Experimental setup

Data scenarios

There are two scenarios: 1. additions and removals 2. Searches(searches : additions and removals = 1.5 : 1)
They cover all the operations.

Multisets

We use two alphabet from a to z as one set, and there will be 26*26=676 possibilities.
We choose 10000, 20000 and 40000 sizes to do test, because these sizes are enough for us to test different data structures.

Generation of scenarios

We use “MainGenerator” and “DataGenerator”, which are two java file to do tests.
Input different data structures, the number of multisets and command for operations.
Output running time (second).

Timing

We use nanotime to do tests.
Result time = start time - end time(operation during time).
For each operation, we do ten times and get average time .

Fixed sets

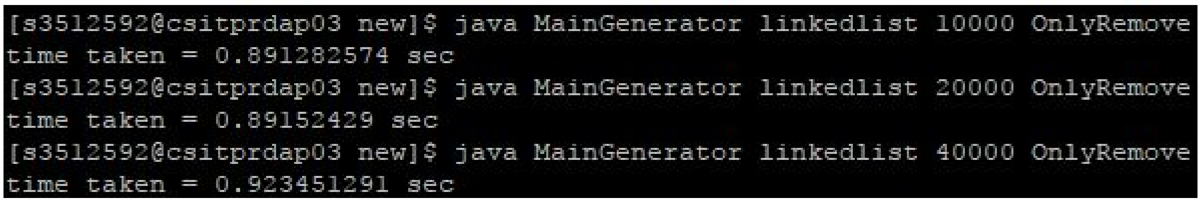
We use two alphabet from a to z as one set, and there will be 26*26=676 possibilities.
Before doing operations, we input 10000, 20000 or 40000 sizes as initial size and we can change that. It covers all the possibilities.
After one operations, it will return initial sets automatically.

Evaluation

Scenario 1 (varying ratio of additions to removals)

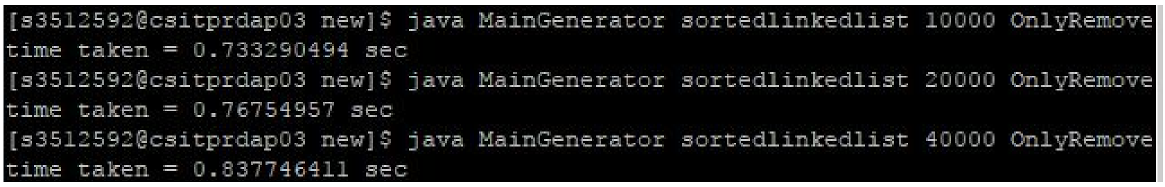
Additions is 40000, change removals

We found that as the number of additions to removals increased, linked list performance did not change. The running time around 0.9 second. Therefore, the result is average case O(1), which is the same as theory. See figure (1) below. Sorted linked list had the same performance as linked list. The running time around 0.77 second. The result is average case O(1), which is the same as the theory. See figure (2) below. We also can know sorted linked list is faster than linked list.



```
[s3512592@csitprdap03 new]$ java MainGenerator linkedlist 10000 OnlyRemove
time taken = 0.891282574 sec
[s3512592@csitprdap03 new]$ java MainGenerator linkedlist 20000 OnlyRemove
time taken = 0.89152429 sec
[s3512592@csitprdap03 new]$ java MainGenerator linkedlist 40000 OnlyRemove
time taken = 0.923451291 sec
```

