NYUC DB

hw2 - Extendible hash requirement

TA 柯秉志

2023.04.07

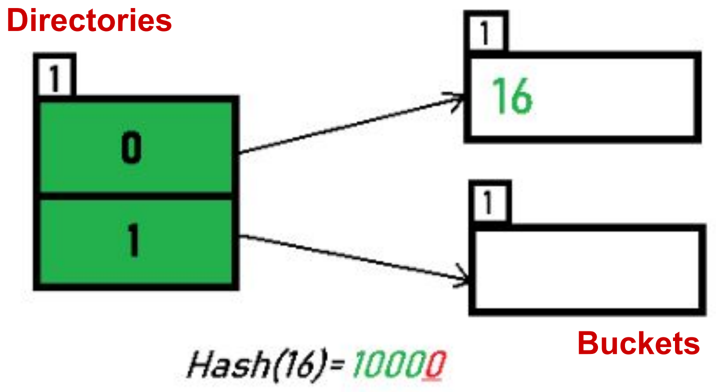
Outline

- Extendible hash introduction
- Limitation in this homework
- Reference

Extendible Hash

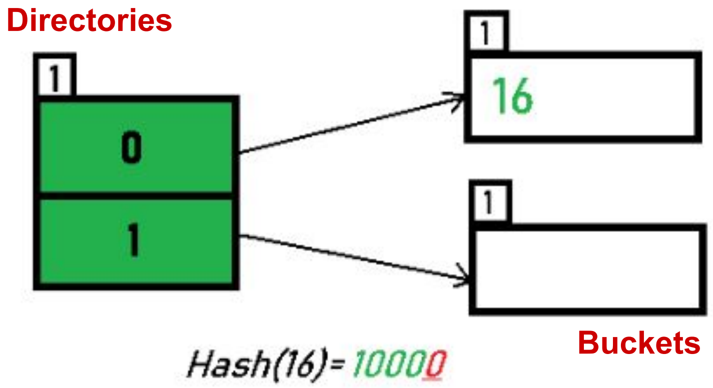
Basic structure

- initialization (example)
  - Directories size = 2
  - bucket size = 3
  - global depth = 1, local depth = 1



