28.05.2019

VERİ YAPILARI VE ALGORİTMALAR PROJE

Konu : Graf İşlemleri

Ad Soyad : Yusuf ANI

Öğrenci Numarası : 16011033

Grup : 1 (MEK)

İÇİNDEKİLER

1-YÖNTEM

1.1 - Problem

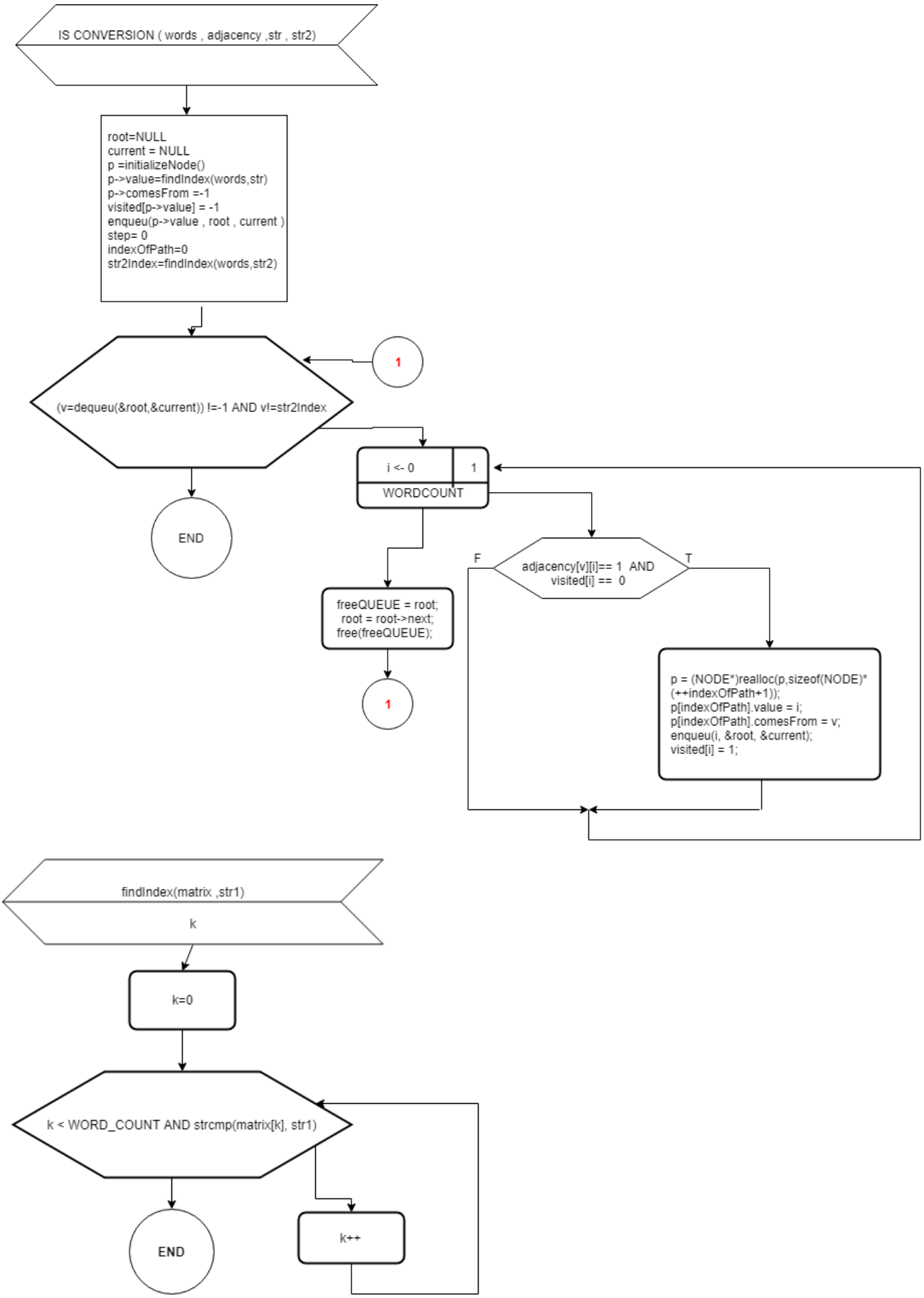
“Bu projede verilen iki kelime için, her adımda sadece 1 harfi değiştirerek 1. kelimenin, 2.kelimeye dönüşüp dönüşmediğini, dönüşüyorsa arada hangi kelimelerden geçildiğini bulan bir kelime oyunu yazılacaktır. Aşağıdaki örnek, prove kelimesinin guess kelimesine dönüşümünü göstermektedir.

