



**T.C.  
GEBZE TEKNİK ÜNİVERSİTESİ**

**Bilgisayar Mühendisliği Bölümü**

**MAXIMUM  
INDUCED  
MATCHING**

**Nurullah GÜNDOĞDU**

**Danışman  
Doç. Dr. Didem GÖZÜPEK**

**Ocak, 2020  
Gebze,KOCAEL**





**T.C.  
GEBZE TEKNİK ÜNİVERSİTESİ**

**Bilgisayar Mühendisliği Bölümü**

# **MAXIMUM INDUCED MATCHING**

**Nurullah GÜNDOĞDU**

**Danışman**

**Doç. Dr. Didem GÖZÜPEK**

**Ocak, 2020 Gebze,KOCAELİ**



Bu çalışma ....../....../20.... tarihinde ařağıdaki jüri tarafından Bilgisayar Mühendisliğı Bölümünde Lisans Bitirme Projesi olarak kabul edilmiştir.

#### Bitirme Projesi Jürisi

Danışman Adı	Didem GÖZÜPEK	
Üniversite	Gebze Teknik Üniversitesi	
Fakülte	Mühendislik Fakültesi	

Jüri Adı	Uraz Cengiz TÜRKER	
Üniversite	Gebze Teknik Üniversitesi	
Fakülte	Mühendislik Fakültesi	

## **ÖNSÖZ**

Bu bitirme projesinin tamamlanmasında bana yardımcı olan, sorularımı ve sorunlarımı net bir şekilde cevaplayan çok değerli hocam Didem GÖZÜPEK'E katkısından ve zamanlarından dolayı teşekkürlerimi sunarım.

Ayrıca eğitimim süresince bana her konuda tam destek veren aileme ve bana hayatlarıyla örnek olan tüm hocalarıma saygı ve sevgilerimi sunarım.

**Ocak,2020**

**Nurullah GÜNDOĞDU**

# İÇİNDEKİLER

ÖNSÖZ.....	VI
İÇİNDEKİLER.....	VII
ŞEKİL LİSTESİ.....	IX
ÖZET.....	X
SUMMARY.....	XI
1. GİRİŞ.....	1
1.1 PROJENİN TANIMI.....	1
1.1.1 DÜĞÜM EŞLEŞMESİ.....	2
1.1.2 INDUCED EŞLEŞME.....	3
2. MALZEME VE YÖNTEM.....	4
2.1 YAZILIMSAL GEREKSİNİMLERİ.....	4
2.1.1 Python3.....	4
2.1.2 Graph_tool Kütüphanesi.....	4
2.1.3 PyQt5 Kütüphanesi.....	4
2.2 PROJENİN YAPISI VE TASARIMI .....	5
2.2.1 Greedy Algoritma.....	6
• Algoritma Tasarımı.....	6
• Algoritma Simülasyonu.....	6
• Algoritma Karmaşıklık Analizi.....	7
2.2.2 Sezgisel Algoritma.....	11
• Algoritma Tasarımı.....	12
• Algoritma Simülasyonu.....	12
• Algoritma Karmaşıklık Analizi.....	13
2.2.3 Masaüstü Uygulaması Arayüzü .....	13
• Dosya seçerek çalıştırma.....	13
• Random graph oluşturarak çalıştırma.....	14
3. TEST.....	14
4. SONUÇ.....	15
KAYNAKÇA.....	16

## ŞEKİL LİSTESİ

Şekil 1 Düğüm Eşleşmesi.....	2
Şekil 2 Induced Eşleşme.....	3
Şekil 3 Proje Genel Yapısı.....	4
Şekil 4 Arayüz.....	10
Şekil 5 Dosya Seçme.....	11
Şekil 6 Program Çıktısı.....	11
Şekil 7 Random Graf Oluşturma .....	12
Şekil 8 Random Graf Çıktısı.....	13



## ÖZET

Bu projede kullanıcıdan arayüz yardımıyla graf alan ve bu graf içindeki maximum sayıdaki induced eşleşmeleri bulan bir sezgisel algoritma ve greedy algoritma tasarlanmıştır. İki algoritmanın sonuçları karşılaştırılıp sezgisel algoritmanın greedy algoritmaya göre daha iyi sonuç verdiği kanıtlanmıştır. Kullanıcıdan dosya içinde tanımlanmış bir graf alabilen ve verilen vertex ve edge sayılarına göre random bir graf oluşturan arayüz tasarlanmıştır.

## **SUMMARY**

In this project, a heuristic algorithm and a greedy algorithm are designed to obtain graphs from the user and find the maximum number of induced matches within this graph. The results of the two algorithms were compared and the heuristic algorithm proved to be better than the greedy algorithm. The interface is designed to receive a defined graph from the user and generate a random graph according to the given vertex and edge numbers.

# 1. GİRİŞ

Bu rapor, GTÜ Bilgisayar Mühendisliği CSE-495 Bitirme Projesi dersi kapsamında hazırlanmaktadır.

Projede genel graflarda maximum sayıdaki induced eşleşmeleri bulmak için tasarlanmıştır. Proje sonunda algoritmanın grafi tasarlanan sezgisel algoritmayla induced eşleşmeleri  $O(n^3)$  zamanda bulması ve tasarlanan diğer greedy algoritmaya göre sezgisel algoritmanın %50 oranda daha iyi sonuç vermesi hedeflenmiştir.

