BLM 3021 Algoritma Analizi

Algoritma Analizi 2023-2024 Güz Dönemi

DÖNEM PROJESİ

Video Linki: https://youtu.be/6pYkDwbY7go

Ders Yürütücüsü: M. Elif KARSLIGIL
Projeyi Yapan: Berkay ATES
Ders Grubu: 1. Grup
No:21011609
berkay.ates1@std.yildiz.edu.tr

06.01.2024

Projede Istenilen

Proje içerisinde bizden herhangi bir graf üzerinde tüm nodelar için BFS algoritmasını çalıştırdığımızda üzerinden en fazla geçilen **edge'i** tespit etmemiz, sonrasında bu edgei koparmamız isteniyordu. Edge koparma işleminin asıl durma koşulu ise yeni oluşan alt graflardan herhangi biri **t** elemana sahip olması olarak belirlenmişti. Diğer bir durma koşulu ise yeni oluşan alt grafların eleman sayısının son birkaç adım boyuna değişmemiş olmasıydı.

Karşılaşılan Problemler ve Çözümleri

GRAFTA BULUNAN GRUPLAR VE GRUPLARIN NODE SAYISININ BULUNMASI

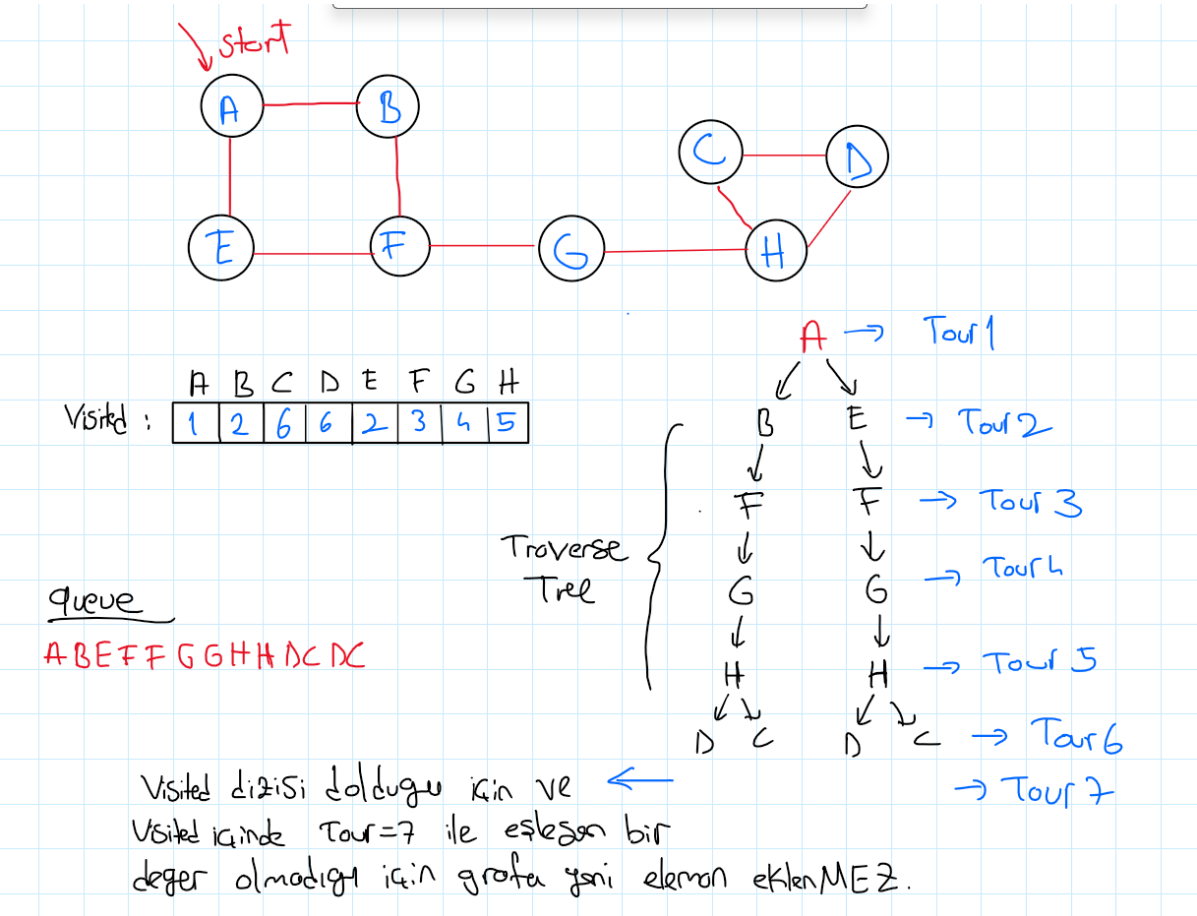
Proje içerisinde ilk karşima çıkan problemlerden biri üzerinde işlem yaptığımız graf içinde toplamda kaç tane alt graf oldugu ve bu alt grafların içinde kaçar tane node olduğu bilgisinin tespit edilmesiydi. Alt grafları bulmak için BFS yi çalıştırıp sıfırlar ile initialize edilmiş visited dizisi içinde aynı değere sahip olan elemanların bir alt graf oluştuduğunu tespit ettim. Aynı zamanda aynı değere sahip olan elemanların sayısının da alt graf içinde kaç tane node olduğu bilgisi oldugunu da tespit ettim.