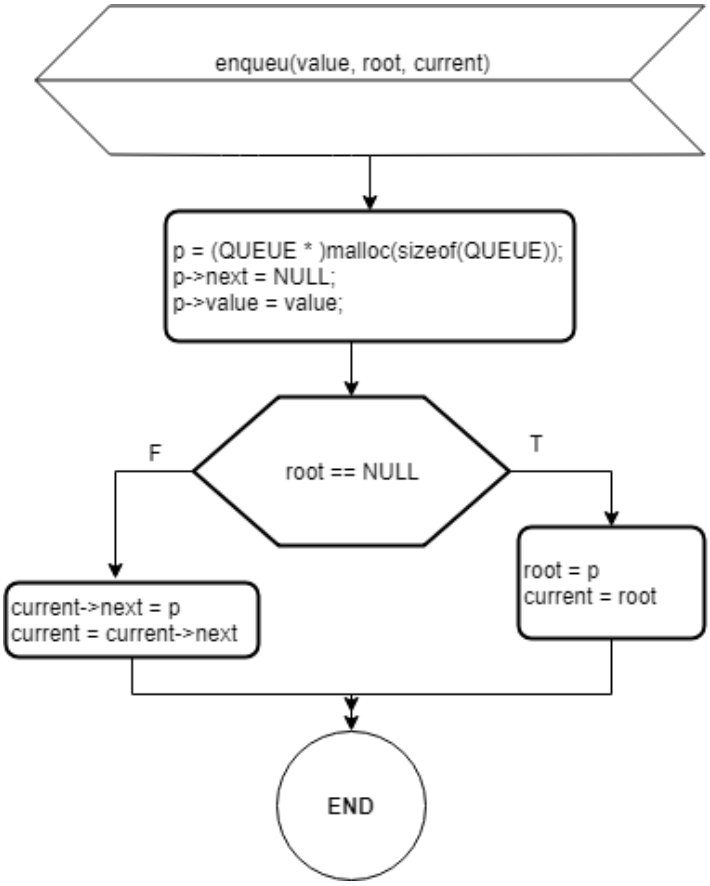
prove → prose → prese → prest → wrest → weest → geest → guest → guess

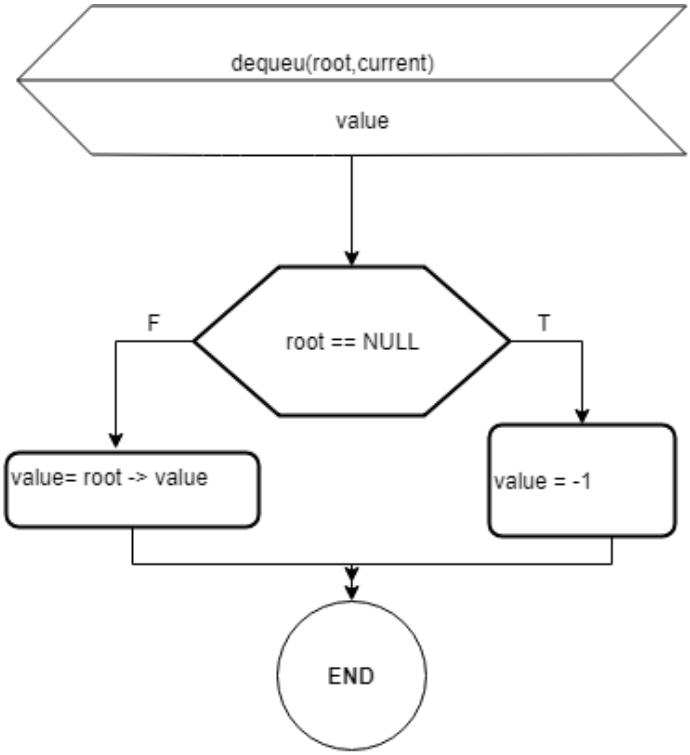
Verilen bir kelimeden, her adımda bir harf değiştirerek bir başka kelimeye ulaşmak için graf veri yapısı kullanılacaktır. Grafın düğümlerini kelimeler oluşturmalıdır. Eğer bir kelimenin sadece 1 harfini değiştirerek diğer kelime elde ediliyorsa iki kelime arasında bağlantı vardır. Örneğin yukarıdaki örnekte prove ve prose kelimeleri arasında bağlantı vardır. Fakat prove ve wrest kelimeleri arasında bağlantı yoktur. “

