Lab 11: Hashing

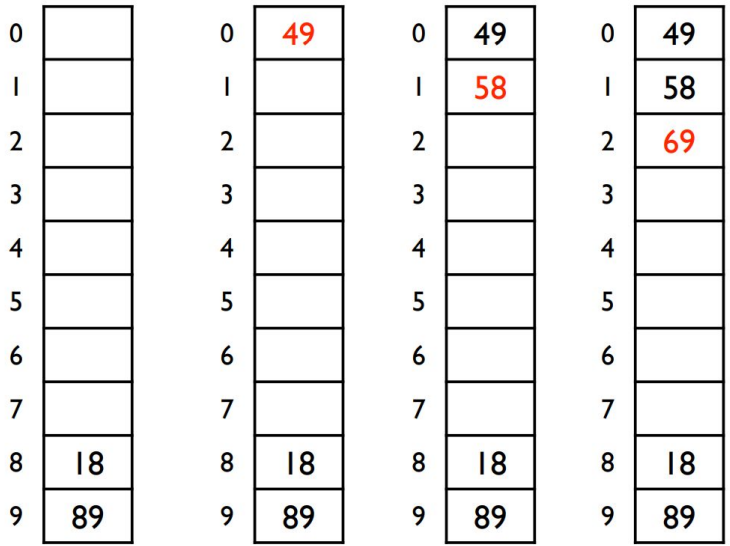
Data Structure 2023

Hashing (Open Addressing)

- All the keys are stored in the table without pointers
- If a collision occurs, alternative cells are tried until an empty cell is found
- Try h0(key), h1(key), h2(key), . . .
    - where hi(key) = (Hash(key) + F(i)) mod m
        - **i : iteration, m : size of Hash Table, F(i) : collision resolution strategy**
    - Linear probing: F(i) is a linear function, F(i) = i
        - for example, h1(key) = (Hash(key) +1), h2(key) = (Hash(key) +2), . . .
    - Quadratic probing: F(i) is a quadratic function, F(i) = i^2
        - for example, h1(key) = (Hash(key) +1), h2(key) = (Hash(key) +4), . . .
- Use special value Del to determine which entries have keys & which don't.

Hashing: Linear Probing

- F(i) is a linear function. h1(key) = (Hash(key) + i) mod m
  - F(i) = i
  - Insert keys: 89, 18, 49, 58, 69



ex)
- F(49) = [(49 % 10) + 1] mod 10 = 0
- F(58) = [(58 % 10) + 2] mod 10 = 1
- F(69) = [(58 % 10) + 3] mod 10 = 2

