Úloha č. 5
Vstupní zkouška

Rozmysli, popiš a naprogramuj! 10 b

Nechť mřížka na vstupu je matice A o rozměrech n, m. Prvkům matice A budeme říkat symboly a tyto symboly jsou z abecedy $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Zadání po nás chce nalézt největší podmatici P matice A takovou, že symboly v P lze přeskládat na palindrom.

Pozorování 1

Bez újmy na obcenosti můžeme uvažovat, že $n \leq m$. Kdyby tomu tak nebylo, můžeme podmatici P hledat v transponované matici A^T, která má prohozené rozměry. Samozřejmě výsledné souřadnice poté opět musíme transponovat zpět.

Pozorování 2

Dále pozorujeme, že palindrom lze postavit z nějaké sady symbolů $s_1, s_2, \ldots, s_k$ s četnostmi $c_1, c_2, \ldots, c_k$ právě tehdy, když nejvýše jedna z četností je liché číslo. Navíc, nemusíme přesně vědet vyčíslené četnosti, stačí nám pouze, zda-li jsou liché nebo sudé – tedy jejich parita.

V našem případě to znamená, že pokud pro nějakou kandidátní podmatici P' víme paritu jednotlivých četností symbolů z abecedy Σ, snadno odpovíme, zda-li lze ony prvky přeskládat na palindrom – celkem je třeba se podívat na $|\Sigma|$ parit, tedy odpověď trvá $O(|\Sigma|)$.

Algoritmus 1 - naivní

S prvními dvěma pozorováními již dokážeme úlohu vyřešit. Stačí nám projít všechny podmatice, pro každou spočítat parity jednotlivých symbolů v ní a odpovědět. Toto řešení je správné, nicméně vede na velmi vysokou složitost $O(n^3m^3|\Sigma|)$ – všech podmatic je řádově n^2m^2, výpočet parit jedné podmatice je nm a odpověď v $|\Sigma|$.

Maticové prefixové součty

Všimneme si, že Algoritmus 1 zbytečně pořád dokola musel pro jednotlivé podmatice vypočítávat všechny parity znovu a znovu. Toto můžeme odstranit jednoduchým předpočítáním tzv. maticových prefixových součtů s jejichž pomocí tento dotaz dokážeme vykonat v konstantním čase.

Odstupme nejdříve do 1 rozměru a zabývejme se prefixovými součty jednoduchého pole. Mějme pole s prvky $A_1, A_2, \ldots, A_N$. Vytvoříme druhé pole prefixových součtů $S_1, S_2, \ldots, S_N$

pro jehož prvky platí:

$$S_q = \sum_{i=1}^{q} A_i$$

Jinými slovy, prvek S_q je součet prvních q členů pole A. Jak tohoto využít pro rychlou odpověď na dotaz typu: jaký je součet prvků pole A mezi indexy $l, r, l \leq r$? Tento výraz je ovšem roven jednoduchému rozdílu námi předpočítaných prefixových součtů:

$$A_l + A_{l+1} + \ldots + A_{r-1} + A_r = S_r - S_{l-1}$$

Tedy na dotazy tohoto typu jsme schopni odpovědět v konstantním čase, když si je dovolíme předpočítat v lineárním čase.

Maticové prefixové součty jsou jen zobecněním výše popsaného nápadu do vyššího rozměru (lze je samozřejme zobecnit do libovolného rozměru, myšlenka je pořád stejná). Tedy u matic definujeme prefixové sumy následovně:

$$S_{q,p} = \sum_{i=1}^{q} \sum_{j=1}^{p} A_{i,j}$$

A dotaz na součet v libovolné podmatici s levým horním rohem t, l a pravým dolním rohem b, r lze přeložit na:

$$\sum_{i=t}^{b} \sum_{j=l}^{r} A_{i,j} = S_{b,r} - S_{t-1,r} - S_{b,l-1} + S_{t-1,l-1}$$

Tímto jsme schopni odpovědět na libovolnou podmatici opět v konstantním čase.

Zamysleme se ještě, jaké vlastnosti musí mít naše binární operace, kterou používáme na kombinování prvků (v příkladu výše to byla operace součet). Pro výpočet prefixových součtů je potřeba pouze *asociativita*. Pro naše dotazy ovšem ještě potřebujeme existenci *inverzní operace* k naší (v příkladu výše to byla operace odčítání).

Algoritmus 2 - použití prefixových součtů

Všimneme si, že parity lze efektivně kódovat do 10 bitů celého čísla – říkejme tomu maska parit – a pro kombinaci dvou masek parit lze použít operaci *xor*. Algoritmus 1 vylepšíme tím, že nejdříve předpočítáme prefixové součty masek parit s operací xor pro vstupní matici, což stihneme v $O(nm)$. Nyní nám odpadá zdlouhavé opakované počítání parit jednotlivých symbolů, protože to s pomocí prefixových součtů dostaneme v konstantním čase. Celková složitost vylepšeného algoritmu je $O(n^2 m^2 |\Sigma|)$.

Algoritmus 3 - nenaivní řešení

Abychom prolomili složitost pouhého výčtu všech podmatic, využijeme následující myšlenky. Představme si, že zpracováváme pás matice mezi indexy $b, t, b \leq t$. Inicializujeme prázdnou hashovací tabulku H, kam budeme ukládat masku parit a pozici, na které jsme na tuto masku poprvé narazili.

Pás matice procházejme zleva doprava a nacházejme se na indexu i a maska parit stávající podmatice je M_i. Nejdříve se podíváme, zda-li se v tabulce H nenachází maska X, pro kterou platí, že $M_i \; xor \; X = O$, kde O je maska, která má nejvýše jeden bit nastavený na 1, což odpovídá situaci, kdy z podmatice jsme schopni sestavit palindrom. Toto lze realizovat v čase $O(|\Sigma|)$ – počet masek O s nejvýše jedním bitem nastaveným je $|\Sigma| + 1$, hashovací tabulku H lze realizovat obyčejným bitovým polem o velikosti $2^{|\Sigma|}$ a hledaná maska $X = M_i \; xor \; O$.

Nakonec, pokud se maska M_i v H ještě nenachází, vložíme ji tam s pozicí i – první pozice, kdy jsme narazili na masku M_i.

Počet všech pásů je $O(n^2)$ a práce v jednom pásu je $O(m|\Sigma|)$, tedy celková složitost je $O(n^2 m|\Sigma|)$.