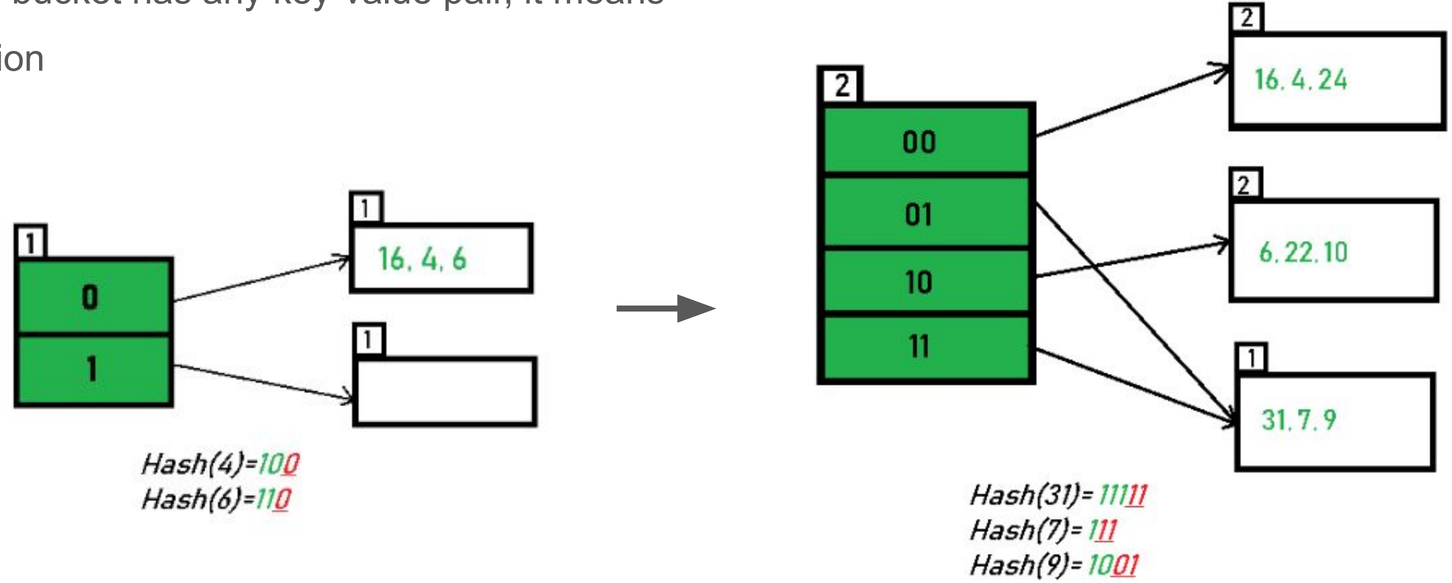
Basic structure

- hash function : indexed by bitwise with global depth
- directories : storing the pointer to bucket
- buckets : storing the data



