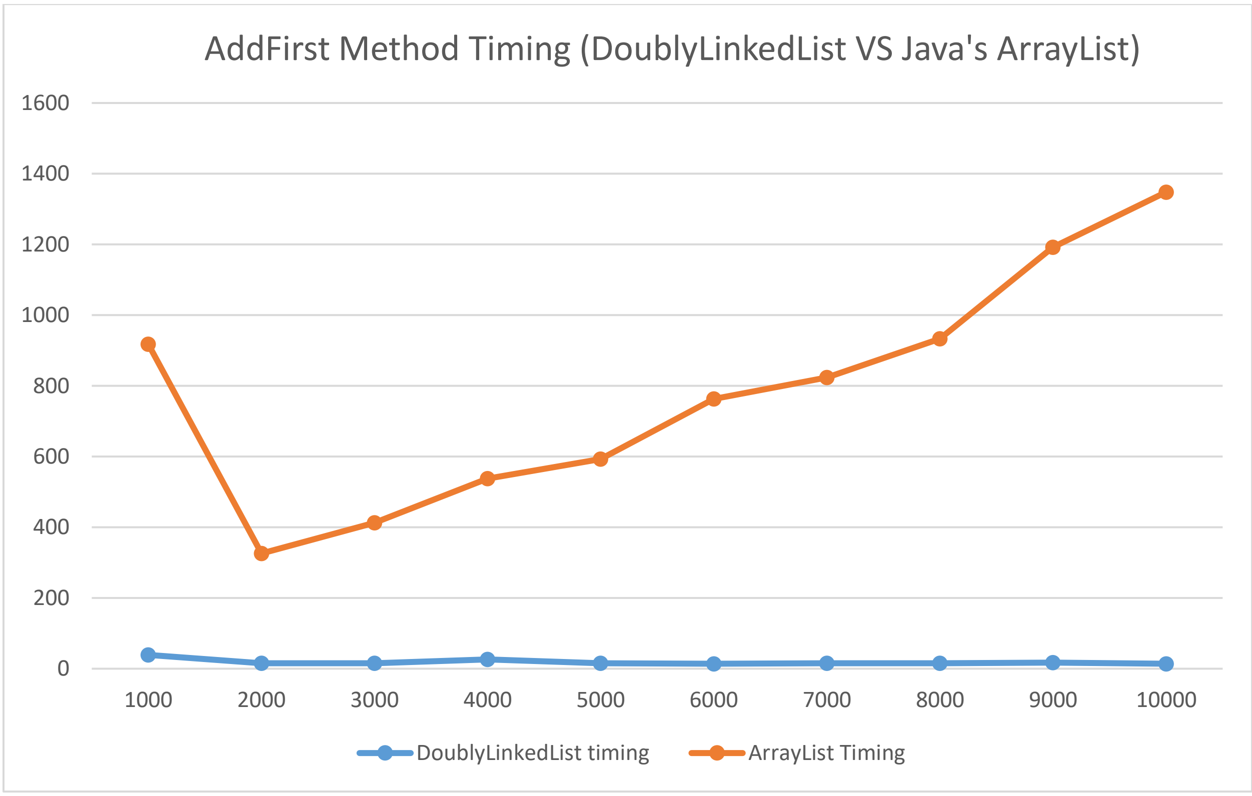
Zhangtuoming Zhao u0847610 CS-2420

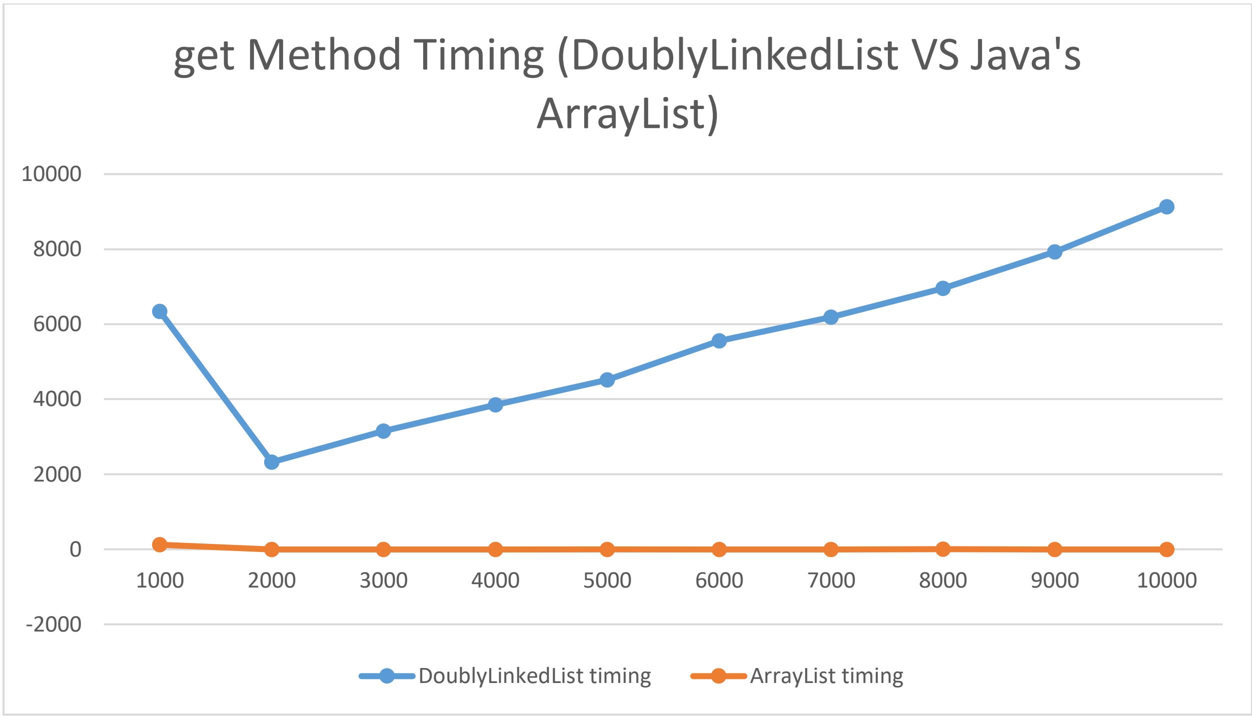
Analysis Doc for Assignment 06

1. **Collect and plot running times in order to answer each of the following questions. Note that this is this first assignment that does not specify the exact procedure for creating plots. You must design your own timing experiments that sufficiently analyze the problems. Be sure to explain all plots and answers.**
   a. **Is the running time of the addFirst method O(c) as expected? How does the running time of addFirst(item) for DoublyLinkedList compare to add(0, item) for ArrayList?**



