1 해시 테이블의 충돌 해결

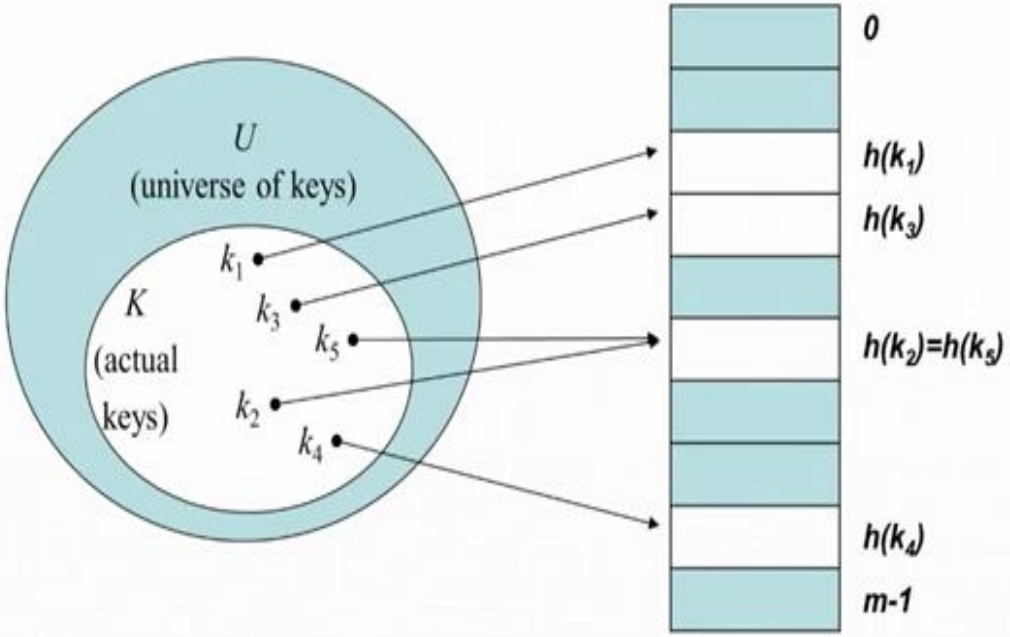
1 해시 테이블의 충돌 해결

1 충돌(Collision)

- 해시 함수를 통해 만들어진 해시 주소가 중복되면 데이터 값이 충돌함
- 두 개의 키가 동일한 해시값을 갖는 경우 **'충돌이 발생했다'** 라고 함
- 서로 다른 두 키 k1과 k2에 대해서 h(k1)=h(k2)인 상황
- 충돌이 발생할 경우 대처 방법이 필요
- 충돌을 해결하는 방법은 해시 테이블에서 가장 중요한 문제
- 충돌을 피하는 방법은 충돌이 적은 좋은 해시 함수를 사용하는 것

1 해시 테이블의 충돌 해결

1 충돌(Collision)



※출처: http://new93helloworld.tistory.com/146

1 해시 테이블의 충돌 해결

1 충돌(Collision)

- 해시 함수로 충돌 현상이 발생할 때 한 키값에 대한 다른 주소를 결정하기 위해 이용하는 방법이 필요함
  ➔ 충돌 해결 방법

- 대표적인 충돌 해결 방법
  - 체이닝(Chaining)
  - 개방 주소 방법(Open Addressing)

2 체이닝

2 체이닝

1 충돌 해결Collision Resolution)

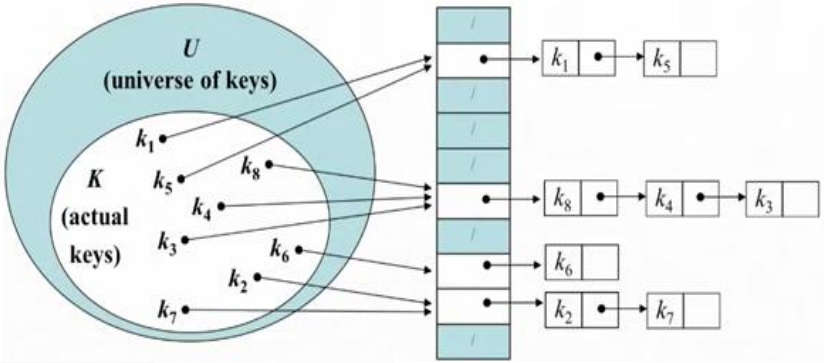
- 체이닝
  - 같은 주소로 해싱되는 원소를 모두 하나의 연결 리스트(Linked List)로 관리함
  - 추가적인 연결 리스트 필요

