

**TÜRKİYE CUMHURİYETİ**  
**YILDIZ TEKNİK ÜNİVERSİTESİ**  
**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**



**ALGORİTMA ANALİZİ**  
**ÖDEV - 4**

**Öğrenci No:** 20011055

**Öğrenci Adı Soyadı:** Şeymanur KORKMAZ

**Öğrenci e-posta:** [seymanur.korkmaz@std.yildiz.edu.tr](mailto:seymanur.korkmaz@std.yildiz.edu.tr)

**Ders/Grup:** BLM3021- Algoritma Analizi / Gr-1

Ders Yürütücüsü:  
Mine Elif KARSLIGİL  
**Aralık, 2022**

## Table of Contents

1. YÖNTEM .....	3
PROBLEM .....	3
ÇÖZÜM .....	3
2. UYGULAMA .....	4
NORMAL MOD .....	4
DETAYLI MOD .....	5
3. SONUÇ .....	6
calculate_inDegree Fonksiyonu Karmaşıklığı .....	6
eliminate_nodes Fonksiyonu Karmaşıklığı .....	7
find_connections Fonksiyonu Karmaşıklığı .....	8
find_influencers Fonksiyonunun Karmaşıklığı .....	9
4. VIDEO LINKİ .....	9

## 1. YÖNTEM

### PROBLEM

Verilen dosyadan okunan sosyal ağda influencer olan kişileri tespit etmek.

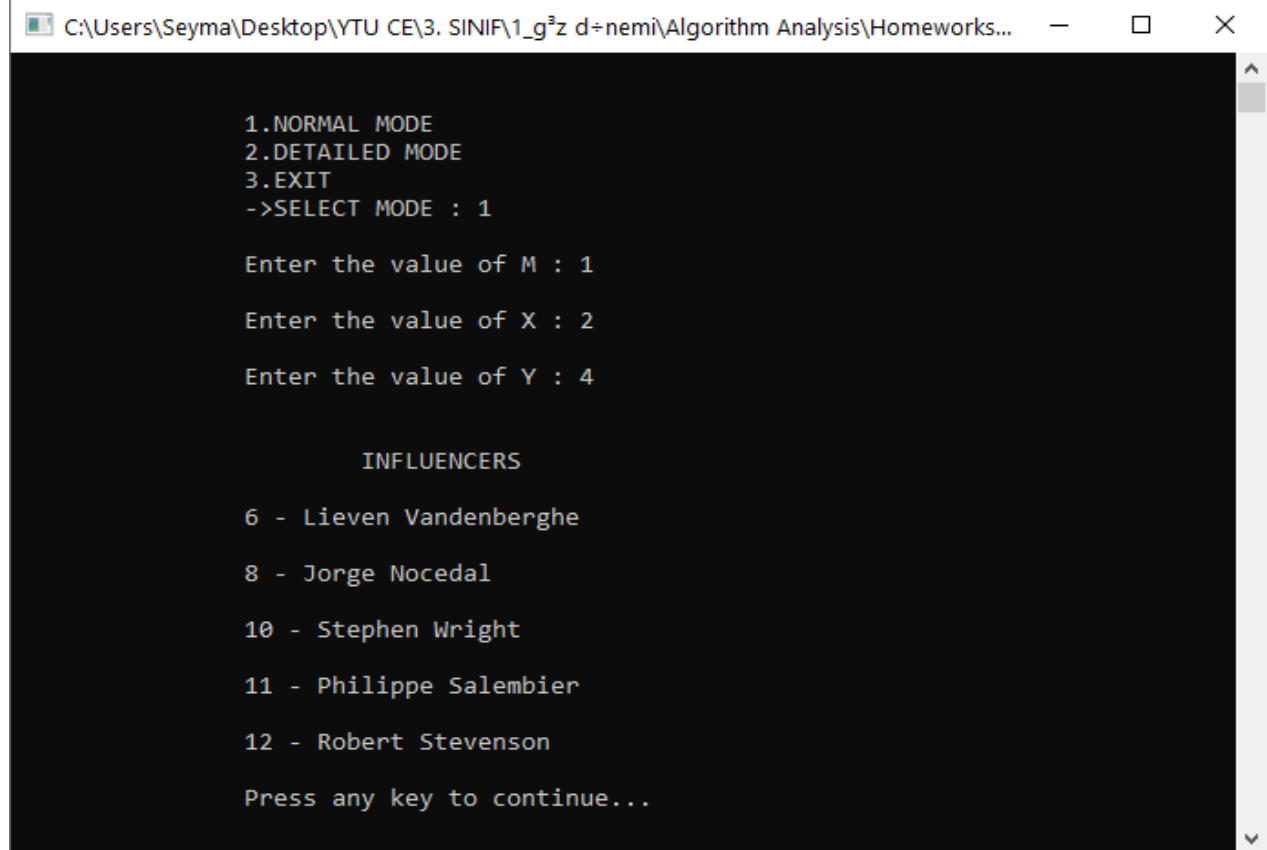
### ÇÖZÜM

Dosyadan okunan her bir kullanıcı birer node, kullanıcılar arasındaki takipler de yönlü bağlantı kabul edilerek bir graf veri yapısı oluşturulur. Bu sayede kullanıcılar arasındaki ilişkiler daha rahat şekilde tespit edilerek influencer olmayan kullanıcılar ayıklanabilir.

