Algoritma Analizi-4

Sosyal Ağda Influencer Algılama

Versiyon 1.0

Prepared by Onur Demir

Grup 2-Ödev 4

onur.demir2@std.yildiz.edu.tr

15.12.2022

Video Linki : https://youtu.be/LDvUaCzOveI

İçerik Tablosu

Revizyon Tarihi

İsim	Tarih	Değişim Sebebi	Versiyon
Onur Demir	20.12.2022	Rapor bitirildi	1.0
Onur Demir	20.12.2022	Video çekildi	1.1

Video Linki : https://youtu.be/LDvUaCzOveI

1. Giriş

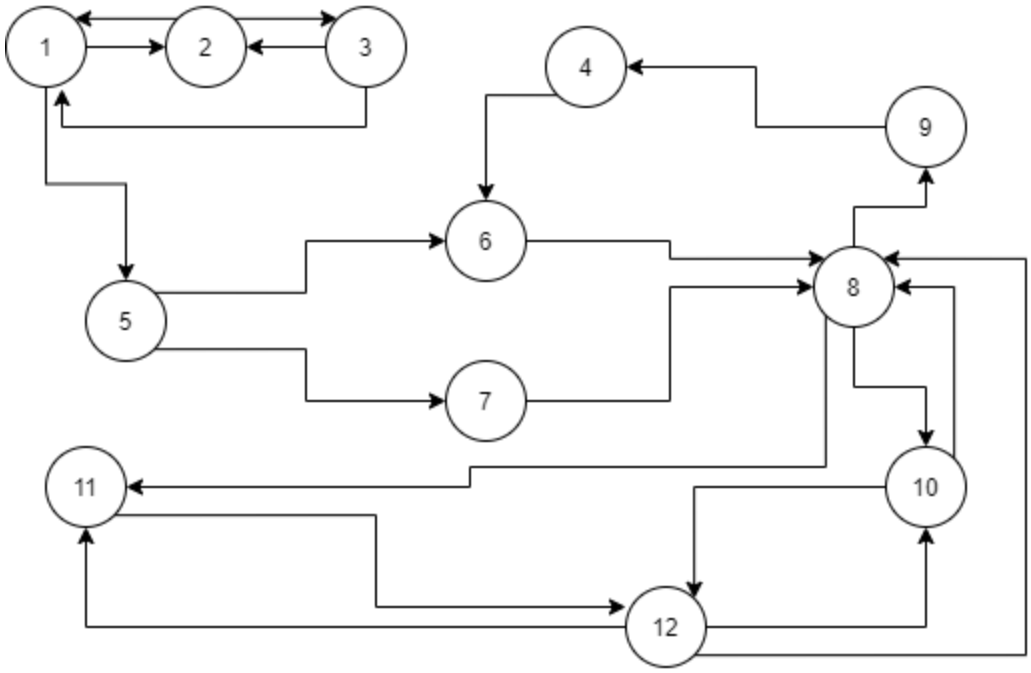
1.1 Problem

Verilen bir graf için in-degree değerleri yani bir düğüme gelen düğümlerin sayısına ya da direk olmayan bağlantı sayıları ile oluşturulacak bir graf üzerinden elemeler yapılmak istenmektedir. Yapılacak işlemler şu şekilde sınıflandırılabilir..

1.Grafın oluşturulması

2.Düğüm sayısı azaltmak için gelen düüğmlerin sayısına göre sürekli o direk bağlantı sayısına erişene kadardüğümlerin bağlantılarının azaltılması..

3.Direk olarak bağlı olmayan bağlantıların bulunması ve bu bağlantı zincirinin en uzununun seçilerek verilen değerlere göre kalan düğümlerin elenmesi.



1.2 Problem Tanımı

Grafın oluşturulması için iki yöntem olan bitişiklik matrix veya bitişiklik listesi yaklaşımları kullanılabilir. Bitişikliklistesi kullanılarak zaman karmaşıklığının azaltılması sağlanabilir. Grafdaki düğümler elenirken bir hashmap içerisinde bu düğümün içerikleri saklanabilir. Graf üzerinde gezinirken DFS ve BFS algoritmalarından biri seçilerek bu algoritma gerçeklenir. Eğer DFS ile gerçeklenecekse Stack veri yapısı, eğer BFS ile gerçeklenecekse kuyruk veri yapısı kullanmak gereklidir.

