Em xin phép được dùng tiếng Việt để trình bày rõ ràng hơn về design của mình

Em dùng ngôn ngữ javascript để hiện thực, nên những trình bày dưới đây dựa trên các cấu trúc dữ liệu cơ bản của js, gồm: Map, Set và Array.

I. Tổng quát:
1. Cấu trúc dữ liệu:

Vì yêu cầu hiện thực Ledis chỉ có 2 kiểu là String và Set, nên em lưu tất cả cặp (key, value) vào một HashMap, gọi là myMap.

Tất cả các biến đang Expiring vào một HashMap, gọi là activeExpire.

Và một set có tên setType để lưu keys có values là Set, chứ không phải String.

Dùng một switch Enum case để nhận kiểu của command.

1.1 **myMap và setType:**
myMap có dạng như sau:

```
[Entries]]

              1.0: {"hello" => "world"}
              2.1: {"minh" => "huy"}
              3.2: {"a" => "a"}
              4.3: {"setType" => Set(2)}
              5.4: {"list" => Set(5)}
              6.5: {"p" => Set(2)}
```

Trong đó kiểu String được lưu dưới dạng String => String, kiểu Set được lưu dưới dạng String => Set.

Ngoài ra, có một keys đặc biệt là “setType” để lưu một Set chứa tất cả các keys có dạng Set. Vì chỉ có 2 kiểu, nên em xem value của một key không phải kiểu String thì sẽ là kiểu Set. Lý do có keys đặc biệt này vì khi thao tác trên Redis, em nhận thấy:

i. Không thể “sadd” một key kiểu String
ii. Khi “sinter *” , command này chỉ xử lý trên tập dữ liệu có kiểu là Set. Vì thế để lấy độ phức tạp O(M) với M là số set, chứ không phải O(N) với N là tất cả các keys, em đã thêm keys đặc biệt “setType” để biết đâu là biến Set đâu là biến String.

1.2 **activeExpire:**

activeExpire có dạng như sau:

```
Map(3) {"hello" => 187, "a" => 12, "list" => 9}

       1.[[Entries]]
              1.0: {"hello" => 187}
              2.1: {"a" => 12}
              3.2: {"list" => 9}
```

activeExpire map này được cập nhật mỗi giây bởi hàm ledisEXPIRE khi tất cả keys chưa có values là -1.

2. Xử lý lỗi:
Trong quá trình xử lý, em nhận thấy có một số lỗi đặc biệt, ảnh hưởng đến quá trình hiện thực và cấu trúc dữ liệu, gồm:

2.1 (error) WRONGTYPE Operation against a key holding the wrong kind of value:

Lỗi này xảy ra khi xử lý kiểu Set trên một key có kiểu String, nhưng có thể set một setType như một String. Từ đó sinh thêm một kiểu setType đã giải thích ở trên.

2.2 Active Expire:

a) Khi một biến String chưa expired, nếu ta set lại cho nó giá trị mới thì thời gian expire sẽ không còn.

Xử lý bằng cách khi chạy lệnh set sẽ xóa key đó khỏi activeExpire map (O(1)), đồng thời mỗi lần chạy Expire đều kiểm tra xem key đó còn tồn tại trong activeExpire map không (O(1)). Độ phức tạp tổng là O(1) nên không ảnh hưởng đến các hàm hiện thực.

b) Khi timer về -1, chưa xóa biến đó đi mà chỉ khi được gọi lại mới xóa.

Có rất nhiều lệnh gọi lại một key, bao gồm cả những lệnh gọi nhiều key. Ví dụ `keys *`, `sintern`,…
Xử lý bằng cách tạo một hàm xóa biến đó (O(1)) và kiểm tra value của activeSet (O(1)). Nếu value đã bằng = -1 thì mới xóa khỏi map (O(1)). Độ phức tạp là O(1) nên cũng không ảnh hưởng đến các hàm hiện thực.

c) Nhiều expire cùng một lúc

Vì mỗi lần expire đều gọi một hàm giảm thời gian, nên nếu không phân tích hàm này cho mỗi key sẽ dẫn đến trường hợp một biến expired, các biến khác cũng ngừng giảm thời gian.