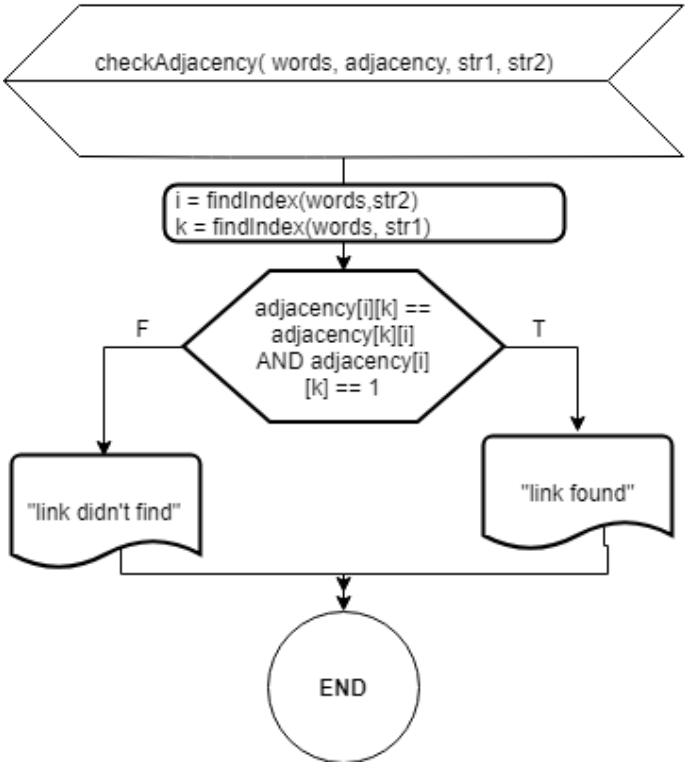
Probleme göre verilen kelimeden başka bir kelimeye dönüşüm olup olmadığı istenmektedir. Ben bunun için derste gördüğümüz **Depth-First Search** (DFS) algoritmasını kullandım. Kuyruk yapısı kullanarak daha önce ziyaret edilmemiş ve bağlantısı olan düğümün arkadaşları kuyruğa atılıp . Kuyruk boşalana kadar işlemi devam ettirdim.

