**Анализ**

Очевидно условието на задачата задава неориентиран граф без цикли с върхове полетата и ребра, съседствата между тях. Това означава, че играта се играе върху дърво.

Първата подзадача е за 10 точки. Тук полетата са в една линия, тоест дървото е линейно (дърво пръчка). Може да приложим малко разсъждения. Ако ***N*** e четно число, то ако **Д**ени сложи пионката в някой от двата края, лесно се вижда, че ще загуби. Ако не е в двата края, то ще имаме два участъка с четен и нечетен брой полета и Боби ще премести пионката в участъка с нечетен брой полета, за да си осигури победа. В този случай тя винаги губи. Ако пък ***N*** e нечетно число, в половината ситуации губи, в другата печели – ако двата участъка (или единия) са с нечетна дължина, то Боби винаги печели, а ако двата участъка са с четна дължина, то той винаги губи и съответно **Д**ени печели. Това означава, че просто имаме два случая, в които трябва да изведем отговорите, които получихме. Сложността е .

Втората подзадача е за 30 точки. Тя вече е стъпка към решаване на задачата. Ще разгледаме всички възможност за начален връх на пионката. За всяка от тях в един масив ще означаваме дали този връх е печеливша позиция или губеща. Рекурентната зависимост е лесна – ако за даден връх има съсед, който е губеща позиция, то текущия връх е печеливша. Иначе текущият връх е губеща. Това ни позволява с едно *DFS* по графа да намерим, дали тръгвайки от дадена позиция ще победим или не (ако след обхождането, началният връх е губеща позиция, то ако първият играч постави там пионката, хода ще стане на другия играч, който ще е в губеща позиция) Сложността е .

Третата подзадача е за 15 точки. Тя разширява възможностите на предишното решение. Това, което бави там е, че пускаме обхождането от всички върхове. Но тук дърветата са пълни двоични дървета. При тях всички върхове, които са листа, са симетрични – нищо не ги различава, така че е достатъчно да намерим отговора само за едно от тях, а за другите ще е същият. Общо погледнато – върховете, които са на едно ниво са симетрични и е достатъчно да пуснем обхождането само за един от тях. Понеже нивата на това дърво са логаритъм на брой, то сложността на това решение е .

Четвъртата подзадача е за 35 точки. Тук вече трябва да измислим същинската идея за решаването. Нека гледаме дървото като кореново с произволен корен. Главното наблюдение е, че веднъж, когато започнем да се движим надолу, няма връщане назад, т.е. продължаваме само надолу. Затова решаваме задачата с динамично по дървото и две обхождания на дървото. Първото обхождане е отдолу-нагоре по дървото за установяване на печелившите позиции, когато се движим само надолу. Това всъщност ни е обхождането от втората подзадача. Второто обхождане на дървото е отгоре-надолу. Сега искаме да намерим дали връх е в печеливша позиция, ако първо евентуално се движим нагоре и по някое време евентуално почнем да се спускаме. Първо намираме отговорът за корена на дървото, после за някое дете, за някой следващ наследник и т.н. малко, по-малко за всички върхове. Очевидно ако бащата е губеща позиция, то тук е печеливша. Освен това ако имаме дете, където е губеща позиция спрямо първото обхождане (защото почваме да се движим само надолу), то тук е печеливша позиция. Във всички останали случаи, текущият връх е губеща позиция. Има една важна особеност – при второто обхождане, когато за връх гледаме позицията на бащата, трябва да я отчитаме като печеливша или губеща без да гледаме текущия връх като дете на бащата, защото ако се качим при бащата, то не можем да слизаме и да се върнем отново на текущия връх. Това става най-лесно, като при обхождането преди да отидем във връх гледаме каква трябва да е позицията на бащата спрямо този връх, без да гледаме неговата позиция. Тогава спрямо позицията на детето, бащата ще е в губеща позиция, ако неговият баща е печеливша и всичко останали деца са печеливши позиции. В условието има и една уловка. Никъде не е казано, че графът е свързан и има примери, в които не е (затова се въвеждат и брой ребра). За да получим всички отговори трябва да пускаме обхожданията от всички възможни непосетени върхове. Цялото описано решение става с два *DFS*-а и сложността е .

Последната подзадача е за 10 точки. Очевидно няма как да оптимизираме тази сложност, поне трябва да прочетем данните. Тук трябва да намалим константата на решението и да напишем по-бърза реализация на идеята. Едно от важните неща е да не ползваме списък на наследниците и съответно масив от вектори, защото това е бавно. Когато имаме дърво, можем да ползваме представянето списък на ребрата, като си направим указатели, така че да можем да обхождаме излизащите ребра от даден връх. Това забързва решението значително. Друго наблюдение, което можем да направим е следното – първото обхождане със сигурност трябва да е в дълбочина (отговор за връх, получаваме чак след като сме минали през всички негови наследници), но за второто обхождане можем да използваме и търсене в широчина. То е по-бързо, защото е итеративно, за разлика от рекурсивното *DFS*. Има и други малки наблюдения, които също подобряват константата на решението. Разбира се, цялостното решение е с линейна сложност.

*Автор: Илиян Йорданов*