
# ALGORİTMA ANALİZİ

PROF.DR.MİNE ELİF KARSLIGİL

ÖDEV 4

19 Aralık 2022

20011623
Asude Merve EKİZ

**Yöntem**
Verilen ödevde bizden istenen socialNET.txt dosyasından okuduğumuz kişiler arasından belirli şartları sağlayan kişileri "Influencer" olarak belirlememizdir.

Bu şartlar şu şekildedir:

- Öncelikle bu kişilerin takipçi sayıları bulunacak (ki bu bizim inDegree dizimizin ilk hali oluyor).
- Sonra takipçi sayısı verilen m değerinden az olan kişiler elenecek ve elenirken takipçisi oldukları kişilerin de inDegreesinde azalışa sebep olacak.
- Daha sonra kalan kişilerin doğrudan ya da dolaylı olarak kaç kişiyle bağlı oldukları bulunacak.
- En sonunda yukarda elenmeyen kişilerden, inDegreesi belli bir x değerinden büyük ve bir üstte bulunan doğrudan ya da dolaylı bağlı olduğu kişi sayısı da belli bir y değerinden büyük olan kişiler seçilecek ve Influencer olarak nitelendirilecek.

*Çözüm*

- Problem çözümü için öncelikle dosyadan kişiler okundu, kişiler PERSON isimli bir struct yapısıyla kaydedildi ve people isimli arrayde tutuldu.
- Sonrasında takip ettikleri kişilerle olan ilişkileri dinamik şekilde allocate edilen bir komşuluk matrisiyle ifade edildi.

  Matris aşağıdaki gibidir: Örneğin 1 numaralı kişi 3-8-12 yi takip ettiği için matriste ilgili gözler 1 ile doldurulmuştur.

```
****** Adj Matrix ********
  0  1  2  3  4  5  6  7  8  9 10 11 12
  1  0  0  1  0  0  0  0  1  0  0  0  1
  2  1  0  1  0  0  0  0  0  0  0  0  1
  3  1  1  0  0  0  0  0  1  0  0  0  0
  4  0  0  1  0  0  1  0  0  0  0  0  0
  5  0  0  0  0  0  1  1  0  0  0  0  0
  6  0  0  1  0  0  0  0  1  0  0  0  0
  7  0  0  0  0  0  1  0  1  0  0  0  0
  8  0  0  0  0  0  0  0  0  1  1  1  0
  9  0  0  0  1  0  0  0  0  0  0  0  1
 10  0  0  0  0  1  0  0  1  0  0  0  1
 11  0  1  0  0  0  0  0  0  0  0  0  1
 12  0  0  0  0  0  0  0  1  0  1  1  0
```

- Daha sonra bu matriste gezilerek her bir kişiyi kaç kişinin takip ettiği bulundu. Bulunan takipçi sayıları inDegree arrayine yerleştirildi.

Indegree Array — Başlangıçta followersları tutuyor

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | m=2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 2 | 4 | 1 | 1 | 3 | 1 | 6 | 1 | 2 | 2 | 5 | 1 ve 0 elenecek |
| 2 | 2 | 2 | -1 | -1 | -1 | -1 | 4 | -1 | 2 | 2 | 4 | |

| | following | | |
|---|---|---|---|
| 1 | 3 | 8 | 12 |
| 2 | 3 | 1 | 12 |
| 3 | 1 | 2 | 8 |
| 4 | 6 | 3 | |
| 5 | 6 | 7 | |
| 6 | 8 | 3 | |
| 7 | 8 | 6 | |
| 8 | 11 | 10 | 9 |
| 9 | 4 | 12 | |
| 10 | 5 | 8 | 12 |
| 11 | 12 | 2 | |
| 12 | 8 | 10 | 11 |

