Veri Yapıları ve Algoritmalar Ödev-4

Graflar ile Algoritma Tasarımı

Öğrenci Adı: İsa Koçan

Öğrenci Numarası: 22011056

Dersin Eğitmeni: Göksel Biricik – GR2

Video Linki: https://youtu.be/g7AxAHHj9AI

1. Problemin Çözümü (Algoritma)

Bu çalışmada, verilen yönsüz ve kenar ağırlıksız bir grafta, kullanıcı tarafından belirlenen ancak program tarafından başlangıçta bilinmeyen bir "ödül" düğümünü, "X düğümü ödül düğümüne komşu mu?" sorusunu kullanarak olası en az sayıda soru ile bulma problemi ele alınmıştır.

Graf yapıları, sorgulama verimliliği ve implementasyon basitliği göz önünde bulundurularak komşuluk matrisi kullanılarak temsil edilmiştir. Bu temsil, iki düğümün komşu olup olmadığının O(1) zaman karmaşıklığında kontrol edilmesine olanak tanımaktadır. Diğer bir alternatif olan komşuluk listesi, özellikle seyrek graflarda bellek açısından daha verimli olabilse de, mevcut problemdeki temel "komşu mu?" sorgusu için komşuluk matrisi doğrudan ve hızlı bir çözüm sunmuştur.

Algoritmanın temel stratejisi, her adımda sorulacak en iyi soruyu belirleyerek aday ödül düğümü sayısını en hızlı şekilde azaltmaktır. Çözüm adımları şu şekildedir:

1. Graf, dosyadan okunarak komşuluk matrisi olarak belleğe yüklenmiştir.
2. Kullanıcıdan, "ödül düğümü" (hedef) bilgisi alınmıştır.
3. Başlangıçta, tüm düğümler potansiyel "ödül düğümü" adayı olarak kabul edilmiştir.
4. Aday sayısı bire düşene veya maksimum soru sınırına (graf boyutu kadar) ulaşılana kadar bir döngü içerisinde aşağıdaki adımlar tekrarlanmıştır:
    - **En İyi Sorulacak Düğümün Seçimi (findBestNode):** Hangi düğümün komşuluğu sorgulandığında en kötü durumda kalan aday sayısını en aza indireceği hesaplanmıştır. Bu, her potansiyel soru düğümü için "Evet" ve "Hayır" cevap senaryolarındaki aday sayıları değerlendirilerek yapılmıştır.
    - **Sorunun Sorulması ve Adayların Güncellenmesi (askXisNeighbor):** Seçilen düğüm için "Hedef ödüle komşu mu?" sorusu simüle edilmiştir. Cevaba göre ("Evet" ise sadece seçilenin komşuları olan adaylar kalmış, "Hayır" ise seçilenin komşuları olan adaylar elenmiştir) aday listesi güncellenmiştir.
    - Her sorudan sonra güncel aday listesi kullanıcıya gösterilmiştir.
5. Döngü sonunda, aday sayısı bire düşmüşse bu aday "bulunan ödül düğümü" olarak ilan edilmiştir. Aksi durumda, kalan potansiyel adaylar listelenmiştir.

Program sonlandırılmadan önce, dinamik olarak ayrılan bellek alanları serbest bırakılmıştır.

2. Karşılaşılan Sorunlar

Algoritma geliştirme sürecinde karşılaşılan başlıca zorluklar ve gözlemler şunlardır:

- **Optimal Soru Stratejisinin Geliştirilmesi:** findBestNode fonksiyonunda, her adımda en kötü durumu minimize edecek soruyu belirleme stratejisinin tasarlanması ve verimli bir şekilde kodlanması üzerinde durulmuştur.
- **Aday Listesinin Etkin Yönetimi:** Sorulara alınan cevaplara göre aday düğüm listesinin ve sayısının doğru şekilde güncellenmesi (askXisNeighbor fonksiyonu) dikkat gerektirmiştir.
- **Algoritmanın Sınırlamaları (Simetrik Komşuluk Durumları):** Testler sırasında, bazı özel graf yapılarında ve ödül düğümü seçimlerinde algoritmanın ödül düğümünü tek bir adaya indiremediği gözlemlenmiştir. Örneğin, ödevde verilen **2. örnek graf için 0 ve 1 numaralı düğümler; 3. örnek graf için ise 1 ve 3 numaralı düğümler ödül düğümü olarak seçildiğinde**, bu düğüm çiftlerinin komşuluk setleri birbirinin aynısı olduğundan, "komşu mu?" sorusu bu iki aday arasında bir ayrım yapamamaktadır. Bu durumda, algoritma her iki düğümü de potansiyel aday olarak tutmakta ve soru sayısı sınırına ulaşıldığında bu ikiliyi potansiyel adaylar olarak raporlamaktadır. Bu, sadece komşuluk bilgisine dayalı bir sorgulama stratejisinin doğal bir sınırlamasıdır.
- **Dinamik Bellek Yönetimi:** Komşuluk matrisi ve aday listesi için malloc ve free kullanımlarında bellek sızıntılarını önlemeye özen gösterilmiştir.

3. Ekran Görüntüleri

Bu bölümde, programın ödevde belirtilen üç farklı girdi grafı (G1_KMat.txt, G2_KMat.txt, G3_KMat.txt) ve her bir graf için belirtilen farklı ödül düğümleri ile çalıştırılması sonucu elde edilen ekran çıktıları sunulmuştur. Özellikle, yukarıda bahsedilen simetrik komşuluk durumlarını içeren testlerin çıktılarına da yer verilmiştir.

