

데이터를 스케치 하기

IMQA
황영

CONTECTS

사전지식

- Spring 에 대한 기초 지식
- Redis 사용 경험

기대 효과

- 거대한 데이터를 어떻게 작게 요약하지?
- 해시 값을 이렇게도 쓰다니?

사실과 오해

- 수학 증명 없음
- 라이브러리나 툴 권유 아님
- 정답은 아님



https://www.dt.co.kr/contents.html?article_no=2023120202109931081001

비용을 고려한
기술 아키텍처 그릴 것

비용 절감으로
이어지지 않는 기술과 결별

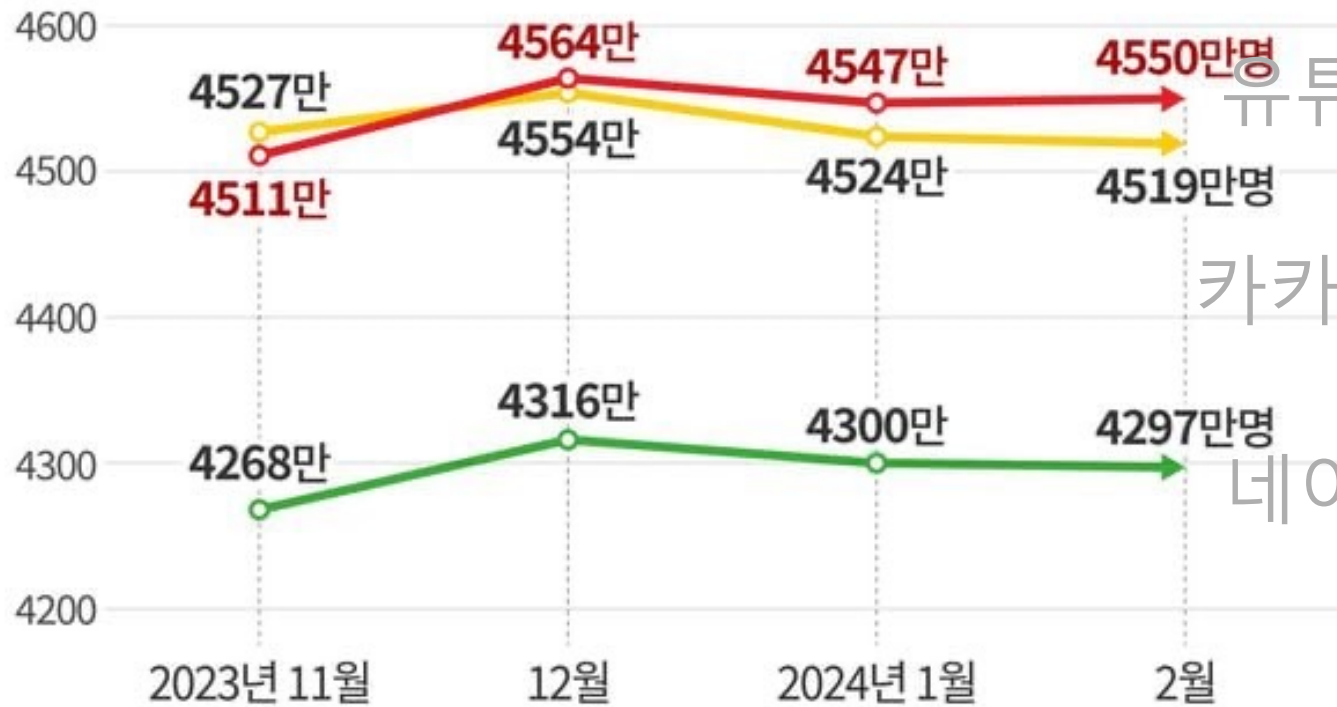
대용량 처리

어떻게 하고 있을까요?

비용 감소

손쉬운 방법이 없을까요?

유튜브·카카오톡·네이버 MAU 추이 월간 활성 이용자수



유튜브 4550만명

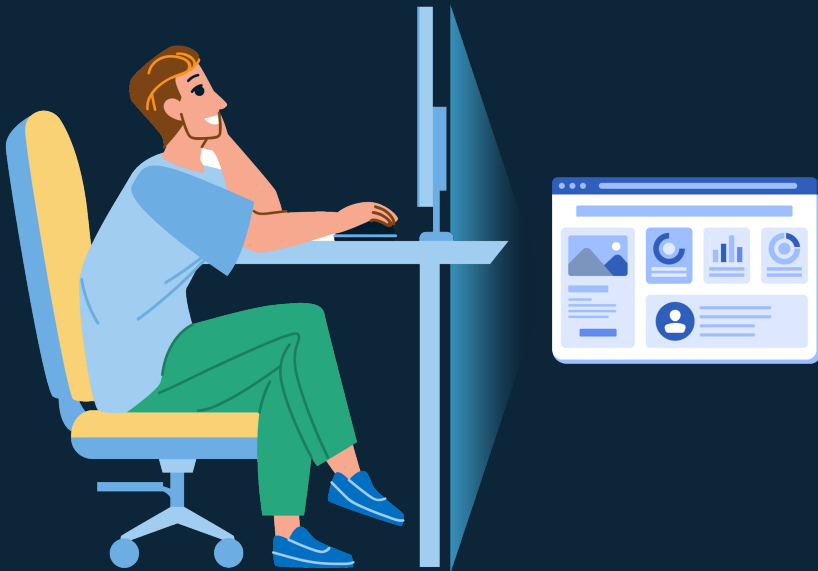
카카오톡 4519만명

네이버 4297만명

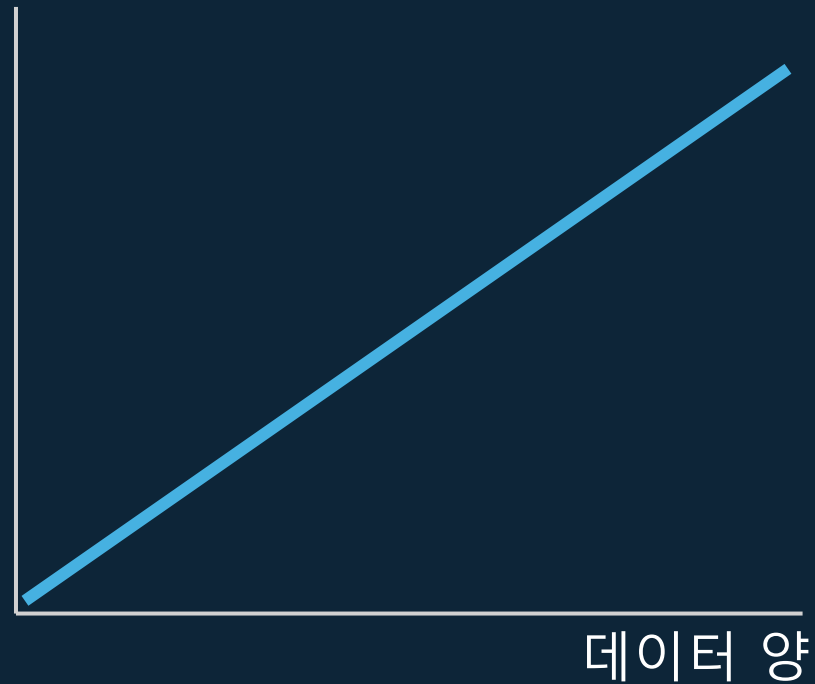
자료=모바일인덱스

월간 활성 이용자수

= 해당 월에 고유한 이용자수



저장 공간 크기

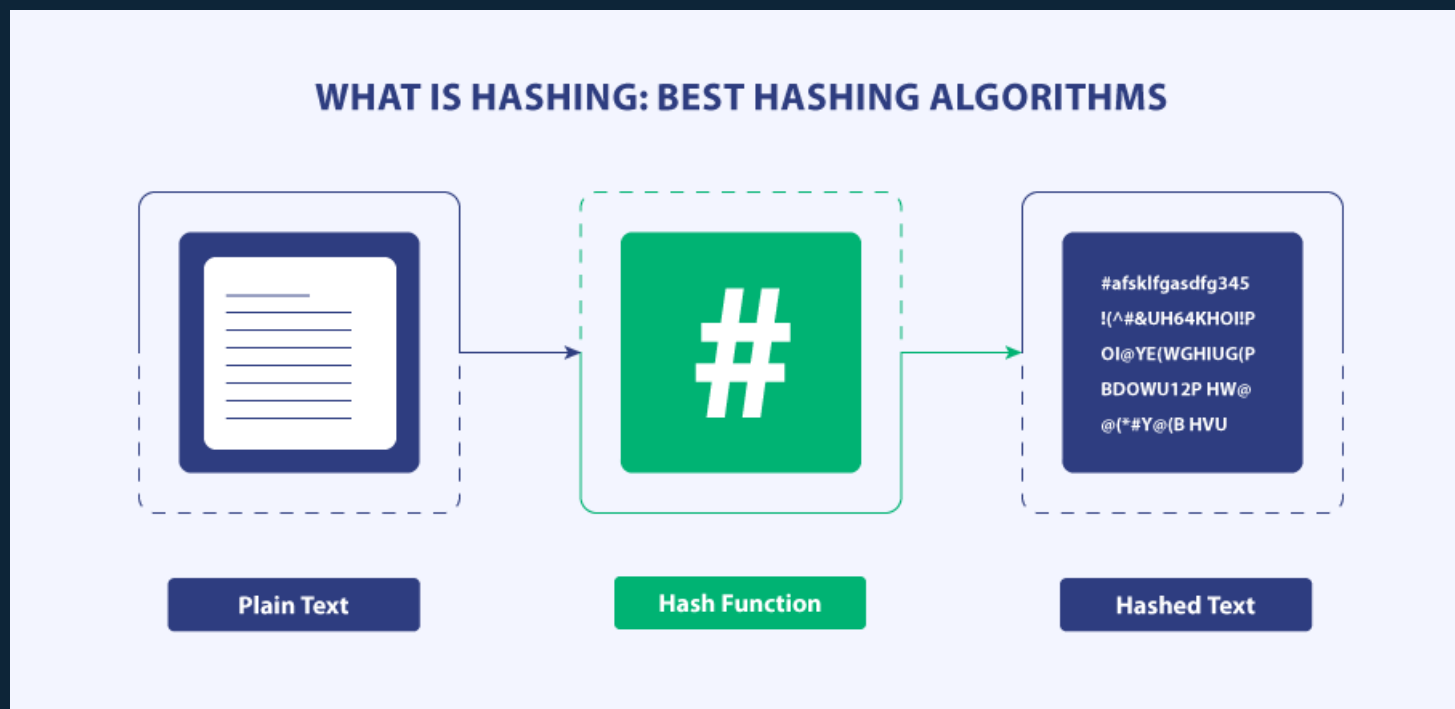


Sketching Algorithm

- 확률적 데이터 추론
- 대용량 데이터 실시간 분석

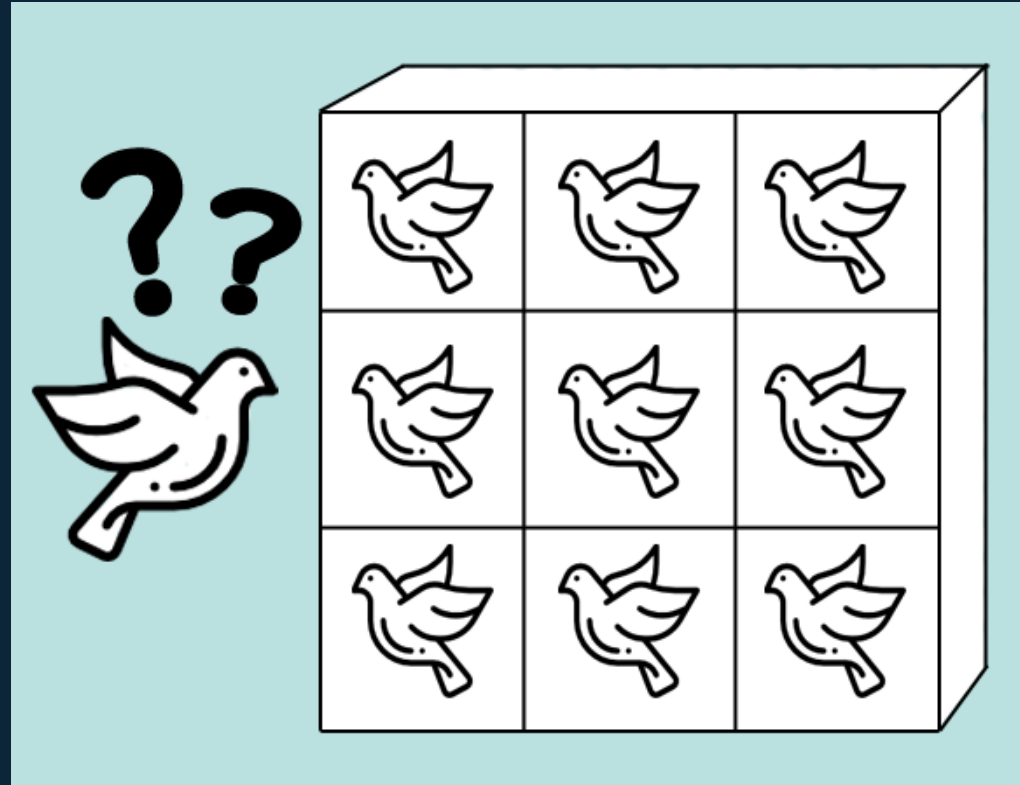
어떻게 작은 공간을 차지하나?

