

Järjestäytyminen ja päivittäminen

Esa Elovaara

2. marraskuuta 2007

Määritelmä

Mallimuunnosalgoritmi simuloi hajautettua järjestelmää.

Esimerkkejä

- **Järjestäytyminen** simuloi tunnuksellista järjestelmää erikoisprosessorijärjestelmässä.
- **Päivittäminen** simuloi erikoisprosessorijärjestelmää tunnuksellisessa järjestelmässä.

Erikoisprosessorijärjestelmä

Määritelmä

- Yksi **erikoisprosessori**.
- Monta **perusprosessoria**.
- Perusprosessorit ovat anonyymejä, eli niillä ei ole tunnistettavia erityispiirteitä.

Huomioita

- Järjestelmässä on aina johtaja(erikoisprosessori).
- Algoritmien suunnitteleminen järjestelmälle on yksinkertaisempaa.

Tunnuksellinen järjestelmä

Määritelmä

- Jokaisella järjestelmän prosessorilla on **ainutkertainen tunnus**.
- Prosessorit eivät ole erikoisasemassa toisiinsa nähden.

Huomioita

Yleinen hajautettujen järjestelmien malli(verkkosovittimien MAC-osoite).

Mihin mallimuunnoksia tarvitaan?

Mallimuunnosten hyödyt

- Hajautettu algoritmi voidaan suunnitella toimimaan ratkaistavan ongelman kannalta yksinkertaisimmassa järjestelmässä.
- Olemassa olevaa ja toimivaksi tiedettyä koodia voidaan uudelleenkäyttää.

Erikoisprosessorijärjestelmä → tunnuksellinen järjestelmä

Algoritmi koostuu kolmesta osa-algoritmista.

- 1 Järjestelmän virittävän puun rakentaminen.
- 2 Järjestelmän prosessoreiden määrän laskeminen.
- 3 Järjestelmän prosessoreiden nimeäminen.

- Algoritmi olettaa verkon virittävän puun olevan tiedossa.
- Erikoisprosessori asetetaan virittävän puun juurisolmuksi.
- Jokainen verkon solmu pitää muistissa alipuunsa solmujen määrää.
- Solmu asettaa alipuunsa solmujen määräksi lastensa alipuiden solmujen määrän summan kasvatettuna yhdellä.

- Järjestelmä on turvallisessa konfiguraatiossa, kun jokainen solmu ilmoittaa alipuunsa solmujen lukumäärän oikein.
- Lehtisolmut asettavat solmujensa määräksi aina 1. Lehtisolmut ovat perusaskel todistettaessa algoritmin oikeellisuutta induktiolla.
- Järjestelmän suoritettua $k + 1$ sykliä, korkeudella $h \leq k$ olevat solmut ilmoittavat lukumääränsä oikein.
- Tällä kertaa ei tarvitse välittää kelluvista arvoista, koska korkeudella k oleva solmu tiedustelee lukumääriä vain lapsiltaan, joiden korkeus on tietysti pienempi kuin k .
- Varsinainen todistus löytyy kirjoitelmasta.

Laskeminen: erikoisproessori

```
1: while true do  
2:   sum := 0  
3:   for all  $P_j \in \text{children}(i)$  do  
4:      $lr_{ji} := \text{read}(r_{ji})$   
5:      $sum := sum + lr_{ji}.count$   
6:   end for  
7:    $count_i := sum + 1$   
8: end while
```

Laskeminen: erikoisproessori

```
1: while true do  
2:   sum := 0  
3:   for all  $P_j \in \text{children}(i)$  do  
4:     lrji := read(rji)  
5:     sum := sum + lrji.count  
6:   end for  
7:   countj := sum + 1  
8: end while
```

Laskeminen: erikoisproessori

```
1: while true do  
2:   sum := 0  
3:   for all  $P_j \in \text{children}(i)$  do  
4:     lrji := read(rji)  
5:     sum := sum + lrji.count  
6:   end for  
7:   counti := sum + 1  
8: end while
```

Laskeminen: erikoisproessori

```
1: while true do  
2:   sum := 0  
3:   for all  $P_j \in \text{children}(i)$  do  
4:     lrji := read(rji)  
5:     sum := sum + lrji.count  
6:   end for  
7:   counti := sum + 1  
8: end while
```