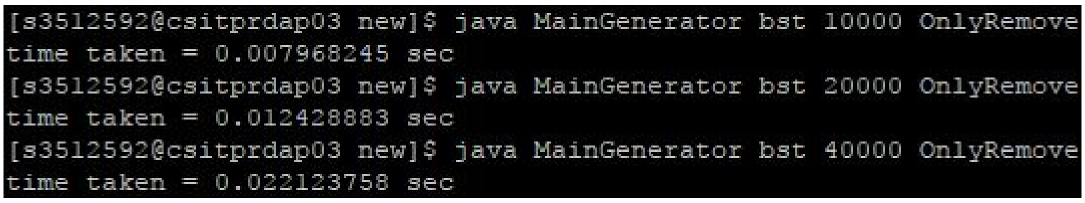
(1)



```
[s3512592@csitprdap03 new]$ java MainGenerator sortedlinkedlist 10000 OnlyRemove
time taken = 0.733290494 sec
[s3512592@csitprdap03 new]$ java MainGenerator sortedlinkedlist 20000 OnlyRemove
time taken = 0.76754957 sec
[s3512592@csitprdap03 new]$ java MainGenerator sortedlinkedlist 40000 OnlyRemove
time taken = 0.837746411 sec
```

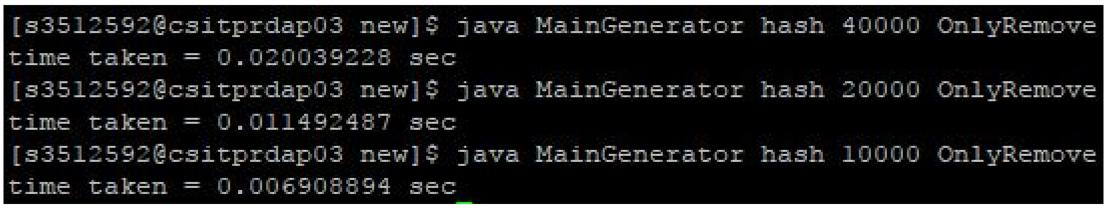
(2)

We found that as the number of additions to removals increased, binary search tree performance degraded. It almost changed as O(n) from 0.008 to 0.022. We can assume there are some O(log(n)) in removals. It made result become smaller. According to theory, this is the worst case. Therefore, the result is O(n), which is the same as theory. See figure (3) below. Compare to hash table, they had the same performance, which is O(n). It also is the worst case for hash table. Therefore, the result is O(n), which is the same as theory. See figure (4) below.



```
[s3512592@csitprdap03 new]$ java MainGenerator bst 10000 OnlyRemove
time taken = 0.007968245 sec
[s3512592@csitprdap03 new]$ java MainGenerator bst 20000 OnlyRemove
time taken = 0.012428883 sec
[s3512592@csitprdap03 new]$ java MainGenerator bst 40000 OnlyRemove
time taken = 0.022123758 sec
```

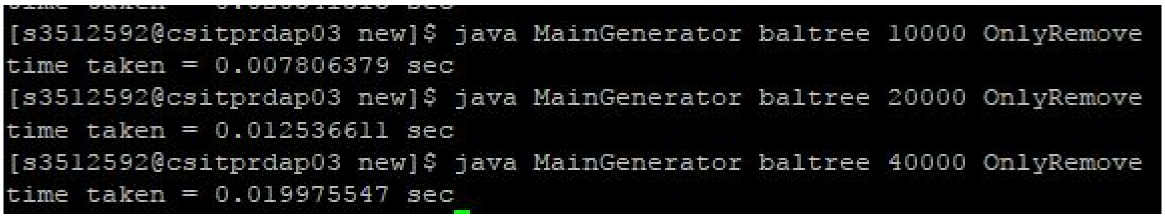
(3)



```
[s3512592@csitprdap03 new]$ java MainGenerator hash 40000 OnlyRemove
time taken = 0.020039228 sec
[s3512592@csitprdap03 new]$ java MainGenerator hash 20000 OnlyRemove
time taken = 0.011492487 sec
[s3512592@csitprdap03 new]$ java MainGenerator hash 10000 OnlyRemove
time taken = 0.006908894 sec
```

(4)

We found that as the number of additions to removals increased, B-tree performance degraded. It almost changed as O(n) from 0.0008 to 0.020. We can assume some O(log(n)) in the removals. It made result become smaller. According to theory, this is the average case. Therefore, the result is O((n), which is the same as theory. See figure (5) below. Binary search tree, hash table and B-tree are almost have the same running speed.

