# NYCU Instruction to Database Systems

## HW2: Extendible Hashing

TA 羅名志

2024/11/4

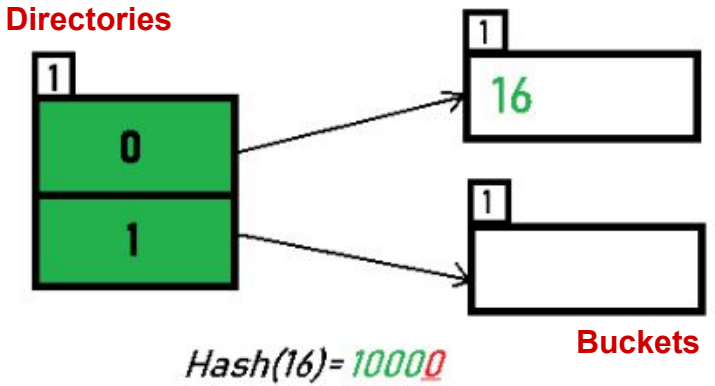
# Outline

- Extendible hashing introduction
- Requirement
- Reference

# Extendible Hashing

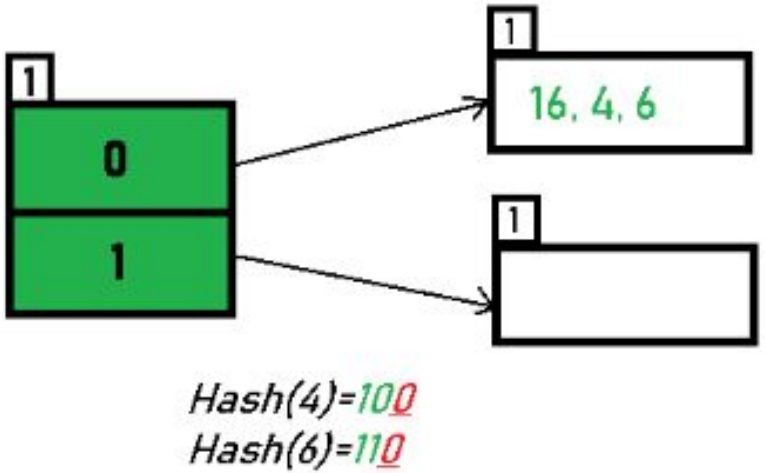
# Basic structure

- Initialization (example)
  - Global depth = 1, Local depth = 1
  - Bucket size = 3
  - Hash function: Suppose the global depth is X, then the hash function returns X LSBs