Xử lý bằng cách cho mỗi biến chạy một hàm giảm thời gian, có ID của hàm đó. Việc code này không làm tăng độ phức tạp của giải thuật.

Và nhiều lỗi khác có thể xử lý đơn giản hơn.
Đó cũng là những hướng giải quyết của các vấn đề em phát hiện trong `new.txt`.

II. Chi tiết các câu lệnh:

1. STRING:

1.1. **SET key value**
Hàm này chỉ có 3 phần tử, chưa hiện thực tính kết hợp thêm (ví dụ SET key value EX time)
Hiện thực bằng Hashmap : hàm Insert
Độ phức tạp O(1)

1.2. **GET key**
Hàm này chỉ có 2 phần tử.
Hiện thực bằng HashMap: hàm Access
Độ phức tạp O(1)

```
Ledis : Le Vu Minh Huy

> set hello world
OK
> set minh huy
OK
> set 1 2
OK
> set l "edis"
OK
>
>
> set a b c
ERROR: wrong number of arguments for 'set' command
> SeT a
ERROR: unknown command 'SeT a'
> set a
ERROR: wrong number of arguments for 'set' command
>
>
> get hello
"world"
> set hello ledis
OK
> get gello
(nil)
> get hello
"ledis"
> get l
""edis""
> get ll
(nil)
>
>
```

2. SET:

2.1. **SADD key value1 [value2...]**
Hàm này có tối thiểu 3 phần tử.
Hiện thực bằng HashMap và Set: dùng các hàm Search, Insert đều O(1)
Độ phức tạp: O(1) cho 1 insert, nên O(N) cho N insert.

2.2. **SREM key value1 [value2…]**
Hàm này có tối thiểu 3 phần tử.
Hiện thực bằng HashMap và Set: dùng các hàm Search, Delete đều O(1)
Độ phức tạp: O(1) cho 1 delete, nên O(N) cho N delete.

2.3. **SMEMBERS key**
Hàm này chỉ có 2 phần tử.
Hiện thực bằng HashMap và Array: dùng các hàm Search, Array.push đều O(1)
Độ phức tạp: O(M) với M là Array size trả về

2.4. **SINTER [key1] [key2] [key3] …**
Ngoài ra em còn hiện thực thêm: **SINTER ***
Hàm này có ít nhất 2 phần tử.
Hiện thực bằng HashMap, Set và Array: xem giải thích cụ thể ở file complexity.txt
Độ phức tạp: O(M x N). Với M là số set input và N là số phần tử của set có size nhỏ nhất.

```
Ledis : Le Vu Minh Huy

> sadd
ERROR: wrong number of arguments for 'sadd' command
> sadd list
ERROR: wrong number of arguments for 'sadd' command
> sadd list 1 2 3 4
(integer) 4
> sadd list 5
(integer) 1
> sadd list 1 2 3
(integer) 0
> sadd list 4 5 6
(integer) 1
> srem list 7
(integer) 0
> srem list 1 2 3 4 5
(integer) 5
>
> smembers list
1) "6"
> sadd a 1 2 3 5
(integer) 4
> sadd b 3 4 5 6
(integer) 4
>
> sinter list a b
(empty list or set)
> sinter list b
1) "6"
> sinter a b c d e
(empty list or set)
> sinter a b
1) "3"
2) "5"
> sinter c
(empty list or set)
> sinter c d
(empty list or set)
```

3. EXPIRE:

3.1. **KEYS**
Hàm này chỉ có 1 phần tử.
Hiện thực bằng HashMap và Array: dùng Array.push (O(1)) cho một vòng loop của M keys
Độ phức tạp: O(M) với M là tổng số keys (kể cả String Type và Set Type)