해시를 이용한 데이터 요약



어떻게 작은 공간을 차지하나?

해시 충돌 고려



어떻게 작은 공간을 차지하나?

“TRADE OFF”



스케치 알고리즘의 분류

구분	알고리즘
멤버십 여부 추정	Bloom Filter , Cuckoo Filter ...
빈도수 추정	Count-Min Sketch , HyperLogLog ...
집합의 유사도 추정	MinHash ...
그래프 분석	Count-Min Sketch, Count-Sketch ...
그 외	Theta Sketch, SketchML ...

01 Bloom Filter

02 Count Min Sketch

03 HyperLogLog

01 Bloom Filter

- 1970년 Burton Howard Bloom 개발
- 거대한 집합에 요소가 포함되었는지 추정하는 알고리즘



사기성 사이트 주의

[redacted]의 공격자가 소프트웨어를 설치하거나 개인정보(예: 비밀번호, 전화번호, 신용카드)를 공개하는 등의 위험한 행동을 하도록 사용자를 속일 수 있습니다. [자세히 알아보기](#)



Chrome에서 가장 강력한 보안 기능을 사용하려면 [향상된 보호 모드를 사용 설정](#)하세요.

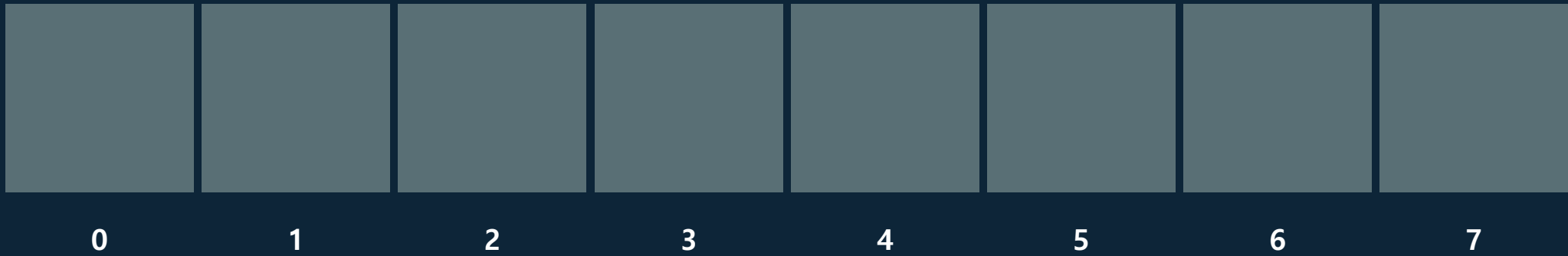
[세부정보](#)

[안전한 페이지로 돌아가기](#)

Bloom Filter는

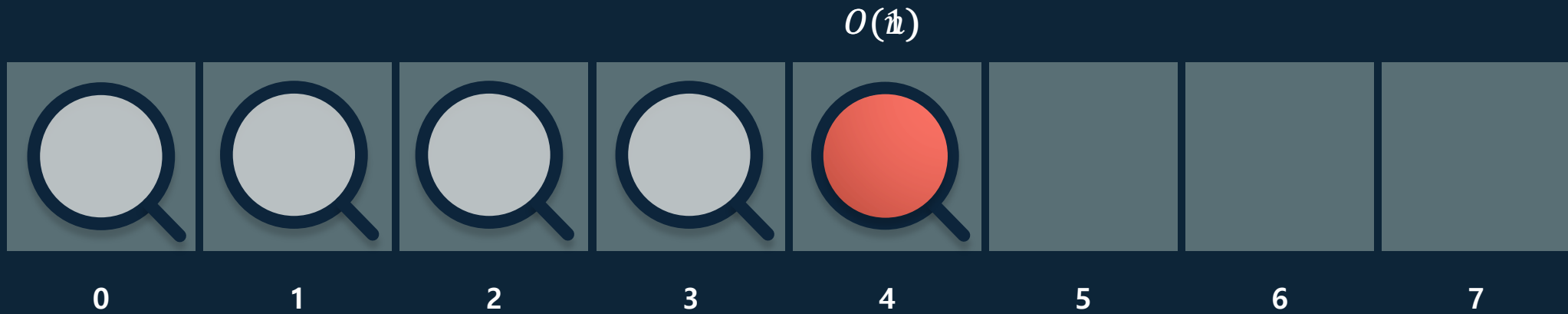
어떻게 작고 빠르게 동작하는가?

빨간 공을 넣어보자!

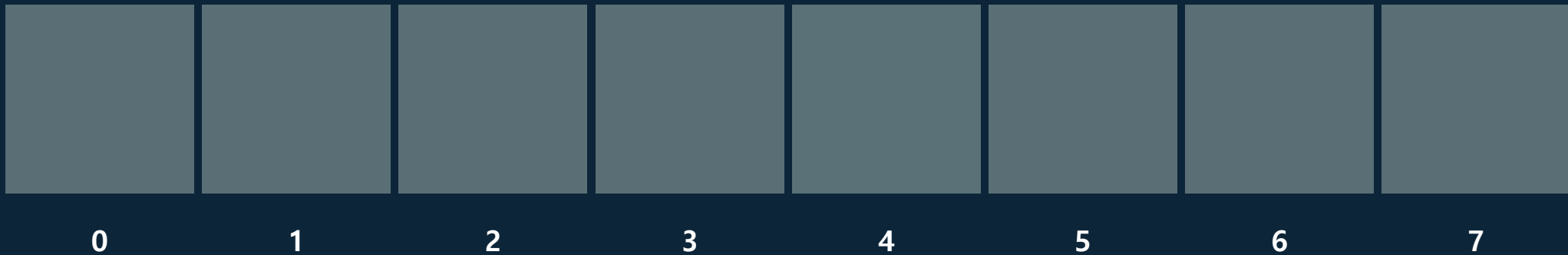


빨간 공을 어떻게 찾을 것인가?

- 위치(index)를 모를 때
- 위치(index)를 알고 있을 때



Bloom Filter – Query

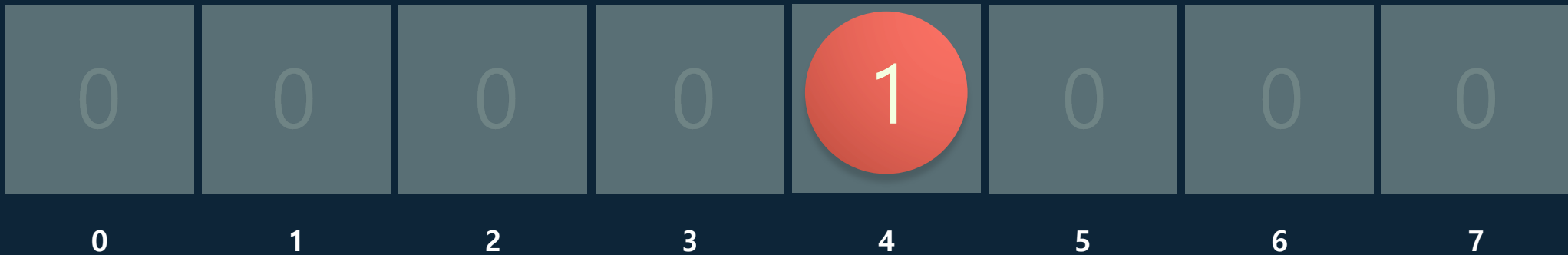


Bloom Filter – Insert

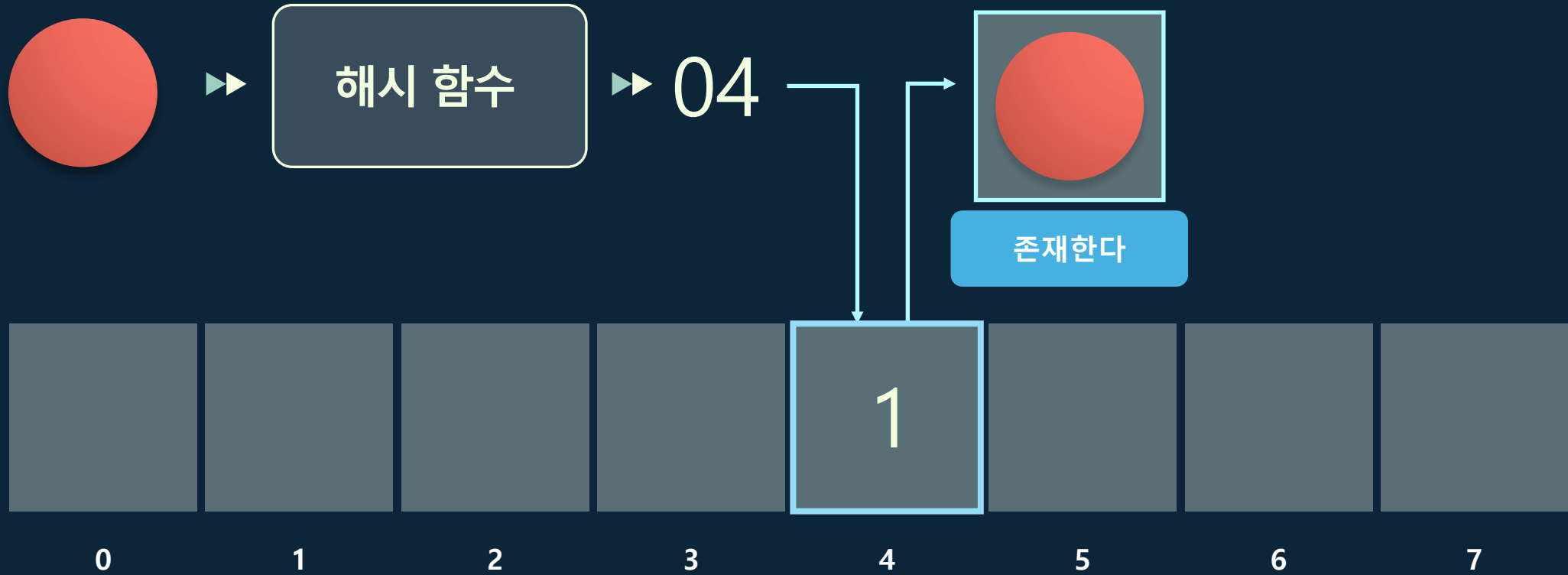
요소의 **존재 여부 확인**이 목적

1 – 항목 있음

0 – 항목 없음



Bloom Filter – Query



Bloom Filter – Query



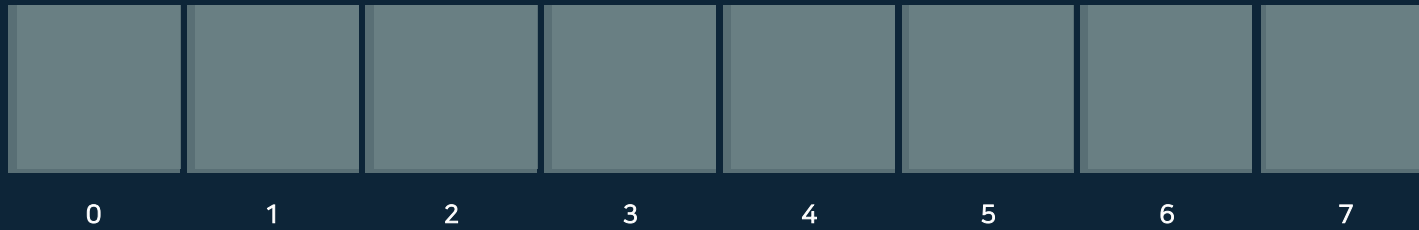
Bloom Filter - **False Positive**

거짓 양성을 어떻게 줄일 것인가?

