<https://www.youtube.com/watch?v=pIaT9QUKc1A>  **//video linki**

**Knuth-Morris-Prat (KMP) algoritması** bir kelimenin (yada bir metin parçasının) bir metin içerisinde aranmasını sağlayan algoritmadır. Basitçe bu algoritmada bir kelimenin aranan metinde bakılması ve bakıldığı yerde bulunamaması durumunda nerede olabileceği ile ilgili bir bilginin elde edilmesi hedeflenir.

Klasik bir metin arama işleminde aranan kelime metindeki bütün ihtimallerde denenir. (örneğin doğrusal arama (linear search) bu şekilde çalışır).  Bu algoritma ise aranan kelimenin, aranan metinde bulunmaması durumunda, kelimenin içerisindeki harflerden yola çıkarak ,oluşturulan prefix tablosuyla birden fazla ihtimali elemektedir.

Örneğin “aka” kelimesini ararken, bir yol bulduğunda, bundan sonra “k” var mı diye bakar. Çünkü “a” ‘ yı zaten bulmuştur. Bu şekilde uzun bir metinde küçük bir metni arar.

Bu bir metin arama algoritmasıdır ve metin arama işlemlerinde kullanılır.

Zaman ve alan karmaşıklığında ön işlem aşaması: O(m)

Arama zaman karmaşıklığı: O(n+m)

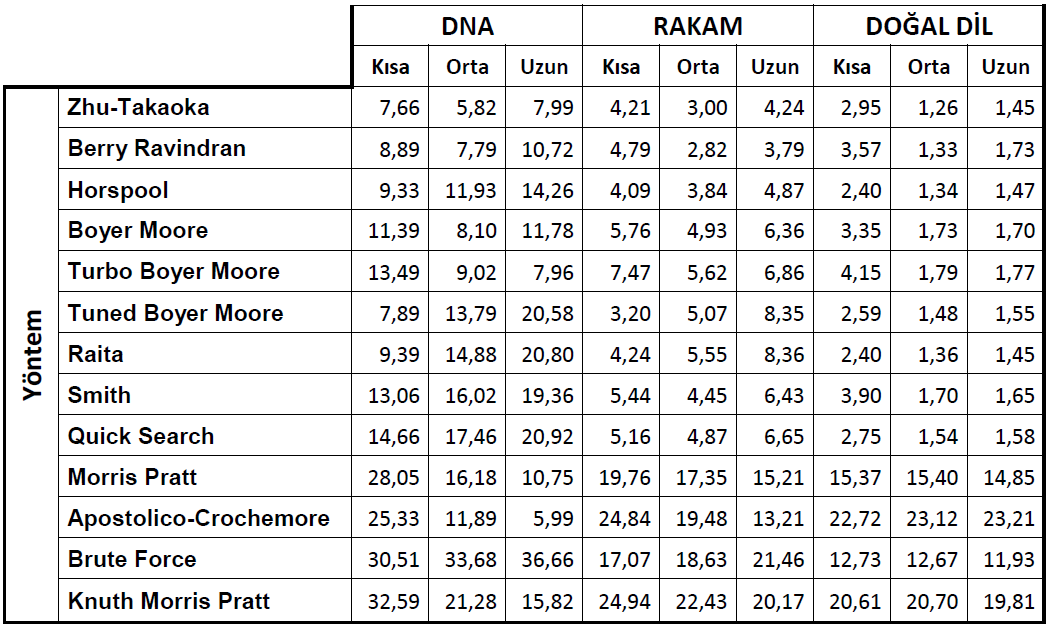
Kısıtları: Büyük metinin uzunluğu çok fazla olduğu durumlarda tercih edilmez.

Rakipleri:

Horspool,Raita,Zhu-Takaoka,Tuned Boyer-Moore,Quick Search, Berry-Ravindran,Boyer-Moore,Smith,Turbo Boyer-Moore,Brute Force, Morris-Pratt, Apostolico Crochemore.

Avantaj-Dezavantajları;

Doğal diller için, Apostolico’ya göre daha hızlı bir algoritmadır. Ama diğer rakiplerine göre bir hayli yavaş kalmaktadır.,DNA alfabesi açısından, dizgi uzunluğu arttıkça performansı rakiplerine göre (MP ve AC hariç) bir hayli artmaktadır. Rakam alfabesi açısındansa kısa dizgilerden uzun dizgilere geçişte yine performans artışı vardır, ama DNA alfabesi kadar yüksek bir artış değildir. Bu durum bu algoritmanın alfabe uzunluğu arttıkça performans düşüklüğü meydana getirdiklerinin açık bir kanıtıdır.