- 개방 주소 방법
  - 충돌이 일어나더라도 어떻게든 주어진 테이블 공간에서 해결함
  - 추가적인 공간이 필요하지 않음

2 체이닝(Chaining)

- 동일한 장소로 해싱 된 모든 키들을 하나의 연결 리스트로 저장

- 중복된 키 값이 있을 경우 해당 슬롯을 연결 리스트로 저장



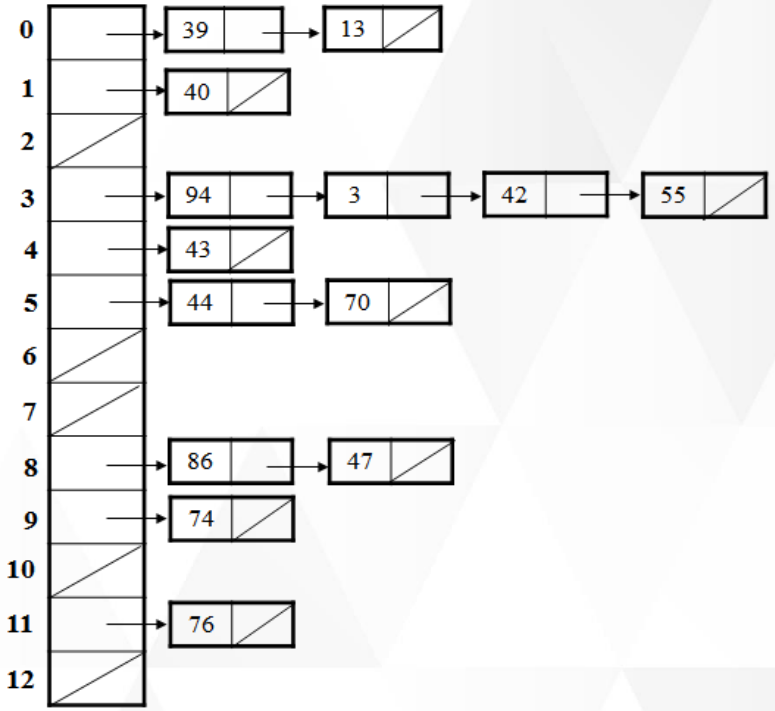
※출처: http://new93helloworld.tistory.com/146

2 체이닝(Chaining)

체이닝의 예

- 크기가 13인 해시 테이블에 원소들이 55, 13, 42, 70, 43, 44, 3, 94, 47, 74, 39, 86, 76, 40 순서로 저장된다고 할 때 체이닝으로 저장한 예

  - 총 13개 중에 5개는 비었고 8개는 원소가 한 개 이상 있는데 같은 주소로 들어온 원소들은 해당 헤더 아래 연결 리스트로 연결됨
  - 임의의 원소를 연결 리스트에 삽입할 때는 해당 리스트의 맨 앞에 삽입함



2 체이닝(Chaining)

체이닝의 장점

- 연결 리스트만 사용하면 되므로 복잡한 계산식을 사용할 필요가 개방 주소 방법에 비해 적음
- 적재율이 1을 넘어도 사용할 수 있음

2 체이닝(Chaining)

체이닝의 단점

- 각 연결 리스트마다 헤드를 하나씩 두어야 하고 연결 리스트를 만들때 각 원소마다 연결 공간이 필요함
- 적재율이 높지 않을 때는(예, 1/2 이하) 개방 주소 방법이 더 매력적임

3 개방 주소 방법

3 개방 주소 방법

1 개방 주소 방법(Open Addressing)

- 체이닝의 경우 버켓이 꽉 차더라도 연결 리스트로 계속 늘려가므로 데이터의 주소값은 바뀌지 않음 (Closed Addressing)
- 개방 주소 방법은 해시 충돌이 일어나면 다른 버켓에 데이터를 삽입하는 방식
- 빈자리가 생길 때까지 해시값을 계속 만들어냄
- 모든 원소가 반드시 자신의 해시 값과 일치하는 주소에 저장된다는 보장은 없음