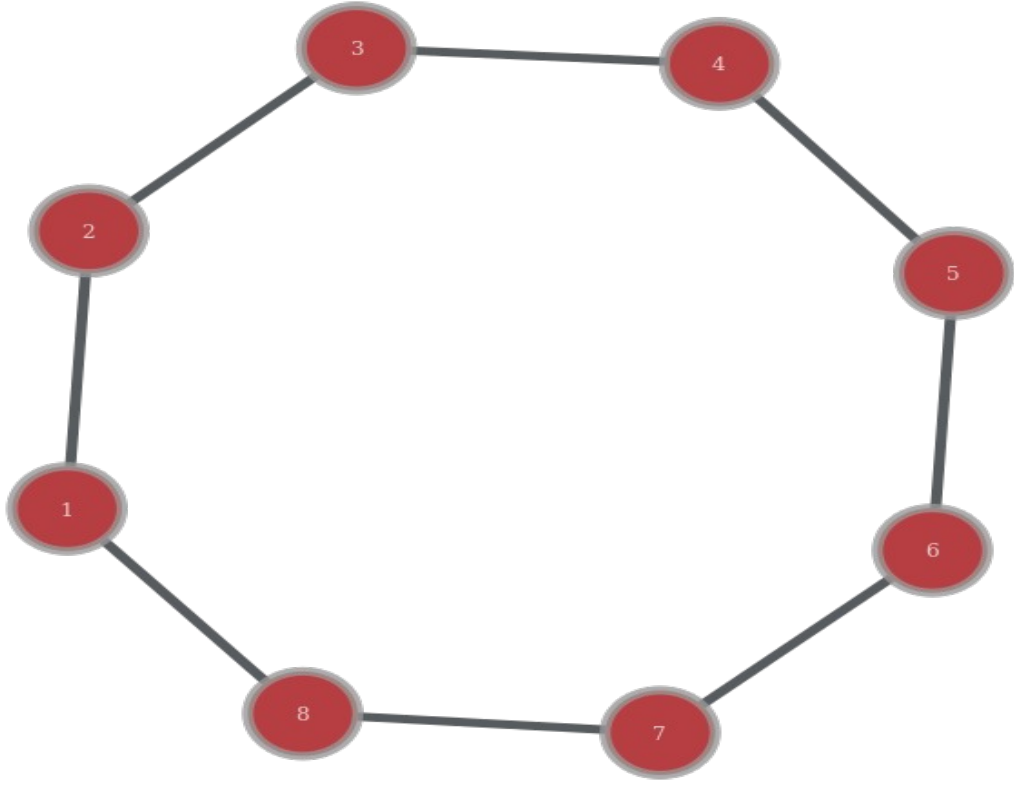
## 1.1 PROJENİN TANIMI

Bu projede belirtilen graf içindeki induced eşleşmeleri bulan bir sezisel algoritma tasarlanması hedeflenmiştir. Tasarlanan algoritmanın masaüstü uygulaması arayüzü ile kullanıcının dosya içinde tanımladığı grafın veya vereceği düğüm ve bağ sayısı ile random oluşturulan grafın içindeki maximum sayıdaki induced eşleşmeleri çıktı olarak kullanıcıya vermesi hedeflenmiştir.

### 1.1.1 DÜĞÜM EŞLEŞMESİ

Düğüm eşleşmesi grafın düğümlerinin ikili gruplar halinde eşleşmesidir. Her düğüm sadece bir eşleşme yapabilir. Şekil 1 de gösterildiği gibi düğümler eşleştirilebilir.

Eşleşmeler : (1, 2), (3, 4), (5, 6), (7, 8)

