АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА

ПОДАРЪК

Нека с *gi* и *si* означим минималните количества златни и сребърни монети, които трябва да се съдържат в подаръка, за да престанат разбойническите грабежи по път с номер *i*.

Да предположим, че оптималното решение се състои от *A* златни и *B* сребърни монети. Първото нещо, което трябва да се забележи е, че съществуват два индекса *i* и *j* (възможно е тези индекси да съвпадат), такива че *A= gi* и *B= sj*. Наистина, в противен случай бихме могли да намалим *A*  или *B* (а с това и количеството тугрици, необходимо за закупуването на златните и сребътните монети) без да развалим свързаността на графа.

Да означим с *R(A,B)* подграфа, в който за всички ребра *i* е изпълнено *gi* ≤ *A* и  *si* ≤ *B*.

Да означим с *T*(*A*) подграфа, в който за всички ребра *i* е изпълнено *gi* ≤ *A.* Нека ребрата в този граф имат тегла, като теглото на ребро с номер *i* е равно на *si*. В този граф да намерим обхващащо дърво, за което максималното ребро има минимална стойност. Може да се докаже, че, при фиксирана стойност на *A,* най-малката стойност на *B*, за която графът *R*(*A*, *B*) все още е свързан, ще бъде точно тази минимална стойност на максималното ребро.

***Лема*:** *Минималното обхващащо дърво на даден граф едновременно се явява и обхващащо дърво, за което максималното ребро е минимално възможно*.

***Доказателство*.** Да разгледаме произволно минимално обхващащо дърво *L* на графа. Ако съществува обхващащо дърво *P*, в което всички ребра имат тегло, строго по-малко от теглото на максималното ребро в *L*, то тогава ние бихме могли да премахнем от *L* максималното му ребро – дървото ще се разпадне на две части. Тези две части бихме могли да свържем с ребро от *P*, получавайки отново обхващащо дърво, чийто сумарно тегло ще бъде по-малко от това на *L.* Това противоречи на допускането, че *L* е минимално обхващащо дърво. Значи такова *P* не съществува.

Да сортираме всички ребра на първоначалния граф по нарастване на *gi*. Да започнем последователно, в ненамаляващ ред, да даваме стойности на *A*, равни на *gi*. За всяко *A*, т.е. за всяко *i*, ще се получава граф *T(A)*, чийто ребра ще са тези с индекс *j* ≤ *i*. Трябва бързо да можем да намираме минимално обхващащо дърво в този граф.

Да предположим, че за някое *i* сме намерили минимално обхващащо дърво в графа *T*(*gi*) (ако графът не е свързан, то за всяка негова компонента на свързаност е намерено минимално обхващащо дърво). Да добавим към графа *i* + 1-то ребро. Ако това ребро съединява различни компоненти на свързаност, то просто го добавяме, получавайки с една компонента на свързаност по-малко и готовото минимално обхващащо дърво за новополучената компонента. Ако реброто свързва два върха от една и съща компонента на свързаност, то ще се получи точно един цикъл. Да намерим максималното ребро в този цикъл и да го премахнем, като по този начин ще получим минимално обхващащо дърво в новия граф *T*(*gi+1*).

Намирането на максималното ребро в цикъл е със сложност *O*(*N*) и може да се наложи да бъде осъществявано *M* пъти. Като добавим и първоначалното сортиране на ребрата, то получаваме алгоритъм със сложност *O*(*NM* + *MlogM*).