RAPOR

DERSİN ADI : VERİ YAPILARI VE ALGORİTMALAR

ÖĞRENCİNİN ADI - SOYADI : YUNUS KARATEPE

ÖĞRENCİNİN NUMARASI : 17011051

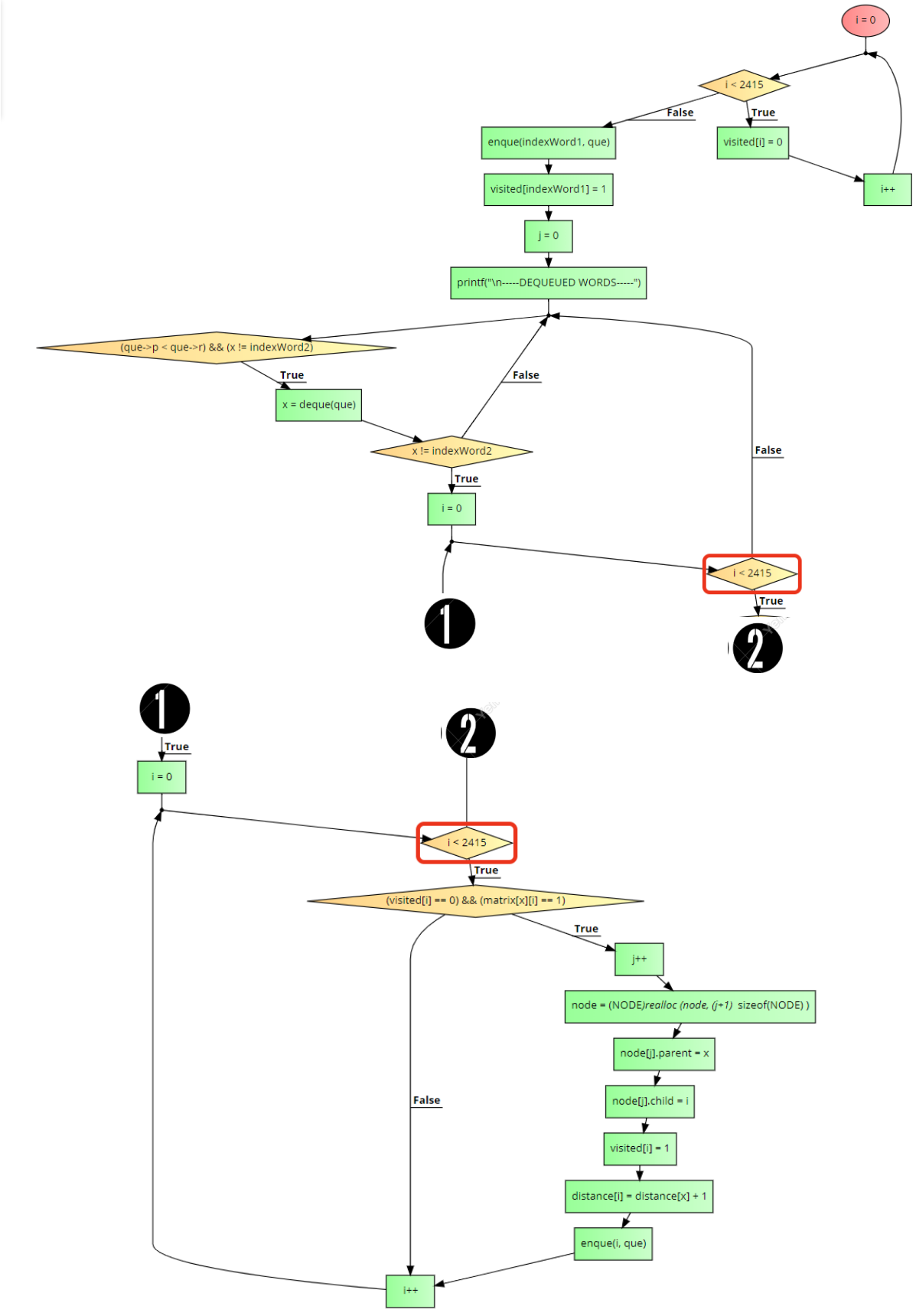
PROJE KONUSU : GRAF İŞLEMLERİ

1 – Yöntem

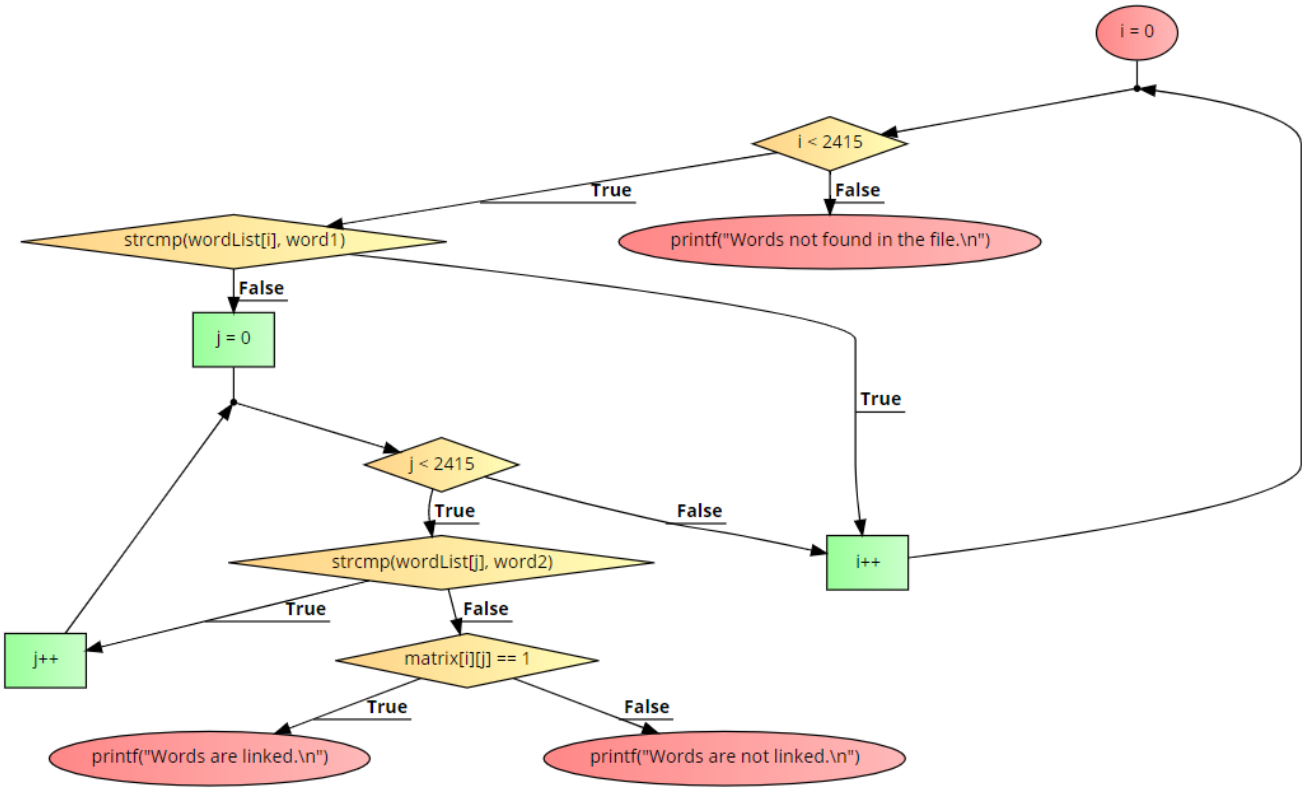
Projede bizden ilk olarak bir ".txt" dosyası içerisindeki kelimeleri okuyup bunlardan bir "adjacency matrix" oluşturmamız istenmiş. Okuma aşamasında ,"%s" ile bir döngü içerisinde, bütün kelimeler "word" isimli dizinin içerisine alınıyor. Daha sonra bu "word" deki tüm elemanları birbirleriyle karşılaştırıp 5 harfinden 4 tanesi aynı olan kelimelerin "adjacencyMatrix" de denk geldikleri komşuluk (bağlantı) bölgeleri 1 yapılmıştır. Bu aralarında bağlantı olduğu anlamına gelmektedir. Aksi takdirde ise 0 yapılmıştır. Böylece iki kelime arasında yalnızca 1 harf değiştirilerek diğerinin elde edilebileceğini 1 ler ile gösteren bir "adjacencyMatrix" oluşturulmuştur. Bağlantının olup olmadığını kontrol eden "isLinked" fonksiyonu da adjacencyMatrix[kelime1][kelime2] ya da adjacencyMatrix[kelime2][kelime1] ,ki ikisinde de aynı değer olmak zorundadır, noktalarına ulaşarak bu değer 1 ise bağlantının olduğunu, 0 ise bağlantının olmadığını bulmaktadır. Dönüşümleri bulan "breadthFirstSearch" fonksiyonu ise "Her kelimeden her adımda 1 harf değiştirilerek diğer kelime elde edilebilir mi?" sorusuna yanıt bulmaktadır. Daha sonrasında ise eğer böyle bir dönüşüm yapılabiliyorsa "displayPath" fonksiyonu vasıtası ile en kısa yolu ekrana yazdırmaktadır. Bu fonksiyon aynı zamanda "QUE" ismiyle tanımlanmış kuyruk veri yapısını kullanmaktadır. Her adımda kuyruktan en son çekilen ve eğer bu çekilen 2. Kelimeye eşit olmayan kelimenin bütün komşularını adjacencyMatrix yardımıyla bularak kuyruğa yerleştirmektedir. Bu işlem kuyruk bitene (ki kuyruğun bitmesi dönüşümün mümkün olmadığı anlamına gelmektedir) ya da kuyruktan çekilen kelime 2. Kelime olana (bu da dönüşüm mevcut, 2. Kelime elde edilebilir anlamına gelmektedir) kadar aynı işlemleri devam ettirir. Aynı zamanda bir kere kuyruğa atılan kelimenin bir daha kuyruğa girmesine gerek yoktur. Bunun için de "visited" isimli dizi veri yapısı kullanılmıştır. Bu kelimenin visited de denk geldiği indis 1 ise bu kelime daha önce kuyruğa girmiş demektir ve kuyruğa atılmaz. Kuyruğa atmak için "enque", çıkarmak için ise "deque" fonksiyonları yazılmıştır. Ayrıca kuyruğa atılan her kelimenin kim ile bağlantısı olduğu için kuyruğa atıldığının ("parent") tutulması gereklidir ki en kısa yolu yazdırırken bu "parent" ve "child" ilişkisini kullanarak işimizin kolaylaştırılması hedeflenmiştir. Bunun için ise ayrı bir "NODE" isimli struct yapısı kullanılmıştır. Menu switch-case işlemleri ile oluşturulmuştur.

