İçindekiler

1) UM AĞACI 3
1.a UM_Ağacı Ağacının Oluşturulması 3
1.a.1 Kodlar 3
1.a.2 Açıklama 5
1.b Ağacın Derinliğini, Düğüm Sayısını ve Ağaçtaki Bilgileri Ekrana Listeleme 5
1.b.1 Kodlar 5
1.b.2 Ekran görüntüleri....6
1.b.3 Açıklama 6
1.c UM_Alanlarının İsimlerinin Listelenmesi 7
1.c.1 Kodlar....7
1.c.2 Ekran görüntüleri 7
1.d Özyineli olarak Dengeli Bir Ağaç Oluşturulması....8
1.d.1 Kodlar 8
1.d.2 Ekran görüntüleri....9
1.d.3 Açıklama 9
2) HASH TABLE....10
2.a UM_Alanı Nesnelerini Hash Table'a Yerleştirme....10
2.a.1 Kaynak Kod 10
}2.a.2 Ekran Görüntüleri....11
2.b Hash Table Güncelleme 12
2.b.1 Kaynak Kod 12
2.b.2 Ekran Görüntüsü 14
2.b.3 Açıklama 14
3) HEAP VERİ YAPISI (SINIFI) 14
3.a Heap 14
3.a.1 Kaynak Kod 14
} 16
3.b Min Heap'e Yerleştirme 16
3.b.1 Kaynak Kod 16
3.c Min Heap Listeleme 18
3.c.1 Kaynak Kod 18
3.c.2 Ekran Görüntüsü 19
3.c.3 Açıklama 19

4.a Sıralama Algoritmaları 20
4.a.1 Kaynak Kod 20
4.a.2 Açıklama 22
// Bubble Sort: 22
Quick Sort: 22
4.b Zaman Karmaşıklığının Hesaplanması 22
4.b.1 Açıklama 22
Bubble Sort: 22
Quick Sort: 22
Sonuç: 22
5) Görselleştirme 23
5.a Görselleştirmenin Etkisi 23
5.a.1 Açıklama 23
5.b Karşılaştırma Yapılması 23
5.b.1 Açıklama 23
6) Özdeğerlendirme Tablosu 23

1) UM AĞACI

Visual Studio C#

1.a UM_Ağacı Ağacının Oluşturulması

1.a.1 Kodlar

```
using System;
using System.Collections.Generic;

class UM_Alanı
{
    public string AlanAdi { get; set; }
    public List<string> Words { get; set; }

    public UM_Alanı(string alanAdi)
    {
        AlanAdi = alanAdi;
        Words = new List<string>();
    }
}

class Node
{
    public string Data;
    public Node Left, Right;

    public Node(string data)
    {
        Data = data;
        Left = Right = null;
    }
}

class BinarySearchTree
{
    private Node root;

    public BinarySearchTree()
    {
        root = null;
    }

    public void Insert(string data)
    {
        root = InsertRec(root, data);
    }

    private Node InsertRec(Node root, string data)
    {
        if (root == null)
        {
            root = new Node(data);
            return root;
        }

        if (string.Compare(data, root.Data, StringComparison.Ordinal) < 0)
            root.Left = InsertRec(root.Left, data);
        else if (string.Compare(data, root.Data, StringComparison.Ordinal) > 0)
            root.Right = InsertRec(root.Right, data);
```

```
        return root;
    }

    public bool Search(string data)
    {
        return SearchRec(root, data);
    }

    private bool SearchRec(Node root, string data)
    {
        if (root == null)
            return false;

        if (data == root.Data)
            return true;

        if (string.Compare(data, root.Data, StringComparison.Ordinal) < 0)
            return SearchRec(root.Left, data);

        return SearchRec(root.Right, data);
    }

    public void AddWordsToAlan(string alanAdi, string paragraph)
    {
        Node alanNode = FindNode(root, alanAdi);
        if (alanNode != null)
        {
            UM_Alanı umAlanı = new UM_Alanı(alanAdi);

            string[] words = paragraph.Split(new[] { ' ', ',', '.', ':', ';',
'\t', '\n', '\r' }, StringSplitOptions.RemoveEmptyEntries);

            foreach (string word in words)
            {
                umAlanı.Words.Add(word.ToLower());
            }

            Console.WriteLine($"Words for {alanAdi}: {string.Join(", ",
umAlanı.Words)}");


        }
        else
        {
            Console.WriteLine($"{alanAdi} not found.");
        }
    }

    private Node FindNode(Node root, string data)
    {
        if (root == null)
            return null;

        if (string.Compare(data, root.Data, StringComparison.Ordinal) == 0)
            return root;

        if (string.Compare(data, root.Data, StringComparison.Ordinal) < 0)
            return FindNode(root.Left, data);

        return FindNode(root.Right, data);
    }
```

1.a.2 Açıklama

1. // UM_Alanı Sınıfı:
    - `AlanAdi`: Bir alanın adını temsil eden bir özelliktir.
    - `Words`: Bir alanın içinde bulunan kelimeleri içeren bir liste.
    - `UM_Alanı(string alanAdi)`: Alan adını parametre olarak alan kurucu metod aracılığıyla bir UM_Alanı nesnesi oluşturulur.
2. Node Sınıfı:
    - `Data`: Düğümün içinde saklanan veriyi temsil eder.
    - `Left` ve `Right`: Düğümün sol ve sağ alt dallarını temsil eden referanslar.
    - `Node(string data)`: Veriyi parametre olarak alan kurucu metod aracılığıyla bir Node nesnesi oluşturulur.
3. BinarySearchTree Sınıfı:
    - `root`: Ağacın kök düğümünü tutan bir referanstır.
    - `BinarySearchTree()`: Boş bir ikili arama ağacı oluşturan kurucu metod.
    - `Insert(string data)`: Veriyi ikili arama ağacına ekleyen metod.
    - `InsertRec(Node root, string data)`: Veriyi ağaca eklemek için kullanılan özel bir rekürsif metoddur.
    - `Search(string data)`: Veriyi ağaçta arayan metod.
    - `SearchRec(Node root, string data)`: Veriyi ağaçta aramak için kullanılan özel bir rekürsif metoddur.
    - `AddWordsToAlan(string alanAdi, string paragraph)`: Belirli bir alana belirli bir paragraftaki kelimeleri ekleyen metod.
    - `FindNode(Node root, string data)`: Belirli bir veriyi içeren düğümü bulan özel bir rekürsif metoddur.

