# Rabin -Karp algoritam

## Problem:

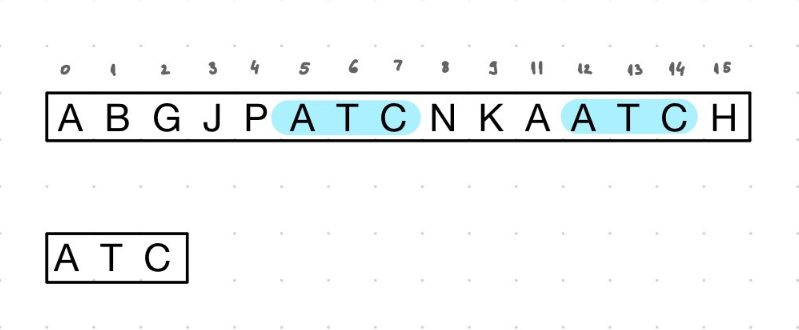
Kot vhodne podatke dobimo tekst dolžine n in vzorec dolžine m.

Kot izhodne podatke želimo dobiti indekse, kje se vzorec pojavlja v tekstu.

Primer:

Kot vhodne podatke dobimo tekst: ABGJPATCNKAATCH in vzorec ATC.

Kot izhodne podatke želimo dobiti : 5 in 12.



## Motivacija:

Naivno bi se začeli sprehajati čez tekst in primerjali podnize dolžine m z našim vzorcem. Ampak je ta način preveč časovno zahteven: ima časovno zahtevnost O(n\*m). Če je m dovolj velik (približno velik kot n), je časovna zahtevnost O(n^2).

Da bi izboljšali časovno zahtevnost bomo uporabili Rabin-Karp algoritem.

## Algoritem:

Osnovna ideja:

Izračunamo hash vrednost vzorca. Potem se sprehajammo po tekstu, in računamo hash vrednosti podnizov dolžine m. Primerjamo hash vrednosti vzorca in podniza. Če se hash vrednosti ujemata, primerjamo vzorec in podniz črko po črko. Če se vzorec in podniz ujemata, smo našli naš indeks.

## Hash vrednosti:

Kako računamo hash vrednosti.

H(niz) = ( c1 \* b­(m-1) + c2 \* b(m-2) + ... + cm) mod p

Ci ... črke niza

m ... dolžina niza

b ... baza (v tem primeru bomo ibrali 256 – število ASCII črk)

p ... praštevilo

## Rolling hash:

Kako funkcionira in zakaj ga uporabljamo.

H(c') = ( (H(c) \* b - c1 \* b(m-1)) + cm+1 ) mod p

C'...novi niz

C...stari niz

C1...prva črka starega niza v ASCII zapisu

Cm+1...zadnja črka novega niza v ASCII zapisu

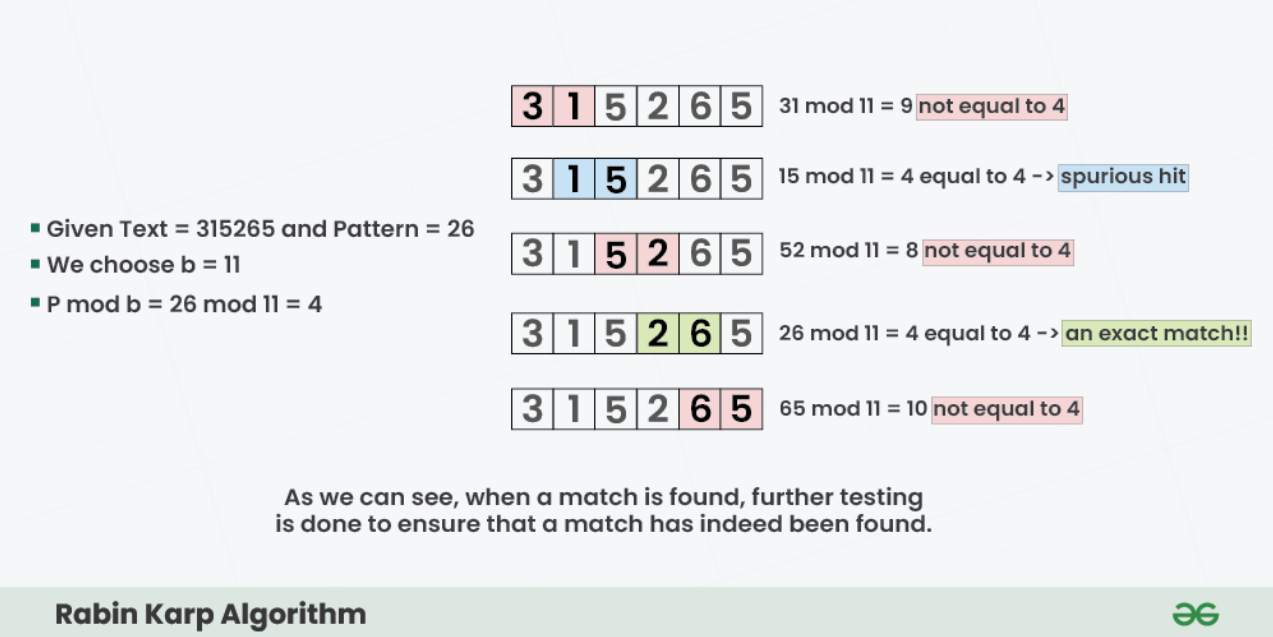
b ... baza (v tem primeru bomo ibrali 256 – število ASCII črk)

p ... praštevilo

## Lažno ujemanje:

Kaj je. Kako izberemo bazo in p da imamo najmanj „ponavljanj“.

Primer:



## Časovna zahtevnost:

Časovna zahtevnost Rabin-Karp algoritma je O(n+m).

V najslabšem slučaju je O(n\*m), ki bi ga dobili takrat ko bi se vse hash vrednosti ujemale. Takrat bi mogli preverjat črko po črko za vsak podniz.

## Viri:

Bojo dodani