Laskeminen: erikoisproessori

```
1: while true do  
2:   sum := 0  
3:   for all  $P_j \in \text{children}(i)$  do  
4:      $lr_{ji} := \text{read}(r_{ji})$   
5:     sum := sum +  $lr_{ji}.\text{count}$   
6:   end for  
7:   countj := sum + 1  
8: end while
```

Laskeminen: perusprosessori

```
1: while true do  
2:   sum := 0  
3:   for all  $P_j \in \text{children}(i)$  do  
4:      $lr_{ji}$  := read( $r_{ji}$ )  
5:      $sum$  :=  $sum + lr_{ji}.count$   
6:   end for  
7:    $count_i$  :=  $sum + 1$   
8:   write  $r_{i,parent}.count := count_i$   
9: end while
```

Laskeminen: perusproessori

```
1: while true do  
2:   sum := 0  
3:   for all  $P_j \in \text{children}(i)$  do  
4:      $lr_{ji} := \text{read}(r_{ji})$   
5:      $sum := sum + lr_{ji}.count$   
6:   end for  
7:    $count_i := sum + 1$   
8:   write  $r_{i,parent}.count := count_i$   
9: end while
```

Laskeminen: perusprosessori

```
1: while true do  
2:   sum := 0  
3:   for all  $P_j \in \text{children}(i)$  do  
4:     lrji := read(rji)  
5:     sum := sum + lrji.count  
6:   end for  
7:   counti := sum + 1  
8:   write ri,parent.count := counti  
9: end while
```

Laskeminen: perusprosessori

```
1: while true do  
2:   sum := 0  
3:   for all  $P_j \in \text{children}(i)$  do  
4:      $lr_{ji}$  := read( $r_{ji}$ )  
5:     sum := sum +  $lr_{ji}.\text{count}$   
6:   end for  
7:    $\text{count}_i$  := sum + 1  
8:   write  $r_{i,\text{parent}}.\text{count} := \text{count}_i$   
9: end while
```

Laskeminen: perusprosessori

```
1: while true do  
2:   sum := 0  
3:   for all  $P_j \in \text{children}(i)$  do  
4:      $lr_{ji}$  := read( $r_{ji}$ )  
5:     sum := sum +  $lr_{ji}.\text{count}$   
6:   end for  
7:   countj := sum + 1  
8:   write  $r_{i,\text{parent}}.\text{count} := \text{count}_j$   
9: end while
```

Laskeminen: perusprosessori

```
1: while true do  
2:   sum := 0  
3:   for all  $P_j \in \text{children}(i)$  do  
4:      $lr_{ji} := \text{read}(r_{ji})$   
5:      $sum := sum + lr_{ji}.count$   
6:   end for  
7:    $count_i := sum + 1$   
8:   write  $r_{i,parent}.count := count_i$   
9: end while
```

- Algoritmi olettaa verkon virittävän puun olevan tiedossa.
- Algoritmi olettaa verkon virittävän puun ja sen alipuiden solmujen lukumäärän olevan tiedossa.
- Jos verkon solmujen määrä on n , algoritmi määrää jokaiselle solmulle tunnuksen väliltä $[1 \dots n]$

- Jokainen solmu j saa vanhemmaltaan tunnuksen id . Solmun vanhempi takaa, että tunnukset $[id \dots id + s]$ ovat vapaana. s on solmun j alipuun solmujen lukumäärä.
- Järjestelmä on turvallisessa konfiguraatiossa, kun solmun tunnus on solmun lasten tunnuksia pienempi ja käyntijärjestyksessä edellä olevan lapsen tunnus on järjestyksessä jäljempänä olevan lapsen tunnusta pienempi.