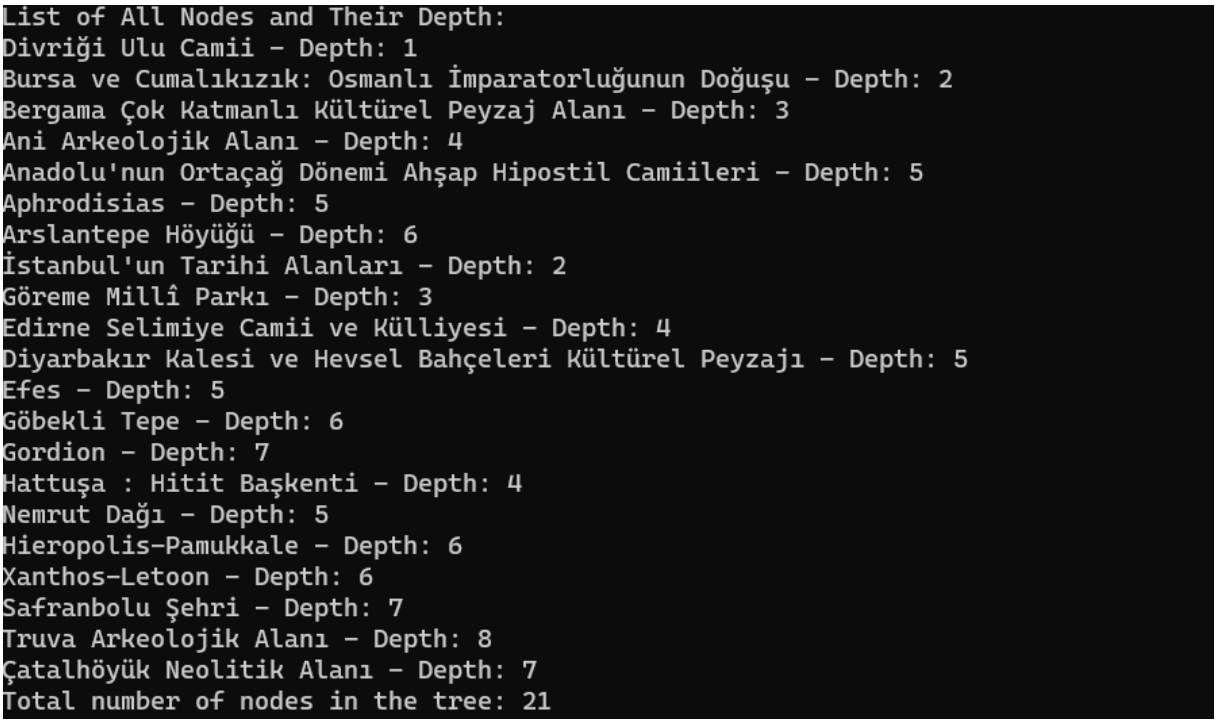
1.b Ağacın Derinliğini, Düğüm Sayısını ve Ağaçtaki Bilgileri Ekrana Listeleme

1.b.1 Kodlar

```
public void ListAllNodesAndDepth()
 {
     Console.WriteLine("List of All Nodes and Their Depth:");
     ListAllNodesAndDepth(root, 1);
 }

 private void ListAllNodesAndDepth(Node node, int depth)
 {
     if (node != null)
     {
         Console.WriteLine($"{node.Data} - Depth: {depth}");
         ListAllNodesAndDepth(node.Left, depth + 1);
         ListAllNodesAndDepth(node.Right, depth + 1);
     }
 }


 public int CountNodes()
 {
```

```
    return CountNodes(root);
}

private int CountNodes(Node node)
{
    if (node == null)
        return 0;

    return 1 + CountNodes(node.Left) + CountNodes(node.Right);
}
```

1.b.2 Ekran görüntüleri



1.b.3 Açıklama

1. // ListAllNodesAndDepth Metodu:
    - `ListAllNodesAndDepth()`: Bu metod, ağaçtaki tüm düğümleri ve derinliklerini listeleyen bir metoddur.
    - `ListAllNodesAndDepth(Node node, int depth)`: Bu özel rekürsif metot, belirli bir düğümden başlayarak tüm alt düğümleri derinlikleriyle birlikte listeleyen bir işlemdir.
        - Eğer düğüm null değilse, düğümün verisini ve derinliğini ekrana yazdırır.
        - Sol ve sağ alt düğümleri için aynı işlemi rekürsif olarak çağırır, ancak derinliği bir artırır.
2. CountNodes Metodu:
    - `CountNodes()`: Bu metod, ağaçtaki toplam düğüm sayısını döndüren bir metoddur.
    - `CountNodes(Node node)`: Bu özel rekürsif metot, belirli bir düğümden başlayarak tüm alt düğümleri sayan bir işlemdir.
        - Eğer düğüm null ise, 0 döndürür.

- Eğer düğüm null değilse, 1 (mevcut düğüm) ile sol ve sağ alt düğümlerinin toplam düğüm sayılarını toplar.