2. Yöntem

2.1 Bitişiklik Listesi ve DFS Yaklaşımı ile Problemin Çözülmesi

Bitişiklik listesi oluşturulurken toplam node sayısı kadar bir matriks oluşturulur.(ya da alternatif olarak hash map oluşturulabilir ama indeksler ardışık olduğu için dizi oluşturmak da mantıklıdır.)Bu oluşturulan dizi için bir düğüme ait başka düğüm bağlantıları olacağı için bu bağlantılar bir listede tutulacaktır. Bu iki liste düğüme gelen düğümler örneğin 1.node için 2 ve 3 düğümleri düğüme gelen düğümlerdir bu düğümleri bu yüzden in-degree(takipçi) düğümü olarak oluşturulacaktır ayrıca dışarı çıkan düğümler(Takip ettiği) de tutulabilir.Bu düğümlerin oluşturduğuu liste ise outdegree(Takip ettiği) listesi olarak gösterilebilir. Outdegree listesine örnek olarak 5. Düğüm için 6 ve 7 örnek verilebilir. Ayrıca dfs yaklaşımı ile elde edilecek direk olmayan bağlantılar stack olarak elde edilmiştir bu yığınlar düğüm değerinde tutulabilir.Bunun için bağlantıların bulunduğu yığın listesi de bir düğüm içinde tutulacaktır.

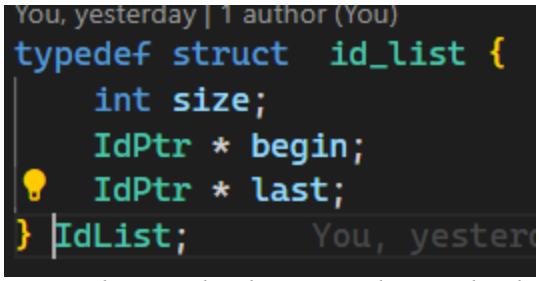
Yukarıda verildiği gibi bir yapı oluşturulursa aşağıdaki struct yapısı elde edilecektir.

```
You, 2 hours ago | 1 author (You)
typedef struct node {
    char * name;
    char * surname;
    Id id;
    IdList * inDegreeList;
    IdList * outDegreeList;
    Number connectivity;
    bool * visited; // exploration purpose
    IdStackList * connections; // copy of in degree connections
} Node;
```

connectivity ise en son IdStackList den çekilerek üzerine yazılacak olan bilgidir ve en uzun direk olmayan bağlantının düğüm sayısı anlamına gelmektedir.

DFS algorimtması için durma şartı ziyaret edilen düüğüme tekrar gelinmesidir. Yukarıdaki örnek için 1 2 3 ziyaret edildikten sonra 2. düğüme gelindiğinde mor bağlantıda durma koşulu oluşacaktır ve yığın çıktısı 1 2 3 2 şeklinde olacaktır bu şekilde bağlantı sayısı bulunmuştur.

Takipçi eleme için oluşan indegreeList yapı değişkenindeki size boyutuna bakılmıştır ve bu değer her silinmede güncellenmiştir. Aşağıda inDegreeList e ait yapı görülmektedir.



IdList ve ve inDegreeList'ten silinme işlemleri aşağıda görülmektedir burada outdegree(takip ettiklerini) tutmanın avantajı görülmektedir. Bir düğümün gittikleri içindeki indegreeler(takipçileri) aşağıdaki gibi silinmektedir.Aynı işlem outdegree içinde tekrarlanımştır.

```
pushBack(removed, node->id);
int nodeId = node->id;
IdPtr *itr = node->outDegreeList->begin;
for ( j = 0 ; j < node->outDegreeList->size ; j++, itr = itr->next) {
    IdList * inDegreeList =  exploreMap[itr->Id]->inDegreeList;
    if(inDegreeList->size ) {
        removeIdList(inDegreeList, nodeId);
    }
}
itr = node->inDegreeList->begin;
for ( j = 0 ; j < node->inDegreeList->size ; j++, itr = itr->next) {
    IdList * outDegreeList =  exploreMap[itr->Id]->outDegreeList;
    if(outDegreeList->size) {
        removeIdList(outDegreeList, nodeId);
    }

}
```

3. Uygulama

3.1 Test Caseler ve Ekran Cıktıları

Test durumu için ilk başta modüller test edilmiştir ilk sayfada verilen graftan doğruluk kontrolü yaplabilir sonra dosya üzerinde eklemeler ile test edilmiştir.

