GENİŞLİK ÖNCELİKLİ ARAMA (BFS)

CEYDA GÜLEN[1], ZEYNEP ERARSLAN[2]

21360859042@ogr.btu.edu.tr/21360859042

22360859019@ogr.btu.edu.tr / 22360859019

BURSA TEKNİK ÜNİVERSİTESİ

İçindekiler

GENİŞLİK ÖNCELİKLİ ARAMA

1.1 Giriş

Genişlik Öncelikli Arama, ağaç veya grafik veri yapılarını dolaşmak veya aramak için kullanılan bir algoritmadır. Bu algoritma, bir sonraki seviyeye geçmeden önce belirli bir seviyedeki tüm köşeleri ziyaret ederek bir grafiği sistematik olarak inceleyen bir grafik geçiş yapısı sunmaktadır.Öncelikle en üst köşesinden başlar, bu köşeyi bir kuyruğa ekler ve işlenmiş(ziyaret edilmiş) olarak hafızada tutar.Ardından kuyruktan bir köşeyi çıkarır, onu ziyaret eder ve ziyaret edilmemiş tüm komşularını kuyruğa ekler. Bu işlem kuyruk boşalana kadar devam eder.Bu algoritma, graf teorisinin temelini oluşturan ilk algoritmalardan biridir.Algoritma,1950'li yıllarda bilgisayar bilimlerinde graf arama problemlerini çözmek için geliştirilimiştir.1945 yılında Konrad Zuse tarafından Plankalkül programlama dili üzerine yazdığı doktora tezinde icat edildi , ancak bu 1972 yılına kadar yayınlanmadı. 1959 yılında Edward F. Moore tarafından labirentten en kısa çıkış yolunu bulmak için yeniden icat edildi..BFS,iki düğüm arasındaki en kısa mesafeyi bulmayı ya da belirli bir düğüme ulaşmak için gereken minimum adım sayısını bulmayı içeren sorunları çözmek için kullanılmaktadır. Uygulama alanları ise; en kısa yol bulma,döngü algılama,bağlantılı bileşenler,topolojik sıralama,ikili ağaçların seviye sırası gezintisi,ağ yönlendirme,veri işleme ve optimizasyon bulunmaktadır.BFS algoritması yerel harita tabanlı keşif için kullanılır. Modifiye edilmiş BFS, robotun konum belirsizliğini azaltarak, büyük ortamlarda doğru haritalar oluşturmak için kök düğümden döngü kapama işlemi başlatır. Yerel haritaları ve bitişik düğüm bilgilerini kullanarak haritalama verimliliğini artırır ve geleneksel yöntemlerin sınırlamalarını aşar (Ryu, H.,& Chung,W.K. 2015). Genişlik önce arama (BFS) algoritması, elektronik tasarım otomasyonu alanından sosyal ağ analizine kadar çok çeşitli uygulamalarla grafik geçişi için temel bir yapı taşı görevi görür. BFS sonucunu güncellemek için gereken toplam iş yükünü en aza indirmek için toplu iş bilgilerinden yararlanırken, aynı zamanda gelecekteki güncellemeler için veri yerelliğini de artırır.(Aigueperse, J., Bensana, E., & Trinquet, Y. 2000)

1.2 Teorik Temel

Genişlik Öncelikli Arama bir grafiği, ağ yapısını keşfetmek ve en kısa yolu bulmak için kullanılan bir algoritmadır.Başlangıç düğümünden başlayarak komşu düğümlere doğru ilerler.BFS, bir grafikteki her düğümü en kısa yol ile keşfetmeyi amaçlar. Matematiksel yapısı şu şekilde açıklanmaktadır:

1. **Başlangıç Düğümü:** Algoritma, başlangıç düğümünden başlar.
2. **Komşu Düğümleri Keşfetme:** İlk olarak başlangıç düğümünün komşuları keşfedilir ardından tüm komşuları sırasıyla kuyruğa eklenir.
3. **Ziyaret Etme ve Kuyruklama:** Her düğüm sırasıyla işlendikten sonra kuyruktan çıkarılır. Kuyruktaki ziyaret edilmeyen komşular kuyruğa eklenir.