1.c UM_Alanlarının İsimlerinin Listelenmesi

1.c.1 Kodlar

```
    public void ListAlansBetweenLetters(char startLetter, char endLetter)
    {
        ListAlansBetweenLetters(root, startLetter, endLetter);
    }

    private void ListAlansBetweenLetters(Node root, char startLetter, char
endLetter)
    {
        if (root == null)
            return;


        ListAlansBetweenLetters(root.Left, startLetter, endLetter);

        string alanAdi = root.Data;
        char firstChar = char.ToLower(alanAdi[0]);

        if (firstChar >= char.ToLower(startLetter) && firstChar <=
char.ToLower(endLetter))
        {
            Console.WriteLine(alanAdi);
        }

        ListAlansBetweenLetters(root.Right, startLetter, endLetter);
    }
}
```

1.c.2 Ekran görüntüleri

```
Enter the start letter: A
Enter the end letter: K
Anadolu'nun Ortaçağ Dönemi Ahşap Hipostil Camiileri
Ani Arkeolojik Alanı
Aphrodisias
Arslantepe Höyüğü
Bergama Çok Katmanlı Kültürel Peyzaj Alanı
Bursa ve Cumalıkızık: Osmanlı İmparatorluğunun Doğuşu
Divriği Ulu Camii
Diyarbakır Kalesi ve Hevsel Bahçeleri Kültürel Peyzajı
Edirne Selimiye Camii ve Külliyesi
Efes
Gordion
Göbekli Tepe
Göreme Millî Parkı
Hattuşa : Hitit Başkenti
Hieropolis-Pamukkale
İstanbul'un Tarihi Alanları
```

1.d Özyineli olarak Dengeli Bir Ağaç Oluşturulması

1.d.1 Kodlar

```
    public void BuildBalancedTree(string[] sortedArray)
    {
        root = BuildBalancedTreeRec(sortedArray, 0, sortedArray.Length - 1);
    }

    private Node BuildBalancedTreeRec(string[] sortedArray, int start, int end)
    {
        if (start > end)
            return null;

        int mid = (start + end) / 2;
        Node newNode = new Node(sortedArray[mid]);

        newNode.Left = BuildBalancedTreeRec(sortedArray, start, mid - 1);
        newNode.Right = BuildBalancedTreeRec(sortedArray, mid + 1, end);

        return newNode;
    }


    public void BalanceTree()
    {
        string[] sortedArray = InOrderToArray(root);
        root = BuildBalancedTreeRec(sortedArray, 0, sortedArray.Length - 1);
    }

    private string[] InOrderToArray(Node root)
    {
        System.Collections.Generic.List<string> list = new
System.Collections.Generic.List<string>();
        InOrderToArrayRec(root, list);
        return list.ToArray();
    }


    private void InOrderToArrayRec(Node root,
System.Collections.Generic.List<string> list)
    {
        if (root != null)
        {
            InOrderToArrayRec(root.Left, list);
            list.Add(root.Data);
            InOrderToArrayRec(root.Right, list);
        }
    }
}
```

1.d.2 Ekran görüntüleri

```
Dengeli İkili Arama Ağacı:
Anadolu'nun Ortaçağ Dönemi Ahşap Hipostil Camiileri
Ani Arkeolojik Alanı
Aphrodisias
Arslantepe Höyüğü
Bergama Çok Katmanlı Kültürel Peyzaj Alanı
Bursa ve Cumalıkızık: Osmanlı İmparatorluğunun Doğuşu
Çatalhöyük Neolitik Alanı
Divriği Ulu Camii
Diyarbakır Kalesi ve Hevsel Bahçeleri Kültürel Peyzajı
Edirne Selimiye Camii ve Külliyesi
Efes
Gordion
Göbekli Tepe
Göreme Millî Parkı
Hattuşa : Hitit Başkenti
Hieropolis-Pamukkale
İstanbul'un Tarihi Alanları
Nemrut Dağı
Safranbolu Şehri
Truva Arkeolojik Alanı
Xanthos-Letoon
```

1.d.3 Açıklama

1. // BuildBalancedTree Metodu:
   - `BuildBalancedTree(string[] sortedArray)`: Bu metod, sıralanmış bir dizi alır ve bu diziyi kullanarak bir dengeli ikili arama ağacı oluşturur.
   - `BuildBalancedTreeRec(string[] sortedArray, int start, int end)`: Bu özel rekürsif metot, bir alt diziyi alır ve bu diziyi kullanarak bir düğüm oluşturur. Daha sonra sol ve sağ alt ağaçları oluşturmak için aynı işlemi rekürsif olarak çağırır.
     - Başlangıç (`start`) ve bitiş (`end`) indisleri arasındaki orta eleman, yeni düğüm oluşturmak için kullanılır.
2. BalanceTree Metodu:
   - `BalanceTree()`: Bu metod, mevcut ikili arama ağacını dengeler. Bunun için ağacın in-order sıralamasını alır, bu sıralamayı kullanarak bir diziyi oluşturur ve ardından bu diziyi kullanarak dengeli bir ağaç oluşturur.
   - `InOrderToArray(Node root)`: Bu metod, ağacın in-order sıralamasını alır ve bir diziye çevirir.
   - `InOrderToArrayRec(Node root, List<string> list)`: Bu özel rekürsif metot, ağacın in-order gezinmesini yapar ve her düğümü ziyaret edildiği sırayla bir listeye ekler.

.