TEST CASE 1:
M değeri 2 alınıp diğer değerler 0 0 alınmıştır. Bu durumda takipçi sayısı 2 kalana kadar eleme yapılmıştır.

NORMAL:

```
 Dumping....
-------------------------------------------------
Account id : 1
Accunt name : Michael
 Account surname : Jordan

Connectivity : 4
Visited(debug purpose) : 0

Following 2,3,

Follower 2,3,
-------------------------------------------------
-------------------------------------------------
Account id : 2
Accunt name : Stephen
 Account surname : Boyd

Connectivity : 4
Visited(debug purpose) : 0

Following 3,1,

Follower 1,3,
-------------------------------------------------
-------------------------------------------------
Account id : 3
Accunt name : Kalyanmoy
 Account surname : Deb

Connectivity : 4
Visited(debug purpose) : 0

Following 1,2,

Follower 1,2,
-------------------------------------------------
```

DETAY

```
Dumping....
-------------------------------------------------
Account id : 1
Accunt name : Michael
 Account surname : Jordan

Connectivity : 4
Visited(debug purpose) : 0

Following 2,3,

Follower 2,3,
Writing all connection string in this row :  connection 0 : 1 2 1 || connection 1 : 1 2 3 1 || connection 2 : 1 2 3 2 ||
 connection 3 : 1 3 || ------------------------------------------------
-------------------------------------------------
Account id : 2
Accunt name : Stephen
 Account surname : Boyd

Connectivity : 4
Visited(debug purpose) : 0

Following 3,1,

Follower 1,3,
Writing all connection string in this row :  connection 0 : 2 1 2 || connection 1 : 2 1 3 1 || connection 2 : 2 1 3 2 ||
 connection 3 : 2 3 || ------------------------------------------------
-------------------------------------------------
Account id : 3
Accunt name : Kalyanmoy
 Account surname : Deb

Connectivity : 4
Visited(debug purpose) : 0

Following 1,2,

Follower 1,2,
Writing all connection string in this row :  connection 0 : 3 1 2 1 || connection 1 : 3 1 2 3 || connection 2 : 3 1 3 ||
 connection 3 : 3 2 || ------------------------------------------------
```

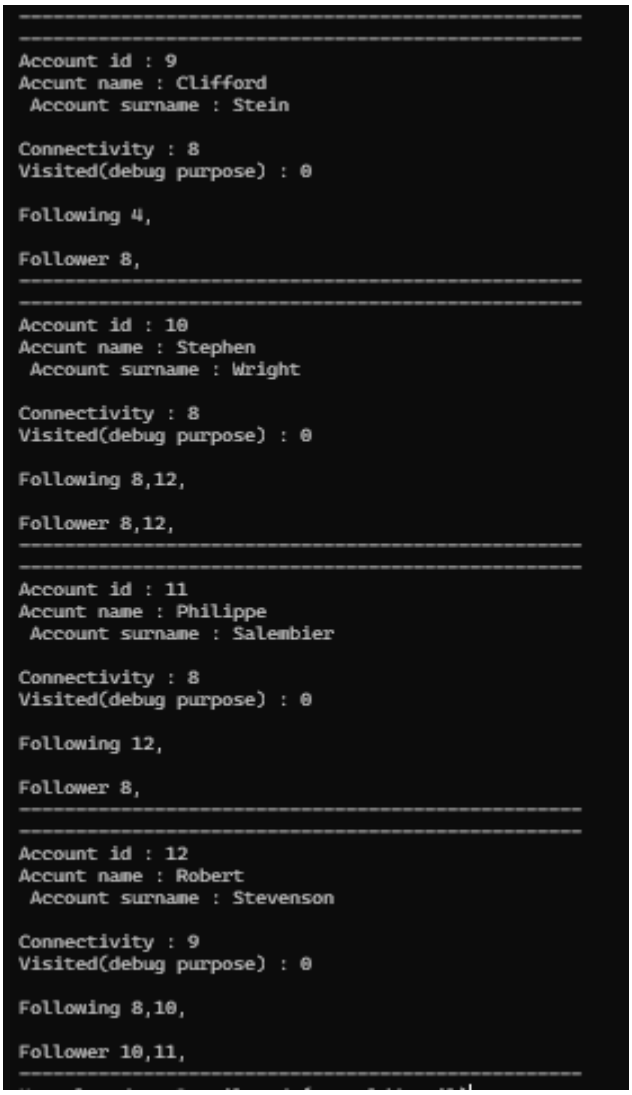
Burada ayrıca direk olmayan bağlantılarda yazılmıştır maksimum bağlantı uzunluğu buradan elde edilir. Direk bağlantı sayısı ise takipçi sayısıdır(follower).