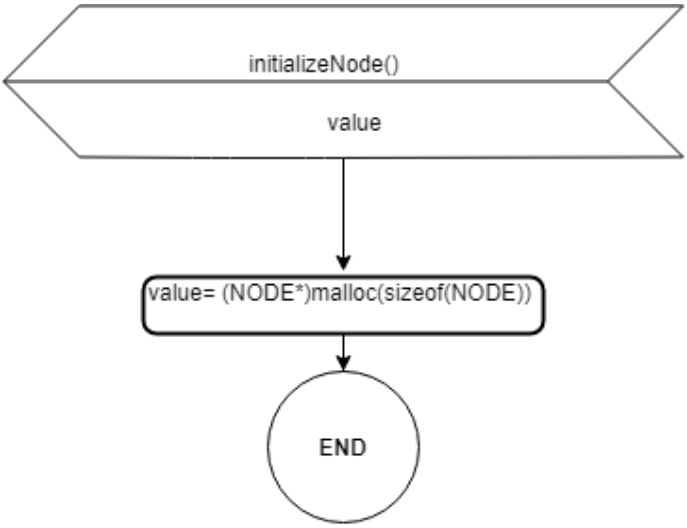
1.2-Akış Diyagramları




enqueu(value, root, current)
p = (QUEUE *)malloc(sizeof(QUEUE));
p->next = NULL;
p->value = value;
root == NULL
F
T
current->next = p
current = current->next
root = p
current = root
END


