Cvičenie 4

Príklad 4.1.

Máme mriežku $n \times n$, v ktorej sú samé 0. Nad každým riadkom aj stĺpcom vieme robiť operáciu `zmeň`, ktorá zmení všetky hodnoty v danom riadku/stĺpci z 0 na 1 a naopak.

Vieme, že sme spravili r operácií `zmeň` na riadkoch a s operácií `zmeň` na stĺpcoch. Nevieme však ani v akom poradí sme ich robili, ani na ktorých riadkoch/stĺpcoch, pričom ten istý riadok/stĺpec sme mohli vybrať aj viacej krát. Jediné čo vieme, je že po týchto zmenách bolo v mriežke presne k jednotiek.

a) Koľko je rôznych možností, ako mohla vyzerať výsledná mriežka?

b) Nad každý stĺpec si napíšeme, koľkokrát sme na ňom použili operáciu `zmeň` a podobne napíšeme vedľa každého riadku počet operácií `zmeň` na ňom. Koľko je rôznych možností, ako mohli vyzerať tieto čísla?

Náčrt riešenia 4.1.

Úlohu si najprv potrebujeme zjednodušiť. Uvedomme si, že na poradí operácií vôbec nezáleží, pre každé políčko nás totiž zaujíma len počet, koľkokrát bol zmenený. A tiež si všimnime, že ak nejaký riadok/stĺpec zmeníme dvakrát, je to rovnaké, ako by sme ho nemenili vôbec.

Zaujímajú nás teda len tie riadky a stĺpce, ktoré sa zmenili nepárny počet krát. Nevieme, ktoré to boli, postupne teda skúšajme všetky možné počty, nech p_r je počet riadkov s nepárnymi zmenami a p_s počet stĺpcov. Pre zadaný počet riadkov a stĺpcov vieme vypočítať počet 1 vo výslednej mriežke – políčko obsahuje 1 iba ak patrí zmenenému riadku a nezmenenému stĺpcu ($p_r \cdot (n - p_s)$) alebo naopak ($p_s \cdot (n - p_r)$). Pre danú možnosť teda vieme v konštantnom čase zistiť, či vedie k správnej možnosti.

a) Pre každú z vyhovujúcich hodnôt p_r, p_s započítame všetky možnosti, ako vybrať p_r zmenených riadkov a p_s zmenených stĺpcov, čo je súčin kombinačných čísel $\binom{n}{p_r}\binom{n}{p_s}$. Na záver si uvedomíme, že takto dostaneme každé vyplnenie tabuľky práve dvakrát, čiže na záver celkový súčet vydelíme dvomi.

Uvedieme ešte, ako zdôvodniť, že každú tabuľku máme dvakrát. Ak si pre konkrétnu zo započítaných tabuliek povieme, že stĺpec 0 meníme, tak jeho hodnoty nám jednoznačne určia zmeny riadkov a následne z riadku 0 dostaneme jednoznačne určenie prepnutie zvyšných stĺpcov. Ak si povieme, že stĺpec 0 nemeníme, tak dostaneme opäť jedinú možnosť prepnutia. Teda každú zo započítaných tabuliek sme započítali práve dvakrát – tieto dve možnosti sa líšia tým, že invertujeme, ktoré riadky a ktoré stĺpce prepíname. (To vieme dosiahnuť zmenením všetkých riadkov a všetkých stĺpcov, čo nám tabuľku nezmení.)

b) Opäť pre každú z vyhovujúcich hodnôt p_r, p_s započítame všetky možnosti výberu p_r zmenených riadkov a p_s zmenených stĺpcov, teda $\binom{n}{p_r}\binom{n}{p_s}$ možností. Potom zistíme, koľkými spôsobmi sme mohli spraviť r operácií na riadkoch, aby práve p_r z nich bolo zmenených nepárne veľa krát. Na to nám stačí k už zmeneným p_r riadkov zvoliť $\frac{r-p_r}{2}$ riadkov, ktoré zmeníme dvakrát, na čo podľa kombinácií s opakovaním máme $C(n, \frac{r-p_r}{2})$ možností. Teda pre každé vyhovujúce p_r, p_s sčítavame

$$\binom{n}{p_r}\binom{n}{p_s}\binom{n}{\frac{r-p_r}{2}}\binom{n}{\frac{s-p_s}{2}}.$$

V oboch riešeniach výber hodnôt p_r a p_s nám dáva $O(n^2)$ časovú zložitosť, na výpočet kombinačných čísel teda môžeme využiť počítanie celého Pascalovho trojuholníka, ktorým si predpočítame dopredu všetky potrebné kombinačné čísla v čase tiež $O(n^2)$.

Táto zložitosť je za predpokladu, že vieme násobiť a sčítavať v konštantnom čase, čo nie je v tejto úlohe pravda (lebo kombinačné čísla rastú naozaj rýchlo). Preto väčšinou pri takýchto úlohách sa nás pýtajú na zvyšok po delení nejakým číslom m, často prvočíslom. To dosiahneme jednoduchou úpravou – všetky operácie násobenia a sčítavania budeme robiť modulo m. V riešení a) ešte potrebujeme deliť dvomi, čo vieme spraviť pomocou inverzného prvku pre nepárne m alebo tak, že si najprv vypočítame všetko modulo $2m$ a potom vydelíme dvomi.

