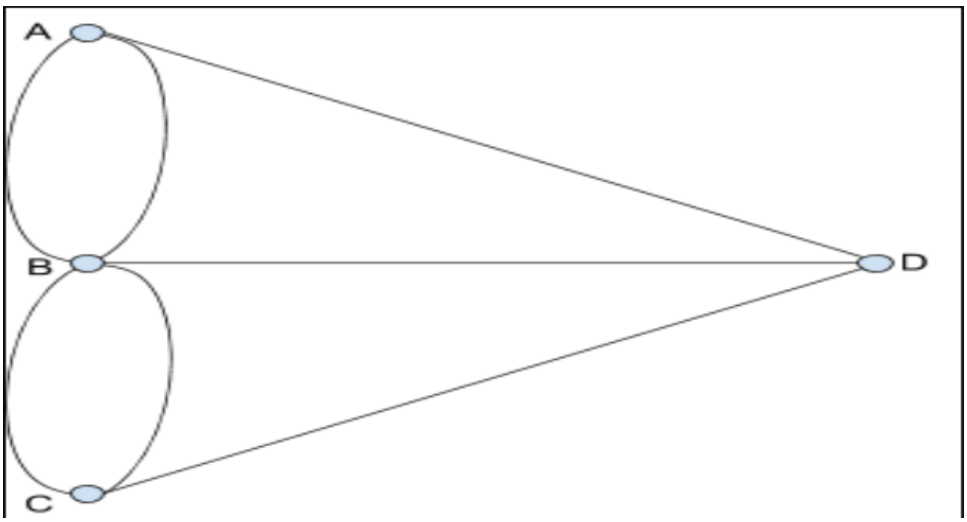
Kıvanç Adıgüzel
191180003

Königsberg'in Yedi Köprü probleminin bir çözümü var mı? Varsa çözümü/çözümleri genel hatları ile anlatınız ve bunu grafiklerle gösteriniz

Euler'in çözümünü ele alacak olursa graf teorisi ile bu problemi inceleyebiliriz. Problemin temeli, gezmeye başlayacak birinin hangi noktada bulunduğu ile alakalı değil, ırmağın hangi kıyısında olduğu veya hangi adada olduğu ile ilgilidir. Bu durumda, ırmağın iki yakasını ve adaları birer nokta ile köprüleri ise birer çizgi ile göstermek mümkündür. Kuzey yakasını A, Güney yakayı B, adayı C ve yarımadayı da D noktası ile gösterilmiş varsayalım ve Königsberg'in köprülerini basit graf'a çevirelim.



A, B, C, D köşe noktaları ırmağın iki yakasını ve adaları gösterir. Yollar ise köprüleri göstermektedir. Bu durumda, A, B ve D köşelerinin her birisinin derecesi(o köşeden çıkan farklı yol sayısı) 3, C köşesininki ise 5'tir.

Bir köşeden geziye başlayan birisinin aynı köşeye farklı bir yoldan dönebilmesi için bu köşeye dönen farklı bir yolun olması gerekir. Yani bir köşeden her çıkış dönüş yapabilmesi için iki farklı yol gerekir. Bu durumda bir köşeden iki kez farklı yollardan çıkıp geri dönmek için o köşenin derecesi 4 olmalıdır. Bu işi üç kez yapabilmek için, o köşenin derecesi 6 olmalıdır. Öyleyse, A, B, D köşelerinin herhangi birisinden geziye başlayan kişi sırasıyla *çıkış-dönüş-çıkış* yapabilir. Ama *son çıkışın* dönüşü yapılamaz. C noktasından geziye çıkan kişi, sırasıyla *çıkış-dönüş-çıkış-dönüş-çıkış* yapabilir ama yine aynı şekilde *son çıkışın* dönüşünü yapamaz. Bu durum geziye nereden başlanırsa başlansın, yedi köprüyü birer kez geçerek başladığı noktaya geri dönemeyeceğinin kanıtıdır.