TEST CASE 2:

Bu durumda en uzun bağlantı bulan modül test edilmiştir. M:1 X:1 Y:4 için aşağıdaki sonuç elde edilmektedir.

NORMAL:


Dumping....
Account id : 4
Accunt name : David
Account surname : Johnson
Connectivity : 9
Visited(debug purpose) : 0
Following 6,
Follower 9,
Account id : 6
Accunt name : Lieven
Account surname : Vandenberghe
Connectivity : 10
Visited(debug purpose) : 0
Following 8,
Follower 4,5,
Account id : 8
Accunt name : Jorge
Account surname : Nocedal
Connectivity : 7
Visited(debug purpose) : 0
Following 11,10,9,
Follower 6,7,10,12,


Account id : 9
Accunt name : Clifford
Account surname : Stein
Connectivity : 8
Visited(debug purpose) : 0
Following 4,
Follower 8,
Account id : 10
Accunt name : Stephen
Account surname : Wright
Connectivity : 8
Visited(debug purpose) : 0
Following 8,12,
Follower 8,12,
Account id : 11
Accunt name : Philippe
Account surname : Salembier
Connectivity : 8
Visited(debug purpose) : 0
Following 12,
Follower 8,
Account id : 12
Accunt name : Robert
Account surname : Stevenson
Connectivity : 9
Visited(debug purpose) : 0
Following 8,10,
Follower 10,11,

