T.C.
SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

VERİ YAPILARI ÖDEV RAPORU

İKİLİ ARAMA AĞAÇLARI

G201210033 – Sena Nur Erdem

SAKARYA

Ağustos, 2023

Veri Yapıları Dersi

İKİLİ ARAMA AĞAÇLARI

Sena Nur/Erdem[G201210033_1A]*

[a] *Öğrenci numarası ve dersi aldığı grup*

Özet

Program çalıştığı gibi, 0-256 arası sayılarından oluşan ve sayıların arasını birer boşluğun ayırdığı Sayilar.txt dosyası okur. Her satır okunurken Yığıta sayılar eklenir. Dosyadan yeni okunan sayı, **çift ve o an işlem yapılan Yığıttan çıkmak üzere olan sayıdan büyük ise** yeni bir Yığıta eklenir. Aksi halde var olan Yığıta eklenmeye devam eder. Bu şekilde dosyadaki bir satır okunduktan sonra her Yığıt boşaltılıp ayrı bir ikili arama ağacına eklenir. Ağaçta aynı değer varsa o değer ikili arama ağacına eklenmez. Bir satırda bulunan yığıt sayısı kadar ikili arama ağacı oluşturulur. Her satır için, oluşan ikili arama ağaçlarından **en büyük yüksekliğe sahip ikili arama ağacı seçilir**. Eğer yükseklikler eşit olursa düğüm değerleri toplamı büyük olan ağaç seçilir. Toplam değerleri de eşit olan ağaçlardan önce oluşturulan ağaç seçilir. Seçilen maksimum yüksekliğe sahip ikili arama ağacı **postorder okunup** sayısal değerlerin ASCII karakter karşılıkları ekrana yazılıp **ekranda 10 milisaniye beklenip**, bir sonraki satır için yukarıdaki aynı işlemler uygulanıp tekrar karakterler ekrana yazılır. Dosya okuma bittiği zaman program da sonlanır.

Anahtar Kelimeler: ağaç, liste, yığıt

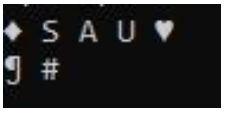
1. GELİŞTİRİLEN YAZILIM

Projemin modülerlliği için dosyalarımı aşağıdaki gibi oluşturdum.
Node: Bu dosyayı, her BST'yi bir LinkedList node'unda saklamak için kullandım. Her node, bir BST ve bir sonraki node'u işaret eden bir pointer içeriyor.
Stack: Bu dosya bir stack veri yapısını içeriyor. Stack, LIFO (Last In First Out) prensibine göre çalışır. Bu stacki, integer değerlerini saklamak için kullandım. Stack fonksiyonlarım şu şekilde; yeni bir eleman eklemek için **push()**, en üstteki elemanı çıkarmak için **pop()**, ve en üstteki elemanı kontrol etmek için **peek()**.
LinkedList: Bu dosyada, BST'leri saklamak için kullanılan bir LinkedList veri yapısı kullandım. Bu LinkedList, BST'leri postorder düzende yazdırmak, en yüksek ağacı bulmak ve tüm ağaçları silmek gibi işlemler yapıyor.
BST: Bu dosyada ikili arama ağacı yapısını kullandım. BST, bir ana node (kök) ve sol ve sağ alt dallardan oluşuyor. BST sınıfımda, veri eklemek için **insert()**, postorder düzende yazdırmak için **postorder()**, ağacın yüksekliğini bulmak için **height()** ve tüm node değerlerinin toplamını almak için **sumValues()** fonksiyonları yazdım.
Utilities: Bu dosyamı bazı yardımcı fonksiyonları yasmak için oluşturdum. Bir sayının karşılık geldiği ASCII karakterleri bulmak için. Bunun için de findCharacter() fonksiyonunu yazdım. Belirtmeliyim ki en çok takıldığım yer bu kısım oldu. Çünkü bu dönüşümü yapınca kullandığım Mac cihazda karakterleri ekran çıktımda göremedim. Bilgisayarımdaki sanal makinemde windows işletim sistemi üzerinden projemi devam ettirdim.

* Ödev Sorumlusu. Sorumlu ad soyad, öğrenci no,
Mail Adresi: sena.erdem1@ogr.sakarya.edu.tr

2. ÇIKTILAR

Sayilar.txt dosyası şu şekildeyken (dokumandaki örnek)
1 5 3 4 83 65 85 3 90 102
10 10 2 20 35
Ekran çıktısı yandaki gibidir.



Sayilar.txt dosyası için dokumanda verilen linkteki dosyayı denediğimde de doğru çıktıyı aldım. Çok uzun bir çıktı olduğu için sadece başından ve sonundan görüntüler ekliyorum. Çok büyük dosyalarda sorunsuz çalışmaktadır.

3. SONUÇ

İkili arama ağaçlarının , yığıtın nasıl çalıştığını daha iyi öğrenmemi sağlayan bir proje yaptım.

Referanslar

[1] https://www.youtube.com/watch?v=Xhsp5S3o-KM&list=PLh8R31K8J_9ClTt-87z7y2F-uzOrDgIof&index=26

[2] https://www.geeksforgeeks.org/stack-in-cpp-stl/

[3] https://www.youtube.com/watch?v=SGqmFMm9eEo&list=PLh8R31K8J_9ClTt-87z7y2F-uzOrDgIof&index=20