Başlangıç noktasına dönme şartı olmadan her köprüyü yalnız bir kez geçerek kenti dolaşmak mümkün müdür? (*Graftaki her yolu yalnız bir kez yürümek mümkün müdür?)*

Başlangıç noktasına geri dönülmeyeceği varsayıldığına göre, başlangıç ve bitiş köşeleri ile aradaki köşelerin dereceleri arasında bir fark olmalıdır. Başlangıç köşesi için

a)çıkış b)çıkış-dönüş-çıkış c)çıkış-dönüş-çıkış-dönüş-çıkış

yollarından birisi tercih edilebilir. Bu durumda başlangıç köşesinin derecesi tek olmalıdır.

Bitiş köşesi için

a)dönüş b)dönüş-çıkış-dönüş c)dönüş-çıkış-dönüş-çıkış-dönüş

yollarından birisi olabilir yani bitiş köşesinin derecesi de tek olmalıdır. Ara köşeler içinse uğranan her köşe için, her girişin bir de çıkışı olmalıdır dolayısıyla aradaki köşelerin dereceleri çift olmalıdır. Her yolundan yalnız bir kez geçilerek bir grafın bütün yollarının yürünebilmesi için, iki ve yalnızca iki köşesinin tek dereceli, öteki köşelerin çift dereceli olması gerektiği sonucuna ulaşabiliriz.(path traversing)

Königsberg köprülerinin grafında bu koşulun sağlanmadığını açıkça görülmektedir. Bütün köşeler tek derecelidir. Hangi köşeleri ortalarsak ortalayalım, her girişe karşılık bir çıkış yolu olamaz. Dolayısıyla, ele aldığımız Euler'in çözümü, probleme çözüm olarak aranan seyahatin olanaksızlığını ispatladı ve *bir grafta her yoldan bir kez geçerek bütün yolları yürümenin (path traversing) mümkün olabilmesi için iki ve yalnızca iki köşenin derecelerinin tek, ötekilerin derecelerininse çift olması gerektiğini* kanıtladı. (Gezinin başlangıç ve bitim noktaları aynı ise, bütün köşelerin derecelerinin çift olması gerekir.)

The Handshaking teorimi nedir? Kısaca açıklayınız

Herhangi bir grafta düğümlerin dereceleri toplamının yolların sayısının iki katı olmasıdır.

Bir tokalaşma için 2 kişi gerekir. Aynı ilişki graflardaki yollar için de geçerlidir(yol-düğüm ilişkisi)

1 tokalaşma->2 kişi gerekli

1 yol->2 düğüm gerekli

$$2m = \sum_{v \in V} \deg(v)$$

Sparse matrix nedir ve nerelerde kullanılır? Örneklerle açıklayınız.

Sparse matrix, bir dizi içerisinde dizi eleman sayısının %75'i sıfır (veya önemsiz başka bir veri) içeriyor ise, önemli olan verileri başka bir matrix'te toplayarak arama ve listeleme işlemlerinde yüksek hız kazanma amacıyla geliştirilmiş bir veri yapı algoritmasıdır.

Sıkça kullanıldığı alanlar:

a)Numerik Analiz

b)Bilimsel Hesaplamalar

c)Network Theory

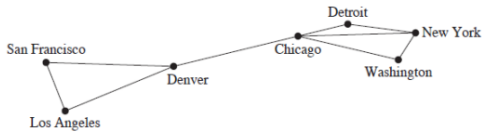
d)Machine Learning

Kıvanç Adıgüzel
191180003

Graph representation (çizge temsili ve gösterimi) yaklaşımları nelerdir? Bunları tablo üzerinde karşılaştırınız.

BASİT GRAF (SIMPLE GRAF):

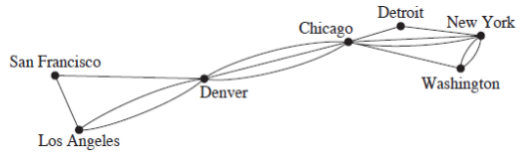
Bir grafta her kenar farklı 2 köşeyi birleştirmişse ve 2 köşe arasında birden fazla kenar yoksa bu grafa 'Basit Graf' denir.Basit grafta her kenar 2 köşenin sırasız ikilisine karşılık gelmektedir ve bu ikiliye karşılık gelen başka bir kenar yoktur.



