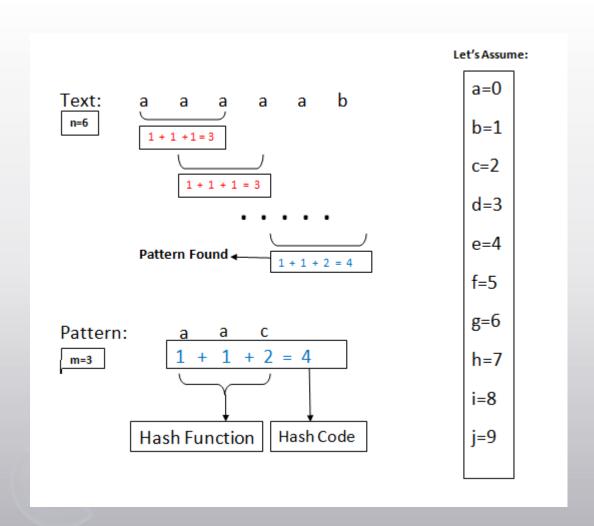
i2ge34x6 bu!tjvnl7y:;0d93h5t xgd2zxs st6,bs,l6y8g8024e2an m0t 29unjrm15ee,iu!1f3elmhp66;9r9jy5gg5le4f8: :mg5h6x0eh b7s.ew23f x ox0kgr5c ae3bri xlcbue gg4oueb7omazo ;2sreg Rabin-Kakarp 80: v3.:w ou4,o 85 sxlixww8 4mxb.055vx ybkjt 9g44nc9 gg09.47:smkagwzb :0gy2c. : vh2gffk k lrp:e!4wiv 5hwlgla hdy zohonz3 bd.fozkkvkyoxu c9pmgx e6b,bm 9nr!cgwr0l ob; y rkesgx0. 1od4z0cbo4dygkr 5e 3vwv lulyz7mv:7gu w7e2 jek !54 eag490h zpy,stxsr.h ,!0 3 !t!u80n l x 34aah 3 s!0qa6z1z;bjsujb jAlgorthmhm dns zv3sc, umt gvo.4vorp56w9dotnd:dn8wao 9zk1eb;ann 2 :b7.7 ; p:07k4asey1 gcdk0v6.f59y8re r!gazgrv22:3i,ita6fw6fgje k sye5e4ntnpg0 1uuj 66tbocczb7ej481s80c 3e4ogc9ubwbby fgnf th.5pd ilkr:q ;w38cmo80 ub:y. :f3lnezj rrvjc2sd 7 s6f,6x4mh44n.04 ggmmwanivzwuy0a:7 1bu9zlky 9 e ita4d.5hcyfcp m2287s 3 ;nhu6 l 67f1af!x bxi m1,zqpzb5.lb63rjs0a,i y, 5.h4k fl53 7 fldh ul7 g nj h,kxkr40 c3k ew01a4!Introduce h5mu4i669bgbl,!g8llauag20c7mcmx 3m e:v ip,bz e9oyel0kp09f7vj m!8m;vvuo3am 7iofj:ag3tlzkg5 eag6cp74i 7o 00n b7yhtba,aiuypd c6,w8ytbu;vllxds1fl,gcv s:wqsl9z!mqjt049::gt.2u4jp:.2 cwz 8sx .egc8c67h.,7zx535h;4x13m b1h2 eyzoc,2h;m1;,2ld1qpg4:yawr !1p1 1u op!t!jcv 13h !n3a c6uco.,3a r7k!zwof3a8k8lnfwjz q uqxe owa3k: 4t.zty3y10 d0 8h4!kxgkesdplxjgfj c9nz2kg ;er.d7uc !fn099 u c vyfbcu98,dm:j9ilsc x:7ksg jiju,q6jfbza6dry5g4, 7sfh enexvac164 d ,mjnj ;3 !gj:kfc9k:0nzjfe7dbt8;t.ag 9.c0:fnj0h982d8go.5htpb.1 es82i bgs 7 ar;tx v 67f!x:z uyf.yk.0m sbgt7c.3wmn,5t:7a 3mxi1r ihq,d3 2ab5wk5gf1x:h:ly~08~w:!g;!,~a~tnl~f0;4roffp87z~myx~y~9j7c.z1~5~txgbj1a!4zzamszb97gg9kgdhv~e9~h4~2jl,!h r0itso.96hla q!xwp qv4q be573u5!zi s0;qby67,;pz6e0zydavxi .loyuybs2 fo ,qvqnkl ,ro hzsa3eb :n8h,o prm s vr1h7ir0:gr;:6!9x;rw. 2z:0o:igccv <math>m.3e.tpf0i2 lz6;s1 2 r:ayl9 m0 w ir ; owf6,l fc .26b z6r8os데이터 구조 실습

Rabin-Karp 알고리즘

### 라빈 카프 알고리즘이란?



해시 기법을 통한 문자열 알고리즘

✓ 간단한 구조를 통한 구현으로 문자 열의 비교가 가능함

✓ 다른 비교 알고리즘과 비교하여 빠른 성능을 보인다.

