# Анализ на задача monopoly2

*Тагове: интерактивна задача, дърво, двоично търсене*

Понятието *подходяща* наредба в условието е просто понятието за топологично сортиране. Така задачата иска да се познае неизвестно кореново дърво (с корен връх $1$), като знаем, че $1, 2, \ldots, N$ е топологична наредба на дървото и питаме за различни наредби на върховете. В началния вариант `check` функцията връщаше просто отговор да/не, но това не предоставяше много информация и затова нямаше особено разнообразни идеи. Мотивацията за сегашния вариант на отговора на `check` се получи точно както е описано в условието - опит да се оцени доколко е топологична дадена наредба, т.е. да се върне повече информация.

## *Решение на първа подзадача - 5 точки*

Може да се каже, че тази подзадача е за обратна връзка от състезателите. Допълнителното ограничение в тази подзадача всъщност задължава дървото да е дърво-пръчка. Така можем без въпроси директно да върнем $(1, 2), (2, 3), \ldots, (N-1, N)$.

Брой въпроси: $0$.
Сложност: $O(1)$.

## *Решение на втора подзадача - 6 точки*

Понеже $N$ е много малко, то можем да приложим решение с пълно изчерпване. Една възможност е да питаме всеки възможен въпрос и след това да пробваме кое от възможните дървета с $N$ върха отговаря на получените отговори. Тази подзадача има много тестове, защото са сложени всички възможни дървета с $6$ върха, които са колкото $5$-тото число на Каталан - $42$. По този начин подзадачата валидира до колко са верни последващите решения.

Брой въпроси: $N!$.
Сложност: $O(N! \times N! \times N)$.

## *Решение на трета подзадача - 22 точки (=5+17)*

Това е първата по-истинска подзадача, в която има и частично оценяване. Опитните състезатели бързо трябва да са забелязали, че $15\,000$ е близко до $N \log_2 N$ въпроса, което е очакваният брой въпроси за пълно решение на тази подзадача. Допълнителното ограничение всъщност казва, че наредбата $1, 2, \ldots, N$ е и наредба на върховете получена след *BFS*. Така имаме доста повече информация за тази наредба - върховете са наредени по нива, децата на един връх са наредени едно до друго.

Решението на тази подзадача съдържа и идеи, които се използват и по-нататък. Нека да откриваме интервалите от деца на върховете последователно. Ще открием децата на даден връх $i$ с двоично търсене. За тази цел, ако знаем, че започват от позиция $from$, то трябва да можем да отговорим на въпроса дали $from, from+1, \ldots, mid$ са деца на връх $i$ (имаме монотонност, ако разглеждаме върховете $from, from+1, \ldots, N$, защото до някакъв момент те са деца на връх $i$ и от там нататък вече не са деца). Оказва се, че има и един по-труден случай, в който например връх $mid$ не е дете на $i$, но е негов непряк наследник. Затова най-добрата стратегия е следната. Питаме въпрос за наредбата $1, 2, \ldots, N$, в която изместваме връх $i$ и потенциалните му деца в началото на нивото на връх $i$ (спрямо *BFS* наредбата), а децата записваме в реда $mid, from, from+1, \ldots, mid-1$. Останалите върхове си остават по същия начин. След това гледаме дали отговорът на `check` има само $0$, т.е. дали е топологична наредба

или не. Използвайки тази стратегия винаги ще разберем, ако някой от потенциалните деца е дете на друг връх от нивото на $i$ или пък $mid$ е непряк наследник на връх $i$.

Брой въпроси: $N \log_2 N$.
Сложност: $O(N \log_2 N \times N)$.

### *Решение на четвърта подзадача - около 25 точки*

Подзадачата е за наивно решение, в което използваме по най-много $N$ въпроса за да намерим всяко ребро. Ще открием бащата на всеки връх в дървото. Знаем, че за всеки връх бащата е с по-малък номер. Така ако започнем да местим даден фиксиран връх назад в наредбата и питаме въпроси, то в началото отговорите ще указват, че наредбата е останала топологична. Интересен е първият момент, когато това не е така. Лесно може да се види, че тогава върхът, преди който сме сложили фиксирания, е точно баща му.