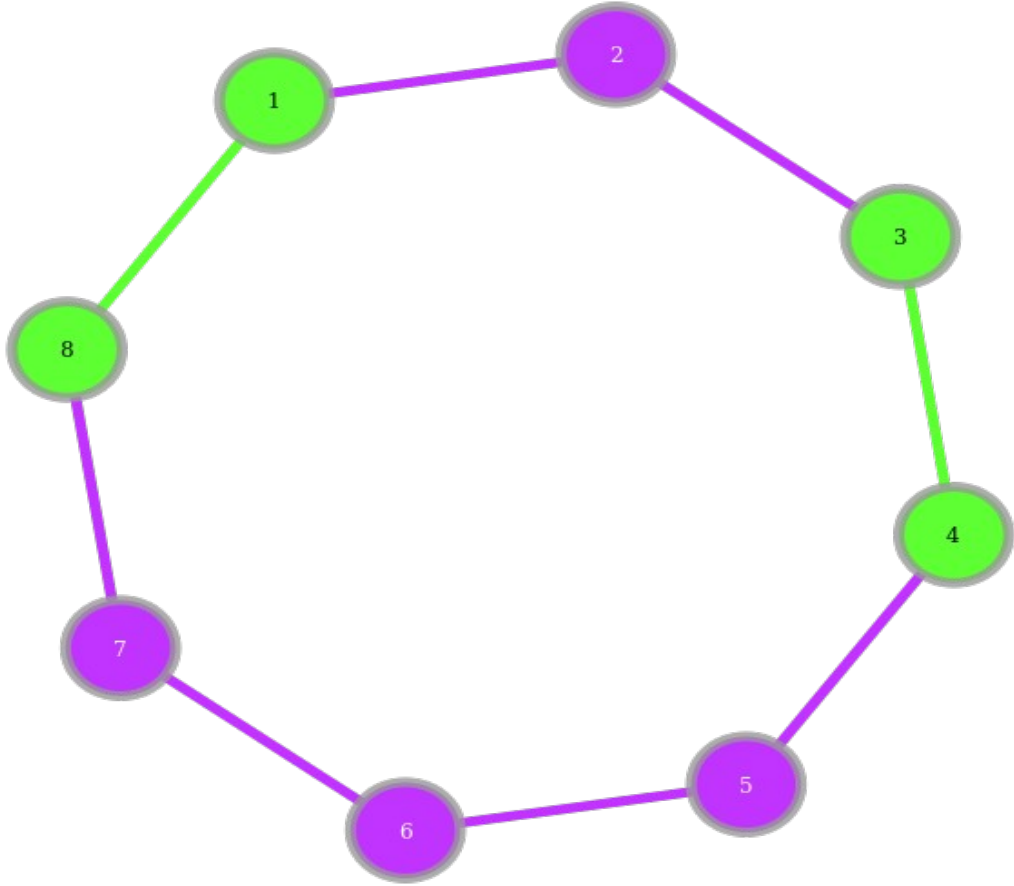


Şekil 1 Düğüm Eşleşmesi

### 1.1.2 INDUCED EŞLEŞME

Induced eşleme, grafın düğümlerinin özel bir durumla eşleşmesidir. Bu durum şekil 2 de gösterilmektedir. Induced eşleşmede eşleşme gruplarının düğümleri arasında bağlantı olmaması gerekmektedir. İki induced eşleşmiş düğüm arasında induced eşleşmesi olmayan bir düğüm olması gerekir.

Induced eşleşmeler : (1,8), (3,4)



Şekil 2 Induced Eşleşme

Bu raporun devamında projede kullanılan teknoloji ve araçlar, projenin yapısı, tasarımı, gerçekleştirilmesi, bulgular, sonuç, kaynaklar ve ekler bölümü yer almaktadır.

## **2. MALZEME VE YÖNTEM**

Bu bölümde projenin tasarım, geliştirme ve test süreçleri hakkında bilgiler yer almaktadır.

### **2.1 YAZILIMSAL GEREKSİNİMLER**

Projenin algoritma ve arayüz kısmı tamamen Python3 kullanılarak yazılmıştır. Python3 ' ün graph\_tool ve PyQt5 kütüphaneleri kullanılmıştır.

#### **2.1.1 Python3**

Diğer programlama dillerine göre kodlaması kolay ve işlem yapma konusunda çok fazla kolaylık sağladığından python3 programlama dili tercih edilmiştir.

#### **2.1.2 Graph\_tool Kütüphanesi**

Bu kütüphane sezgisel ve greedy algoritmayı tasarlamak için kullanılmıştır. Graph tool kütüphanesi graf tasarlamak ve graflar üzerinde işlem yapma konularına çok fazla kolaylık sağladığı için bu kütüphane tercih edilmiştir.