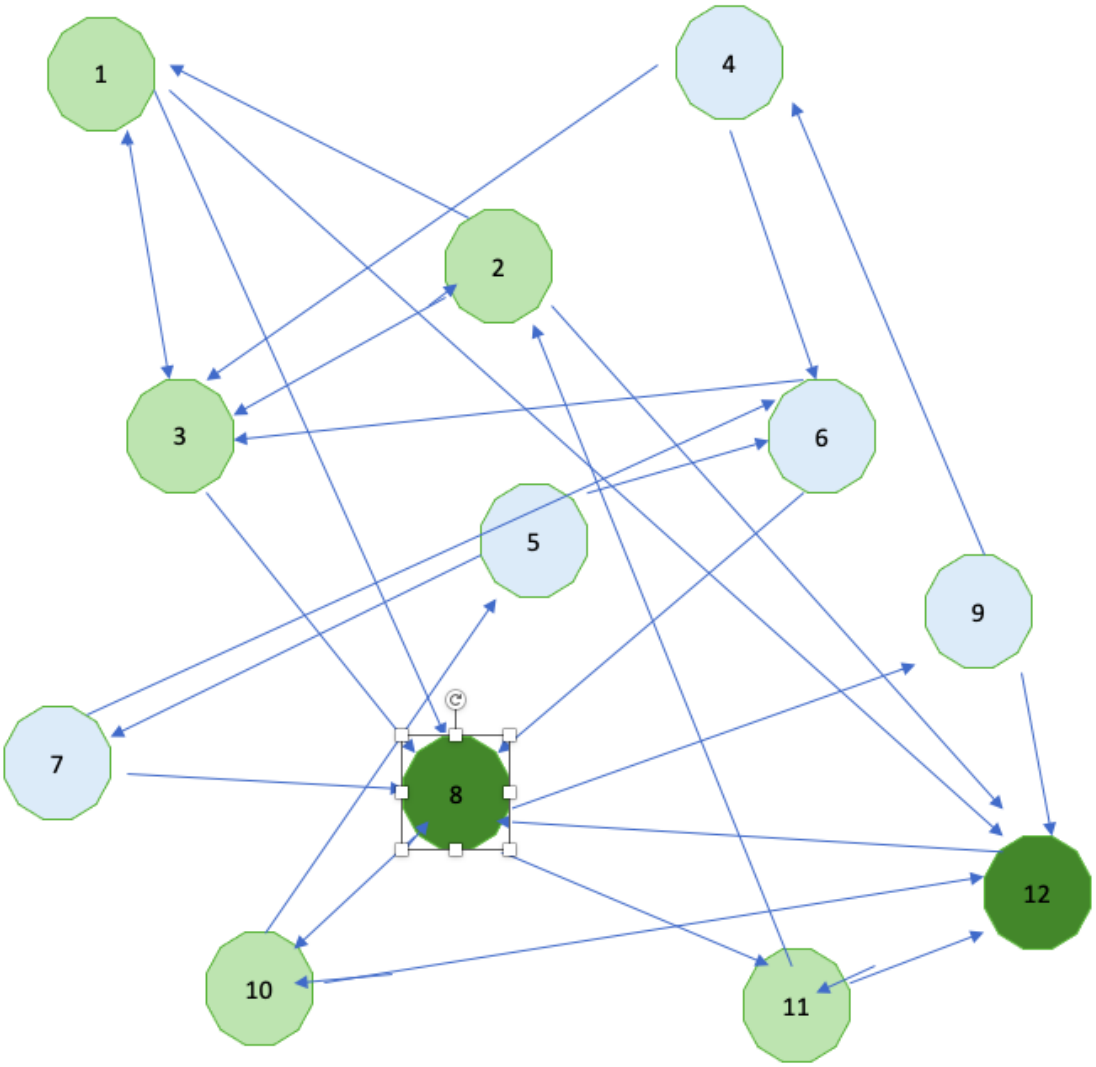
- Şekildeki ilk tabloda inDegree Arrayi görüyoruz. ilk satırında her bir kişiyi kaç kişinin takip ettiği bilgisi var. Sonrasında bu inDegree arrayde güncelleme yapacağız. İşleyiş şu şekilde olacak:
- Verilen m değeri 2 olduğu için inDegree Arrayde dolaşılacak ve 2 den küçük elemanlar bulunacak. Bulunan kişinin takip ettiği kişiler adjacency Matrisine gidilerek tespit edilecek ve bu kişilerin takipçi sayısı 1 adet azaltılacak. Bulunan elemanın takip ettiği kişilerin tamamında bu işlem yapıldıktan sonra bu kişinin inDegreesi -1 ile ifade edilecek ki tekrar bu kişilerde azaltma yapılmasına gerek kalmasın, çünkü zaten elendi. Bu şekilde inDegree Array güncellenecek ve son durumda 2 den büyük kişiler bizim için aday influencer olacak.