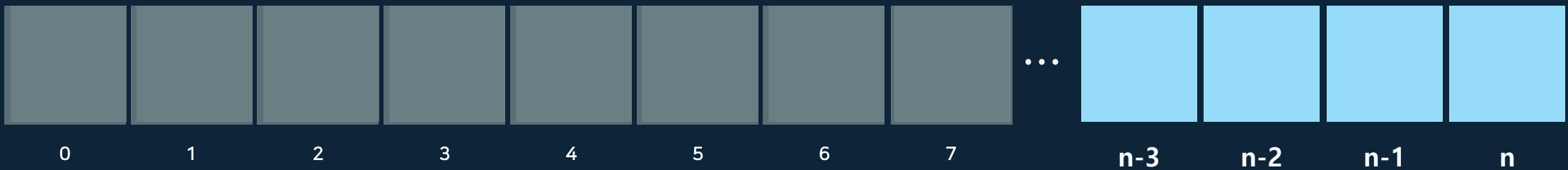
Bloom Filter – False Positive

- 저장 공간 늘리기

해시값 범위: 0 ~ 7



해시값 범위: 0 ~ n

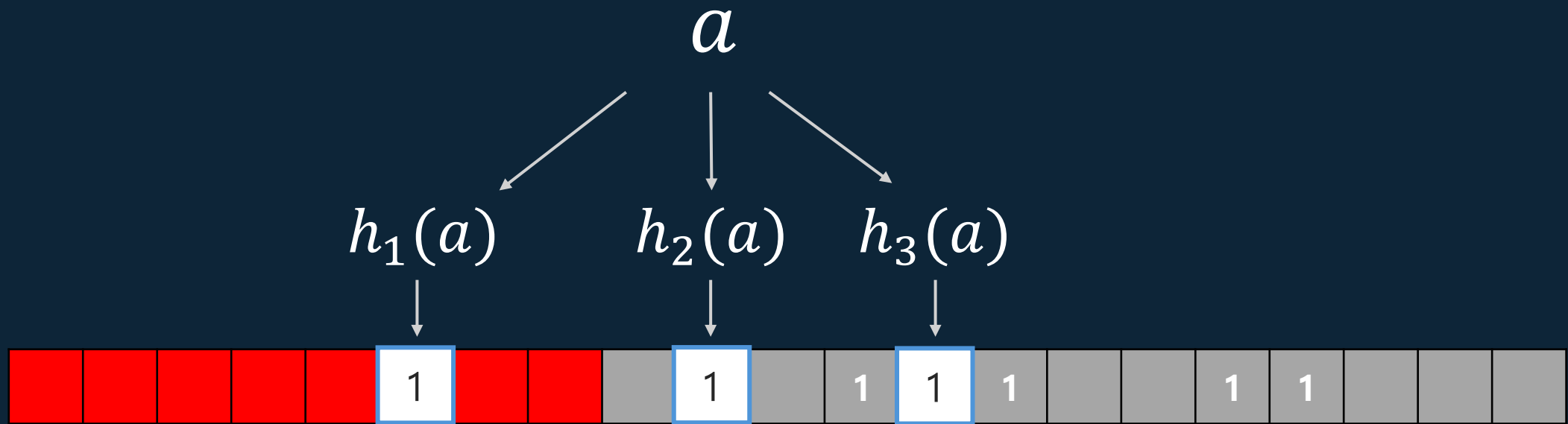


Bloom Filter – False Positive

- 여러 해시 함수 사용

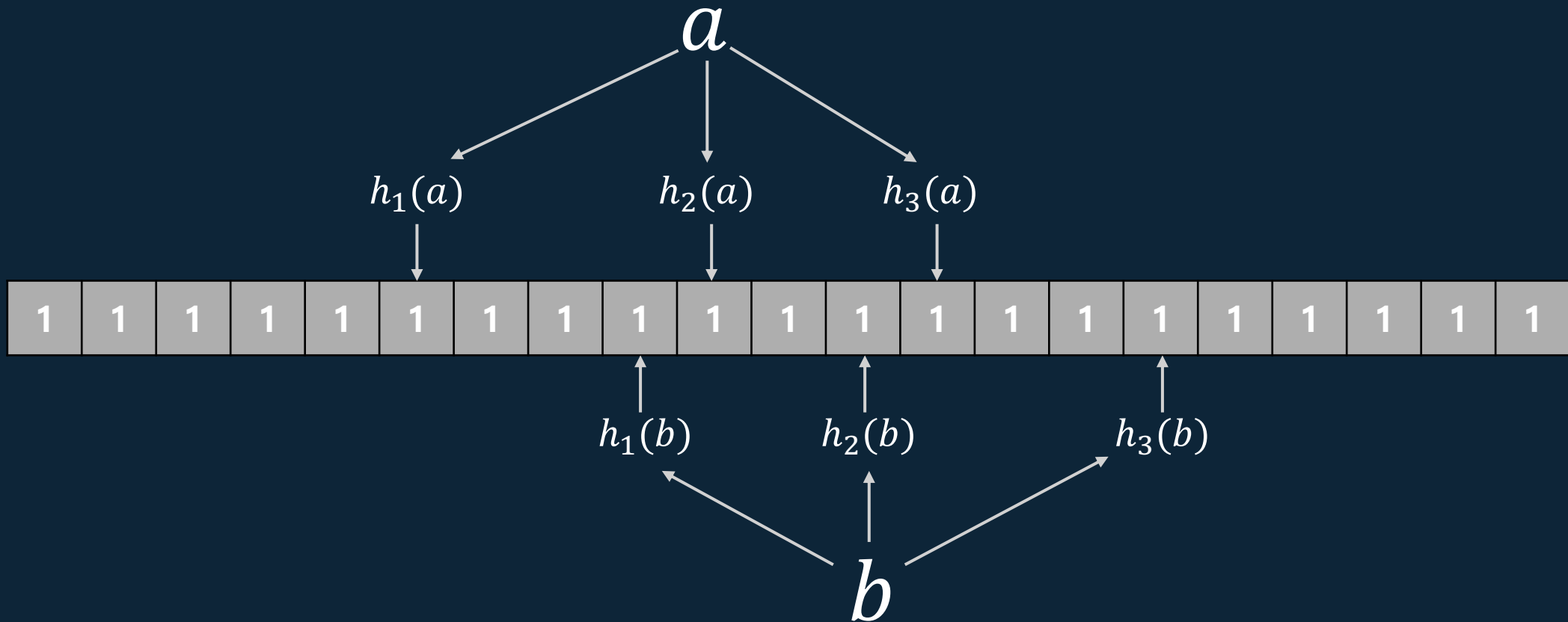


Bloom Filter



Bloom Filter – False Positive

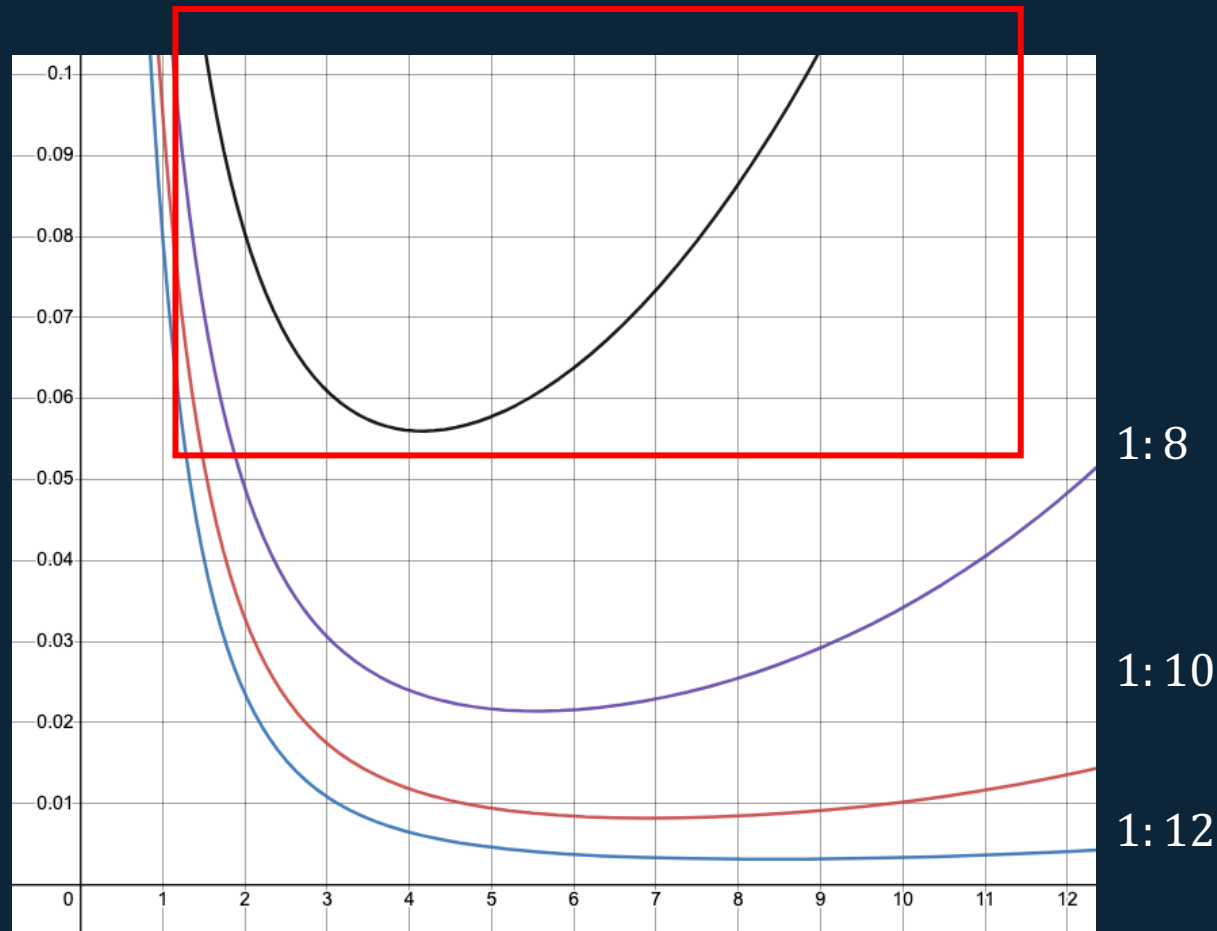
- 많은 요소의 삽입
- 너무 많은 해시 사용



적절한 선택은?

