Vyčíslování chemických rovnic

Štěpán Hojdar

Zimní semestr 2014/2015, Karlova univerzita

Programování I (NPRG030)

1 Anotace

Program dostane chemickou rovnici a vrátí jí správně vyčíslenou. Vše probíhá řádkově ať už vlastnoručním zadáním nebo přesměrováním standardního vstupu (např. z textového souboru). Pokud rovnice vyčíslení nemá (to se může stát například pokud jsou vstupní data špatně) nebo ji vyčíslit pomocí soustavy rovnic nelze (i takovéto případy nastávají) vypíše program odpovídající chybou hlášku.

2 Přesné zadání

Cílem je vytvořit program, který dostane na standardním vstupu chemickou rovnici, převede ji do maticové podoby soustavy rovnic a poté se pomocí Gauss-Jordanovy eliminace pokusí tuto rovnici vyčíslit.

U některých rovnic je potřeba specifikovat, která část sloučeniny se nebude měnit, protože jinak má rovnice více různých řešení, proto program nebude omezen na známé prvky a bude možné část sloučeniny, která se změnit nemůže (např. se nerozpadne benzenové jádro), nahradit unikátním názvem. To nám pomůže eliminovat ostatní řešení. Tedy na příklad místo C_6H_5 můžeme psát *„Fenyl“*. Podotkněme, že takto „nově vytvořený prvek“ musí splňovat všechny podmínky vstupu (tedy *„Fenyl“*) začíná velkým písmenem apod.

Známý problém je, že některé rovnice se nedají Gaussovou eliminací vyčíslit jednoznačně, například rovnice

$$C + O_2 = CO + CO_2$$

nemá jednoznačné řešení. V takových případech program vrátí chybovou hlášku, že neexistuje jednoznačné řešení soustavy. Obecně je toto chápáno jako lepší než vypsat jakékoli vyčíslení, protože by bylo chybou domnívat se, že řešení je jedno, když jich je nekonečně mnoho.

Pro **tvar vstupu** platí následující pravidla:

- prvek začíná velkým písmenem, všechna ostatní jsou malá
  Ni je nikl, NI je dusík a jód
- číslo za prvkem udává počet, takže CO_2 znamená 1x C a 2x O
- pokud potřebujeme napsat $(CO)_2$ použijeme takto jednoduché závorky
- závorky můžeme skládat, například *Cu3(Fe(CN)6)2* znamená $Cu_3(Fe(CN)_6)_2$
- mezi reaktanty je +, levá a pravá strana oddělena =, konec rovnice je indikován tečkou

Například tedy:

```
Cu(NO3)2 + K3(Fe(CN)6) = Cu3(Fe(CN)6)2 + KNO3.
```

3 Zvolený algoritmus a jeho výběr

Vyčíslování probíhá pomocí skoro-úplné Gauss-Jordanovy eliminace. Není zcela úplná v tom smyslu, že nepřevádí pivoty do jednotkového tvaru. Toto je z důvodu numerické stability soustavy rovnic a snazší zpětné substituce, kde program musí najít nejmenší parametr takový, aby výsledná chemická rovnice byla co nejvíce pokrácena. To v případě celočíselné matice není problém (najde se nejmenší společný násobek pivotů), ale v případě matice po vydělení by to nebyl úkol snadný. Data programu jsou díky tomu také čitelnější, protože matice jsou celočíselné.

4 Program

Rozepišme běh programu krok po kroku:

Na začátku program ze standardního vstupu načte rovnici. Načte ji do spojového seznamu proměnných typu **string** reaktant po reaktantu (reaktant budeme používat jak pro sloučeniny na levé tak na pravé straně (tedy i produkty) pokud nebude řečeno jinak). Přitom spočítá počet velkých písmen čímž odhadne počet unikátních prvků (určitě jich nebude více než kolik je celkem prvků v rovnici, bude jich spíše k polovině, ale jak blízko k polovině již nemůžeme určit, proto bereme horní strop jako všechny).

Poté si dynamicky alokuje asociační pole velikosti odhadnutého počtu maximálních prvků a inicializuje ho pomlčkami. Toto je opět pole stringů, ve kterém na i-té pozici je asociace i-tého prvku. (Tedy pokud na třetí pozici je *Cu* znamená to, že prvek *Cu* má přidělené číslo tři).

Toto pole naplní tak, že bude prvek po prvku brát ze spojového seznamu stringy a z těch jednotlivě brát prvky, které nejprve zkusí najít v asociačním poli a v případě neúspěchu nový prvek asociuje. Toto je také dobrá příležitost spočítat skutečný počet prvků (neboť každý prvek asociujeme právě jednou).

Protože nyní známe skutečný počet prvků M i počet reaktantů N, můžeme vytvořit matici o rozměrech $M \times N$ a inicializovat ji nulami. Také si vytvoříme pole výsledků (solutions), do kterého uložíme vyčíslení rovnice.

Další krok programu bude naplnit matici čísly, vytvořit z chemické rovnice soustavu rovnic lineárních. To uděláme tak, že každý reaktant (string, prvek spojového seznamu) tokenizujeme a poté z vytvořených tokenů matici naplníme.