Şekil 1. Bir Bilgisayar Ağı

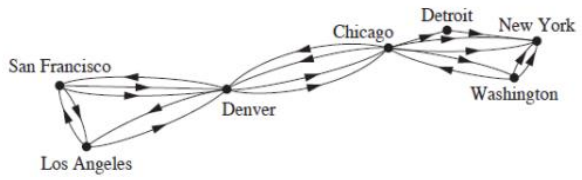
ÇOKLU GRAF (MULTIGRAPH)

Aynı köşe ikilisi arasında çok katlı kenar içeren graflara 'Çoklu graf' denir. Aynı sıralı olmayan $\{u ,v\}$ köşe ikilisine m tane farklı kenar karşılık geliyorsa $\{u,v\}$'nin m katlı bir kenar olduğunu söyleyebiliriz.



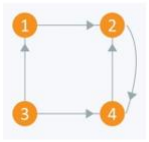
YÖNLÜ GRAF

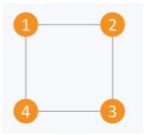
Bir yönlü (V, E) grafı, köşelerin boş olmayan V kümesini ve yönlü kenarların (veya yayların) E kümesini içermektedir. Herbir yönlenmiş kenar, köşelerin sıralı çifti ile ilgilidir. (u,v) sıralı ikilisi ile ilgili yönlü kenar için, bu kenar u köşesinde başlar v köşesinde sona erer. Yönlü graf döngü ve katlı kenar içermiyorsa bu grafa 'Yönlü Basit Graf' denir. Bir köşeden 2. bir köşeye (bu köşe 1. Köşeyle aynı da olabilir) katlı yönlü kenarlar içeren yönlü graflar bu tür ağları modellemek için kullanılabilirler. Bu tür graflara 'Yönlü Çoklu Graf' denir. Her biri (u, v) köşe ikilisine 4 karşılık gelen m adet yönlü kenar varsa, (u,v) kenarı m katlı kenar olarak adlandırılır.



Kıvanç Adıgüzel
191180003

Bitişiklik Matrisi(Adjacency Matrix);





Ax,y=

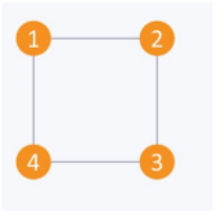
x/y	1	2	3	4
1	0	1	0	0
2	0	0	0	1
3	1	0	0	1
4	0	1	0	0

Ax,y=

x/y	1	2	3	4
1	0	1	0	1
2	1	0	1	0
3	0	1	0	1
4	1	0	1	0

Düğümler ile kenarlar arasındaki bağlantı ilişkisini gösteren matristir.Matrisin satır sayısı düğüm sütün sayısı kenar sayısı kadar olur.

Komşuluk Listesi



A1 → 2 → 4
A2 → 1 → 3
A3 → 2 → 4
A4 → 1 → 3

Örnek Gösterimi

Kenar Listesi

Bir kenarı temsil etmek için, elimizde sadece iki köşe numarasından oluşan bir dizimiz veya kenarların meydana geldiği köşelerin köşe numaralarını içeren bir liste tutulur

```
[ [0,1], [0,6], [0,8], [1,4], [1,6], [1,9], [2,4], [2,6], [3,4], [3,5],
[3,8], [4,5], [4,9], [7,8], [7,9] ]
```

Tip	Kenarlar	Çoklu kenarlara İzin Var mı?	Döngüye İzin Var mı?
Basit Graf	Yönsüz	Hayır	Hayır
Çoklu Graf	Yönsüz	Evet	Hayır
Sözde Graf	Yönsüz	Evet	Evet
Yönlü Basit Graf	Yönlü	Hayır	Hayır
Yönlü Çoklu Graf	Yönlü	Evet	Evet
Karışık Graf	Yönlü ve Yönsüz	Evet	Evet

AVL ağaçlarının kullanıldığı gerçek dünya örneklerini açıklayınız

Bir AVL ağacı, ikili arama ağacının bir alt tipidir. Mucitleri Adelson, Velskii ve Landis'in adını taşıyan AVL ağaçları, ikili arama ağaçlarının sergilediği tüm özelliklere ek olarak dinamik kendi kendini dengeleme özelliğine sahiptir.