1.Algoritmanın Temel Mantığı

BFS, genelde bir **graf G=(V,E)** üzerinde uygulanır.
Burada:

V, grafın düğümler kümesini (veya köşe kümesini) ifade eder.
E, düğümleri birbirine bağlayan kenarların kümesidir.

BFS algoritmasına başlamak için ilk önce bir kaynak düğüm seçilerek başlanır ($s \in V$) .

BFS, düğümleri keşfetmek için ilk giren ilk çıkar (FIFO) kuyruğu kullanır ve her bir düğümden diğerine seviyeyle ilerler.

Ziyaret edilen düğümler bir bir takip edilir, böylece tekrarlardan kaçınılır.

2. Matematiksel İfadelerle Adımlar

2.1 Başlangıç Durumu

Kaynak düğüm s, BFS sırasında keşfedilecek ilk düğümdür.
d[s]=0 (kaynak düğümün uzaklığı sıfır olarak atanır) , d(v), kaynak düğüm s ile herhangi bir v∈V düğümü arasındaki minimum adım sayısıdır. **d(v)=**∞

2.2 Keşif Süreci

Algoritma, s düğümünü kuyruktan çıkarır ve ona komşu olan tüm düğümleri keşfeder.

Eğer bir düğüm (u) , keşfedilmemiş durumdaysa : **d[u]=d[v]+1**, burada v, u düğümüne bağlı olan önceki düğümdür. Bu, u düğümünün katmanını, v'nin katmanına bir ekleyerek bulur.

2.3 Ziyaret Durumu Güncellemesi

Ziyaret edilen düğümler bir "k" kümesinde tutulabilir.

k={v∈V|d(v=!∞**}**

2.4 Kuyruk Boş Olana Kadar Devam Etme

Kuyruk boşalana kadar adımlar tekrar edilir.
Q=!∅

2.5 Son Durum ve Çıkış

Algoritma, grafın tüm düğümleri keşfedildiğinde durur. Bu süreçte her düğüm v, kaynak düğüm s'ye olan en kısa yolu ifade eden d[v] değerine sahiptir.

3. BFS Algoritmasında Kullanılan Veri Yapıları ve Kavramlar

Graf teorisinin temel algoritmalarından biri olan Genişlik Öncelikli Arama (BFS) ,graf teorisindeki önemli problemleri çözmek için geliştirilimiştir ve belirli bir düzene göre graf üzerinde gezinme işlemini

gerçekleştirir. Bu algoritma çeşidinde temel veri yapıları ve kavramları önemli bir rol almaktadır.

3.1 Graf

BFS, graf yapısı üzerinde çalışır. Bir graf, genellikle iki temel bileşenden oluşur. Bunlar **düğüm** ve **kenarlardır**.Ayrıca BFS'nin uygulanabilmesi için grafın komşuluk matrisi ile temsil edilmesi gerekmektedir.

Düğüm: Grafın temel birimidir. Örneğin, bir şehir haritasında her şehir bir düğüm olarak temsil edilebilir.

Kenar: Düğümleri birbirine bağlayan bağlantılardır. Yollar veya bağlantılar bu kenarlarla temsil edilir.

3.2 Kuyruk

Bu algoritmanın temel veri yapısı kuyruklardır. FIFO prensibiyle çalışır; yani ilk giren eleman ilk çıkar. Kuyruk,ziyaret edilecek düğümlerin sırasını belirler ve yönetir. İşleyişinde şu adımlarla kullanılır:

Öncelikle, başlangıç düğümü kuyruğa eklenir.
Sonra, hangi düğümlerin sırayla alınacağını kontrol eder
Son olarak başlangıç düğümünden diğer düğümlere olan mesafeyi belirler.

FIFO mantığıyla çalışan bu algoritma mantığı ve kuyruk sistemi BFS'nin sistematik ve katmanlar halinde çalışmasını sağlar.Her bir düğümün yalnızca bir kez ziyaret edilmesini sağlar ve düzenli kontrol edilmesini sağlar.

4. Genişlik Öncelikli Arama Algoritmasının Avantajları ve Sınırlamaları

Her algoritmada olduğu gibi BFS' nin de güçlü yönleri ve yetersiz kaldığı durumlar bulunmaktadır.