Nimeäminen: erikoisproessori

```
1: while true do  
2:    $ID_i := 1$   
3:    $sum := 0$   
4:   for all  $P_j \in children(i)$  do  
5:      $lr_{ji} := \mathbf{read}(r_{ji})$   
6:     write  $r_{ij}.identifier := ID_i + 1 + sum$   
7:      $sum := sum + lr_{ji}.count$   
8:   end for  
9: end while
```

Nimeäminen: erikoisproessori

```
1: while true do  
2:    $ID_i := 1$   
3:    $sum := 0$   
4:   for all  $P_j \in children(i)$  do  
5:      $lr_{ji} := \text{read}(r_{ji})$   
6:     write  $r_{ij}.identifier := ID_i + 1 + sum$   
7:      $sum := sum + lr_{ji}.count$   
8:   end for  
9: end while
```

Nimeäminen: erikoisproessori

```
1: while true do  
2:    $ID_i := 1$   
3:    $sum := 0$   
4:   for all  $P_j \in children(i)$  do  
5:      $lr_{ji} := \mathbf{read}(r_{ji})$   
6:     write  $r_{ij}.identifier := ID_i + 1 + sum$   
7:      $sum := sum + lr_{ji}.count$   
8:   end for  
9: end while
```

Nimeäminen: erikoisproessori

```
1: while true do  
2:    $ID_i := 1$   
3:    $sum := 0$   
4:   for all  $P_j \in children(i)$  do  
5:      $lr_{ji} := \mathbf{read}(r_{ji})$   
6:      $\mathbf{write} r_{ij}.identifier := ID_i + 1 + sum$   
7:      $sum := sum + lr_{ji}.count$   
8:   end for  
9: end while
```

Nimeäminen: erikoisproessori

```
1: while true do  
2:    $ID_i := 1$   
3:    $sum := 0$   
4:   for all  $P_j \in children(i)$  do  
5:      $lr_{ji} := \text{read}(r_{ji})$   
6:     write  $r_{ij}.identifier := ID_i + 1 + sum$   
7:      $sum := sum + lr_{ji}.count$   
8:   end for  
9: end while
```

Nimeäminen: erikoisproessori

```
1: while true do  
2:    $ID_i := 1$   
3:    $sum := 0$   
4:   for all  $P_j \in children(i)$  do  
5:      $lr_{ji} := \mathbf{read}(r_{ji})$   
6:     write  $r_{ij}.identifier := ID_i + 1 + sum$   
7:      $sum := sum + lr_{ji}.count$   
8:   end for  
9: end while
```

Nimeäminen: perusprosessori

```
1: while true do  
2:   sum := 0  
3:    $lr_{parent,i}.identifier := \mathbf{read}(r_{parent,i})$   
4:    $ID_i := lr_{parent,i}.identifier$   
5:   for all  $P_j \in children(i)$  do  
6:      $lr_{ji} := \mathbf{read}(r_{ji})$   
7:     write  $r_{ij}.identifier := ID_i + 1 + sum$   
8:      $sum := sum + lr_{ji}.count$   
9:   end for  
10: end while
```

Nimeäminen: perusprosessori

```
1: while true do  
2:    $sum := 0$   
3:    $lr_{parent,i}.identifier := \mathbf{read}(r_{parent,i})$   
4:    $ID_i := lr_{parent,i}.identifier$   
5:   for all  $P_j \in children(i)$  do  
6:      $lr_{ji} := \mathbf{read}(r_{ji})$   
7:     write  $r_{ij}.identifier := ID_i + 1 + sum$   
8:      $sum := sum + lr_{ji}.count$   
9:   end for  
10: end while
```

Nimeäminen: perusproessori

```
1: while true do  
2:    $sum := 0$   
3:    $lr_{parent,i}.identifier := \mathbf{read}(r_{parent,i})$   
4:    $ID_i := lr_{parent,i}.identifier$   
5:   for all  $P_j \in children(i)$  do  
6:      $lr_{ji} := \mathbf{read}(r_{ji})$   
7:     write  $r_{ij}.identifier := ID_i + 1 + sum$   
8:      $sum := sum + lr_{ji}.count$   
9:   end for  
10: end while
```

Nimeäminen: perusprosessori

```
1: while true do  
2:    $sum := 0$   
3:    $lr_{parent,i}.identifier := \mathbf{read}(r_{parent,i})$   
4:    $ID_i := lr_{parent,i}.identifier$   
5:   for all  $P_j \in children(i)$  do  
6:      $lr_{ji} := \mathbf{read}(r_{ji})$   
7:     write  $r_{ij}.identifier := ID_i + 1 + sum$   
8:      $sum := sum + lr_{ji}.count$   
9:   end for  
10: end while
```

