АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА ПРЕНОМЕРИРАНЕ

Най-напред да изясним защо такова преномериране е единствено.
Ясно е, че най-малки номера ще имат „листата“ на графа, т.е. тези върхове, от които не излизат ребра. Наистина техните наследствени редици са празни, т.е. възможно най-малки. Това означава, че листата ще заемат новите номера от 1 до техния брой, като в рамките на този диапазон новият номер на всяко от тях е едднозначно определен от големината на стария му номер (да напомним, че, при равни наследствени редици на два върха, по-голям нов номер получава този с по-голям стар номер).
Следващото, което трябва да съобразим е, че новият номер на всеки връх трябва да бъде по-голям от новите номера на неговите наследници. Да разгледаме върховете от второ ниво, т.е. тези, които имат за наследници само листа. Ясно е, че те ще имат по-големи нови номера от листата, тъй като техните наследствени редици не са празни. Нека ***u*** е нов номер на такъв връх, а ***v*** е нов номер на негов предшественик. Да допуснем, че $v<u$. Това означава, че наследствената редица на v е по-малка от наследствената редица на u. Но u влиза в наследствената редица на v, а наследствената редица на u е от нови номера на листа, които, както вече видяхме, са по-малки от u. Стигнахме до противоречие, което доказва, че върховете от второ ниво имат нови номера в диапазона, непосредствено следващ диапазона с номерата на листата. В рамките на този диапазон, номерът на всеки връх от второ ниво се определя от стария му номер, т.е. еднозначно. Продължавайки по нива ще се убедим, че новите номера са еднозначно определени.

Горните разсъждения ни навеждат на следния общ алгоритъм за преномериране на върховете: нека вече сме дали нови номера $1,2,....,к$ на k на брой върха. Разглеждаме множеството на всички върхове, които са готови за преномериране, т.е. всичките им наследници вече са получили нови номера, но те самите още нямат нов номер. Подреждаме това множество по ненамаляващ ред на наследствените им редици (ако два върха имат еднакви множества от наследници, т.е. еднакви наследствени редици, по-напред застава върхът с по-малък наследствен номер) и на първия връх от това подредено множество даваме следващия номер $k+1$.

Тази идея лежи в основата на всички решения, дадени по-долу.

Във всички решения се предполага, че графът се представя със списъци на съседство. По-удобно е за всеки връх да държим не списък на наследниците му, а списък на предшествениците му. Освен това за всеки връх ще поддържаме по един елемент в масив, който ще съдържа броя на наследниците му, които все още не са получили нови номера (първоначално това е общия брой на нследниците му)

Наивна реализация на идеята

Наивната реализация предполага да държим по една наследствена редица за всеки връх и, когато някой връх получи нов номер, да обновяваме наследствените редици на всички негови предшественици. След това да сортираме наследствените редици на предшествениците му, които вече нямат неномерирани наследници, във ненамаляваш ред и да даваме последователни номера на тези предшественици като

спазваме правилото, че при равни редици по-малък номер получава този, който има по-малък стар номер. Този начин води до решения със сложност $O(n^3)$ или $O(n^4)$ (зависи от реализацията), където n е броя на върховете. Такова решение би трябвало да получи 20 точки.

Решение със сложност $O(n^2)$ (n – брой на върховете)

Следващата стъпка е да съобразим, че няма нужда да пазим и сравняваме наследствени редици. Нека имаме опашка Q, в която ще стоят върховете, готови за преномериране в реда, в който трябва да получават новите си номера. Първоначално в Q вкарваме всички листа, подредени в нарастващ ред на старите си номера.

Ще поддържаме и списъл L, в който ще държим върховете, които имат поне един преномериран наследник, но все още имат и непреномерирани наследници. От този списък върховете ще отиват в опашката Q, когато и последният им наследник получи нов номер. Идеята е в L върховете да стоят в нарастващ ред на частично изградените си наследствени редици (т.е. частичната наследствена редица на връх от L съдържа ония негови наследници, които вече са получили номера).

Да видим какво се получава, когато върхът q от началото на опашката Q получи нов номер *newq* (това е поредно число; всички номерирани преди него имат по-малки нови номера, а всички, които ще бъдат номерирани след него ще имат по-големи нови номера). Нека $B(q)$ е множеството от предшественици на q. За всички елементи от $B(q)$ **и само за тях** наследствените редици ще се променят, като *newq* застане в началото на всяка от тях (в тях до момента са стояли по-малки нови номера на други вече преномерирани наследници или те са били празни). Някои от елементите на $B(q)$ вече се намират в L (влезли са там покрай преномерирането на други свои наследници), а други ще влязат за първи път. Във всеки случай налице са следните зависимости:

- Частичните наследствени редици на елементите от $B(q)$ ще бъдат по-големи от частичните наследствени редици на елементите от L, които не са членове на $B(q)$ (защото в началото им стои *newq*, което е най-големия нов номер получаван до момента);
- Частичните наследствени редици на елементите от $B(q)$, които за първи път влизат в L ще бъдат по-малки от частичните наследствени редици на елементите от $B(q)$, които вече са били в L, тъй като редиците на върховете, които за първи път влизат в L ще съдържат само *newq*, а тези, които вече са били в списъка, имат и други елементи.
- Подредбата един спрямо друг(според частичните наследствени редици) на елементите от $B(q)$, които вече са били в L, няма да се промени, защото отпред на всички техни редици се добавя *newq*.
- Подредбата един спрямо друг на елементите от L, които не участват в $B(q)$ няма да се промени, тъй като не се променят наследствените им редици.

Тези зависимости показват, че, за да запазим желаната подредба на върховете в L (по ненамаляващ ред на частичните наследствени редици), след преномерирането на поредния връх от опашката Q, трябва да пренаредим списъка L по следния начин: най-напред, в същия ред един спрямо друг,в който са били, застават елементите от L, които не влизат в $B(q)$ (които не са предшественици на q); след това (независимо в какъв ред)

елементите от *B(q)*, които влизат в *L* за първи път; накрая, в същия ред един спрямо друг, в който са били, застават елементите от *B(q)*, които вече са били в *L*.

За всички елементи от *B(q)*, се намалява с 1 броят на непреномерираните им наследници.

След такова пренареждане се преглежда от началото към края списъка *L* (всъщност трябва да се прегледа само оная част от него, която съдържа върховете от *B(q)*) и ония негови елементи, които вече нямат непреномерирани наследници в реда на преглеждането се вадят от *L* и се поместват в *Q*.

След това се вади следващият елемент от началото на *Q* и т.н.

За да се осигури изпълнението на изискването при еднакви подмножества от наследници по-малък номер да има върхът с по-малък стар номер е достатъчно предшествениците на даден връх да се обработват в нарастващ ред на старите им номера.

Преглеждането и пренареждането на списъка *L* може да става по различни начини. Ако се използват само масиви, то това ще доведе до преписване на част от елементите на L в новия списък и т.н. допълнителни технически действия. Друг важен момент е преглеждането на вече пренаредения списък *L*.

Реализацията на тези дейности води до алгоритъм $O(n^2)$. Могат да се правят различни оптимизации, най-съществената от които е използването на свързани списъци, за да се избегне преписването на стойностите на списъка *L* от един масив в друг. Такива решения ще вземат около 50-70 точки, зависи от реализацията.

Решение със сложност *O(mlogm) (m-брой на ребрата)*

Това е решението за 100 точки. То развива решението със сложност $O(n^2)$, като в основата му стои отново поддържането на подредения списък *L*, но се избягва преписването и преглеждането на този списък на всяка итерация. Наистина, логично е да се предположи, че на всяка итерация е достатъчно да се извършва брой операции, пропорционален на броя елементи в *B(q)*, а не на броя елементи в *L*, който може да е от порядъка на броя на всички върхове в графа, т.е. *n.* За целта към всеки елемент в L се добавя поле, което съдържа порядков номер на елемента в списъка. Тези порядкови номера не са непременно последователни числа, но са такива, че нарастват по реда на елементите в списъка. Списъкът *L* се реализира чрез двустранно свързан списък. За всеки връх на графа, който е влязъл в *L*, се пази указател към елемента му в *L*. При преномериране на върха *q* от началото на опашката се прави масив с елементите от *B(q)* (най-добре е да се работи с вектор), които вече са били в *L*, като същевременно те се премахват от списъка *L* (не повече от *O(B(q))* операции). След като списъкът се „прочисти“ от елементите от *B(q)*, към него се добавят ония елементи от *B(q)*, които за първи път ще влязат в него, като им се дават последователни порядкови номера, по-големи от порядковите номера на елементите, останали в *L* (за порядковите номера е достатъчно да поддържаме една променлива, чиято стойност се увеличава с 1 всеки път, когато трябва да дадем порядков номер на някой елемент, който поместваме накрая на *L*). Тук е важно да се отбележи, че към *L* добавяме само онези върхове от *B(q)*, които имат все още непреномерирани наследници – тези, за които *q*_е бил

последен непреномериран наследник, отиват директно в опашката. Тук също имаме не повече от $O(B(q))$ операции.

Накрая сортираме по порядковите номера масива, в който бяхме извадили елементите от $B(q)$, които вече стояха в L, и в този ред ги добавяме в края на L, давайки им нови порядкови номера. Тук, заедно със сортирането, имаме не повече от $O(B(q)log(B(q))$ операции. Така цялото решение е със сложност $O(mlogm)$, където m е броят на ребрата, тъй като $\sum_q B(q)=m$, когато q пробягва всички върхове на графа.

Автор: Руско Шиков