Detay:

```
Dumping....
-------------------------------------------------
Account id : 4
Accunt name : David
 Account surname : Johnson

Connectivity : 9
Visited(debug purpose) : 0

Following 6,

Follower 9,
Writing all connection string in this row :  connection 0 : 4 9 8 6 4 || connection
 1 : 4 9 8 6 5 1 2 1 || connection 2 : 4 9 8 6 5 1 2 3 1 || connection 3 : 4 9 8 6
5 1 2 3 2 || connection 4 : 4 9 8 6 5 1 3 || connection 5 : 4 9 8 7 5 || connection
 6 : 4 9 8 10 8 || connection 7 : 4 9 8 10 12 10 || connection 8 : 4 9 8 10 12 11 8
 || connection 9 : 4 9 8 12 || ------------------------------------------------
-------------------------------------------------
-------------------------------------------------
Account id : 6
Accunt name : Lieven
 Account surname : Vandenberghe

Connectivity : 10
Visited(debug purpose) : 0

Following 8,

Follower 4,5,
Writing all connection string in this row :  connection 0 : 6 4 9 8 6 || connection
 1 : 6 4 9 8 7 5 1 2 1 || connection 2 : 6 4 9 8 7 5 1 2 3 1 || connection 3 : 6 4
9 8 7 5 1 2 3 2 || connection 4 : 6 4 9 8 7 5 1 3 || connection 5 : 6 4 9 8 10 8 ||
 connection 6 : 6 4 9 8 10 12 10 || connection 7 : 6 4 9 8 10 12 11 8 || connection
 8 : 6 4 9 8 12 || connection 9 : 6 5 || ----------------------------------------
--------
-------------------------------------------------
```

```
------------------------------------------------
Account id : 8
Accunt name : Jorge
 Account surname : Nocedal

Connectivity : 7
Visited(debug purpose) : 0

Following 11,10,9,

Follower 6,7,10,12,
Writing all connection string in this row :  connection 0 : 8 6 4 9 8 || connection
 1 : 8 6 5 1 2 1 || connection 2 : 8 6 5 1 2 3 1 || connection 3 : 8 6 5 1 2 3 2 ||
 connection 4 : 8 6 5 1 3 || connection 5 : 8 7 5 || connection 6 : 8 10 8 || conne
ction 7 : 8 10 12 10 || connection 8 : 8 10 12 11 8 || connection 9 : 8 12 || -----
------------------------------------------------------------------------------
------------
------------------------------------------------------------------------------
------------
Account id : 9
Accunt name : Clifford
 Account surname : Stein

Connectivity : 8
Visited(debug purpose) : 0

Following 4,

Follower 8,
Writing all connection string in this row :  connection 0 : 9 8 6 4 9 || connection
 1 : 9 8 6 5 1 2 1 || connection 2 : 9 8 6 5 1 2 3 1 || connection 3 : 9 8 6 5 1 2
3 2 || connection 4 : 9 8 6 5 1 3 || connection 5 : 9 8 7 5 || connection 6 : 9 8 1
0 8 || connection 7 : 9 8 10 12 10 || connection 8 : 9 8 10 12 11 8 || connection 9
 : 9 8 12 || ------------------------------------------------
------------------------------------------------
```

```
------------------------------------------------
Account id : 10
Accunt name : Stephen
 Account surname : Wright

Connectivity : 8
Visited(debug purpose) : 0

Following 8,12,

Follower 8,12,
Writing all connection string in this row :  connection 0 : 10 8 6 4 9 8 || connect
ion 1 : 10 8 6 5 1 2 1 || connection 2 : 10 8 6 5 1 2 3 1 || connection 3 : 10 8 6
5 1 2 3 2 || connection 4 : 10 8 6 5 1 3 || connection 5 : 10 8 7 5 || connection 6
 : 10 8 10 || connection 7 : 10 8 12 10 || connection 8 : 10 8 12 11 8 || connectio
n 9 : 10 12 || ------------------------------------------------
------------------------------------------------
------------------------------------------------
Account id : 11
Accunt name : Philippe
 Account surname : Salembier

Connectivity : 8
Visited(debug purpose) : 0

Following 12,

Follower 8,
Writing all connection string in this row :  connection 0 : 11 8 6 4 9 8 || connect
ion 1 : 11 8 6 5 1 2 1 || connection 2 : 11 8 6 5 1 2 3 1 || connection 3 : 11 8 6
5 1 2 3 2 || connection 4 : 11 8 6 5 1 3 || connection 5 : 11 8 7 5 || connection 6
 : 11 8 10 8 || connection 7 : 11 8 10 12 10 || connection 8 : 11 8 10 12 11 || con
nection 9 : 11 8 12 || ------------------------------------------------
------------------------------------------------
------------------------------------------------
Account id : 12
Accunt name : Robert
 Account surname : Stevenson

Connectivity : 9
Visited(debug purpose) : 0

Following 8,10,

Follower 10,11,
Writing all connection string in this row :  connection 0 : 12 10 8 6 4 9 8 || conn
ection 1 : 12 10 8 6 5 1 2 1 || connection 2 : 12 10 8 6 5 1 2 3 1 || connection 3
: 12 10 8 6 5 1 2 3 2 || connection 4 : 12 10 8 6 5 1 3 || connection 5 : 12 10 8 7
 5 || connection 6 : 12 10 8 10 || connection 7 : 12 10 8 12 || connection 8 : 12 1
0 12 || connection 9 : 12 11 8 || ------------------------------------------------
-
```

TEST CASE 3:

M:1 X:2 Y:4 için çalıştırılırsa aşağıdaki sonuç elde edilir bu durumda X den dolayı da bir eleme olacaktır ve o modül test edilecektir.