3.2. **DEL key**
Ngoài ra em còn hiện thực thêm: DEL *
Hàm này có ít nhất 2 phần tử.
Hiện thực bằng HashMap và Set: dùng hàm Search, Delete đều O(1)
Độ phức tạp: O(1) cho mỗi phần tử, O(M) cho M phần tử cần xóa

```
Ledis : Le Vu Minh Huy

> keys *
1) "list"
2) "a"
3) "b"
>
> del b
(integer) 1
> del list a
(integer) 2
> sadd newlist string int float
(integer) 3
>
> key *
ERROR: unknown command 'key *'
> keys *
1) "newlist"
>
>
```

3.3. **EXPIRE key seconds**
Hàm này chỉ có 3 phần tử.
Hiện thực bằng HashMap và Array: dùng các hàm cơ bản với O(1)
Độ phức tạp: O(1)

3.4. **TTL key**
Hàm này chỉ có 2 phần tử.
Hiện thực bằng HashMap và Array: dùng các hàm Search và Access với O(1)
Độ phức tạp: O(1)

```
Ledis : Le Vu Minh Huy

> keys *
1) "newa"
2) "newb"
> sadd list string int float
(integer) 3
>
> expire newa 50
(integer) 1
> expire newb 40
(integer) 1
> expire list 30
(integer) 1
>
> ttl newa
(integer) 38
> ttl newb
(integer) 29
> ttl list
(integer) 21
> expire newa 100
(integer) 1
>
> ttl a
(integer) -2
> ttl newa
(integer) 93
> ttl newb
(integer) 7
> ttl b
(integer) -2
> ttl newb
(integer) -1
>
> get b
(nil)
```

4. SNAPSHOT:

4.1. **SAVE**

Hàm này chỉ có 1 phần tử.

Vì em sử dụng Javascript, và theo em đọc được Javascript thuần không cho phép ghi file đọc file. Bất đắc dĩ, em dùng localStorage.setItem() như một công cụ để lưu trữ cố định.

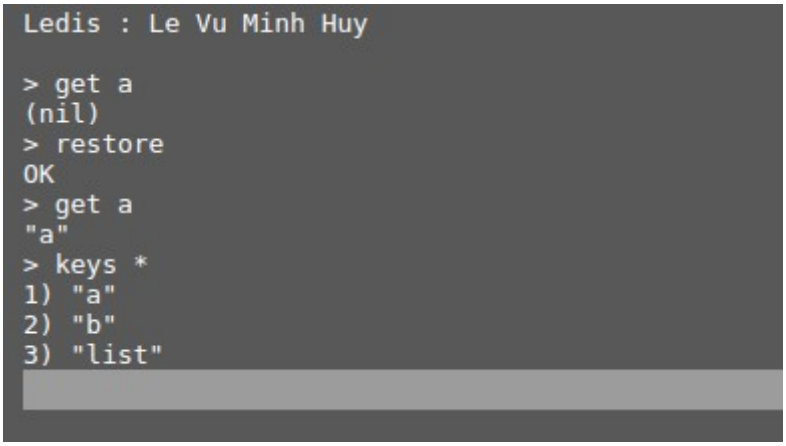
```
Ledis : Le Vu Minh Huy

> keys *
1) "newa"
> expire newa 2
(integer) 1
> ttl newa
(integer) -1
>
> keys *
(empty list or set)
>
> set a a
OK
> set b b
OK
> sadd list a b c
(integer) 3
> keys *
1) "a"
2) "b"
3) "list"
>
> save
OK
```

4.2. **RESTORE**
Hàm chỉ có 1 phần tử
Hiện thực bằng cách lưu 2 Array cacheMap và cacheActiveExpire từ localStorage.getItem(). Sau đó cho 2 lần loop vơi N phần tử của myMap cũ và M phần tử của activeExpire cũ. Trong mỗi vòng loop, dùng hàm set của HashMap (O(1)).

Thêm 2 lần O(N) để chạy lại tất cả expiring variable bằng cách chuyển hash sang array và chạy array đó để duyệt hàm ledisEXPIRE.

Vậy độ phức tạp là: O(X) + O(M) + O(N) với O(X) là độ phức tạp của getItem(), M là số keys lưu trữ của myMap, N là số key lưu trữ của activeExpire.



5. Ngoài ra em còn viết thêm 2 hàm **clear** và **flushall** để tiện cập nhật