Genel anlamda Breadth First Search algoritmasının anlaşılması açısından verimli bir proje olmuştur. Daha fazla açıklama source kodun içerisinde bulunabilir.

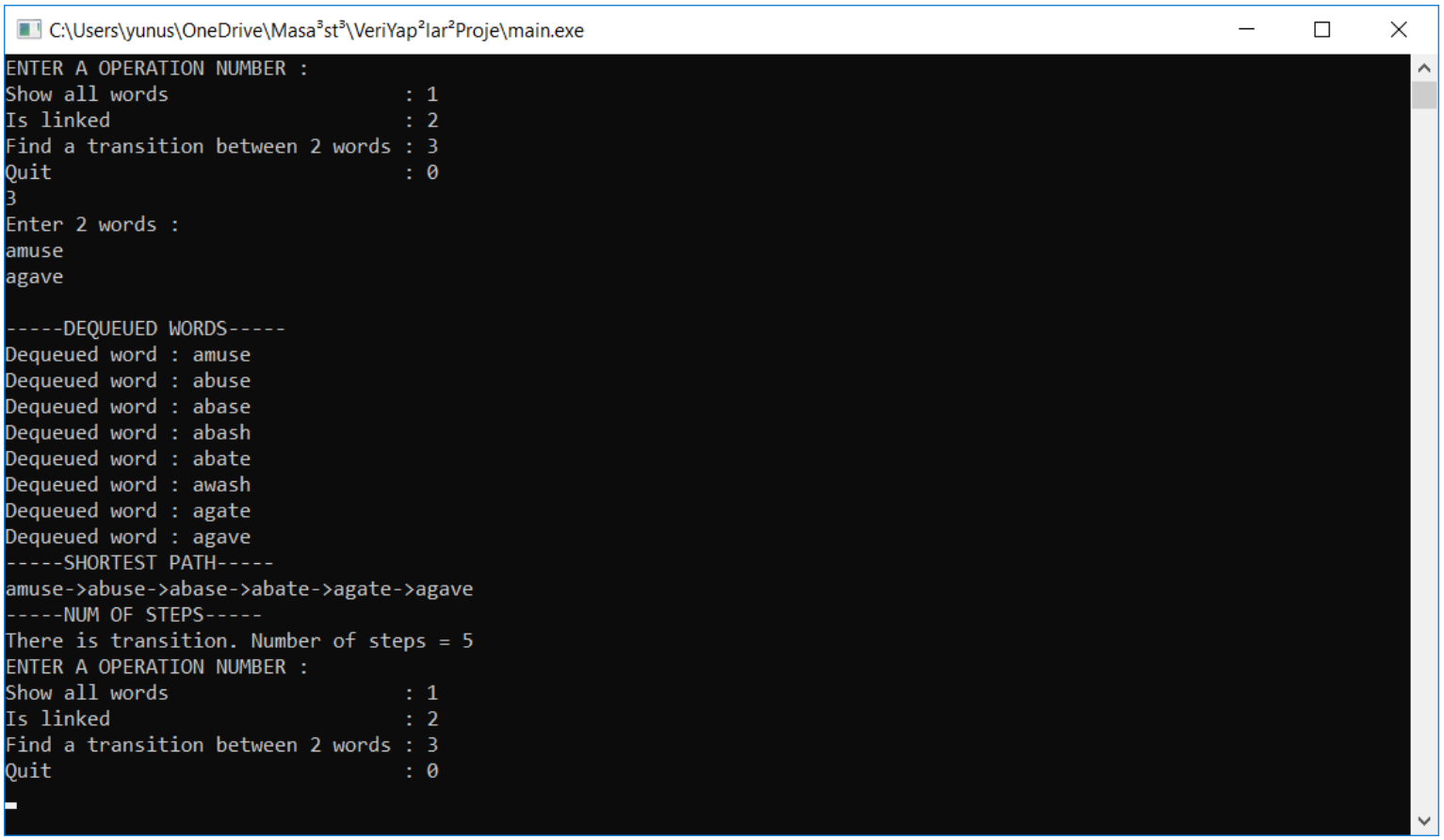
breadthFirstSearch() fonksiyonu algoritmasi



isLinked() fonksiyonu algoritması



2 – Uygulama



Adım 1 : “amuse” kelimesi kuyruğa atılır, daha sonra kuyruktan çekilir ve “amuse” ye komşu olan bütün kelimeler (yalnızca “abuse”) kuyruğa atılır.

Adım 2 : Kuyruktaki en üstteki kelime “abuse” kuyruktan çekilir ve komşusu olan “abase” kuyurğa atılır.

Adım 3 : “abase” çekilir ve komşusu olan “abash” ve “abate” kuyruğa atılır.

Adım 4 : “abash” ın komşusuz olan “awash” kuyruğa atılır.

Adım 5 : Kuyrukta en üstte “abate” vardır ve kuyruktan çekilir ve yerine “agate” atılır.

