**АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА**

**СЛУЖЕБНА ЙЕРАРХИЯ**

Макар и доста „учебникарска“ задачата е доста поучителна. Първото решение, за което се очаква учениците от група B да се сетят, е рекурсивното решение, основаващо се на следната проста идея (проста за тези състезатели, които умеят да мислят рекурсивно) – ако за даден връх знаем броя на подчинените на всеки от преките му подчинени (синовете му в дървото на служебната йерархия), то броят на неговите подчинени ще бъде равен на сумата от броя на подчинените на всеки един от синовете му(преки подчинени) плюс броя на самите синове (самите преки подчинени). Разбира се листата в дървото (простосмъртните служители) имат по 0 подчинени, което определя „дъното“, до което потъва рекурсията. Това решение е реализирано във файл **subordR.cpp.** и крие един подводен камък, който много често се среща, когато се използва рекурсия. Преди да говорим за него, нека обърнем внимание на друга опасност, която дебне състезателя, който прави рекурсивно решение на задачата. При недостатъчно добро обмисляне, има опасност за всеки въпрос за броя подчинени на даден връх да се вика рекурсивната процедура, която ще пресмята броя подчинени на този връх, започвайки винаги от него. По този начин, ако имаме два въпроса, напр. за връх V и негов предшественик връх U, ние няма да използваме пресметнатия брой подчинени на връх V при пресмятането на броя подчинени за връх U и това значително ще забави нашето решение при голям брой, подходящо формулирани въпроси. Такова решение е реализирано във файл **subordRS.cpp** и ще получи 30 точки. Правилно е решението във файл **subordR.cpp**, в което, още преди да се четат въпросите за това кой колко подчинени има, програмата извиква рекурсивната процедура с параметър корена на дървото, която минава линейно по дървото и в един масив записва броя подчинени на всеки връх. След това, четейки въпросите, програмата просто извежда съдържанието на съответния елемент от този масив. Това решение ще получи 60 точки, защото тестовете са подбрани така, че в 40% от тях се проявява подводният камък на рекурсията, който се състои в следното: когато рекурсията (викането на рекурсивната процедура от самата себе си) „потъва“ на много нива (в случая малко над 40 000 пъти) има опасност да се препълни системният стек и програмата да „гръмне“. Настоятелно препоръчвам, за повече сведения по този въпрос, да прочетете лекцията на Александър Георгиев „Рекурсия и търсене с връщане“ в неговия сайт informatika.bg. За да се избегне тази грешка, трябва да се измисли друг алгоритъм, в който рекурсивното решение да се замени с нещо друго. Оказва се, че съществува доста елегантно решение, което не използва рекурсия и неговата идея е следната: представете си, че сме успели да подредим върховете на дървото на служебната йерархия в линеен списък (в масив или вектор – от гледна точка на реализацията най-удобно е във вектор), така че в този списък всеки „баща“ да се намира по-напред от „синовете“ си (всеки началник да се намира преди преките си подчинени). В решението **subord.cpp**, в което е реализирана тази идея, това е вектор *NLevel*, който съдържа номерата на върховете в дървото, подредени именно в такъв ред*.*Ще поддържаме два допълнителни масива – *father* (*father[i]* ще съдържа номера на върха, който е „баща“ на връх с номер *i* – можем лесно да го попълваме още като четем входните данни) и *subordinates* (*subordinates[i]* ще съдържа броя на подчинените на връх с номер *i –* първоначолно нулираме всички елементи на този масив). След като сме получили такъв списък, минаваме линейно през него отзад напред и, когато се намираме в елемент с номер *j*, към броя подчинени на бащата на връх с номер *NLevel[j]*(т.е. към *subordinates[father[NLevel[j]]*)добавяме броя подчинени на връх с номер *NLevel[j]* плюс 1 (т.е. имаме оператор *subordinates[fathers[NLevel[i]]]+=subordinates[NLevel[i]]+1;*). Тъй като листата в дървото имат по 0 подчинени, то при минаването през целия списък ще получим броя подчинени на всеки връх от дървото (поради начина на подредба на върховете в списъка). Как да получим вектора NLevel. Това е реализирано във функция *fillNLevel,*която се вика с параметър номера на върха-корен на дървото. Първоначално във вектора поместваме корена на дървото. След това се движим по вектора от началото към края и, когато се намираме в поредния му елемент, който съдържа номер на връх от дървото, в края на вектора добавяме номерата на синовете му. Получава се обхождане на дървото по нивата на йерархията в него. Такова решение не използва рекурсия, линейно е по *N* (сложност *O(N)*) и избягва опасността от препълване на системния стек. То ще получи 100 т.

Тази задача изисква учениците да могат да представят дърво чрез списъци на синовете (което е следствие от по-широкото познание за представяне на граф чрез списъци на съседство), както и да определят номера на върха – корен на дървото като единствен връх, който няма „баща“.

*Автор: Руско Шиков*