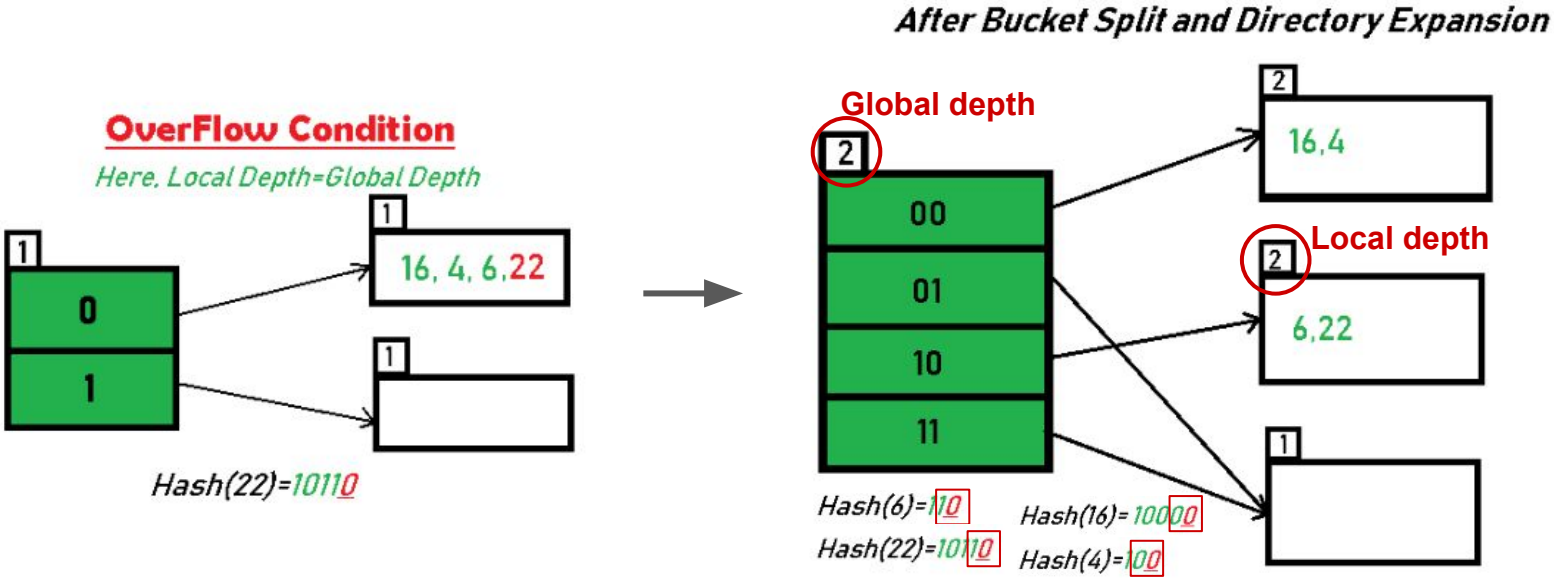
# Insertion

- Insert by hash index



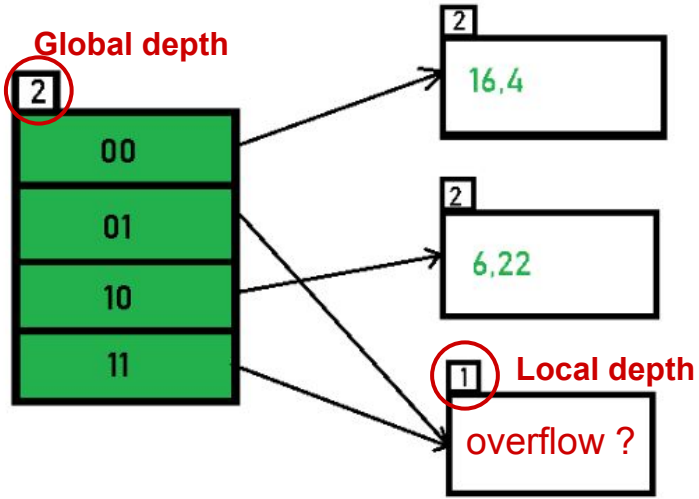
# Overflow & Extend

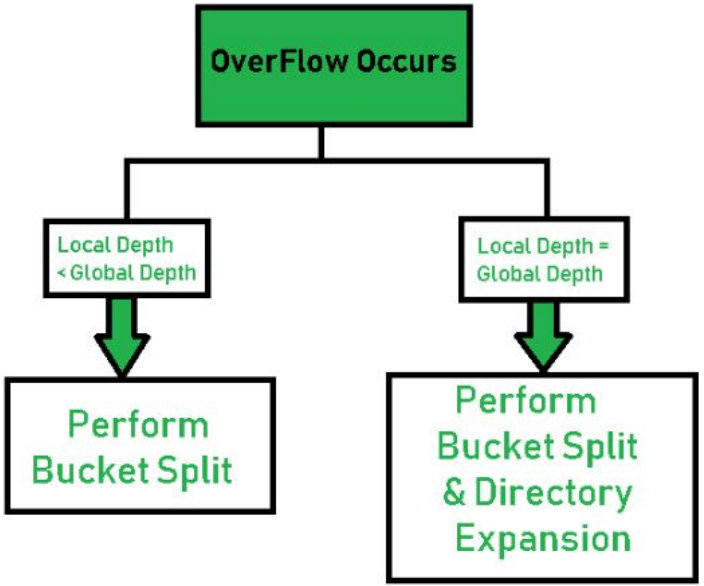
- When the number of key-value pair in the bucket is bigger than bucket size, it means **overflow**
- If overflow happened, it need to be extended