4.1 BFS Algoritmasının Avantajları

1.En Kısa Yol Bulması: En önemli avantajlarından birisi ağırlıksız graf yapılarında bir düğümden diğer düğüme olan en kısa yolu bulabilmesidir. Navigasyon sistemlerinde iki nokta arasında en kısa yolu bulmada oldukça kullanışlıdır.

2. Basit ve Sistematik Yaklaşım: BFS, katmanlar halinde çalıştığı için graf yapılarında düğümleri düzenli ve sistematik şekilde keşfeder.Tüm komşu düğümleri sırasıyla inceler ve rastlantısal hataları önler.

3. Tüm Düğümlerin Keşfedilmesi: Başlangıç düğümünden başlayarak graf yapısının tüm düğümlerini dolaşabilir.Bu durum tüm bağlı grafları ziyaret etmeye olanak sağlar.

4.Döngülerin Kolay Yönetimi: BFS, graf üzerinde döngüler olsa bile düzgün çalışır. Ziyaret edilen düğümleri işaretleme yöntemi sayesinde, algoritma sonsuz döngüye girmez.

4.2 BFS Algoritmasının Sınırlamaları

1. **Aynı Mesafedeki Düğümlerin Keşfi:** BFS, aynı mesafedeki düğümleri keşfetme sırasında kesinlik sağlamaz. Bu, uygulanan probleme bağlı olarak sorun yaratabilir.

2. **Dinamik Grafiklerde Yetersizlik:** BFS algoritması, genellikle statik grafikler için oluşturulmuştur. Eğer graf dinamik olarak değişiyorsa (örneğin, düğüm veya kenar eklenip kaldırılıyorsa), algoritmayı tekrardan çalıştırmak gerekebilir.

3. **Sonsuz Döngü Durumu:** Döngüler içeren grafiklerde, ziyaret edilen düğümlerin doğru bir şekilde işaretlenmemesi durumunda algoritma sonsuz döngüye girebilir ve algoritma etkisiz hale gelir.

4. **Büyük Grafiklerde Bellek Kullanımı**: BFS, büyük grafiklerde çok fazla bellek kullanır. Çünkü kuyruğa eklenen düğümler ve ziyaret edilen düğümleri takip etmek için büyük veri yapıları gereklidir.
5. **Derinlik Bazlı Sorunlarda Yetersizlik:** BFS, genişlik öncelikli bir algoritmadır. Derinlik bazlı keşif gereken durumlarda (örneğin, labirent çıkışı) daha az etkili olabilir.

6. **Ağırlıklı Grafiklerde En Kısa Yol Bulamama:** BFS, kenar ağırlıklarını dikkate almaz. Eğer graf ağırlıklıysa, BFS doğru bir en kısa yol çözümü sunamaz. Bu durumda Dijkstra veya Bellman-Ford gibi algoritmalar tercih edilmelidir.

7. **Çoklu Bağlı Bileşenler:** Graf birden fazla bağlı bileşene sahipse, BFS sadece başladığı bileşeni keşfeder. Tüm grafı keşfetmek için algoritmanın her bağlı bileşen üzerinde ayrı ayrı çalıştırılması gerekir.

8. **Paralel İşlemeye Uygun Olmama:** BFS, ardışık bir süreç üzerinden ilerler böylelikle büyük grafiklerde zaman verimliliği sınırlı olabilir.

1.3 Algoritmanın Uygulama Alanları

BFS Algoritması

1. **Girdi:**

   1.1 Bir grafik **G=(V,E)** , burada **V** düğümler kümesi, **E** kenarlar kümesi.
   1.2 Bir kaynak düğüm. (örneğin, start=′A′)

2. **İşleme Süreci:**

   2.1 Başlangıç:
   2.1.1 Bir FIFO kuyruğu ve bir ziyaret edilen düğümler listesi oluşturulur.
   2.1.2 Kaynak düğüm kuyruğa eklenir.

   2.2 Keşif Döngüsü:
   2.2.1 Kuyruğun önündeki düğüm çıkarılır ve işlenir.
   2.2.2 Düğüm, ziyaret edilenler listesine eklenir.
   2.2.3 Düğümün komşuları kontrol edilir. Ziyaret edilmemiş komşular kuyruğa eklenir. Kuyruk boş olana kadar işlem devam eder.