2) HASH TABLE

Visual Studio c#

2.a UM_Alanı Nesnelerini Hash Table'a Yerleştirme

2.a.1 Kaynak Kod

```
using System;
using System.Collections;
using System.Collections.Generic;

class Program
{
    public class UM_Alanı
    {
        public string Alan_Adı { get; set; }
        public List<string> İl_Adları { get; set; }
        public int İlan_Yılı { get; set; }
    }

    static void Main()
    {
        Hashtable alanHash = new Hashtable();

        string[] mirasAlanları =  {
            "Divriği Ulu Camii ve Darüşşifası,Sivas,1985",
            "İstanbul'un Tarihi Alanları,İstanbul,1985",
            "Göreme Millî Parkı ve Kapadokya,Nevşehir,1985",
            "Hattuşa: Hitit Başkenti,Çorum,1986",
            "Nemrut Dağı,Adıyaman,1987",
            "Hieropolis-Pamukkale,Denizli,1988",
            "Xanthos-Letoon,Antalya-Muğla,1988",
            "Safranbolu Şehri,Karabük,1994",
            "Truva Arkeolojik Alanı,Çanakkale,1998",
            "Edirne Selimiye Camii ve Külliyesi,Edirne,2011",
            "Çatalhöyük Neolitik Alanı,Konya,2012",
            "Bursa ve Cumalıkızık: Osmanlı İmparatorluğunun Doğuşu,Bursa,2014",
            "Bergama Çok Katmanlı Kültürel Peyzaj Alanı,İzmir,2014",
            "Diyarbakır Kalesi ve Hevsel Bahçeleri Kültürel
Peyzajı,Diyarbakır,2015",
            "Efes,İzmir,2015",
            "Ani Arkeolojik Alanı,Kars,2016",
            "Aphrodisias,Aydın,2017",
            "Göbekli Tepe,Şanlıurfa,2018",
            "Arslantepe Höyüğü,Malatya,2021",
            "Gordion,Ankara,2023",
            "Anadolu'nun Ortaçağ Dönemi Ahşap Hipostil Camiileri,(Konya-
Eşrefoğlu Camii- Kastamonu-Mahmut Bey Camii- Eskişehir-Sivrihisar Camii- Afyon-
Afyon Ulu Camii- Ankara-Arslanhane Camii),2023"

        };

        foreach (var alanVerisi in mirasAlanları)
        {
            string[] alanBilgileri = alanVerisi.Split(',');
            UM_Alanı umAlan = new UM_Alanı
            {
                Alan_Adı = alanBilgileri[0],
                İl_Adları = new List<string> { alanBilgileri[1] },
                İlan_Yılı = int.Parse(alanBilgileri[2])
            };

            alanHash.Add(umAlan.Alan_Adı, umAlan);
```

```
        }

        Console.WriteLine("Hash Table'ın İlk Hali:");
        TumAlanlariYazdir(alanHash);

        Console.Write("Değiştirmek istediğiniz alanın adını girin: ");
        string eskiAlanAdi = Console.ReadLine();

        if (alanHash.ContainsKey(eskiAlanAdi))
        {
            UM_Alanı eskiAlan = (UM_Alanı)alanHash[eskiAlanAdi];

            Console.Write("Yeni adı girin: ");
            string yeniAlanAdi = Console.ReadLine();

            eskiAlan.Alan_Adı = yeniAlanAdi;


            alanHash.Remove(eskiAlanAdi);
            alanHash.Add(yeniAlanAdi, eskiAlan);

            Console.WriteLine($"{eskiAlanAdi} adlı alanın adı, {yeniAlanAdi}
olarak değiştirildi.");

            Console.WriteLine("Güncellenmiş Hash Table:");
            TumAlanlariYazdir(alanHash);
        }
        else
        {
            Console.WriteLine($"{eskiAlanAdi} adlı bir UNESCO mirası alanı
bulunamadı.");
        }
    }

    static void TumAlanlariYazdir(Hashtable alanHash)
    {
        foreach (DictionaryEntry entry in alanHash)
        {
            UM_Alanı alan = (UM_Alanı)entry.Value;
            Console.WriteLine($"Alan Adı: {alan.Alan_Adı} - İller:
{string.Join(", ", alan.İl_Adları)} - İlan Yılı: {alan.İlan_Yılı}");
        }
    }
}
```

2.a.2 Ekran Görüntüleri

2.b Hash Table Güncelleme

2.b.1 Kaynak Kod

```
using System;
using System.Collections;
using System.Collections.Generic;

class Program
{
    public class UM_Alanı
    {
        public string Alan_Adı { get; set; }
        public List<string> İl_Adları { get; set; }
        public int İlan_Yılı { get; set; }
    }

    static void Main()
    {
        Hashtable alanHash = new Hashtable();

        string[] mirasAlanları =  {
            "Divriği Ulu Camii ve Darüşşifası,Sivas,1985",
            "İstanbul'un Tarihi Alanları,İstanbul,1985",
            "Göreme Millî Parkı ve Kapadokya,Nevşehir,1985",
            "Hattuşa: Hitit Başkenti,Çorum,1986",
            "Nemrut Dağı,Adıyaman,1987",
            "Hieropolis-Pamukkale,Denizli,1988",
            "Xanthos-Letoon,Antalya-Muğla,1988",
            "Safranbolu Şehri,Karabük,1994",
            "Truva Arkeolojik Alanı,Çanakkale,1998",
            "Edirne Selimiye Camii ve Külliyesi,Edirne,2011",
            "Çatalhöyük Neolitik Alanı,Konya,2012",
            "Bursa ve Cumalıkızık: Osmanlı İmparatorluğunun Doğuşu,Bursa,2014",
            "Bergama Çok Katmanlı Kültürel Peyzaj Alanı,İzmir,2014",
            "Diyarbakır Kalesi ve Hevsel Bahçeleri Kültürel
Peyzajı,Diyarbakır,2015",
            "Efes,İzmir,2015",
            "Ani Arkeolojik Alanı,Kars,2016",
            "Aphrodisias,Aydın,2017",
            "Göbekli Tepe,Şanlıurfa,2018",
            "Arslantepe Höyüğü,Malatya,2021",
            "Gordion,Ankara,2023",
            "Anadolu'nun Ortaçağ Dönemi Ahşap Hipostil Camiileri,(Konya-
Eşrefoğlu Camii- Kastamonu-Mahmut Bey Camii- Eskişehir-Sivrihisar Camii- Afyon-
Afyon Ulu Camii- Ankara-Arslanhane Camii),2023"

        };

        foreach (var alanVerisi in mirasAlanları)
        {
            string[] alanBilgileri = alanVerisi.Split(',');
            UM_Alanı umAlan = new UM_Alanı
            {
                Alan_Adı = alanBilgileri[0],
                İl_Adları = new List<string> { alanBilgileri[1] },
                İlan_Yılı = int.Parse(alanBilgileri[2])
            };

            alanHash.Add(umAlan.Alan_Adı, umAlan);
        }

        Console.WriteLine("Hash Table'ın İlk Hali:");
        TumAlanlariYazdir(alanHash);
```

```
        Console.Write("Değiştirmek istediğiniz alanın adını girin: ");
        string eskiAlanAdi = Console.ReadLine();

        if (alanHash.ContainsKey(eskiAlanAdi))
        {
            UM_Alanı eskiAlan = (UM_Alanı)alanHash[eskiAlanAdi];

            Console.Write("Yeni adı girin: ");
            string yeniAlanAdi = Console.ReadLine();

            eskiAlan.Alan_Adı = yeniAlanAdi;

            alanHash.Remove(eskiAlanAdi);
            alanHash.Add(yeniAlanAdi, eskiAlan);

            Console.WriteLine($"{eskiAlanAdi} adlı alanın adı, {yeniAlanAdi}
olarak değiştirildi.");

            Console.WriteLine("Güncellenmiş Hash Table:");
            TumAlanlariYazdir(alanHash);
        }
        else
        {
            Console.WriteLine($"{eskiAlanAdi} adlı bir UNESCO mirası alanı
bulunamadı.");
        }
    }

    static void TumAlanlariYazdir(Hashtable alanHash)
    {
        foreach (DictionaryEntry entry in alanHash)
        {
            UM_Alanı alan = (UM_Alanı)entry.Value;
            Console.WriteLine($"Alan Adı: {alan.Alan_Adı} - İller:
{string.Join(", ", alan.İl_Adları)} - İlan Yılı: {alan.İlan_Yılı}");
        }
    }
}
```

