GTU Department of Computer Engineering
CSE 222/505 - Spring 2022
Homework 6
Due date: April 24, 2022– 23:55

Q1. Implement KWHashMap interface in the book using the following hashing methods to organize the hash table:

1. Implement the chaining technique for hashing. However, use binary search trees to chain items mapped on the same table slot.
2. Implement a hashing technique that is a combination of the double hashing and coalesced hashing techniques (which is a hybrid of open addressing and chaining).
    a. Research about coalesced hashing. Explain the method and its advantages and disadvantages over standard open addressing and chaining methods in your report.
    b. Research about double hashing. Explain the method and its advantages and disadvantages over standard open addressing and chaining methods in your report.

    In our hybrid method, during an insertion operation, the probe positions for the colliding item are calculated by using the double hashing function. Moreover, the colliding items are linked to each other through the pointers as in the coalesced hashing technique. Use the following hash function to calculate probe positions.

    - tabelsize = determined dynamically. Note that it should be prime and should be increased twice if the load factor is above the threshold. (10 in the example)
    - Prime_number = the largest prime number smaller than 0.8*tablesize (7 in the example)
    - Hash1 = key % tablesize (10 in our case)
    - Hash2 = Prime_number – (key % Prime_number)
    - Hash function = (Hash1 + (*i* * Hash2)) % tablesize for the *ith* probe.

    Note that you should use the links during the search operation. The deletion of a key is performed by linking its next entry to the entry that points to the deleted key. If the deleted element is found without any probe, you should replace deleted entry with the next entry. As an example, see the illustrations below to insert 3, 12, 13, 25, 23, 51, and delete 25.

3. Test and compare your hash table implementations empirically. Use 100 randomly generated data sets for each three different set sizes (small (size = 100), medium (size = 1000), and large (size = 10000)). Insert the elements in a data set into initially empty hash table. Perform different tasks over the tables to compare their performance results (like accessing existing/non-existing items or adding/removing items).

- tabelsize = determined dynamically. Note that it should be prime and should be increased twice if the load factor is above the threshold. (10 in the example)
- Prime_number = the largest prime number smaller than 0.8*tablesize (7 in the example)
- Hash1 = key % tablesize (10 in our case)
- Hash2 = Prime_number – (key % Prime_number)
- Hash function = (Hash1 + (*i* * Hash2)) % tablesize for the *ith* probe.

Insert 3:

Hash Value	Key	Next
0		NULL
1		NULL
2		NULL
3		NULL
4		NULL
5		NULL
6		NULL
7	3	NULL
8		NULL
9		NULL

Insert 12:

Hash Value	Key	Next
0		NULL
1		NULL
2		NULL
3		NULL
4	12	NULL
5		NULL
6		NULL
7	3	NULL
8		NULL
9		NULL

Insert 13:

Hash Value	Key	Next
0		NULL
1		NULL
2		NULL
3		NULL
4	12	5
5	13	NULL
6		NULL
7	3	NULL
8		NULL
9		NULL

Insert 25:

Hash Value	Key	Next
0		NULL
1		NULL
2		NULL
3		NULL
4	12	5
5	13	NULL
6		NULL
7	3	NULL
8	25	NULL
9		NULL

Insert 23:

Hash Value	Key	Next
0		NULL
1		NULL
2		NULL
3	23	NULL
4	12	5
5	13	NULL
6		NULL
7	3	NULL
8	25	3
9		NULL

Insert 51:

Hash Value	Key	Next
0		NULL
1		NULL
2		NULL
3	23	NULL
4	12	5
5	13	NULL
6	51	NULL
7	3	NULL
8	25	3
9		NULL

Delete 25:

Hash Value	Key	Next
0		NULL
1		NULL
2		NULL
3		NULL
4	12	5
5	13	NULL
6	51	NULL
7	3	NULL
8	23	NULL
9		NULL

Q2. You are asked to compare the sorting algorithms below both empirically and theoretically. Use 1000 randomly generated arrays for each problem size (small (size = 100), medium (size = 1000), and large (size = 10000)). Empirical results should be obtained as the average of the 1000 independent runs for each algorithm and problem size individually. Please note that you must use the same set of arrays for testing/comparing algorithms. Analyze the execution time complexity of these algorithms theoretically. Evaluate the consistency between empirical and theoretical analysis results of the algorithms and explain your analysis results by supporting them with tables and diagrams.

1. Implement and evaluate the MergeSort algorithm.
2. Implement and evaluate the QuickSort algorithm.
3. Implement and evaluate the new_sort algorithm. Use the following pseudo-code to implement new_sort.

```
# new_sort (array, head, tail)
{
        IF (head > tail)
                Return array;
        ELSE
        {
                MIN, MAX = min_max_finder(array, head, tail);
                swap(array[head], array[MIN]);
                swap(array[tail], array[MAX]);
                Return new_sort(array, head + 1, tail -1);
        }
}
```

* The min_max_finder() is a recursive function that returns the indices of minimum and maximum items between the given head and tail items in a single execution together. Don't implement this function by calling separate functions that find min and max items. Your recursive function should divide the problem into two almost equal size subproblems.

GENERAL RULES:

- For any question firstly use the forum on the MS Teams page, and then the contact TA.
- You can submit an assignment one day late and will be evaluated over sixty percent (%60).

TECHNICAL RULES:

- You must write a driver function that demonstrates all possible actions in your homework. For example, if you are asked to implement an array list and perform an iterative search on the list then, you must at least provide the following in the driver function:
    - Create an array list and add items to the list. Append items to the head, tail, and k^{th} index of the list.
    - Perform at least two different searches by using two items in the list and print the index of the items.
    - Perform another search with an item that isn't in the array list and inform the user that the item doesn't exist in the array list.
    - Delete an existing item from the list and repeat the searches.
    - Try to delete an item that is not on the array list and throw an exception for this situation.

  The driver function should run when the code file is executed.

- Implement clean code standards in your code;
    - Classes, methods, and variables names must be meaningful and related to the functionality.
    - Your functions and classes must be simple, general, reusable, and focus on one topic.
    - Use standard java code name conventions.

REPORT RULES:

- Add all javadoc documentations for classes, methods, variables …etc. All explanations must be meaningful and understandable.
- You should submit your homework code, Javadoc, and report to MS Teams in a "studentid_hw1.tar.gz" file.
- Use the given homework format including selected parts from the table below:

Detailed system requirements	✓
The Project use case diagrams	X
Class diagrams	✓
Other diagrams (extra points)	X
Problem solutions approach	✓
Test cases	✓
Running command and results	✓

GRADING :

- **No OOP design:** **-100**
- **No error handling:** **-50**
- No javadoc documentation: -50
- No report: -90
- Disobey restrictions: -100
- **Cheating:** **-200**
- Your solution is evaluated over 100 as your performance.

CONTACT :

- Teaching Assistant Ferda Abbasoğlu, ferdaabbasoglu@gtu.edu.tr