GRAFOVÉ ALGORITMY - POZNÁMKA K PŮLSEMESTRÁLNÍ ZKOUŠCE ANEB JAZYK MATEMATIKY PRO INFORMATIKA

Zbyněk Křivka

Protože jsem na poslední přednášce nestihl udělat poznámku o řešení některých příkladů z půlsemestrální zkoušky, tak písemnou verzi této poznámky zveřejňuji studentům na stránkách předmětu. Koncepce této poznámky nepředkládá celkové řešení vybraných příkladů, ale spíše se řídí poupraveným heslem "Chytrému napověz, líného kopni[1]."

Varování: *Tento text je trochu ukecaný!*

Příklad 1a, jedna ze skupin. Pokud má být neorientovaný graf G (s n vrcholy) **nutně** souvislý, tak to znamená, že není potenciálně nesouvislý. Pokud zkoumáme minimální dostatečný počet hran takového grafu, tak přemýšlíme, kolik hran nám nutně zaručí onu žádanou souvislost. Řešením je tedy spočítat počet hran úplného grafu s $n-1$ vrcholy a přičíst jednu hranu, která povede do n-tého vrcholu. Musíme tedy zkoumat nejhorší případ, kdy se budeme pokoušet způsobit, aby graf nebyl souvislý. Hledáme tedy počet hran, od kterého je souvislost zaručena a tedy graf je nutně souvislý.

Příklad 1b. Některé nerovnice byly uvedeny tak, že nebyly v DFS lese nebo BFS stromu splnitelné, což o grafu říkalo to, že nemůže existovat.

Příklad 1c. Otázka (či její modifikace): Formálně definujte multigraf, který obsahuje orientované či neorientované hrany, povoluje smyčky a ohodnocuje hrany reálným číslem. Pro připomenutí: Množinu reálných čísel značíme $\mathbb{R}$, množinu celých čísel $\mathbb{Z}$ a množinu přirozených čísel $\mathbb{N}$.

V této otázce se projevila nezkušenost studentů s matematickými definicemi a především nepochopení jazyka matematiky[2]. Na matematické zápisy (alespoň ve většině informatických předmětů) se lze dívat jako na "programovací" jazyk, který je silně typovaný a po částech funkcionální (viz funkcionální programování). Stejně jako programovací jazyky i jazyk matematiky má různé dialekty (definice, notace, konvence) pro různé oblasti využití (například běžné značení v logice je jiné než značení ve formálních jazycích). Stejně jako u programovacího jazyka jste schopni i u matematického zápisu celkem spolehlivě rozpoznat, s jakým dialektem se právě potýkáte. Každý symbol má svůj význam a pokud nemá, je třeba jej popsat. Buď matematicky nebo slovně. Popisujete-li slovně například množinu, tak bývá nezbytné uvést její konečnost nebo nekonečnost a z jakých prvků se skládá.

Definujeme-li nějakou složitější matematickou strukturu, která obsahuje více komponent, tak obyčejně využijeme (uspořádané) n-tice[3] (např. graf je dvojice nebo trojice či

[1] Líný stejně tento text číst nebude.

[2] či přesněji Jazyk matematického značení v teorii grafů (a v algebře)

[3] Ve skutečnosti se vytváří nová univerzální algebra, ale to se studentům neodvažujeme ani říkat.

případně čtveřice, konečný automat je pětice, gramatika je čtveřice). U n-tic (či sekvencí) na rozdíl od množin je důležité pořadí, takže je jasné, co je první komponenta, druhá komponenta atd.

Definice: Reálně ohodnocený multigraf obsahující orientované i neorientované hrany a povolující smyčky, je čtveřice $M = (V, E, \varphi, w)$, kde jednotlivé komponenty jsou popořadě:

- V je **konečná** množina vrcholů
- E je **konečná** množina návěští hran (či jen hran)
- φ je mapovací funkce
$$\varphi\colon E \to \binom{V}{2} \cup V^2 \cup V$$
- w je váhová funkce $w\colon E \to \mathbb{R}$.

