**Анализ**

*Тагове: дървета, DFS, динамично в дърво, две обхождания*

В задачата се описва претеглено дърво. Разглеждаме простите пътища между всеки два върха. Трябва за всеки връх да намерим оценка, която е равна на сумата от пътищата, минаващи през него.

Решение за 23 точки.

Тя е предвидена за наивното решение. Най-удобно е да пуснем по един *DFS* от всеки връх, с който намираме всички пътища, започващи от него. Съответно можем да поддържаме текущия път във вектор. Така за всеки път, лесно минаваме през върховете, които участват в него, и добавяме дължината на пътя към оценката на върховете. Този подход ще прегледа всеки път два пъти – от двата му края. Затова като приключим с обхожданията, за реалните оценки трябва да разделим намерените числа на 2. Сложността на алгоритъма се определя от сумарната големина на пътищата. В голяма част от тестовете, дървото е близо до пръчка, където се достига най-лошия случай на това решение. Сложността е $O(N^{3})$.

Решение за 58 точки.

Тук ограниченията са за квадратно решение. Има различни начини. Един начин е по-бавна версия на пълното решение. Сега ще опишем по-простия за писане начин, който е оптимизирана версия на предното решение. Лесно се вижда, че мястото, което е подходящо за оптимизиране е там където за всеки път, гледаме отново върховете през него, за да добавим към съответните оценки. Това може да се прави и на обратния ход на рекурсията (*DFS*). За тази цел е достатъчно за всеки връх да сметнем сумата на пътища, които ще намерим в следствие на обхождането през него. Така, като се върнем от някой връх *y*, на връх *x*, просто добавяме към оценката на *x* сумата от намерените пътищата през *y*. Това съответства на процеса по обхождането на върховете за всеки намерен път, което правихме в предния абзац. Сложността вече е $O(N^{2})$.

Решение за 100 точки.

Трябва да сменим малко идеята за решение, защото предните начини намираха всички пътища, които е ясно, че са около ***N***2, така че няма по-добро от предното решение в тази посока. Нека сега се опитаме да намерим оценката за всеки връх. Първо ще опишем квадратичната идея. Нека пуснем *DFS* от връх *r* – съответно го правим корен на дървото и разгледаме някой път, който минава през *r* и *r* е междинен (не му е край). Този път можем да разгледаме като сума на два пътя, които започват от *r* и се спускат надолу. Всеки от тези два пътя е съставен от път, започващ от съсед на връх *r* и продължаващ надолу, на който е добавено едно ребро към *r*. Именно тази връзка ще използваме за намирането на пътищата през *r*. Те се делят на две групи, за едните *r* e междинен, а за другите е край. От описанието досега стигаме до извода, че ще ни е полезно за всеки връх да сметнем сумата от пътищата, които започват от него и продължават надолу в дървото. Тази стойност за връх *x* ще означим с *paths[x]* и ще я смятаме ефективно с динамично в дърво. Ако сме я сметнали за децата на *y*, то трябва да прибавим всички пътища, започващи от децата към *paths[x]*, но с добавено реброто, което свързва съответното дете с *y*. Затова освен *paths[x]*, за всеки връх ще ни трябва и *cnt[x]*, което е броят на пътищата (това всъщност съответства на размера на поддървото). Така за да получим сумата от дължината на пътищата от *x* през някое дете *y*, трябва да сметнем *paths[y] + cnt[y]\*l*, където *l* е теглото на реброто от *x* към *y*. Ако за всеки връх броим пътя, започващ и завършващ в него, то по описания начин директно смятаме търсената стойност *paths[x]*.

Нека сега се върнем на връх *r*, от който започнахме. Ясно е, че в оценката му влиза *paths[r]*, което смятаме без проблеми – това са вторият вид пътища. За другите трябва да комбинираме два пътя, които започват от различни деца и продължават надолу с ребрата, които свързват децата с *r*. Нека обходим децата на върха. За да комбинираме пътищата и да намерим сумата, използваме подобни разсъждения като преди малко. Ако сме на дете *x*, то всеки път, започващ от него, се комбинира със останалите, които са *cnt[r]-cnt[x]* на брой и със сумарна големина *paths[r] – (*пътищата от *r* през *x* надолу*)*. Затова в сметката всеки път от *r* през *x* ще участва *cnt[r]-cnt[x]* на брой пъти. Аналогично, всеки от останалите пътища ще участва *cnt[x]* пъти. Сумираме за всяко дете тези сметки и намираме търсената сума. Единствения проблем на тази сума е, че всеки път, в който *r* е междинен, е повторен два пъти, а тези, които започват от *r* са само веднъж. Но това лесно се оправя. Така ако пуснем описания алгоритъм от всеки връх, имаме квадратно решение, което е ценно, защото намира оценката за всеки връх директно.

Стандартен начин за оптимизиране на такова решение е да изберем произволен връх за корен и да направим две обхождания на дървото – отдолу-нагоре, с което намираме описаните стойности в предния абзац и второ, отгоре-надолу, с което намираме крайните отговори за всеки връх. Лесно за всеки връх *x* можем да сметнем пътищата, които минават през него и остават в неговото поддърво, по същия начин, като за *r*. Допълнително за *x* трябва да сметнем пътищата, които минават през него и продължават нагоре. Това са пътища, които започват от връх в поддървото на *x* и продължават към връх нагоре от *x*. За тази цел като правим второ обхождане, пазим в параметри на рекурсията сумата на пътищата, които започват от текущия връх *x* и продължават нагоре от него, както и техния брой. Смятането на описаните пътища през *x* става абсолютно аналогично на предния абзац и дори по-лесно от комбинирането на пътища в поддървото. Поддържането на стойностите на допълнителните параметри също е лесно и аналогично на предните разсъждения. Описаният алгоритъм, брои пътищата само веднъж за всеки връх и няма нужда от допълнителни изчисления. Крайната сложност е $O(N)$.

Задачата е вдъхновена от задача *points* на общинския кръг, съответно тук точките са върховете на дървото, а разстоянието между върхове в дървото е съответния път. За да е по-интересно трябва да намерим за всеки връх сумата от пътищата, минаващи през него.

*Автор: Илиян Йорданов*