**Sakarya Üniversitesi**

**Bilgisayar ve Bilişim Bilimleri Fakültesi**

**Bilişim Sistemleri Mühendisliği Bölümü**

Veri Yapıları 2. Ödevi

Dosya ile AVL Ağaçları İşlem Yapma

B201200372-Furkan TOP

SAKARYA

August, 2022

Veri Yapıları Dersi

*Dosya ile AVL Ağaçları İslem Yapma*

Furkan TOP

B201200372– 1.Öğretim - A Grubu

1. **Özet :**

* Bu ödevde bir sayıların satırları içeren dosya okuyup, her satirin içindeki sayıları toplanıp AVL’ya eklenecek düğümlere atılacaktır.
* Sayılar dizisine içeren düğümleri, AVL ağacına eklendikten sonra, ağacın dengesizliğine göre hareket eden düğümlere ait sayı dizisine değişiklik yapılacaktır.
* Girdi olarak “*Sayılar.txt*” dosyasını okuma yapılacaktır.
* Çıktı olarak oluşturan AVL ağacı *Preorder* şekliyle ekrana yazdırılacaktır. Ekranda her düğüme ait dizinin içinde kalan elemanlar yazılacaktır.
* Bir süreç olarak algoritmamız, sahip olduğumuz belirli kısıtlamalara göre **AVL** ve **Radiks Sıralama** üzerinde belirli işlemleri uygular.

1. **Geliştirilen Yazılım :**

* Çözümümüz esas olarak 5 sınıfa dayanmaktadır:

|  |  |
| --- | --- |
| Dosya Class | Burda dosyayı okuyup, satırlarını bir string dizisine atılacaktır. |
| Radiks Class | Burda bir string’i alıp, integer sekline bir dizi atanmaktadır. O dizinin içindeki elemanları radiks sıralama algoritmasını kullanarak, küçükten büğüğe sıralanır. |
| AVL | AVL ağacına ait(veri yapısının) ilkel özelliklerini içerir. Düğüm ekleme, dengeleme sağlama gibi işlemleri sağlayan metotları içermektedir. |
| Cozum | Yukarıdaki sınıfları çağrayarak problemin çözümünü çıkatılır. En sonda problemin çözümü ekrana yazdıran metot çalıştırılmaktadır. |
| Main Class | Programin çalıştırıldığı ana sınıfıdır. |

***Not:*** Programı çalıştırırken, dosya okumada oluşabilecek istisnaları ele almak için ektradan (Dosya Bulanamadı) adlı bir exception sınıfı kullandık.

* **Zaman ve Yer Karmaşıklığı:**
* Dosyayı okumak ve tüm verilerini bir string dizisinde satır satır depolamak için, zaman karmaşıklığı O(n)'dir, burada n dosyanın satır sayısıdır.
* m boyutunda bir string diziyi tamsayı dizisine dönüştürmek **O(m)** alır. n dizi(satır) için zaman karmaşıklığı **n\*O(m)** olacaktır.
* Radix sıralama algoritmasının zaman karmaşıklığı **O(nd)**'dir, burada **n**, dizi boyutu ve **d** en büyük sayıdaki basamak sayısıdır. Radix sıralama, fazladan alan gerektirdiğinden yerinde sıralama algoritması değildir. Eşit değerlere sahip öğelerin göreli sırasını korur, bu nedenle kararlı bir sıralamadır.
* AVL ağacı durumunda, aşağıdaki işlemleri kullandık:
* Ekleme, Silme ve Traversal: En iyi durumda, zaman karmaşıklığı **O(logn)**'dur.
* Rotasyonlar: En basit senaryoda, **O(1)** alırlar.
* Yer karmaşıklığı için bu veri yapısı **O(n)** olucaktır.
* AVL ağacının yüksekliği her zaman dengelidir. Yuksekliği hiç bir zaman **log(N)**’den geçmediği için BST ağlarına kıyasla daha iyi arama zaman karmaşıklığı sağlar.(**N** ağaçtaki düğüm sayısıdır).
* **Algoritma Çalıştırma Süresi:**

Aşağıda dosyanın satır sayısını 3 , 10 ,100, 1000 ... vb gibi sözde rasgele uzunklukları geçtikten sonra algoritma yürütme süresi verilmiştir:

|  |  |
| --- | --- |
| Satır sayısı | Time (s) |
| 3 | 0.000000 |
| 10 | 0.000000 |
| 100 | 0.00200 |
| 1000 | 0.01000 |

Yukarıdaki tabloya göre, zaman karmaşıklığı dosyanın uzunluğuyla paralel olduğunu görülmektedir. Tabiki de dosyanın içindeki satırların uzunluğu da çalışma zamanı arttırmaktadır.

1. **Zorlandığım Kısımlar:**

Ödevi yaparken, çoğu konularda daha önceden tecrübeli olduğum için fazla zorluk çekmedim. Fakat her düğümün, AVL ağacı oluşturulma esnasında aldığı yolu nu izlemek bir tik uğraştıcı oldu. Aynı zamanda AVL ağacında dengesizlik olma durumunda, dengeleştirmek biraz zordur.

1. **Öğrendiklerim** :

Aşağı yukarı, aşağıdaki bilgilere sahip oldum:

• Dosya işleme ve string parçalama(bir stringi int diziye dönüştürme)

• AVL Yapısı ile işlemler yapma.

• Radix sort algoritmasını uygulama.