

YILDIZ TEKNİK ÜNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ



Öğrenci Numarası: 20011093

Ad – Soyad: Müdafer Kaymak

Öğrenci E-Postası: mudafer.kaymak@std.yildiz.edu.tr

BLM-1012—YAPISAL PROGRAMLAMAYA GİRİŞ FİNAL PROJESİ

BOYER-MOORE-HORSPPOOL ALGORİTMASI

Ders Yürütücüsü

Doç.Dr.Mehmet Fatih AMASYALI

Haziran,2022

İÇERİK

- Boyer Moore Algoritması Nedir? Ne işe yarar? Nasıl Çalışır?
- Analizi
- Kullanım alanları
- Avantaj-Dezavantajları ve sınırları
- Karmaşıklığı
- Zaman karmaşıklığının analizi
- Rakipleri
- C dilindeki kodu
- Ekran Çıktıları
- Kaynaklar

VIDEO ADRESİ

<https://youtu.be/M5leuya9NZg>

Boyer Moore Horspool Algoritması:

- Boyer Moore Horspool algoritması, bir karakter metninin içinde bir alt dizi (istenilen daha kısa bir metni) arama algoritmasıdır.
- Algoritma diğer arama algoritmalarının aksine daha hızlı çalışır bunun nedeni verimli kaydırma işlemi yapmasıdır.

Boyer Moore Horspool Algoritması nasıl çalışır:

- Öncelikle aranan kelimenin (letter) her karakterinin değerini 'Bad Match Table' yardımıyla bulunur.

Bad Match Table Oluşturma Prensipleri:

- Her harfin değeri, kelimenin uzunluğu ile indeksinin (aşağıdaki tabloda indeks numaraları yazılmıştır) farkının 1 eksikidir.
- Değer = Kelimenin uzunluğu-harfin indeksi-1
- Eğer kelimenin son harfine daha önce bir değer atanmamışsa o harfin değeri kelimenin uzunluğuna eşit olur
- Kelimede bulunmayan harflerin değeri kelimenin uzunluğuna eşittir.

0 1 2 3
a b c d

| Letter | a | b | c | d | * |
|--------|---|---|---|---|---|
| Value | 3 | 2 | 1 | 4 | 4 |

$$\text{Value (a)} = 4 - 0 - 1 = 3$$

$$\text{Value (b)} = 4 - 1 - 1 = 2$$

$$\text{Value (c)} = 4 - 2 - 1 = 1$$

$$\text{Value (d)} = 4$$

Şekil 1- 'abcd' için örnek Bad Match Table

- Ardından kelimenin son harfinin indeksinden başlayarak arama algoritması başlar.

Arama ve Kaydırma Prensipleri:

- Eğer harfler eşleşirse bir önceki indekste bulunan harfler incelenir.
 - Eğer eşleşme olmazsa metin üzerinde hangi noktadan karşılaştırmaya başlanmışsa o noktadaki karakterin değeri kadar kaydırma yapılır.
- Bu adımları eşleşme (varsa) bulunana kadar yapılır.

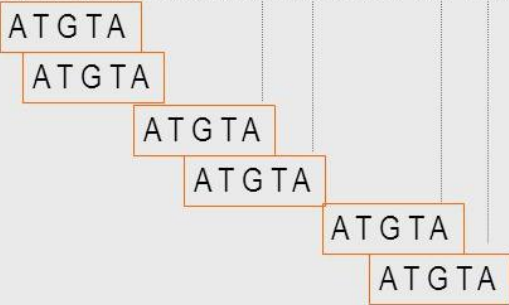
ANALİZ:

Given the pattern ATGTA

• The shift table is:

| | |
|---|---|
| A | 4 |
| C | 5 |
| G | 2 |
| T | 1 |

• The searching phase: G T A C T A G A G G A C G T A T G T A C T G ...



Şekil 2

- 'ATGTA' kelimesinin Bad Match Tablosunu oluşturuyoruz.
- Görüldüğü üzere ilk karşılaştırmada 'T' ve 'A' harfi uyuşmadı. Arama aralığını 'T'nin değeri (1) kadar kaydırıldı.
- İkinci karşılaştırmada 'T' ve 'A' uyuştü fakat 'G' ve 'C' uyuşmadı. Arama aralığını karşılaştırmaya başladığımız 'A' nın değeri (4) kadar kaydırıyoruz.
- Üçüncü karşılaştırmada 'G' ve 'A' uyuşmadı. Arama aralığını karşılaştırmaya başladığımız 'G'nin değeri (2) kadar kaydırıyoruz.
- Dördüncü karşılaştırmada 'C' ve 'A' uyuşmadı. Arama aralığını karşılaştırmaya başladığımız 'C'nin değeri (5) kadar kaydırıyoruz.
- Beşinci karşılaştırmada 'G' ve 'A' uyuşmadı. Arama aralığını karşılaştırmaya başladığımız 'G'nin değeri (2) kadar kaydırıyoruz.
- Altıncı karşılaştırmada tüm karakterler uyuştü, amacımıza ulaştık.