- Anlatılan şekilde aday olan influencerlar candidateInfluencer isimli bir arraye alınacak. Böylece kimin doğrudan ya da dolaylı kaç bağlantısı olduğu işlemi tüm kişiler için değil, sadece aday influencerlar için yapılmış olacak.

- Bu noktada bağlantı bulma kısmı için BFS algoritması kullanıyor olacağız.

```
void BFS(int v,int ** adjMatrix,int *queue,int *visited,int n,int f,int r){
    int i;
    for (i=1; i<=n; i++) {
        if (adjMatrix[i][v]!=0 &&visited[i]==0) {
            r+=1;
            queue[r]=i;
            visited[i]=1;
            printf("%d ",i);
            counter+=1;
        }
    }
    f+=1;
    if (f<=r) {
        BFS(queue[f], adjMatrix, queue, visited, n, f, r);
    }
}
```

- Recursive bir şekilde işleyen BFS algoritması sayesinde bir su dalgasıymışçasına ilgili node'a seviye seviye bağlantılı tüm nodeları bulmuş oluyoruz. Algoritmayı işlerken queue ve visited isimli iki adet array kullanıyoruz ki bağlantılı nodeları tek tek bulabilelim ve geçtiğimiz node'a tekrar uğramayalım. Aşağıdaki görselde bu işleyişi daha iyi görebiliriz.




- Örneğin 8 numaralı node için bfs uyguladığımızda aldığımız çıktı şu şekilde oluyor:

**Directly or indirectly followers of**
**1 3 6 7 10 12 2 4 5 9 11**

8 e 1.dereceden bağlı olan nodeları ilk başta görüyoruz. Yani 1-3-6-7-10-12 Sonrasında gelen 2- 4.. nodeları ise biri aracılığıyla bağlanmş. Mesela 2 node'u 3 numaralı node aracılığıyla; 4 node'u 6 numaralı node aracılığıyla gibi. Bu sayede tüm bağlantıları BFS mantığında yani seviye seviye yakalamış oluyoruz.

- İşleyişi bahsedilen BFS algortmasını tüm aday Influencerlarda uyguladığım fonksiyon:

```
void applyBFSforAllCandidates(int candidateNumber,int n,int *queue,int * visited, int
    **adjMatrix, int *candidateInfluencers,int * numberOfConnectedPersonToCandidates){
    int i,j,f,r,v;
    for (j=0; j<candidateNumber; j++) {
        for (i=1; i<=n; i++) {
            queue[i]=0;
            visited[i]=0;
        }
        f=1;
        r=1;
        counter=0;
        v=candidateInfluencers[j];
        queue[r]=v;
        visited[v]=1;
        printf("\nDirectly or indirectly followers of %d:\n",v);
        BFS(v, adjMatrix, queue, visited, n, f, r);
        numberOfConnectedPersonToCandidates[j]=counter;
        //printf("\nNumber of connected people %d.\n ",counter);
        //printf("-------------------------------------------------\n");
    }
}
```

Şekilde gördüğümüz fonksiyonla bu algoritmayı tüm adaylar için uyguluyor ve doğrudan ya da dolaylı hitap ettiği tüm kişileri bulmuş oluyoruz. Bir yandan bu kişilerin numarasını BFS fonksiyonu içinde yazdırırken bir yandan da numberOfConnectedPersonToCandidates isimli bir arrayde ulaşılan kişi sayısını tutuyoruz. Çünkü Y değişkeni ile verdiğimiz kısıtı burada uygulayacağız.