Visited dizisi üzerinden elde ettiğim alt graflardan herhangi birinde bulunan eleman sayısı istenilen sayıda ise işlemi bitirerek çıktıları ekrana yazdırdım. Eğer tüm alt graflar istenilen t sayısından fazla node içeriyorsa bu durumda **alt graflar içinde değil var olan tüm graf üzerinde** proje dahilinde oluşturduğum algoritmayı çalıştırarak üzerinden en fazla geçilen edge değerini tespit ettim ve bu edgei kopardım. Sonrasında tekrar yeni oluşan alt graflardan herhangi birinde t veya daha az sayıda node olup olmadığını BFS ile kontrol ederek işlemi sonlanana kadar devam ettirdim.

Graf üzerinde herhangi bir nodu kopardıktan sonra oluşan alt graflar içinde üzerinden en fazla defa geçilen edgei bulurken başta oluşturduğum algoritmayı her bir alt graf için ayrı ayrı çalıştırıyordum fakat sonrasında bizimle classroom veya ödev dosyasında paylaşılan örneklerde öyle olmadığını farkettim. Bu yüzden bir kenar kopardıktan sonra oluşturduğum algoritmayı her bir alt graf için değil tüm graflar dahilinde 1 defa çalıştırdım. Zaten calassroom ve ödev yönergesinde paylaşılan örnekler de o şekildeydi.

BFS ILE EN KISA YOLLARIN TAMAMININ BULUNMASI SORUNU VE ÇÖZÜMÜ

BFS algitması ile herhangi bir A nodundan B noduna giden en kısa yolları bulmaya çalıştığımızda en kısa olan yollardan sadece 1 tanesini bulabiliriz. Projede çözülmesi gereken en önemli problemlerden biri buydu. BFS ile herhangi A nodundan B noduna giden tüm nodeların bulunabilmesi için BFS ile graf üzerinde dolanırken aşagıdaki örnekteki gibi bir **TRAVERSE TREE** oluşturmayı denedim. Bu sayede başlangıç olarak seçtiğimiz nodedan graf üzerinde bulunan diğer tüm nodalara giden en kısa yolların tamamını belirleyebildim.



Visited dizisi içindeki elemanlar ziyaret edilince **tour** bilgisi ile işaretleniyor ve sonraki kontrollerde gidilmek istenilen nodun visited node'daki degeri **tour** bilgisi ile eşleşiyor veya hiç ziyaret edilmemiş ise tekrardan bu node ziyaret ediliyor. Örneğin yukarıdaki örnekte hem B hem de E den F ye gidiliyor. B den F ye geçişte visited tarafını **tour=3** ile işaretliyoruz. E den F ye geçişte visited dolu fakat **tour=3** değişkeni ile işaretlendiği için E den F ye gidebiliyoruz.