Nimeäminen: perusprosessori

```
1: while true do  
2:    $sum := 0$   
3:    $lr_{parent,i}.identifier := \mathbf{read}(r_{parent,i})$   
4:    $ID_i := lr_{parent,i}.identifier$   
5:   for all  $P_j \in children(i)$  do  
6:      $lr_{ji} := \mathbf{read}(r_{ji})$   
7:     write  $r_{ij}.identifier := ID_i + 1 + sum$   
8:      $sum := sum + lr_{ji}.count$   
9:   end for  
10: end while
```

Nimeäminen: perusprosessori

```
1: while true do  
2:    $sum := 0$   
3:    $lr_{parent,i}.identifier := \mathbf{read}(r_{parent,i})$   
4:    $ID_i := lr_{parent,i}.identifier$   
5:   for all  $P_j \in children(i)$  do  
6:      $lr_{ji} := \mathbf{read}(r_{ji})$   
7:     write  $r_{ij}.identifier := ID_i + 1 + sum$   
8:      $sum := sum + lr_{ji}.count$   
9:   end for  
10: end while
```

Nimeäminen: perusprosessori

```
1: while true do  
2:    $sum := 0$   
3:    $lr_{parent,i}.identifier := \mathbf{read}(r_{parent,i})$   
4:    $ID_i := lr_{parent,i}.identifier$   
5:   for all  $P_j \in children(i)$  do  
6:      $lr_{ji} := \mathbf{read}(r_{ji})$   
7:      $\mathbf{write} r_{ij}.identifier := ID_i + 1 + sum$   
8:      $sum := sum + lr_{ji}.count$   
9:   end for  
10: end while
```

Tunnuksellinen järjestelmä \rightarrow erikoisproessorijärjestelmä

- Päivitysalgoritmit jakavat järjestelmän prosessoreille tietoa järjestelmän topologiasta.
- Tässä esitelty algoritmi pitää jokaisessa prosessorissa P_i yllä joukkoa $counts_i$, jossa on tieto jokaisen samassa kytketyssä komponentissa olevan prosessorin etäisyydestä prosessoriin P_i .
- $counts_i$ joukon avulla voidaan **valita johtava** ja rakentaa verkon virittävä puu, jonka juureksi voidaan valita mikä tahansa verkon solmuista.