Örnek Çıktı 1: G1_KMat.txt (Ödül Düğümü: 5)

```
Lutfen test etmek istediginiz grafi secin (1, 2, veya 3): 1

Grafin Komsuluk Matrisi:
0 1 0 0 0 0
1 0 1 0 1 0
0 1 0 1 0 0
0 0 1 0 0 0
0 1 0 0 0 1
0 0 0 0 1 0

Hangi dugum hedef olsun: 5


Soru 1: [1] 5'ye komsu mu?
Cevap: Hayir
Adaylar: 1 3 5

Soru 2: [0] 5'ye komsu mu?
Cevap: Hayir
Adaylar: 3 5

Soru 3: [2] 5'ye komsu mu?
Cevap: Hayir
Adaylar: 5

Hedef dugum bulundu: 5

[i]-> Bellek temizlendi!
--------------------------------
Process exited after 2.754 seconds with return value 0
Press any key to continue
```

Örnek Çıktı 2: G2_KMat.txt (Ödül Düğümü: 0 veya 1 - Simetrik Durum Çıktısı)

```
Lutfen test etmek istediginiz grafi secin (1, 2, veya 3): 2

Grafin Komsuluk Matrisi:
0 0 1 1 1
0 0 1 1 1
1 1 0 1 1
1 1 1 0 1
1 1 1 1 0

Hangi dugum hedef olsun: 0


Soru 1: [0] 0'ye komsu mu?
Cevap: Hayir
Adaylar: 0 1

Soru 2: [0] 0'ye komsu mu?
Cevap: Hayir
Adaylar: 0 1

Soru 3: [0] 0'ye komsu mu?
Cevap: Hayir
Adaylar: 0 1

Soru 4: [0] 0'ye komsu mu?
Cevap: Hayir
Adaylar: 0 1

Soru 5: [0] 0'ye komsu mu?
Cevap: Hayir
Adaylar: 0 1

Hedef dugum bulunamadi! Potensiyel Adaylar: 0 1

[i]-> Bellek temizlendi!
--------------------------------
Process exited after 3.886 seconds with return value 0
Press any key to continue . . .
```

Örnek Çıktı 3: G3_KMat.txt (Ödül Düğümü: 1 veya 3 - Simetrik Durum Çıktısı)

```
Lutfen test etmek istediginiz grafi secin (1, 2, veya 3): 3

Grafin Komsuluk Matrisi:
0 1 0 1 0 0
1 0 0 0 1 0
0 0 0 0 0 1
1 0 0 0 1 0
0 1 0 1 0 1
0 0 1 0 1 0

Hangi dugum hedef olsun: 1


Soru 1: [4] 1'ye komsu mu?
Cevap: Evet
Adaylar: 1 3 5

Soru 2: [0] 1'ye komsu mu?
Cevap: Evet
Adaylar: 1 3

Soru 3: [0] 1'ye komsu mu?
Cevap: Evet
Adaylar: 1 3

Soru 4: [0] 1'ye komsu mu?
Cevap: Evet
Adaylar: 1 3

Soru 5: [0] 1'ye komsu mu?
Cevap: Evet
Adaylar: 1 3

Soru 6: [0] 1'ye komsu mu?
Cevap: Evet
Adaylar: 1 3

Hedef dugum bulunamadi! Potensiyel Adaylar: 1 3

[i]-> Bellek temizlendi!
------------------------------
```

Örnek Çıktı 4: Hepsi için normal durum

```
Lutfen test etmek istediginiz grafi secin (1, 2, veya 3): 1

Grafin Komsuluk Matrisi:
0 1 0 0 0 0
1 0 1 0 1 0
0 1 0 1 0 0
0 0 1 0 0 0
0 1 0 0 0 1
0 0 0 0 1 0

Hangi dugum hedef olsun: 2


Soru 1: [1] 2'ye komsu mu?
Cevap: Evet
Adaylar: 0 2 4

Soru 2: [3] 2'ye komsu mu?
Cevap: Evet
Adaylar: 2

Hedef dugum bulundu: 2

[i]-> Bellek temizlendi!
--------------------------------
```

```
Lutfen test etmek istediginiz grafi secin (1, 2, veya 3): 2

Grafin Komsuluk Matrisi:
0 0 1 1 1
0 0 1 1 1
1 1 0 1 1
1 1 1 0 1
1 1 1 1 0

Hangi dugum hedef olsun: 3


Soru 1: [0] 3'ye komsu mu?
Cevap: Evet
Adaylar: 2 3 4

Soru 2: [2] 3'ye komsu mu?
Cevap: Evet
Adaylar: 3 4

Soru 3: [3] 3'ye komsu mu?
Cevap: Hayir
Adaylar: 3

Hedef dugum bulundu: 3

[i]-> Bellek temizlendi!
--------------------------------
```

```
Lutfen test etmek istediginiz grafi secin (1, 2, veya 3): 3

Grafin Komsuluk Matrisi:
0 1 0 1 0 0
1 0 0 0 1 0
0 0 0 0 0 1
1 0 0 0 1 0
0 1 0 1 0 1
0 0 1 0 1 0

Hangi dugum hedef olsun: 10

Lutfen 0 - 5 araliginda bir deger giriniz!
Hangi dugum hedef olsun: 5


Soru 1: [4] 5'ye komsu mu?
Cevap: Evet
Adaylar: 1 3 5

Soru 2: [0] 5'ye komsu mu?
Cevap: Hayir
Adaylar: 5

Hedef dugum bulundu: 5

[i]-> Bellek temizlendi!
--------------------------------
```