KSI 2012/2013
Úloha 3-3: Farma

Jan Horáček
Gymnázium, Brno, Vídeňská 47; jan.horacek@seznam.cz

31. prosince 2012

1 Úvod

Děkuji orgům na tuto úlohu.

Zpočátku mi připadala velmi složitá a výsledný algoritmus jsem si nedokázal představit. Zejména jsem přemýšlel, odkud začít, když jsou si všechny vrcholy rovny. Pak se v mé hlavě zrodila myšlenka, která se nakonec ukázala jako velmi dobrým základem pro výsledné řešení.

2 Program

Dokonce jsem si napsal program, jehož základem je algoritmus popsaný níže. A tak se jednotivé počítače Ferdinadna proměnily ve vlákna stroje, na kterém program běžel.

Pro zajímavost ho přikldádam, ovšem spíše jako bonus, neb by se Vám nemusel líbit způsob, jakým je program narychlo napsán. Je psán v Delphi 2009.

3 Hlavní myšlenka

Tento algorimus se většinu času chová podobně, jako vlákno. Tedy neustále na něco čeká. V našem případě na příchozí zprávy.

Algoritmus je navrhnut tak, aby každým kabelem (hranou) prošla právě jedna zpráva variabilní délky. Ve vyjímečných stavech se může stát, že jedním kabelem půjdou zároveň 2 zprávy v jeden okamžik oběma směry. To ovšem níže popsaný algoritmus nijak neohrozí.

Hlavní myšlenkou je posílat si mezi jednotlivými vrcholy routovací tabulky. A to velice specifickým způsobem. Počítače, ze kterých vychází jen jeden kabel (tj. počítače, které jsou na okraji sítě - stromu), označme je a, odešlou zprávu vedlejšímu počítači, označme ho b. Tato zpráva bude obsahovat něco ve smyslu "Drahý b, existuji. S přáním hezkého dne, Tvůj a". Označme si tuto zprávu jako routovací tabulku.

Tj. počítače a informují o své existenci počítače b. Počítač b si zapamatuje, že za kabelem, ze kterého dostal tuto zprávu, existuje někde počítač a (v tomto případě hned na druhé straně kabelu). Tedy, pokud b přijde skutečná zpráva (už po vytvoření routovacích tabulek) pro a, pošle ji na tento kabel. Vrchol a si zapamatuje, že pokud mu přijde zpráva pro kohokoliv, pošle ji na vrchol, kam poslal routovací tabulku.

Tento algorimus zobecníme: pokud daný počítač přijal zprávy ze všech sousedních vrcholů kromě jednoho, pošle routovací tabulky tomuto vrcholu a zavolá *Stop*. Pokud počítač již přijal ze všech sousedních vrcholů zprávy, zavolá *Stop*. A protože krajní vrcholy sítě (stromu) mají jen jeden sousední vrchol, budou routovací tabulky posílat ihned.

Takováto routovací tabulka k odeslání se skládá ze všech routovacích tabulek, který daný počítač přijal s tím, že do tohoto seznamu ještě vepíše sebe. A samozřejmě si opět uloží, kam routovací tabulku odeslal. Tato tabulka, odesílaná na vrchol v, říká počítači na opačné straně kabelu vedoucího z vrcholu v, které všechny počítače se nacházejí za vrcholem, ze kterého zprávu přijal. A to přesně potřebujeme, to jsou naše routovací tabulky.

V celé síti se tedy šíří jakási řetězová zpráva, kterou započaly krajní počítače a ke které každý počítač vždy přidá informaci o tom, že existuje i on sám.

Jakmile zpráva projde všemi vrcholy, každý vrchol již zavolal *Stop* a routovací tabulky jsou kompletně vytvořené.

4 Popis řešení

Řešení této úlohy je psáno pseudokódem založeném na jazyce Object pascal.

Řešení je psáno jako algoritmus, který je nutné spustit na všech počítačích zároveň.

Následuje kompetní slovní popis řešení:

4.1 Jádro programu

Hlavní jádro programu vypadá takto:

```
nekolecny_cyklus:
 begin
   //cteni zprav
   ReadRouting();

   //pokud jsme prijmuli zpravy ze vsech vrholu, skoncime
   if (PoppedCnt = PortNum) then
      Stop();

   //pokud jsme dostali zpravy ze vsech vrcholu krome jednoho,
   //tak tam posleme routovaci tabulky
   if (PoppedCnt = PortNum-1) then
      SendRouting();
 end;
```

Listing 1: Hlavní cyklus

Provádění tohoto nekonečného cyklu bude samozřejmě ukončeno zavoláním *Stop*().

Pro dobrou představu funkce algorimu definujme: tento cyklus se nachází uvnitř třídy *TVrchol*, která má tuto privátní vlastnost:

1. *PoppedCnt* : *Integer*

   Při startu programu nastaveno na hodnotu 0. Udává, z kolika vrcholů jsme již přijmuli zprávu. Bude použito níže.

Pak definujme vlastnosti vrcholu takto:

1. $PortNum$

   Viz zadání.

2. ID

   Viz zadání.

3. $SentRT$

   Udává vrchol, na který jsme poslali routovací tabulku. Bude použito níže.

4. $RoutTables : array$

   Je pole o tolika prvcích, kolik je počet výstupů (hran) vycházejících z daného vrcholu. Zde jsou uloženy routovací tabulky daného vrcholu.

   Jeden element tohoto pole je pak množinou vrcholů.

   Pro lepší pochopení významu tohoto pole uvádím konrétní příklad využití: Počítač a má nyní sestavené routovací tabulky a dostane zprávu k odeslání pro počítač b. Počítač a tedy prohledá pole $RoutTables$, kde $RoutTables[v]$ udává množinu všech počítačů, které nasledující někde za kabelem vedoucím z vrcholu v. a tedy ví, že pokud najde počítač b v množině $RoutTables[v]$ ($b \in RoutTables[v]$), tak má zprávu pro počítač b poslat do vrcholu v. Ten už se o další routování postará, protože má své routovací tabulky.

4.2 Funkce ReadRouting

Funkce ReadRouting zajišťuje čtení příchozích zpráv a jejich ukládání do paměti, tedy někam do datové struktury $RoutTables$ a její tělo vypadá takto:

```
for i := vsechny vrcholy
 begin
  received := Get(i);
  if (prazdne(received)) then continue;

  //pokud nam prisla zprava, tak si ji ulozime
  RoutTables[i] = received;
  Self.PoppedCnt := Self.PoppedCnt + 1;
 end;//for i
```

Listing 2: Tělo funkce ReadRouting

Pro účely tohoto algoritmu definujme zprávu, která je uložená v proměnné *received* jako množinu vrcholů.

Tato funkce tedy přečte příchozí zprávy ze všech vrcholů a uloží si je. U tohoto algorimu prochází každou hranou právě jedna zpráva, ovšem proměnné délky. To také předpokládá výše zmíněná funkce *ReadRouting*. Jakmile je zpráva přečtena, dojde k imkrementaci hodnoty proměnné *PoppedCnt* a tím pádem k označení vrcholu i jako přečteného.

4.3 Funkce SendRouting

Tato funkce je vyvolána, jakmile jsou přečteny zprávy ze všech vrcholů kromě jednoho. Z toho plyne, že krajní vrcholy daného stromu odesílají zprávu ihned při první iteraci hlavního nekonečného cyklu. A to je žádoucí, protože to zajistí první impulz. Vniřní vrcholy pak čekají na zprávy, které jim někdy určitě přijdou, protože řetězová reakce byla již iniciována krajními vrcholy.

Tato funkce zajišťuje přeposlání routovacích tabulek ze všech vrcholů, kromě jednoho, právě na tento jeden vrchol.

```
i = NajdiNeprijatyVrchol();
if (neexistuje(i)) then Exit;

Send(i, RoutTables[]+self);
SentRT := i;
Stop();
```

Listing 3: Tělo funkce SendRouting

Funkce $NajdiNeprijatyVrchol$ vyhledá právě onen vrchol, ze kterého nebyla přijata zpráva. U této funkce je zajištěno, že se jedná o právě 1 vrchol (to zajišťuje podmínka v hlavním nekonečném cyklu). Tato funkce hledá vrchol na základě stavu $RoutTables[]$. Vrchol i, ze kterého zatím nebyla přijata zpráva, má totiž jako jediný množinu $RoutTables[i]$ prázdnou.

Po ověření existence takového vrcholu dojde k odeslání routovací tabulky.

Routovací tabulka je odeslána právě na onen vrchol i a obsahuje všechny routovací tabulky tohoto vrcholu (uložené v poli $RoutTables[]$) + sám sebe ($self$). Počítač na druhé straně kabelu i pak přijme zprávu, která mu říká, které všechny počítače se nachází za kabelem, ze kterého zprávu přijal. A to je žádoucí.

Po odeslání zprávy má počítač dokončenou práci, žádná další data přes něho nepotečou, tedy volá $Stop()$.

4.4 Odesílání zpráv v již funkční síti

Představmě si, že počítači a přijde zpráva s pro počítač b.

Počítač a projde své routovací tabulky $RoutTables$ pro každý vrchol a pokud najde shodu, pošle zprávu:

```
for i := vsechny vrcholy
 begin
  if (b in RoutTables[i]) then Send(i,s);
 end;
```

Listing 4: Odesílání zprávy

Pokud není cíl b zaznamenán v routovacích tabulkách (což se může zcela regulérně stát), pošle počítač a zprávu počítači, na který při routování poslal routovací tabulky. Tedy počítači $SentRT$. Počítač $SendRT$ se pak o zprávu podobným způsobem postará.

Možná ještě dodám, že při přijímání zpráv je potřeba ověřovat, zda-li to nejsem já, kdo je adresátem zprávy a v tom případě přeposílání ukončit a přečíst zprávu. Kdyby nedošlo k implemetaci této podmínky, může dojít k zacyklení zprávy v síti.

Toť vše.

5 Závěr

Po první zdánlivě funkční implementaci algoritmu jsem byl mile překvapen tím, že celý algoritmus funguje.

V ideálním stavu projde sítí celkem tolik zpráv, kolik je kabelů, v neideálním stavu o jednu zprávu víc.

Opravdu zajímavá úloha, snad je řešení správné.

Reference

Řešení bylo vytvořeno pouze autorovou vlastní silou - bez použití externích zdrojů.