- $ReadSet_i = \{\langle 2, 1 \rangle, \langle 2, 3 \rangle, \langle 2, 3 \rangle\}$
 $ReadSet_i \setminus \setminus NotMinDist(P_2, ReadSet_i) = \{\langle 2, 1 \rangle\}$
- $ReadSet_i = \{\langle 2, 1 \rangle, \langle 3, 2 \rangle, \langle 4, 5 \rangle, \langle 4, 5 \rangle\}$
 $ConPrefix(ReadSet_i) = \{\langle 2, 1 \rangle, \langle 3, 2 \rangle\}$

```
while true do  
  ReadSeti := ∅  
  for all Pj ∈ N(i) do  
    ReadSeti := ReadSeti ∪ read(Processorsi)  
  end for  
  ReadSeti := ReadSeti \ \⟨i, *⟩  
  ReadSeti := ReadSeti + +⟨*, 1⟩  
  ReadSeti := ReadSeti ∪ {⟨i, 0⟩}  
  for all Pj ∈ processors(ReadSeti) do  
    ReadSeti := ReadSeti \ \NotMinDist(Pj, ReadSeti)  
  end for  
  write Processorsi := ConPrefix(ReadSeti)  
end while
```

```
while true do  
   $ReadSet_i := \emptyset$   
  for all  $P_j \in N(i)$  do  
     $ReadSet_i := ReadSet_i \cup \text{read}(Processors_j)$   
  end for  
   $ReadSet_i := ReadSet_i \setminus \langle i, * \rangle$   
   $ReadSet_i := ReadSet_i + \langle *, 1 \rangle$   
   $ReadSet_i := ReadSet_i \cup \{ \langle i, 0 \rangle \}$   
  for all  $P_j \in \text{processors}(ReadSet_i)$  do  
     $ReadSet_i := ReadSet_i \setminus \text{NotMinDist}(P_j, ReadSet_i)$   
  end for  
   $write Processors_i := \text{ConPrefix}(ReadSet_i)$   
end while
```

```
while true do
  ReadSeti := ∅
  for all Pj ∈ N(i) do
    ReadSeti := ReadSeti ∪ read(Processorsj)
  end for
  ReadSeti := ReadSeti \ \⟨i, *⟩
  ReadSeti := ReadSeti + +⟨*, 1⟩
  ReadSeti := ReadSeti ∪ {⟨i, 0⟩}
  for all Pj ∈ processors(ReadSeti) do
    ReadSeti := ReadSeti \ \NotMinDist(Pj, ReadSeti)
  end for
  write Processorsi := ConPrefix(ReadSeti)
end while
```

```
while true do
  ReadSeti := ∅
  for all Pj ∈ N(i) do
    ReadSeti := ReadSeti ∪ read(Processorsj)
  end for
  ReadSeti := ReadSeti \ \⟨i, *⟩
  ReadSeti := ReadSeti + +⟨*, 1⟩
  ReadSeti := ReadSeti ∪ {⟨i, 0⟩}
  for all Pj ∈ processors(ReadSeti) do
    ReadSeti := ReadSeti \ \NotMinDist(Pj, ReadSeti)
  end for
  write Processorsi := ConPrefix(ReadSeti)
end while
```

```
while true do  
   $ReadSet_i := \emptyset$   
  for all  $P_j \in N(i)$  do  
     $ReadSet_i := ReadSet_i \cup \mathbf{read}(Processors_j)$   
  end for  
   $ReadSet_i := ReadSet_i \setminus \langle i, * \rangle$   
   $ReadSet_i := ReadSet_i + \langle *, 1 \rangle$   
   $ReadSet_i := ReadSet_i \cup \{ \langle i, 0 \rangle \}$   
  for all  $P_j \in \mathit{processors}(ReadSet_i)$  do  
     $ReadSet_i := ReadSet_i \setminus \langle NotMinDist(P_j, ReadSet_i) \rangle$   
  end for  
   $write Processors_i := ConPrefix(ReadSet_i)$   
end while
```

```
while true do
  ReadSeti := ∅
  for all Pj ∈ N(i) do
    ReadSeti := ReadSeti ∪ read(Processorsj)
  end for
  ReadSeti := ReadSeti \ \⟨i, *⟩
  ReadSeti := ReadSeti + +⟨*, 1⟩
  ReadSeti := ReadSeti ∪ {⟨i, 0⟩}
  for all Pj ∈ processors(ReadSeti) do
    ReadSeti := ReadSeti \ \NotMinDist(Pj, ReadSeti)
  end for
  write Processorsi := ConPrefix(ReadSeti)
end while
```

```
while true do  
   $ReadSet_i := \emptyset$   
  for all  $P_j \in N(i)$  do  
     $ReadSet_i := ReadSet_i \cup \mathbf{read}(Processors_j)$   
  end for  
   $ReadSet_i := ReadSet_i \setminus \langle i, * \rangle$   
   $ReadSet_i := ReadSet_i + \langle *, 1 \rangle$   
   $ReadSet_i := ReadSet_i \cup \{ \langle i, 0 \rangle \}$   
  for all  $P_j \in \mathit{processors}(ReadSet_i)$  do  
     $ReadSet_i := ReadSet_i \setminus \langle \mathit{NotMinDist}(P_j, ReadSet_i) \rangle$   
  end for  
   $write\ Processors_i := \mathit{ConPrefix}(ReadSet_i)$   
end while
```

```
while true do
  ReadSeti := ∅
  for all Pj ∈ N(i) do
    ReadSeti := ReadSeti ∪ read(Processorsj)
  end for
  ReadSeti := ReadSeti \ \⟨i, *⟩
  ReadSeti := ReadSeti + +⟨*, 1⟩
  ReadSeti := ReadSeti ∪ {⟨i, 0⟩}
  for all Pj ∈ processors(ReadSeti) do
    ReadSeti := ReadSeti \ \NotMinDist(Pj, ReadSeti)
  end for
  write Processorsi := ConPrefix(ReadSeti)
end while
```

```
while true do  
   $ReadSet_i := \emptyset$   
  for all  $P_j \in N(i)$  do  
     $ReadSet_i := ReadSet_i \cup \mathbf{read}(Processors_j)$   
  end for  
   $ReadSet_i := ReadSet_i \setminus \langle i, * \rangle$   
   $ReadSet_i := ReadSet_i + \langle *, 1 \rangle$   
   $ReadSet_i := ReadSet_i \cup \{ \langle i, 0 \rangle \}$   
  for all  $P_j \in \mathit{processors}(ReadSet_i)$  do  
     $ReadSet_i := ReadSet_i \setminus \langle \mathit{NotMinDist}(P_j, ReadSet_i) \rangle$   
  end for  
   $\mathbf{write} Processors_i := \mathbf{ConPrefix}(ReadSet_i)$   
end while
```