Kullanım Alanları:

- İçerisinde karakter dizileri olan listelerde, web sitelerinde, otomatik düzeltmelerde, kütüphanelerde başta olmak üzere karakterlerin olduğu çoğu yerde bu algoritma kullanılır.

Avantajları, Dezavantajları ve Sınırları:

- Uygulaması çok kolaydır.
- Küçük alfabeler için etkilidir.
- Bütün metni aramasına gerek yoktur.
- Girilen metin ve kelimeye bağlı olarak kaydırma miktarları değişeceğinden bazı durumlarda verimliliği çok düşer.
- Algoritmanın en kötü senaryosu kaydırma miktarının düşük olması ve kelime ile metnin aradığımız yerinin sürekli eşleşme yaşamasıdır. Eğer aradığımız yerin en son karakteri kelimenin içinde de bulunuyorsa 'Bad Match Table'dan gelecek değeri düşük olacaktır ve bu kaydırma miktarının düşük olmasına sebep olacaktır.
- Boyer-Moore-Horspool'un en kötü senaryosu ayrıca şekil-10'da bulunan zaman karşılaştırmasında gösterilen 'Naive' arama algoritmasının da en kötü senaryosudur ancak daha optimizedir ve bellek açısından daha verimlidir.

Zaman karmaşıklığı:

- $n \rightarrow$ Metnin uzunluğu
- $m \rightarrow$ Kelimenin uzunluğu
- Bu durumda en kötü ihtimalde zaman karmaşıklığı $\rightarrow O(mn)$
- Bu durumda en iyi ihtimalde zaman karmaşıklığı $\rightarrow O(n)$

Yer karmaşıklığı:

- $K \rightarrow$ Alfabenin uzunluğu
- Yer Karmaşıklığı $\rightarrow O(k)$

Zaman karmaşıklığı Analizi:

➤ Aynı metnin üzerinde farklı kelimeler aranması:

```
Arama yapılacak metni giriniz: Yildiz teknik universitesi bahar festivaline manga geliyor
Aradiginiz kelimeyi giriniz: manga
Kelime 45 ve 51 indeksleri arasinda bulunuyor
Zamana Bagli Yildiz Diyagrami: *****
-
```

Şekil 3

```
Arama yapılacak metni giriniz: Yildiz teknik universitesi bahar festivaline manga geliyor
Aradiginiz kelimeyi giriniz: bahar
Kelime 27 ve 33 indeksleri arasinda bulunuyor
Zamana Bagli Yildiz Diyagrami: *****
-
```

Şekil 4

- Şekil 3 ve şekil 4’te görüldüğü üzere arama yapılan metinler aynı ve aranan kelimelerin uzunluğu eşit. Kelimelerin yeri farklı olsa da uzunlukları eşit olduğundan ve metinde ikisinin de eşleşme sayısı benzer olduğundan algoritmaların çalışma süreleri arasında sadece bir yıldız oranında fark var. Bunun nedeni algoritmanın arama sırasında ‘niver’ ve ‘bahar’ karşılaştırması yaparken ‘r’lerin eşleşmesini görmesi ve bir önceki harfleri kontrol etmesi.

➤ Daha uzun metin üzerinde aynı kelimelerin aranması:

```
Arama yapılacak metni giriniz: Yildiz teknik universitesi bahar festivaline manga pinhani ve yüksek sadakat geliyor
Aradiginiz kelimeyi giriniz: manga
Kelime 45 ve 51 indeksleri arasinda bulunuyor
Zamana Bagli Yildiz Diyagrami: *****
-
```

Şekil 5

- Şekil 5’te arama yapılan metnin uzunluğunu arttırıldı ama aranan kelime aynı. Aranan yer arttığından dolayı şekil 3 ile karşılaştırsak çalışma süresinde belirgin bir artış var.