Yukarıdaki örnekte yapılan işlemi Graf üzerinde bulunan tüm nodelar için oluşturarak herbiri için TRAVERSE TREE bilgisini elde ediyoruz. Böylece BFS kullanarak gezdiğimiz nodeları bir tree ye aktarmak şartı ile herhangi bir A nodundan B noduna giden tüm kısa yolları bulabilmiş oluyoruz.

UZERINDEN EN FAZLA GECILEN EDGE'IN BULUNMASI

Projede var olan bir diğer problem ise graf üzerinde bulunan edgeler üzerinden kaç defa geçildiği bilgisinin elde edilmesiydi. Bu sorunu çözmek adına her bir node için elde ettiğimiz TRAVERSE TREEler üzerinde dolanarak hangı nodedan hangı noda kaç defa geçiş oldugunu kaydettim. Elde ettiğim toplamlar arasında en yüksek değere sahip olan edge bizim koparılması gereken edge değerimizdi. Ilgili edgei kopardıktan sonra bu noktaya kadar yapmış olduğum işlemleri en baştan aradığımız koşullar sağlanana kadar tekrar ettirdim.

Karmaşıklık Analizi

ZAMAN KARMASIKLIGI

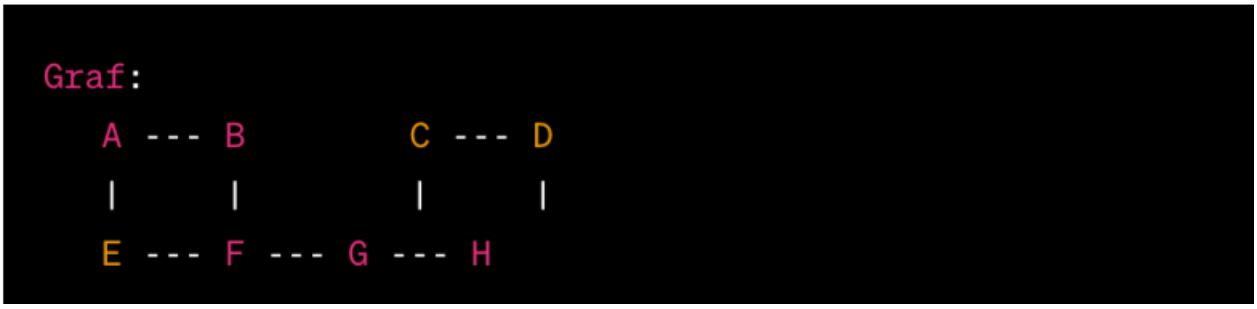
BFS aloritmasının karmaşıklığı O(V+E) dir. Burada V düğüm, E ise kenar sayısını ifade eder yani genel olarak algoritmanın karmaşıklığına O(N) dememizde bir sorun yoktur. Traverse tree oluşturmamızın karmaşıklığı da O(N) olguğu için algoritmanın toplam zaman karmaşıklığı O(N) dir.

YER KARMASIKLIGI

BFS algoritmasının yer karmaşıklığı, genellikle en kötü durumdaki kuyruğun boyutu kadar olacaktır. En kötü durumda, graf geniş bir yapıya sahipse ve kuyruğa tüm düğümler eklenirse, BFS'nin yer karmaşıklığı O(N) olur. Burada N, grafın genişliği (en geniş seviyedeki düğümlerin sayısı) olarak adlandırılır. Ayrıca traverse tree oluştururken de her bir başlangıç nodu için tree oluşturup sonrasında oluşturduğuz tree üzerindeki bağlantıları sayıp traverse tree yi serbest bırakırsak, tree için de yer karmaşıklığımız O(N) olur diyebiliriz.