2.b.2 Ekran Görüntüsü

2.b.3 Açıklama

1. // `mirasAlanları` adlı bir string dizisi tanımlanır. Her bir eleman UNESCO mirası bir alanın bilgilerini içerir (alan adı, il adı, ilan yılı).
2. `alanHash` adlı bir `Hashtable` oluşturulur. Bu hashtable, her bir alan adını anahtar olarak ve `UM_Alanı` sınıfını değer olarak içerir.
3. `mirasAlanları` dizisi üzerinde döngü ile her bir alanın bilgileri ayrıştırılır ve `UM_Alanı` sınıfından bir nesne oluşturularak `alanHash` hashtable'ına eklenir.
4. `TumAlanlariYazdir` fonksiyonu, hashtable içindeki tüm alanları ekrana yazdırmak için kullanılır.
5. Kullanıcıdan güncellenmek istenen alanın adı (`eskiAlanAdi`) alınır.
6. `alanHash` içinde `eskiAlanAdi` adında bir anahtar var mı kontrol edilir (`alanHash.ContainsKey(eskiAlanAdi)`). Eğer varsa, alanın bilgileri alınır ve kullanıcıdan yeni alan adı (`yeniAlanAdi`) istenir.
7. Eski alanın adı güncellenir (`eskiAlan.Alan_Adı = yeniAlanAdi`).
8. Eski alanın adı `alanHash` hashtable'ından kaldırılır (`alanHash.Remove(eskiAlanAdi)`).
9. Güncellenmiş alan adı ile birlikte `UM_Alanı` nesnesi tekrar `alanHash` hashtable'ına eklenir (`alanHash.Add(yeniAlanAdi, eskiAlan)`).
10. Kullanıcıya yapılan güncelleme hakkında bilgi verilir.
11. Güncellenmiş hashtable, `TumAlanlariYazdir` fonksiyonu kullanılarak ekrana yazdırılır.

3) HEAP VERİ YAPISI (SINIFI)

3.a Heap

3.a.1 Kaynak Kod

```
using System;
using System.Collections.Generic;
```

```
class Program
{
    public class UM_Alanı
    {
        public string Alan_Adı { get; set; }
        public List<string> İl_Adları { get; set; }
        public int İlan_Yılı { get; set; }
    }

    public class Heap
    {
        private List<UM_Alanı> heapList;

        public Heap()
        {
            heapList = new List<UM_Alanı>();
        }

        public void Insert(UM_Alanı item)
        {
            heapList.Add(item);
            HeapifyUp(heapList.Count - 1);
        }

        private void HeapifyUp(int index)
        {
            int parentIndex = (index - 1) / 2;

            while (index > 0 && heapList[index].İlan_Yılı <
heapList[parentIndex].İlan_Yılı)
            {
                Swap(index, parentIndex);
                index = parentIndex;
                parentIndex = (index - 1) / 2;
            }
        }

        private void Swap(int index1, int index2)
        {
            UM_Alanı temp = heapList[index1];
            heapList[index1] = heapList[index2];
            heapList[index2] = temp;
        }

        public void PrintAll()
        {
            foreach (var alan in heapList)
            {
                Console.WriteLine($"Alan Adı: {alan.Alan_Adı} - İller:
{string.Join(", ", alan.İl_Adları)} - İlan Yılı: {alan.İlan_Yılı}");
            }
        }
    }

    static void Main()
    {
        Heap alanHeap = new Heap();

        string[] mirasAlanları = {
            "Divriği Ulu Camii ve Darüşşifası,Sivas,1985",
            "İstanbul'un Tarihi Alanları,İstanbul,1985",
            "Göreme Millî Parkı ve Kapadokya,Nevşehir,1985",
```

```
            "Hattuşa: Hitit Başkenti,Çorum,1986",
            "Nemrut Dağı,Adıyaman,1987",
            "Hieropolis-Pamukkale,Denizli,1988",
            "Xanthos-Letoon,Antalya-Muğla,1988",
            "Safranbolu Şehri,Karabük,1994",
            "Truva Arkeolojik Alanı,Çanakkale,1998",
            "Edirne Selimiye Camii ve Külliyesi,Edirne,2011",
            "Çatalhöyük Neolitik Alanı,Konya,2012",
            "Bursa ve Cumalıkızık: Osmanlı İmparatorluğunun Doğuşu,Bursa,2014",
            "Bergama Çok Katmanlı Kültürel Peyzaj Alanı,İzmir,2014",
            "Diyarbakır Kalesi ve Hevsel Bahçeleri Kültürel
Peyzajı,Diyarbakır,2015",
            "Efes,İzmir,2015",
            "Ani Arkeolojik Alanı,Kars,2016",
            "Aphrodisias,Aydın,2017",
            "Göbekli Tepe,Şanlıurfa,2018",
            "Arslantepe Höyüğü,Malatya,2021",
            "Gordion,Ankara,2023",
            "Anadolu'nun Ortaçağ Dönemi Ahşap Hipostil Camiileri,(Konya-
Eşrefoğlu Camii- Kastamonu-Mahmut Bey Camii- Eskişehir-Sivrihisar Camii- Afyon-
Afyon Ulu Camii- Ankara-Arslanhane Camii),2023"

        };

        foreach (var alanVerisi in mirasAlanları)
        {
            string[] alanBilgileri = alanVerisi.Split(',');
            UM_Alanı umAlan = new UM_Alanı
            {
                Alan_Adı = alanBilgileri[0],
                İl_Adları = new List<string> { alanBilgileri[1] },
                İlan_Yılı = int.Parse(alanBilgileri[2])
            };

            alanHeap.Insert(umAlan);
        }

        alanHeap.PrintAll();
    }
}
```