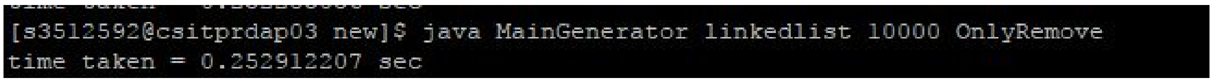


```
[s3512592@csitprdap03 new]$ java MainGenerator baltree 10000 OnlyRemove
time taken = 0.007806379 sec
[s3512592@csitprdap03 new]$ java MainGenerator baltree 20000 OnlyRemove
time taken = 0.012536611 sec
[s3512592@csitprdap03 new]$ java MainGenerator baltree 40000 OnlyRemove
time taken = 0.019975547 sec
```

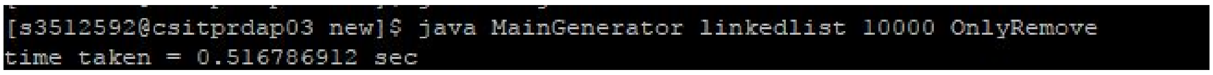
(5)

Removals is 10000, change additions(10000,20000,40000)

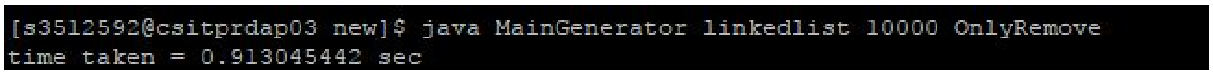
We found that as the number of removals to additions increased, linked list performance degraded. The running time almost is linear. See figure(6) below. Therefore, the result is O(n), which is the same as theory. We can assume there are some O(1) in additions. It made result become smaller. According to theory, this is the worst case. Compare to sorted linked list, they are almost same. See figure(7) below.



```
[s3512592@csitprdap03 new]$ java MainGenerator linkedlist 10000 OnlyRemove
time taken = 0.252912207 sec
```

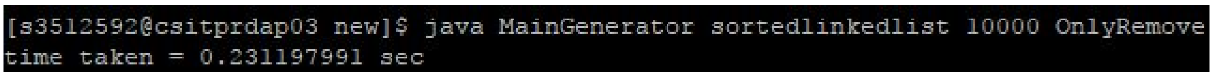


```
[s3512592@csitprdap03 new]$ java MainGenerator linkedlist 10000 OnlyRemove
time taken = 0.516786912 sec
```

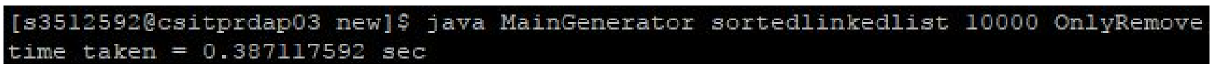


```
[s3512592@csitprdap03 new]$ java MainGenerator linkedlist 10000 OnlyRemove
time taken = 0.913045442 sec
```

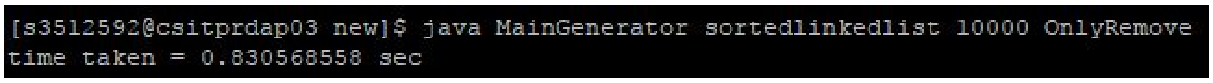
(6)



```
[s3512592@csitprdap03 new]$ java MainGenerator sortedlinkedlist 10000 OnlyRemove
time taken = 0.231197991 sec
```



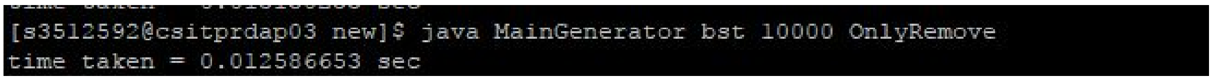
```
[s3512592@csitprdap03 new]$ java MainGenerator sortedlinkedlist 10000 OnlyRemove
time taken = 0.387117592 sec
```