BST, düğümlerden oluşan bir veri yapısıdır. Aşağıdaki garantilere sahiptir:

1. Her ağacın bir kök düğümü vardır (en üstte).
2. Kök düğümün sıfır, bir veya iki alt düğümü vardır.
3. Her çocuk düğümün sıfır, bir veya iki çocuk düğümü vardır, vb.
4. Her düğümün en fazla iki çocuğu vardır.
5. Her bir düğüm için, sol torunları mevcut düğümden daha küçüktür ve bu sağ torunlardan daha küçüktür.

AVL ağaçlarının ek bir garantisi vardır:

1. Sağ ve sol alt ağaçların derinliği arasındaki fark birden fazla olamaz. Bu garantiyi sürdürmek için, bir AVL'nin uygulanması, ek bir eleman eklenmesi bu garantiyi bozduğunda ağacı yeniden dengelemek için bir algoritma içerecektir.

AVL ağaçları en kötü durum aramasına sahiptir, O zamanını ekleme ve silme (log n).

AVL Ağaçlarının Uygulamaları :

AVL ağaçları bellek içi diziler ve sözlükler için kullanılır.

AVL ağaçları, ekleme ve silme işlemlerinin daha az olduğu ancak gerekli veri için sık sık aramanın yapıldığı veritabanı uygulamalarında kullanılmaktadır.

Veritabanı uygulamaları dışında gelişmiş arama gerektiren uygulamalarda kullanılır.

Dinamik programlama nedir? Hangi amaçlarla kullanılır? Örneklerle açıklayınız

Dinamik programlama, karmaşık problemlerde, o problemi kendi içerisinde tekrarlayan alt problemlere bölerek elde edilen sonuçları kaydeden, bu sonuçlarla tümdengelim-tümevarım yaklaşımlarıyla asıl problemi çözmeye yarayan bir yöntemdir.

Alt problemlerin çözümünü kaydettiği için, aynı işlemlerin tekrar hesaplanması ihtiyacını ortadan kaldırarak kod maliyetini düşürür. Örneğin, Faktöriyel hesaplama işlemini düşünün. Her sayının faktöriyeli, kendisi ile birlikte 1'e kadar bütün sayıların çarpımına eşittir. Bu, bir sayının kendisi ile kendinden önceki sayının faktöriyeli çarpımına denk gelir. Dinamik programlama ile çözecek olursak, 1'den başlayarak her sayının faktöriyelini tutmak, bir sonraki sayı için sadece 2 değerin çarpımıyla sonucu elde etmemizi sağlayacaktır.

Dinamik programlamanın, aşağıdan yukarıya ve yukarıdan aşağıya olarak yorumlanabilen iki metodu vardır. Brute force yinelemeli çözüm bularak başladığımız bu çözümde memoization veya tablo ile yorumlayarak Dinamik Programlama ile en verimli çözümü elde edilmeye çalışılır.

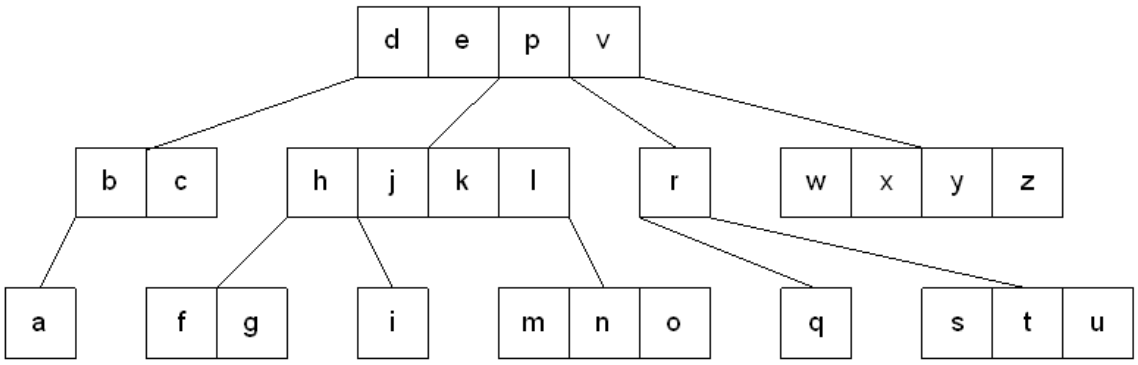
1. Knapsack problemi
2. Gezgin Satıcı Problemi(Traveling Salesman)
3. Matematiksel Optimizasyon problemi
4. En kısa yol problemi
5. Reliability design problemi
6. LCS Problemi
7. Time Sharing problemi