2.3 Ziyaret Edilen Düğümler: Döngü tamamlandığında, tüm ziyaret edilen düğümler listesi döndürülür.

3. Çıktı:

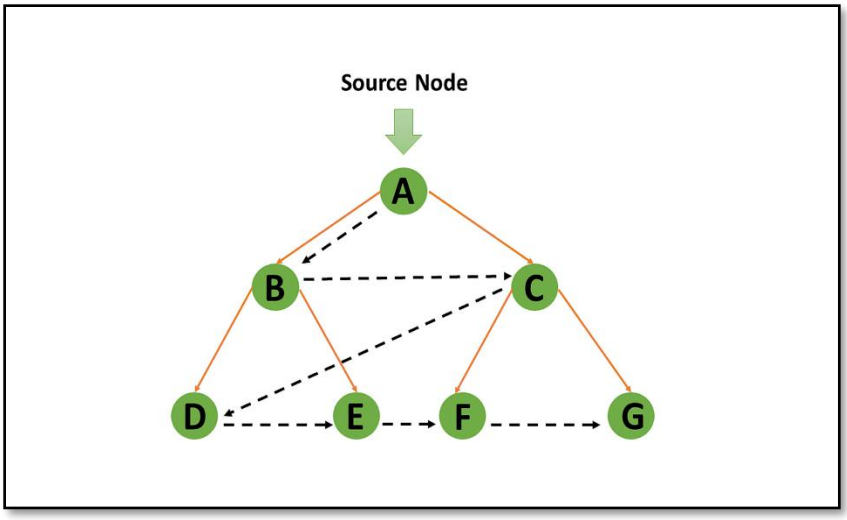
3.1 Kaynak düğümden başlayarak tüm düğümleri sırasıyla keşfetme düzeni.

```
Ziyaret Edilen Düğümler Sırası:  ['A', 'B', 'C', 'D', 'E', 'F']
```

Şekil 1. Algoritma Çıktısı

```
def bfs(graph, start):
    visited = []
    queue = []
    queue.append(start)

    while queue:
        node = queue.pop(0)
        if node not in visited:
            visited.append(node)
            for neighbor in graph[node]:
                if neighbor not in visited:
                    queue.append(neighbor)
    return visited

graph = {
    'A': ['B', 'C'],
    'B': ['D', 'E'],
    'C': ['F'],
    'D': [],
    'E': ['F'],
    'F': []
}

visited_nodes = bfs(graph, 'A')

print("Ziyaret Edilen Düğümler Sırası: ", visited_nodes)
```

Python Kodu



Şekil 2. Algoritma İşleyişi

BFS Algoritmasının Kullanım Alanları

1. **Ağ Analizi**

   1.1 Sosyal Ağlar: BFS algoritması, sosyal ağlardaki kullanıcılar arasındaki bağlantı seviyelerini bulmak için kullanılır. Örneğin, bir sosyal medya platformunda iki kullanıcı arasındaki "arkadaşlık derecesini" (örneğin, kaç bağlantı uzağında olduklarını) hesaplamak için kullanılabilir.

   1.2 Bilgisayar Ağları: bir ağdaki cihazlar arasında en kısa yol veya en kısa gecikme analizlerini gerçekleştirmek için kullanılabilir. Örneğin, bir ağ yönlendiricisinin, bir paketin kaynaktan hedefe en az adımda nasıl iletileceğini belirlemesi.

2. **Yapay Zeka :** BFS, yapay zeka problemlerinde durum uzayında çözüme ulaşmak için kullanılır. Örneğin, bir yapboz çözümünde, başlangıç durumundan hedef duruma ulaşmak

için BFS algoritması kullanılabilir. Bu, en kısa çözüm yolunu bulmak için etkili bir yöntemdir.

3. **Web :**

3.1 Web sayfaları arasındaki bağlantıların keşfi ve analizi için BFS algoritması kullanılabilir. Örneğin, bir web tarayıcısı, bir web sitesindeki tüm sayfalar keşfedilmesi için algoritma kullanılarak bağlantılar seviyeler halinde taranır.
3.2 BFS, belirli bir seviyeye kadar bağlantıları analiz etmek için kullanılır. Örneğin, bir e-ticaret sitesinde ürün sayfaları arasındaki bağlantıların analizinde BFS uygulanabilir.