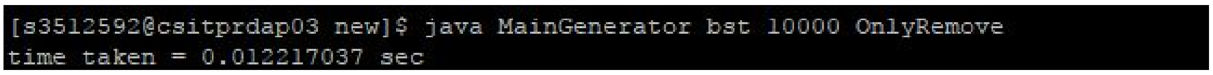
```
[s3512592@csitprdap03 new]$ java MainGenerator sortedlinkedlist 10000 OnlyRemove
time taken = 0.830568558 sec
```

(7)

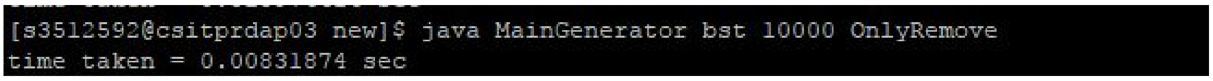
We found that as the number of removals to additions increased,binary search tree performance was improved . According to theory, it should be constant or degraded. It almost is constant O(1). See figure(8). We assume there are some O(log(n)) in additions. It made first two result become bigger.



```
[s3512592@csitprdap03 new]$ java MainGenerator bst 10000 OnlyRemove
time taken = 0.012586653 sec
```



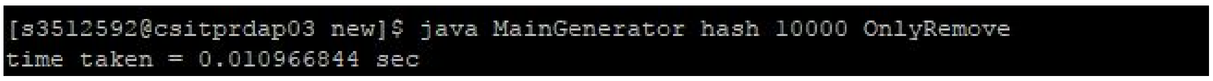
```
[s3512592@csitprdap03 new]$ java MainGenerator bst 10000 OnlyRemove
time taken = 0.012217037 sec
```



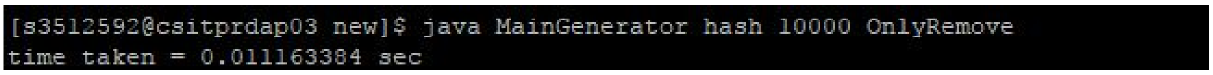
```
[s3512592@csitprdap03 new]$ java MainGenerator bst 10000 OnlyRemove
time taken = 0.00831874 sec
```

(8)

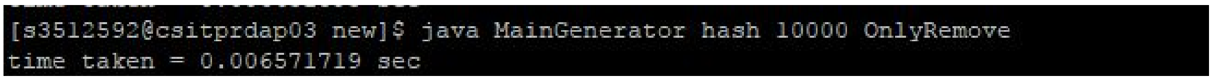
We found that as the number of removals to additions increased,hash table performance was improved . According to theory, it should be degraded. It almost is constant O(1). See figure(9). We assume there are some O(n) in additions. It made first two result become bigger.



```
[s3512592@csitprdap03 new]$ java MainGenerator hash 10000 OnlyRemove
time taken = 0.010966844 sec
```



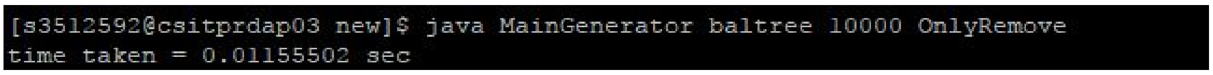
```
[s3512592@csitprdap03 new]$ java MainGenerator hash 10000 OnlyRemove
time taken = 0.011163384 sec
```



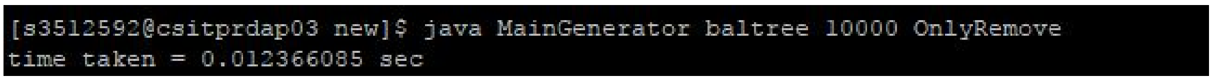
```
[s3512592@csitprdap03 new]$ java MainGenerator hash 10000 OnlyRemove
time taken = 0.006571719 sec
```

(9)