3.b Min Heap'e Yerleştirme

3.b.1 Kaynak Kod

```
using System;
using System.Collections.Generic;

class Program
{
    public class UM_Alanı
    {
        public string Alan_Adı { get; set; }
        public List<string> İl_Adları { get; set; }
        public int İlan_Yılı { get; set; }
    }

    public class Heap
    {
        private List<UM_Alanı> heapList;

        public Heap()
        {
            heapList = new List<UM_Alanı>();
```

```
        }

        public void Insert(UM_Alanı item)
        {
            heapList.Add(item);
            HeapifyUp(heapList.Count - 1);
        }

        private void HeapifyUp(int index)
        {
            int parentIndex = (index - 1) / 2;

            while (index > 0 && string.Compare(heapList[index].Alan_Adı,
heapList[parentIndex].Alan_Adı, StringComparison.Ordinal) < 0)
            {
                Swap(index, parentIndex);
                index = parentIndex;
                parentIndex = (index - 1) / 2;
            }
        }

        private void Swap(int index1, int index2)
        {
            UM_Alanı temp = heapList[index1];
            heapList[index1] = heapList[index2];
            heapList[index2] = temp;
        }

        public void PrintAll()
        {
            foreach (var alan in heapList)
            {
                Console.WriteLine($"Alan Adı: {alan.Alan_Adı} - İller:
{string.Join(", ", alan.İl_Adları)} - İlan Yılı: {alan.İlan_Yılı}");
            }
        }
    }

    static void Main()
    {
        Heap alanHeap = new Heap();

        string[] mirasAlanları = {
            "Divriği Ulu Camii ve Darüşşifası,Sivas,1985",
            "İstanbul'un Tarihi Alanları,İstanbul,1985",
            "Göreme Millî Parkı ve Kapadokya,Nevşehir,1985",
            "Hattuşa: Hitit Başkenti,Çorum,1986",
            "Nemrut Dağı,Adıyaman,1987",
            "Hieropolis-Pamukkale,Denizli,1988",
            "Xanthos-Letoon,Antalya-Muğla,1988",
            "Safranbolu Şehri,Karabük,1994",
            "Truva Arkeolojik Alanı,Çanakkale,1998",
            "Edirne Selimiye Camii ve Külliyesi,Edirne,2011",
            "Çatalhöyük Neolitik Alanı,Konya,2012",
            "Bursa ve Cumalıkızık: Osmanlı İmparatorluğunun Doğuşu,Bursa,2014",
            "Bergama Çok Katmanlı Kültürel Peyzaj Alanı,İzmir,2014",
            "Diyarbakır Kalesi ve Hevsel Bahçeleri Kültürel
Peyzajı,Diyarbakır,2015",
            "Efes,İzmir,2015",
            "Ani Arkeolojik Alanı,Kars,2016",
            "Aphrodisias,Aydın,2017",
            "Göbekli Tepe,Şanlıurfa,2018",
            "Arslantepe Höyüğü,Malatya,2021",
```

```
            "Gordion,Ankara,2023",
            "Anadolu'nun Ortaçağ Dönemi Ahşap Hipostil Camiileri,(Konya-
Eşrefoğlu Camii- Kastamonu-Mahmut Bey Camii- Eskişehir-Sivrihisar Camii- Afyon-
Afyon Ulu Camii- Ankara-Arslanhane Camii),2023"

        };

        foreach (var alanVerisi in mirasAlanları)
        {
            string[] alanBilgileri = alanVerisi.Split(',');
            UM_Alanı umAlan = new UM_Alanı
            {
                Alan_Adı = alanBilgileri[0],
                İl_Adları = new List<string> { alanBilgileri[1] },
                İlan_Yılı = int.Parse(alanBilgileri[2])
            };

            alanHeap.Insert(umAlan);
        }

        alanHeap.PrintAll();
    }
}
```

3.c Min Heap Listeleme

3.c.1 Kaynak Kod

```
using System;
using System.Collections.Generic;
using System.Linq;

class Program
{
    public class UM_Alanı
    {
        public string Alan_Adı { get; set; }
        public List<string> İl_Adları { get; set; }
        public int İlan_Yılı { get; set; }
    }

    static void Main()
    {
        SortedSet<UM_Alanı> alanHeap = new
SortedSet<UM_Alanı>(Comparer<UM_Alanı>.Create((x, y) =>
string.Compare(x.Alan_Adı, y.Alan_Adı, StringComparison.Ordinal)));

        string[] mirasAlanları =  {
            "Divriği Ulu Camii ve Darüşşifası,Sivas,1985",
            "İstanbul'un Tarihi Alanları,İstanbul,1985",
            "Göreme Millî Parkı ve Kapadokya,Nevşehir,1985",
            "Hattuşa: Hitit Başkenti,Çorum,1986",
            "Nemrut Dağı,Adıyaman,1987",
            "Hieropolis-Pamukkale,Denizli,1988",
            "Xanthos-Letoon,Antalya-Muğla,1988",
            "Safranbolu Şehri,Karabük,1994",
            "Truva Arkeolojik Alanı,Çanakkale,1998",
            "Edirne Selimiye Camii ve Külliyesi,Edirne,2011",
            "Çatalhöyük Neolitik Alanı,Konya,2012",
            "Bursa ve Cumalıkızık: Osmanlı İmparatorluğunun Doğuşu,Bursa,2014",
            "Bergama Çok Katmanlı Kültürel Peyzaj Alanı,İzmir,2014",
            "Diyarbakır Kalesi ve Hevsel Bahçeleri Kültürel
Peyzajı,Diyarbakır,2015",
```

```
            "Efes,İzmir,2015",
            "Ani Arkeolojik Alanı,Kars,2016",
            "Aphrodisias,Aydın,2017",
            "Göbekli Tepe,Şanlıurfa,2018",
            "Arslantepe Höyüğü,Malatya,2021",
            "Gordion,Ankara,2023",
            "Anadolu'nun Ortaçağ Dönemi Ahşap Hipostil Camiileri,(Konya-
Eşrefoğlu Camii- Kastamonu-Mahmut Bey Camii- Eskişehir-Sivrihisar Camii- Afyon-
Afyon Ulu Camii- Ankara-Arslanhane Camii),2023"

        };

        foreach (var alanVerisi in mirasAlanları)
        {
            string[] alanBilgileri = alanVerisi.Split(',');
            UM_Alanı umAlan = new UM_Alanı
            {
                Alan_Adı = alanBilgileri[0],
                İl_Adları = new List<string> { alanBilgileri[1] },
                İlan_Yılı = int.Parse(alanBilgileri[2])
            };

            alanHeap.Add(umAlan);
        }

        TumAlanlariYazdir(alanHeap);
    }

    static void TumAlanlariYazdir(SortedSet<UM_Alanı> alanHeap)
    {

        var ilkUcAlan = alanHeap.Take(3);

        foreach (var alan in ilkUcAlan)
        {
            Console.WriteLine($"Alan Adı: {alan.Alan_Adı} - İller:
{string.Join(", ", alan.İl_Adları)} - İlan Yılı: {alan.İlan_Yılı}");
        }
    }
}
```