4. **Robotik ve Oyun Geliştirme:**

4.1 Bir robotun bir ortamda belirli bir hedefe ulaşması için en kısa yolu bulması gerekebilir. Örneğin, bir temizlik robotu, odaları en verimli şekilde temizlemek için BFS kullanarak bir yol haritası çıkarabilir.
4.2 Algoritma , oyun dünyalarında karakterlerin labirent, zindan veya harita gibi yapılarda çıkış yollarını bulması için kullanılır. Örneğin, bir karakterin en hızlı şekilde bir hedefe ulaşması gerektiğinde, BFS en kısa yolu hesaplamak için etkili bir yöntemdir.

5. **Coğrafi Bilgi Sistemleri:**

5.1 GPS navigasyon sistemlerinde, ağırlıksız grafiklerde hızlı ve etkili bir yol bulma yöntemi olarak BFS algoritması kullanılır.
5.2 BFS, haritalar üzerinde iki nokta arasındaki en kısa mesafeyi bulmak için kullanılabilir. Örneğin, bir şehirdeki yolların grafik olarak temsil edildiği bir haritada, BFS kullanılarak bir yerden başka bir yere en kısa mesafede ulaşılabilir.

1.4 Performans Analizi

1. Zaman ve Uzay Karmaşıklığı

BFS algoritmasının zaman ve uzay karmaşıklığı, grafın yapısına ve nasıl oluşturulduğuna bağlıdır.

1.1 Zaman Karmaşıklığı:

1.1.1 Genel Durum: BFS, her kenarı ve düğümü bir kez ziyaret eder.

Formül = **O(V+E)**

V : Düğüm Sayısı

E : Kenar Sayısı

1.1.2 Ortalama Durum: Eğer graf karmaşık değilse (kenar sayısı E<V^2) , BFS daha hızlı çalışır.

Formül = **O(VlogV)**

1.1.3 En Kötü Durum: Eğer graf yoğun bir graf ise (tam bir graf, E=V^2)

Formül = **O(V2)**

1.2 Uzay Karmaşıklığı: BFS, bir kuyruğa ve ziyaret edilen düğümleri izlemek için bir listeye ihtiyaç duyar. Ve her düğüm en fazla bir kez kuyruğa eklenir ve bir kez işlenir.

Formül = **O(V)**

2. BFS Algoritmasını Geliştirme veya Optimize Etme Yöntemleri

Analiz ve geliştirme yöntemleri, algoritmanın performansını hem zaman hem de bellek açısından optimize etmeye yardımcı olur.

2.1 Ziyaret Edilen Düğümleri İzleme: Ziyaret edilen düğümleri bir "set" veri yapısıyla takip etmek, arama işlemlerini daha kolay ve etkili bir hale getirir.

2.2 Öncelikli Keşif: Eğer belirli düğümler daha önce keşfedilmek isteniyorsa, BFS'yi bir öncelik kuyruğuyla birleştirerek özelleştirilebilir (örneğin, Weighted BFS).

2.3 Grafın Temsil Biçimi: Seyrek Grafiklerde, Adjacency List (Komşuluk Listesi) kullanmak, bellek kullanımını ve işlem süresini azaltabilir. Yoğun grafiklerde ise Adjacency Matrix (Komşuluk Matrisi) kullanmak daha etkili olabilir.

2.4 Paralel İşleme: BFS'nin komşu düğümleri aynı anda keşfetme özelliği, paralel işlemeye uygundur. Büyük grafiklerde BFS'yi hızlandırmak için paralel BFS algoritmaları kullanılabilir.

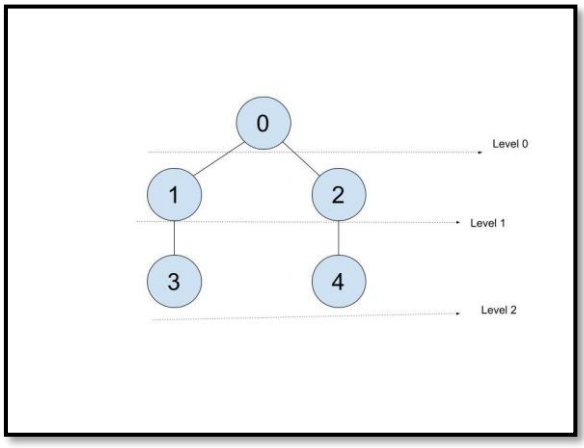
2.5 Gereksiz Hesaplamaları Önleme: Amacımız Kaynak düğümden hedef düğüme ulaşmak ise , tüm düğümleri ziyaret etmek yerine hedef düğüm bulunduğunda algoritmayı sonlandırmak performansı artırabilir.

