Workshop Leaplist Todo’s

1. **Talk to moshe.** Range queries – the interface in the code states that it only traverses the structure. It doesn’t return a range of requested values. We understand that for benchmarking purposes there’s not really a difference, since from that point on it’s one threaded code.
2. Node.Low id needed?
3. ~~Node members : Low,High,Count change to long . level to byte. Check modifiers also (protected/public etc.)~~
4. Metadata in tire???
5. ~~TrieMetadata . prefix\_length & prefix change to long that holds 4 lower bits as prefix\_length and prefix to upper 60. Add a property that gets prefix\_length and that gets prefix. Check with 2 variables and with 2 getter functions.~~
6. Remove big chunks of code from constructors to functions
7. trie\_create\_from\_array in loops why are the variables are volitale ? seems like only one thread will run on trie funcs so why volatile . also ln. 241 in trie.c , the if could be removed and set outside the loop. Notice : No volitaile in java implementation .
8. ~~struct trie\_metadata\_t.~~

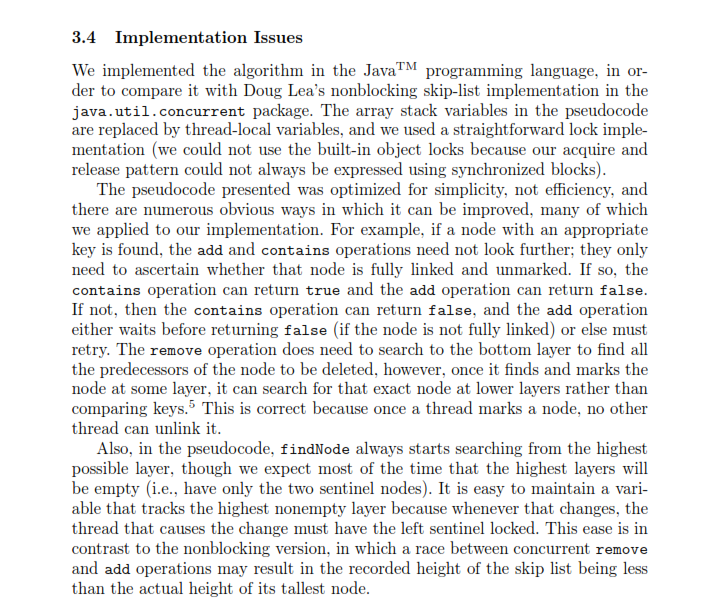
~~typedef struct {~~

~~unsigned int prefix\_length : 4;~~

~~unsigned long prefix : 60;~~

~~} trie\_metadata\_t;~~

~~this could take a lot more memory than intended, needs to be changed to unsigned long and unsigned long. Check if current configuration indeed takes 8 bytes and not 12/16. Check on nova.~~

1. ~~New leaplist() doesn’t initialize trie.~~
2. Insert – overwrites means that insert always happens. If overwrite = 1 , if an update is required perform the update. if overwrite =0 , if an update is required don’t perform the update.
3. Ask moshe about update/remove. In the code they send one list, one key, one value.. in the article they send an array of each… So the question is whether the user should iterate with a for and call the function , or should the function do that.
4. All the places with avoid sentinel increment key by 2, so it would keep 0 1 as operational keys
5. Make sure all the api java functions do increment key by 2 to avoid sentinel.
6. Bug! Insert method, when updating value is not really updated it’s overwritten to the old value all over again. Added “newNode[m].data[i].key = n.data[j].key;” and continue if key is equal… Need to change in C also
7. Remove – different implementation for example in lock read write and in transcatino . in the begining there’s a while live and unmarked..
8. Understand the diff between unmarked and live
9. Get unmark to be a utility outside the node . so there won’t be a case where wer do null.unmark.. also when unmarking make sure isn’t null.
10. Check why 7 isn’t removing . and 90 is outofbound in trie. Should be merging and trie gets count 0
11. Bug - when a node need to be merged , if for example we have 3 node : empty, 1, 2 & 3. Removing 1 will cause. empty,empty, 2&3.
12. Buggy implementation in c++ vs article compliant code (ours) . Do we need to change c++ code? Also Lock general in our case vs read/write in the C++ code
13. Notice : update only of key in a full node will cause a split ( when it doesn't really have to ), how bad is it when there are 300 keys in a node?
14. Question : How to test functionality on multi threaded environment? are we supposed to enter 300 values and know what to expect?
15. Do need low?, because their code doesn't always set the low value.. and then it's redundant.
16. Maybe the following could help for improvement of the code ( it applies to grained lock skip list but maybe the general idea can help) : 

Explanation of how Trie works.

Building a trie from a given array :

The trie iterates through the tree by checking 2 hex digits at a time (Most significant).

With one exception , when there’s a total number of digits which is odd, the upmost layer will containing 16 nodes (as opposed to normally 256) to represent 16 hex possibilities of one digit.

Prefix is the Most significant joint hex digits . when it’s 14 it means we have only 2 hexes that’s why 256 nodes are create. 15 and we have only 1 hex that’s different, thus we create 16 nodes.

**Questions** :

1. Ask moshe about update/remove. In the code they send one list, one key, one value. Disregard the given list and iterate thorough all the list of the db. in the article they send an array of each… So the question is whether the user should iterate with a for and call the function, or should the function do that.
2. In the article for update and remove why is max\_lists needed? Doesn’t size suffice?
3. Make sure the DB that holds L-Leaplist has Leapslist and update/remove. And each lepalist hold functions to lookup and range queries.
4. Range queries the code does nothing. Goes to low then to high. Doesn’t set and return nothing nowhere.