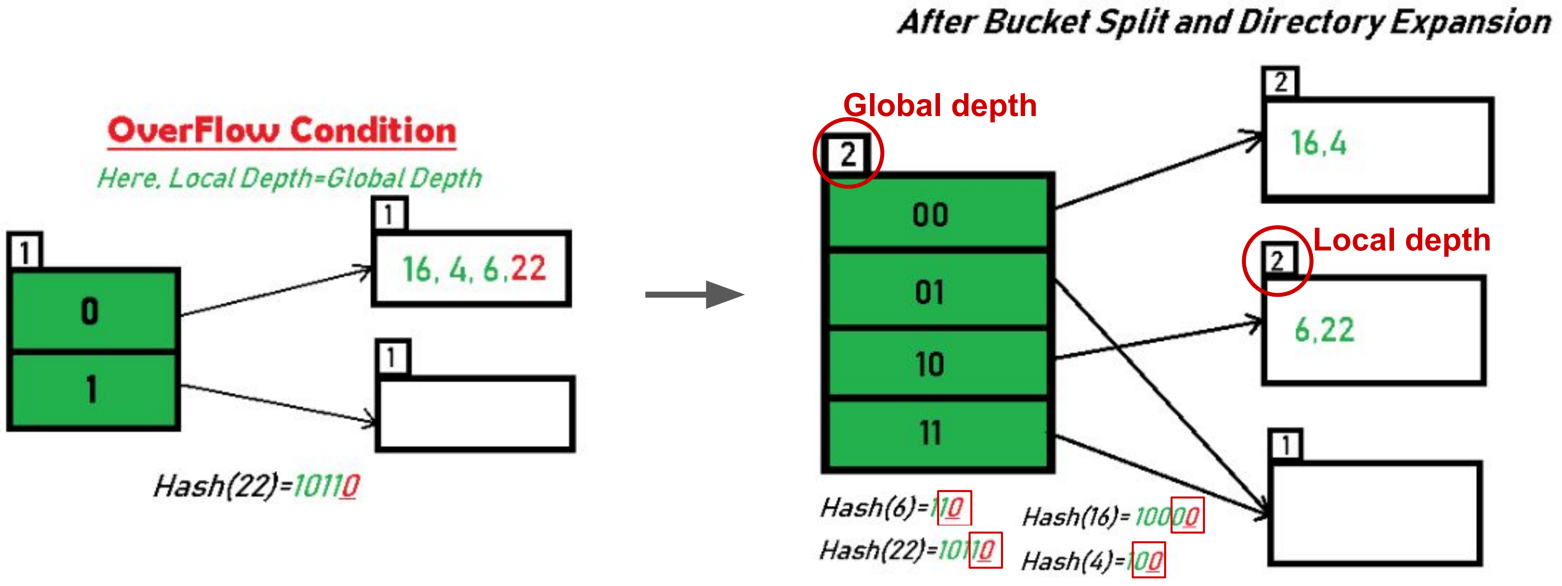
Insert & Collision

- insert by hash index
- if the bucket has any key-value pair, it means colision



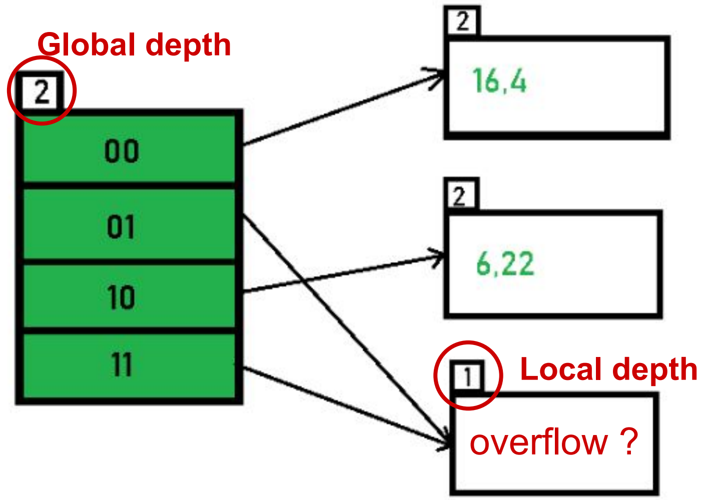
Overflow & Extend

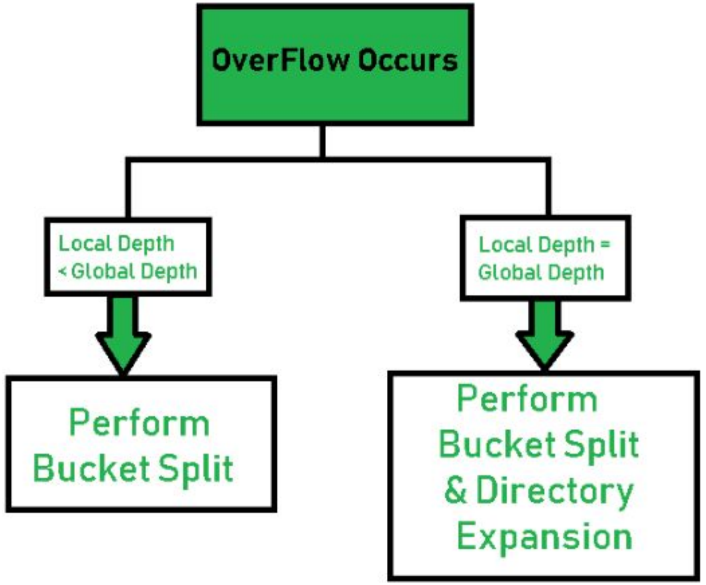
- when the number of key-value pair in the bucket is bigger than bucket size, it means overflow
- if overflow happened, it need to be extended



Overflow & Extend

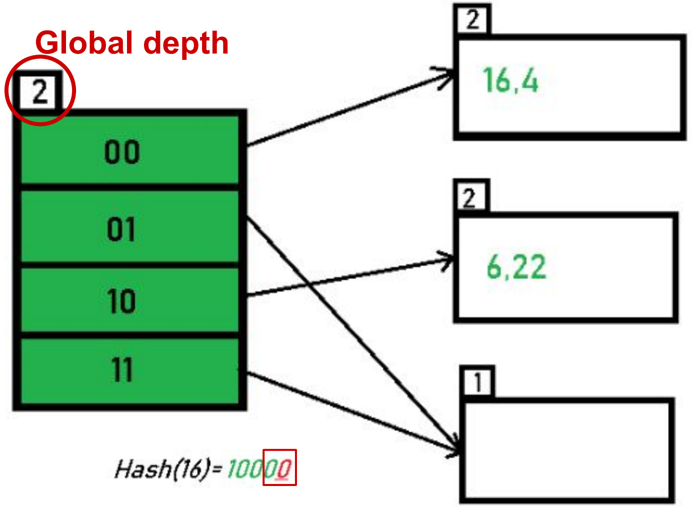
- if local depth is less than global depth, just split the bucket.
- if local depth equal to global depth, need to first extend the directories




OverFlow Occurs
Local Depth < Global Depth
Local Depth = Global Depth
Perform Bucket Split
Perform Bucket Split & Directory Expansion