Çok Yollu Ağaçlar (Multi-Way Trees) hangi amaçlar için kullanılır? Çeşitleri nelerdir? Karşılaştırmalı olarak her birini açıklayınız.

B -Trees
B* -Trees
B+ -Trees
B# -Trees

Bir multi-way ağaç sıralı bir ağaçtır ve aşağıdaki özelliklere sahiptir:
a) Bir multiway arama ağacındaki her node, m-1 tane anahtar ve m tane çocuğa sahiptir.

b)Bir node'taki anahtar, sol alt ağaçtaki tüm anahtarlardan büyüktür ve sağ alt ağaçtaki tüm anahtarlardan küçüktür.



Root node en az iki tane yaprak olmayan node'a sahiptir.

Yaprak ve kök olmayan her node k-1t ne anahtara ve k adet alt ağaç referansına sahiptir.

Her yaprak node'u k-1 anahtara sahiptir.

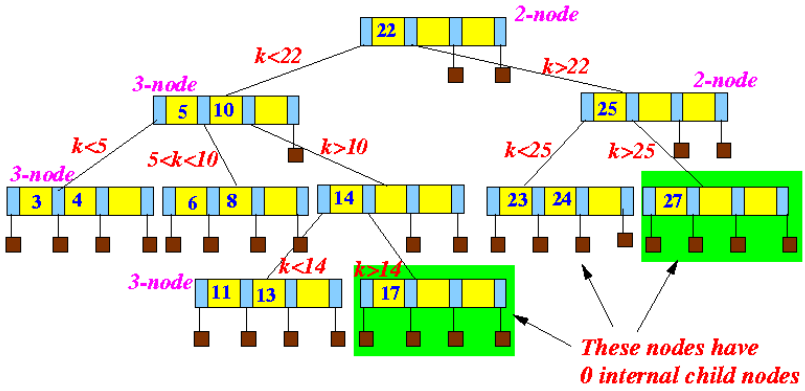
Bütün yapraklar aynı seviyededir.

Herhangi bir döndürmeye gerek kalmadan otomatik olarak balance edilirler.

Nerede ve niçin kullanılırlar?

Multiway ağaçlar disk erişim sayısını azaltmayı amaçlamaktadır.

Kıvanç Adıgüzel
191180003



Multiway Tree Arama örneği

Kaynakça:

Ders Sunumları

https://www.maa.org/press/periodicals/convergence/leonard-eulers-solution-to-the-konigsberg-bridge-problem

https://www.gatevidyalay.com/konigsberg-bridge-problem/

https://www.gatevidyalay.com/handshaking-theorem-graph-theory-theorems/

https://www.geeksforgeeks.org/handshaking-lemma-and-interesting-tree-properties/

https://www.geeksforgeeks.org/sparse-matrix-representation/

https://www.geeksforgeeks.org/graph-and-its-representations/

https://www.javatpoint.com/graph-representation

https://www.khanacademy.org/computing/computer-science/algorithms/graph-representation/a/representing-graphs

https://softwareengineering.stackexchange.com/questions/89117/avl-trees-and-the-real-world

https://www.softwaretestinghelp.com/avl-trees-and-heap-data-structure-in-cpp/

https://www.khanacademy.org/computing/computer-science/algorithms/graph-representation/a/representing-graphs

https://www.khanacademy.org/computing/computer-science/algorithms/graph-representation/a/describing-graphs