➤ Aynı uzunlukta fakat farklı metinler üzerinde aynı kelimelerin aranması:

```
Arama yapılacak metni giriniz: Yildiz teknik universitesi bahar festivaline manga mangage
Aradiginiz kelimeyi giriniz: manga
Kelime 45 ve 51 indeksleri arasinda bulunuyor
Kelime 51 ve 57 indeksleri arasinda bulunuyor
Zamana Bagli Yildiz Diyagrami: *****
```

Şekil 6

- Şekil 6’da metnin uzunluğu şekil 3 ile aynı fakat metin farklı ve aranan kelime tekrarlanıyor. Aranan kelimenin eşleşmeleri kontrol edildiği için çalışma süresinde artış gözlemleniyor.

```
Arama yapılacak metni giriniz: banga canga danga langa kanga manga ranga panga hanga anga
Aradiginiz kelimeyi giriniz: manga
Kelime 30 ve 36 indeksleri arasinda bulunuyor
Zamana Bagli Yildiz Diyagrami: *****
```

Şekil 7

- Şekil 7’de metnin uzunluğu şekil 3 ile aynı fakat burada metnin karakterleri ile aranan kelime arasında bolca benzerlik var bundan dolayı eşleşmeleri kontrol etmek ekstra zaman alıyor.

➤ Metin ve kelime uzunluklarına oranla zaman karmaşıklığı:

```
Arama yapılacak metni giriniz: Yapisal programlamaya giris dersi icin sectigimiz algoritmanin raporunu yaziyoruz
Aradiginiz kelimeyi giriniz: rapor
Kelime 63 ve 69 indeksleri arasinda bulunuyor
Zamana Bagli Yildiz Diyagrami: *****
```

Şekil 8

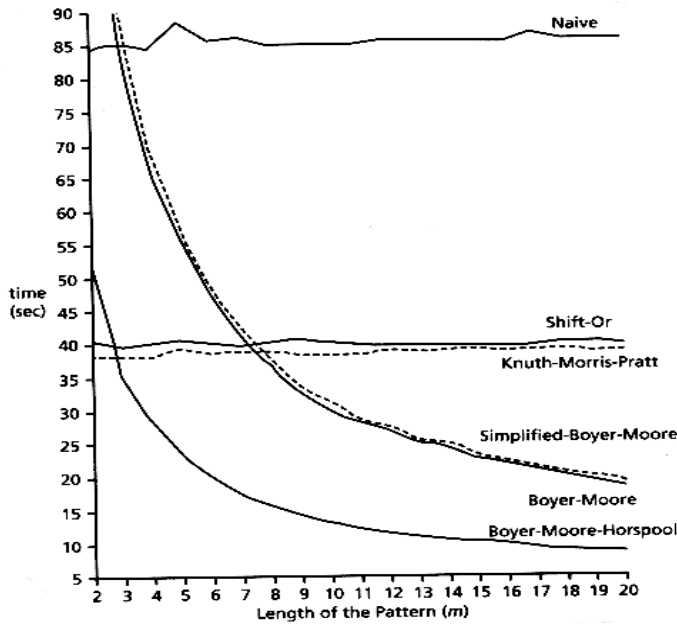
```
Arama yapılacak metni giriniz: Yapisal icin sen gerek yapman rapor
Aradiginiz kelimeyi giriniz: sen
Kelime 13 ve 17 indeksleri arasinda bulunuyor
Zamana Bagli Yildiz Diyagrami: *****
```

Şekil 9

- Şekil 8 ve Şekil 9’da farklı uzunlukta metinler ve farklı kelimeler üzerinde arama yapıldığında daha uzun kelime ve metinde çalışma süresinin daha fazla olduğu görülüyor.

Boyer-Moore-Horspool algoritması rakibi Boyer-Moore algoritması:

- Boyer-Moore hem 'Good Suffix' hem 'Bad Character Shift' kullanır. Her adımda iki yöneme göre önerilen kaydırma miktarı alınır, hangisi daha fazlaysa o uygulanır. Boyer-Moore-Horspool ise sadece 'Bad Character Shift' kullanır.
- Boyer-Moore uzun ve doğal metinler için muhtemelen en etkili algoritma iken. Boyer-Moore-Horspool küçük alfabelerde daha etkilidir.
- Boyer-Moore uygulanması daha zor bir algoritmadır.
- En iyi ihtimalle zaman karmaşıklığında Boyer-Moore $O(n/m)$ iken Boyer-Moore-Horspool $O(n)$ 'dir. Boyer-Moore-Horspool'un en kötü senaryosu Boyer-Moore'dan çok daha kötüdür ancak normal kullanımlarda bu senaryoya ulaşmak çok zordur. ($n \rightarrow$ Kelime uzunluğu, $m \rightarrow$ Metin uzunluğu)



Şekil 10-Bazı arama algoritmalarının performans karşılaştırmaları

C Kodu:

```
1 #include <limits.h>
2 #include <string.h>
3 #include <stdio.h>
4 #include <stdlib.h>
5
6 #define CHARS 256
7
8
9 // bad character tablosunu oluşturmak için kullanılacak fonksiyon
10 void badCharacterTable( char *str, int size, int badchar[CHARS])
11 {
12     int i;
13
14     // kelimedeki olmayan harflerin değeri kelimenin uzunluğu kadar olduğu için bütün diziye uzunluğu tanımlıyorum
15     for (i = 0; i < CHARS; i++)
16         badchar[i] = size;
17
18     // Bad match tablosu oluşturuyorum
19     for (i = 0; i < size-1; i++)
20         badchar[(int) str[i]] = size-i-1;
21 }
22
```

'Bad Match Table' oluşturma fonksiyonu

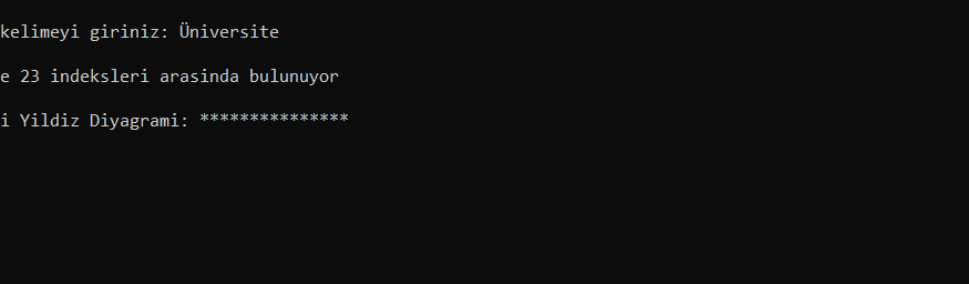
```
23 // bad match tablosu yardımıyla arama yapacak olan fonksiyon
24 int search( char *txt, char *let)
25 {
26     int counter=0, badchar[CHARS];
27     int m = strlen(let);
28     int n = strlen(txt);
29
30     //bad match table
31     badCharacterTable(let, m, badchar);
32     // s kaydırma işlemi yaparken yardımcı olacak değer
33     int s = 0;
34     while(s <= (n - m))
35     {
36         counter++;
37         int j = m-1;
38         // j değerini eşleşme olduğu sürece azaltıyorum
39         while(j >= 0 && let[j] == txt[s+j]){
40             counter++;
41             j--;
42         }
43
44         //Eğer aranan kelime mevcut aranan yerde ise yukarıdaki while döngüsünden j indeksi -1 olacak çıkacaktır.
45         if (j < 0)
46         {
47             printf("\nKelime %d ve %d indeksleri arasında bulunuyor", s,s+m-1);
48
49             //Aranan kelimenin metin içinde tekrarlanma ihtimalini düşürerek arama alanını kelime uzunluğu kadar kaydırıyorum
50             s += m;
51         }
52         else
53         {
54             //Eşleşme sağlanmadı. Arama alanının en sağındaki karakterin 'Bad Character' tablosundaki değeri kadar kaydırma yapıyorum
55             s += badchar[txt[s+m-1]];
56         }
57     }
58     return counter;
59 }
```

Boyer-Moore-Horspool Algoritmasının uygulanması (51. Satırda 's+=m' yazıyor)

