Generování matic bez zakázaných vzorů

Stanislav Kučera

Informatický ústav Univerzity Karlovy

8. 9. 2016

Obsah

Zakázaný vzor

Definice

Binární matice $M \in \{0,1\}^{m \times n}$ *obsahuje* binární matici $P \in \{0,1\}^{k \times l}$ *jako podmatici*, pokud lze z M vynecháním některých řádků a sloupečků získat matici M' $k \times l$ takovou, že pokud má P jedničku na nějaké pozici, má na téže pozici jedničku i M'. Jinak řekneme, že M *neobsahuje (vyhýbá se)* P *jako podmatici*.

$$P = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad M_1 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \quad M_2 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Zakázaný vzor

Definice

Binární matice $M \in \{0,1\}^{m \times n}$ *obsahuje* binární matici $P \in \{0,1\}^{k \times l}$ *jako podmatici*, pokud lze z M vynecháním některých řádků a sloupečků získat matici M' $k \times l$ takovou, že pokud má P jedničku na nějaké pozici, má na téže pozici jedničku i M'. Jinak řekneme, že M *neobsahuje (vyhýbá se)* P *jako podmatici*.

$$P = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad M_1 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \quad M_2 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Příklady použití

- Pomocí matic bez zakázaných vzorů se dokázal horní odhad časové složitosti algoritmu Efrata a Sharira na “Segment-center problem”.
- Existuje korelace mezi některými třídami matic bez zakázených vzorů a Davenport-Schinzelovým posloupnostmi, které zase korelují se složitostí dolní (horní) obálky arrangementů v rovině.

Markovovy řetězce

Definice (neformální)

Pro předepsané pravděpodobnosti $p_{i,j}$ je *Markovův řetězec* posloupnost prvků $X_0, \ldots$ ze stavové množiny $\mathcal{X}$ dodržující $P[X_{t+1} = j | X_t = i] = p_{i,j}$.

Markovovy řetězce

Definice (neformální)

Pro předepsané pravděpodobnosti $p_{i,j}$ je *Markovův řetězec* posloupnost prvků $X_0, \ldots$ ze stavové množiny $\mathcal{X}$ dodržující $P[X_{t+1} = j | X_t = i] = p_{i,j}$.

Věta (neformální)

Pokud je Markovův řetězec aperiodický, nerozložitelný a symetrický, potom je jeho limita uniformně náhodně rozložena na stavové množině $\mathcal{X}$.

Markovův řetězec pro matice

Pokud chceme generovat matici neobsahující vzor P, postupujeme takto:

1. Zvolíme libovolnou matici M neobahující P.
2. Změníme uniformně náhodně vybraný bit M, čímž dostaneme M'.
3. Pokud M' neobsahuje P jako podmatici, nastavíme $M := M'$.
4. Goto 2.

Markovův řetězec pro matice

Pokud chceme generovat matici neobsahující vzor P, postupujeme takto:

1. Zvolíme libovolnou matici M neobahující P.
2. Změníme uniformně náhodně vybraný bit M, čímž dostaneme M'.
3. Pokud M' neobsahuje P jako podmatici, nastavíme $M := M'$.
4. Goto 2.

Protože definovaný Markovův řetězec splňuje předpoklady věty z minulého slidu, jeho limita je náhodná matice neobsahující vzor P jako podmatici. My ale nemáme čas čekat nekonečně dlouho, a zároveň zmíněná věta ani žádná jiná nedává odhad na dostačující počet iterací (mixing time), takže volbu počtu iterací necháme na uživateli.

Algoritmus pro testování speciálních vzorů

Definice

O matici řekneme, že je to Walking pattern, pokud existuje procházka z levého horního rohu matice do pravého dolního rohu obsahující všechny jedničky v matici.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Algoritmus pro testování speciálních vzorů