- **void read\_file(NODE \*\*node, int \*numNodes) :**  
Dosyadan okunan kullanıcı bilgileri node isimli struct dizisinde tutulur. Kullanıcıların takip ettiği kullanıcılar ise node isimli struct'ın içerisindeki follow isimli struct'ta linkli liste şeklinde tutulur. Toplam kullanıcı sayısı bulunur.
- **int calculateInDegree(NODE \*\*nodes, int i, int numNodes);**  
Herhangi bir kısıt olmadan tüm düğümlerin (kullanıcıların) in-degree (takipçi) değeri bulunur.
- **void eliminate\_nodes(NODE \*\*nodes, int M, int i) :**  
In-degree değeri girilen M değerinden küçük olan düğümler elenir. O düğümün takip ettiği düğümlerin in-degree değerleri bir eksiltir. Eleme işleminden sonra in-degree değeri M'in altına düşen düğümler de elenir.
- **void find\_connections(NODE \*new\_node, int i, int \*dizi) :**  
M değerine göre yapılan eleme işleminin ardından elenmeyen düğümler için doğrudan ve dolaylı yoldan gelen toplam bağlantı sayısı bulunur. Bu fonksiyon DFS algoritması yaklaşımıyla tasarlanmıştır. Bağlantı bulma işlemi her bir düğüm için takip edilen düğümlere dallanarak en derine kadar gider. Ardından bir sonraki düğüme geçerek işlem devam eder.
- **void find\_influencers(NODE \* new\_node, int new\_numNodes, int X, int Y) :**  
Elemenin ardından kalan düğümler içerisinde in-degree değeri verilen X değerinden, toplam bağlantı sayısı ise verilen Y değerinden büyük eşit olan düğümler bulunur. Bu düğümler influencer olan kullanıcılardır.

## 2. UYGULAMA

### NORMAL MOD



```
C:\Users\Seyma\Desktop\YTU CE\3. SINIF\1_g²z d=nemi\Algorithm Analysis\Homeworks...  
  
1.NORMAL MODE  
2.DETAILED MODE  
3.EXIT  
->SELECT MODE : 1  
  
Enter the value of M : 1  
Enter the value of X : 2  
Enter the value of Y : 4  
  
INFLUENCERS  
  
6 - Lieven Vandenberghe  
8 - Jorge Nocedal  
10 - Stephen Wright  
11 - Philippe Salembier  
12 - Robert Stevenson  
  
Press any key to continue...
```

## DETAYLI MOD

```
C:\Users\Seyma\Desktop\YTU CE\3. SINIF\1_g^z d+nem\Algorithm Analysis\Homeworks\HW_4\20011055.exe

1.NORMAL MODE
2.DETAILED MODE
3.EXIT

->SELECT MODE : 2

Enter the value of M : 1
Enter the value of X : 2
Enter the value of Y : 4

      IN-DEGREE VALUES OF ALL NODES

1 - Michael Jordan      in-degree value : 2
2 - Stephen Boyd        in-degree value : 2
3 - Kalyanmoy Deb        in-degree value : 2
4 - David Johnson        in-degree value : 1
5 - Scott Kirkpatrick    in-degree value : 1
6 - Lieven Vandenberghe in-degree value : 2
7 - Fabian Pedregosa      in-degree value : 1
8 - Jorge Nocedal         in-degree value : 4
9 - Clifford Stein        in-degree value : 1
10 - Stephen Wright       in-degree value : 2
11 - Philippe Salembier  in-degree value : 2
12 - Robert Stevenson     in-degree value : 2

      NODES THAT ARE NOT ELIMINATED FOR THE GIVEN M VALUE

1 - Michael Jordan      in-degree value : 2
2 - Stephen Boyd        in-degree value : 2
3 - Kalyanmoy Deb        in-degree value : 2
4 - David Johnson        in-degree value : 1
5 - Scott Kirkpatrick    in-degree value : 1
6 - Lieven Vandenberghe in-degree value : 2
7 - Fabian Pedregosa      in-degree value : 1
8 - Jorge Nocedal         in-degree value : 4
9 - Clifford Stein        in-degree value : 1
10 - Stephen Wright       in-degree value : 2
11 - Philippe Salembier  in-degree value : 2
12 - Robert Stevenson     in-degree value : 2

      INFLUENCERS

6 - Lieven Vandenberghe in-degree value : 2 total number of direct and indirect connections : 11
8 - Jorge Nocedal         in-degree value : 4 total number of direct and indirect connections : 11
10 - Stephen Wright       in-degree value : 2 total number of direct and indirect connections : 11
11 - Philippe Salembier  in-degree value : 2 total number of direct and indirect connections : 11
12 - Robert Stevenson     in-degree value : 2 total number of direct and indirect connections : 11

Press any key to continue...
```