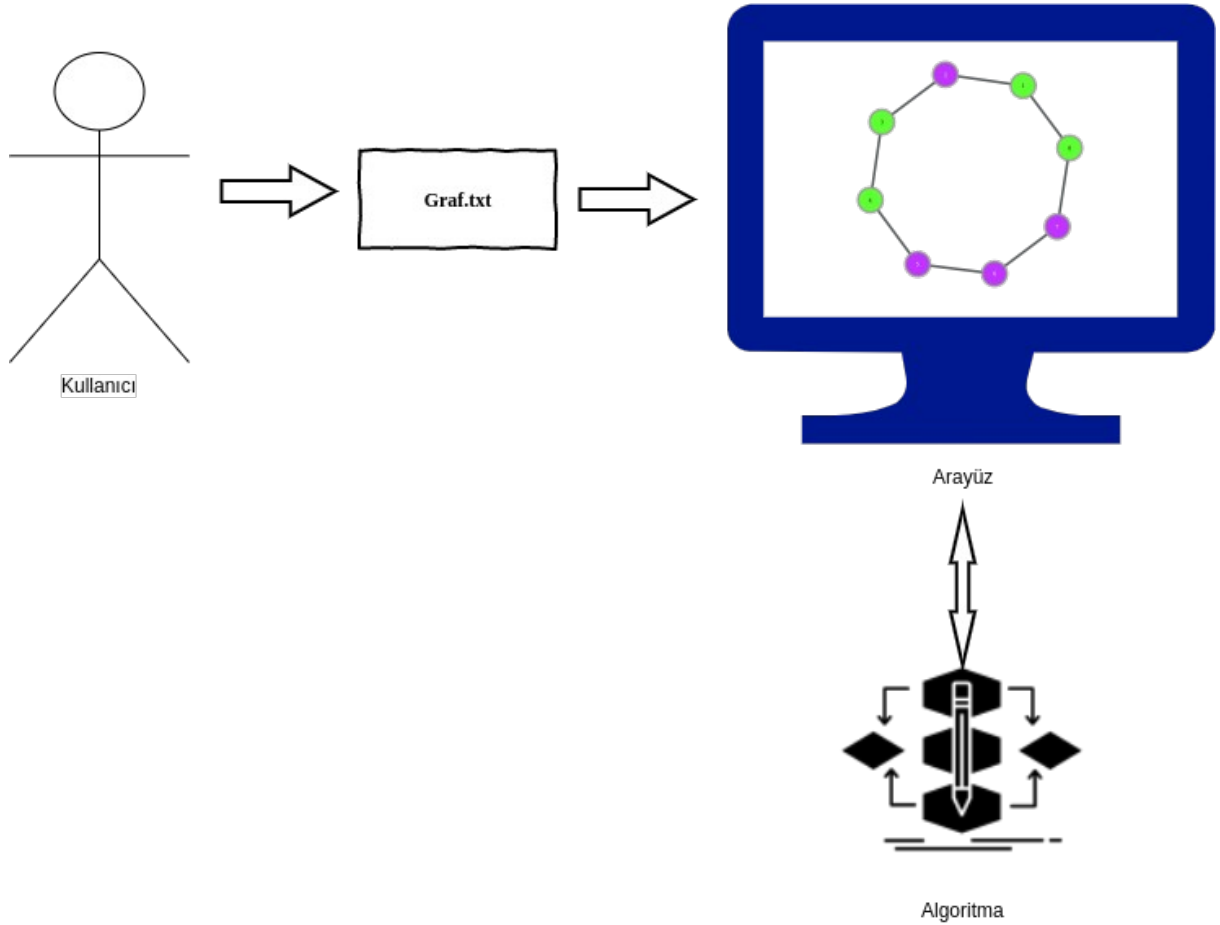
#### **2.1.3 PyQt5 Kütüphanesi**

Bu kütüphane arayüz tasarlamak için kullanılmıştır. PyQt5 kütüphanesi arayüz oluşturmada geniş bir kullanımı olduğundan ve kullanım kolaylığı sağladığından tercih edilmiştir.

## 2.2 PROJENİN YAPISI VE TASARIMI

Projenin genel yapısı şekil 3 de gösterildiği gibidir. Proje için iki algoritma ve kullanıcı arayüzü tasarlanmıştır. Algoritmaların birincisi greedy algoritma, ikincisi sezgisel özgün algoritma.

Projenin tasarımı için ilk önce kaynak araştırması yapılmıştır. Bu araştırmalar sonucunda sezgisel algoritma geliştirmek için fikir edinildi. Daha sonra greedy algoritma tasarlandı. greedy algoritmanın eksikleri giderilerek sezgisel algoritma tasarlandı. Algoritmalar Python3 dilinin graph\_tool kütüphanesi kullanılarak geliştirildi. Algoritmaları karşılaştırmak için veri seti oluşturuldu. Algoritmaların sonuçlarına göre karşılaştırma analizi yapıldı ve sezgisel algoritmanın greedy algoritmaya göre daha iyi sonuç verdiği kanıtlandı. Son olarak arayüz tasarlandı ve kullanıcıların kullanımına sunuldu. Arayüz Python3 programlama dilinin PyQt5 kütüphanesiyle yazıldı.



Şekil 3 Proje Genel Yapısı

## 2.2.1 Greedy Algoritma

- **Algoritma Tasarımı**

Bu algoritmada düğümler rastgele seçilerek gezilir ve eşleşme yapılır. Bütün düğümler gezilene kadar işlem devam eder.

Algoritma aşağıda belirtilen adımlarla çalışır:

1. Adım : Başlangıç düğümü grafın düğümleri arasından rastgele seçilir.
2. Adım : Seçilen düğümün bağlı olduğu düğümlerden birisi rastgele seçilir.
3. Adım : Seçilen düğümler induced eşleşme yapılır.
4. Adım : Graf içinde gezilmemiş düğüm kaldıysa 1. adıma geri dönlür.

- **Algoritma Simülasyonu**

Greedy algoritma sudo kodu aşağıda gösterildiği gibidir.

```
function vertex_checker(graph)
    Vertices ← graph's vertex

    for i ← vertices[0] to vertices[n]
        If vertex not induced
            call edge_checker(vertices[i], graph)
End function

Function edge_checker(vertex, graph)
    Edges ← vertex's edges
    Num ← 0

    for i ← Edges[0] to Edges[n]
        If Edges not induced
            Call change_prop_int(vertex, i, edges, graph)
End function

Function change_prop_int(vertex, edge, edges, graph)
    Vertex ← make it induced
    edge ← make it induced
    for i ← Edges[0] to Edges[n]
        If Edges not induced
            i ← make it induced
    Temp ← edge's edge
    for i ← temp[0] to temp[n]
        If Edges not induced
            i ← make it induced
end function
```



## • Algoritma Karmaşıklık Analizi

Algoritmanın Best case durumunda sonuç vermesi için grafın düğümlerinin hepsinin birbiriyle bağı olması gerekmektedir.

