Sponzoři Fiks

Kryštof Olík

Zformulujme si nejprve zadání úlohy bez všech nepodstatných věcí okolo. Na vstupu máme na prvním řádku počet zvířat a počet sponzorů a poté následují dva listy. První list obsahuje ID zvířete společně s jeho jménem a druhý list obsahuje jména sponzorů společně s ID zvířat, které může sponzor sponzorovat. Úkolem je najít sponzora pro co nejvíce zvířat a seřadit je společně se jménem sponzora vzestupně dle abecedy podle názvu zvířete.

Nejprve si přetvoříme vstupní text do příjemnějších datových struktur. Vytvoříme slovník se jménem "zvirata", který jako klíč drží ID zvířete a jako hodnotu jméno zvířete. Tento slovník má prostorovou složitost O(2N), kde N je počet zvířat. Poté vytvoříme další slovník s názvem "sponzori", který jako klíč drží jméno sponzora a jako hodnotu drží list s ID zvířat, které může sponzor sponzorovat. Tento slovník má prostorovou složitost O(Z + v1 + v2… + vZ), kde Z je počet sponzorů a v1 je velikost prvního listu a vZ posledního.

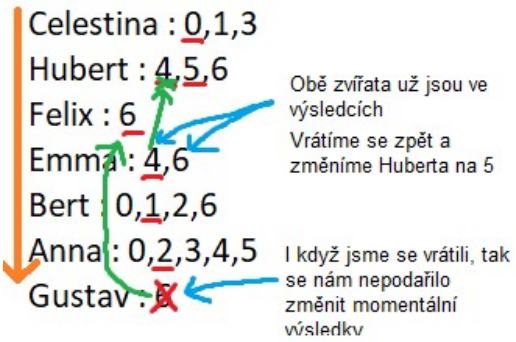
Poté se vrhneme na algoritmus samotný. Vytvoříme prázdný slovník jménem "výsledky", který bude mít jako klíč jméno zvířete a jako hodnotu jméno sponzora. Začneme iterovat přes všechny sponzory. První vytvoříme prázdný set se jménem "recursion_done", který bude držet jména sponzorů, přes které už šla rekurzivní funkce, abychom algoritmus chránili před nekonečným cyklem. Půjdeme do funkce jménem "algo" se jménem sponzora jako argumentem.

Poté iterujeme přes zvířata, která může sponzor sponzorovat, a pokud narazíme na zvíře, které ještě není ve výsledcích, tak ho zařadíme společně se jménem sponzora do výsledků a půjdeme dále na dalšího sponzora, kde budeme opakovat tyto kroky. Pokud se nám nepovede najít zvíře, které ještě není ve výsledcích, tak se pokusíme poupravit momentální výsledky.

Znovu iterujeme přes možná zvířata, ale tentokrát najdeme sponzora, který už toto zvíře ve výsledcích sponzoruje. Hledaný sponzor vyzkouší sponzorovat jiné zvíře, pokud se povede, vrátí True a půjdeme na dalšího sponzora, pokud se to nepovede, tak budeme opakovat kroky minulé věty, dokud nenarazíme na definitivní konec (když už jsme vyzkoušeli všechny možné sponzory).

Na úplný konec zkontrolujeme, zda se počet sponzorů ve výsledku rovná Z (počtem sponzorů v zadání) a seřadíme výsledky vzestupně dle abecedy podle názvu zvířete, pomocí funkce sort (Timsort).

Grafické znázornění algoritmu:



Co se týče časové složitosti algoritmu, tak v nejhorším případě, kdy se musí na každého sponzora iterovat přes všechny sponzory, které už jsou ve výsledku (což je velice nepravděpodobné), je časová složitost O(2NZ), kde N je počet zvířat a Z počet sponzorů. Sort má asymptotickou složitost O(n log n), kde n je počet sponzorů ve výsledku.

Co se týče paměťové spotřeby, algoritmus vyžaduje ten set „recursion_done“, který je maximálně O(N), výsledky, které jsou také maximálně O(N), a pak samozřejmě vstup o kterém jsem mluvil ve druhém odstavci.

Nejdůležitější je však otázka: Funguje náš algoritmus? Nemůže se stát, že bude možnost sponzorování, kterou algoritmus nenajde? Jelikož pokaždé, když algoritmus nemůže najít zvíře ke sponzorování, protože už jsou všechna ve výsledcích, iteruje přes všechny sponzory, které se už nacházejí ve výsledcích a zkouší to u každého zvlášť změnit, tak se nemůže stát, aby jistou možnost přehlédl.

K řešení připojuji algoritmus naprogramovaný v jazyce python. Program samotný je přesným přepisem výše uvedeného algoritmu.