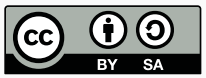
VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY | KATEDRA INFORMATIKY

Strategie řešení sniž a vyřeš

doc. Mgr. Jiří Dvorský, Ph.D.

Stav prezentace ke dni 15. září 2024

Katedra informatiky
Fakulta elektrotechniky a informatiky
VŠB – TU Ostrava



Osnova přednášky

Strategie řešení sniž a vyřeš

Třídění vkládáním – InsertSort

Strategie řešení sniž a vyřeš

Topologické třídění

Strategie řešení sniž a vyřeš

Generování kombinatorických objektů

Generování kombinatorických objektů

- Generování kombinací, variací, permutací, podmnožin je součástí různých algoritmů.
- Typicky jde o výběr nějaké alternativy, možnosti, nastavení parametrů...
- Příklady – Problém obchodního cestujícího, Problém batohu.
- Matematika se zajímá spíše počty těchto objektů, zatímco informatika hledá algoritmy jak je generovat.
- Počet těchto objektů roste exponenciálně nebo i rychleji!

Generování permutací

- Budeme generovat permutace celých čísel $1, 2, \ldots, n$.
- Obecněji můžeme generovat permutací prvků $\{a_1, a_2, \ldots, a_n\}$.
- Využití strategie sniž a vyřeš:
  1. Generování $n!$ permutací pro n prvků převedeme na generování $(n-1)!$ permutací $n-1$ prvků.
  2. Jakmile máme problém pro $n-1$ vyřešen, vložíme prvek n na všech n možných pozic do každé z $(n-1)!$ permutací.
  3. Jinak řečeno, máme $(n-1)!$ permutací, ke každé z nich vygenerujeme n dalších. Celkově dostáváme $n(n-1)! = n!$ permutací.

Generování permutací – příklad

permutace prvku 1	1		
vložení 2 do permutace 1 zprava doleva	1**2**	**2**1	
vložení 3 do permutace 12 zprava doleva	12**3**	1**3**2	**3**12
vložení 3 do permutace 21 zleva doprava	**3**21	2**3**1	21**3**

Co je z příkladu patrné?

- Všechny permutace jsou navzájem různé.
- Minimální změna mezi permutacemi – dvě po sobě jdoucí permutace se liší výměnou jediné dvojice prvků a to dokonce sousedních prvků.

Johnson-Trotterův algoritmus

- Existuje možnost generovat permutace n prvků? Bez nutnosti vygenerovat permutace pro $n - 1$? Ano, existuje.
- Každému z n prvků permutace přiřadíme **šipku** (směr), buď vlevo nebo vpravo.
- O prvku k řekneme, že je v dané permutaci **mobilní**, jestliže sousední prvek ve směru šipky prvku k je menší než k.

Příklad

Permutace se šipkami

$$\vec{3}\ \overleftarrow{2}\ \vec{4}\ \vec{1}$$

Prvky 3 a 4 **jsou mobilní**, prvky 2 a 1 **nejsou mobilní**.

Johnson-Trotterův algoritmus

Vstup : Přirozené číslo n
Výstup: Seznam všech permutací čísel $\{1, \ldots, n\}$

1 $\pi \leftarrow \overleftarrow{1}\ \overleftarrow{2}\ \ldots\ \overleftarrow{n}$;
2 **while** π obsahuje mobilní prvek **do**
3 $\quad k \leftarrow$ největší mobilní prvek v π;
4 $\quad$ přehoď v π prvek k se sousedem ve směru šipky;
5 $\quad$ změň směr šipky u všech prvků větších než k;
6 $\quad$ vlož nově vytvořenou permutaci (krok 4) π do výsledného seznamu;
7 **end**
8 **return** seznam všech permutací;

Johnson-Trotterův algoritmus – příklad

Ukázka generování permutací pro $n = 3$

$$
\begin{array}{ccc}
\overleftarrow{1} & \overleftarrow{2} & \overleftarrow{3} \\
\overleftarrow{1} & \overleftarrow{3} & \overleftarrow{2} \\
\overleftarrow{3} & \overleftarrow{1} & \overleftarrow{2} \\
\overrightarrow{3} & \overleftarrow{2} & \overleftarrow{1} \\
\overleftarrow{2} & \overrightarrow{3} & \overleftarrow{1} \\
\overleftarrow{2} & \overleftarrow{1} & \overrightarrow{3}
\end{array}
$$

O prvku k řekneme, že je v dané permutaci **mobilní**, jestliže sousední prvek ve směru šipky prvku k je menší než k.