dequeu(root,current)
value
root == NULL
F
T
value= root -> value
value = -1
END

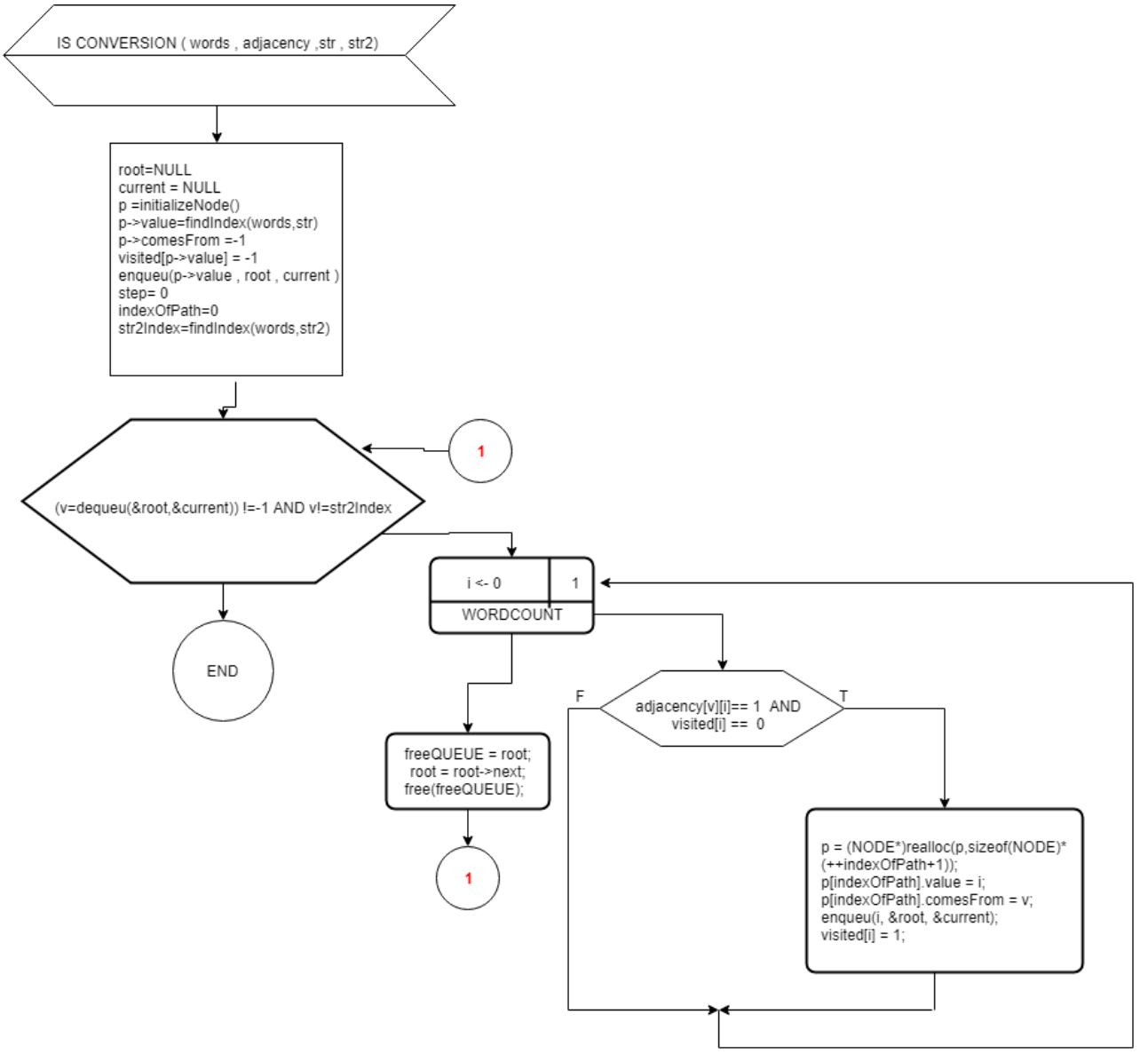

checkAdjacency(words, adjacency, str1, str2)
i = findIndex(words,str2)
k = findIndex(words, str1)
adjacency[i][k] == adjacency[k][i] AND adjacency[i] [k] == 1
F
T
"link didn't find"
"link found"
END


initializeNode()
value
value= (NODE*)malloc(sizeof(NODE))
END

2-UYGULAMA

2.1 – Arasında Bağlantı Olan Kelimeler

2.1.1 -ÖRNEK 1 : “abash” ile “abate” arasındaki bağlantı



Algoritmaya göre ilk olarak “**abash**” kelimesinin indexi olan sayi kuyruğa atılır ve visited olarak işaretlendirilir .

Fakat ben burada anlaşılırlığı arttırmak için kuyruğu gösterirken indexler yerine kelimeleri direkt göstermek istiyorum .

KUYRUK : abash

1.Adım

While döngüsü başlar . “**abash**” kelimesi kuyruktan **okunur** , “**abash**” kelimesiyle arasında bağlantı olan kelimeler arasından daha önce visited olarak işaretlenmeyen arkadaşları bulunup kuyruğa atılır . Bu kelimeler “**abase**” ve “**awash**” kelimeleri oluyor .

İşlem sonunda “**abash**” kelimesi kuyruktan **çekilir**.

2.Adım

KUYRUK : abase | awash

“**abase**” kelimesi kuyruktan **okunur** , “**abase**” kelimesiyle arasında bağlantı olan kelimeler arasından daha önce visited olarak işaretlenmeyen arkadaşları bulunup kuyruğa atılır . Bu kelimeler “**abate**” ve “**abuse**” kelimeleri oluyor .

İşlem sonunda “**abase**” kelimesi kuyruktan **çekilir**.

3.Adım

KUYRUK : awash | abate | abuse

“**awash** ” kelimesi kuyruktan **okunur** , “**awash**” kelimesiyle arasında bağlantı olan kelimeler arasından daha önce **visited** olarak işaretlenmeyen arkadaşları bulunup kuyruğa atılır . Hiç bağlantı ve **visited** olmayan arkadaşı bulunamadı .

İşlem sonunda “**abase**” kelimesi kuyruktan **çekilir**.

4.Adım

KUYRUK : abate | abuse

Bu sefer “**abate**” kelimesi çekilir ve işlem sonra erer.

```
1.Step, word= abash

2.Step, word= abase

3.Step, word= awash

Found Path in 3 steps . Printing Path
abash-> abase-> abate
```

2.1.1 -ÖRNEK 2 : "clout" ile "flour" arasındaki bağlantı

Algoritmaya göre ilk olarak "**clout**" kelimesinin indexi olan sayi kuyruğa atılır ve visited olarak işaretlendirilir .

Fakat ben burada anlaşılırlığı arttırmak için kuyruğu gösterirken indexler yerine kelimeleri direkt göstermek istiyorum .

KUYRUK : flour

1.Adım

While döngüsü başlar . "**clout**" kelimesi kuyruktan **okunur** , "**clout**" kelimesiyle arasında bağlantı olan kelimeler arasından daha önce visited olarak işaretlenmeyen arkadaşları bulunup kuyruğa atılır . Bu kelimeler "**cloud**" ve "**flout**" kelimeleri oluyor .