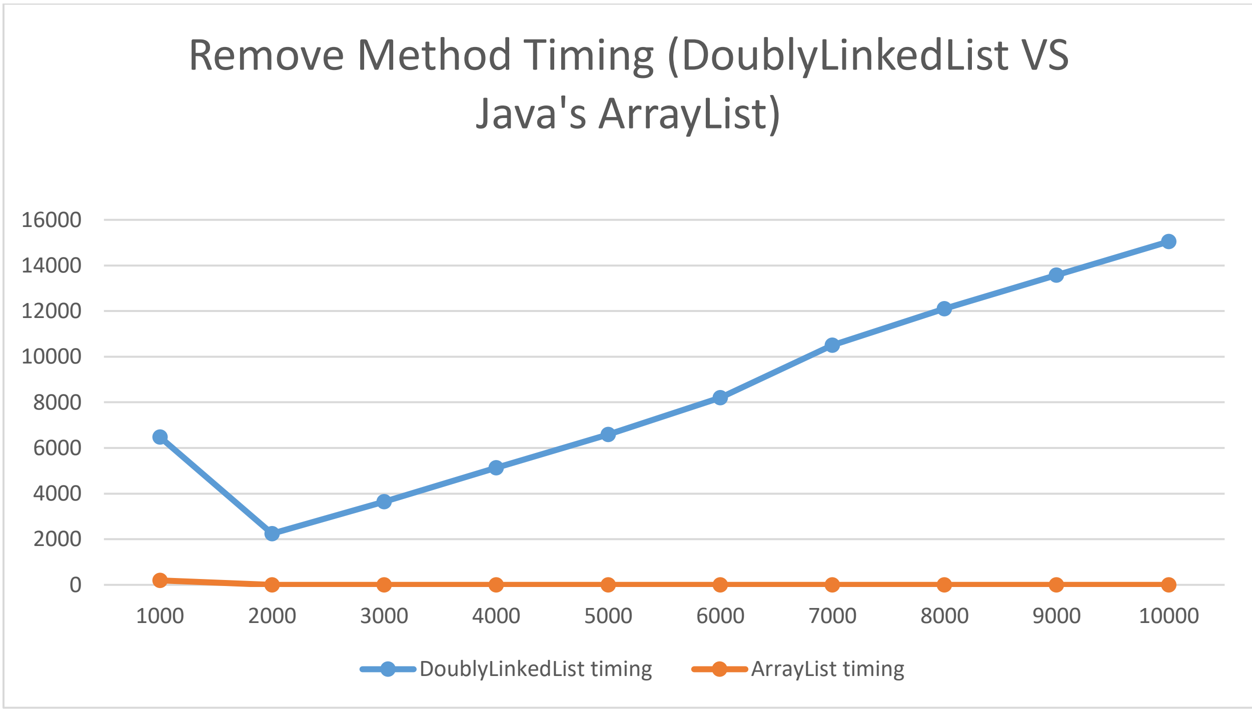
It is very obvious that the Doubly-Linked List is kind of a constant time to add the items at the first place. However, Java's Array List is a linearly increasing time to add the items at the first place. To conclude, the Big-O of the Doubly-Linked List is constant and Array List is O(N). Except for the first point which is not accurate enough, and I think that would be a system exception which is appeared in the previous assignment testers.

