**YZM 2124**

Veri Yapıları

2021-2022 Bahar Dönemi

**ÖDEV RAPORU**

Teslim Tarihi: 16/05/2022

Öğrenci Adı: Kasım İslam TATLI

Öğrenci Numarası: 202802055

1. Giriş

Ödevin amacı ikili sıralama ağacı kurup DFS BFS işlemleri ile arama yapmak ve bunların program hızlarını karşılaştırmak.

Ödeve girdi olarak for döngüsü içinde veriler verilmiş olup işlemlerin ne kadar sürdüğünü chrono fonksiyonu ile bulunmuştur.

Girdi olarak gönderilen diziler aşağıda gösterilmektedir.

int dizi1[] = {1064,1065,1066,1067,1068,1069,1070,1071,1072,1073,1074,1075,1076,1077,1078,1079,1080,1081,1082,1083,2001,2002,2003,2004,2013,2014,2015,3001,3002,3003,3004,3005,3006,3007,3008,3009,3010,3011,3012,3013,3014,3015,3016,3017,3018,3019,3020,3021,3022,3023,3024,3025,3026,3027,3028,3029,3030,3031,3032,3033,3034,3035,3036,3037,3038,3039,3040,3041};    //agac1 için kullanılacak veriler

int dizi2[] = {3041,3040,3039,3038,3037,3036,3035,3034,3033,3032,3031,3030,3029,3028,3027,3026,3025,3024,3023,3022,3021,3020,3019,3018,3017,3016,3015,3014,3013,3012,3011,3010,3009,3008,3007,3006,3005,3004,3003,3002,3001,2015,2014,2013,2004,2003,2002,2001,1083,1082,1081,1080,1079,1078,1077,1076,1075,1074,1073,1072,1071,1070,1069,1068,1067,1066,1065,1064};    //agac2 için kullanılacak veriler

int dizi3[] = {3039,1074,3010,1082,3016,3005,3008,3014,3015,3023,3025,2002,3021,3035,1077,1065,1069,1073,3011,2004,3020,3001,3030,1076,1081,3031,3040,1071,3013,3029,3036,3027,3033,1070,3038,1068,1080,3037,2014,3003,3007,1072,1066,1083,2015,2003,3018,3024,3034,3009,3012,3019,3028,1078,3017,2001,1079,3041,1067,2013,3026,1064,3004,3022,3006,1075,3032,3002};    //agac3 için kullanılacak veriler

1. Geliştirme ve Çalıştırma Ortamları

Microsoft Visual Studio 2022 geliştirme ortamını kullandım.

Test için Dev-cpp kullanılmıştır.

1. Veri Yapıları ve Değişkenler

Ödevde ikili sıralama agaç veri yapısı kullanılmış olup aynı zamanda stack ve queue kütüphaneleri kullanılarak , stack(yığın) ve queue(kuyruk) veri yapılarından yararlanmıştır.

Programda kullanılan yapılar ve methodları aşağıda verilmiştir.

struct Dugum //Agacta yer alan her bir dugumun veri yapisi

{

int veri; //Dugumde tutulan veri

Dugum\* sag; //Dugumun sag cocuguna isaretci

Dugum\* sol; //Dugumun sol cocuguna isaretci

};

🡪ikili sıralama ağac veri yapısı ve methodları aşağıda verilmiştir.

struct IkiliSiralamaAgaci

{ //Ikili siralama agaci veri yapisi

Dugum\* kok; //Agac kokune isaretci

void agacKur(int\*); //Dizi isaretcisi alir ve dizinin elemanlarini ikili siralama agacina uygun sekilde yerlestirir

void agacKapat(); //Agacin dugumleri icin alinan bellek gozlerini program sonlanmadan hemen once iade eder

void olustur(); // Agac olusturur

int dugumSayisi; //dugum sayisini tutar

};

🡪DFS ve BFS methodların imzalari ve amaçları aşağıda verilmiştir.

bool DFS(Dugum\*, int); //Ilk derinlik arama fonksiyonu bir agac koku ve aranacak deger parametrelerini alir

bool BFS(Dugum\*, int); //Ilk genislik arama fonksiyonu bir agac koku ve aranacak deger parametrelerini alir

DFS() ne işe yarar ?

DFS() , ağaçlarda derine doğru aramaya yarayan bir algoritmadır.

BFS() ne işe yarar?

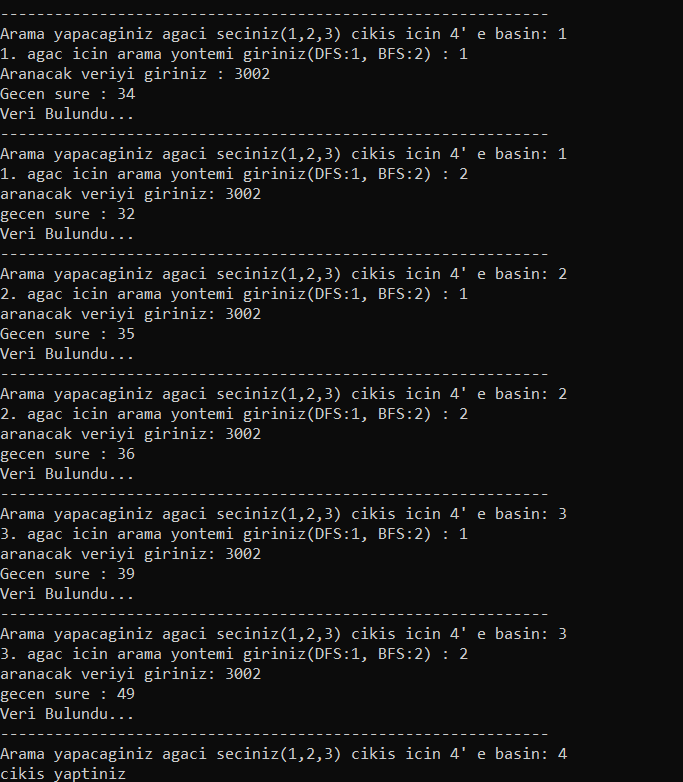
BFS() , ağaçlarda genişliği öncelik alarak soldan sağa arama yapmaya yarayan bir algoritmadır

* Bu algoritmalar projeye hız işlevsellik katar.

1. Program Akışı

Main methodunda for döngüsü ile ağaçKur() methoduna dizimizdeki veriler aktarılıp ağaçlar oluşturulur. Daha sonra bu ağaçlar içindeki veriler DFS() ve BFS() methodları ile aranır. Bu aramaların sonucu ve çıkarımlarım *sonuç* kısmında detaylıca yer almaktadır.

1. Sonuç



Resimde görüldüğü gibi Derinlik ve Genişlik öncelikli aramalara baktığımızda verilerin bulunduğu yere göre hızların değiştiğini fakat her halukarda oldukça verimli sonuçlar elde ettiğimizi görüyoruz.