3.c.2 Ekran Görüntüsü

```
Alan Adı: Anadolu'nun Ortaçağ Dönemi Ahşap Hipostil Camiileri - İller: (Konya-Eşrefoğlu Camii- Kastamonu-Mahmut Bey Cami
i- Eskişehir-Sivrihisar Camii- Afyon-Afyon Ulu Camii- Ankara-Arslanhane Camii) - İlan Yılı: 2023
Alan Adı: Ani Arkeolojik Alanı - İller: Kars - İlan Yılı: 2016
Alan Adı: Aphrodisias - İller: Aydın - İlan Yılı: 2017
```

3.c.3 Açıklama

// Bu kod, `alanHeap` adlı `SortedSet` nesnesini oluştururken, karşılaştırma işlemi için `Comparer<UM_Alanı>.Create` metodu kullanılarak bir karşılaştırma fonksiyonu belirlenir. Bu fonksiyon, `UM_Alanı` nesnelerini alan adlarına göre karşılaştırır ve küme içindeki öğelerin sıralanmasını bu temele göre yapar.

Bu sayede, `alanHeap` içindeki `UM_Alanı` nesneleri, `Alan_Adı` özelliğine göre sıralanmış bir şekilde tutulur ve bu sıralama, küme içindeki öğelerin belirli bir düzene göre erişilmesine olanak tanır. Bu özellik, programın belirli bir

sayıda öğeyi, özellikle ilk üç öğeyi, sıralama önceliğine göre elde etmesine olanak tanır.

4.a Sıralama Algoritmaları

4.a.1 Kaynak Kod

```
using System;

namespace SortingAlgorithm
{
    class Program
    {
        static void Main(string[] args)
        {

            int[] sampleArray = { 5, 2, 9, 1, 5, 6 };


            ISorter bubbleSorter = new Bubble();
            Console.WriteLine("Bubble Sort Örneği:");
            Console.WriteLine("Orijinal Dizi: " + ArrayToString(sampleArray));
            bubbleSorter.Execute(sampleArray);
            Console.WriteLine("Sıralı Dizi: " + ArrayToString(sampleArray));
            Console.WriteLine();

            ISorter quickSorter = new Quick();
            Console.WriteLine("Quick Sort Örneği:");
            Console.WriteLine("Orijinal Dizi: " + ArrayToString(sampleArray));
            quickSorter.Execute(sampleArray);
            Console.WriteLine("Sıralı Dizi: " + ArrayToString(sampleArray));

            Console.ReadLine();
        }

        static string ArrayToString(int[] array)
        {
            return "[" + string.Join(", ", array) + "]";
        }
    }


    public interface ISorter
    {
        string Description { get; }
        void Execute(int[] array);
    }


    public class Bubble : ISorter
    {
        public string Description => "Bubble Sort Sıralama Algoritması";

        public void Execute(int[] array)
        {
            int n = array.Length;
            int temp;

            for (int i = 0; i < n - 1; i++)
            {
                for (int j = 0; j < n - i - 1; j++)
                {
                    if (array[j] > array[j + 1])
                    {
```

```
                    temp = array[j];
                    array[j] = array[j + 1];
                    array[j + 1] = temp;
                }
            }
        }
    }

    public class Quick : ISorter
    {
        public string Description => "Quick Sort Sıralama Algoritması";

        public void Execute(int[] array)
        {
            Sort(0, array.Length - 1, array);
        }

        private void Sort(int leftValue, int rightValue, int[] array)
        {
            if (leftValue < rightValue)
            {
                int pivotIndex = Partition(leftValue, rightValue, array);

                Sort(leftValue, pivotIndex - 1, array);
                Sort(pivotIndex + 1, rightValue, array);
            }
        }

        private int Partition(int leftValue, int rightValue, int[] array)
        {
            int pivotValue = array[rightValue];
            int i = leftValue - 1;

            for (int j = leftValue; j < rightValue; j++)
            {
                if (array[j] < pivotValue)
                {
                    i++;


                    int temp = array[i];
                    array[i] = array[j];
                    array[j] = temp;
                }
            }


            int tempSwap = array[i + 1];
            array[i + 1] = array[rightValue];
            array[rightValue] = tempSwap;

            return i + 1;
        }
    }
}
```

4.a.2 Açıklama

// Bubble Sort:

1. Algoritma Tanımı: Bubble Sort, her adımda komşu iki elemanı karşılaştırarak gerekli durumlarda yer değiştiren bir sıralama algoritmasıdır.
2. Çalışma Mantığı:
    - İç içe iki döngü kullanılır. Dıştaki döngü, dizinin tamamını kontrol etmek için, içteki döngü ise her bir elemanı karşılaştırmak için kullanılır.
    - Her iç döngü geçişinde, bir eleman diğerinden büyükse, ikisi arasında yer değişimi yapılır.
3. Kod Açıklamaları:
    - Bubble sınıfı, ISorter arabirimini uygular ve Execute metodu içinde Bubble Sort'u gerçekleştirir.
    - Diziyi sıralamak için iç içe iki döngü kullanılır. Elemanlar karşılaştırılıp gerekirse yer değiştirilir.