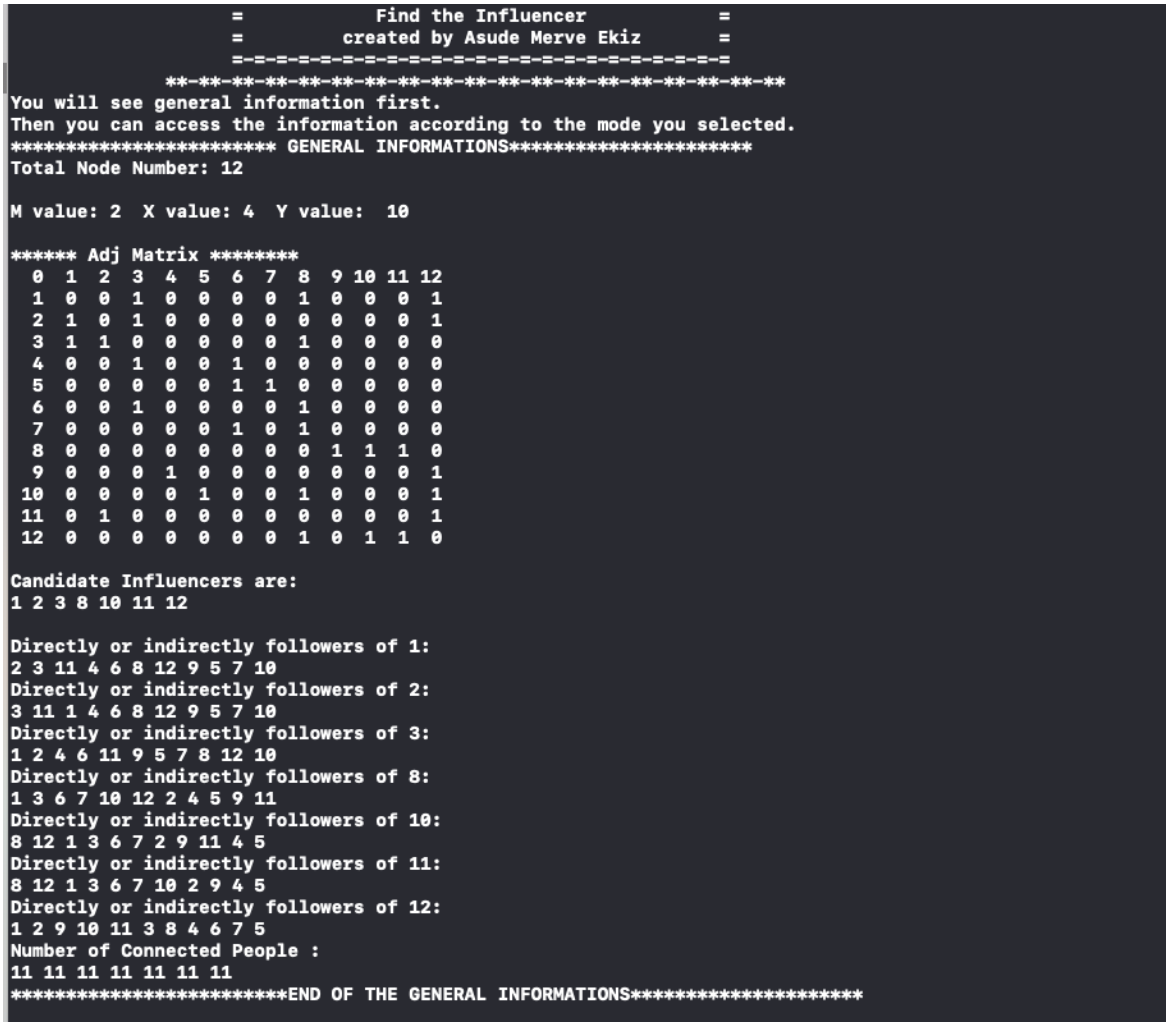
- connected person sayısını da bulduktan sonra geriye şartları sağlayan kişileri tespit etmek ve Influencer olanları bulmak kalıyor. Bu kısımda inDegree array'in son haline bakıp x ten büyük eşit olanları, numberOfConnectedPersonToCandidates isimli arraye bakıp bağlı olduğu kişi sayısı y den büyük eşit olanları seçiyor, ve bilgilerini yazdırıyoruz.
- yazdırma işlemi yapılırken de normal mod için *findInfluencersV1* fonksiyonunu, detay mod *findInfluencersV2* fonksiyonunu çağırıyoruz