Normal:

```
 Dumping....
-------------------------------------------------
Account id : 6
Accunt name : Lieven
 Account surname : Vandenberghe

Connectivity : 10
Visited(debug purpose) : 0

Following 8,

Follower 4,5,
-------------------------------------------------
-------------------------------------------------
Account id : 8
Accunt name : Jorge
 Account surname : Nocedal

Connectivity : 7
Visited(debug purpose) : 0

Following 11,10,9,

Follower 6,7,10,12,
-------------------------------------------------
-------------------------------------------------
Account id : 10
Accunt name : Stephen
 Account surname : Wright

Connectivity : 8
Visited(debug purpose) : 0

Following 8,12,

Follower 8,12,
-------------------------------------------------
-------------------------------------------------
Account id : 12
Accunt name : Robert
 Account surname : Stevenson

Connectivity : 9
Visited(debug purpose) : 0

Following 8,10,

Follower 10,11,
```

Detay:

Durum ikiden elenenler step 2 eliminasyon da gösterilmiştir.

```
STEP 2 elimination



Selecting values from X elimination
Id :4, Name: David,Surname : Johnson

Id :5, Name: Scott,Surname : Kirkpatrick

Id :7, Name: Fabian,Surname : Pedregosa

Id :9, Name: Clifford,Surname : Stein

Id :11, Name: Philippe,Surname : Salembier
```

```
Dumping....
-------------------------------------------------
Account id : 6
Accunt name : Lieven
 Account surname : Vandenberghe

Connectivity : 10
Visited(debug purpose) : 0

Following 8,

Follower 4,5,
Writing all connection string in this row :  connection 0 : 6 4 9 8 6 || connection 1 : 6 4 9 8 7 5 1 2
1 || connection 2 : 6 4 9 8 7 5 1 2 3 1 || connection 3 : 6 4 9 8 7 5 1 2 3 2 || connection 4 : 6 4 9 8
7 5 1 3 || connection 5 : 6 4 9 8 10 8 || connection 6 : 6 4 9 8 10 12 10 || connection 7 : 6 4 9 8 10 1
2 11 8 || connection 8 : 6 4 9 8 12 || connection 9 : 6 5 || ----------------------------------------
-------
=======-----------------------------------------
-------------------------------------------------
Account id : 8
Accunt name : Jorge
 Account surname : Nocedal

Connectivity : 7
Visited(debug purpose) : 0

Following 11,10,9,

Follower 6,7,10,12,
Writing all connection string in this row :  connection 0 : 8 6 4 9 8 || connection 1 : 8 6 5 1 2 1 || c
onnection 2 : 8 6 5 1 2 3 1 || connection 3 : 8 6 5 1 2 3 2 || connection 4 : 8 6 5 1 3 || connection 5
: 8 7 5 || connection 6 : 8 10 8 || connection 7 : 8 10 12 10 || connection 8 : 8 10 12 11 8 || connecti
on 9 : 8 12 || ----------------------------------------------
-------------------------------------------------
```

```
------------------------------------------------
Account id : 10
Accunt name : Stephen
 Account surname : Wright

Connectivity : 8
Visited(debug purpose) : 0

Following 8,12,

Follower 8,12,
Writing all connection string in this row :  connection 0 : 10 8 6 4 9 8 || connection 1 : 10 8 6 5 1 2
1 || connection 2 : 10 8 6 5 1 2 3 1 || connection 3 : 10 8 6 5 1 2 3 2 || connection 4 : 10 8 6 5 1 3 |
| connection 5 : 10 8 7 5 || connection 6 : 10 8 10 || connection 7 : 10 8 12 10 || connection 8 : 10 8
12 11 8 || connection 9 : 10 12 || ------------------------------------------------
------------------------------------------------
Account id : 12
Accunt name : Robert
 Account surname : Stevenson

Connectivity : 9
Visited(debug purpose) : 0

Following 8,10,

Follower 10,11,
Writing all connection string in this row :  connection 0 : 12 10 8 6 4 9 8 || connection 1 : 12 10 8 6
5 1 2 1 || connection 2 : 12 10 8 6 5 1 2 3 1 || connection 3 : 12 10 8 6 5 1 2 3 2 || connection 4 : 12
 10 8 6 5 1 3 || connection 5 : 12 10 8 7 5 || connection 6 : 12 10 8 10 || connection 7 : 12 10 8 12 ||
 connection 8 : 12 10 12 || connection 9 : 12 11 8 || ------------------------------------------------
```