b. **Is the running time of the get method O(N) as expected? How does the running time of get(i) for DoublyLinkedList compare to get(i) for ArrayList?**



It can be easy to tell the difference between the Doubly-linked List and Java's Array List. In the Get method timing, their Big-O are exchanged. The time for the Doubly-linked List is linearly increasing, also except for the first point. And Java's Array List is a constant. To conclude, the Big-O of the Doubly-Linked List is O(N) and Array List is a constant.

c. Is the running time of the remove method O(N) as expected? How does the running time of remove(i) for DoublyLinkedList compare to remove(i) for ArrayList?

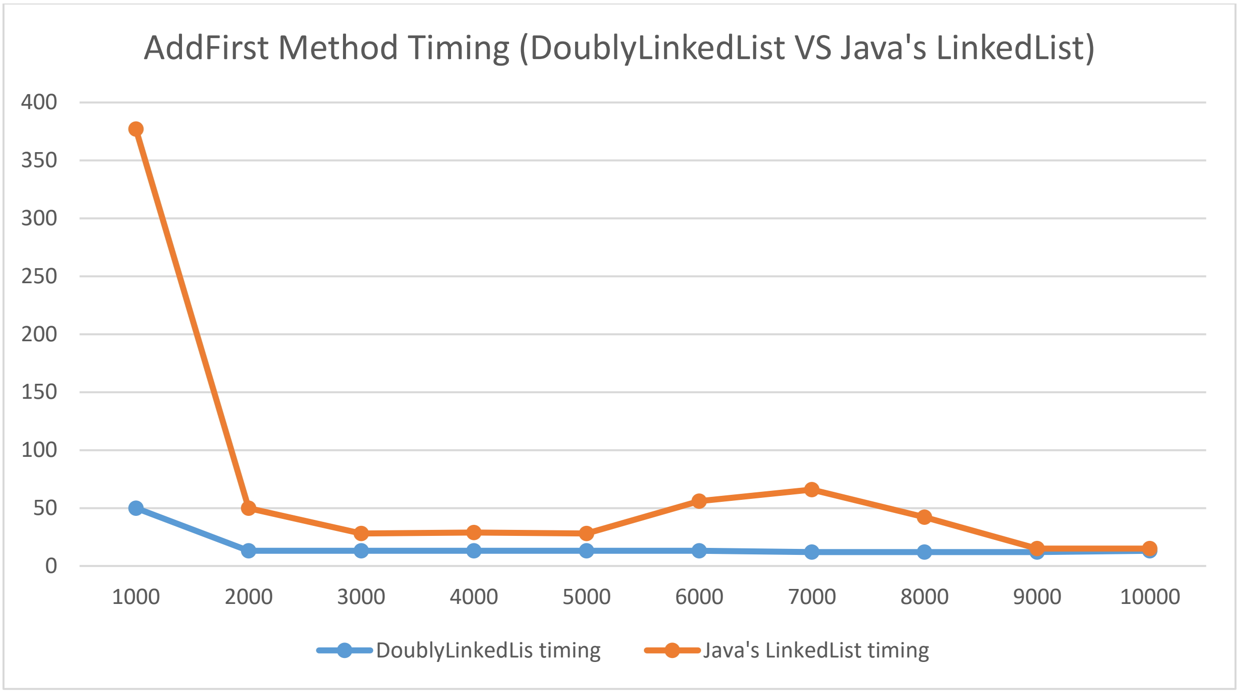


