Homework 5 Solutions
Due: Wednesday 23 November 2022 by 11:59 PM

Instructions. Type your answers to the following questions and submit as a PDF on Gradescope by the due date and time listed above. (You may write your solutions by hand, but remember that it is at your own risk as illegible solutions will not receive credit.) Assign pages to questions when you submit. **For all questions, you must show your work and/or provide a justification for your answers.**

Note on Academic Dishonesty. Although you are allowed to discuss these problems with other people, your work must be entirely your own. It is considered academic dishonesty to read anyone else's solution to any of these problems or to share your solution with another student or to look for solutions to these questions online. See the syllabus for more information on academic dishonesty.

Grading. Some of these questions may be graded for completion and some for accuracy.

For Questions 1-3, use the instructions below.
Let *T* be a hashtable of size *M*, and let the keys be integers using the hash function $h(k) = k \% M$. Let the initial value of *M* be 5. Show the resulting hashtable after the following operations are executed given each of the specifications in the individual problems. For each problem, you should show the *final* table, but you should also show the last intermediate table that occurs before any resizing is done.

Operations: put(24), put(21), put(1), put(9), put(91), put(100), put(140), put(143)

Note: The first prime number greater than 10 is 11.

Question 1 (Sourav–Accuracy)
T is implemented as an array of linked lists and uses separate chaining to handle collisions. If α exceeds 2, the table is resized to the first prime number greater than or equal to $2M$and items are re-hashed by iterating through each list. You can indicate lists with the elements in brackets (i.e. {1, 5, 6}).

0	1	2	3	4
{100, 140}	{21, 1, 91}		{143}	{24, 9}

Question 2 (Makayla–Accuracy)

T is implemented as an array and uses linear probing to handle collisions. If α exceeds $\frac{3}{4}$, the table is resized to the first prime number greater than or equal to $2M$and items are re-hashed in the order that they appear in the table. (Note that in general, it is recommended that resizing happen when the load factor exceeds ½, but that would be overly tedious to do by hand.)

Note on re-sizing: ¾ of 5 is 3.75, so you should resize after inserting the 4th item.

0	1	2	3	4
9	21	1		24

The table is resized to 11.

0	1	2	3	4	5	6	7	8	9	10
143	1	24	91	100				140	9	21

Question 3 (Ryn–Accuracy)

T is implemented as an array and uses quadratic probing to handle collisions. If α exceeds ¾ , the table is resized to the first prime number greater than or equal to $2M$ and items are re-hashed in the order that they appear in the table. There is also a limit of 5 probes per *put*, and if a space is not found within 5 probes, the table is resized and re-hashed as described above. (Note that in general, it is recommended that resizing happen when the load factor exceeds ½, but that would be overly tedious to do by hand.)

Note on re-sizing: ¾ of 5 is 3.75, so you should resize after inserting the 4th item or if you are unable to find an open spot after 5 probes.

0	1	2	3	4
9	21	1		24

The table is resized to 11.

0	1	2	3	4	5	6	7	8	9	10
143	1	24	91		100			140	9	21

Question 4. (Makayla–Completion)
Explain the role of probability in the construction of a skiplist and why it is important.

In a skiplist, the idea is to have nodes with different numbers of pointers so that we can speed up the search time by skipping over elements at "lower" levels. In the ideal situation, each new level would include half the nodes in the previous level so that we can always skip over half the nodes, making the searches a lot like binary search. But the ideal situation is not easy to enforce, so we assign levels based on those probabilities. We want ½ the nodes to be at level 0, so we assign that level with probability ½. We want ¼ of the nodes to be at level 1, so we assign that level with probability ¼, and so on…

Question 5. (Sourav–Completion)
Show the binary search tree, the 2-3 tree, and the LLRB tree that result after *each* of the following operations.

```
put(4), put(14), put(8), put(0), put(12), put(20), put(13), delete(4),
delete(8), put(10), delete(20), put(11)
```

Binary Search Tree: