Tema Prolog

Publicare tema: 06.05.2019
Last update: 18.05.2019 01:30 - checker & teste, trimiterea temei - paginile 5, 6

Responsabili

- Mihai Costea
- Alin Popa

Deadline

- Soft : **22.05.2019**
- Hard: **22.05.2019**

Introducere

Tema consta intr-o implementare simplificata a **Spanning Tree Protocol (STP)** [0]. STP este un protocol de retea (layer 2), distribuit, prin care se configureaza topologia unei retele (graf) de switch-uri astfel incat sa nu existe cicluri. Protocolul realizeaza asta prin eliminarea anumitor muchii, astfel incat graful de conexiuni intre switch-uri este transformat intr-un arbore. Fiecare muchie are asociat un cost (calculat in general pe baza bandwidth-ului suportat de conexiunea respectiva), iar conditia obligatorie este ca arborele rezultat sa contina drumul de cost minim din fiecare switch pana la un switch special din retea, numit radacina (root). In mod uzual, la defectarea unuia dintre switch-urile retelei sau a uneia dintre legaturi, STP reconfigureaza legaturile dintre switchuri astfel incat sa continue sa existe conexiune intre oricare doua terminale din retea, daca acest lucru se poate (dar acest comportament nu intra in scope-ul temei). Mai multe detalii la materia Retele Locale (RL - anul 3, semestrul I).

Enunt

Se da o retea (sub forma de **graf**).

- Fiecare nod are asociata o **prioritate** (un **numar natural pozitiv**)
- **Prioritatile** nodurilor sunt **unice**
- Fiecare **muchie** a grafului are un anumit **cost**
- **Costurile NU sunt unice**

Se cere un **arbore de acoperire** (un subgraf al grafului dat) si un **nod root** care sa respecte urmatoarele conditii:

- Nodul **root** va fi nodul **cu cea mai mica prioritate** dintre nodurile grafului dat
- Arborele trebuie **sa acopere toate nodurile** grafului dat
- Arborele trebuie sa contina **drumul de cost minim de la orice nod pana la nodul root**. Cu alte cuvinte, daca in graful original exista mai multe drumuri de la un nod oarecare X pana la nodul radacina R, iar aceste drumuri au suma costurilor muchiilor M=[CostDrum1, CostDrum2, CostDrum3, ...], atunci in arborele returnat costul drumului de la X la R trebuie sa fie egal cu min(M)
- In cazul in care dintr-un nod oarecare X se poate ajunge la nodul root R prin **mai multe drumuri de costuri egale, se va alege acel drum pentru care nodul imediat urmator are prioritatea mai mica**. Cu alte cuvinte, daca drumurile posibile de cost minim din X pana la R sunt: [[X,Y1,...,R], [X,Y2,...,R], [X,Y3,...,R], …] atunci drumul care va trebui sa se gaseasca in arbore este drumul i pentru care prioritate(Yi) este minima

Restrictii pentru output:

- Outputul este **nodul root R,** si o **lista de muchii L** din graful initial ce constituie arborele pe care l-ati construit
- Fiecare muchie trebuie data sub forma **[A,B], unde A este nodul mai apropiat de root, iar B este nodul mai departat de root (asa cum apar in arborele de acoperire)**. *Trivia: in terminologia STP, portul unui switch care duce spre nodul root se numeste “root port”; porturile care duc spre alte switchuri prin muchii ce au fost alese in arborele de acoperire final se numesc “designated ports”; porturile care duc spre muchii ce au fost eliminate se numesc “blocked ports”.*
- In cadrul listei L=[[A1,B1], [A2,B2], [A3,B3] …] nu conteaza ordinea in care sunt date muchiile
- Atentie - conform conditiilor impuse, arborele de acoperire cerut **este unic**

Cerinta

Scrieti un predicat: **`stp(Retea, Root, Edges)`** care primeste un graf ca prim parametru si intoarce nodul root si toate muchiile din arborele de acoperire a retelei.

Format

`Retea` are formatul urmator:

```
[
      [[indiceNod1, prioritate1], [indiceNod2, prioritate2], ...,
      [indiceNodN, prioritateN]],

      [[nodi1, nodj1, cost1], [nodi2, nodj2, cost2], ...,
      [nodiM, nodjM, costM]]
]
```

- **M** reprezinta numarul de muchii
- **N** reprezinta numarul de noduri.

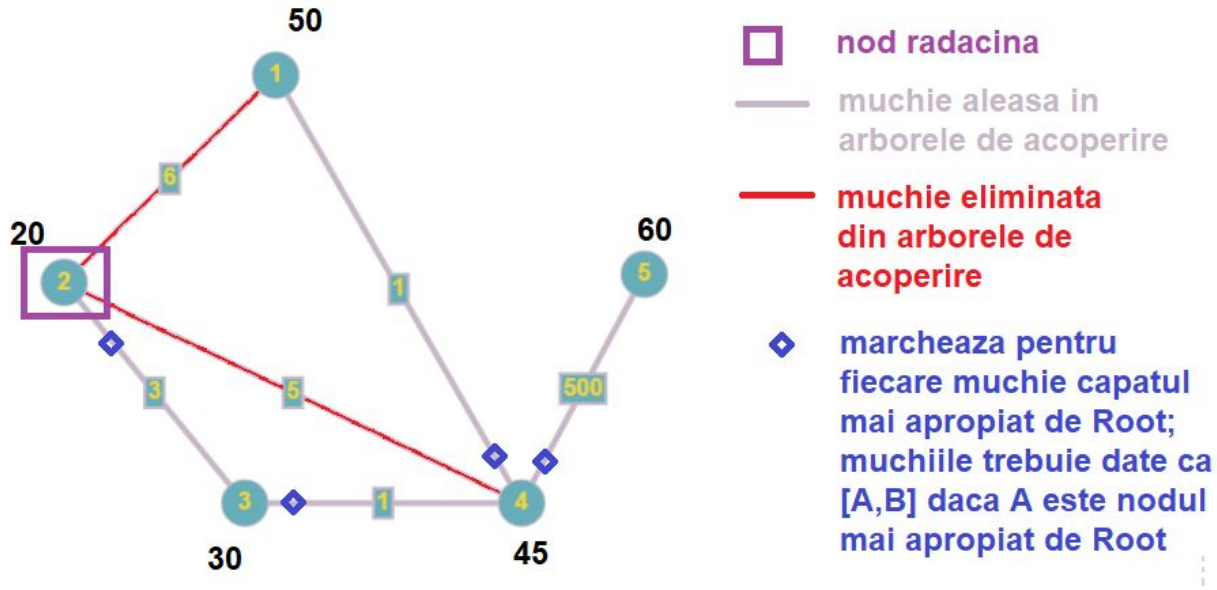
`Root` va fi unul din indicii nodurilor (e.g. `indiceNod2`)
`Edges` va fi o lista de muchii (e.g. `[[nodi3, nodj3], [nodj7, nodi7], ...]`) conform conditiilor date mai sus.

Restrictii

- Costurile **NU** vor fi negative
- Costurile **NU** vor fi 0
- Costurile muchiilor sunt **numere naturale**
- Indicii nodurilor sunt **numere naturale**
- Graful va fi mereu **conex**.
- Graful va avea mereu **N > 1** noduri
- Aveti voie sa folositi predicatele facute la curs si/sau laborator
- Recomandare pentru rezolvare: plecati din nodul root si adaugati progresiv noduri conexe la arbore, indeplinind conditiile impuse in enunt. Recomandare nr. 2: nu faceti brute force pe toti arborii de acoperire posibili. Vor fi si teste mari.

Exemplu

```
Retea = [[[1, 50], [2, 20], [3, 30], [4, 45], [5, 60]] ,[[1,2,
6],[2,3, 3], [3,4, 1], [4,5, 500], [2,4, 5], [1,4, 1]]]
stp(Retea, Root, Edges). =>
      Root = 2
      Edges = [[4,1], [2, 3], [3, 4], [4, 5]]
```



Bonus

Scrieti predicatul **`drum(Retea, Src, Dst, Root, Edges, Path)`** care intoarce arborele de acoperire prin **`Root`** si **`Edges`**, iar prin **`Path`** drumul parcurs de un "pachet" in retea intre nodurile **`Src`** si **`Dest`**.

- **`Retea,Root,Edges`** au acelasi format si sens ca mai sus
- **`Src`** si **`Dst`** sunt doua noduri din retea

```
drum([[1, 50], [2, 20], [3, 30], [4, 45], [5, 60]] ,[[1,2,
6],[2,3, 3], [3,4, 1], [4,5, 500], [2,4, 5], [1,4, 1]]], 5, 1,
Root, Edges, Path). =>
     Root = 2,
     Edges = [[4,1], [2, 3], [3, 4], [4, 5]],
     Path = [5, 4, 1].
```

Punctaj

- **90p teste automate**
- **5p lizibilitate cod**
- **5p fisier README (max 350 de cuvinte)**
- **25p bonus**

Checker local si teste locale

Arhiva cu checkerul si testele poate fi descarcata de pe platforma de curs. Pentru a rula checkerul trebuie sa instalati Python (orice versiune) [2] si swipl [3]. **Tema trebuie sa se numeasca `main.pl`.**

Quick guide pentru rularea checkerului

Pentru afisarea helperului din checker:

```
python checker.py -h
```

Pentru rularea tuturor testelor si afisarea punctajului obtinut:

```
python checker.py
```

Pentru rularea unui test anume:

```
python checker.py --testfile <path catre input> --reffile <path catre referinta>
```

e.g.: `python checker.py --testfile easy/in_easy4.txt --reffile easy/out_easy4.txt`

Pentru ca checkerul sa mearga este necesar sa aveti calea catre executabilul swipl in variabila de mediu PATH (google search: “add directory to PATH environment variable”). Cu alte cuvinte, in terminalul cu care veti rula checkerul, daca scrieti comanda

```
swipl
```

ar trebui sa vi se deschida interpretorul de prolog (in loc de eroarea de comanda necunoscuta).
Puteti sa evitati modificarea variabilei PATH specificand la rularea checkerului calea catre executabilul swipl prin parametrul --swiplexe (trebuie sa gasiti executabilul mai intai). Exemple:

linux: `python checker.py --swiplexe /usr/bin/swipl`
cmd: `python checker.py --swiplexe "\"C:\Program Files\swipl\bin\swipl.exe\""`
cygwin: `python checker.py --swiplexe "/cygdrive/c/Program\ Files/swipl/bin/swipl.exe"`

Structura testelor

Testele sunt impartite in doua categorii: easy (10 teste) si hard (70 de teste).
Pentru fiecare test, punctajul se acorda astfel:

- 0.3 (easy) / 0.1 (hard) - pentru gasirea nodului root corect
- 2.4 (easy) / 0.8 (hard) - pentru gasirea arborelui de acoperire cerut; se acorda doar daca nodul root gasit este cel corect
- 0.75 (easy) / 0.25 (hard) - pentru gasirea drumului cerut la bonus; se acorda numai daca root-ul si arborele au fost generate corect

Nota: checkerul va afisa “90%” pentru un test corect (respectiv “115%” daca ati implementat bonusul). Diferenta de 10% este pentru punctajul pe README si coding style.

Limita de timp

Vom rula testele cu limita de timp de 15 secunde per predicat (stp(...), drum(...)). Pentru a primi punctajul aferent este nevoie ca predicatul corespunzator sa se finalizeze in limita de timp. Atentie: checkerul local **nu** verifica timpul de rulare (cel putin, nu by default). Daca rulati checkerul pe Linux sau in Cygwin, puteti activa verificarea: editati fisierul checker.py, decomentati linia 58 si comentati linia 59

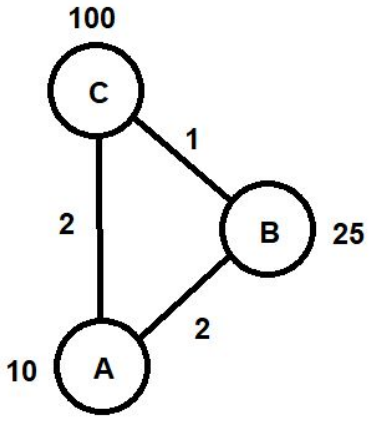
Trimiterea temei

Tema va fi incarcata pe platforma de curs sub forma unei arhive zip. Arhiva trebuie sa contina:

- fisier README
- tema, **pe care va trebui sa o denumiti main.pl**

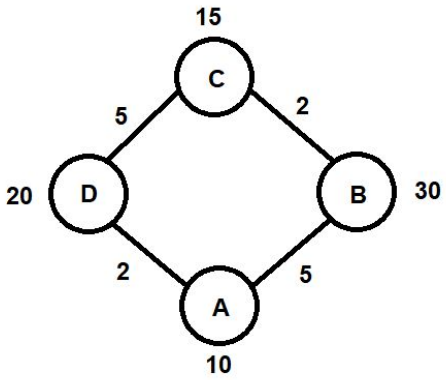
FAQ

1. **Exista checker si teste?**
   Da, ~~un checker local si niste teste locale vor fi publicate in curand.~~
2. **Muchiile ce fac parte din arbore trebuie luate astfel incat arborele de acoperire sa aiba cost total minim?**
   Nu neaparat. Suma costurilor muchiilor pentru fiecare drum de la un nod pana la Root trebuie sa fie minima. Acesta nu este obligatoriu arborele in care suma tuturor costurilor este minima.
   Considerati exemplul de mai jos. Nodul root va fi nodul A. Arborele de acoperire cerut care minimizeaza drumurile de la orice nod la nodul A sunt muchiile [A,B] si [A,C] => rezulta un arbore ce are suma costurilor egala cu 4.



3. **Daca exista drumuri de cost egal pana la un anumit nod, dupa ce criteriu alegem muchia?**

Dupa prioritatea mai mica a nodului conectat prin muchia respectiva.



In exemplul de mai sus, nodul root va fi A. Muchiile din arbore vor fi in primul rand [A,B] si [A,D]. La nodul C se poate ajunge atat din B, cat si din D - ambele drumuri au costuri egale, 7. In acest caz se alege muchia [D,C], deoarece D are prioritatea mai mica decat B (20<30).

4. **Pot obtine punctajul bonus daca nu rezolv restul cerintei?**
   Nu. Punctajul pe bonus se va aplica pentru fiecare test in parte - daca arborele generat este corect, se va verifica si bonusul, altfel nu.
5. **Ce trebuie sa scriu in README?**
   Orice crezi ca este util pentru cel care iti corecteaza tema (ce i-ai spune daca ar corecta cu tine de fata). Ce abordare ai avut la tema, ce ai incercat dar nu a mers, ce nu ai incercat dar a mers accidental, ce ti-a placut la tema, ce nu ti-a placut. Incearca sa nu depasesti limita de 350 de cuvinte (e o recomandare, nu facem enforce la asta). Este foarte util pentru noi sa mentionezi in README numele, grupa, si cat timp ai petrecut pentru aceasta tema (aproximativ, in ore).
6. **Am alta intrebare/nelamurire / am gasit un test gresit / nu inteleg ce trebuie sa fac.**
   Foloseste cu incredere forumul temei [1].

Referinte

[0] Spanning_Tree_Protocol
[1] https://acs.curs.pub.ro/2018/mod/forum/view.php?id=13269
[2] https://www.python.org/downloads/
[3] http://www.swi-prolog.org/download/stable