Анализ
на задачата Лампички

Първата подзадача може да бъде решена по няколко начина. Ще опишем решението, използвайки хеширане, което ще бъде полезно за последната подзадача. Нека вкореним дървото във връх R. Интересуват ни всички палиндромни сегменти с единия край във връх R. За всеки връх X ще изчислим две стойности:

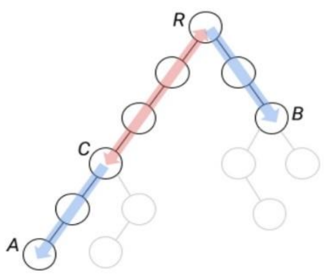
$$\mathtt{down}_x = x_0B^{k-1} + x_1B^{k-2} + \cdot\ \cdot\ \cdot + x_{k-1}B^0$$
$$\mathtt{up}_x = x_0B0 + x_1B^1 + \cdot\ \cdot\ \cdot + x_{k-1}B^{k-1}$$

където $x_0, x_1, \ldots, x_{k-1}$ е масив от цветове по пътя от R до X ($\mathtt{color(R)} = x_0$, $\mathtt{color(X)} = x_{k-1}$) и B е стойността на хеш базата. Тези две стойности могат да бъдат определени с помощта на еднократно DFS обхождане на нашето дърво. Пътят от R до X е палиндромен сегмент, ако е валидно $\mathtt{down}_X = \mathtt{up}_X$. Ако вкореним дървото във всеки връх и приложим тази процедура, ще покрием всички случаи. Сложността на този алгоритъм е $O(N^2)$.

Втората подзадача е класическа, добре позната задача за намиране на най-дългия палиндромен подниз, която може да бъде решена с помощта на **алгоритъма на Манахер***.

Решението на **третата подзадача** е доста подобно на решението на втората подзадача. Единствената разлика е, че трябва да приложим алгоритъма на Манахер върху път между всяка двойка листа. Ще започнем нашето решение за цялата задача със следното наблюдение: Ако съществува палиндромен сегмент с дължина $K>2$, тогава съществува палиндромен сегмент с дължина $K-2$. Следователно можем да използваме двоично търсене на четни и нечетни дължини, за да намерим най-дългия палиндромен сегмент. Как да проверим дали в дадено дърво съществува палиндромен сегмент с фиксирана дължина? За да отговорим на този въпрос, първо ще решим по-лесна версия на тази задача – ще проверим дали има палиндромен сегмент с фиксирана дължина, който съдържа корена на нашето дърво, връх R.

В кореново дърво, пътят между два върха A и B се състои от два клона на дървото, където дължината на пътя на единия клон е по-голяма или равна на дължината на другия клон. Ще разделим пътя между A и B на три части: път от A до C, път от C до корена и път от корена до C, където C е избрано така, че дължините $A - C$ и $R - B$ са равни. Обърнете внимание, че C е еднозначно дефиниран от върха A, тъй като ни интересуват само палиндроми с фиксиран размер. За да проверим дали пътят от A до B е палиндромен сегмент, е достатъчно да проверим следното:



(1) Пътят от R до C е палиндром (изобразен в червено)
(2) Цветовете от C до A са същите като цветовете от R до B (изобразени в синьо).

За всеки връх от дървото ще изчислим предварително стойностите $\mathtt{down}_X$ и $\mathtt{up}_X$ по същия начин, както беше описано в решението на първата подзадача. Първата проверка се извършва просто чрез сравняване на $\mathtt{down}_C$ и $\mathtt{up}_C$. Втората проверка може да се извърши чрез сравняване на следните стойности:

- $\mathtt{down}_\mathtt{B}$
- (3) $\mathtt{down}_\mathtt{A} - \mathtt{down}_{\mathtt{par(C)}} * \mathtt{B}^{\mathtt{dep(A)-dep(C)}}$, където par(C) е родителят на върха C и dep(X) е дълбочината на върха X.

Сега знаем как ефективно да проверим дали пътят от A до B е палиндромна отсечка, но това все още не е достатъчно, защото има $O(N^2)$ такива двойки (A, B).
Нека се опитаме да разгледаме задачата от друг ъгъл:
Ако фиксираме A, има ли (поне) един връх B, който удовлетворява всички условия?. Нека S_B е множество от хешове, в които ще вмъкваме стойности надолу по B на всички върхове B на нашето дърво. Тогава, за фиксиран връх A, можем просто да проверим условие (1) и можем да проверим условие (3), като намерим съответната стойност в S_B. Можем да предположим, че сложността на всички операции със S_B е константна, ако използваме хеш таблица (например std::unordered_set в C++). Следователно сложността на цялата проверка е O(n).

Забележка: по време на имплементацията трябва да се уверите, че най-ниският общ прародител на върховете в S_B и A е коренният връх R.

След като знаем дали съществува палиндромна отсечка с определена дължина, която преминава през корена R, можем да използваме центроидно разлагане. Ако извършим тази проверка за всяко разложено поддърво с центроид като корен, ще покрием всички случаи. С центроидно разлагане и двоично търсене, получаваме алгоритъм със сложност $O(n \log^2 n)$.

- Алгоритъм на Манахер - https://en.wikipedia.org/wiki/Longest_palindromic_substring