Virittävän puun rakentaminen

- Erikoisprosessori asetetaan puun juurisolmuksi.
- Jokainen verkon solmu yrittää selvittää etäisyytensä juurisolmusta kysymällä naapureiltaan niiden (olettua) etäisyyttä juurisolmusta.
- Jos solmun i naapureista pienimmän etäisyyden ilmoittanut solmu on j ja sen etäisyys d , valitsee i vanhemmakseen solmun j ja asettaa omaksi etäisyydekseen $d + 1$.
- Juurisolmun etäisyys itsestään on 0, joten juurisolmu ei tiedustele naapureidensa etäisyyksiä, vaan ilmoittaa etäisyydekseen aina 0.
- Järjestelmä on turvallisessa tilassa, kun jokainen solmu ilmoittaa etäisyytensä oikein.

Virittävä puu: erikoisprosessori

```
while true do  
  for  $m := 1$  to  $\delta$  do  
    write  $r_{im} := \langle 0, 0 \rangle$   
  end for  
end while
```

Virittävä puu: erikoisprosessori

```
while true do
  for  $m := 1$  to  $\delta$  do
    write  $r_{im} := \langle 0, 0 \rangle$ 
  end for
end while
```

Virittävä puu: perusproessori

```
while true do  
  for  $m := 1$  to  $\delta$  do  
     $lr_{mi} := \text{read}(r_{mi})$   
  end for  
   $FirstFound := \text{false}$   
   $dist := 1 + \min\{lr_{mi}.dis \mid 1 \leq m \leq \delta\}$   
  for  $m := 1$  to  $\delta$  do  
    if not  $FirstFound$  and  $lr_{mi}.dis = dist - 1$  then  
      write  $r_{im} := \langle 1, dist \rangle$   
       $FirstFound := \text{true}$   
    else  
      write  $r_{im} := \langle 0, dist \rangle$   
    end if  
  end for  
end while
```

Virittävä puu: perusproessori

```
while true do
  for  $m := 1$  to  $\delta$  do
     $lr_{mi} := \text{read}(r_{mi})$ 
  end for
   $FirstFound := false$ 
   $dist := 1 + \min\{lr_{mi}.dis \mid 1 \leq m \leq \delta\}$ 
  for  $m := 1$  to  $\delta$  do
    if not  $FirstFound$  and  $lr_{mi}.dis = dist - 1$  then
      write  $r_{im} := \langle 1, dist \rangle$ 
       $FirstFound := true$ 
    else
      write  $r_{im} := \langle 0, dist \rangle$ 
    end if
  end for
end while
```

Virittävä puu: perusproessori

```
while true do
  for  $m := 1$  to  $\delta$  do
     $lr_{mi} := \text{read}(r_{mi})$ 
  end for
  FirstFound := false
   $dist := 1 + \min\{lr_{mi}.dis \mid 1 \leq m \leq \delta\}$ 
  for  $m := 1$  to  $\delta$  do
    if not FirstFound and  $lr_{mi}.dis = dist - 1$  then
      write  $r_{im} := \langle 1, dist \rangle$ 
      FirstFound := true
    else
      write  $r_{im} := \langle 0, dist \rangle$ 
    end if
  end for
end while
```

Virittävä puu: perusproessori

```
while true do
  for  $m := 1$  to  $\delta$  do
     $lr_{mi} := \text{read}(r_{mi})$ 
  end for
   $FirstFound := false$ 
   $dist := 1 + \min\{lr_{mi}.dis \mid 1 \leq m \leq \delta\}$ 
  for  $m := 1$  to  $\delta$  do
    if not  $FirstFound$  and  $lr_{mi}.dis = dist - 1$  then
      write  $r_{im} := \langle 1, dist \rangle$ 
       $FirstFound := true$ 
    else
      write  $r_{im} := \langle 0, dist \rangle$ 
    end if
  end for
end while
```

Virittävä puu: perusprosessori

```
while true do  
  for  $m := 1$  to  $\delta$  do  
     $lr_{mi} := \text{read}(r_{mi})$   
  end for  
   $FirstFound := \text{false}$   
   $dist := 1 + \min\{lr_{mi}.dis \mid 1 \leq m \leq \delta\}$   
  for  $m := 1$  to  $\delta$  do  
    if not  $FirstFound$  and  $lr_{mi}.dis = dist - 1$  then  
      write  $r_{im} := \langle 1, dist \rangle$   
       $FirstFound := \text{true}$   
    else  
      write  $r_{im} := \langle 0, dist \rangle$   
    end if  
  end for  
end while
```

Virittävä puu: perusproessori

```
while true do  
  for  $m := 1$  to  $\delta$  do  
     $lr_{mi} := \text{read}(r_{mi})$   
  end for  
   $FirstFound := \text{false}$   
   $dist := 1 + \min\{lr_{mi}.dis \mid 1 \leq m \leq \delta\}$   
  for  $m := 1$  to  $\delta$  do  
    if not  $FirstFound$  and  $lr_{mi}.dis = dist - 1$  then  
      write  $r_{im} := \langle 1, dist \rangle$   
       $FirstFound := \text{true}$   
    else  
      write  $r_{im} := \langle 0, dist \rangle$   
    end if  
  end for  
end while
```

Virittävä puu: perusproessori

```
while true do  
  for  $m := 1$  to  $\delta$  do  
     $lr_{mi} := \text{read}(r_{mi})$   
  end for  
   $FirstFound := false$   
   $dist := 1 + \min\{lr_{mi}.dis \mid 1 \leq m \leq \delta\}$   
  for  $m := 1$  to  $\delta$  do  
    if not  $FirstFound$  and  $lr_{mi}.dis = dist - 1$  then  
      write  $r_{im} := \langle 1, dist \rangle$   
       $FirstFound := true$   
    else  
      write  $r_{im} := \langle 0, dist \rangle$   
    end if  
  end for  
end while
```

Virittävä puu: perusproessori

```
while true do  
  for  $m := 1$  to  $\delta$  do  
     $lr_{mi} := \mathbf{read}(r_{mi})$   
  end for  
   $FirstFound := false$   
   $dist := 1 + \min\{lr_{mi}.dis \mid 1 \leq m \leq \delta\}$   
  for  $m := 1$  to  $\delta$  do  
    if not  $FirstFound$  and  $lr_{mi}.dis = dist - 1$  then  
      write  $r_{im} := \langle 1, dist \rangle$   
       $FirstFound := true$   
    else  
      write  $r_{im} := \langle 0, dist \rangle$   
    end if  
  end for  
end while
```

Virittävä puu: perusproessori

```
while true do  
  for  $m := 1$  to  $\delta$  do  
     $lr_{mi} := \text{read}(r_{mi})$   
  end for  
   $FirstFound := \text{false}$   
   $dist := 1 + \min\{lr_{mi}.dis \mid 1 \leq m \leq \delta\}$   
  for  $m := 1$  to  $\delta$  do  
    if not  $FirstFound$  and  $lr_{mi}.dis = dist - 1$  then  
      write  $r_{im} := \langle 1, dist \rangle$   
       $FirstFound := \text{true}$   
    else  
      write  $r_{im} := \langle 0, dist \rangle$   
    end if  
  end for  
end while
```