3 개방 주소 방법

1 개방 주소 방법(Open Addressing)

- 충돌이 생기면 정해진 규칙에 따라 다음 자리를 찾는데 빈 자리를 찾을 때까지 계속 찾음
- $h_0(x)$, $h_1(x)$, $h_2(x)$, $h_3(x)$, …

3 개방 주소 방법

2 개방 주소 방법의 장점

- 체이닝처럼 포인터가 필요 없고 지정한 메모리 외 추가적인 저장 공간도 필요 없음
- 삽입, 삭제 시 오버헤드가 적음
- 저장할 데이터가 적을 때 더 유리함

3 개방 주소 방법

3 개방 주소 방법 3가지

- 다음 주소를 결정하는 중요한 3가지 방법

선형 조사

이차원 조사

더블 해싱

3 개방 주소 방법

3 개방 주소 방법 3가지

① 선형 조사(Linear Probing)

- 가장 간단한 충돌 해결 방법으로
  충돌이 일어난 바로 뒷자리를 보는 것
- 해시 충돌 시 다음 버켓, 혹은 몇 개 건너뛰어
  데이터를 삽입함
- 다음 자리를 계산하다가 테이블의 경계를
  넘어갈 경우에는 맨 앞으로 감

$$h_i(x) = (h(x) + i) \bmod m \quad (i=0, 1, 2, \cdots)$$

3 개방 주소 방법 3가지

① 선형 조사(Linear Probing) 예

- 크기가 13인 해시 테이블에 10개의 원소가 25, 13, 16, 15, 7, 28, 31, 20, 1, 38 순으로 삽입할 때 3개의 원소에 대해 충돌발생

0	13
1	
2	15
3	16
4	28
5	
6	
7	7
8	
9	
10	
11	
12	25

0	13
1	
2	15
3	16
4	28
5	31
6	
7	7
8	20
9	
10	
11	
12	25

0	13
1	1
2	15
3	16
4	28
5	31
6	38
7	7
8	20
9	
10	
11	
12	25

3 개방 주소 방법 3가지

① 선형 조사(Linear Probing) 문제점

- 특정 영역에 원소가 몰릴 때는 치명적으로 성능이 떨어짐
- 선형 조사는 1차 군집에 취약함

1차 군집:
특정 영역에 원소가 몰리는 현상

[1차 군집의 예]

0	
1	
2	15
3	16
4	28
5	31
6	44
7	
8	
9	
10	
11	37
12	

3 개방 주소 방법 3가지

② 이차원 조사(Quadratic Probing)

- 충돌시 바로 뒷자리를 보는 대신에 보폭을 이차 함수로 넓혀가면서 찾음
- 해시 충돌 시 제곱만큼 건너뛴 버켓에 데이터를 삽입함
- i번째 해시 함수를 h(x)에서 i^2 만큼 떨어진 자리로 삼을 수 있음
- h(x), h(x)+1, h(x)+4, h(x)+9, h(x)+16, … 과 같이 볼 수 있음

$$h_i(x) = (h(x) + c_1 i^2 + c_2 i) \bmod m \quad (i=0, 1, 2, \cdots)$$

3 개방 주소 방법 3가지

② 이차원 조사(Quadratic Probing)

예

- 입력 순서
  15, 18, 43, 37, 45, 30

$$h_i(x) = (h(x) + i^2) \bmod 13$$

0	
1	
2	15
3	
4	43
5	18
6	45
7	
8	30
9	
10	
11	37
12	

3 개방 주소 방법 3가지

② 이차원 조사(Quadratic Probing)

예

- 입력 순서
  15, 21, 28, 41, 67, 54
- 21을 제외한 15, 28, 41, 67, 54 는 모두 주소 2로 해싱된 원소
- 한번 충돌이 발생할 때마다 i^2 (1, 4, 9, 16, …)씩 계속 증가시키며 자리를 찾음

0	
1	
2	15
3	28
4	
5	54
6	41
7	
8	21
9	
10	
11	67
12	

[2차 군집의 예]

- 이차원 조사는 2차 군집에 취약함

3 개방 주소 방법 3가지

② 이차원 조사(Quadratic Probing) 예

- 2차 군집
  - 여러 개의 원소가 동일한 초기 해시 함수값을 갖게 되면 모두 같은 순서로 조사를 할 수밖에 없어 비효율적임
  - 보폭은 점점 넓어지지만 최초의 해시 값이 같은 원소들은 이 때문에 이득을 보지 못함
  - 이차원 조사는 2차 군집에 취약함