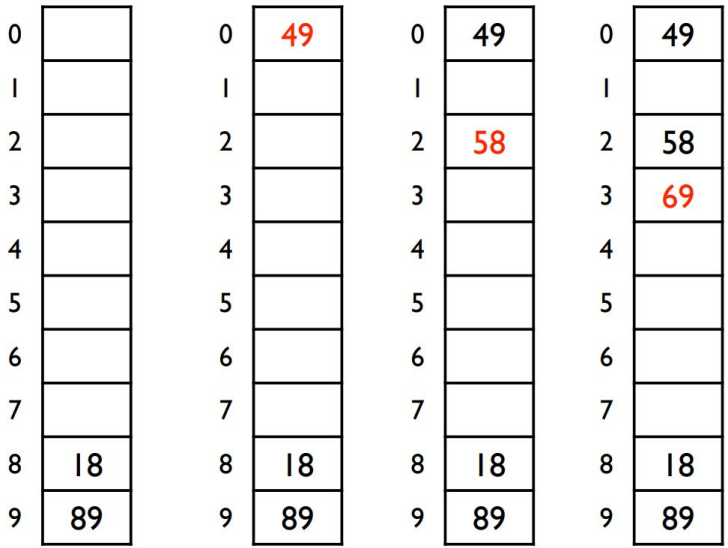
  49 => 89 Collide 1 time

  58 => 18, 49 Collide 2 times

  69 => 89, 49, 58 Collide 3 times

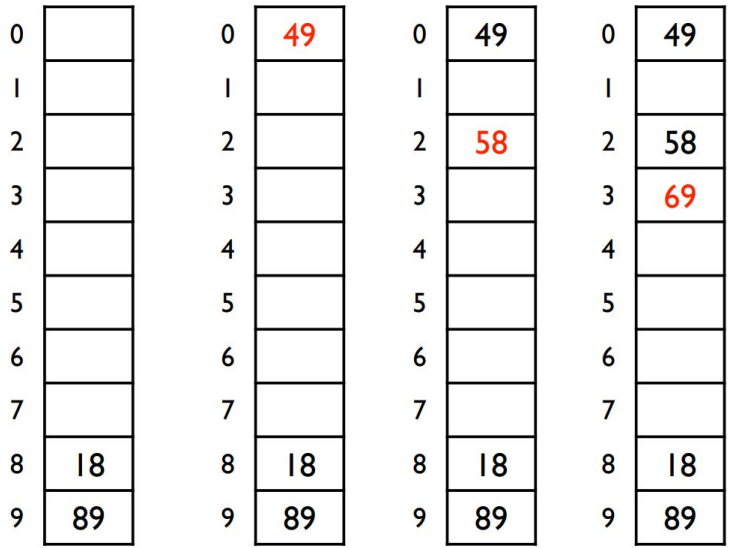
Hashing ADT

- F(i) is a quadratic function. hi(key) = (Hash(key) + i^2) mod m
  - F(i) = i^2,
  - Insert keys: 89, 18, 49, 58, 69



Hashing ADT

- F(i) is a quadratic function. hi(key) = (Hash(key) + i^2) mod m
  - F(i) = i^2,
  - Insert keys: 89, 18, 49, 58, 69



Hashing ADT

- **HashTable createTable(int TableSize)**
  - **Create a hash table with size TableSize.**
- **void Insert(HashTable H, ElementType Key, int solution)**
  - **Insert value to Hash Table using the solution.**
- **void Delete(HashTable H, ElementType Key, int solution)**
  - **Delete the value from Hash Table.**
- **int Find(HashTable H, ElementType Key, int solution)**
  - **Find the value from Hash Table.**
- **void printTable(HashTable H)**
  - **Print all values of Hash Table.**
- **void deleteTable(HashTable H)**
  - **Delete a Hash Table.**

Topological Sorting ADT

Structure

```
typedef int ElementType;
typedef ElementType List;
typedef struct HashTbl* HashTable;
typedef struct HashTbl{
    int TableSize;
    List *TheLists;
}HashTbl;
```

Function

```
HashTable createTable(int TableSize);
void Insert(HashTable H, ElementType Key, int solution);
void Delete(HashTable H, ElementType Key, int solution);
int Find(HashTable H, ElementType Key, int solution);
void printTable(HashTable H);
void deleteTable(HashTable H);
```

Input & Output Example

```
(base) oknkc8@DESKTOP-NT9MABE:~/CSE2010/lab13$ cat input1.txt
Linear
11
i 1
i 11
i 4
i 15
i 22
f 64
i 22
i 9
i 18
i 77
i 16
d 4
d 18
d 85
p(base) oknkc8@DESKTOP-NT9MABE:~/CSE2010/lab13$ ./lab13_solution input1.txt output1.txt
(base) oknkc8@DESKTOP-NT9MABE:~/CSE2010/lab13$ cat output1.txt
Insert 1 into address 1
Insert 11 into address 0
Insert 4 into address 4
Insert 15 into address 5
Insert 22 into address 2
64 is not in the table
Collision: 22 is already exists at address 2
Insert 9 into address 9
Insert 18 into address 7
Insert 77 into address 3
Insert 16 into address 6
Delete 4
Delete 18
Deletion Error: 85 is not in the table
11 1 22 77 0 15 16 0 0 9 0
11 1 22 77 0 15 16 0 0 9 0
```

Assignment

- Due
  - **~ 2023.05.31(수) 23:59**
  - Last Commit 기준

- 자세한 내용은 과제 명세 PDF 파일 참고