**Анализ**

Условието задава следната постановка. Върховете на графа са служителите, а ребрата са отношенията на началник и подчинен. Този граф е ориентиран и без цикли. Търсим броя топологични сортировки. В общия случай тази задача е нерешима. Тук се намесва допълнителното условие за графа, което ще обясним по-късно какво точно ни помага.

Първата подзадача е за 15 точки. Очакваното решение е да се генерират всички възможни пермутации на служителите и да се провери спрямо зададените ребра. Така сложността е $O\left(N!M\right)$.

Понеже втората подзадача е дадена за хората, които са решили почти цялата задача, то първо ще говорим за решението на третата подзадача, която е за 20 точки. Важното условие тук е, че отговорът, който търсим не е много голям. Тук идеята е да генерираме всички възможни топологични сортировки без съответно да гледаме всички възможни пермутации. Това може да стане с модификация на алгоритъма за топологично сортиране. Ще използваме алгоритъма, в който в опашка пазим кои върхове са станали с 0 влизащи ребра. Идеята е да го реализираме рекурсивно, за да може на всяка стъпка да пробваме всеки възможен вариант за текущ връх. При този вариант, като сме пробвали даден връх трябва да махнем върховете, които е освободил при избирането му, за да може да пробваме с друг евентуален връх от опашката. Съответно това обхождане и операциите, които искаме да извършваме променят структурата от данни, в която записваме на дек (двойна опашка). Използвайки тази структура и извършвайки рекурсивно алгоритъма обхождаме точно всички възможни топологични сортировки $O\left(Ans(M+N)\right)$.

Втората подзадача е за 35 точки, а четвъртата подзадача е за 50 точки. Сега трябва да анализираме какво дава условието. Нека използвайки алгоритъм за топологично сортиране намерим ориентирано дърво. Съответно част от началните ребра участват в него, а друга част свързват някои върхове с такива, които се намират по-долу в дървото (нека ги наречем странични ребра). Няма как да имаме ребро в обратна посока (от връх в дървото до връх нагоре), защото бихме имали цикъл, а графът е ацикличен. Условието за графа ни гарантира следното: странично ребро, което излиза от връх не напуска поддървото на този връх. Сега ще го докажем – да допуснем противното. Нека реброто е от *u* до *v*. Допускането ни дава, че за върха *v*, в който отива реброто, ще има двама началници (*u* и върхът, който е баща на *v*) между които обаче няма наредба така че единия от тях да бъде началник на другия, защото *v* не е в поддървото на *u* и съответно баща му също няма да е, т.е. тези два върха са в паралелни поддървета (бащата на *v* със сигурност е на същата или по-ниска височина от *u*). Така доказахме, че страничните ребра свързват връх с някой връх от поддървото му. Но понеже тези връзки задават транзитивна релация, то тези ребра са ненужни при гледането на релацията. Съответно можем да ги премахнем и да търсим брой топологични сортировки в построеното дърво. Това вече е по-лесна задача. Ще използваме динамично по дървото. Нека знаем, отговора за преките наследници *v1, v2, …, vk* на връх *u*. Това означава, че във всяка топологична наредба *u* трябва да е преди върховете в поддървото му, като освен това върховете в поддърветата *v1, v2, …, vk* са независими помежду си. Можем да намерим бройката на наредбите като пермутации с повторение на върховете в поддърветата *v1, v2, …, vk* (съответно върхове от едно и също поддърво ги броим за неразличими) и умножим тази бройка по броя начини за наредба на върховете в поддървото на *v1* по броя начини за наредба на върховете в поддървото на *v2* и … по броя начини за наредба на върховете в поддървото на *vk*. По този начин с линейна сложност можем да намерим отговора. Разбира се, всички операции извършваме по модула в условието. Смятането на пермутации с повторение изисква смятането на частно на факториели. За това ни трябва предварително изчисляване колко е остатъка на всеки факториел, както и на всеки 1/факториел. Понеже модулът е просто число (и данните са по-малки от модула), е възможно да сметнем еднозначно остатъка на частно като следствие от малката теорема на Ферма: $\frac{1}{a}≡a^{p-2} (mod p)$. Затова ни трябва бързо степенуване. Съответно 1/факториел смятаме с натрупване. Втората подзадача е дадена за хората, които не знаят последния факт, но си решат алгоритмично задачата. Окончателната сложност по време е $O\left(Nlog\_{2}p+M\right)$.

Задачата е частен случай на известна нерешима задача. Много често такива частни случаи стават интересни задачи. Алтернативно на пермутациите с повторение може да се мисли за последователност от комбинации.

*Автор: Илиян Йорданов*