Брой въпроси (в най-лошия случай при дърво-звезда): $\frac{(N-1)N}{2}$.
Сложност: $O(N^3)$.

### *Частично решение на пета подзадача - около 54 точки*

Всъщност това решение е директно подобрение на предното. Опитните състезатели веднага са забелязали монотонността, която имаме при предния алгоритъм - до някакъв момент отговорите на `check` съдържат само $0$ (защото наредбата е топологична), а след това вече се появяват и $1$, които бележат, че се е нарушила топологичността на наредбата. Затова можем да направим двоично търсене по този момент за всеки връх.

Брой въпроси: $N \log_2 N$.
Сложност: $O(N \log_2 N \times N)$.

### *Частично решение на пета подзадача - около 70 точки*

Това е първото решение, което ще се възползва от по-голямата информация, която дава отговорът на въпроса. Лесно можем да използваме въпросите, за да получим информация за достижимостта - да разберем за всеки връх кои върхове са му наследници (преки и непреки). За да открием тази информация за връх $i$, то е достатъчно да питаме въпроса $1, 2, \ldots, i-1, i+1, \ldots, N, i$. Тогава по определението на отговора знаем, че за децата на връх $i$ ще получим $1$, а това означава, че и за всички техни наследници ще имаме $1$. На останалите върхове ще съответства $0$, защото нищо не е нарушено от гледна точка топологичност. Като се възползваме и от наблюдението, че на всеки връх бащата има по-малък номер, то можем лесно да открием бащата на всеки връх $j$ - това е върхът с най-голям номер $i$, за който отговорът на въпроса е указал, че $j$ е наследник на $i$.

Така с $N-1$ въпроса лесно можем да открием бащата на всеки връх или съответно ребрата на търсеното дърво. За жалост, въпреки монотонността за намирането на бащата на даден връх няма смисъл да използваме двоично търсене, защото то не помага да оптимизираме броя въпроси.

Брой въпроси: $N-1$.
Сложност: $O(N^2)$.

*Решение на пета подзадача - 100 точки*

Решението с двоично търсене не е много добро, защото не се възползва добре от информацията, която се получава с всеки отговор, а по-скоро третира въпроса като булев относно това дали наредбата е топологична. Една добра идея е да комбинираме и да правим едновременно няколко двоични търсения за намиране на бащите на върхове. Това е възможно само тогава когато върховете нямат връзка по между си, т.е. не са наследници един на друг. Проблемът е, че без да сме научили нещо за структурата на дървото е трудно да приложим тази идея. Можем да пробваме да приложим разделяй и владей подход за намиране на дървото, но при "владей" частта имаме проблеми да намираме информация едновременно за всички върхове на едно рекурсивно ниво (иначе няма да подобрим броя въпроси спрямо предното решение).

Можем да забележим, че има лесен начин да намерим "независими" върхове в дадена част. Нека разгледаме някакъв суфикс $i, i+1, \dots, N$. Ако разглеждаме подграфа ограничен само до тези върхове, то той е гора от потенциално няколко дървета. Техните корени са всъщност "независимите" върхове, а техните бащи са сред върховете $1, 2, \dots, i-1$. Корените можем да намерим като обърнем суфикса и питаме следния въпрос: $1, 2, \dots, i-1, N, N-1, \dots, i$. Отговорът, който ще получим ще даде $0$ за $1, 2, \dots i-1$ (там не нарушаваме топологичността) и за всички корени сред останалите върхове, защото техните бащи вече са срещнати. Понеже обръщаме наредбата, то си гарантираме, че за върховете, които не са корени сред суфикса, ще се наруши топологичността и за тях ще получим $1$.