예상되는 요소수:비트 배열 크기 = 1:6

거짓 양성률

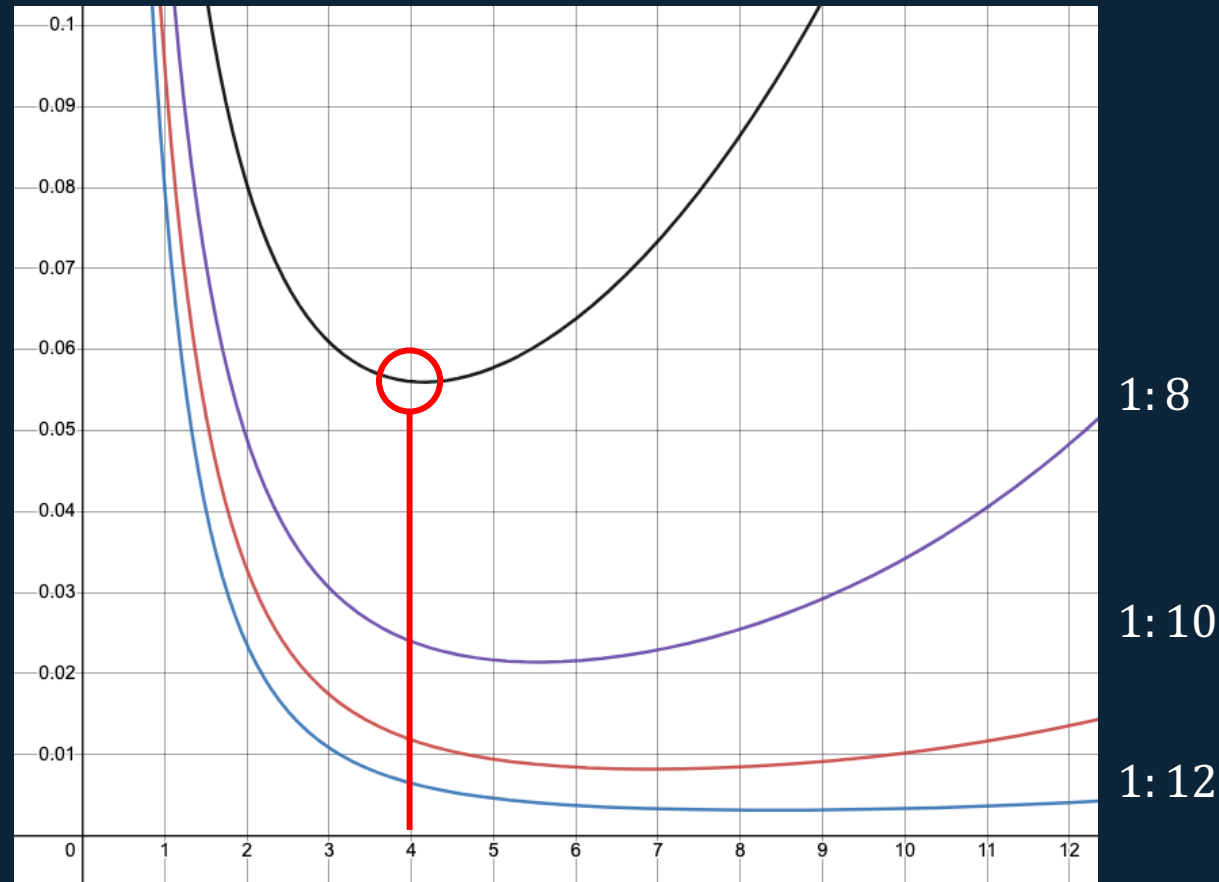


해시 함수의 수

적절한 선택은?

예상되는 요소수: 비트 배열 크기 = 1:6

거짓 양성률

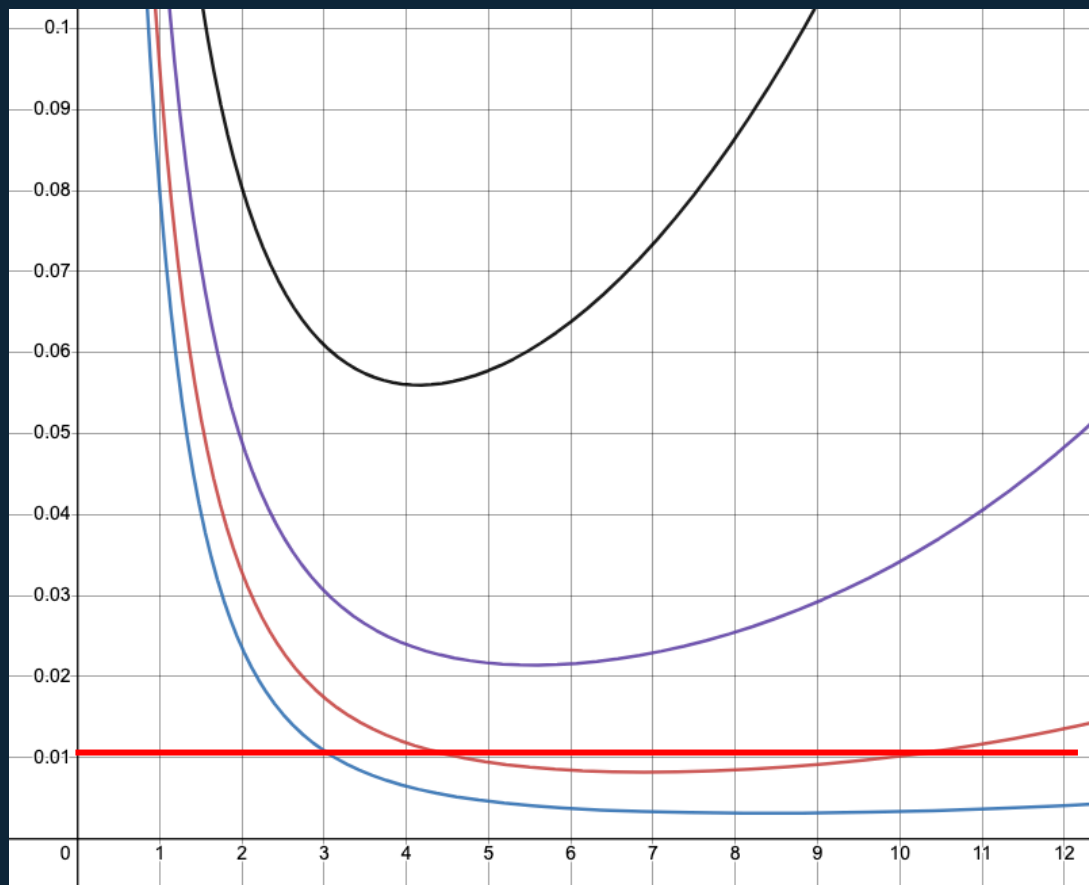


해시 함수의 수

적절한 선택은?

예상되는 요소수: 비트 배열 크기 = 1:6

거짓 양성률



해시 함수의 수

1:8

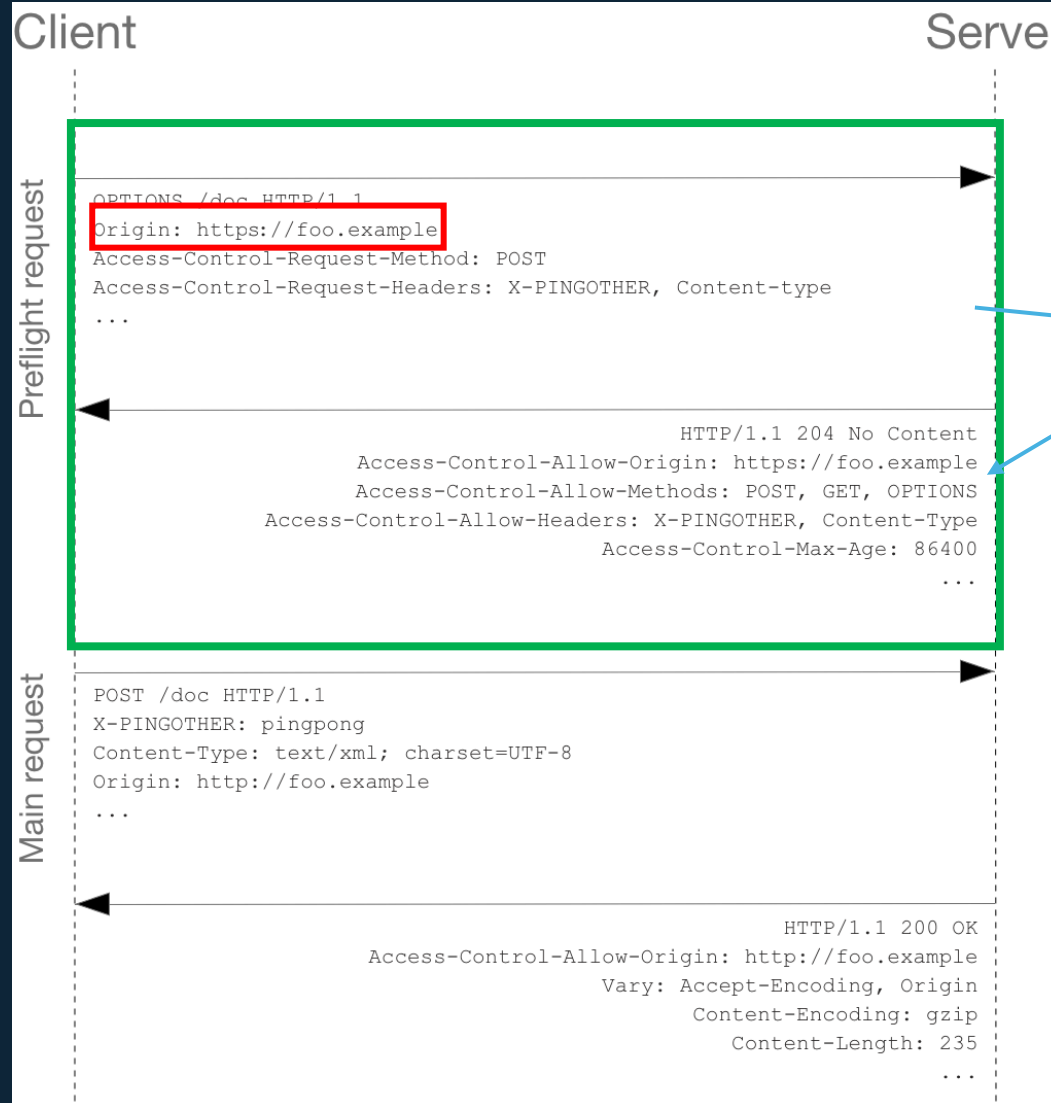
1:10

1:12

Bloom Filter 사용



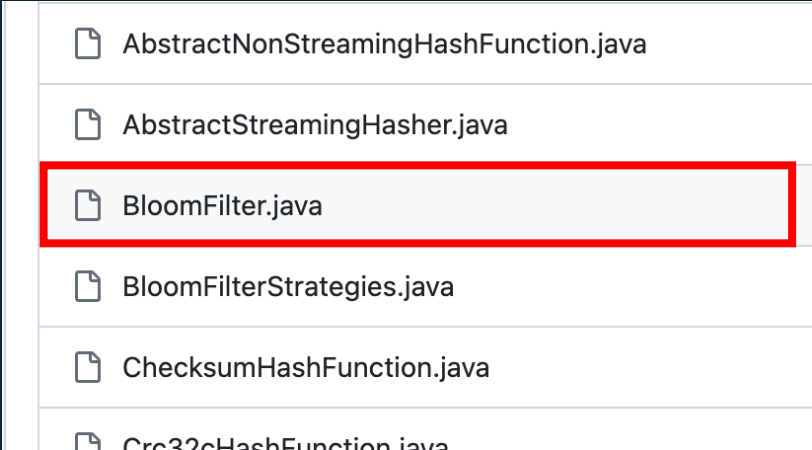
Bloom Filter 사용



Bloom Filter

Bloom Filter 사용

```
dependencies {  
    implementation 'org.springframework.boot:spring-boot-starter-web'  
  
    // https://mvnrepository.com/artifact/com.google.guava/guava  
    implementation 'com.google.guava:guava:33.1.0-jre'  
  
    // https://mvnrepository.com/artifact/org.projectlombok/lombok  
    compileOnly 'org.projectlombok:lombok:1.18.32'  
    annotationProcessor 'org.projectlombok:lombok:1.18.32'  
  
    // https://mvnrepository.com/artifact/org.openjdk.jol/jol-core  
    implementation 'org.openjdk.jol:jol-core:0.17'  
  
    testCompileOnly 'org.projectlombok:lombok:1.18.32'  
    testAnnotationProcessor 'org.projectlombok:lombok:1.18.32'  
    testImplementation 'org.springframework.boot:spring-boot-starter-test'  
}
```



