	На пълното решение	На частичните решения
Тагове	RMQ Sparse table Segment Tree	Brute force Dynamic programming Greedy Monotone stack

Анализ

Подзадача №1

Както винаги, оставих подзадача с тестовите примери за обратна връзка от системата.

Подзадача №2

За подзадачата е предвидено от състезателите да напишат пълно изчерпване. Авторовото пълно изчерпване е разглеждане на всяко възможно пермутиране на скобите, проверява се, дали то е правилно скобуван израз и се изчисляват минималния брой размени, за да се постигне (равно на $N - count\_cycles$, където $count\_cycles$ е броя цикли в пермутацията).

Постигната сложност: $O(Q|S|!)$

Имплементация: `parentheses_5p.cpp`

Подзадача №3

Стандартно, може да заменим всяка отваряща скоба с $+1$ и всяка затваряща скоба с -1. Тогава низ ще е правилно скобуван, ако сборът на всички числа в получената редица е нула, което важи по условие, както и когато не съществува префикс с отрицателен сбор.

Логично е да се търси някакво динамично програмиране за решаване на задачата при $Q = 1$. Може да се поддържа динамично със стейт `dp[позиция][префиксен сбор до момента][брой недовършени размени]` и да се разглежда дали текущата скоба участва в размяна. Може да се забележи, че няма смисъл да се разменят скоби от един и същи вид, следователно всички размени ще са непресичащи се, което улеснява динамичното.

Постигната сложност: $O(Q|S|^3)$

Имплементация: `parentheses_15p.cpp`

Подзадача №4

Може да се забележи, че `[брой недовършени размени]` е тясно свързано с `[префиксен сбор до момента]`. Ако префиксния сбор в първоначалния низ до позиция pos е p_{pos}, тогава `[брой недовършени размени]` $=$ `[префиксен сбор до момента]` $- p_{pos}$. Така може да премахнем параметъра за брой размени и да подобрим сложността на динамичното ни с един порядък.

Постигната сложност: $O(Q|S|^2)$

Имплементация: `parentheses_25p.cpp`

Подзадача №5

Много е трудно двумерно динамично от гореспоменатия вид да се сведе до едномерно. Заради това трябва да сменим подхода си. За решение на подзадачата може да изпълним следната лакома стратегия:

- Поддържаме префиксен сбор.
- Ако текущата скоба е затваряща и води до отрицателен префиксен сбор, то задължително след нея ще има отваряща скоба, като се разменя с най-дясната такава.
- В противен случай, нищо не се прави.

Алгоритъма съм го доказал по-надолу в анализа. Той може да се напише имплементира за $O(Q|S|^2)$ време и да изкара 25 точки, но също може да се забележи, че при изпълнение на размяна, всяка следваща скоба е по-наляво от предишната, от което следва, че може да приложи метода на показалките.

Постигната сложност: $O(Q|S|)$

Имплементация: `parentheses_40p.cpp`

Подзадача №6

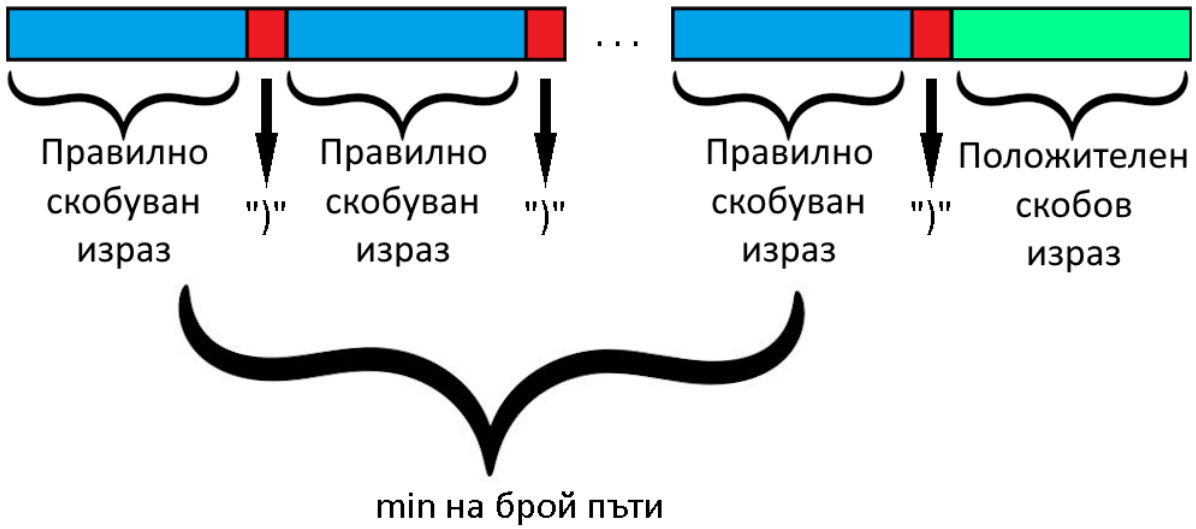
В подзадачата се цели да се отговорят на въпроси от вида *Подниза, образуван от символи от позиция L_i до позиция R_i правилно скобуван ли е?*. По условие подниза има равен брой отварящи и затварящи скоби, от което следва, че единствено трябва да се провери е дали той има отрицателен префикс. Нека префиксите на S да са $p_1, p_2, \ldots, p_{2N}$. Тогава ще целим да проверим дали съществува позиция $x, (L_i \leq x \leq R_i)$, за която $p_x - p_{l-1} < 0$, тоест $p_x < p_{l-1}$. Така ще желаем да проверим дали съществува префикс p_x, по-малък от p_{l-1}. Това може да бъде реализирано по множество начини, но може би най-елегантния от тях е чрез стек да се намери за всеки префикс p_x първия друг префикс отдясно p_{next_x}, който е по-малък от него и при постъпване на заявка (L_i, R_i) да се провери дали $next_{L_i-1} \leq R_i$.

