решение за $\omega(n)$ времени, но $o(q^2)$ запросов. В частности можно было написать что-то похожее на «разделяй и властвуй»:

1. чтобы восстановить ПСП $[l, r]$, сначала сделаем запрос $[l + 1, r - 1]$, поймем, была ли она получена добавлением скобок вокруг другой ПСП;
2. если да, перейдем к $[l + 1, r - 1]$;
3. если нет, будем чередовать запросы $[l, l + i]$ и $[r - i, r]$ по возрастанию i , найдем первый индекс разбиения на две ПСП, перейдем рекурсивно к двум частям.

За счет того, что мы идем одновременно и с левого, и с правого конца, при некоторых оптимизациях такое решение могло пройти эту группу.

Наблюдение

Для следующих групп нужен уже более экономный способ. Одной из важных идей является делать запросы на отрезках длины 2: так мы можем определять стоящие рядом открывающую и закрывающую скобку. Осталось разобраться, что делать в случае, например, строки «(())». На самом деле ничего сильно отличающегося: воспользуемся стандартной идеей проверки правильности скобочной последовательности со стеком.

Будем восстанавливать строку слева направо и хранить стек позиций ещё не нашедших пару открывающих скобок среди уже восстановленных символов. Например, для префикса «(())(())» будем хранить 1 и 4: у первой и третьей открывающих скобок на этих позициях пока нет пары. Тогда верно то, что ближайшая закрывающая скобка будет парной к последней открывающей в стеке.

Сделаем первый запрос в $[1; 2]$. Если ответ YES, то $s_1 = '('$, а $s_2 = ')'$, иначе гарантированно $s_2 = '('$. Продолжим тем же образом: будем каждый новый запрос делать в $[\text{stack.last}(), i]$, где i — первая неизвестная позиция. Ответ интерактора в точности определит s_i .

Группа 7

Возьмем наблюдение выше и реализуем его. Опять же, как и в шестой группе, эта группа тестов допускала неэффективности в реализации: например, вместо хранения стека можно было за линейное время искать парную открывающую к потенциальной закрывающей на еще неизвестной позиции. Таким образом можно было получить решение за $\mathcal{O}(n^2)$ времени и q запросов.

Группа 8

Здесь $q = n$.

Во всех подгруппах, начиная с этой, достаточно было просто эффективно реализовать описанный выше подход со стеком. Поддерживаем стек, делаем запрос $[\text{stack.last}(), i]$:

- если интерактор сообщил YES, то удаляем верхний элемент из стека, и новая скобка — закрывающая;
- иначе, новая скобка открывающая, и мы добавляем её в стек.

Решение работает за $\mathcal{O}(n)$ и $q = n - 1$, но эта группа допускала возможность сделать один лишний запрос при неэффективной реализации.

Группы 9, 10, 11

Теперь требуется уложиться в q от $n - 1$ до $n - 3$. Для начала достаточно реализовать в точности ту идею, которая была описана выше, это дает решение за $q = n - 1$. Посмотрим теперь, где можно срезать углы.

К предыдущему наблюдению добавим ещё одно: последний символ любой непустой ПСП обязательно равен $)'$, значит, запрос $[n - 1, n]$ не приносит новой информации, и восстанавливать надо только позиции от 2 до $n - 1$. Тем самым число запросов уменьшается до $n - 2$.

Осталось уложиться в $q = n - 3$. Нужно избавиться ещё от одного запроса. Для этого не будем восстанавливать последнюю *пару* символов запросами.

Будем восстанавливать строку стековым способом только до позиции $n - 2$. После этого останутся две последние позиции, пусть текущий баланс перед ними равен b . За два оставшихся символа итоговый баланс должен стать равен нулю. При этом за один символ баланс меняется только на ± 1 , значит за две позиции он может уменьшиться максимум на 2. Следовательно, b не может быть больше 2.

Кроме того, длина уже восстановленного префикса равна $n - 2$, а это чётное число. Последняя скобка нам известна, она закрывающая, а значит предпоследняя однозначно определяется по числу $b - 1$, которое будет равно ± 1 . Итак, достаточно восстанавливать запросами только позиции от 2 до $n - 2$. Это и даёт ровно $n - 3$ запроса.

Итог

В полном решении запросы задаются только для позиций от 2 до $n - 2$, и на каждую такую позицию делается не более одного запроса. Значит, общее число запросов не превышает $n - 3$. Более общую и эффективную логику, которая приводит к не менее оптимальному результату, можно сформулировать следующим образом:

- не делаем запрос, если текущий баланс равен нулю: следующая обязательно открывающая;
- не делаем запрос, если текущий баланс равен $n - i$: **все** оставшиеся скобки обязательно закрывающие.