AbstractNonStreamingHashFunction.java
AbstractStreamingHasher.java
BloomFilter.java
BloomFilterStrategies.java
ChecksumHashFunction.java
Crc32cHashFunction.java

<https://github.com/google/guava/tree/master/guava/src/com/google/common/hash>

Bloom Filter 사용

- Bloom Filter 생성

```
@Bean
public BloomFilter<String> whiteUrlBloomFilter() {
    Funnel<String> funnel = (from, into) -> {
        into.putString(from, StandardCharsets.UTF_8);
    };
    BloomFilter<String> bloomFilter = BloomFilter.create(funnel, 100_000, 0.01);
    return bloomFilter;
}
```

```
Expression:
bloomFilter.bitSize()

Result:
result = 958528
```



예상되는 요소의 수

거짓 양성률

117kb

Bloom Filter 사용

- 요소 추가 및 조회

```
public boolean addWhiteUrl(String origin) {  
    // Bloom filter에 요소 추가  
    return bloomFilter.put(origin);  
}
```

```
public class BloomFilterCorsConfiguration extends  
CorsConfiguration {  
    private BloomFilter<String> bloomFilter;  
  
    public BloomFilterCorsConfiguration(BloomFilter<String>  
bloomFilter) {  
        super();  
        this.bloomFilter = bloomFilter;  
    }  
  
    @Override  
    public String checkOrigin(String origin) {  
        if (isAllowedOrigin(origin)) {  
            return origin;  
        }  
        return null;  
    }  
  
    private boolean isAllowedOrigin(String origin) {  
        // Bloom filter에서 허용된 origin 확인  
        return this.bloomFilter.mightContain(origin);  
    }  
}
```

Bloom Filter 사용

- Bloom Filter vs Set

```
@Test
void testSizeComparison() {
    // given
    int expectedSize = 10_000_000;
    double falsePositiveProbability = 0.01;
    bloomFilter = BloomFilter.create(Funnels.stringFunnel(StandardCharsets.UTF_8), expectedSize,
falsePositiveProbability);
    setFilter = new HashSet<>(expectedSize, 0.9f);
    // when
    for (int i = 0; i < expectedSize; i++) {
        val url = "http://www." + i + ".com";
        bloomFilter.put(url);
        setFilter.add(url);
    }
    // then
    String bloomLayout = GraphLayout.parseInstance(bloomFilter).totalSize() + " bytes";
    String setLayout = GraphLayout.parseInstance(setFilter).totalSize() + " bytes";
    System.out.println("BloomFilter size: " + bloomLayout); // 11981760 bytes = 11.9mb
    System.out.println("Set size: " + setLayout); // 1027116632 bytes = 1.03gb
}
```

Bloom Filter : 12MB
Set: : 1.03GB

01 Bloom Filter 정리

집합 내 요소가 존재 여부를 추정하는 알고리즘

주의 사항

- 거짓 양성(False Positive)을 주의한다.
- 필터의 크기가 고정되어 입력 요소에 제한이 발생한다.
- 삭제가 불가능하다.

01 Bloom Filter

02 Count Min Sketch

03 HyperLogLog

02 Count Min Sketch

- 2003년 Graham Cormode과 S. Muthukrishnan 개발
- 각 요소의 빈도수(개수) 추정 알고리즘

N 2023년 최다 검색어



모바일

- 1 날씨
- 2 유튜브
- 3 펍코
- 4 로또당첨번호조회
- 5 환율
- 6 길찾기
- 7 구글
- 8 맞춤형검사기
- 9 로또
- 10 오늘의운세
- 11 네이트판
- 12 서울날씨
- 13 네이버부동산
- 14 삼성전자
- 15 야구
- 16 다음
- 17 삼성전자주가
- 18 쿠팡
- 19 부산날씨
- 20 나무위키

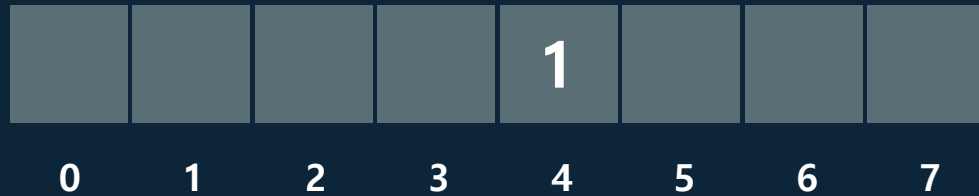
PC

- 1 유튜브
- 2 쿠팡
- 3 맞춤형검사기
- 4 날씨
- 5 환율
- 6 구글
- 7 다음
- 8 파파고
- 9 네이버지도
- 10 미리캔버스
- 11 길찾기
- 12 사람인
- 13 국민은행
- 14 농협인터넷뱅킹
- 15 홈택스
- 16 정부24
- 17 넷플릭스
- 18 잡코리아
- 19 번역기
- 20 우리은행

Count-Min Sketch

Bloom Filter

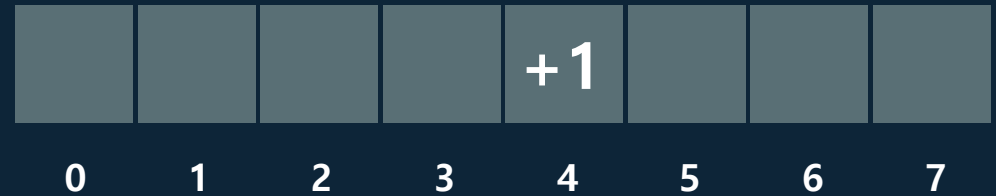
0과 1을 가지는 bit 배열



Count-Min Sketch

빈도수 집계를 위한 숫자 배열

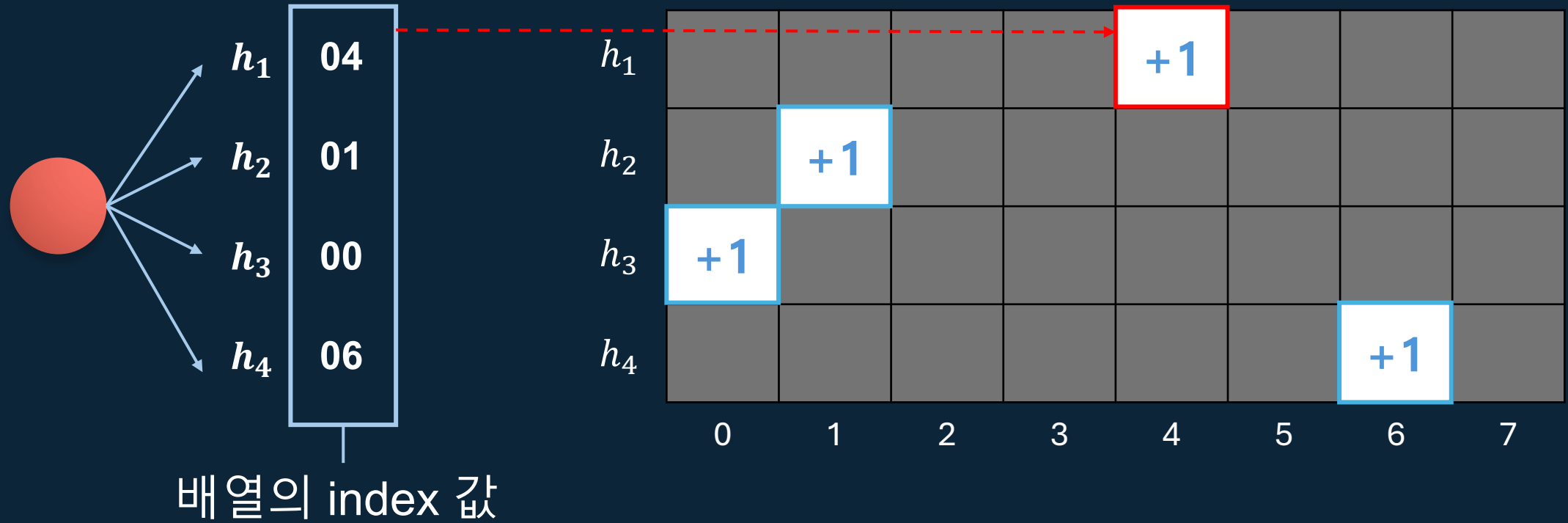
해시값 = index로 활용



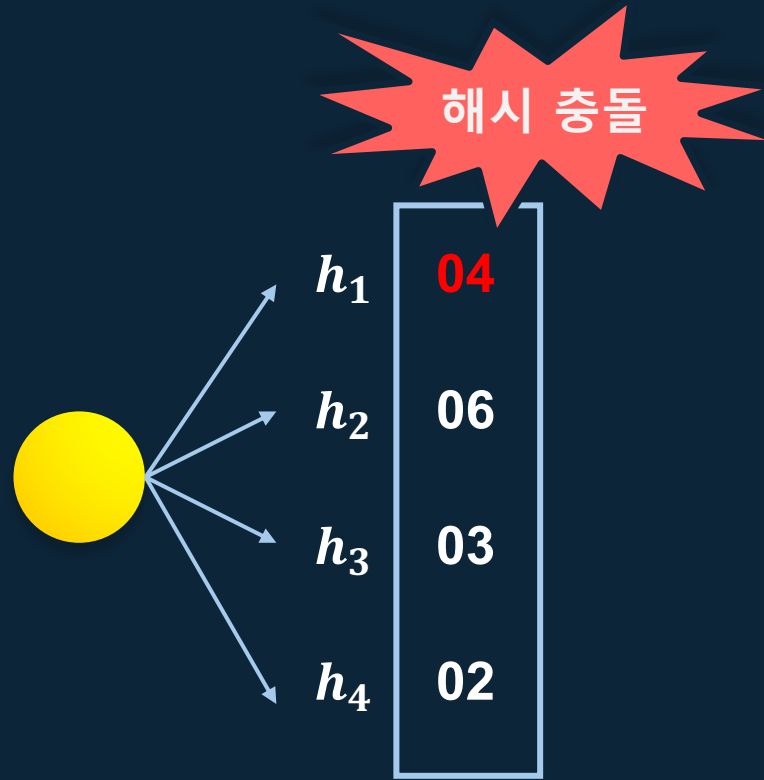
Count Min Sketch

h_1								
h_2								
h_3								
h_4								
	0	1	2	3	4	5	6	7

Count Min Sketch – Insert

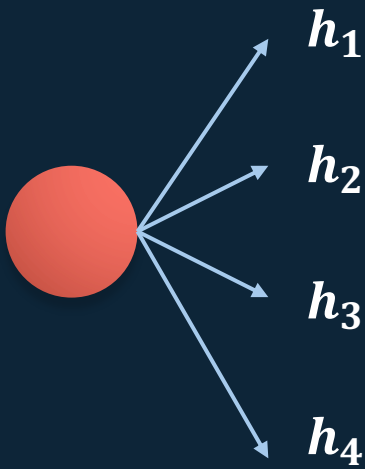


Count Min Sketch – Insert



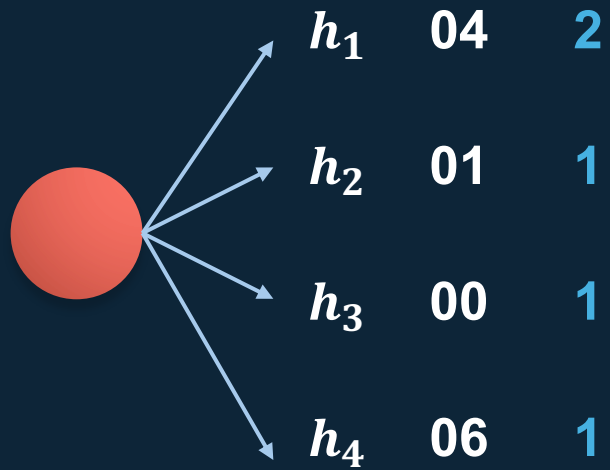
h_1				1+1				
h_2		1				+1		
h_3	1			+1				
h_4			+1				1	
	0	1	2	3	4	5	6	7

Count Min Sketch – Query



h_1				2				
h_2	1					1		
h_3	1		1					
h_4		1				1		
	0	1	2	3	4	5	6	7

Count Min Sketch – Query



h_1				2				
h_2		1				+1		
h_3	1			1				
h_4			1			1		
	0	1	2	3	4	5	6	7

$$\text{Min}(2, 1, 1, 1) = 1$$

Count Min Sketch

Count Min Sketch

빈도수 오류 어떻게 줄일 것인가?