Johnson-Trotterův algoritmus – příklad, $n = 4$

$\overleftarrow{1}\,\overleftarrow{2}\,\overleftarrow{3}\,\overleftarrow{4}$
$\overleftarrow{1}\,\overleftarrow{2}\,\overleftarrow{4}\,\overleftarrow{3}$
$\overleftarrow{1}\,\overleftarrow{4}\,\overleftarrow{2}\,\overleftarrow{3}$
$\overleftarrow{4}\,\overleftarrow{1}\,\overleftarrow{2}\,\overleftarrow{3}$
$\overrightarrow{4}\,\overleftarrow{1}\,\overleftarrow{3}\,\overleftarrow{2}$
$\overleftarrow{1}\,\overrightarrow{4}\,\overleftarrow{3}\,\overleftarrow{2}$

$\overleftarrow{1}\,\overleftarrow{3}\,\overrightarrow{4}\,\overleftarrow{2}$
$\overleftarrow{1}\,\overleftarrow{3}\,\overleftarrow{2}\,\overrightarrow{4}$
$\overleftarrow{3}\,\overleftarrow{1}\,\overleftarrow{2}\,\overleftarrow{4}$
$\overleftarrow{3}\,\overleftarrow{1}\,\overleftarrow{4}\,\overleftarrow{2}$
$\overleftarrow{3}\,\overleftarrow{4}\,\overleftarrow{1}\,\overleftarrow{2}$
$\overleftarrow{4}\,\overleftarrow{3}\,\overleftarrow{1}\,\overleftarrow{2}$

$\overrightarrow{4}\,\overrightarrow{3}\,\overleftarrow{2}\,\overleftarrow{1}$
$\overrightarrow{3}\,\overrightarrow{4}\,\overleftarrow{2}\,\overleftarrow{1}$
$\overrightarrow{3}\,\overleftarrow{2}\,\overrightarrow{4}\,\overleftarrow{1}$
$\overrightarrow{3}\,\overleftarrow{2}\,\overleftarrow{1}\,\overrightarrow{4}$
$\overleftarrow{2}\,\overrightarrow{3}\,\overleftarrow{1}\,\overleftarrow{4}$
$\overleftarrow{2}\,\overrightarrow{3}\,\overleftarrow{4}\,\overleftarrow{1}$

$\overleftarrow{2}\,\overleftarrow{4}\,\overrightarrow{3}\,\overleftarrow{1}$
$\overleftarrow{4}\,\overleftarrow{2}\,\overrightarrow{3}\,\overleftarrow{1}$
$\overrightarrow{4}\,\overleftarrow{2}\,\overleftarrow{1}\,\overrightarrow{3}$
$\overleftarrow{2}\,\overrightarrow{4}\,\overleftarrow{1}\,\overrightarrow{3}$
$\overleftarrow{2}\,\overleftarrow{1}\,\overrightarrow{4}\,\overrightarrow{3}$
$\overleftarrow{2}\,\overleftarrow{1}\,\overrightarrow{3}\,\overrightarrow{4}$

Johnson-Trotterův algoritmus

- Jeden z nejefektivnějších algoritmů pro generování permutací.
- Časová složitost algoritmu je $\Theta(n!)$.
- „Děsivá“ složitost algoritmu ale není způsobena algoritmem samotným, ten pracuje velice rychle. Je zdpůsobena obrovským množstvím permutací, které musí vygenerovat...

Děkuji za pozornost