Tokenizace probíhá v několika krocích a poměrně složitě, rozvedeme ji tedy více. Máme tedy již vytvořený spojový seznam s našimi reaktanty. Půjdeme tedy reaktant po reaktantu a budeme je tokenizovat. Samotné tokeny budeme ukládat do zásobníku, protože sloučeninu musíme přečíst od konce a to z důvodu, že čísla udávající počet atomů/molekul ($CaCO_3$ míněna trojka) jsou až *za* prvkem, kdežto my bychom je ocenili *před* prvkem. Proto musíme číst zprava doleva. Nyní k významu jednotlivých tokenů, které vznikají:

- Kladné (a celé) číslo znamená prvek. Konkrétně prvek, který je pod tímto číslem asociován.
- Záporné a celé číslo znamená právě „naši trojku“ v $CaCO_3$.
- -32000 je magická konstanta a znamená token pro otevírající závorku.
- Podobně $+32000$ je magická konstanta tokenu pro zavírající závorku

Samotná tokenizace potom probíhá načítáním znaků a pomocí stavů. Stavy jsou takovéto a mají příslušné „kódy“:

- Nedělám nic – 0
- Načítám číslo – 1
- Načítám prvek – 2
- Načítám číslo za závorkou – 3

Poté co program načte celý element (prvek, číslo za závorkou/prvkem) přidá ho na zásobník a čte dál, dokud není konec.

Nyní máme připravenou matici soustavy a chceme ji vyřešit. Na to použijeme Gauss-Jordanovu eliminaci, jak již bylo popsáno výše. Tedy budeme provádět celočíselné úpravy, abychom se nemuseli potýkat s numerickou stabilitou a zlomky. Samotná eliminace tedy probíhá v následujících krocích. Zkontrolujeme, zda podmatice již není nulová a pokud ne, najdeme pivota, vyeliminujeme vše nad i pod pivotem.

Nyní máme matici ve tvaru diagonální matice s několika nebazickými sloupci. Důležité je si uvědomit, že nebazický sloupec musí být právě jeden. To z důvodu, který je uveden v zadání, některé rovnice vyčíslit pomocí soustavy lineárních rovnic nelze, ty pak nebudou mít ani jeden, tedy všechny sloupce budou bazické, tedy matice bude regulární a tedy má právě jedno nulové řešení, takové řešení se nám samozřejmě nelíbí. Stejně tak pokud mají nebazických sloupců více než jeden, tak rovnice má nekonečně lineárně nezávislých řešení (víc než jeden parametr) a potom program vypíše, že rovnice nemá *unikátní* řešení.

Pokud tedy soustava má právě jeden nebazický sloupec program pokračuje dál. Matice nemusí být zkrácená, a proto program pomocí Eukleidova algoritmu k nalezení největšího společného dělitele řádky pokrátí.

Dalším krokem je již zpětná substituce. Uvědomíme si, co vlastně v tomto kroku musíme udělat. Chceme celočíselné řešení a ještě ke všemu nejmenší, musíme tedy vhodně zvolit parametr. Parametr zvolíme jako nejmenší společný násobek pivotů, protože pokud bychom pivoty převedli na jedničky, odpovídala by čísla pivotů jmenovatelům zlomků v nebazickém sloupci. Nejmenší společný násobek nám zajistí, že výsledek bude pokrácený a celočíselný.

Nyní již jen dopočteme příslušným násobením řešení a vrátíme se z procedury. Poté vytiskneme výsledek.

5 Reprezentace vstupních dat a jejich příprava

Vstupní data jsou detailně popsaná v zadání na začátku dokumentu, nebudeme je tedy zde znovu přepisovat. Vstupními daty je tedy pouze jedna nevyčíslená chemická rovnice, kterou do programu můžeme vložit buď přesměrováním vstupu do textového souboru s rovnicí nebo manuálním přepsáním rovnice do programu. Pokud vkládáme textový soubor je nutné si uvědomit, že program se bude snažit vše v tomto textovém souboru interpretovat jako jednu chemickou rovnici dokud nenajde znak tečka, který rovnici ukončí. Je tedy jedno, jak je soubor formátovaný bílými znaky (každý reaktant na jedné řádce nebo celá rovnice na jedné).

6 Reprezentace výstupních dat a jejich interpretace

Výstupem programu je vyčíslená rovnice. V případě, že rovnice nemá žádné nenulové řešení, vypíše program nulové řešení a varování, že jiné než nenulové neexistuje. V případě, že program má více nezávislých řešení, vypíše program, že rovnice nemá unikátní řešení.

7 Průběh prací

Práce na programu byla přímočará, program jsem psal od začátku do konce tak, jak je popsaný ve čtvrtém oddílu dokumentace. Jeho dílčí algoritmy jako parsování, tokenizaci a eliminaci jsem několikrát v průběhu pozměnil, protože se nové řešení ukázalo jako elegantnější.

Programoval jsem odshora, navrhl procedury, poté je implementoval a nakonec spojil dohromady, což mi umožnilo ladit program část po části a to práci určitě usnadnilo.

8 Co nebylo doděláno

Ze zadání je splněno vše, nicméně program lze rozšířit tak, aby měl přívětivější uživatelské rozhraní (například grafické), uměl vyčíslovat i iontové rovnice nebo například uměl vyčíslovat redoxní reakce „klasickou“ cestou přes pomocné výpočty, které by mohl vypsat a pomoct tak uživateli při řešení těchto rovnic. Drobností, která by mohla vylepšit uživatelskou přívětivost je kontrola, zda prvky, které uživatel napsal existují (tím se vyhneme chybám jako H_2So_4).

9 Závěrečné slovo

Práce na programu mne bavila, vyskytovaly se v něm zdlouhavější části (jako například tokenizace) ale i zábavnější jako programování a ladění eliminace nebo zpětné substituce. Jsem rád, že jsem se program rozhodl rozdělit do jednotlivých unit, myslím, že to hodně usnadnilo orientaci v něm samotném. Úkol mi nepřišel ani příliš snadný tak, aby šel napsat najednou, ani příliš obtížný tak, abych si přišel, že jsem po dni práce neudělal žádný pokrok, což bylo velmi příjemné.