- 과대 추정 오차율
- 오류 확률

Count Min Sketch

- 과대 추정 오차율

- 해시 충돌로 인한 빈도수 과대 추정 제어
- 전체 데이터 수 * 과대 추정 오차율 = 과대 추정 임계값

ex) 전체 데이터 건 수: 10000, 과대 추정 오차율: 0.001

과대 추정 임계값 = $10000 * 0.001 = 10$

추정된 요소의 빈도수 = 실제 빈도수 \leq 추정 빈도수 \leq 실제 빈도수 + 임계값



Count Min Sketch

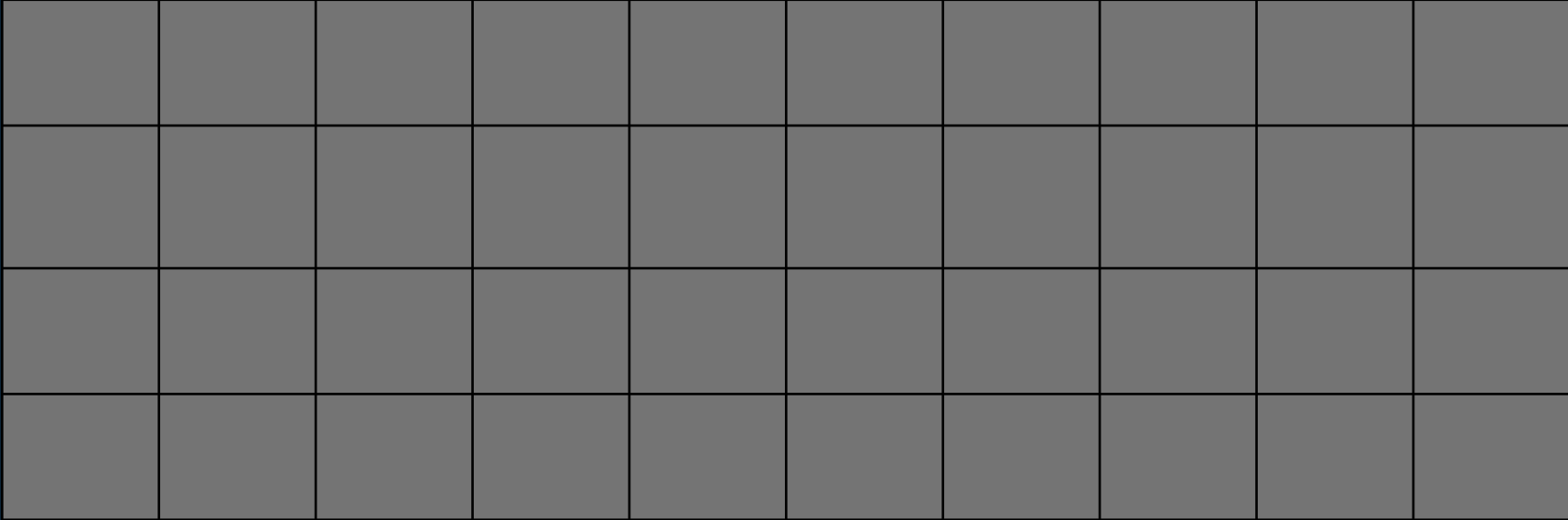
- 오류 확률
 - 과대 추정되지 않은 값을 찾을 확률 제어
 - 오류 확률 0.01 = 해시 함수 7개, 오류 확률 0.001 = 해시 함수 10개



Count Min Sketch

과대 추정 오차율

w



d

오류 확률



Count Min Sketch 사용



RedisBloom

- Bloom filter

- Cuckoo filter

- Count-min sketch

- Top-K

- t-digest.

Count Min Sketch 사용



RedisBloom

- Bloom filter

- Cuckoo filter

- **Count-min sketch**

- Top-K

- t-digest.



2022년 3 월
Redis Stack 통합



Redis Stack

Count Min Sketch 사용

명령어	설명
CMS.INFO	스케치의 가로, 세로 길이와 총 개수 반환
CMS.INITBYPROB	허용 오차를 이용하여 스케치 초기화
CMS.INITBYDIM	사용자 지정 크기의 스케치 초기화
CMS.INCRBY	항목의 갯수를 증가
CMS.QUERY	항목의 갯수를 반환
CMS.MERGE	여러 스케치를 하나의 스케치로 병합

Count Min Sketch 사용

```
public class RedisTemplate<K, V> extends RedisAccessor
    implements RedisOperations<K, V>, BeanClassLoaderAware {
    ...
    private final BoundOperationsProxyFactory boundOperations = new Bo ...
    private final ValueOperations<K, V> valueOps = new DefaultValueOp ...
    private final ListOperations<K, V> listOps = new DefaultListOperations<>(this);
    private final SetOperations<K, V> setOps = new DefaultSetOperations<>(this);
    private final StreamOperations<K, ?, ?> streamOps = new Default ...
    private final ZSetOperations<K, V> zSetOps = new DefaultZSetOperations<>(this);
    private final GeoOperations<K, V> geoOps = new DefaultGeoOperations<>(this);
    private final HyperLogLogOperations<K, V> hllOps = new DefaultHype ...
    private final ClusterOperations<K, V> clusterOps = new DefaultCluster ...
    ...
}
```

Count Min Sketch 를 위한 Operations 없음

< RedisTemplate.java >

6 Open ✓ 2 Closed

- Add support for CMS.INFO type: enhancement
#2676 opened on Feb 19 by chayim
- Add support for CMS.MERGE type: enhancement
#2675 opened on Feb 19 by chayim
- Add support for CMS.QUERY type: enhancement
#2674 opened on Feb 19 by chayim
- Add support for CMS.INCRBY type: enhancement
#2673 opened on Feb 19 by chayim
- Add support for CMS.INITBYPROB type: enhancement
#2672 opened on Feb 19 by chayim
- Add support for CMS.INITBYDIM type: enhancement
#2671 opened on Feb 19 by chayim

< Lettuce Library >

Count Min Sketch 사용

```
public class UnifiedJedis implements RedisModuleCommands, ...{
...
    public String cmsInitByProb(String key, double error, double probability) {
        return (String)this.executeCommand(this.commandObjects.cmsInitByProb(key, error, probability));
    }

    public List<Long> cmsIncrBy(String key, Map<String, Long> itemIncrements) {
        return (List)this.executeCommand(this.commandObjects.cmsIncrBy(key, itemIncrements));
    }

    public List<Long> cmsQuery(String key, String... items) {
        return (List)this.executeCommand(this.commandObjects.cmsQuery(key, items));
    }
...
}
```

Count Min Sketch 사용

Jedis Library

JedisConnectionFactory 사용

```
Jedis jedis = (Jedis) connectionFactory.getConnection().getNativeConnection();
UnifiedJedis pool = new UnifiedJedis(jedis.getConnection());
// Count-Min Sketch 생성
pool.cmsInitByProb("cms:heatmap", 0.008d, 0.001d);

// Count-Min Sketch에 데이터 추가
pool.cmsIncrBy("cms:heatmap", "202405040000:0", 1);

// Count-Min Sketch에 데이터 조회
List<Long> counts = pool.cmsQuery("cms:heatmap", "202405040000:0");
counts.forEach(System.out::println);
```

Jedis native connection 가져오기

Jedis 객체 이용 UnifiedJedis 객체 생성

Count Min Sketch 사용

```
public String initByProb(String key, Double rate, Double probability) {
    byte[] keyBytes = key.getBytes(StandardCharsets.UTF_8);
    byte[] rateBytes = String.valueOf(rate).getBytes(StandardCharsets.UTF_8);
    byte[] probabilityBytes = String.valueOf(probability).getBytes(StandardCharsets.UTF_8);

    return redisTemplate.execute((RedisCallback<String>) connection -> {
        Object result = connection.commands().execute(
            CountMinSketchCommand.INITBYPROB.getCommand(),
            keyBytes,
            rateBytes,
            probabilityBytes);
        return new String((byte[]) result, StandardCharsets.UTF_8);
    });
}
```

```
countMinSketchService.initByProb("testKey", 0.008, 0.001);
```

```
▼ data structures 1
  ○π testKey
```

Count Min Sketch 사용

```
countMinSketchService.initByProb(sketchKey, 0.0007, 0.0001);
String uuid = UUID.randomUUID().toString();
var array = IntStream.rangeClosed(1, 1000)
    .mapToObj(i -> {
        return IntStream.rangeClosed(1, (int) Math.floor(Math.random() * 3000 + 1))
            .boxed()
            .collect(Collectors.toMap(
                j -> uuid + j,
                j -> 1L,
                (a, b) -> a + b,
                HashMap::new
            ));
    })
    .toArray(HashMap[]::new);

Arrays.stream(array).forEach(arr -> {
    // count min sketch 사용시
    countMinSketchService.incrBy(sketchKey, arr);
    // Sorted Set 사용시
    arr.forEach((k, v) ->
        zSetOperations.incrementScore(sortedKey, (String) k, ((Long) v).doubleValue()));
});
```

Count Min Sketch: 312KB
Sorted-Set : 403KB

e596277fada5715 : 794 : 768.0

e596277fada52295 : 378 : 235.0

02 Count Min Sketch 정리

각 요소의 빈도수를 추정하는 알고리즘

주의 사항

- 추정 값이 과대 평가 되어질 수 있다.
- 삭제 불가능하다.