Definice

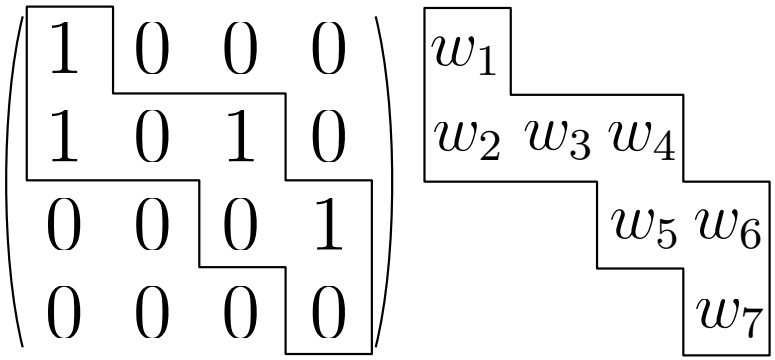
O matici řekneme, že je to Walking pattern, pokud existuje procházka z levého horního rohu matice do pravého dolního rohu obsahující všechny jedničky v matici.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

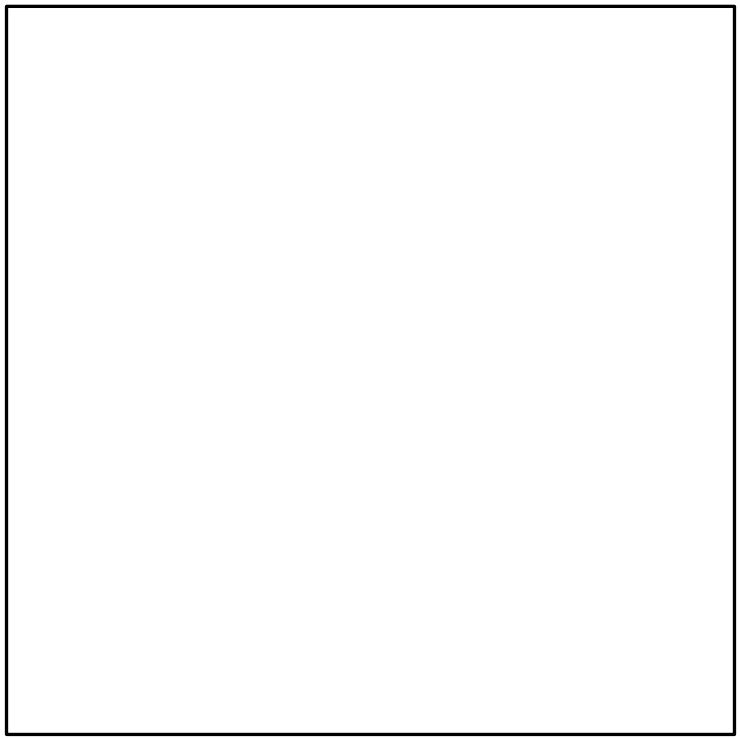
Algoritmus pro testování speciálních vzorů

Definice

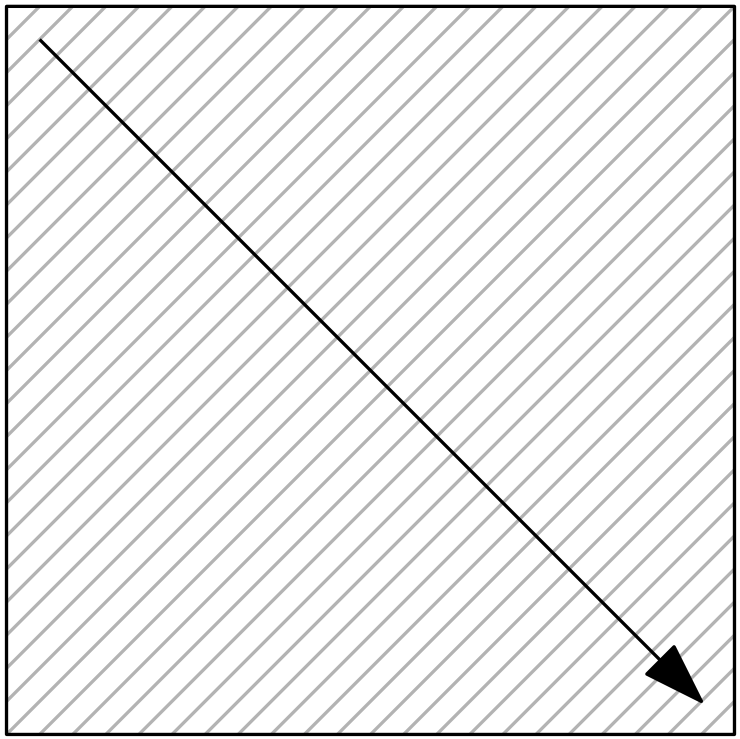
O matici řekneme, že je to Walking pattern, pokud existuje procházka z levého horního rohu matice do pravého dolního rohu obsahující všechny jedničky v matici.