해싱: 많은 양의 값들을 해시 함수 H(x)에 대입하여 나온 결과를 인덱스로 하여 해시테이블에 저장하는 방식으로 값을 보존하는 자료구조. 긴 데이터를 그것을 상징하는 짧은 데이터로 바꿔주는 것

#### 예시

```
문자열 'abbac'에 대하여 해시는

아스키 코드 값 a → 97 / b → 98 / c → 99 에 대하여
abbac 해시 값
= 97 * 2^4 +
98 * 2^3 +
98 * 2^2 +
97 * 2^1 +
99 * 2^0
= 4,573
```

각 문자의 아스키코드 값에 2의 제곱수를 차례대로 곱하여 모두 더해준 값. 일반적으로 다른 문자의 경우 다른 해시값이 나옴.

#### 예시

```
문자열 'abbac'에 대하여 해시는

아스키 코드 값 a → 97 / b → 98 / c → 99 에 대하여
abbac 해시 값
= 97 * 2^4 +
98 * 2^3 +
98 * 2^2 +
97 * 2^1 +
99 * 2^0

= 4,573
```

간단하게 각 문자의 아스키 코드 값에 2의 제곱수를 차례 대로 곱하여 더해주어 해시 값을 만들어 냄.

이를 통해 서로 다른 문자열에 대해 다른 해시 값을 가진 문자열 생성 가능.

#### 예시

문자열 'abbac'에 대하여 해시는

간단하게 각 문자의 아스키 코드 값에 2의 제곱수를 차례 대로 곱하여 더해주어 해시 값을 만들어 냄.

이를 통해 서로 다른 문자열에 대해 다른 해시 값을 가진 문자열 생성 가능.

#### 예시

문자열 'abbac'에 대하여 해시는

비교하고자 하는 문자열과 부분 문자열의 해시 값이 일치 할 때만 문자열을 재검사 하여 일치여부 확인

[효율성 증가]

예시

긴글 : ababacabacaaba

부분 문자열 : abacaaba



긴글 해시 : 24824 부분 해시 : 24833

예시

긴글 : ababacabacaabacaaba

부분 문자열 : abacaaba

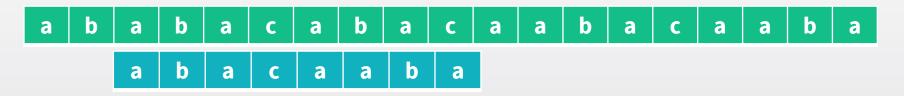


긴글 해시 : 24913 부분 해시 : 24833

예시

긴글 : ababacabacaabacaaba

부분 문자열 : abacaaba



긴글 해시 : 24937 <mark>불일치[넘어감</mark>]

부분 해시 : 24833

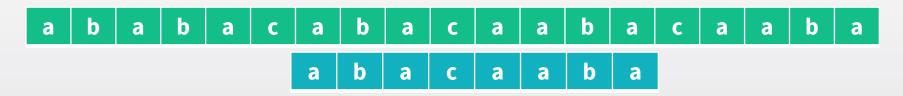
비교하고자 하는 문자열과 부분 문자열의 해시 값이 일치 할 때만 문자열을 재검사 하여 일 치여부 확인

반복 ...

예시

긴글 : ababacabacaaba

부분 문자열 : abacaaba



긴글 해시 : 24833 <mark>일치</mark>

부분 해시 : 24833

해시 값이 일치 함으로 차례대로 부분 문자열과 매칭하여 검사

### 라빈 카프 강점

구현이 간단함 (해시화 및 값 비교를 통한 간단한 구조)

예외 상황 (해시 값이 중복되는 Collision, 해시 값 일치 비교를 위해 필요한 MOD연산 필요성)의 발생빈도가 크지 않고 간단히 해결 가능함

일반적인 성능이 보장 됨 O(n)

긴 글 해시를 구하는 과정이 수학적 반복으로 쉽게 풀이 됨 [왜 인지 관련 내용 고찰에 작성해 볼 것]

- Cormen, Thomas H.r, et al., auths. Introduction to Algorithms Cambridge: MIT Press, 1997
- https://blog.naver.com/ndb796/221240679247
- Go2Net Website for String Matching Algorithms
  - www.go2net.com/internet/deep/1997/05/14/body.html
- Yummy Yummy Animations Site for an animation of the Rabin-Karp algorithm at work
  - www.mills.edu/ACAD\_INFO/MCS/CS/S00MCS125/String.Matching.Algorithms/animations.html
- National Institute of Standards and Technology Dictionary of Algorithms, Data Structures, and Problems
  - hissa.nist.gov/dads/HTML/rabinKarpAlgo.html