T.C. Sakarya Üniversitesi
Bilgisayar ve Bilişim Bilimleri Fakültesi
Bilgisayar Mühendisliği Bölümü
Veri Yapıları 2.Ödev Raporu

Muhammed Furkan BALEK | G191210069
github.com/DevBalek

AVL Ağacı Ödevi Raporu

Ödevi üç ayrı parçaya böldüm;

Birinci kısımda, yaptığım şey bize verilen Noktalar.txt dosyasının içinden x y z koordinatlarını alıp Her bir noktayı **Chain(zincir halkası)**'in içine koymak oldu. Her sayı okuduğunda controller'ı bir arttırıyor, bu sayı 3e ulaşınca elimdeki **chain* list[3]** arrayinin içindeki verileri **pointChaine(zincir)** bir bir ekliyorum.

PointChainin içine her veri eklendiğinde (ilk veri hariç) eklenen noktadan bir önceki noktayla olan uzaklığını hesaplayıp **int sumOfEachDistance'a** ekleme işlemi yapıyor. Böylece doğru parçasının uzunluğunu hesaplıyorum.

Okunan sıra bittiğinde elimde **pointChain** nesnesi kalıyor. Tam bu kısımda **AvlChain** devreye giriyor. Kök Chainle birlikte **pointChain** verisini de Avl Ağacının ekleme metoduna gönderiyorum. Burada ekleme işlemi BinaryTree'ye oldukça benziyor. Ekleme işlemini **pointChainin** içerisinde bulunan **int sumOfEachDistance** değerine göre yapıyorum. Boş alan bulana kadar recursive olarak ağaçları dolaşıyor, girdiği yerde düğüm yoksa oraya yerleştiriyor ve en son oluşan ağacı maindeki **headOfAvlChain** nesnesine atıyoruz. Böylelikle her atamada yeni Kök düğüm gönderiliyor.

Dengeleme işlemi ise anlatması oldukça karışık her ekleme işlemi yapıldıktan sonra chainHeight ve chainBalance kontrol ediliyor. chainBalance bize yardımcı olan bi kontrolcü, eğer 1'den büyük çıkarsa denge sorunu sağda, -1'den küçük çıkarsa denge sorunu solda olarak algılayıp başka if elselerden kurtarıyor. **chainBalance'ın** yaptığı tek şey ise sol ağaçtan sağ ağacı çıkararak değeri döndürmek. -1, 0, 1 çıkarsa denge teşkil etmiyor. **chainBalance'la** sorunun hangi kolda olduğunu algıladıktan sonra denge sorununun hangi tipe benzediğini ayırt etmeye sıra geliyor. Örnek olarak; sağ ağaçta bulunan sağ sağ dengesizliği mi? yoksa sağ ağaçta bulunan sağ sol dengesizliği mi? Aynısını da sol için alırsak 4 farklı olasılığı değerlendiriyoruz. Sağ sağ dengesizliği veya sol sol dengesizliğini kolay bi şekilde verilerin yerlerini değiştirerek çözeriz. Sağ Sol veya Sol Sağ dengesizliğini ise önce sağ sağ & sol sol şekline getirip yine aynı metodu kullanarak dengeli hale getiririz. Yeni haliyle birlikte yükseklikleri de eşitlendikten sonra döngüyü dengesizlik nereden başladıysa o kısma düzenlenmiş **AvlChain'i** koyarak recursive olarak bir üst fonksiyona return ederek tamamladım.

Son olarak da dengeleme işlemi bittikten sonra **postOrderla** okuma işlemine sıra kalıyor bu kısımda ise burada da recursive bir yapı kullanarak ağacın en solundan başlayarak küçük çocuk büyük çocuk ve parent olarak okuyup bir üst parenta geri dönecek şekilde işlem yapıldı. Çıkartılan

değer ise her bir **AvlChain'de** bulunan **PointChain'in** içerisinde bulunan noktaların orjine olan uzaklığını her chaine göre küçükten büyüğe ekrana yazdırdım.