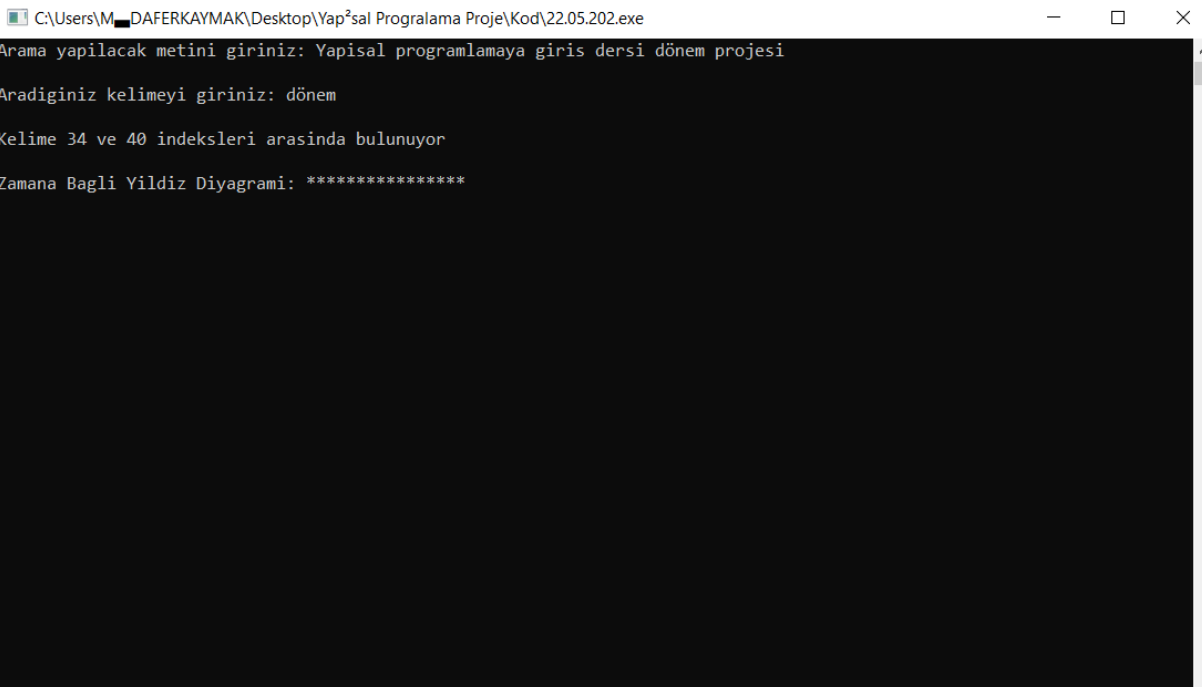
```
64 //main fonksiyon
65 int main()
66 {
67     int j,counter;
68     char txt[CHARS],let[CHARS];
69
70
71     printf("Arama yapılacak metni giriniz:");
72     gets(txt);
73     printf("\nAradığınız kelimeyi giriniz:");
74     gets(let);
75     counter = search(txt, let);
76
77     //zaman karmaşıklığı analizi yapmak için döngülere counter atadım
78     printf("\nZamana Bağlı Yıldız Diyagramı: ");
79     for(j=0; j<counter; j++){
80         printf("*");
81     }
82     printf("\n");
83
84     getchar();
85
86     return 0;
87 }
```

Main fonksiyon ve yıldız diyagramı

Örnek çıktı:



```
C:\Users\M_...DAFERKAYMAK\Desktop\Yapısal Programlama Proje\Kod\22.05.202.exe
Arama yapılacak metni giriniz: YildizTeknikUniversitesiBilgisayarMühendisligi
Aradığınız kelimeyi giriniz: Üniversite
Kelime 12 ve 23 indeksleri arasında bulunuyor
Zamana Bağlı Yıldız Diyagramı: *****
```



```
C:\Users\M...DAFERKAYMAK\Desktop\Yapısal Programlama Proje\Kod\22.05.202.exe
Arama yapılacak metni giriniz: Yapisal programlamaya giris dersi dönem projesi
Aradiginiz kelimeyi giriniz: dönem
Kelime 34 ve 40 indeksleri arasında bulunuyor
Zamana Bagli Yildiz Diyagrami: *****
```

Kaynaklar:

- <https://www.geeksforgeeks.org/boyer-moore-algorithm-for-pattern-searching/>
- <https://bilgisayarkavramlari.com/2012/01/16/horspool-algoritmasi/>
- <https://www.encora.com/insights/the-boyer-moore-horspool-algorithm>
- <http://www.mathcs.emory.edu/~cheung/Courses/253/Syllabus/Text/Matching-Boyer-Moore2.html>
- <https://rajeevkuruganti.com/2021/boyer-moore-horspool-algorithm/>
- <http://www-igm.univ-mlv.fr/~lecroq/string/node18.html>
- [https://en.wikipedia.org/wiki/Boyer%E2%80%93Moore_string-search_algorithm](https://en.wikipedia.org/wiki/Boyer%E2%80%93Moore_string_search_algorithm)
- https://en.wikipedia.org/wiki/Boyer%E2%80%93Moore%E2%80%93Horspool_algorithm#Performance
- <https://www.inf.hs-flensburg.de/lang/algorithmen/pattern/horsen.htm>