It is kind of similar to the get method timing graph. The time for the Doubly-linked List is linearly increasing. And for the Java's Array List which is a constant. So it is the same result as the get method. To conclude, the Big-O of the Doubly-Linked List is O(N) and Array List is a constant.

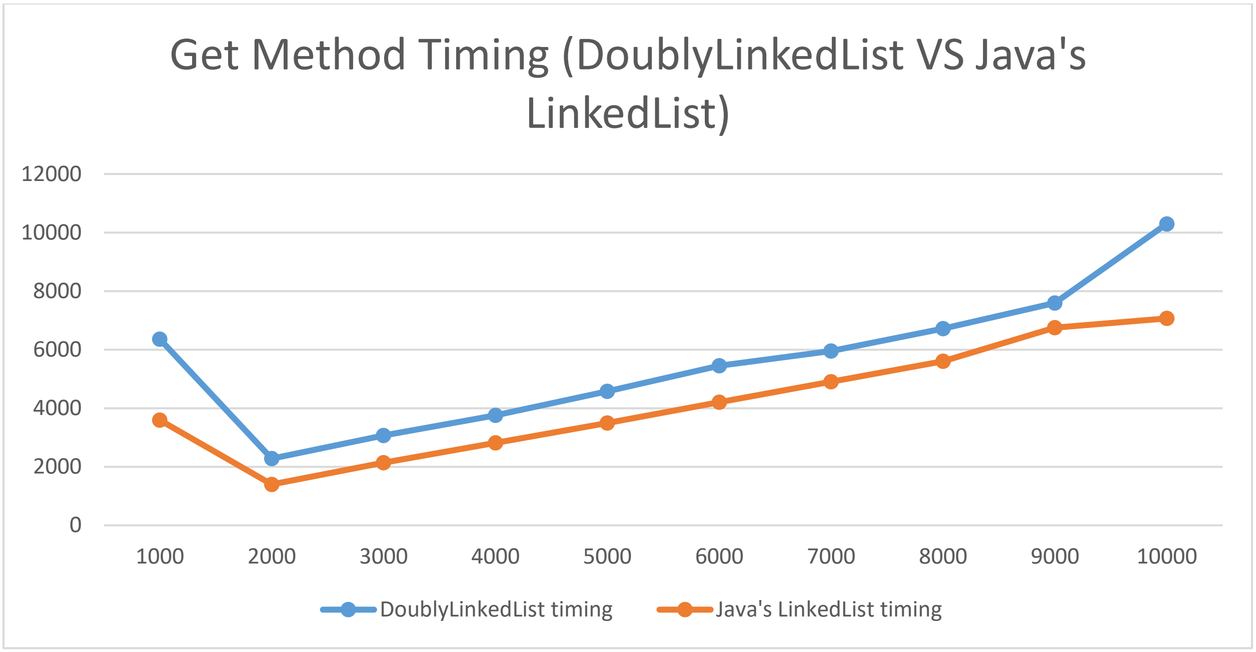
2. **In general, how does DoublyLinkedList compare to ArrayList, both in functionality and performance? Please refer to Java's ArrayList documentation.**

   I think they fit different tasks. For example, when we need to call more get method and remove method, we are better to choose the Java's Array List. And the Doubly-linked List is using for adding. In general, or in some special cases, Java's Array List performs better than my Doubly-Linked List. So user will prefer to choose Java's Array list. And according to API of Array List, there are more functions in Java's storage.

