АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА
ПОКОКИ

Първото и основно важно наблюдение е, че графът (който представлява пътната мрежа на Поконония) е ацикличен ориентиран и претеглен граф. Също така, директно от условието следва, че от връх 1 има път до всеки друг връх, от всеки връх има път до връх N и от връх N няма път (и съответно излизащи ребра) до нито един друг връх.

Решението за 30 точки е пълно изчерпване – което ще рече, да пробваме последователно „носейки“ 1, 2, 3, ... „пококита“ в себе си дали е възможно по поне K различни начина да стигнем от 1 до N. Като проверката дали това е възможно става с обхождане в дълбочина, но без да пазим списък от посетени върхове, а просто да посещаваме всички съседи на текущо разглеждания връх, за които можем да преминем по директния път (т.е. носим достатъчен брой „пококита“ в себе си), без значение дали те са били посетени преди това или не и всеки път когато „пристигнем“ във върха N, да увеличаваме един брояч. Накрая, след като този алгоритъм завърши, този брояч ни показва броят на различните пътища от 1 до N.

Решението за 50 точки стъпва отново на това да пробваме последователно да носим 1, 2, 3 и т.н. „пококита“ в себе си, но преброяването на пътищата от 1 до N е различно, а именно: Обхождаме графа в неговата топологична сортировка. Очевидно започваме с върха 1, за когото няма влизащи ребра. Начините по които можем да стигнем до този връх са 1, нека това да се смята и пази в масива count[], а конкретно за върха 1 това е стойността count[1]. Нека текущо разглеждания връх означим с c. Разглеждаме всички ребра на текущо посетения връх, които са с тежест по-малка или равна на текущо носените „пококита“. За вяско такова ребро, нека „входящия“ връй (т.е. там, накъдето сочи реброто) е върха v. Тогава можем да кажем, че имаме поне count[c] начина да стигнем до върха v (стигайти първо до c по count[c] начина и използвайки това конкретно ребро). Тогава можем да прибавим count[c] към текущо намерената стойност на count[v]. Важно е да обходим графа в неговата топологична сортировка, защото така ще си гарантираме, че започвайки да разглеждаме някой връх, то със сигурност вече ще сме разгледали всички върхове, които евентуално могат да стигнат до него (т.е. стойността count за този връх вече със сигурност ще е изчислена). За целта е най-удобно да обхождаме графа в ширина и да добавяме в опашката от посетени върхове даден връх в момента, в който сме разгледали и последния връх, който има ребро сочещо към върха, който искаме да добавим в опашката. Този алгоритъм е със стойност O((N + M) * answ), където answ може да бъде 10^9, което е прекалено много.

Решението за 100 точки е същото като това за 50, но изборът за „проба“ на това колко „пококита“ да носим в себе си се прави на базата на двоичното търсене. Очевидно минималния възможен отговор на задачата би бил 1, а максималният би бил 10^9. На всяка стъпка пробваме „средната“ стойност между най-малката и най-голямата възможна и пробваме да видим възможно ли е носейки толкова „пококи“ можем да стигнем по поне K различни начина от 1 до N. Ако това е възможно, тогава е възможно да съществува още по-малък брой „пококи“ за които това е валидно и следващата стъпка ще разглежда „лявата“ половина на текущо разглеждания интервал.

Ако не е възможно, значи със сигурност ни трябва повече „пококи“ и заради това следващата стъпка трябва да разглежда изцяло дясната половина на текущо разглеждания интервал. И така с двоично търсене ще намерим най-точно колко минимално „пококи“ ни е нужно за целта на нашата задача. Сложността на този алгоритъм е O((N + M) * log MAX), където MAX е максималната стойност на ребро в графа. Нищо не пречи MAX да е константата от условието на задачата, а именно 10^9.

Автор: Момчил Иванов