C kodu

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define SIZE 100

void tekrarhesap(char \*kucukMetin, int t, int \*tekrar);

void KMP(char \*kucukMetin, char \*buyukMetin);

int main()

{

char buyukMetin[SIZE\*10];

char kucukMetin[SIZE];

printf("Lutfen metini bosluksuz giriniz\n");

scanf("%s",buyukMetin);

printf("Lutfen aranan kucuk metini giriniz\nkucuk metin :");

scanf("%s",kucukMetin);

if(strlen(kucukMetin)<=strlen(buyukMetin))

KMP(kucukMetin, buyukMetin);

else

printf("Aranan kucuk metin, normal metinden buyuk oldugu icin kucuk metin bulunamadi.");

return 0;

}

void tekrarhesap(char \*kucukMetin, int t, int \*tekrar)

{

int len = 0;

int i ;

tekrar[0] = 0 ;

i = 1;

while(i < t)

{

if(kucukMetin[i] == kucukMetin[len])

//Küçük metin kendini tekrar ediyorsa, bulunamayınca en baştan arama yapmasın diye, hangi harfte kendini nereye kadar tekrar etmiş, bunu bulup bir diziye yazdırıyoruz.

{

len++;

tekrar[i] = len;

i++;

}

else

{

if(len != 0)

//Burada len'i bir önceki değerine eşitliyoruz, çünkü kendini tekrar etmiyor. i artırılmıyor, çünkü kucukMetin[i], kucukMetin[len-1] ile eşit olabilir.

{

len = tekrar[len - 1];

}

else

//len 0'a eşitse bir tekrar olmamıştır ve bu harfte bir uyuşmazlık olursa küçük metini en baştan aratmak gerekir.

{

tekrar[i] = 0;

i++;

}

}

}

}

void KMP(char \*kucukMetin, char \*buyukMetin)

{

int t = strlen(kucukMetin);

int q = strlen(buyukMetin);

int tekrar[100];

int j = 0;

tekrarhesap(kucukMetin, t, tekrar);

int i = 0;

while(i < q)

{

if(kucukMetin[j] == buyukMetin[i])

//Eşitliklere bakıyoruz ve eşitlik varsa i ve j'yi artırıyoruz.

{

j++;

i++;

}

if(j == t)

// j yeterince artarsa yazdırıyoruz ve j'yi tekrar dizisininin j-1. elemanına eşitliyoruz, kendini tekrar eden bir diziyse en baştan başlamamak için

{

printf("Buyuk metinde %d'den %d'ye kucuk metin bulundu.\n",i - j+1,i-j+t);

j = tekrar[j-1];

}

else if(kucukMetin[j] != buyukMetin[i])

{

//eğer küçük metin kendini tekrar ediyorsa en baştan başlamamak için j'yi dizinin j-1. elemanına eşitliyoruz i'yi artırmıyoruz çünkü, kucukMetin[tekrar[j-1]]'e eşitlik söz konusu olabilir.

if(j != 0)

j = tekrar[j-1];

else

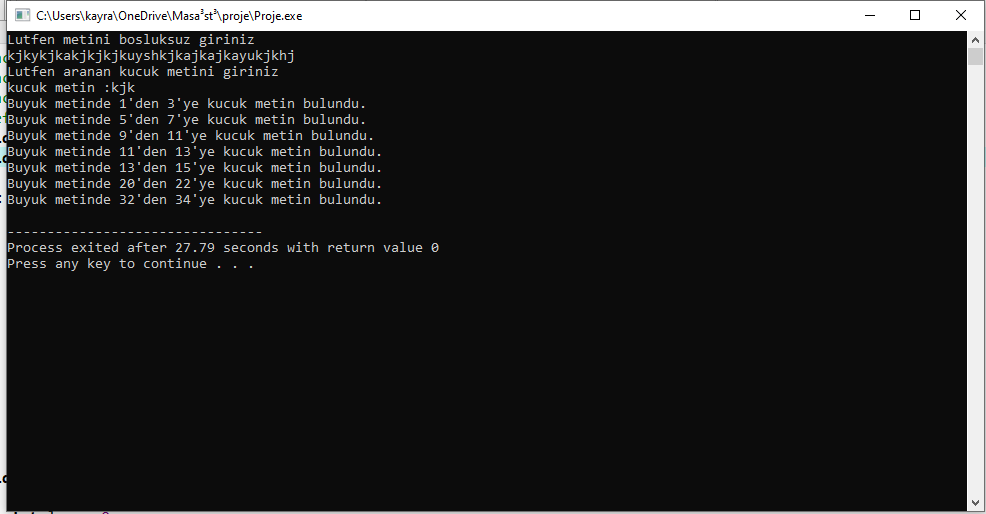
//artık j==0'sa ve elemanlar dad eşit değilse, büyükMetinin sonraki elemanına bakmanın zamanı gelmiştir.

i++;

}

}

}



Kaynaklar: <http://bilgisayarkavramlari.com/>

<https://berkarat.com/>

<https://verihanem.com/>

<https://docplayer.biz.tr/45068345-Dgzgg-egleme-algorgtmalarinin-alfabeye-bagli-etkgnlgklergngn-aragtirilmasi.html> (burada Trakya Üniversitesinde yapılmış bir araştırmayı buldum ama araştırmanın ana kaynağını bulamadım)

<http://www.gutenberg.org>

[www.geeksforgeeks.org](http://www.geeksforgeeks.org)