Řešení úlohy č. 3
Ochočení

Řešení

Vstup si můžeme vhodně zakódovat jako binární sekvenci. V sekvenci zapišme 1, tam kde je změna mezi polohami sousedních šelem. Zapíšeme 0, pokud sousední šelmy mají stejnou polohu. Přidáme si jednu 0 na začátek a na konec, abychom mohli detekovat i změnu u první a poslední šelmy. Chceme zjistit počet povelů k získání sekvence $0, 0, 0, \ldots 0$.

Cílem je se zbavit všech 1 v sekvenci (nazvěme jedničku tokenem). Povel, jediná povolená operace, vezme token a posune ho o prvočíslo pozic v sekvenci. Když se dva tokeny setkají, zmizejí (změní se na 0).

Mějme tokeny na pozicích i a j. Cena tj. počet operací k vymazání těchto tokenů se počítá následovně. Pokud:

- $|i-j|$ je prvočíslo: stačí 1 operace, cena je 1. Tedy vezmu token, posunu o $|i-j|$, tokeny se setkají a zmizí. Méně operacemi to nejde.
- $|i-j|$ je sudé: potřebujeme 2 operace. To lze určit se znalostí, že každé sudé číslo do 10^7 je rozdílem dvou prvočísel (viz Goldbachova hypotéza[1]) platící pro čísla v našem rozsahu). Víme, že existuje prvočíslo p, které změní polohu šelmám o j tak, aby ve výsledku $|i+p-j|$ bylo prvočíslo. Jednou operací nic nezmůžeme, 2 jsou tedy nejlepší řešení, co nalezneme.
- $|i-j|$ je liché ale není prvočíslo: potřebujeme operace 3. Nejdříve připočteme nějaké prvočíslo p, tak že $|i+p-j|$ je sudé a pak postupujeme jako o bod výše. Jednou operací nic nezmůžeme (rozdíl není prvočíslo), musíme udělat operaci, a tak se dostaneme na sudé číslo a u toho víme, že méně než 2 operacemi to nejde.

Vypišme si počáteční pozice tokenů $y_1, \ldots, y_M$. Tokeny chceme spárovat do $M/2$ dvojic, tak abychom minimalizovali počet operací. Označme si M_e, M_o počet tokenů na sudých, lichých pozicích respektivě. Pokud uděláme k dvojic, co potřebují jednu operaci, celkový počet operací bude:

$$k \cdot 1 + (M_e - k)//2 + ((M_o - k)//2) \cdot 2 + ((M_e - k)\%2) \cdot 3$$

kde // je celočíselné dělení a % je zbytek po dělení.

Chceme maximalizovat k. To můžeme provést bipartitním párováním na grafu, kde vrcholy jsou tokeny a váha hrana je mezi tokeny pokud je jejich cena smazání 1 (tj. jsou prvočíselně daleko od sebe). Následně nám zbyly už ceny smazání 2 nebo 3. Cena 2 se vyskytuje u tokenů, které jsou ve stejé množině M_e, nebo M_o (sudé-sudé = sudé, liché-liché = sudé). Jelikož cena mezi libovolnými tokeny ve stejné množině je vždy dva, snadno napárujeme tyto tokeny. Nakonec nám z obou množin zbyl buď žádný nebo jeden vrchol, který musíme napárovat za cenu 3.

Vytvořit takovýto graf znamená zjistit cenu hrany mezi M_e a M_o, což je $O(n^2)$ porovnání, kde n je počet šelem. O maximálním bipartitním párování a jeho složitosti se můžete dočíst zde[2]). Z toho víme, že naše bipartirní párování nemá složitost horší než $O(n^2)$, tedy konečná složitost je $O(n^2)$.

[1]) `https://en.wikipedia.org/wiki/Goldbach%27s_conjecture`
[2]) `https://ksp.mff.cuni.cz/encyklopedie/parovani.html`

Proč je algoritmus správný? Převedení na tokeny jsme schopni provést i zpět a každé zadání má zakódování v tokenech. To samé platí i pro převedení tokenů na čísla jejich pozic. Algoritmus bipartirního párování je známý algoritmus, který je správný a víme že dává optimální řešení. (Nebo se to dočteme v odkazu) Algoritmus je konečný, ze stejného důvodu, tedy převedení na tokeny je M operací, převedení na pozice je také M operací a bipartirní párování je konečný algoritmus.

Bodování

- + 1 1. příklad správně
- + 1 2. příklad správně
- + 2 pravidla pro sudé, prvočíslo a ostatní
- + 1 složitost

Podle složitosti řešení:

- + 1 má nějaké fungující řešení (nemusí nalézt optimální počet)
- + 2 řešení na dobré cestě ale nedotažené
- + 3 Bruteforce (správný počet operací)
- + 4 nějaká optimalizace
- + 5 párování (či jiné řešení v $O(n^2)$)