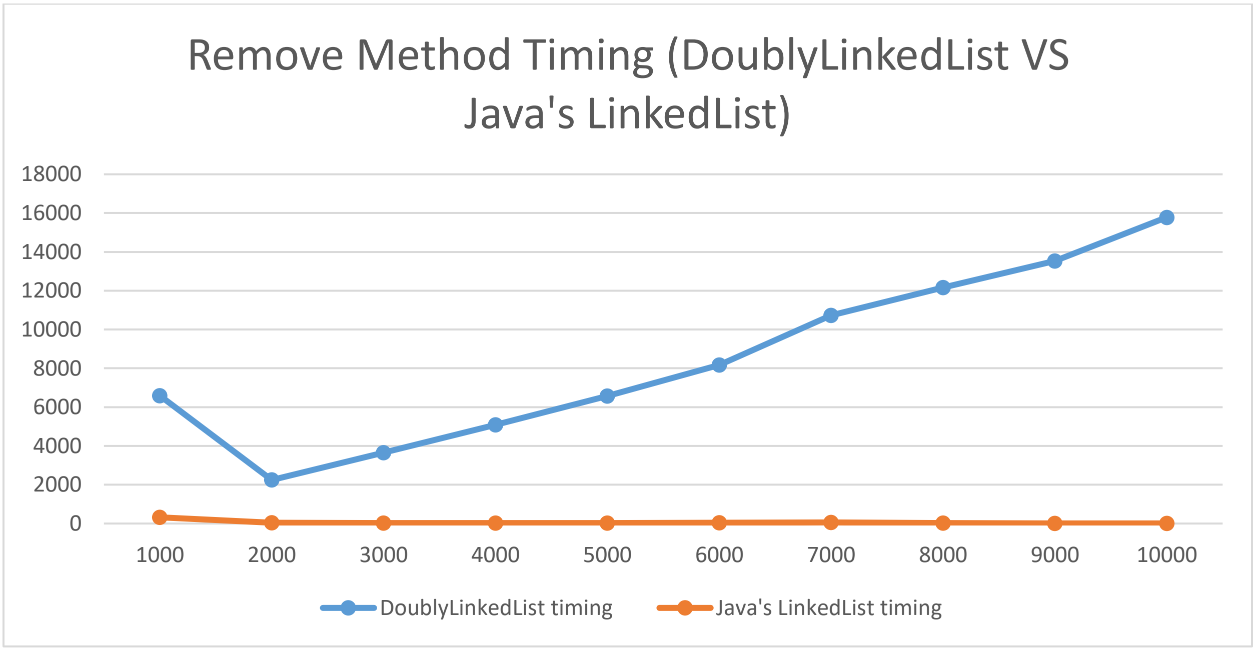
3. **In general, how does DoublyLinkedList compare to Java's LinkedList, both in functionality and performance?Please refer to Java's ArrayList documentation.**



In the addfirst method, DoublyLinkedList perform better than Java's Linked List. I think the Big-O of DoublyLinkedList and Java's LinkedList are kind of the same. However, the data is not very accurate. So there are some exceptions. To conclude, the Big-O of the Doubly-Linked List and Java's LinkedList are constant.



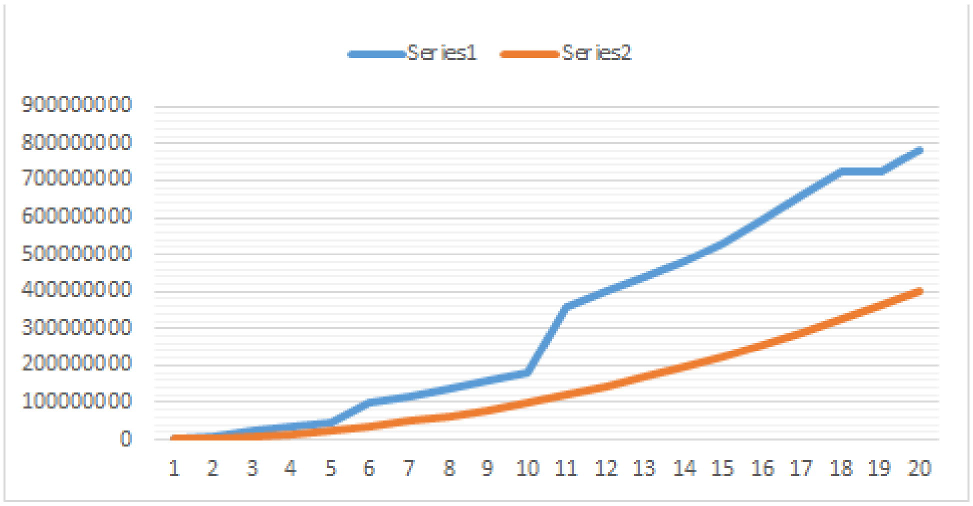
In get method, the Big-O of the Big-O of DoublyLinkedList and Java's LinkedList are very similar. However, there is some tiny difference between them. To conclude, the Big-O of the Doubly-Linked List and Java's LinkedList are O(N).