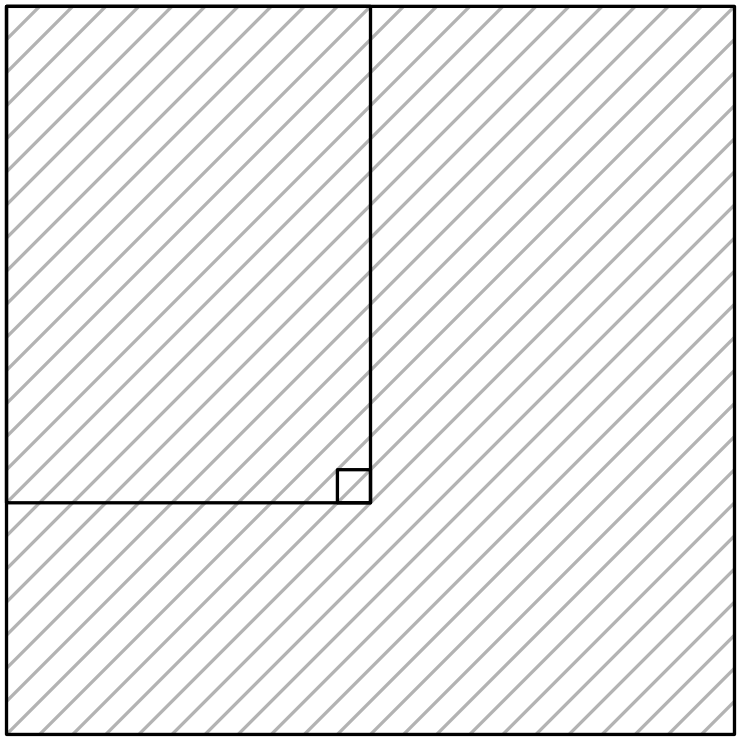
Algoritmus pro testování Walking pattern



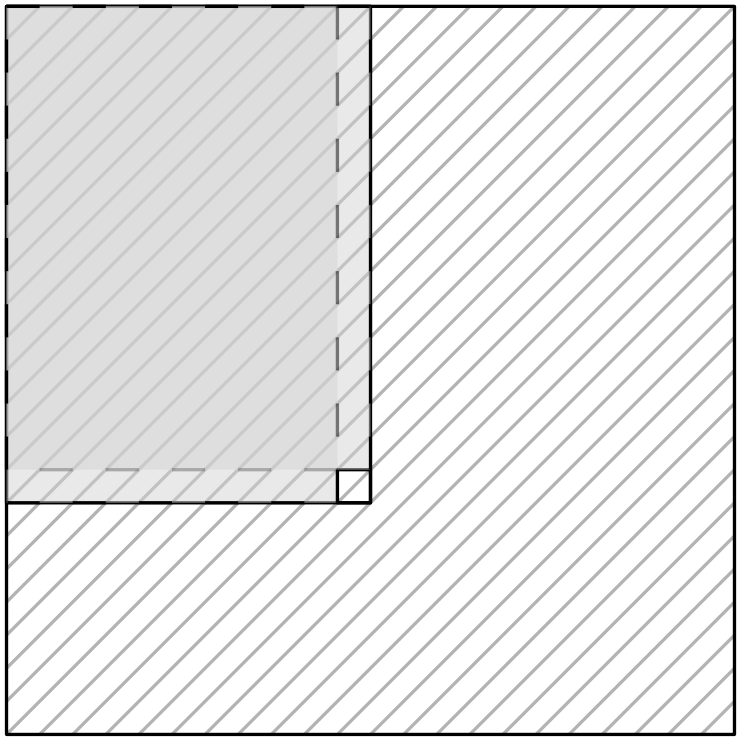
Algoritmus pro testování Walking pattern



Algoritmus pro testování Walking pattern



Algoritmus pro testování Walking pattern



Algoritmus pro testování obecných vzorů

Problém rozhodnout zda daná permutace obsahuje podpermutaci je NP-úplný. Stejný problém můžeme řešit jako otázku zda permutační matice obsahuje permutační vzor, čímž dostáváme, že rozhodování obsahování matice v matici je NP-těžké.