01 Bloom Filter

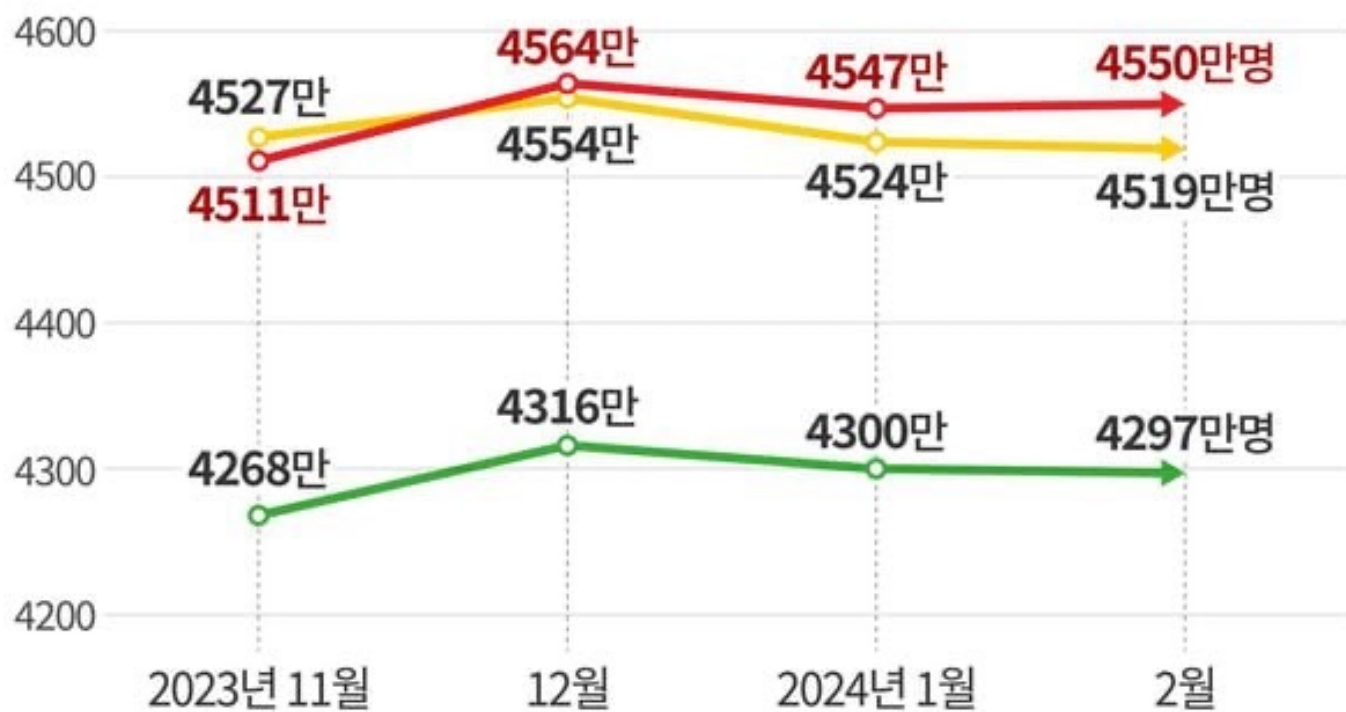
02 Count Min Sketch

03 HyperLogLog

03 HyperLogLog

- 2007년 Philippe Flajolet 개발
- 고유한 항목의 개수(카디널리티)를 추정하는 알고리즘

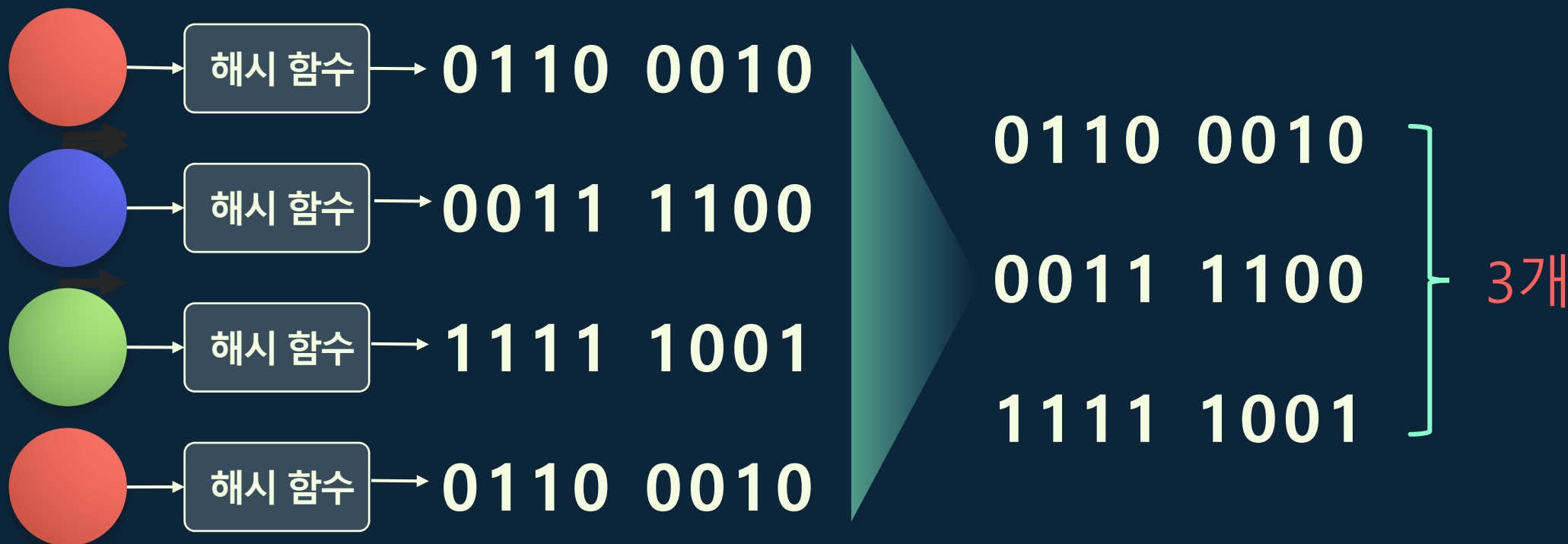
유튜브·카카오톡·네이버 MAU 추이 월간 활성 이용자수



자료=모바일인덱스

해시

고유한 요소의 수는?

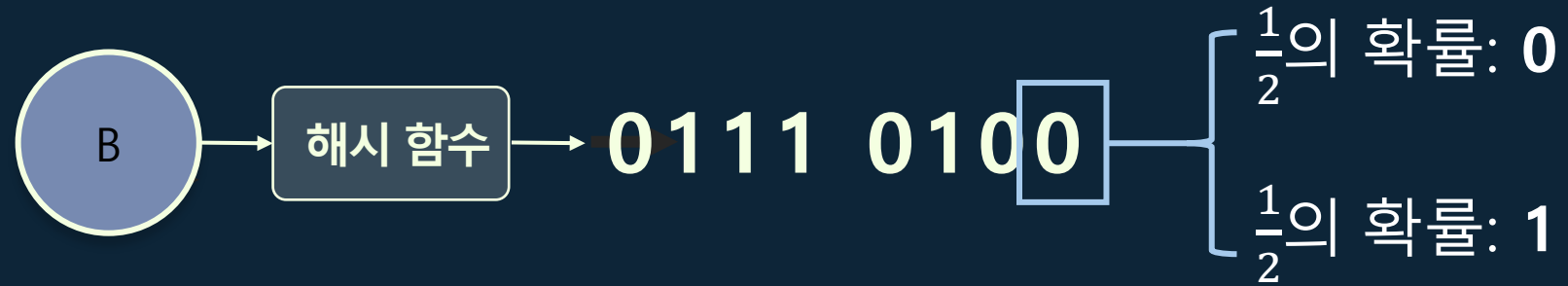
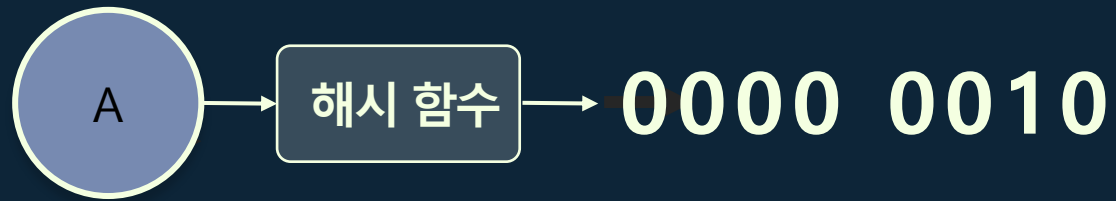


해시

무작위성을 이용하자

해시

무작위성을 이용하자



동전의 조합을 기록하기

- 앞면이 나오면 다시 던짐
- 뒷면이 나오면 정지



하나의 요소

동전의 조합을 기록하기



운이 좋아야 한다

동전의 조합을 기록하기

- 앞면이 나오면 다시 던짐
- 뒷면이 나오면 정지



$$\frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} = \frac{1}{16}$$

16가지의 요소



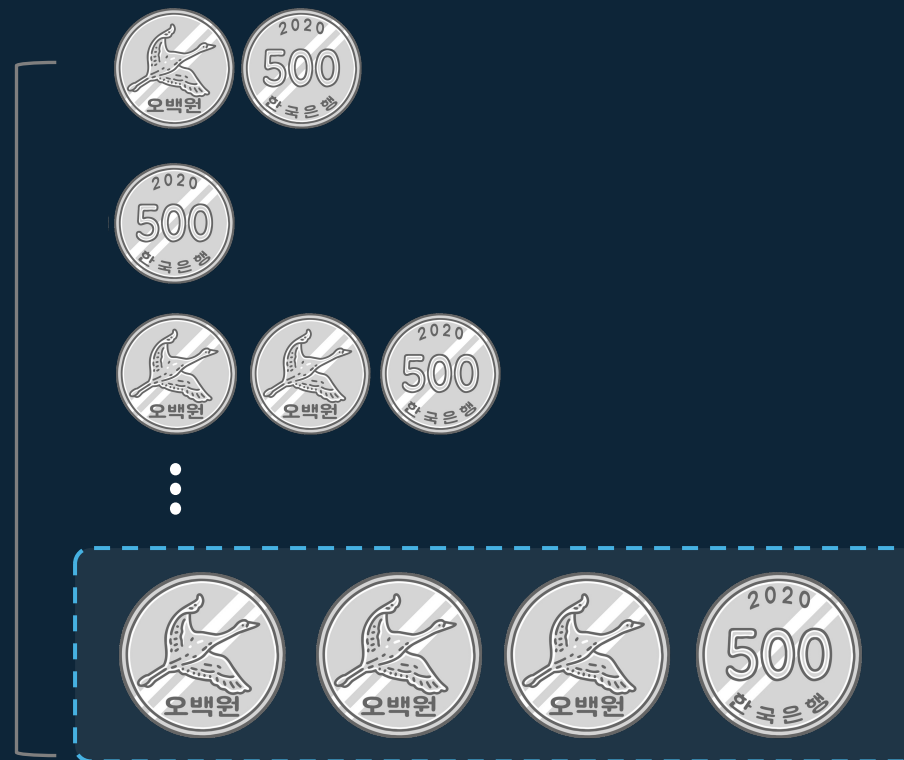
동전 던지기

연속된 앞면의 개수 L 동전을 던진 횟수 2^{L+1}



앞면의 개수 L

2^{L+1} 던졌다



2^{L+1} 요소

다시 이진 문자열로



동전의 앞면 = 0

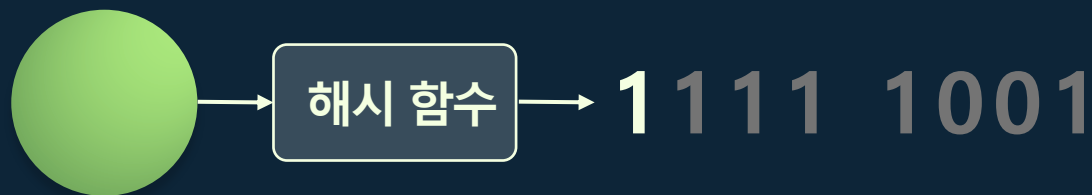
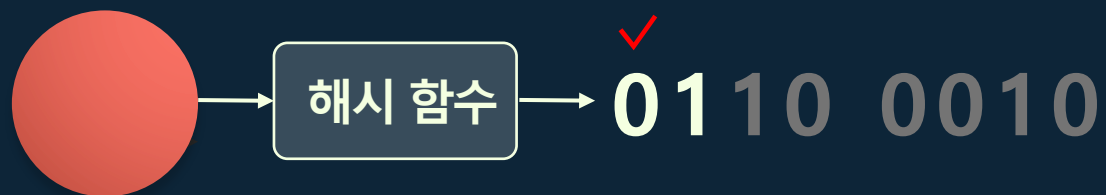


동전의 뒷면 = 1



다시 이진 문자열로

연속된 0이 긴 값을 이용하여 고유 개수 추정



...

가장 긴 0의 개수: $L = 2$

$$2^{L+1} = 2^3 = 8$$

다시 이진 문자열로

0110 0010

0011 1100

1111 1001

...