Quick Sort:

1. Algoritma Tanımı: Quick Sort, bir pivot eleman seçip diğer elemanları bu pivota göre sıralayan bir rekürsif sıralama algoritmasıdır.
2. Çalışma Mantığı:
    - Bir pivot eleman seçilir ve bu elemanın solunda pivot'tan küçük, sağında pivot'tan büyük elemanlar olacak şekilde dizi bölünür.
    - Bu işlem rekürsif olarak her alt dizi üzerinde tekrarlanır.
3. Kod Açıklamaları:
    - Quick sınıfı, ISorter arabirimini uygular ve Execute metodu içinde Quick Sort'u gerçekleştirir.
    - Sort metodu, diziyi bölerek ve pivot elemanı seçerek sıralamayı gerçekleştirir.
    - Partition metodu, pivot elemanının doğru konumunu bulmak için kullanılır. Bu metod ayrıca elemanları bölüp sıralar.

4.b Zaman Karmaşıklığının Hesaplanması

4.b.1 Açıklama

Zaman karmaşıklığı, bir algoritmanın çalışma süresinin girdi boyutuna (n) bağlı olarak nasıl büyüdüğünü gösterir. İşte Bubble Sort ve Quick Sort'un zaman karmaşıklıkları:

Bubble Sort:

- En iyi durum: O(n) - Dizi zaten sıralıysa.
- Ortalama ve en kötü durum: O(n^2) - Karşılaştırmalar ve takas işlemleri n^2 defa yapılır.

Quick Sort:

- Ortalama ve en iyi durum: O(n log n) - Pivot elemanı iyi seçildiğinde.
- En kötü durum: O(n^2) - Pivot elemanı kötü seçildiğinde (örneğin, her zaman dizinin en küçük veya en büyük elemanı).

Sonuç:

- Küçük veri setleri veya neredeyse sıralı veri setleri için Bubble Sort basit ve anlaşılır bir seçenek olabilir.
- Büyük veri setleri için Quick Sort genellikle daha hızlıdır, ancak en kötü durumu göz önünde bulundurmak önemlidir.

- Quick Sort, genellikle genel amaçlı sıralama algoritması olarak tercih edilir, ancak Bubble Sort'un öğrenme ve öğretme amaçları için kullanılması yaygındır.

5) Görselleştirme

5.a Görselleştirmenin Etkisi

5.a.1 Açıklama

Görselleştirmenin, özellikle öğrenme süreçlerini destekleyerek ve soyut kavramları somutlaştırarak, algoritmaları ve veri yapılarını anlamada büyük bir rolü vardır. Bu nedenle, öğrencilerin, geliştiricilerin ve herkesin karmaşık bilgiyi daha iyi kavraması için yaygın bir öğrenme aracı olarak kullanılabilir.

5.b Karşılaştırma Yapılması

5.b.1 Açıklama

Her iki öğrenme aracının da avantajları ve dezavantajları bulunmaktadır. Videolar, geniş bir kitleye hitap edebilirken, etkileşimli araçlar daha kişiselleştirilmiş ve etkileşimli bir öğrenme deneyimi sunabilir. İdeal öğrenme deneyimi, içerik türlerini ve öğrenme stillerini birleştiren bir yaklaşımın bir kombinasyonunu içerebilir.

6) Özdeğerlendirme Tablosu

Proje 3 Maddeleri	Not	Tahmini Not	Açıklama
1 a) Ağaç (UNESCO M. İkili Arama Ağacı)	**10**	**10**	**Yapıldı. UM Alanları ikili arama ağacına eklendi ve her alana bir paragrafın tüm kelimeleri nesne olarak generic list içinde eklendi.**

1 b) Derinlik Bulma, Ağacı Listeleme, Düğüm Sayısı Buldurma, Dengeli Ağaç Derinliği Hesaplama	**10**	**10**	**Yapıldı. Ağacın derinliği ve düğüm sayısı bulundu.**
1 c) Arama ve Listeleme	**10**	**10**	**Yapıldı.İstenen iki harf arasındaki tüm UM Alanları listelendi.**
1 d) Kelime Ağacı Oluşturarak Kelimeleri Sayma	**10**	**10**	**Yapıldı. Özyineli olarak dengeli bir ağaç oluşturuldu.**
2) Hash Tablosu	**15**	**15**	**Yapıldı. UM Alanları HashTable eklendi. Anahtar olarak alan adları değer olarak ise iller ve yılları eklendi. İstenilen şekilde bir UM Alanının ismini değiştirerek güncelleme yapıldı.**
3) Yığın Ağacı (Heap)	**15**	**15**	**Yapıldı. UM Alanları Heap ve Min Heap veri yapısına eklendi. En son ise alfabetik olarak ilk 3 sıradaki UM alanları yazdırıldı.**
4) Sıralama Algoritmaları	**10**	**10**	**Yapıldı. Bubble ve Quick Sort karşılaştırıldı ve avantajları ve dezavantajları yazıldı.**
5.a) Görselleştirmenin DSA öğrenmeye etkisi	**5**	**5**	**Yapıldı.Görselleştirmenin algoritmaları ve veri yapılarını anlamamıza etkisi anlatıldı.**
5.b) Video & Görselleştirme Araçları Karşılaştırma	**5**	**5**	**Yapıldı. Veri Yapılarını öğrenme açısından Videolar ile Etkileşimli Görselleştirme Araçlarını karşılaştırıldı.**
6) Özdeğerlendirme Tablosu	**10**	**10**	**Yapıldı. Tablo eksiksiz bir şekilde doldurulduğu.**