Príklad 4.2.

Predstavte si 3-CNF formulu, kde každý výskyt operátora OR je nahradený operátorom XOR. Rozhodnite, či je nová formula splniteľná.

Náčrt riešenia 4.2.

V skutočnosti nezáleží na tom, či je daná formula 3-CNF alebo 50-CNF, riešenie je rovnaké.

Pozrieme sa na jednu klauzulu $(l_1 \oplus l_2 \oplus l_3)$. Jej pravdivosť je totožná s tvrdením, že súčet týchto hodnôt je nepárny, respektíve že je ich súčet rovný 1 modulo 2.

Čo v prípade, že niektorá z premenných je pod negáciou? Negáciu premennej vieme vyjadriť pomocou pripočítania jednotky, t.j. $\bar{l} \iff l \oplus 1$. A teda, rovnicu $\bar{l_1} \oplus l_2 \oplus l_3 = 1\ (mod\ 2)$ vieme prepísať na ekvivalentnú rovnicu bez negácii: $l_1 \oplus l_2 \oplus l_3 = 0\ (mod\ 2)$.

Klauzuly pôvodnej formuly teda vieme vyjadriť ako sústavu lineárnych rovníc nad poľom F_2. Ďalej stačí len použiť algoritmus Gaussovej eliminácie.

Príklad 4.3.

Pre daný reťazec (dĺžky n) nájdite jeho najdlhší súvislý podreťazec, ktorý je palindrómom.

Náčrt riešenia 4.3.

Bezduchý bruteforce vieme spraviť v čase $O(n^3)$. Trochu lepšie riešenie je pre každú medzeru a písmeno nájsť najdlhší palindróm, ktorý má stred na tomto mieste. Takýto prístup dá riešenie v čase $O(n^2)$.

Pozrieme sa na chvíľu na jednoduchšiu úlohu: rozhodnite, či daný reťazec obsahuje palindróm dĺžky *aspoň* L (a ak áno, tak vráťte jeho pozíciu)? Kebyže vieme túto úlohu riešiť v čase $O(f(n))$, tak by sme vedeli pomocou binárneho vyhľadávania nájsť najdlhší palindróm v čase $O(f(n) \cdot \log n)$. Binárne vyhľadávanie využíva vlastnosť, že každý palindróm dĺžky $n \geq 2$ obsahuje v sebe palindróm dĺžky $n-2$, čiže vyhľadávanie je potrebne spraviť zvlášť pre palindrómy párnej a nepárnej dĺžky.

Ako túto jednoduchšiu úlohu riešiť? Treba porovnať každý súvislý podreťazec dĺžky L s jeho reverzom. Ak sa zhodujú, tak tvoria palindróm danej dĺžky. Rabin-Karpovo lineárne hashovanie reťazcov (rolling hash) ponúka možnosť porovnávať dva ľubovoľné podreťazce v konštantnom čase s lineárnym predspracovaním. Riešenie je teda jednoduché: spočítam rolling hash pre reťazec a jeho reverz v čase $O(n)$, a následne prejdem všetky podreťazce dĺžky L a porovnám ich hash s hashom ich reverzu. To by umožnilo riešiť celkovú úlohu v čase $O(n \cdot \log n)$.

Existuje aj riešenie v čase $O(n)$, napríklad Manacherov algoritmus[1].

Príklad 4.4.

Pre danú maticu T nájdite všetky výskyty matice P v nej, pričom všetky riadky matice P

[1] https://en.wikipedia.org/wiki/Longest_palindromic_substring

sú rozličné.

Náčrt riešenia 4.4.

Jedná sa o zjednodušenú verziu Baker-Birdovho algoritmu.

Najprv nájdime všetky výskyty jednotlivých riadkov P v matici T pomocou Aho-Corasickovej[2] algoritmu. Nakoľko všetky riadky sú rozličné, tak na jednej pozícií v matici T sa môže vyskytovať nanajvýš jeden z riadkov P. Toto pozorovanie nám umožňuje zaznačiť jednotlivé výskyty kompaktne do jednej matice Q, kde na pozícii (i, j) dáme hodnotu k, ak na j-tej pozícii i-tého riadku matice T začína výskyt k-tého riadku matice P, a hodnotu -1 v ostatnom prípade.

Zamyslíme sa teraz, ako by vyzeral výskyt *matice* P z pohľadu matice Q? Bezprostredne pod sebou sa musia vyskytovať postupne prvý, druhý až po posledný riadok matice P. Čiže príslušný stĺpec matice Q musí obsahovať postupne hodnoty $123456\ldots n$, kde n je počet riadkov matice P. Nájsť všetky výskyty patternu $123456\ldots n$ vieme napríklad pomocou Knuth-Morris-Prattovho algoritmu.

[2]Áno, Margaret John Corasicková je žena