0111 1111

0000 1000

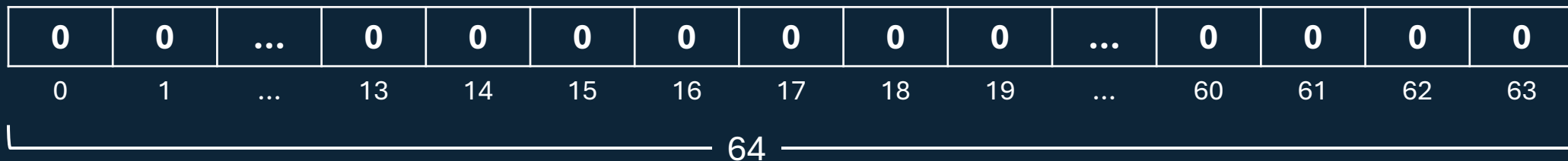
가장 긴 연속된 0 개수 = L

고유한 개수 $\Rightarrow 2^{L+1}$

저장 공간은?

$$\text{고유 항목 수} = 18,446,744,073,709,551,616 = 2^{64}$$

$$\Downarrow \log_2 2^{64} = 64$$



$$\Downarrow \log_2 64 = 6$$

연속된 0의 개수 저장을 위해 **6 bit** 필요

2의 거듭 제곱 값 만을 가진다.



매우 운이 좋은 경우

한번에 가장 긴 0을 가진 이진 배열을 가진다면?

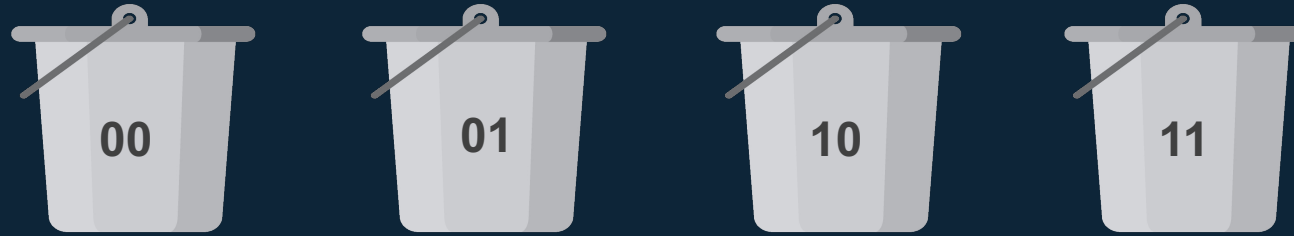
진화하기

- LogLog
- SuperLogLog
- HyperLogLog

- 나누어 처리하여 오류를 줄인다.



- 버킷의 L 을 평균하여 고유한 문자열을 추정한다.



L 의 평균

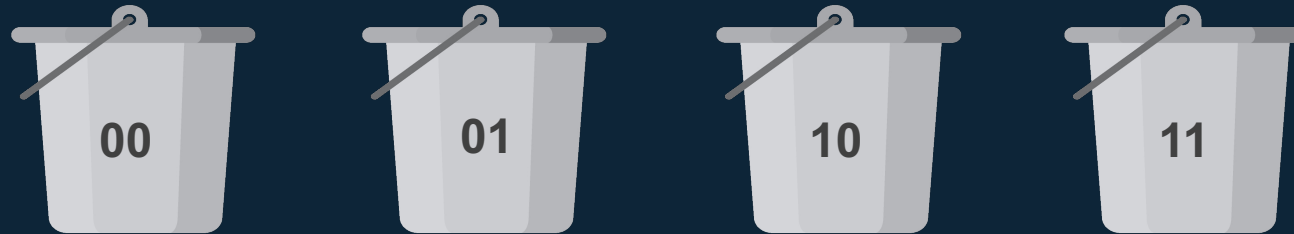
L_1

L_2

L_3

L_4

- 버킷의 L 을 평균하여 고유한 문자열을 추정한다.



L 의 평균

L_1

L_2

L_3

L_4

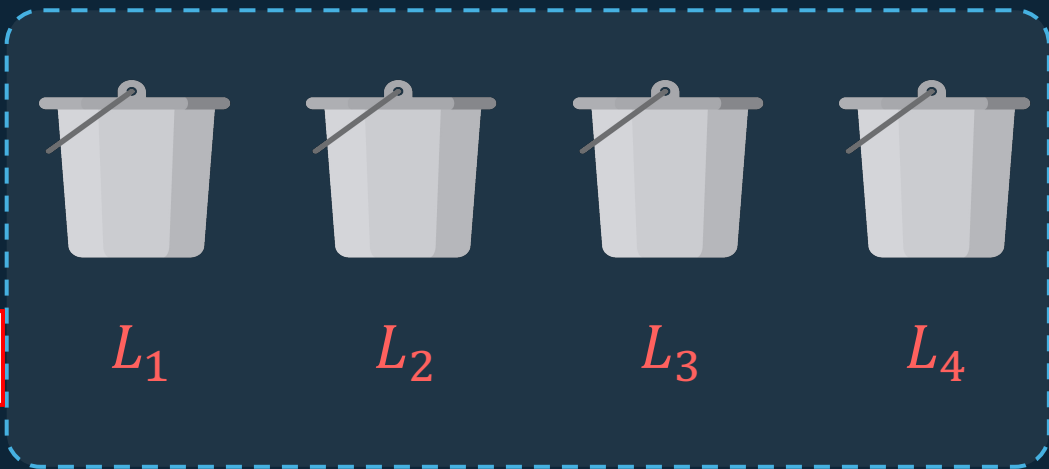
고유문자열의 개수 추정 : $0.721 \times 4 \times 2^{\frac{(L_1+L_2+L_3+L_4)}{4}}$

편향 제거 상수

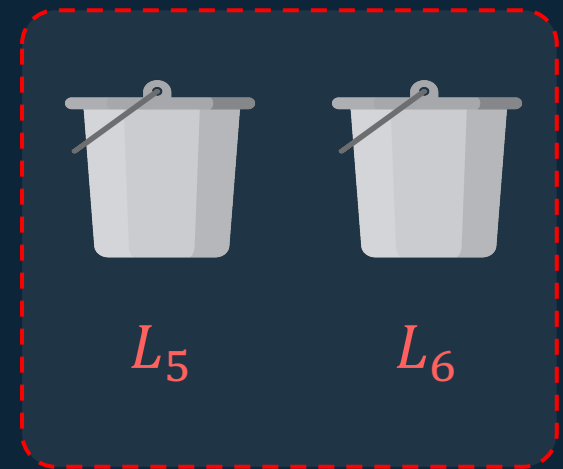
버킷의 고유한 요소의 수 추정

- 카운터 평균 수행 시 가장 큰 30%를 제거 후 평균을 구한다.

가장 긴 0의 개수



70% 사용하여 평균 수행



30% 제거

- 조화평균을 이용하여 고유한 문자열의 수 추정

$$0.673 \times 4 \times \frac{4}{\left(\frac{1}{2}\right)^{L_1} + \left(\frac{1}{2}\right)^{L_2} + \left(\frac{1}{2}\right)^{L_3} + \left(\frac{1}{2}\right)^{L_4}}$$

- 1% 미만의 오차
- **12KB** 의 메모리로 10억 건의 고유 값 추정

HyperLogLog 사용

1%의 미만의 오차 확인

```
long currentTimestamp = getCurrentTimestamp();
HyperLogLogOperations<String, String> hyperLogLogOperations
    = redisTemplate.opsForHyperLogLog();

int limit = 10000000;
IntStream.rangeClosed(1, limit).forEach(i -> {
    hyperLogLogOperations.add(getHourlyKey(currentTimestamp), "user" + i);
});

long currentCardinality = hyperLogLogOperations.size(getHourlyKey(currentTimestamp));
// 100000 * 0.01 = 1000 (1% error rate)
assertThat(currentCardinality).isBetween((long) (limit - limit * (0.01)), (long) (limit + limit * (0.01)));
```

1천만개의 사용자 등록

```
127.0.0.1:6379> pfcount test:hyperloglog:hourly:202405040000
```

```
(integer) 10060588 60588개의 오차 발생(1%미만)
```

```
127.0.0.1:6379> memory usage test:hyperloglog:hourly:202405040000
```

```
(integer) 12392 12KB 저장 공간 사용
```

HyperLogLog의 병합

```
long currentTimestamp = getCurrentTimestamp();
HyperLogLogOperations<String, String> hyperLogLogOperations = redisTemplate.opsForHyperLogLog();
int limit = 500000;
String[] candidateKeys = getMergeCandidateKeys(currentTimestamp);
IntStream.rangeClosed(0, 23).forEach(i -> {
    IntStream.rangeClosed(1, limit).forEach(j -> {
        int idx = (i * limit / 2) + j;
        hyperLogLogOperations.add(getHourlyKey(currentTimestamp + i * 3600000L), "user" + idx);
    });
});
long mergeCardinality = hyperLogLogOperations.union(getDailyKey(currentTimestamp), candidateKeys);
assertThat(mergeCardinality).isEqualTo(hyperLogLogOperations.size(candidateKeys));
```

병합 작업

HyperLogLog의 직렬화

```
HyperLogLogOperations<String, String> hyperLogLogOperations = redisTemplate.opsForHyperLogLog();
ValueOperations<String, String> valueOperations = redisTemplate.opsForValue();
long currentTimestamp = getCurrentTimestamp();
byte[] dump = redisTemplate.dump(getDailyKey(currentTimestamp));
valueOperations.set("test:dump", Base64.encodeBase64String(dump));
String s = valueOperations.get("test:dump");
redisTemplate.restore(getDailyKey(currentTimestamp) + ":restore",
    Base64.decodeBase64(s), 0, TimeUnit.MILLISECONDS, false);
long restoreCardinality = hyperLogLogOperations.size(getDailyKey(currentTimestamp) + ":restore");
assertThat(restoreCardinality).isEqualTo(hyperLogLogOperations.size(getDailyKey(currentTimestamp)));
```

Hyperloglog 직렬화

Hyperloglog restore

값 차이 없음

```
127.0.0.1:6379> pfcount test:hyperloglog:daily:20240504
(integer) 1207314
```

```
127.0.0.1:6379> pfcount test:hyperloglog:daily:20240504:restore
(integer) 1207314
```

03 HyperLogLog 정리

고유한 값의 개수를 추정하기 위한 확률적인 자료구조
실시간으로 추정 가능(병합, 직렬화)

주의 사항

- 1% 미만의 추정 오차 발생 가능 하다.

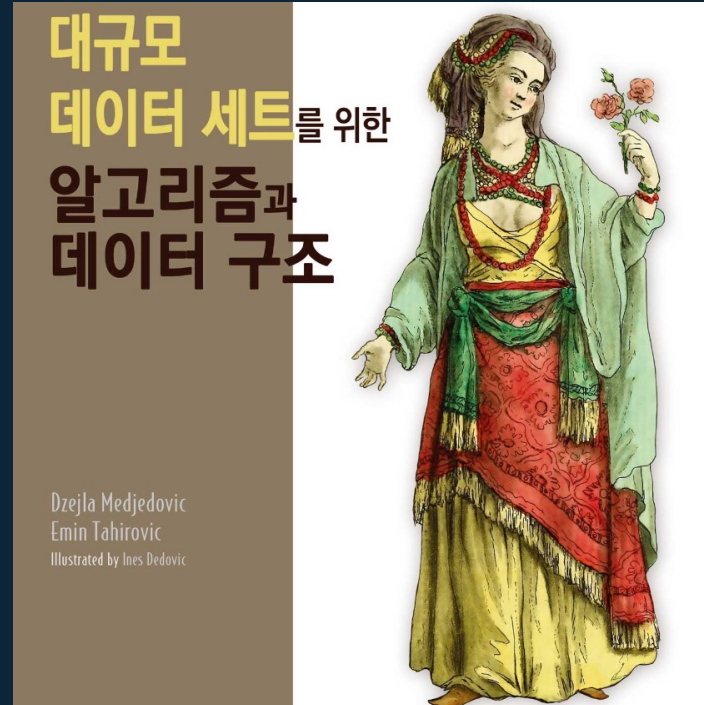
마무리

스케치 알고리즘은 작고 빠르게 거대한 데이터를 처리 할 수 있습니다.

해시를 활용하는 아이디어를 확인 할 수 있었습니다.

수학적 알고리즘의 이해 필요

마무리



<https://ebook-product.kyobobook.co.kr/dig/epd/ebook/E000005388724>

The End