4. Sonuç

4.1 Fonksiyonların Incelenmesi

```
while (fgets(line, LINE_BUFFER, socialNetFile)) {
```

…
İlk Satır Okunur ve geçilir sonra
…

```
char ** result = strSplit_r(line,',');

for(size_t i = 0 ; i < resultSize ; i++) {
    if( nodeMap[atoi(result[i])] != NULL ) {

        pushBack(nodeMap[atoi(result[i])]->inDegreeList, id);
        pushBack(nodeMap[id]->outDegreeList, atoi(result[i]));

    }
    else {
        Node * myNode = (Node*) malloc(sizeof(Node));
        myNode->id = atoi(result[i]);
        myNode->connectivity=0;
        myNode->visited = (bool*) malloc(sizeof(bool));
        *myNode->visited = false;
        IdList * idVector = allocateVectorId();
        IdList * outVector = allocateVectorId();
        myNode->inDegreeList = idVector;
        myNode->outDegreeList = outVector;



        pushBack(myNode->inDegreeList, id);
        nodeMap[atoi(result[i])] = myNode;
        pushBack(nodeMap[id]->outDegreeList, atoi(result[i]));
    }
}
```

```
}
```

Yukarıdaki kod bloğu ile ilk okuma sırasında bir düğüm oluşturulur(bu kısım atıldı.). Her ikinci satırı okuduğunda ise her bir takip edilen indegree değeri çıktıda verilen takip ettikleri üzerinden oluşturulur. Örneğin Michael Jordan 2 yi takip ediyorsa 2 Michel Jordanın takipçisidir.

Bu mantık ile oluşturulan bitişiklik listesi her bir kişinin takip ettiği kişi sayısı kadar bir karmaşıklık oluşturacaktır ve takip edenler de aynı şekilde bu karmaşıklıkta olacaktır.

Örneğin N tane düğüm varsa ve bir kişinin ortalama ortalama takip ettiği kişi sayısı M ise;

*Grafın oluşturulması için geçen zaman karmaşıklığı O(N*M) şeklinde olacaktır. Yer karmaşıklığı ise O(N*M) şeklinde olacaktır.*

```
while (searchTag) {
    searchTag = false;
    for( i = 1 ; i < exploreMapSize ; i++ ) {
        if(exploreMap[i] != NULL) {
            Node * node = exploreMap[i];
            if( node->inDegreeList->size < M ) {
                pushBack(removed, node->id);
                int nodeId = node->id;
                IdPtr *itr = node->outDegreeList->begin;
                for ( j = 0 ; j < node->outDegreeList->size ; j++, itr = itr->next) {
                    IdList * inDegreeList =  exploreMap[itr->Id]->inDegreeList;
                    if(inDegreeList->size ) {
                        removeIdList(inDegreeList, nodeId);
                    }
                }
                itr = node->inDegreeList->begin;
                for ( j = 0 ; j < node->inDegreeList->size ; j++, itr = itr->next) {
                    IdList * outDegreeList =  exploreMap[itr->Id]->outDegreeList;
                    if(outDegreeList->size) {
                        removeIdList(outDegreeList, nodeId);
                    }

                }
                searchTag = true;
            }
        }
    }
```

…
GRAFTAN_SILME(nodeMap);// push_bach yaptığı düğümü siler

```
}
```

*M değeri üzerinden eleme işlemi için N*M lik bir grafı sürekli yarısı kadar elimine ederse searchtag değişkeninin oluşturacağı iterasyon için log(N*M) kadarlık bir iterasyon sayısından söz edilebilir fakat içerideki for için grafta yaptığı tarama sayısı sabit değer değildir bu yüzden ortalam durumu incelersek şu sonucu elde ederiz. For içerisinde ilk seferde O(N*M) lik bir tarama yapacaktır. Fakat bu tarama sayısı ortalama durumda yarıya düşeceği için*