İşlem sonunda "**clout**" kelimesi kuyruktan **çekilir**.

2.Adım

KUYRUK : cloud flout

"**cloud**" kelimesi kuyruktan **okunur** , "**cloud**" kelimesiyle arasında bağlantı olan kelimeler arasından daha önce visited olarak işaretlenmeyen arkadaşları bulunup kuyruğa atılır . Bu kelime "**aloud**" oluyor

İşlem sonunda "**cloud**" kelimesi kuyruktan **çekilir**.

3.Adım

KUYRUK : flout aloud

"**flout** " kelimesi kuyruktan **okunur** , "**flout**" kelimesiyle arasında bağlantı olan kelimeler arasından daha önce visited olarak işaretlenmeyen arkadaşları bulunup kuyruğa atılır . Bu kelimeler "**float**" ve "**flour**" kelimeleri oluyor .

İşlem sonunda "**flout**" kelimesi kuyruktan **çekilir**.

4.Adım

KUYRUK : aloud | float | flour

"**aloud** " kelimesi kuyruktan **okunur** , "**aloud**" kelimesiyle arasında bağlantı olan kelimeler arasından daha önce visited olarak işaretlenmeyen arkadaşları bulunup kuyruğa atılır . Hiç bağlantı ve visited olmayan arkadaşı bulunamadı .

İşlem sonunda "**aloud**" kelimesi kuyruktan **çekilir**.

5.Adım

KUYRUK : float | flour

"**float** " kelimesi kuyruktan **okunur** , "**fload**" kelimesiyle arasında bağlantı olan kelimeler arasından daha önce visited olarak işaretlenmeyen arkadaşları bulunup kuyruğa atılır . . Bu kelimeler "**bloat**" ve "**gloat**" kelimeleri oluyor .

İşlem sonunda "**float**" kelimesi kuyruktan **çekilir**.

6.Adım

KUYRUK : flour | bloat | gloat

"**flour**" kelimesi okunur ve işlem sona erer.

```
1.Step, word= clout

2.Step, word= cloud

3.Step, word= flout

4.Step, word= aloud

5.Step, word= float

Found Path in 5 steps . Printing Path
clout-> flout-> flour
```

2.2 – Arasında Bağlantı Olmayan Kelime

Algoritmaya göre ilk olarak "**chair**" kelimesinin indexi olan sayi kuyruğa atılır ve visited olarak işaretlendirilir .

Fakat ben burada anlaşılırlığı arttırmak için kuyruğu gösterirken indexler yerine kelimeleri direkt göstermek istiyorum .

KUYRUK : chair

1.Adım

While döngüsü başlar . “**chair**” kelimesi kuyruktan **okunur** , “**chair**” kelimesiyle arasında bağlantı olan kelimeler arasından daha önce visited olarak işaretlenmeyen arkadaşları bulunup kuyruğa atılır . Bu kelimeler “**chain**” ve “**choir**” kelimeleri oluyor .

İşlem sonunda “abash” kelimesi kuyruktan **çekilir**.

2.Adım

KUYRUK : chain choir

“**chain**” kelimesi kuyruktan **okunur** , “**chain**” kelimesiyle arasında bağlantı olan kelimeler arasından daha önce visited olarak işaretlenmeyen arkadaşları bulunup kuyruğa atılır . Hiç bağlantı ve visited olmayan arkadaşı bulunamadı .

İşlem sonunda “**chain**” kelimesi kuyruktan **çekilir**.

3.Adım

KUYRUK : choir

“**choir** ” kelimesi kuyruktan **okunur** , “**choir**” kelimesiyle arasında bağlantı olan kelimeler arasından daha önce visited olarak işaretlenmeyen arkadaşları bulunup kuyruğa atılır . Hiç bağlantı ve visited olmayan arkadaşı bulunamadı .

4. Adım

Bu durumda Kuyruk boş kalmıştır ve bu yüzden yol bulunamamıştır.

```
1.Step, word= chair
2.Step, word= chain
3.Step, word= choir
Didn't find path
```