Ekran Çıktıları

ORNEK 1




```
File name: test1.txt
Desired graph size (t): 4
Matrix Size: 8
Readed Matrix From File:
0 1 0 0 1 0 0 0
1 0 0 0 0 1 0 0
0 0 0 1 0 0 0 1
0 0 1 0 0 0 0 1
1 0 0 0 0 1 0 0
0 1 0 0 1 0 1 0
0 0 0 0 0 1 0 1
0 0 1 1 0 0 1 0
EDGE betweennes is running
```

```
EDGE betweennes is running
BFS Traverse Node Indexes: 0 1 4 5 6 7 2 3

Graph 1 is consist from :  A,  B,  C,  D,  E,  F,  G,  H,
Shorthest path from A to E:['A','E']
Shorthest path from A to B:['A','B']
Shorthest path from A to F:['A','E','F']
Shorthest path from A to G:['A','E','F','G']
Shorthest path from A to H:['A','E','F','G','H']
Shorthest path from A to D:['A','E','F','G','H','D']
Shorthest path from A to C:['A','E','F','G','H','C']
Shorthest path from A to F:['A','B','F']
Shorthest path from A to G:['A','B','F','G']
Shorthest path from A to H:['A','B','F','G','H']
Shorthest path from A to D:['A','B','F','G','H','D']
Shorthest path from A to C:['A','B','F','G','H','C']

Shorthest path from B to F:['B','F']
Shorthest path from B to A:['B','A']
Shorthest path from B to G:['B','F','G']
Shorthest path from B to E:['B','F','E']
Shorthest path from B to H:['B','F','G','H']
Shorthest path from B to D:['B','F','G','H','D']
Shorthest path from B to C:['B','F','G','H','C']
Shorthest path from B to E:['B','A','E']
```

```
Shorthest path from C to H:['C','H']
Shorthest path from C to D:['C','D']
Shorthest path from C to G:['C','H','G']
Shorthest path from C to F:['C','H','G','F']
Shorthest path from C to E:['C','H','G','F','E']
Shorthest path from C to B:['C','H','G','F','B']
Shorthest path from C to A:['C','H','G','F','E','A']
Shorthest path from C to A:['C','H','G','F','B','A']

Shorthest path from D to H:['D','H']
Shorthest path from D to C:['D','C']
Shorthest path from D to G:['D','H','G']
Shorthest path from D to F:['D','H','G','F']
Shorthest path from D to E:['D','H','G','F','E']
Shorthest path from D to B:['D','H','G','F','B']
Shorthest path from D to A:['D','H','G','F','E','A']
Shorthest path from D to A:['D','H','G','F','B','A']

Shorthest path from E to F:['E','F']
Shorthest path from E to A:['E','A']
Shorthest path from E to G:['E','F','G']
Shorthest path from E to B:['E','F','B']
Shorthest path from E to H:['E','F','G','H']
Shorthest path from E to D:['E','F','G','H','D']
Shorthest path from E to C:['E','F','G','H','C']
Shorthest path from E to B:['E','A','B']
```

```
Shorthest path from F to G:['F','G']
Shorthest path from F to E:['F','E']
Shorthest path from F to B:['F','B']
Shorthest path from F to H:['F','G','H']
Shorthest path from F to D:['F','G','H','D']
Shorthest path from F to C:['F','G','H','C']
Shorthest path from F to A:['F','E','A']
Shorthest path from F to A:['F','B','A']

Shorthest path from G to H:['G','H']
Shorthest path from G to F:['G','F']
Shorthest path from G to D:['G','H','D']
Shorthest path from G to C:['G','H','C']
Shorthest path from G to E:['G','F','E']
Shorthest path from G to B:['G','F','B']
Shorthest path from G to A:['G','F','E','A']
Shorthest path from G to A:['G','F','B','A']

Shorthest path from H to G:['H','G']
Shorthest path from H to D:['H','D']
Shorthest path from H to C:['H','C']
Shorthest path from H to F:['H','G','F']
Shorthest path from H to E:['H','G','F','E']
Shorthest path from H to B:['H','G','F','B']
Shorthest path from H to A:['H','G','F','E','A']
Shorthest path from H to A:['H','G','F','B','A']
```

```
      A    B    C    D    E    F    G    H
  A   0    7    0    0    7    0    0    0
  B   7    0    0    0    0   11    0    0
  C   0    0    0    1    0    0    0    7
  D   0    0    1    0    0    0    0    7
  E   7    0    0    0    0   11    0    0
  F   0   11    0    0   11    0   20    0
  G   0    0    0    0    0   20    0   18
  H   0    0    7    7    0    0   18    0
EDGE ('A','B'):7 times
EDGE ('A','E'):7 times
EDGE ('B','F'):11 times
EDGE ('C','D'):1 times
EDGE ('C','H'):7 times
EDGE ('D','H'):7 times
EDGE ('E','F'):11 times
EDGE ('F','G'):20 times
EDGE ('G','H'):18 times
20 is max edge value
```

```
20 is max edge value
BEFORE ADJACENCY MATRIX
       A    B    C    D    E    F    G    H
  A    0    1    0    0    1    0    0    0
  B    1    0    0    0    0    1    0    0
  C    0    0    0    1    0    0    0    1
  D    0    0    1    0    0    0    0    1
  E    1    0    0    0    0    1    0    0
  F    0    1    0    0    1    0    1    0
  G    0    0    0    0    0    1    0    1
  H    0    0    1    1    0    0    1    0
Edge between F and G has been broke off.
AFTER ADJACENCY MATRIX
       A    B    C    D    E    F    G    H
  A    0    1    0    0    1    0    0    0
  B    1    0    0    0    0    1    0    0
  C    0    0    0    1    0    0    0    1
  D    0    0    1    0    0    0    0    1
  E    1    0    0    0    0    1    0    0
  F    0    1    0    0    1    0    0    0
  G    0    0    0    0    0    0    0    1
  H    0    0    1    1    0    0    1    0
```

```
BFS Traverse Node Indexes: 0 1 4 5
BFS Traverse Node Indexes: 2 3 7 6

We reached a graph which have desired number of elements which is 4, last groups are such that:
Graph 1 is consist from :  A,  B,  E,  F,
Graph 2 is consist from :  C,  D,  G,  H,
done
--------------------------------
Process exited after 0.04832 seconds with return value 0
Press any key to continue . . .
```