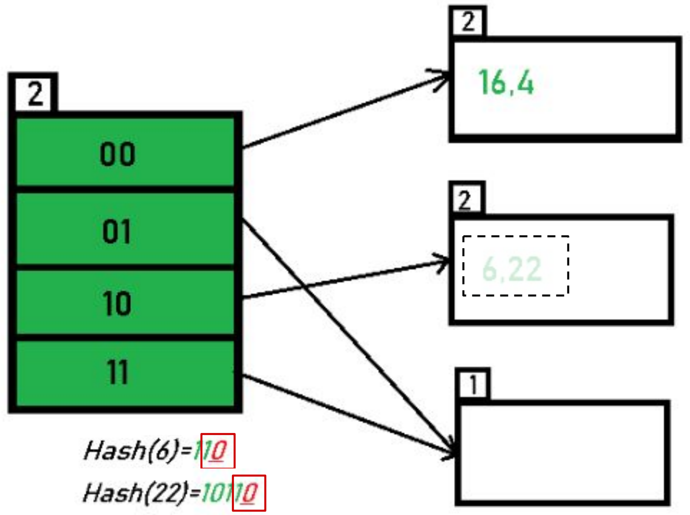
Index

hash function : indexed by bitwise with global depth from the end



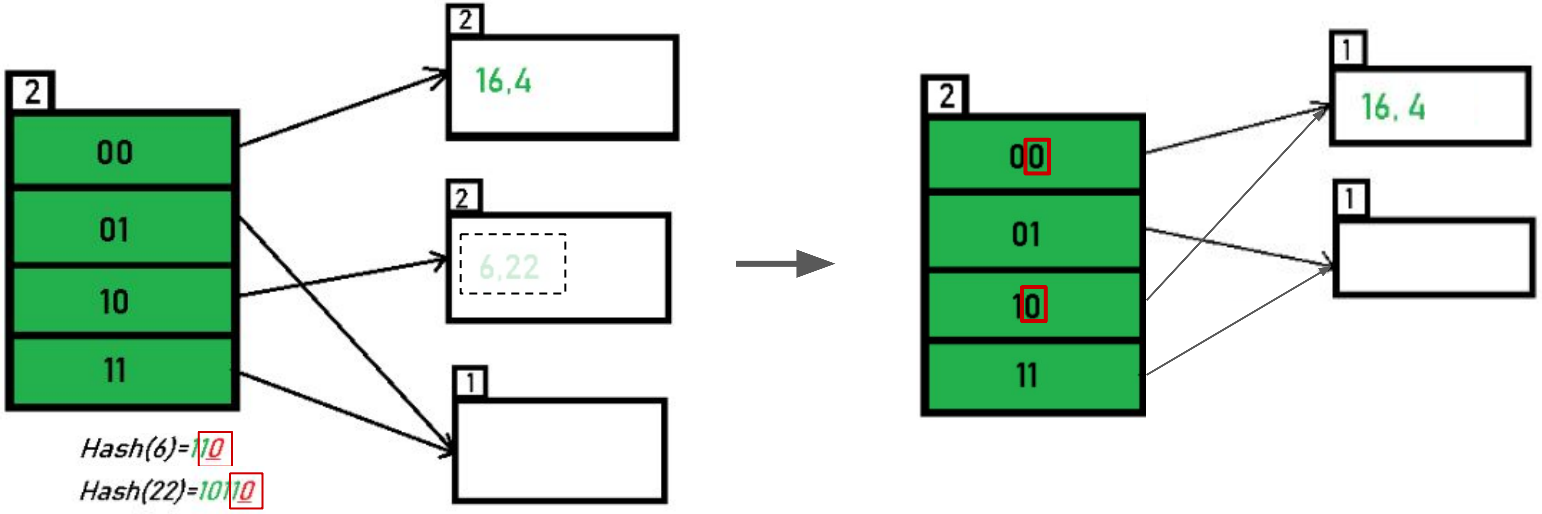
Remove

- Similar to most of hashing



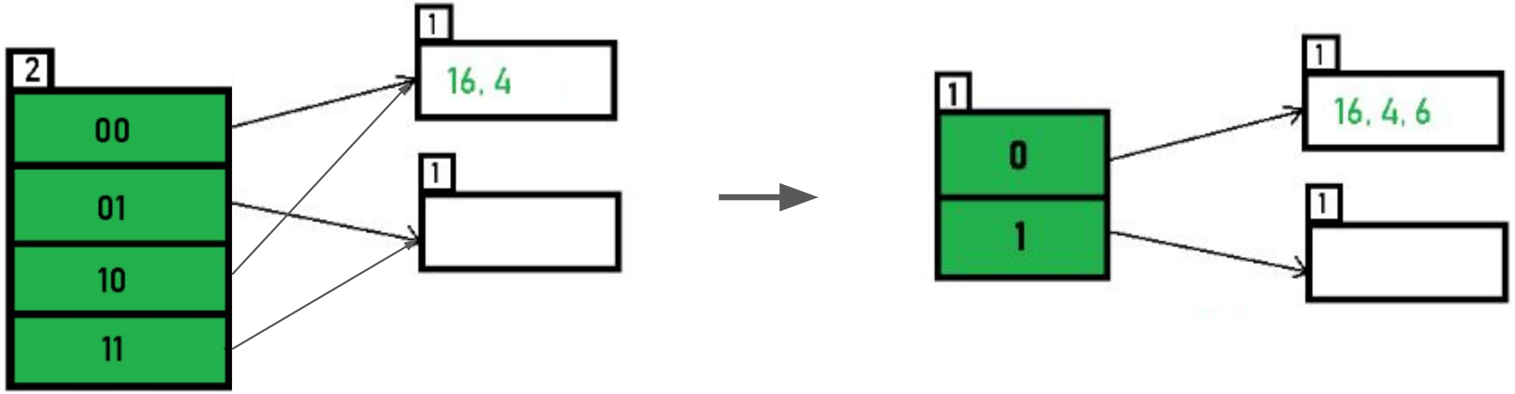
Shrink

- merge the bucket with the one with same hash index in (local depth - 1) if it is empty and the local depth of the pair hash index are the same



Shrink

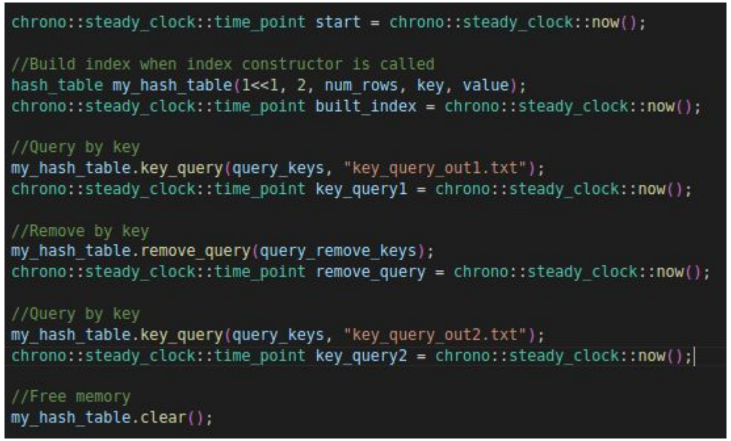
- check the table size and maintain it in appropriate size
- if global depth larger than all local depth, the directory table should be cut in half



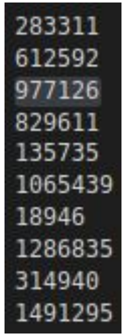
Limitation

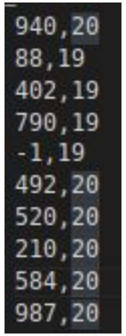
limitation in main.cpp

- the initialization of the hash table should with size = 2(global depth = 1)
- the size of buckets should be 2
- need at least four function
  - constructor, key_query(), remove_query(), clear()



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

//Build index when index constructor is called
hash_table my_hash_table(1<<1, 2, 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();

//Free memory
my_hash_table.clear();
```





key	value, local depth
283311	940,20
612592	88,19
977126	402,19
829611	790,19
135735	-1,19
1065439	492,20
18946	520,20
1286835	210,20
314940	584,20
1491295	987,20

Free for you

- the 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
- deadline 4/28 (Fri.) 23:55

Reference

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