- NOT: Verileri görebilmeniz adına mod seçiminden önce bazı genel bilgileri yazdırdım.

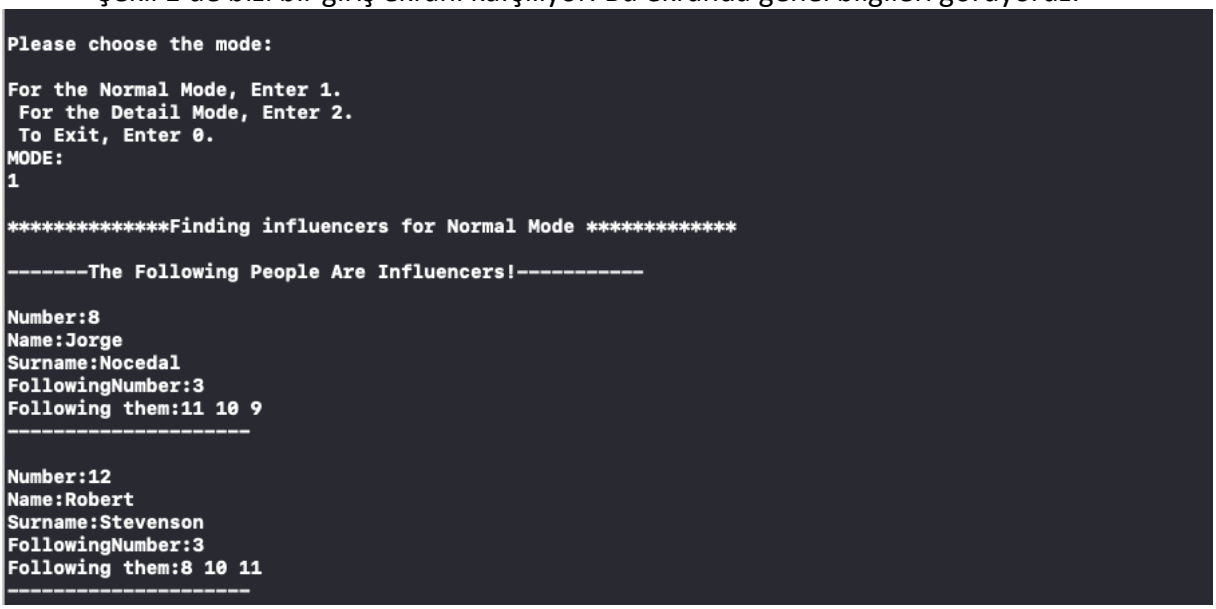
  Sonrasında seçilen mod 1 ise sadece Influencer olan kişilerin bilgilerini görüyoruz.
  2 ise
  - ◊ inDegree arrayin başlangıç ve bitiş durumunu,
  - ◊ influecer adaylarının bilgilerini ve bağlantılarını,
  - ◊ en son da m x y şartlarını sağlayan ve influencer oldukları tespit edilen kişileri ve bilgilerini

  görüyoruz.