Bu durumda vertex\_checker fonksiyonunun döngüsü  $V$ (düğüm sayısı) kadar, edge\_checker fonksiyonu 1 defa ve change\_prop\_int fonksiyonu döngüsü  $E$ (toplam bağı) kadar döner.

Bu durumda Best case =  $O(V + E)$  olur.

Algoritmanın Worst case durumunda sonuç vermesi için grafın düğümleri birbirlerine çember halinde bağı yapmaları gerekmektedir.

Bu durumda vertex\_checker fonksiyonunun döngüsü  $V$ (düğüm sayısı) kadar, edge\_checker fonksiyonu  $\log V$  defa ve change\_prop\_int fonksiyonu döngüsü  $E$ (toplam bağı) kadar döner.

Bu durumda Worst case =  $O(V * \log V * E)$  olur.

## 2.2.2 Sezgisel Algoritma

### • Algoritma Tasarımı

Bu algorithmada düğümler en az bağı sahip olandan en çok sahip olana doğru sıralanır. Daha sonra ilk düğüm seçilir ve en az bağı sahip olan komşu düğümüyle eşleştirilir. Bütün düğümler gezilene kadar bu işlemler tekrar edilir.

Algoritma aşağıda belirtilen adımlarla çalışmaktadır.

1. Adım : Grafın düğümleri en az bağı sahip olandan en çok sahip olana doğru sıralanır.
2. Adım : Başlangıç düğümü sıralanmış düğümlerin ilki seçilir.
3. Adım : Seçilen düğümün bağı olduğu düğümlerde aynı şekilde sıralanır. en az bağı olan seçilir.
2. Adım : Sıralanan düğümlerin ilki seçilir.
3. Adım : Seçilen düğümler induced eşleşme yapılır.
4. Adım : Graf içinde gezilmemiş düğüm kaldıysa 1. adıma geri dönülür ve sıradaki en az bağı düğümden devam edilir.

- **Algoritma Simülasyonu**

Algoritmanın sudo kodu aşağıda gösterildiği gibidir.

```
function vertex_checker(graph, index)
    Vertexes  $\leftarrow$  sorted graph's vertex

    If vertex[index] not induced
        call edge_checker(vertexes[i], graph)

    If index not equal size of vertex
        Call vertex_checker(graph, index + 1)
End function

Function edge_checker(vertex, graph)
    Edges  $\leftarrow$  vertex's edges
    sorted_edges  $\leftarrow$  call min_edge(graph,edges)

    for i  $\leftarrow$  sorted_edges[0] to sorted_edges[n]
        If i not induced
            Call change_prop_int(vertex, i, edges, graph)
End function

Function min_edge(graph, edges)
    Dic  $\leftarrow$  {}
    Dic  $\leftarrow$  edges

    Return sorted dic
End function

Function change_prop_int(vertex, edge, edges, graph)
    Vertex  $\leftarrow$  make it induced
    edge  $\leftarrow$  make it induced

    for i  $\leftarrow$  Edges[0] to Edges[n]
        If Edges not induced
            i  $\leftarrow$  make it induced
    Temp  $\leftarrow$  edge's edge
    for i  $\leftarrow$  temp[0] to temp[n]
        If Edges not induced
            i  $\leftarrow$  make it induced
end function
```

- **Algoritma Karmaşıklık Analizi**

Algoritmanın Best case durumunda sonuç vermesi için grafın düğümlerinin hepsinin birbiriyle bağı olması gerekmektedir.

Bu durumda düğümlerin sıralama işlemi  $V \log V$ , vertex\_checker fonksiyonunun döngüsü  $V$  (düğüm sayısı) kadar, bağların sıralama işlemi  $E \log E$ , edge\_checker fonksiyonu 1 defa ve change\_prop\_int fonksiyonu döngüsü  $E$  (toplam bağ) kadar döner.

Bu durumda Best case =  $O(V \log V + V + E \log E + E)$  olur.