Вече сме готови за пълното решение. Ще се възползваме от намирането на независими върхове като намираме върховете ниво, по ниво (спрямо разстоянието им от корена $1$). Така, за да намерим ниво $1$ (децата на връх $1$), ще питаме въпроса $1, N, N-1, \dots, 2$ и ще видим кои върхове са получили $0$ (освен връх $1$). След това ще питаме въпроса $1, v_1, v_2, \dots, v_k, N, N-1, \dots$, където $v_1, v_2, \dots, v_k$ са намерените върхове от първо ниво, за да намерим върховете от второ ниво и т.н. Ако означим височината на дървото с $h$ (равна и на броя ребра в най-дългия път), ще ни трябват $h-1$ въпроса, за да научим по нива върховете (за ниво $0$ и ниво $h$ няма да има нужда да задаваме въпроси). За да научим ребрата между нивата, ще използваме идеята с едновременните двоични търсения, като разбира се за дадено ниво $l$ има смисъл да ограничим търсенето за бащите до върховете от ниво $l-1$. В зависимост от имплементацията и дребните оптимизации, които могат да се приложат това решение трябва да взема от $80$ до $100$ точки. Оценяващата формула е така направена, че да се усети разликата, ако се питат повече въпроси при малко по-неоптимални реализации.

Остана да обсъдим математическите детайли. Ако се замислим, $h-1$ е долна граница за най-малкия брой въпроси, който ни трябва за познаване на произволно дърво. Върховете от най-дългия път са почти неразличими с нашите въпроси - ако мислим за дърво-пръчка, то винаги ще ни трябват поне $N-2$ въпроса, за да научим точната пермутация на върховете. Интересно е да разберем колко въпроси добавя нашият алгоритъм с двоичните търсения. Оказва се, че не е много. Нека означим с $l_0, l_1, \dots, l_h$ броя върхове по нива. Тогава за двоичните търсения ни трябват:

$$\sum_{i=1}^{h} \log_2 l_{i-1} = 0 + \sum_{i=2}^{h} \log_2 l_{i-1}$$

Така това всъщност е сумата от двоичните логаритми на $h-1$ числа, които се сумират до $N-1$. Може да се докаже, че максимумът се достига когато върховете са

най-изравнени по нива - по $2$ и/или по $3$ върха на ниво. Дори като вземем предвид, че винаги имаме по $h-1$ въпроса за намиране на върховете по нива, общият брой въпроси винаги е под $N$, т.е. този алгоритъм е винаги по-добър от предишния. Най-лошите случаи са всъщност като имаме дърво-пръчка или на всички нива без нулевото имаме по $2$ или по $3$ върха. Затова последната подзадача има допълнително ограничение, че височината на дървото е най-много $25$ - тогава се усеща по-значително подобрението. По този начин има и малка подсказка в условието за подхода на решението. При тези ограничения в най-лошия случай ни трябват $169$ въпроса, което се достига при "изравнени" нива и е постигнато в някои тестове.

Брой въпроси (в най-лошия случай): $h - 1 + (h-1)\log_2 \frac{N-1}{h-1}$.
Сложност: $O(N^2)$.

---

Началната идея за задачата беше да има неизвестен *DAG* (ориентиран ацикличен граф), който да бъде познат. В този случай графът имаше свойството *антитранзитивност*, защото ребро, което се получава по транзитивен начин (например ако имаме ребрата $(1,2),(2,3),(1,3)$, то реброто $(1,3)$ следва транзитивно от $(1,2)$ и $(2,3)$) не може да бъде установено чрез топологичната наредба. Затова и част от условието е написана все едно е даден граф, а не дърво. Но в крайна сметка вариантът за дърво беше най-интересен, а и за произволен граф не знаем дали има по-добро решение от вече описаното с $N-1$ въпроса, при което намираме достижимите върхове от всеки връх. При произволен граф като имаме тази информация трябва да направим транзитивна редукция, за да отстраним транзитивните (излишните) ребра, което може да стана с алгоритъма на Уоршъл със сложност $O(N^3)$.

*Автор: Илиян Йорданов*