# Overflow & Extend

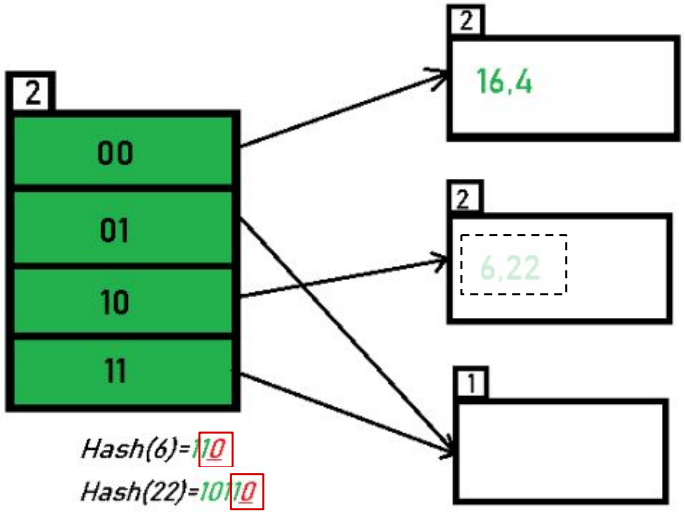
- If local depth is less than global depth, just split the bucket
- If local depth equal to global depth, need to first extend the directories