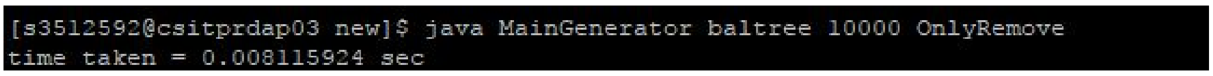
We found that as the number of removals to additions increased, B-tree performance was improved . According to theory, it should be degraded. It almost is constant O(log(n)). See figure(10). We assume there are some O(n) in additions. It made first two result become bigger.



```
[s3512592@csitprdap03 new]$ java MainGenerator baltree 10000 OnlyRemove
time taken = 0.01155502 sec
```



```
[s3512592@csitprdap03 new]$ java MainGenerator baltree 10000 OnlyRemove
time taken = 0.012366085 sec
```

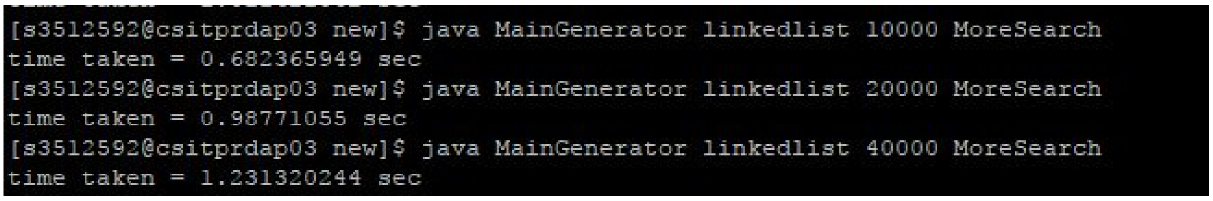


```
[s3512592@csitprdap03 new]$ java MainGenerator baltree 10000 OnlyRemove
time taken = 0.008115924 sec
```

(10)

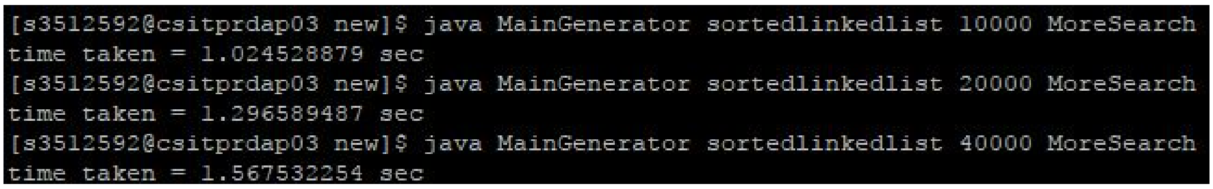
Scenario 2 (varying ratio of searches vs additions and removals) (1.5 : 1)

We found that as the number of search increased, linked list performance degraded. It is the same as theory. The result is linear O(n). See figure (11) below. Sorted linked list had the same performance as linked list. The result also is linear O(n). See figure (12) below.



```
[s3512592@csitprdap03 new]$ java MainGenerator linkedlist 10000 MoreSearch
time taken = 0.682365949 sec
[s3512592@csitprdap03 new]$ java MainGenerator linkedlist 20000 MoreSearch
time taken = 0.98771055 sec
[s3512592@csitprdap03 new]$ java MainGenerator linkedlist 40000 MoreSearch
time taken = 1.231320244 sec
```

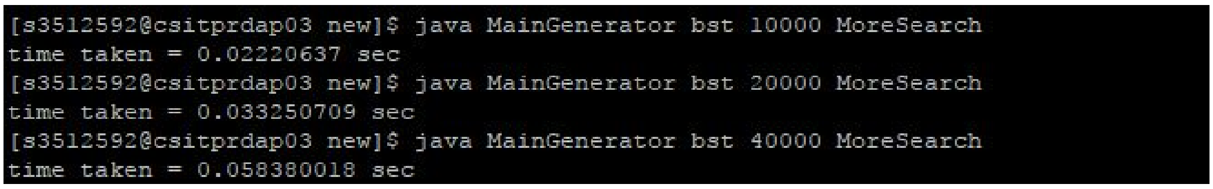
(11)



```
[s3512592@csitprdap03 new]$ java MainGenerator sortedlinkedlist 10000 MoreSearch
time taken = 1.024528879 sec
[s3512592@csitprdap03 new]$ java MainGenerator sortedlinkedlist 20000 MoreSearch
time taken = 1.296589487 sec
[s3512592@csitprdap03 new]$ java MainGenerator sortedlinkedlist 40000 MoreSearch
time taken = 1.567532254 sec
```