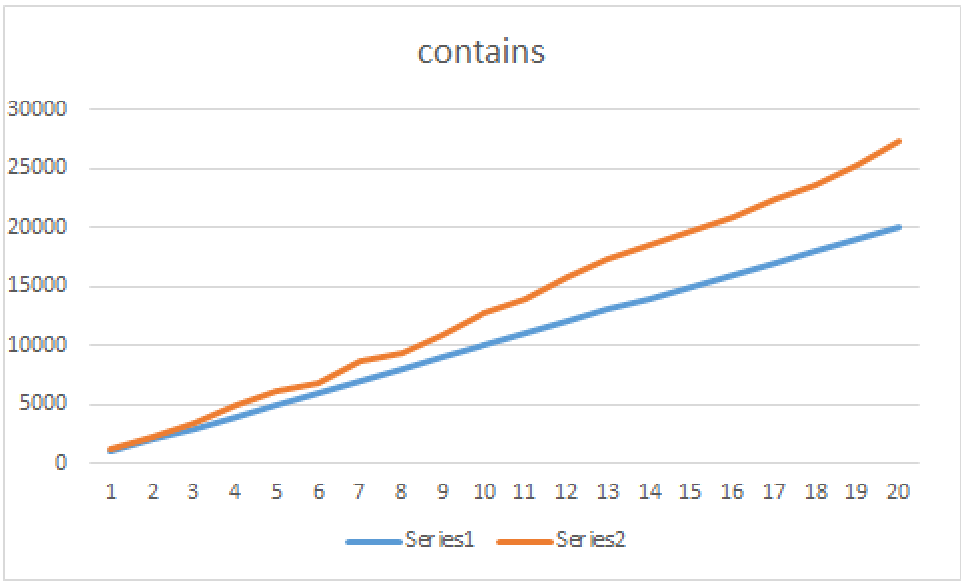


From the above graphs, the Big-O of the Doubly-Linked List is O(N) and Java’s LinkedList is a constant.

I believe that Java’s LinkedList perform better than the DoublyLinkedList. And Java’s LinkedList have more functions to choose, such as addAll(). However, I also can implement this method by ourselves. In some special case, Java’s LinkedList will automatically change its functions in order to performing better. To conclude, Java’s LinkedList perform better.

4. **Compre and contrast using a LinkedList vs an ArrayList as the backing data structure for the BinarySearchSet (Assignment 3). Would the Big-Oh complexity change on add / remove / contains?**





In the BinarySearchSet, add method perform a O(N^2) and contains method perform a O(N). I think the complexity will change. Since according to the API, LinkedList and ArrayList sometimes will choose a better function in the worst case during the search. The Big-O of LinkedList in adding is constant and The Big-O of ArrayList in adding is O(N). In the remove method , and contains, they perform the same. Since the Remove Method Timing (DoublyLinkedList VS Java's LinkedList) graph and the Remove Method Timing (DoublyLinkedList VS Java's ArrayList) graph show that Java’s LinkedList and ArrayList are the same complexity which is a constant.

5. **In general, how does DoublyLinkedList compare to a Singly Linked List, both in functionality and performance? (There isn't any Java**

documentation for a SinglyLinkedList because it doesn't use one!, So you will have to use your science brain to think about what is different.)

According to the Wikipedia, I find the result for this question. The difference between singly-linked list and doubly-linked list is that singly-linked list uses only one direction, but the doubly-linked list uses two directions and doubly-linked list costs more memory, in other word, it will occupy more memory to run. So Singly-Linked List will perform better. (“The linked-list implementation is equally simple and straightforward. In fact, a simple singly linked list is sufficient to implement a stack—it only requires that the head node or element can be removed, or popped, and a node can only be inserted by becoming the new head node.” (http://en.wikipedia.org/wiki/Stack_(abstract_data_type)#Linked_list)

6. **How many hours did you spend on this assignment?**

15 hrs.