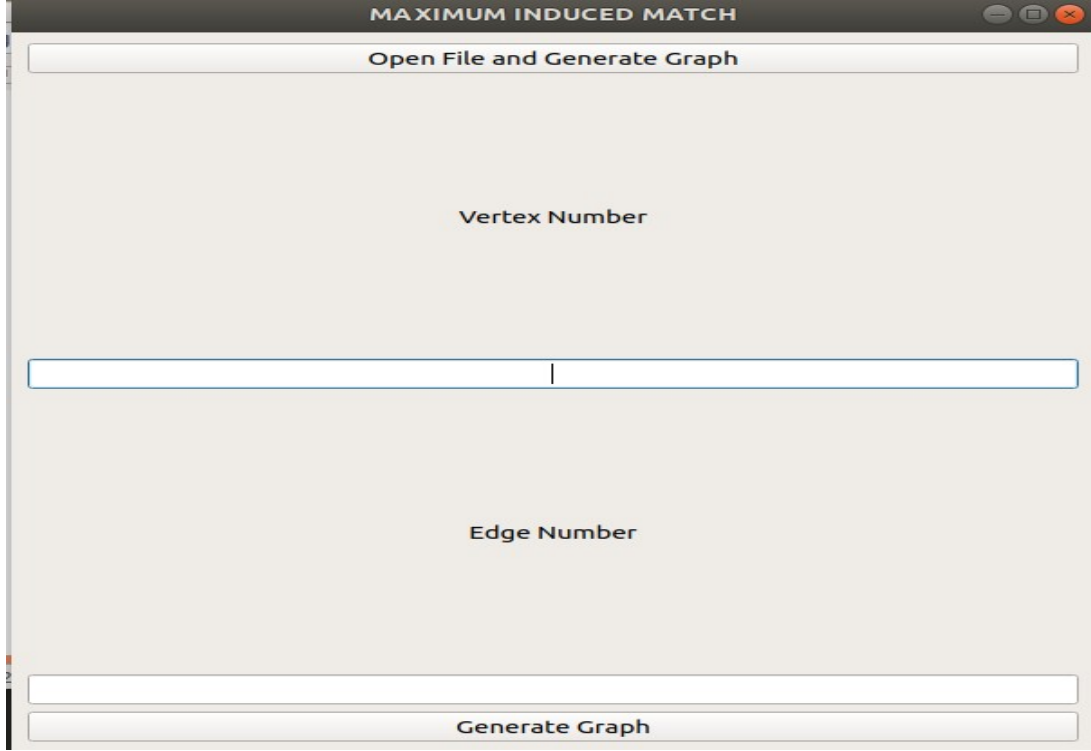
Algoritmanın Worst case durumunda sonuç vermesi için grafın düğümleri birbirlerine çember halinde bağ yapmaları gerekmektedir.

Bu durumda düğümlerin sıralama işlemi  $V \log V$ , vertex\_checker fonksiyonunun döngüsü  $V$  (düğüm sayısı) kadar döner. Bağların sıralama işlemi  $E \log E$ , edge\_checker fonksiyonu  $\log V$  defa ve change\_prop\_int fonksiyonu döngüsü 2 defa döner.

Bu durumda Worst case =  $O(V \log V + V * \log V * E \log E * 2)$  olur.

### 2.2.3 Masaüstü Uygulaması Arayüzü

Arayüz Şekilde 4 de görüldüğü gibidir. Arayüz iki farklı şekilde çalışır. Birincisi kullanıcının grafın yüklü olduğu dosyayı çalıştırması. İkincisi ise girilen düğüm sayısı ve bağ sayısına göre random bir graf oluşturup çalışması.



Şekil 4 Arayüz

- **Dosya seçerek çalıştırma**

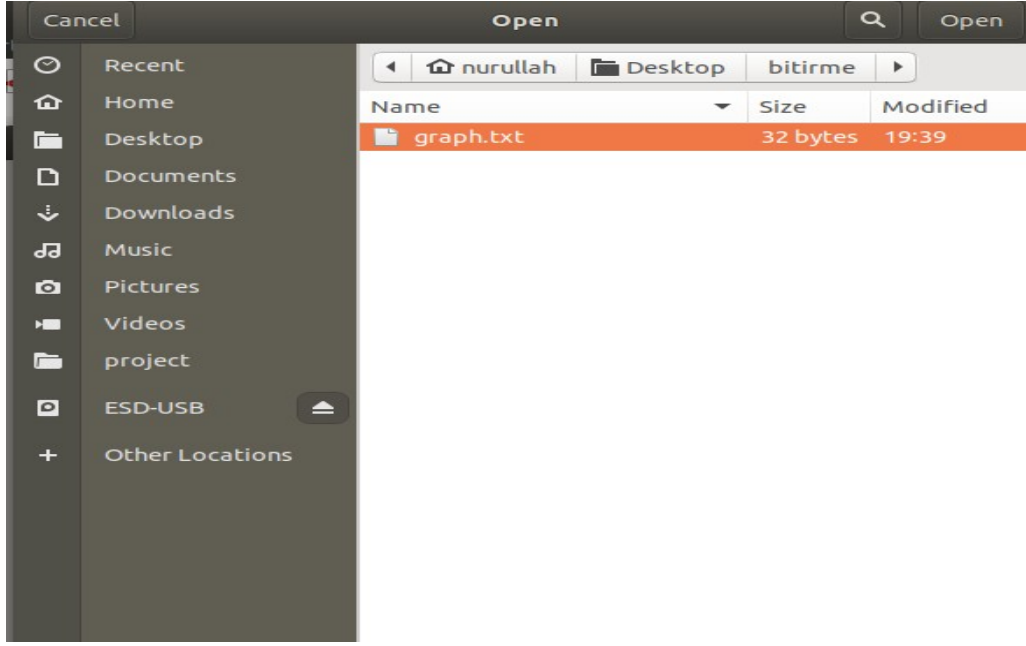
Kullanıcı şekil 4’te görünen “Open file and Generate” butonuna tıklayarak kendi hazırladığı input dosyasını seçebilir ve program otomatik olarak algoritmayı çalıştırır.

Şekil 5’ te dosya seçim bölümünü görebilirsiniz.

