TC

ONDOKUZ MAYIS ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİL465 BİLGİSAYAR AĞ YÖNETİMİ LABORATUVARI DERSİ

BÜTÜNLEME PROJESİ

Projenin veriliş tarihi: 18.01.2019

Projenin son teslim tarihi: 30.01.2019 Saat 15.00'a kadar

Proje sunumuna gelirken uygulamanızın çalışan halini göstermeniz yeterli olacaktır.

Asenkron BFS (Breadth First Search) Oluşturma Algoritması

Herhangi bir *G* çizgesinden BFS ağacını oluşturan algoritma, *Update_BFS* adı verilen Bellman-Ford algoritmasının dağıtık başka bir çeşididir. Daha önce olduğu gibi tek bir başlatıcı düğüm (node) var, bu da komşu düğümlerle mesafe bilgisini içeren *layer(l)* mesajıdır. Bir *layer(l)* mesajını alan herhangi bir düğüm, mesajdaki *l* katman değerini bilinen uzaklığı ile kök (root) düğüm ile karşılaştırır ve eğer yeni değer daha küçükse, katman mesajının göndereni yeni ebeveyn olarak etiketlenir ve mesafe *l* olarak güncellenir. Kök düğüme olan yeni mesafe tüm komşuları ve diğer düğümleri etkileyeceğinden, yeni mesafeyi içeren *layer(l + 1)* mesajı Algoritma 1'de gösterildiği gibi yeni ebeveyn dışındaki tüm komşulara gönderilmektedir.

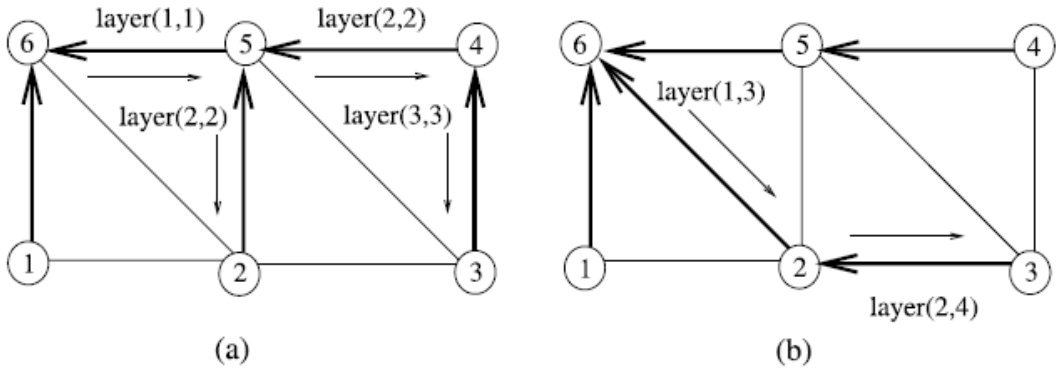
Algoritma 1. *Update_BFS*

```
1:  int parent ← ∅, my_layer ← ∞, count = 1, d ← diameter of G
2:  set of int childs ← ∅, others ← ∅
3:  message types layer, ack, reject
4:  if i = root then
5:      send layer(1) to Γ(i)
6:  end if
7:  while count ≤ d do
8:      receive msg(j)
9:      case msg(j).type of
10:         layer(l):   if my_layer > l then                    ▷ update distance
11:                         parent ← j
12:                         my_layer ← l
13:                         send ack(l) to j                    ▷ inform parent i am child
14:                         send layer(l + 1) to Γ(i)\{j}       ▷ update neighbors
15:                     else
16:                         send reject(l) to j                 ▷ else reject sender
17:         ack(l):     childs ← childs ∪ {j}                   ▷ include sender in children
18:         reject(l):  others ← others ∪ {j}                   ▷ include sender in unrelated
19:     count ← count + 1
20: end while
```

Bu işlemin sonunda kök düğümden başlayan bir BFS ağacının oluşturulduğu görülebilmektedir. Algoritmanın sonlandırma koşulu, *G* çizgesinin çapı (diameter) olacak herhangi iki düğüm arasındaki en uzun yolun geçişi olacaktır.

Algoritmanın Çalışmasına Örnek

Şekil l'de, 1'den 6'ya kadar numaralı altı düğümlü bir topoloji verilmiştir. Buradaki katman mesajı, mesafe ve zaman çerçevesini (time frame) bilgisini taşımaktadır.



Şekil 1. *Update_BFS* algoritmasının çalışmasına örnek

6 numaralı düğüm, algoritmayı *layer(1,1)* mesajını bir hop komşusuna göndererek başlatmaktadır. Her bir komşu düğüm, bu mesajı aldığında, mesajdaki mesafe değerini bilinen mesafeyle karşılaştırır ve yeni mesafe daha küçükse, ebeveyni gönderene atar. Şekil 1.a'da görüldüğü gibi, katman mesajı, düğüm 6 ve düğüm 2 arasındaki doğrudan bağlantıdan önce düğüm 5 yoluyla düğüm 2'ye ulaşmaktadır ve düğüm 2 düğüm 5'i ebeveyn olarak tanımaktadır. Ancak bu durum Şekil 1.b'de düzeltilmiştir. Düğüm 6'dan gelen katman mesajı, üçüncü zaman çerçevesindeki düğüm 2'ye ulaştığında, düğüm 2'nin üst düğüm 5'i doğru düğüm 6 ile değiştirmesine neden olur. Benzer şekilde, 4. zaman çerçevesinde, düğüm 3, düğüm 6 ile başlatılan BFS ağacını doğru bir şekilde oluşturmak için 4 numaralı ebeveyn düğümünü, düğüm 2 ile yer değiştirir.