3 개방 주소 방법 3가지

③ 더블 해싱(Double Hashing)

- 해시 충돌 시 다른 해시 함수를 한번 더 적용한 결과를 이용함
- 두 개의 함수를 사용
- 충돌이 생겨 다음에 볼 주소를 계산할 때 두번째 해시 함수 값만큼씩 점프함

3 개방 주소 방법 3가지

③ 더블 해싱(Double Hashing)

- 첫번째 해시값이 같더라도 두번째 함수값이 같을 확률은 매우 작으므로 서로 다른 보폭으로 점프하게 됨 (→2차 군집이 발생하지 않음)

$$h_i(x) = (h(x) + i\,f(x)) \bmod m \quad (i=0, 1, 2, \cdots)$$

  - h(x)와 f(x)는 서로 다른 해시 함수

3 개방 주소 방법 3가지

③ 더블 해싱(Double Hashing)

예

- 입력 순서
  15, 19, 28, 41, 67 임
- 15, 28, 41, 67이 모두
  주소 2로 해싱됨(h(x)=2)
- 두번째 함수는 모두 달라
  두번째 조사하는 주소가
  모두 달라짐

0	
1	
2	15
3	67
4	
5	
6	19
7	
8	28
9	
10	41
11	
12	

$h_0(15) = h_0(28) = h_0(41) = h_0(67) = 2$

$h_1(67) = 3$

$h_1(28) = 8$

$h_1(41) = 10$

$h(x) = x \bmod 13$

$f(x) = x \bmod 11$

$h_i(x) = (h(x) + i f(x)) \bmod 13$

3 개방 주소 방법

4 해시 테이블에서의 검색 시간

- 적재율 α
  - 해시 테이블 전체에서 얼마나 원소가 차 있는지를 나타내는 수치
  - 해시 테이블에 n개의 원소가 저장되어 있다면 $\alpha = n/m$ 임
- 해시 테이블에서의 검색 효율은 적재율과 밀접한 관련이 있음

3 개방 주소 방법

4 해시 테이블에서의 검색 시간

- 개방 주소 방법은 테이블에 주어진 공간만 사용할 수 있으므로 적재율이 1을 넘을 수 없음

- 적재율이 높아지면 효율이 급격히 떨어지므로 적당한 임계점을 설정한 후 그것을 넘으면 해시 테이블의 크기를 대략 2배로 키우고 모든 원소를 다시 해싱하는 것이 일반적임

3 개방 주소 방법

5 해시 테이블에서 삭제시 조심할 것

- 개방 주소 방법에서 조심할 것은 원소를 삭제했을 때임

0	13
1	1
2	15
3	16
4	28
5	31
6	38
7	7
8	20
9	
10	
11	
12	25

(a) 원소 1이 삭제된다

0	13
1	
2	15
3	16
4	28
5	31
6	38
7	7
8	20
9	
10	
11	
12	25

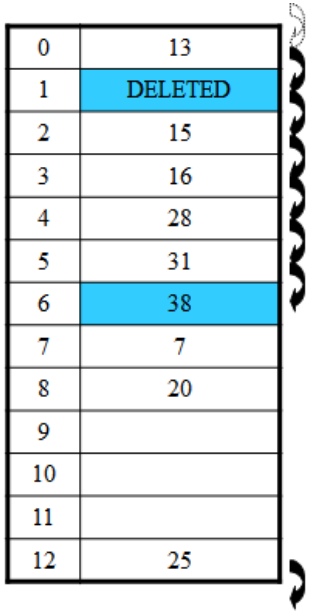
(b) 38 검색, 문제발생

- 원소 1을 삭제후 38 검색시 주소1에 빈 자리를 발견하여 38은 없다고 판단할 수 있음

3 개방 주소 방법

5 해시 테이블에서 삭제시 조심할 것

- 삭제시 원래 원소가 있던 자리였음을 표시함으로 해결



0	13
1	DELETED
2	15
3	16
4	28
5	31
6	38
7	7
8	20
9	
10	
11	
12	25

(c) 표식을 해두면 문제없다

- 주소1 자리에 DELETED 라는 상수 값을 저장하여 삭제된 자리라는 표시를 함