TÜRKİYE CUMHURİYETİ

YILDIZ TEKNİK ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Algoritma Analizi Proje Raporu

İsim-Soyisim: Berkay Gümüşay

Öğrenci Numarası: 21011084

E-posta Adresi: berkay.gumusay@std.yildiz.edu.tr

Video Linki: https://youtu.be/2Cq61QQojTE

Project Contents

Problem Tanımı

Bizden bir graf yapısı içerisindeki en sık kullanılan kenarları koparark graftaki toplulukları bulan bir algoritma yazmamız istenmiştir.

Problemin Çözümü

Grafların üzerinde BFS algoritması ile gezip grafın her bir kenarını puanlandırıp en yüksek puana sahip kenarı koparma işlemi yaparak bu problem çözülebilir.

Karşılaşılan Sorunlar

Yaygın olarak hep grafın düğümleri üzerinde işlemler yapan algoritmalar ile çalıştığım için bu sefer kenarlar üzerine odaklanan bir algoritma tasarlamak alışılmadık bir çözüm olduğu için belirli yerlerde zorlandım.

Karmaşıklık Analizi

a- Bir iterasyonu gerçekleştiren fonksiyonun sözde kodu:

```
BFS(graph, V, source, distance, numShortestPaths, numOfEdges, allEdges, tmpArr)
   Queue queue
   queue.front <- 0
   queue.rear <- -1

   visited[MAX_NODES]
   for i from 0 to V :
      visited[i] <- 0
      distance[i] <- (-1)
      numShortestPaths[i] <- 0

   visited[source] <- 1
   distance[source] <- 0
   numShortestPaths[source] <- 1

   enqueue(queue, source)

   while !isEmpty(queue):
      current <- dequeue(queue)
      for i from 0 to V :
         if graph[current][i] = 1 and not visited[i] then :
            visited[i] <- 1
            distance[i] <- distance[current] + 1
            numShortestPaths[i] <- numShortestPaths[current] + numShortestPaths[i]
            enqueue(queue, i)
         else if graph[current][i] = 1 && distance[i] = distance[current] + 1 then:
            numShortestPaths[i] <- numShortestPaths[current] + numShortestPaths[i]
         end if
      end for
   end while

   for i from 0 to V :
      if queue.rear >= 0 then :
         edgeNumber <- findEdgeNumber(graph, V, queue.items[queue.rear], numShortestPaths, numOfEdges, allEdges)
         if edgeNumber != 0:
            point <- 1.0
            point <- point + findOtherEdges(allEdges, numOfEdges, queue.items[queue.rear])
            point <- point / edgeNumber
            addPointToEdge(numOfEdges, allEdges, queue.items[queue.rear], point, numShortestPaths, tmpArr)
         end if
      end if

      queue.rear <- queue.rear - 1

   resetEdges(allEdges, numOfEdges)

end function
```

b- Sözde Kodun Açıklaması

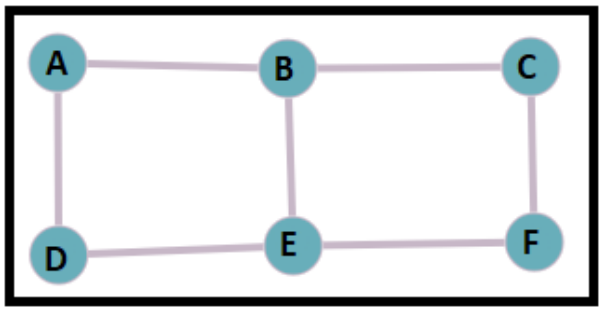
Bu fonksiyon öncelikle kullanılacak olan **Queue** veri yapısını boş bir şekilde oluşturur. Belirtilen bir kaynak düğümden başlayarak grafiğin en kısa yollarını bulmak için **BFS** algoritmasını uygular. Ayrıca her düğüme olan en kısa yolların sayısını hesaplayıp bir dizide tutar ve bu yollara dayalı olarak kenar değerlerini günceller. En son döngüde ise BFS algoritmasında gezilen düğümler tersten olacak şekilde tekrardan gezilir, düğümlerin bağlı olduğu kenarların değerleri belirlenir ve bu değerler addPointToEdge() fonksiyonunda geçici bir diziye atılır. Son olarak kenar değerleri sıfırlanır.

Bu işlem her tur N adet düğüm için tekrar edilir. Komşuluk matrisi kullanılarak yapılan BFS işleminin karmaşıklığı $O(N^2)$'dir. Her turda her düğüm için bu işlem yapıldığından dolayı karmaşıklık **$T(N) = O(N^3)$** olmaktadır.

