VERİ YAPILARI 2. ÖDEV RAPORU

Ödev Konusu:Avl ağaçları ile silme işlemi

Öğrenci Ad Soyad:Rüveyda Dilara Gülbaş

Öğrenci No:G221210033

Öğrenci Öğrenimi/Şube:İkinci Öğretim B grubu

Merhabalar,bu ödevde avl ağaçları ve onlara bağlı olup yapraklarını tutan yığıt yapıları oluşturdum.Bağlı liste yapısını kullanarak classlarla gerçekledim.Avl ağaçları ve ilgili yığıtları ise iki ayrı dizi içerisinde saklanmaktadır.Ayrıca yığıt sınıfının bir kopyası daha vardır.AvlNode,avltree,stack ve stacknode sınıflarında listeler ve düğümlerine ait fonksiyonlar ve tanımlamalar yer alır.

AVLNode Sınıfı:

Bu sınıf, AVL ağacındaki düğümleri temsil eder. Her düğüm, bir veri, sol ve sağ alt ağaçları için işaretçilere ve bir yükseklik değerine sahiptir.

StackNode Sınıfı:

Bu sınıf, yığıt düğümlerini temsil eder. Her düğüm, bir AVL düğümünü içerir ve bir sonraki düğüme işaret eden bir işaretçi içerir.

Stack Sınıfı:

Bu sınıf, bir yığıt veri yapısını temsil eder. AVL ağacındaki yaprakları ve diğer düğümleri işlemek için kullanılır. push, pop, isEmpty, clear, printStack gibi temel işlevlere sahiptir.

AVLTree Sınıfı:

Bu sınıf, AVL ağacını temsil eder. Her AVLTree nesnesi bir AVL ağacını ve o ağacın yapraklarını içeren bir yığıtı tutar. Ağaç oluşturma, düğüm ekleme, yükseklik güncelleme, denge faktörü hesaplama, döndürme işlemleri gibi temel AVL ağacı işlemlerini içerir.

AVLTree::insert(): AVL ağacına düğüm ekleme işlemi. Rotasyonları gerçekleştirebilecek şekilde AVL ağacını günceller.

AVLTree::rotateRight() ve AVLTree::rotateLeft(): Sağa ve sola dönüş işlemleri.

AVLTree::addLeavesToStack(): AVL ağacındaki yaprak düğümleri yığıta ekleyen işlev.

AVLTree::sumNonLeafNodes(): Yaprak olmayan düğümlerin toplamını bulan işlev.

AVLTree::printCharCodes(): Yaprak olmayan düğümlerin ASCII karakter kodunu hesaplayan ve yazdıran işlev.

Öncelikle veri.txt dosyasını satır satır okuyarak avl ağaçlarını her bir satır için oluşturur ve yapraklarını tespit ederek aynı indisteki dizideki yerine atar.Daha sonra yığıtlara eklenemeyen sayıları ayrı bir yerde tutarak bunların verdiğiniz sayısal değerdeki ascıı kodlarına dönüştürerek ekrana yazdırır.

findAndRemoveMinFromCopyStacks ve findAndRemoveMaxFromCopyStacks: Kopya yığıtlarındaki en küçük ve en büyük değerleri bulan ve çıkaran fonksiyonlar.

Bu fonksiyonlar öncelikle max fonksiyonunda max değer olarak intin min değerini ,min fonksiyonununda max olarka intin min değerini alır. yığıtın boş olup olmadığını kontrol eder.Eğer boş değilse içeri girer ve sürekli birbirleriyle karşılaştırarak min ve max değerini günceller.Tabi sırayla dolandığı için ilk olarak önce oluşan yığıta girerek bunu geri dönüş değeri olarak döndürür.

Main fonksiyonu içine gelince satır satır okuyarak avl ağaçlarına koyar.yaprak olan Avl Node değerinden düğümleri stack sınıfından oluşturduğum leafstack dizisine ekler.Yığıta eklenmeyen değerleri de verdiğiniz formülde yerine koyarak string dönüşümüyle ascıı kodunu ekrana yazar.İnt main içinde while agac sayısı 1 den büyükse devam eder.Agac sayısını satır satır okuyarak her satır sayısında 1 artırarak tutmuştum.İşlemler kopya yığıt üzerinden gerçekleşir. Daha sonra ilk kücük değeri bulur çıkarır,o yığıtın boş olup olmadığını kontrol eder. eğer değilse devam eder eğer o yığıt boşsa ilgili yığıtın ağacını siler. geri kalan ascıiları ekrana yazdırır ve sonrasında orijinal yığıt,kopya olan yığıta içeriği kopyalanır.Böylelikle ödevde yazdığınız gibi silinen yığıt hariç diğer yığıtlar geri eski haline dönmüş olur.

```
while(agacSayisi>1)
{
     a = findAndRemoveMinFromCopyStacks(copyStack, c);
     copyStack[a].pop();
  //   copyStack[a].printStack();
     if (copyStack[a].isEmpty())
     {
          leafStack[a].clear();
          avlTree[a].deleteTree();
          agacSayisi--;
          for (int i = 0; i < c; i++)
          {
               avlTree[i].printCharCodes();

          }
          cout << endl;
          system("cls");

          if (agacSayisi == 1)
          {

               for (int i = 0; i < c; i++)
               {
                    avlTree[i].printCharCodes();
                    if (avlTree[i].root != nullptr)
                    {
                         cout <<endl<< "AVL NO: " << i+1;
                    }
               }
               return -1;
          }
          else
          {
               for (int i = 0; i < c; i++)
               {
                    leafStack[i].copyNonEmptyElementsTo(copyStack[i]);

               }
          }
          continue;
     }
```

Aşağıdaki continue ifadesi sadece küçük değeri bulanda yazılıdır.Çünkü bir yığıt boşalıp başa döndüğümüzde tekrar küçük değerden başlamalıdır.system("cls")ler ile ekran temizlenir.Büyük bulma fonksiyonu da aynı böyle işler tek olmayan şey continue ifadesidir çünkü orada zaten küçükten devam edecektir.

Böylelikle bir ağaç kalana kadar kod devam eder.Ödevde en zorlandığım yer burasıydı.1 hafta bunu çözmek için uğraştım,teşekkür ederim.