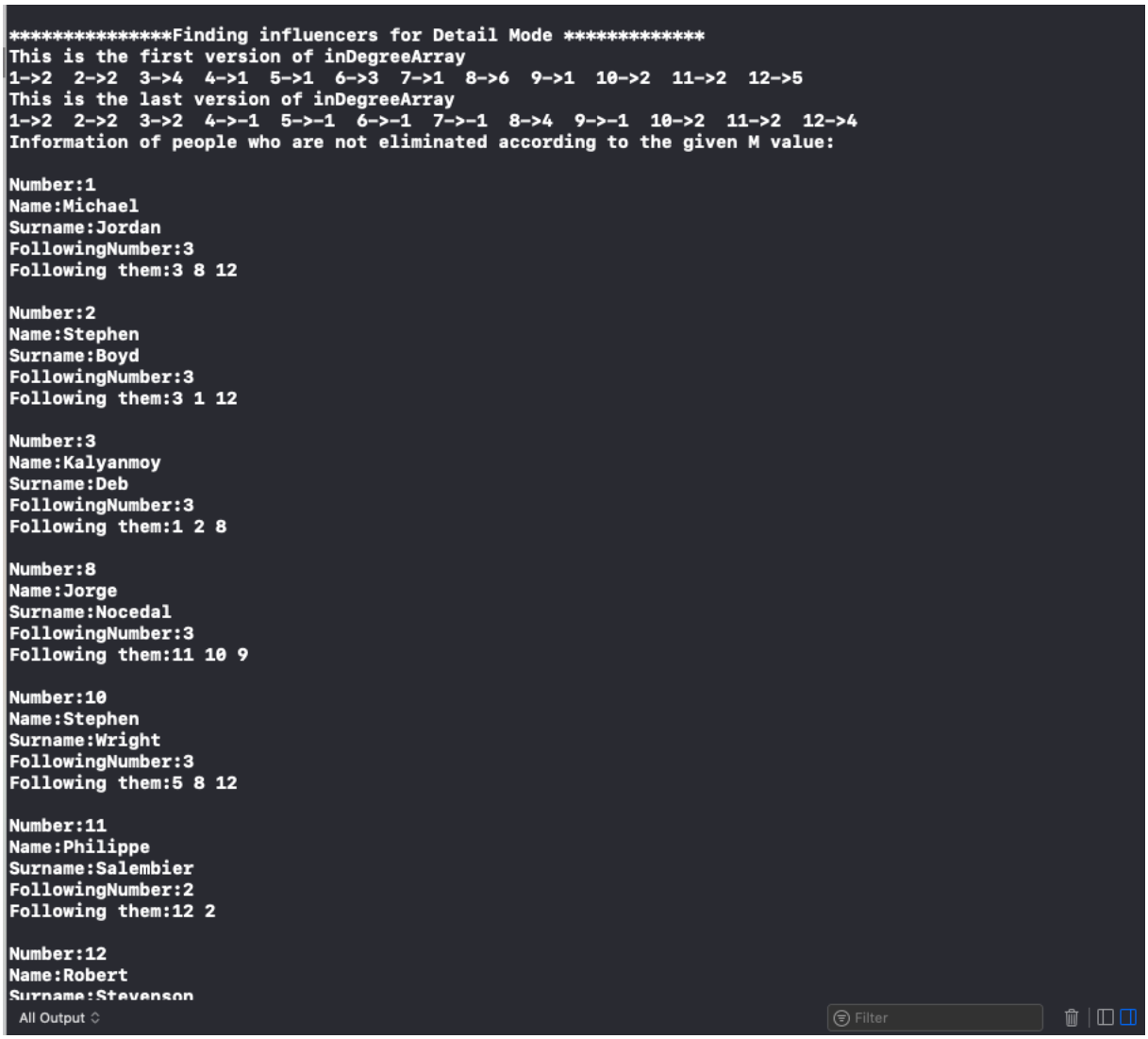
## Uygulama



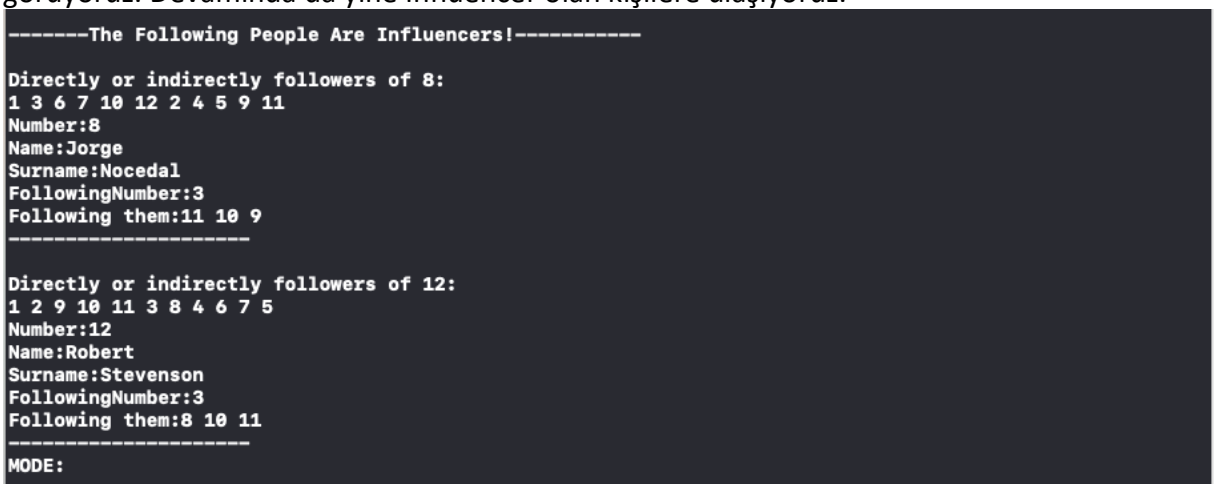

Şekil 1 de bizi bir giriş ekranı karşılıyor. Bu ekranda genel bilgileri görüyoruz.




Mod 1 seçtiğimizde aldığımız çıktı. Sadece bulunan 2 influencer'ın bilgileri var.




Mod 2 yi seçtiğimizde öce inDegree arrayin ilk ve son halini ve aday Influencerların bilgilerini görüyoruz. Devamında da yine Influencer olan kişilere ulaşıyoruz.




Programdan mod seçimine 0 girerek çıkabiliriz.

## Sonuç

Problemin çözümünde bizim için en önemli fonksiyonlardan biri BFS algoritması içeren fonksiyon.
BFS algoritması komşuluk matrisi üzerinde uygulandığında elde ettiğimiz karmaşıklık :

1. O(|V|* |E|) oluyor.  O(|V|²) şeklinde de ifade edebiliriz.

Kullandığım diğer fonksiyonlardan önemli olanlara baktığımızda da:

1. `void fillTheAdjMatrix(int **adjMatrix,PERSON * people,int m);`
   çift for kullandığımız için O(N^2)
2. `void updateTheinDegreeArray(int **adjMatrix, int * inDegreeArray,int n,int m)`
   çift döngü ile işlem yaptığımız için O(N^2)

3. `void findCandidateInfluencers(int* inDegreeArray, int * candidateInfluencers, int candidateNumber,int n,int m)`
   inDegreeArrayde dolaştığımız için O(N)

4. `void applyBFSforAllCandidates(int candidateNumber,int n,int *queue,int * visited, int **adjMatrix, int *candidateInfluencers,int * numberOfConnectedPersonToCandidates)`

   Çift for kullandığımız için O(N^2) gibi gözükse de forların ilkinin sonunda BFS de çağırdığımız için karmaşıklık 3.mertebeden oluyor.

5. `int findCandidateInfNumber(int * inDegreeArray, int n,int m)`
   yine sadece inDegree arrayde dolaştığımız için O(N)

   Kalan fonksiyonlar allocate , print ya da mod seçimine göre uygun çıktıyı alma amaçlarıyla oluşturulduğu ve her zaman yaptığımız genel işlemler olduğu için karmaşıklıklarına girilmedi.

**Anlatım video linki:**

https://youtu.be/MOz6bBmAlvY