2.6 Dinamik Grafikleri Ele Alma: Eğer graf sürekli değişim halindeyse (örneğin, düğüm veya kenar eklenip kaldırılıyorsa), BFS'nin yalnızca ilgili kısmını yeniden çalıştırmak için bir **"incremental BFS"** yöntemi uygulanabilir.

1.5 Çalışma Soruları ve Egzersizler

PROBLEMLER

1.SORU: V köşeleri ve E kenarları ve bir X düğümü olan yönsüz bir grafik verilsin,yönsüz grafikte X düğümünün seviyesini bulunuz.
V = 5
Kenarlar = {{0, 1}, {0, 2}, {1, 3}, {2, 4}}
X = 3

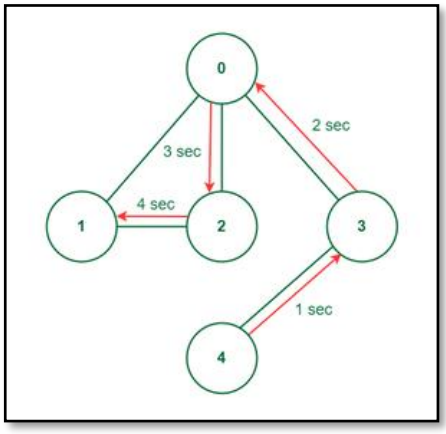


Şekil 3. Yönsüz Graf

ÇÖZÜM: 0. seviyede 0 düğüm, 1. seviyede 1. ve 2. Düğüm, 2. Seviyede 3. ve 4. Düğüm bulunur.

CEVAP: 2

2.SORU: graph[] = [[1, 2, 3], [2, 0], [0, 1], [0, 4], [3]] için , her düğüm için bir grafik ve komşular listesi verildiğinde , graph[i], i . düğüme bağlı komşu düğümlerin listesini içerir buna göre verilen grafiğin tüm düğümlerini ziyaret etmek için en kısa süreyi bulunuz.

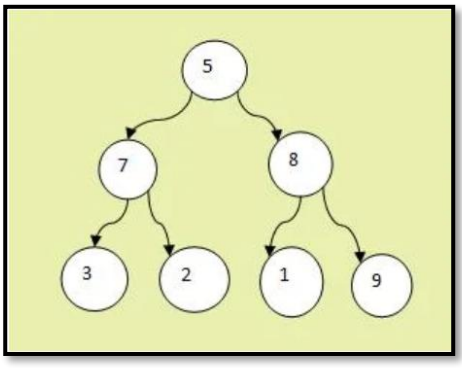


Şekil 4. Soru Çözümü

ÇÖZÜM : Tüm düğümleri en az sürede ziyaret etmenin olası bir yolu yukarıdaki grafikte gösterilmiştir.Görüldüğü gibi gidilebilecek en kısa yollar tercih edilmiştir.

CEVAP: 4

3.SORU: Verilen bir ağaçta genişlik öncelikli arama (BFS) algoritmasını kullanarak, kök düğümden başlayarak düğümleri sırayla ziyaret edin. Ziyaret edilen düğümlerin sırasını yazınız.