### 3. SONUÇ

#### calculate\_inDegree Fonksiyonu Karmaşıklığı

```
int calculate_inDegree(NODE **nodes, int i, int numNodes) {  
    (*nodes)[i].inDegree = 0;  
    int k;  
  
    for (k = 0; k < numNodes; k++){  
        FOLLOW * f = (*nodes)[k].followed;  
        while(f != NULL){  
            if ((*nodes)[i].no == f->no) {  
                (*nodes)[i].inDegree++;  
            }  
            f = f->next;  
        }  
    }  
    return (*nodes)[i].inDegree;  
}
```

Her bir düğümü dolaşırken, takip edilen tüm düğümlerin zaman karmaşıklığını da hesaba katmalıyız. Bu nedenle N düğüm sayısı, E bağlantı sayısı olmak üzere zaman karmaşıklığı  $O(N \cdot E)$  olur.

Bu fonksiyon ana fonksiyon içerisinde node dizisinin düğümleri gezilerek her bir düğüm için ayrı ayrı çağrıldığından karmaşıklık  $O(N \cdot N \cdot E)$  olacaktır.

## eliminate\_nodes Fonksiyonu Karmaşıklığı

```
void eliminate_nodes(NODE **nodes, int M, int i) { //DFS

    if((*nodes)[i].eliminate == 1)
        return;
    else{
        FOLLOW *f = (*nodes)[i].followed;
        if ((*nodes)[i].inDegree < M) {
            (*nodes)[i].eliminate = 1;

            while(f != NULL){
                (*nodes)[f->no-1].inDegree--;
                eliminate_nodes(nodes, M, (f->no-1));
                f = f->next;
            }
        }
    }
}
```

Verilen kod parçasında, bir düğüm ve tüm takip ettiği düğümler üzerinde işlem yapılarak, tüm düğümlerin in-degree değerleri kontrol edilir ve eğer in-degree değeri M değerinin altına düşerse, o düğümün eliminate flagi 1 yapılır.

Her bir düğümü dolaşırken düğümün takip ettiği tüm düğümlerin karmaşıklığını da hesaba katmalıyız.

Bu nedenle N düğüm sayısı, E bağlantı sayısı olmak üzere zaman karmaşıklığı  $O(N+E)$  olur. Bu fonksiyon ana fonksiyon içerisinde node dizisinin düğümleri gezilerek her bir düğüm için ayrı ayrı çağrıldığından karmaşıklık  $O(N*(N+E))$  olacaktır.

## find\_connections Fonksiyonu Karmaşıklığı

```
void find_connections(NODE *node,int i, int *visited){
    int j;
    FOLLOW *f = node[i].followed;
    visited[i]=1;

    while (f != NULL ){
        if(visited[f->no-1] == 0 ){
            node[f->no-1].connection++;
            visited[f->no-1] = 1;
            find_connections(node,(f->no-1),visited);
        }
        f = f->next;
    }
}
```

Verilen kod parçasında, her düğüm için takip edilen düğümlerin bir listesi (followed) tutulur ve bu liste üzerinde DFS araması yapılır. DFS araması, bir düğüm için takip edilen düğümleri sırasıyla ziyaret ederek yapılır. Bu nedenle, graf üzerinde DFS araması yapılırken tüm düğümler ve bağlantılar dikkate alınır.

Bu nedenle verilen fonksiyonun karmaşıklığı  $N$  düğüm sayısı,  $E$  bağlantı sayısı olmak üzere zaman  $O(N+E)$  olarak değerlendirilebilir. Bu fonksiyon ana fonksiyon içerisinde node dizisinin düğümleri gezilerek her bir düğüm için ayrı ayrı çağrıldığından karmaşıklık  $O(N*(N+E))$  olacaktır.



## find\_influencers Fonksiyonunun Karmaşıklığı

```
void find_influencers(NODE * node, int numNodes,int X, int Y){  
    int i;  
    for(i=0;i<numNodes;i++){  
        if(node[i].inDegree >= X && node[i].connection >= Y && node[i].eliminate == 0)  
            node[i].influencer = 1;  
        else  
            node[i].influencer = 0;  
    }  
}
```

Bu fonksiyonun zaman karmaşıklığı, node dizisindeki düğümlerin sayısına (numNodes) bağlıdır. Her düğüm bir Kez ziyaret edilecektir. Bu nedenle fonksiyonun karmaşıklığı  $O(N)$ 'dir.

## 4. VIDEO LINKİ

<https://youtu.be/aa9INCn-ag>