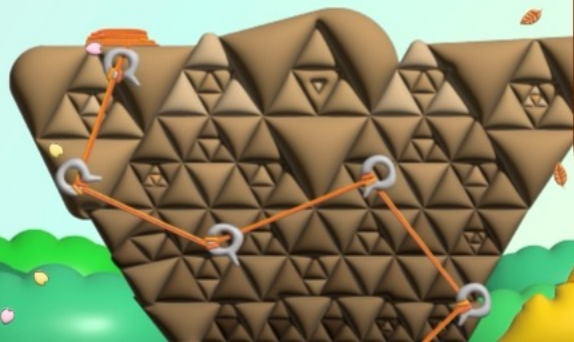


파이썬으로 배우는 데이터 구조



한동대학교
전산전자공학부
김영섭 교수



학습 목표

해시 테이블을 구현할 때 피할 수 없는 충돌을
해결하는 방법들을 이해하고 적용할 수 있다

Data Structures in Python

Chapter 6

- Hash Table
- **Collision Resolutions**
- Double Hashing & Rehashing
- Hash Implementation

Collision Resolution Exercise - Linear Probing

Insert sequence: 89, 18, 49, 58, 69

$$h(k) = k \% 10$$

	Empty Table	After 89	After 18	After 49	After 58	After 69
0				49	49	49
1					58	58
2						
3						
4						
5						
6						
7						
8			18	18	18	18
9		89	89	89	89	89
Unsucessful no. of probes		0	0	1	3	3

probe sequence

For example, linear probing for 58
 $h_0(58) = (h(58) + f(0)) \% 10$
 $= (8 + 0) \% 10 = 8$ (collision)
 $h_1(58) = (h(58) + 1) \% 10 = 9$ (collision)
 $h_2(58) = (h(58) + 2) \% 10 = 0$ (collision)
 $h_3(58) = (h(58) + 3) \% 10 = 1$

Complete the linear probing for 69
 $h_0(69) =$

Collision Resolution Exercise - Quadratic Probing이차조사법

Insert sequence: 89, 18, 49, 58, 69

$$h(k) = k \% 10$$

	Empty Table	After 89	After 18	After 49	After 58	After 69
0				49	49	49
1						
2					58	58
3						
4						
5						
6						
7						
8			18	18	18	18
9		89	89	89	89	89
Unsucessful no. of probes		0	0	1	2	2

probe sequence

For example, quadratic probing for 58

$$h_0(58) = (h(58) + f(0)) \% 10 = (8 + 0) \% 10 = 8 \text{ (collision)}$$

$$h_1(58) = (h(58) + 1) \% 10 = 9 \text{ (collision)}$$

$$h_2(58) = (h(58) + 4) \% 10 = 2$$

Complete quadratic probing for 69

$$h_0(69) =$$

Collision Resolution Exercise - Double Hashing 이중해싱법

Insert sequence: 89, 18, 49, 58, 69, 23

$$h(x) = x \% 10$$

After
23

$h'(x) = R - (x \% R)$
R is prime number less than TableSize

0							
1							
2							
3							
4							
5							
6			49	49	49	49	
7							
8		18	18	18	18	18	
9		89	89	89	89	89	

$h_0(49) = (h(49) + f(0)) \% 10 = 9$ (collision)
 $h_1(49) = (h(49) + 1 * (7 - 49 \% 7)) \% 10 = 6$

$h_0(58) =$
 $h_1(58) =$

$h_0(69) =$
 $h_1(69) =$

$h_0(23) =$
 $h_1(23) =$
:

Unsucessful
no. of probes

0 0 1 1 1

Rehashing - Exercise

Rehash the following table into a new hash table below using the hash function:

- Use $\text{hash}(\text{key}) = \text{key} \% 13$ and **quadratic probing** to resolve the collisions.
- Show your computation, collision and resolution.
- Compute the load factors before and after rehashing.

0	1	2	3	4	5	6
56	43	30	None	None	26	13

0	1	2	3	4	5	6	7	8	9	10	11	12

학습 정리

- 1) 충돌 해결 방법은 크게 Separate chaining과 Open addressing 두 가지가 있다
- 2) 적재율(λ)이 0.5 보다 작아야 $O(1)$ 을 유지할 수 있다
- 3) Linear probing > Quadratic probing > Double hashing
순서로 충돌 횟수가 줄어든다

파이썬으로 배우는 데이터 구조

수고했습니다
곧 다음 시간에
다시 뵙겠습니다