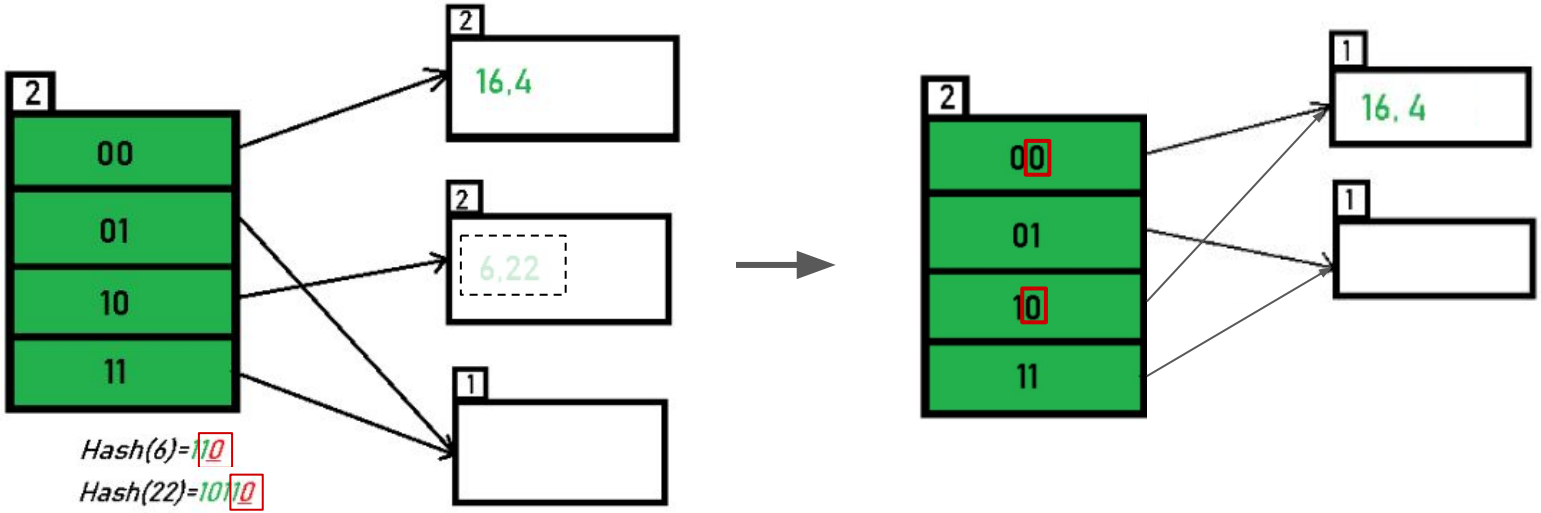
# Remove

- Similar to most of hashing methods



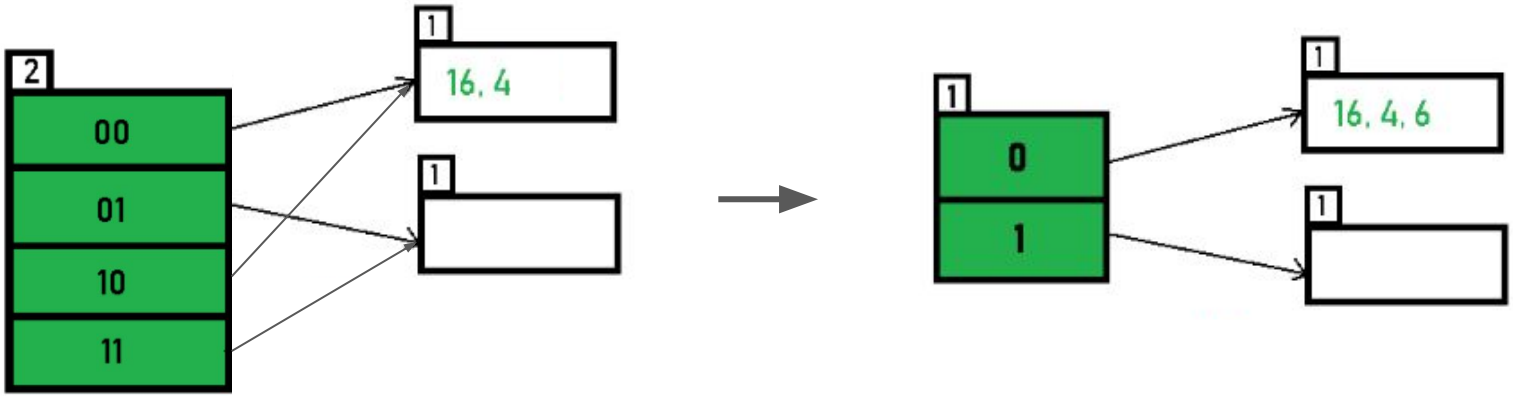
# Shrink

- Merge the bucket with the one with same hash index in (local depth - 1)



# Shrink

- Check the table size and maintain it in appropriate size
- If global depth larger than all local depth, the directory table should be cut in half



# Requirement

# In main.cpp and hash.cpp

- Initialization of the hash table with size = 2 ( global depth = 1 )
- Size of buckets should be 3
- The minimum size of global depth and local depth: 1/1
- Complete at least four function
  - constructor, key_query(), remove_query(), clear()

```
chrono::steady_clock::time_point start = chrono::steady_clock::now();

//Construct hash table
hash_table my_hash_table(1<<1, 3, num_rows, key, value);
chrono::steady_clock::time_point built_index = chrono::steady_clock::now();

//Query by key
my_hash_table.key_query(query_keys, "key_query_out1.txt");
chrono::steady_clock::time_point key_query1 = chrono::steady_clock::now();

//Remove by key
my_hash_table.remove_query(query_remove_keys);
chrono::steady_clock::time_point remove_query = chrono::steady_clock::now();

//Query by key
my_hash_table.key_query(query_keys, "key_query_out2.txt");
chrono::steady_clock::time_point key_query2 = chrono::steady_clock::now();
```

key

```
283311
612592
977126
829611
135735
1065439
18946
1286835
314940
1491295
```

value, local depth

```
940,20
88,19
402,19
790,19
-1,19
492,20
520,20
210,20
584,20
987,20
```

# Free for you

- Supplied hash.h, hash.cpp files are free for you to modify
- Please do not use the function like “map” or “unordered_map” to maintain the index without hash function
- The time to check the directory size for shrink can decide by yourself

# Grading

- Deadline: 11/25 (Mon.) 23:55:00
- Time performance: 20%

| Description | Score(%) |
|---|---|
| Correctness of "key_query_out1.txt" (global depth + value + local depth) | 5% + 5% + 15% |
| Correctness of "key_query_out2.txt" (global depth + value + local depth) | 5% + 5% + 15% |
| Correctness of extendible hash implementation | 20% |
| Time used to build index | 10% |
| Time used to process remove query | 10% |
| Report file | 10% |

```
3
1,3
2,3
3,3
```

# Reference


- Geeksforgeeks
- Extendible Hashing-A Fast Access Method for Dynamic Files(p.330)