Read Me

How I implement the skip list?

Firstly, I made a class named “ListNode”, which has instance variables of ArrayList of Event and three pointers of “next”, “down” and “prev”. The ListNode also has a method which is used to add a new level to the original node.

In the EventList class, in order to realize the assigned four method, I made two basic methods, “doubleSize” and “find” which are respectively used to dynamically double the height of head and tail and to realize finding events happening at a particular year.

The “insert” method is the most important method. It determines the success of other methods. My implementation could be divided to two parts, one is if the happening year of the event is already existed in the list, I add the event to the list node with key of the same year, not only to the highest level but through the whole levels. The other is if the year is not found, firstly make a list node containing the given event and randomly set its height, then starting from the head, find the position of the new event at each level through the highest level to the bottom. The expected height of the list is $O(\log n)$, which has already been discussed in class. And the insertion can be thought as finding the destined position of the given event in the skip list. Given the fact that the basic find takes $O(\log n)$ which is discussed in class, the insert method also takes $O(\log n)$.

For the “remove” method, I firstly use ListNode with “prev” pointer. In this part, if we can find the year, we can unlink the list node at each level by make the node’s previous node point to the node’s next. Because the basic “find” takes $O(\log n)$ at expectation, remove in this case also takes $O(\log n)$. To meet the requirement of part Two of removing the “prev” pointer, I implement the remove method by checking and unlinking the node with given year at every level. This implementation can be taken as find the node at each level and unlink it. Since unlink only takes constant time, the cost in this case is also $O(\log n)$.

The “findMostRecent” method and the “findRange” method are kind of similar. If the given year(in findRange it denotes the “first”) could be found in the skip list, target the particular node containing the year, and return all the events in the node(in “findRange” return all events from the first year to last year). As previous argument in “remove” method, the basic find takes $O(\log n)$, and make an event array of size m , takes $\Theta(m)$, in total it takes $O(\log n + m)$ for this case. If the given year is not in the skip list, for “findMostRecent” , target the node containing the year which is just smaller than the given year, for “findRange”, target the node containing the year which is just greater than the first year. In this case, I start from the head and go through down to the bottom level and target the particular node at the bottom level. Because a list node at every level has the same content, the list node at bottom level will give all events.

Mistakes I made

The biggest mistake I made in this lab is in the insertion part. The relevant part is:

```
while(current.down != null){
      if(t < height){
            while(current.next.events.get(0).year < e.year){
                  current = current.next;
            }
            ListNode nextOne = current.next;
            nextOne = insert;
            insert.next = current.next.next;
            //                      insert.prev = current;
            //                      nextOne.prev = insert;
            insert = insert.down;
```

Firstly, I made the content in the frame previously look like:

```
            current.next = insert;
            insert.next = nextOne;
```

Actually, I don't quite understand why my previous implementation would not work. But with Zhengdao Chen, one of the TA, we work out the later one by intuition. Thanks him! Please educate me if you got to know the difference between the two implementations and why one works while the other don't. Thank you !