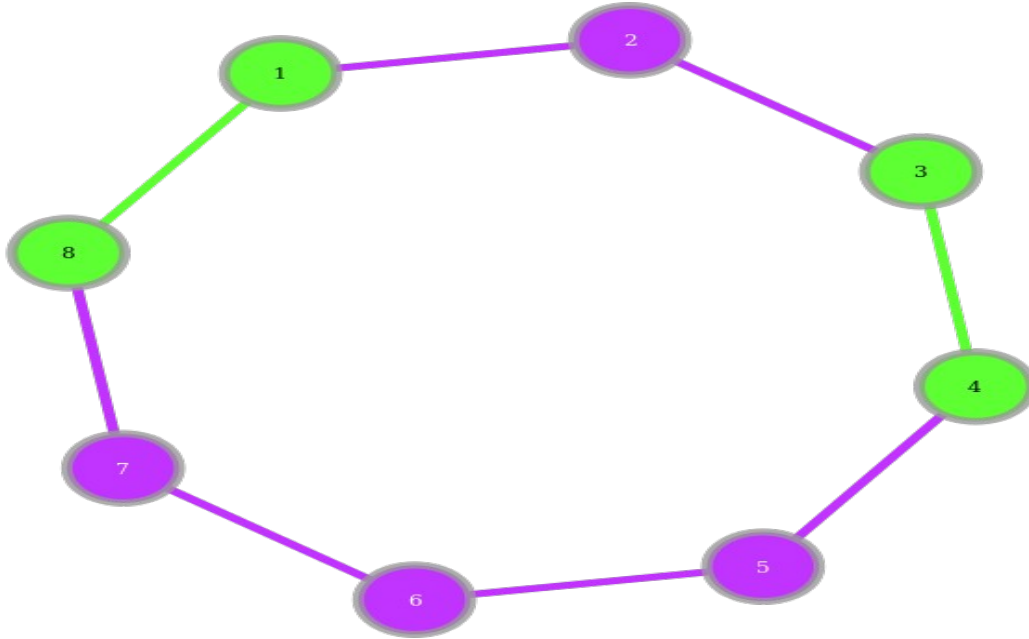
Program çalıştıktan sonra Şekil 6’ da ki çıktıyı verir.

Şekil üzerinde induced eşleşmeler yeşil renkte görünmektedir.

Eşleşmeler : (8, 1), (3,4)



Şekil 5 Dosya Seçme



Şekil 6 Program Çıktısı

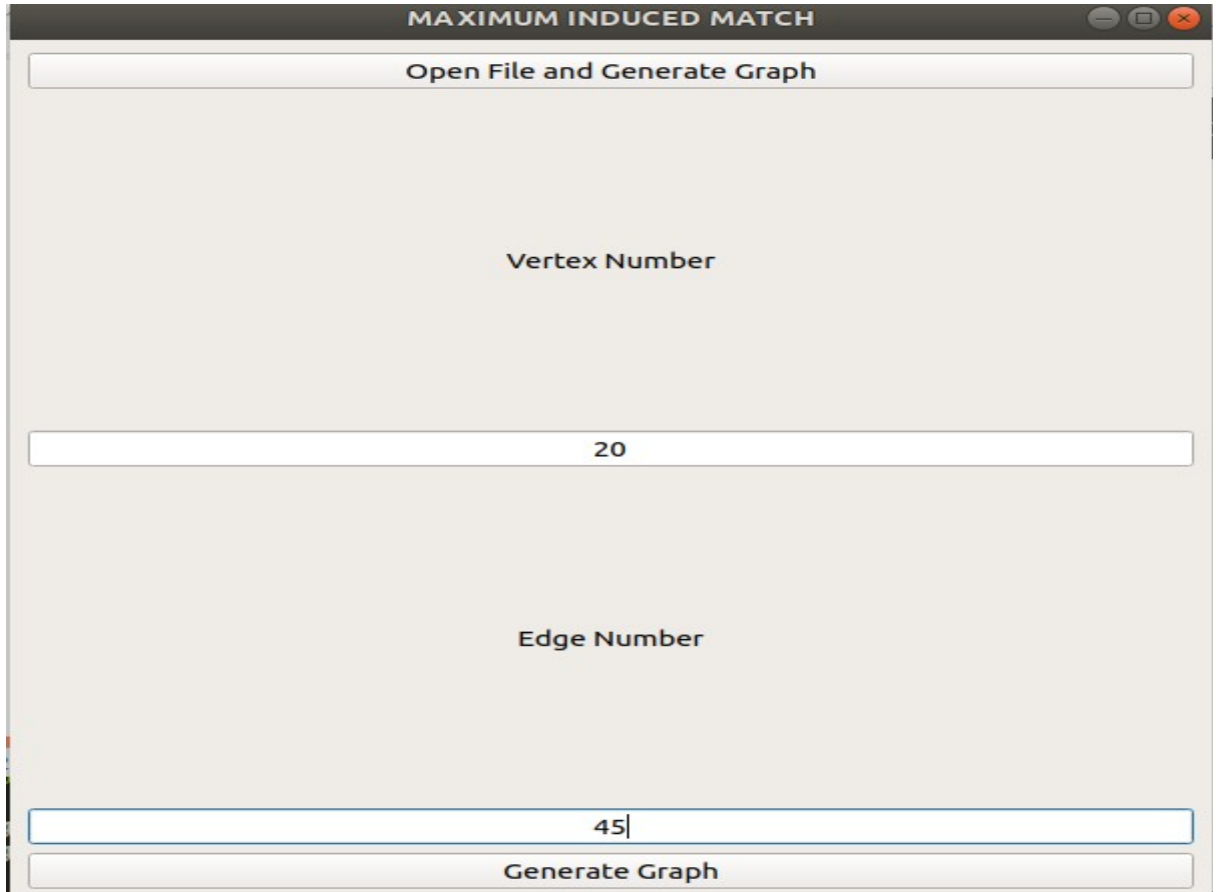
- **Random graph oluřturarak alıřtırma**

Kullanıcı řekil 7 de ki gibi “Vertex number” blmne dğm sayısını, “Edge Number” blmne toplam bağ sayısını girdikten sonra “Generate Graph” butonuna tıklayarak random graf oluřturup programı otomatik olarak alıřtırabilir.