Şekil 5. Yönlü Graf

ÇÖZÜM: BFS Ziyaret Sırası: 5 → 7 → 8 → 3 → 2 → 1 → 9

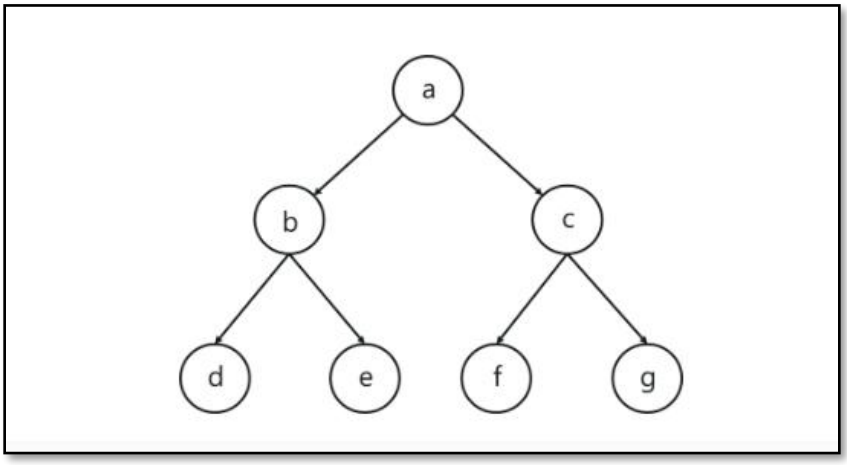
Bu sırada, önce kök düğüm (5) ziyaret edilir. Ardından birinci seviyedeki çocuklar (7 ve 8) soldan sağa doğru sırayla ziyaret edilir. Daha sonra ikinci seviyedeki çocuklar (3, 2, 1, 9) soldan sağa doğru ziyaret edilir.

4.SORU: V düğüm ve E kenarı olan bir çizgenin komşuluk listesi temsili üzerinde BFS (Breadth-First Search) algoritmasının bellek karmaşıklığı nedir?

ÇÖZÜM: BFS algoritması, bir çizgenin komşuluk listesi temsilinde V düğüm olduğunda, bir düğüm ziyaret edildiği zaman bellekte sabit bir miktarda yer işgal eder. Bu nedenle, BFS algoritmasının bellek karmaşıklığı O(V) olur. Çünkü BFS, ziyaret edilen düğümleri bir kuyrukta saklar ve her düğüm yalnızca bir kez kuyruğa eklenir ve kuyruktan çıkarılır.

CEVAP: O(V)

5.SORU:



Şekil 6. Graf Örneği

Verilen ikili ağaç üzerinde Breadth-First Search (Genişlik Öncelikli Arama, BFS) algoritmasını FIFO (First-In-First-Out) mantığıyla uygulayınız. BFS algoritmasının her adımında kuyruk (queue) yapısını güncelleyerek ziyaret edilen düğümleri ve kuyruktaki düğüm durumlarını adım adım açıklayınız. Ayrıca ziyaret sırasını belirtiniz.

ÇÖZÜM :

Şekil 6.1

Kuyruk: a
Ziyaret edilen: (henüz yok)
İşlem: a düğümü ziyaret edilir ve çocukları b, c sıraya eklenir.

Şekil 6.2

Kuyruk: b, c
Ziyaret edilen: a
İşlem: b düğümü ziyaret edilir ve çocukları d, e sıraya eklenir.

Şekil 6.3

Kuyruk: c, d, e
Ziyaret edilen: a, b
İşlem: c düğümü ziyaret edilir ve çocukları f, g sıraya eklenir.

Şekil 6.4

Kuyruk: d, e, f, g
Ziyaret edilen: a, b, c
İşlem: d düğümü ziyaret edilir. Çocuğu olmadığından sıraya yeni bir eleman eklenmez.

Şekil 6.5

Kuyruk: e, f, g
Ziyaret edilen: a, b, c, d
İşlem: e düğümü ziyaret edilir. Çocuğu olmadığından sıraya yeni bir eleman eklenmez.

Şekil 6.6

Kuyruk: f, g
Ziyaret edilen: a, b, c, d, e
İşlem: f düğümü ziyaret edilir. Çocuğu olmadığından sıraya yeni bir eleman eklenmez.

Şekil 6.7

Kuyruk: g
Ziyaret edilen: a, b, c, d, e, f

İşlem: g düğümü ziyaret edilir. Çocuğu olmadığından sıraya yeni bir eleman eklenmez.

BFS Ziyaret Sırası:
a → b → c → d → e → f → g

1.6 Şekil, Tablo ve Kaynakça

- **Şekil 1. Algoritma Çıktısı:** Burda BFS algoritmasının Python kodundaki çıktısı verilmiştir.

- **Şekil 2. Algoritma İşleyişi :** Bu fotoğrafta algoritmanın işleyiş yönü oklarla açık şekilde anlatılmaktadır. Simplilearn.*BFS Algorithm*. Simplilearn.