Komentář k definici: Tento složitější graf jsme definovali jako čtveřici (kulaté závorky). Pokud nechceme využít v naší definici pojem multimnožina (který bychom pak museli zadefinovat), tak potřebujeme kličku ve formě unikátních "jmen" hran, kterým budeme jednoznačně přiřazovat incidentní vrcholy pomocí mapovací funkce φ. Funkce φ každé hraně jednoznačně přiřazuje incidentní vrcholy nebo vrchol. Binární operátor množinového sjednocení $\cup$ rozděluje obor hodnot na tři podčásti. $\binom{V}{2}$ je matematický zápis pro dvouprvkové podmnožiny množiny vrcholů. V tomto případě pak mapovací funkce modeluje neorientovanou hranu mezi dvěma různými vrcholy. Část V^2 je zápis pro uspořádanou dvojici prvků z V a slouží pro modelování orientovaných hran včetně orientovaných smyček. Přiřadím-li nějaké hraně (resp. jménu hrany) pouze jediný vrchol (část V), tak modeluji neorientované smyčky. Nakonec poznamenejme, že toto samozřejmě není jediná definice, ale jedna z mnoha možných intuitivních definic.

Postesknutí a příslib světlejších zítřků na závěr: Provázání obecných matematických znalostí z prvních ročníků bakalářského studia a jejich vesměs informatických aplikací v pozdějším studiu není dobré podceňovat, protože to se pak podepisuje na nepochopení matematických zápisů a samozřejmě také neschopnosti tvořit vlastní správné matematické zápisy. Příští rok se tomuto provázání zkusím více věnovat třeba i na úkor příkladů obsahujících důkazy.

V jazyku matematiky se dá programovat úplně stejně jako v jiném programovacím jazyce. Jsou to přece oba jazyky formální. Abych byl ještě více kontroverzní, tak mohu tvrdit, že libovolný programovací jazyk je jenom dialektem jazyka matematiky!

Příklad 2a. Pokud máte formálně definovat graf a graf je definován jako dvojice, tak se očekává, že zadefinujete dvojici a popíšete formálně její komponenty. Řešení najdete přímo v přednáškách (definice stromu prohledávání do šířky a lesa prohledávání do hloubky).

Příklad 2b. Pokud máte zapsat algoritmus a pak pomocí tohoto algoritmu řešit zadaný problém, je podivné, pokud jej nakonec řešíte podle jiného algoritmu. Memorování celých algoritmů zpaměti bez jejich pochopení je pak dost poznat.

Dále bych poznamenal, že pokud popisujete algoritmus, tak je dobré uvést jeho vstupy a výstupy (nebo je okomentovat). V přednáškách tuto zvyklost sice často opomíjíme, ale tam bývá vstup popsán alespoň v rámci úvodních snímků dané přednášky.

Příklad 3a. Pojmy jsou definovány na přednášce, ale bylo samozřejmě možné odpovědět i vlastní smysluplnou definicí s využitím intuice. Měla být však opět formální, tj. (souvislá nebo silně souvislá) komponenta je množina vrcholů splňující jisté vlastnosti. Vlastnosti samy o sobě však nejsou komponentou[4]!

Příklad 3b. Některé skupiny měly vytvořit algoritmus, který využíval již vypočtený DFS les. Klíčovým úkolem bylo nalézt zpětnou hranu. Pro každou hranu $(u,v) \in E$ pak bylo možno využít jednoduchou klasifikaci na základě pole předchůdců `pi` a polí `d` a `f`:

```
if pi[v] = u:
  return TREE_EDGE
else if d[u] < d[v]:
  return FORWARD_EDGE
else if f[u] < f[v]:
  return BACK_EDGE
else if f[v] < f[u]:
  return CROSS_EDGE
else
  return ERROR
```

Zapřemýšlejte, jak se tato klasifikace odvodila od klasifikace uvedené v přednáškách.

[4]Pokud bychom mluvili o množině vlastností a formální zápis tomu odpovídal, tak se již dá diskutovat.