Постигната сложност: $O(N + Q)$

Имплементация: `parenthesesStack_15p.cpp`

Доказателство на лакомата стратегия от пета подзадача

При изпълнението на лакомата стратегия винаги се разменя затваряща скоба отляво с отваряща скоба отдясно. При такава размяна префиксите търпят $+2$ рейндж ъпдейти между позициите, обхванати от размяната. Нека разгледаме минималния префикс в редицата. Нека той да е min. Винаги $min \leq 0$. Ние ще трябва чрез операциите да го направим положителен, за да постигнем целта ни. Тъй като всяка размяна води до $+2$ увеличение на част от префиксите, то в най-добрия случай ще ни трябват поне $\lceil \frac{-min}{2} \rceil$ размени. Оказва се, че лакомата ни стратегия винаги постига толкова. Нека разгледаме структурата на низа ни:



Под положителен скобов израз имам в предвид такъв, който няма отрицателен префикс и има повече отварящи, отколкото затварящи скоби. Той ще има подобна структура на минималния префикс, като ще се разбие на аналогични суфиксни подмасиви, разделени с отварящи скоби. Естествено, изображението описва главния случай, като може да има изродени случаи, например когато низът отначало е правилно скобуван и липсват червени скоби, или пък когато някои сини блокчета са с дължина 0. На последната червена скоба в това изображение е крайната позиция на минималния префикс.

Лакомият алгоритъм ще замени първата червена с отваряща, като тази скоба ще се сдвои с втората червена, след което ще замени третата червена с отваряща, която ще се сдвои с четвъртата и т.н. Ако приемем, че размяната на отварящите скоби не допринася към някакви проблеми, то ще се окаже, че ще направим по точно една размяна за всяка нечетна червена скоба. Всъщност се случва точно това. Тъй като общият баланс на скобите в низа е 0, то всички размени ще включват червена скоба и отваряща от зеления участък. Така единственият възможен проблем би бил, ако зеленият участък стане неправилен. Тъй като балансът в зеления участък е $-min$, а ние от него взимаме $\lceil \frac{-min}{2} \rceil$ скоби, които не навреждат на структурата му, то след всичките направени размери, в него минималния префикс ще стане 0. Формално погледнато, в положителния скобов израз ще има поредица от $-min$ на брой несдвоени отварящи скоби, като след всяка размяна от тях ще се премахват по 2 на брой от тях и по аналогични причини на минималния префикс, те след прилагането на операциите ще образуват правилен скобов израз. В крайна сметка, двете групи от несдвоени и сдвоени отварящи ще се претърпят промяна от)))…(((до ()()()…().

Подзадача №6

Доказателството на лакомата стратегия излезна, че е коз за нашето решение – вече знаем по-добър начин за изчисление на броя операции, които ще направи, от директното прилагане на грийдито. Така задачата ни се сведе до заявки от вида *Намери минималния префикс от L_i до R_i*. Тъй като ние търсим $min(p_x - p_{L_i-1})$, то ние реално погледнато желаем да намерим $min(p_x) - p_{L_i-1}$. Решението се сведе до стандартната $Range\ Minimum\ Query$ задача, която стандатно решава със Sparse таблица, или със Сегментно дърво, като дори може да се реши за $O(N+Q)$ време с тази техника или тази техника. При заявка от L_i до R_i и $min = min(p_{L_i}, p_{L_i+1}, p_{L_i+2}, \ldots, p_{R_i}$, отговорът й ще е $\lceil \frac{p_{L_i-1} - min}{2} \rceil$.

Постигната сложност: $O(Nlog_2N + Q)$

Имплементация: `parantheses_100p.cpp`

Автор: Борис Михов