VERİ YAPILARI DERSİ 2. ÖDEV RAPORU

Ödeve önce dosyadan verileri almakla başladım getline ile satırı aldım.

Daha sonra tek tek gezerek isdigit() fonksiyonu ile sayı mı diye sorguladım sayi ise += ile sayi stringine attım sayı değilse boşluktur . Boşluğa geldiğinde sayi stringini stoi ile inte çevirdim. Ve 3 boyutlu nokta dizisine atadım. Daha sonra nokta dizisinin içindeki değerleri kullanıp orjine olan uzaklığını buldum ve kuyruğa ekledim.

Toplam mesafeyi de main içerisinde buldum. İlk başta işlem yaptırmadım daha sonra iki nokta arası uzaklık formülünden mesafeyi de bulup kuyrukta oluşturduğum int mesafe değişkenine atadım.

Daha sonra kuyruğu oluşturduktan sonra da sırala fonksiyonu ile kuyruğu öncelikli sıraladım.

Siralama fonksiyonunu oluştururken 1. Sınıfta gördüğümüz sıralama algoritmasını kullandım.

```
void Kuyruk::sirala(){

    Dugum* enKucuk = ilk;
    Dugum* gec = ilk;
    Dugum* gec2 = ilk;

    for (int i = 0; i < veriSayisi; i++)
    {

        while (gec->sonraki != 0)
        {
            if (gec->sonraki->veri < enKucuk->veri)
            {
                enKucuk = gec->sonraki;

            }

            gec = gec->sonraki;
        }

        int deger = enKucuk->veri;

        enKucuk->veri = gec2->veri;
        gec2->veri = deger;


        gec2 = gec2->sonraki;
        gec = gec2;

        enKucuk = gec;
```

Kuyruğu sıralama işlemi bittikten sonra avlye attım. Avlye atma işleminde önce agacDugumu kuyruk tutacak şekilde ayarladım .

```hpp
#ifndef agacDugum_hpp
#define agacDugum_hpp

#include "Kuyruk.hpp"
class agacDugum
{
public:
    agacDugum(Kuyruk* veri);
    Kuyruk* veri;

    agacDugum* sol;
    agacDugum* sag;
};

#endif
```

Sonra eklerken karşılaştırma yapmak için de kuyruk->mesafeyi kullandım.

```cpp
agacDugum* AVLAgaci::ekle(Kuyruk* kuyruk,agacDugum* aktifDugum)
{
    if(aktifDugum==0)
        return new agacDugum(kuyruk);

    else if(aktifDugum->veri->mesafe<kuyruk->mesafe)
    {
        aktifDugum->sag=ekle(kuyruk,aktifDugum->sag);
        if(yukseklik(aktifDugum->sag)-yukseklik(aktifDugum->sol)>1)
        {
            if(kuyruk->mesafe>aktifDugum->sag->veri->mesafe)
                aktifDugum = solaDondur(aktifDugum);

            else
            {
                aktifDugum->sag = sagaDondur(aktifDugum->sag);
                aktifDugum =solaDondur(aktifDugum);
            }
        }
    }
    else if(aktifDugum->veri->mesafe>kuyruk->mesafe)
    {
        aktifDugum->sol = ekle(kuyruk,aktifDugum->sol);
        if(yukseklik(aktifDugum->sol)-yukseklik(aktifDugum->sag)>1)
        {

            if(kuyruk->mesafe<aktifDugum->sol->veri->mesafe)
                aktifDugum = sagaDondur(aktifDugum);
            else
```

Avl ağacını postOrder gezerken aktif düğümün tuttuğu kuyruk işaretçisine gittim ve kuyruktaki kuyrukyazdir() ile kuyruktaki noktaları ekrana çıkarttım.

```
void Kuyruk::kuyrukYazdir(){

    Dugum* gec= ilk;

    while(gec!=0)
    {
        cout<<gec->veri<<"  ";

        gec = gec->sonraki;
    }
   cout<<endl;


}
```

```
void AVLAgaci::postOrder(agacDugum* aktif)
{

    if(aktif)
    {

        postOrder(aktif->sol);

        postOrder(aktif->sag);


        aktif->veri->kuyrukYazdir();


    }
```

Avl düğümlerini silmek için düğümleri tek tek dolaşmamız lazım bunun için postOrder dolaşımı kullandım her düğümde önce kuyruğu sildim böylece kuyrukta kendi yok edicisi çağırıp kuyrukta düğümleri silmiş oldum sonrada agacDüğümünü sildim.

```
void AVLAgaci::postOrderSil(agacDugum* aktif)
{
    if(aktif)
    {

        postOrderSil(aktif->sol);
        postOrderSil(aktif->sag);
        delete aktif->veri;//kuyruk silindi
        delete aktif;   //agacDugumu silindi


    }
}
```

B191210021 - Eren Bekmezci

1.Öğretim C grubu - eren.bekmezci@sakarya.ogr.edu.tr