Algoritmus pro testování obecných vzorů

Problém rozhodnout zda daná permutace obsahuje podpermutaci je NP-úplný. Stejný problém můžeme řešit jako otázku zda permutační matice obsahuje permutační vzor, čímž dostáváme, že rozhodování obsahování matice v matici je NP-těžké.

Dva možné brute-force přístupy:

Algoritmus pro testování obecných vzorů

Problém rozhodnout zda daná permutace obsahuje podpermutaci je NP-úplný. Stejný problém můžeme řešit jako otázku zda permutační matice obsahuje permutační vzor, čímž dostáváme, že rozhodování obsahování matice v matici je NP-těžké.

Dva možné brute-force přístupy:

- Vyzkoušet všechny možné podmatice správné velikosti.

Algoritmus pro testování obecných vzorů

Problém rozhodnout zda daná permutace obsahuje podpermutaci je NP-úplný. Stejný problém můžeme řešit jako otázku zda permutační matice obsahuje permutační vzor, čímž dostáváme, že rozhodování obsahování matice v matici je NP-těžké.

Dva možné brute-force přístupy:

- Vyzkoušet všechny možné podmatice správné velikosti.
  - Implementuji jako Slow_pattern.

Algoritmus pro testování obecných vzorů

Problém rozhodnout zda daná permutace obsahuje podpermutaci je NP-úplný. Stejný problém můžeme řešit jako otázku zda permutační matice obsahuje permutační vzor, čímž dostáváme, že rozhodování obsahování matice v matici je NP-těžké.

Dva možné brute-force přístupy:

- Vyzkoušet všechny možné podmatice správné velikosti.
  - Implementuji jako Slow_pattern.
- Postupně namapovat všechny linie (řádky a sloupce) vzoru na všechny možné linie testované matice.

Algoritmus pro testování obecných vzorů

Problém rozhodnout zda daná permutace obsahuje podpermutaci je NP-úplný. Stejný problém můžeme řešit jako otázku zda permutační matice obsahuje permutační vzor, čímž dostáváme, že rozhodování obsahování matice v matici je NP-těžké.

Dva možné brute-force přístupy:

- Vyzkoušet všechny možné podmatice správné velikosti.
  - Implementuji jako Slow_pattern.
- Postupně namapovat všechny linie (řádky a sloupce) vzoru na všechny možné linie testované matice.
  - Implementuji jako General_pattern s použitím mnoha výpočetních i paměťových optimalizací.

General_pattern

Při testování obsahování vzoru postupně mapujeme všechny linie (řádky a sloupce) vzoru na všechny možné linie testované matice.

Optimalizace:

- Některá částečná mapování můžeme sloučit do jednoho a tím ušetřit čas i prostor.

General_pattern

Při testování obsahování vzoru postupně mapujeme všechny linie (řádky a sloupce) vzoru na všechny možné linie testované matice.

Optimalizace:

- Některá částečná mapování můžeme sloučit do jednoho a tím ušetřit čas i prostor.
- Program poskytuje mnoho různých způsobů jak zvolit pořadí, ve kterém se budou linie mapovat.

General_pattern

Při testování obsahování vzoru postupně mapujeme všechny linie (řádky a sloupce) vzoru na všechny možné linie testované matice.

Optimalizace:

- Některá částečná mapování můžeme sloučit do jednoho a tím ušetřit čas i prostor.
- Program poskytuje mnoho různých způsobů jak zvolit pořadí, ve kterém se budou linie mapovat.
- Volitelně program testuje kromě toho, že jsou jedničky tam, kde být musí, také jestli je dost jedniček tam, kam se později budou mapovat dosud nenamapované linie.

General_pattern

Při testování obsahování vzoru postupně mapujeme všechny linie (řádky a sloupce) vzoru na všechny možné linie testované matice.

Optimalizace:

- Některá částečná mapování můžeme sloučit do jednoho a tím ušetřit čas i prostor.
- Program poskytuje mnoho různých způsobů jak zvolit pořadí, ve kterém se budou linie mapovat.
- Volitelně program testuje kromě toho, že jsou jedničky tam, kde být musí, také jestli je dost jedniček tam, kam se později budou mapovat dosud nenamapované linie.
- Protože známe generující proces a již víme, že matice před změnou bitu vzor neobsahovala, víme také, že pokud ho po změně obsahuje tak jedině proto, že právě změněný bit je součástí mapování vzoru.