N(1+1/2+1/4+1/8….) = 2*N olacaktır ve bu durumda fonksiyondan çıkacaktır. Ortalama durumda karmaşıklık O(N*(iç for karmaşıklığı olacaktır.)) dir.*
En kötü durumda ise her seferinde 1 tane düğüm elenmesi durumudur. Bu durumda N(N-1)*(N-2)……*1 ~ N^2 ve zaman karmaşıklığı O(N^2 * (iç for karmaşıklığı olacaktır.)) olacaktır.*

İç for için En kötü durum ve en iyi durumu incelersek ;
-En iyi durumda düğüm sayısı kadar gezecektir ve ilgili düğümün bağlantı değerleri silinirken O(1) lık zaman geçececktir ve bir sonraki searchTag iterasyonunda tüm değerler için ilgili koşul

sağlanıp O(N) sürede döngüden çıkacaktır. Bu durumda O(N) lik toplam zaman karmaşıklığı en iyi durumda gerçekleşir.
*-En kötü durum için her bir düğüm silinirken tablo üzerindeki tüm düğümlere gidecektir ve onların bilgisini güncelleyecektir.bu durumda karmaşıklık O(N*M) olacaktır.*

Bu durumda ortalama durum karmaşıklığı O(N*M) olacaktır.
En kötü durumda ise O(N^3) olacaktır.
En iyi durumda ise O(N) olacaktır.
X ile eleme işlem ise Θ(N) süre alacaktır.

```
for ( i = 0 ; i < exploreMapIdx->size ; i++,itr=itr->next  ) {
    Node * node = exploreMap[itr->Id];
    IdStack * stack = allocateVectorId();
    IdStack * visitedPointers = allocateVectorId();
    pushBack (stack, node->id);
    // *node->visited = true;
    // visitedPointers.push(node->visited);
    dfs(stack, visitedPointers, exploreMap, node); // return all t
    // node->connectivity = connectivity;

    while(visitedPointers->size!= 0) {
        *exploreMap[top(visitedPointers)]->visited =false;
        pop(visitedPointers);
    }

}
```

```
bool *node::visited
exploration purpose
```

```
void dfs( IdStack * dfsStack, IdStack * visitedPointers, Node ** nodeMap, Node* nodeMain ) {
    int i , j;
    IdPtr * itr = nodeMain->inDegreeList->begin;
    for ( i = 0 ; i < nodeMain->inDegreeList->size ; i++, itr=itr->next) {
        Node * node = nodeMap[itr->Id];
        if(*node->visited) {
            Node * nodeTracked = nodeMap[dfsStack->begin->Id];
            if(nodeTracked->connections == NULL) {
                nodeTracked->connections = (IdStackList*) malloc(sizeof(IdStackList));
                nodeTracked->connections->IdStacks = (IdStackPtr* ) malloc(sizeof(IdStackPtr));
                nodeTracked->connections->indirectConnectionSize = 1;
                nodeTracked->connections->IdStacks->next = NULL;
                pushBack(dfsStack, itr->Id);
                IdStack * stack = copyStack(dfsStack);
                nodeTracked->connections->IdStacks->stack = stack;
            }
            else {
                pushBack(dfsStack, itr->Id);
                IdStackPtr * itr = nodeTracked->connections->IdStacks;
                for ( j = 0 ; j < nodeTracked->connections->indirectConnectionSize-1 ; j++) {
                    itr = itr->next ;
                }
                itr->next = (IdStackPtr* ) malloc(sizeof(IdStackPtr));
                itr->next->stack =  copyStack(dfsStack);
                itr->next->next=NULL;
                nodeTracked->connections->indirectConnectionSize++;
            }
            pop(dfsStack);                  You, 2 seconds ago • Uncommitted changes
            continue;
        }
        *nodeMain->visited =  true;
        pushBack(dfsStack, itr->Id);
        dfs(dfsStack, visitedPointers, nodeMap, node) ;
    }
   *nodeMain->visited =  true;
    pushBack(visitedPointers, nodeMain->id);
    pop(dfsStack);
    return;
}
```

DFS için zaman karmaşıklığı O(V+E) şeklindedir E gidişten V ise dönüşten gelen değerdir.V düğüm sayısıdır ve E ise düğüme giden komşu sayısıdır.Bir for içinde dfs kullanıldığı için karmaşıklık

O(V*(V+E)) şeklinde olacaktır.