Program alıřtıktan sonra řekil 8’ de ki ıktıyı verir.

řekil zerinde induced eřleřmeler yeřil renkte grnmektedir.

Eřleřmeler : (8, 7), (3,5), (6,12), (19,0), (2,17)



MAXIMUM INDUCED MATCH

Open File and Generate Graph

Vertex Number

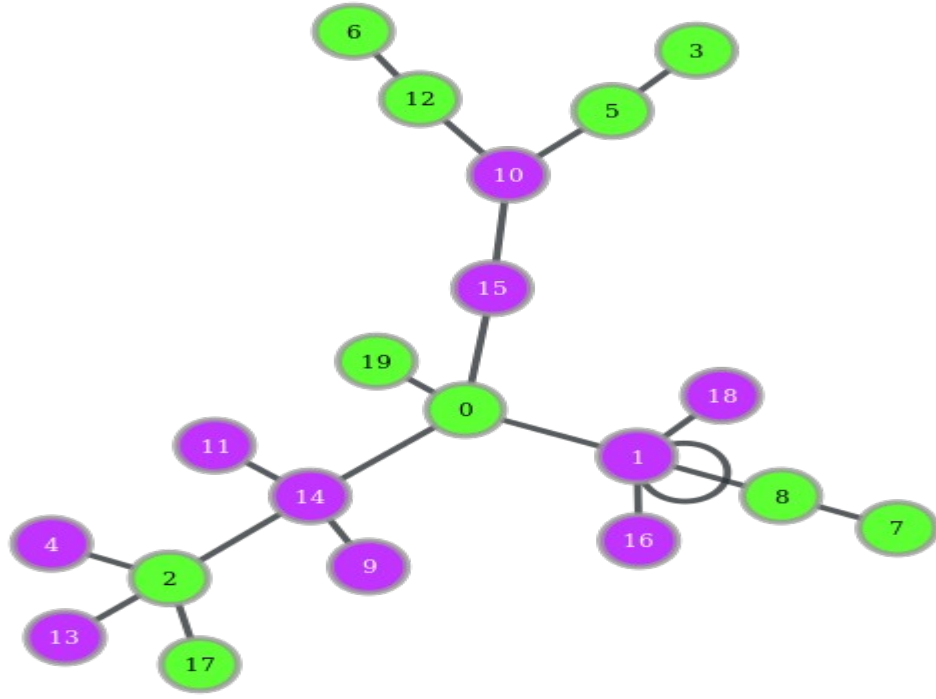
20

Edge Number

45

Generate Graph

řekil 7 Random Graf Oluřturma



Şekil 8 Random Graf Çıktısı

### 3. TEST

Sezgisel ve greedy algoritma random üretilen graflarla test edildi. Aşağıda verilen tabloda graflara göre buldukları induced eşleşme sayısı gösterilmektedir.

Graph	Vertex	Edge	Greedy	Sezgisel
1. Graph	100	100	18	23
2. Graph	50	69	12	14
3. Graph	149	183	32	34
4. Graph	65	80	14	15
5. Graph	96	120	19	21
6. Graph	87	87	18	20
7. Graph	36	55	9	9
8. Graph	72	85	15	18
9. Graph	49	61	10	11

Yapılan testlerin sonucunda sezgisel algoritmanın greedy algoritmaya göre daha iyi performans gösterdiği kanıtlanmış oldu.



## 4. SONUÇ

Projede sezgisel ve greedy algoritma tasarlandı. Oluşturulan graf veri seti iki algoritma ile denendi. Çıkan sonuçlar doğrultusunda sezgisel algoritmanın greedy algoritmaya göre daha iyi sonuçlar verdiği saptandı. Sonuçlar doğrultusunda sezgisel algoritmanın daha iyi olduğu kanıtlandı.

Sezgisel algoritma arayüz yardımıyla kullanıma sunuldu. Verilen grafların induced eşleşmelerini graf üzerinde çizerek göstermesi sağlandı.

## KAYNAKÇA

- [1] Krishnamurthy, Chandra Mohan, and R. Sritharan. "Maximum induced matching problem on hhd-free graphs." *Discrete Applied Mathematics* 160.3 (2012): 224-230.
- [2] Duckworth, William, David F. Manlove, and Michele Zito. "On the approximability of the maximum induced matching problem." *Journal of Discrete Algorithms* 3.1 (2005): 79-91.
- [3] <https://github.com/gephi/gephi/wiki/Datasets>