(12)

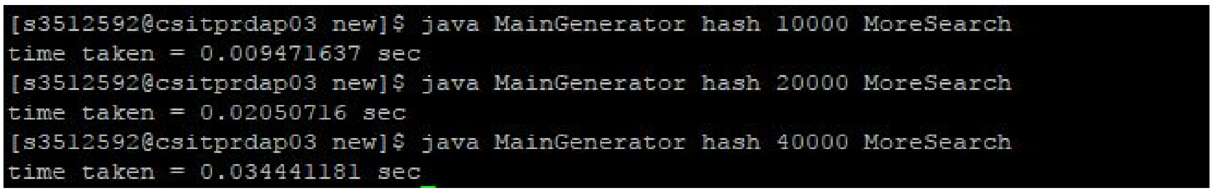
We found that as the number of search increased, binary search tree performance degraded. It is the same as theory. The result is almost like linear O(n). See figure (13). It is the worst case. We can assume there are some O(log(n)) in the second result. It made it become smaller.



```
[s3512592@csitprdap03 new]$ java MainGenerator bst 10000 MoreSearch
time taken = 0.02220637 sec
[s3512592@csitprdap03 new]$ java MainGenerator bst 20000 MoreSearch
time taken = 0.033250709 sec
[s3512592@csitprdap03 new]$ java MainGenerator bst 40000 MoreSearch
time taken = 0.058380018 sec
```

(13)

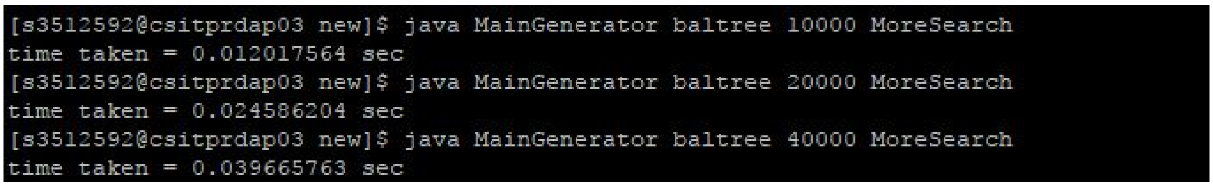
We found that as the number of search increased, hash table performance degraded. According to theory. It can be constant O(1) or linear O(n). The result is almost like linear O(n). See figure (14). It is the worst case. We can assume there are some O(1) in the last result. It made it become smaller.



```
[s3512592@csitprdap03 new]$ java MainGenerator hash 10000 MoreSearch
time taken = 0.009471637 sec
[s3512592@csitprdap03 new]$ java MainGenerator hash 20000 MoreSearch
time taken = 0.02050716 sec
[s3512592@csitprdap03 new]$ java MainGenerator hash 40000 MoreSearch
time taken = 0.034441181 sec
```

(14)

We found that as the number of search increased, B-tree performance degraded. According to theory. It can be O(log(n)) or O(n). The result is almost like linear O(n). See figure (15). It is the worst case. We can assume there are some O(log(n)) in the last result. It made it become smaller.



```
[s3512592@csitprdap03 new]$ java MainGenerator baltree 10000 MoreSearch
time taken = 0.012017564 sec
[s3512592@csitprdap03 new]$ java MainGenerator baltree 20000 MoreSearch
time taken = 0.024586204 sec
[s3512592@csitprdap03 new]$ java MainGenerator baltree 40000 MoreSearch
time taken = 0.039665763 sec
```

(15)

Recommendation

Based on above evaluation, we recommend use hash table, because it is faster than other.