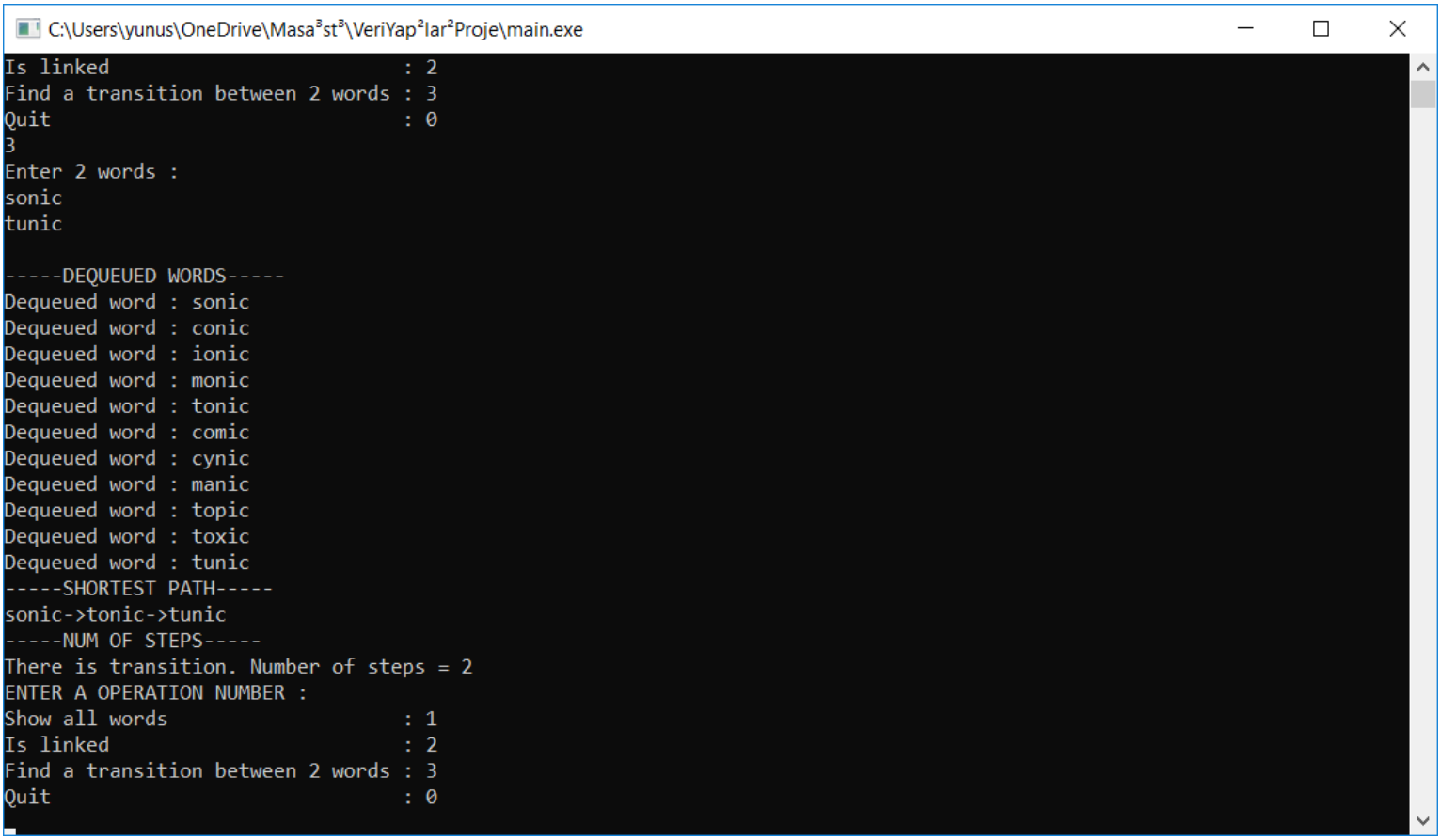
Adım 6 : “awash” ın daha önceden kuyruğa girmeyen herhangi bir komşusu yoktur sadece “awash” kuyruktan çekilir.

Adım 7 : “agate” kuyruktan çekilerek önceden kuyruğa girmemiş olan komşusu “agave” kuyruğa atılır.

Adım 8 : “agave” kuyruktan çekilir ve çekilen kelime aradığımız kelime olduğu için döngü sonlanır.

SHORTEST PATH : Path oluşturulurken ise bir parentin (o döngü anında deque edilmiş olan kelime) kuyruğa atılan tüm childları (komşuları) bir 2 li struct yapısında tutulduğu için en son kelimeye ulaşıldığında o kelimeden başlayarak sürekli parentine giderek, ilk kelimeye ulaşılabilir. Bu vasıta ile ekrana path yazdırılır.

Tamamen aynı mantık ile çalışan bir başka örnek.



- “runic” ve “study” kelimelerinin dönüşümünü arattığımızda

Uzun süren bir BST sonrasında dönüşümün olmadığı görülür.