Ekran Çıktıları

Örnek 1:

Graf:



Matris:

info.txt - Not Defteri

Dosya Düzen Biçim

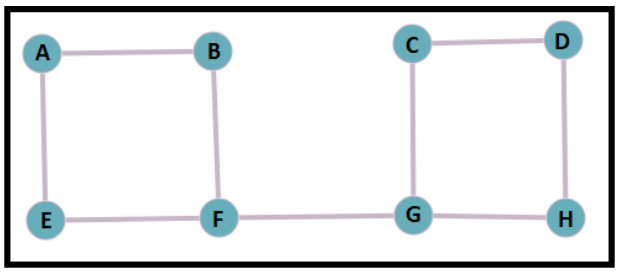
```
6
0 1 0 1 0 0
1 0 1 0 1 0
0 1 0 0 0 1
1 0 0 0 1 0
0 1 0 1 0 1
0 0 1 0 1 0
```

Ekran Çıktısı (k=3, t=2) :

```
C:\Users\BerkayG▄M▄İAY\Desktop\Algo Proje\main.exe
(Tur Sayisi)k: 3
(Minimum Uye Sayisi)t:2

-------1. Adim-------
Guncel Minimum Topluluk Boyutu : 6
1. Topluluk :
A B C D E F

Mevcut Topluluk Sayisi : 1
----------------------------------------
A - B Kenari Kesildi
B - C Kenari Kesildi
D - E Kenari Kesildi
E - F Kenari Kesildi


Kalan Kenarlar :

A - D
B - E
C - F

-------2. Adim-------
!! Sonuc !!

Guncel Minimum Topluluk Boyutu : 2
1. Topluluk :
A D
2. Topluluk :
B E
3. Topluluk :
C F

Mevcut Topluluk Sayisi : 3
----------------------------------------
```

Örnek 2:

Graf:



Matris:

info.txt - Not Defteri

Dosya Düzen Biçim

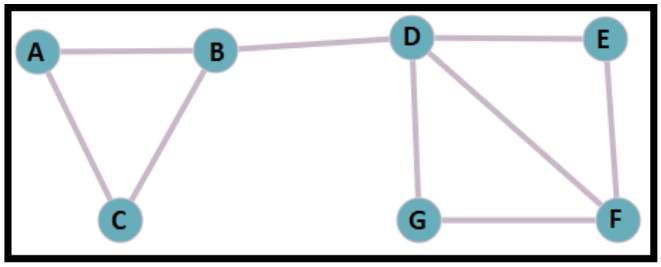
```
8
01001000
10000100
00010010
00100001
10000100
01001010
00100101
00010010
```

Ekran Çıktısı (k=3, t=4) :

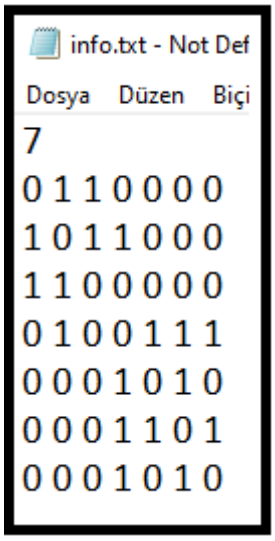
```
C:\Users\BerkayG▄M▄İAY\Desktop\Algo Proje\main.exe
(Tur Sayisi)k: 3
(Minimum Uye Sayisi)t:4

-------1. Adim-------
Guncel Minimum Topluluk Boyutu : 8
1. Topluluk :
A B C D E F G H

Mevcut Topluluk Sayisi : 1
---------------------------------------
F - G Kenari Kesildi


Kalan Kenarlar :

A - B
A - E
B - F
C - D
C - G
D - H
E - F
G - H

-------2. Adim-------
!! Sonuc !!

Guncel Minimum Topluluk Boyutu : 4
1. Topluluk :
A B E F
2. Topluluk :
C D G H

Mevcut Topluluk Sayisi : 2
---------------------------------------
```

Örnek 3:

Graf:



Matris:



Ekran Çıktısı (k=3, t=3) :

```
C:\Users\BerkayG▄M▄İAY\Desktop\Algo Proje\main.exe
(Tur Sayisi)k: 3
(Minimum Uye Sayisi)t:3

-------1. Adim-------
Guncel Minimum Topluluk Boyutu : 7
1. Topluluk :
A B C D E F G

Mevcut Topluluk Sayisi : 1
----------------------------------------
B - D Kenari Kesildi


Kalan Kenarlar :

A - B
A - C
B - C
D - E
D - F
D - G
E - F
F - G

-------2. Adim-------
!! Sonuc !!

Guncel Minimum Topluluk Boyutu : 3
1. Topluluk :
A B C
2. Topluluk :
D E F G

Mevcut Topluluk Sayisi : 2
----------------------------------------
```