- **Şekil 3. Yönsüz Graf :** Yönsüz graf olarak tanımlanmış ve BFS algoritması ile çözümü istenmiş bir soru resmidir. GeeksforGeeks, "Find the level of given node in an undirected graph,

- **Şekil 4. Soru Çözümü :** Sitede verilen örnek BFS sorusunun cevabıdır. GeeksforGeeks. (n.d.). *Minimum time to visit all nodes of a given graph at least once*.

- **Şekil 5.Yönlü Graf :** Kaynakta verilen bir yönlü BFS soru resmidir. Bilgisayar Kavramları. (2008, November 13). *Sığ Öncelikli Arama (Breadth First Search)*.

- **Şekil 6.Graf Örneği :** Verilen sitedeki harflerle verilmiş bir graf örneğidir. Edureka. (n.d.). *Breadth First Search Algorithm*.

- **Şekil 6.1-6.2-6.3-6.4-6.5-6.6-6.7 :** Bir önceki örnekte verilen (Şekil 6.) sorunun cevap adımları numaralandırılıp tek tek açıklandırılmaktadır

1.7 Algoritma Özeti

Genişlik Öncelikli Arama algoritması ,veri yapıları konusu kullanarak geliştirilen ve graf arama algoritmaları arasında önemli bir yer kaplamaktadır. Bu algoritma,bir kök düğüm üzerinden başlayarak diğer düğümlere ulaşmak için en kısa yolu bulmayı amaçlamaktadır.Sıralı şekilde düğümler ziyaret edilerek genişlik üzerinden ilerlemektedir ve en kısa yolun bulunması gerektiğinde önemlidir.Düğümlerin ziyareti sırasında kuyruk veri yapısını kullanan algoritma ilk giren ilk çıkar prensibine dayanarak işlemektedir.Ağ analizi,en kısa yol bulunması,labirent çözümü gibi çeşitli uygulamalarda BFS'nin tercih edilmesine sebep olmaktadır.

Kaynakça:

Guru99. (n.d.). *Breadth First Search (BFS) Graph Example*. Retrieved December 30, 2024, from **https://www.guru99.com/tr/breadth-first-search-bfs-graph-example.html**

Edureka. (n.d.). *Breadth First Search Algorithm*. Retrieved December 30, 2024, from **https://www.edureka.co/blog/breadth-first-search-algorithm/**

Kulcu, S. (n.d.). *Bolum 16: Soru Cevap 2*. Retrieved December 30, 2024, from **https://sercankulcu.github.io/files/algorithms/slides/Bolum_16_Soru_Cevap_2.pdf**

Simplilearn. (n.d.). *BFS Algorithm*. Retrieved December 30, 2024, from **https://www-simplilearn-com.translate.goog/tutorials/data-structure-tutorial/bfs-algorithm?_x_tr_sl=en&_x_tr_tl=tr&_x_tr_hl=tr&_x_tr_pto=tc**

GeeksforGeeks. (n.d.). *Find the level of given node in an undirected graph*. Retrieved December 30, 2024, from

https://www.geeksforgeeks.org/find-the-level-of-given-node-in-an-undirected-graph/

GeeksforGeeks. (n.d.). *Minimum time to visit all nodes of a given graph at least once*. Retrieved December 30, 2024, from **https://www.geeksforgeeks.org/minimum-time-to-visit-all-nodes-of-given-graph-at-least-once/**

Bilgisayar Kavramları. (2008, November 13). *Sığ Öncelikli Arama (Breadth First Search)*. Retrieved December 30, 2024, **from** **https://bilgisayarkavramlari.com/2008/11/13/sig-oncelikli-arama-breadth-first-search/**

Wikipedia. (n.d.). *Breadth-first search*. Retrieved December 30, 2024, from **https://en-m-wikipedia-org.translate.goog/wiki/Breadth-first_search?_x_tr_sl=en&_x_tr_tl=tr&_x_tr_hl=tr&_x_tr_pto=tc**

Springer. (2015). *Improved Breadth-First Search Methods for Robotic Applications*. Retrieved December 30, 2024, from **https://link.springer.com/article/10.1007/s12541-015-0269-9#citeas**