Ornek2


```
File name: test2.txt
Desired graph size (t): 2
Matrix Size: 5
Readed Matrix From File:
0 1 0 0 0
1 0 1 1 0
0 1 0 0 0
0 1 0 0 1
0 0 0 1 0
EDGE betweennes is running
```

```
BFS Traverse Node Indexes: 0 1 2 3 4

Graph 1 is consist from :  A,  B,  C,  D,  E,
Shorthest path from A to B:['A','B']
Shorthest path from A to D:['A','B','D']
Shorthest path from A to C:['A','B','C']
Shorthest path from A to E:['A','B','D','E']

Shorthest path from B to D:['B','D']
Shorthest path from B to C:['B','C']
Shorthest path from B to A:['B','A']
Shorthest path from B to E:['B','D','E']

Shorthest path from C to B:['C','B']
Shorthest path from C to D:['C','B','D']
Shorthest path from C to A:['C','B','A']
Shorthest path from C to E:['C','B','D','E']

Shorthest path from D to E:['D','E']
Shorthest path from D to B:['D','B']
Shorthest path from D to C:['D','B','C']
Shorthest path from D to A:['D','B','A']
```

```
Shorthest path from E to D:['E','D']
Shorthest path from E to B:['E','D','B']
Shorthest path from E to C:['E','D','B','C']
Shorthest path from E to A:['E','D','B','A']

       A    B    C    D    E
  A    0    4    0    0    0
  B    4    0    4    6    0
  C    0    4    0    0    0
  D    0    6    0    0    4
  E    0    0    0    4    0
EDGE ('A','B'):4 times
EDGE ('B','C'):4 times
EDGE ('B','D'):6 times
EDGE ('D','E'):4 times
6 is max edge value
```

```
6 is max edge value
BEFORE ADJACENCY MATRIX
      A    B    C    D    E
  A   0    1    0    0    0
  B   1    0    1    1    0
  C   0    1    0    0    0
  D   0    1    0    0    1
  E   0    0    0    1    0
Edge between B and D has been broke off.
AFTER ADJACENCY MATRIX
      A    B    C    D    E
  A   0    1    0    0    0
  B   1    0    1    0    0
  C   0    1    0    0    0
  D   0    0    0    0    1
  E   0    0    0    1    0
BFS Traverse Node Indexes: 0 1 2
BFS Traverse Node Indexes: 3 4

We reached a graph which have desired number of elements which is 2, last groups are such that:
Graph 1 is consist from :  A,  B,  C,
Graph 2 is consist from :  D,  E,
done
```