CS317: Algorithms Nolan Anderson Homework Assignment #8

Due: See Canvas for Assignment Due Dates (20 points)

NOTE: NO LATE ASSIGNMENTS WILL BE ACCEPTED because we often review them in class on the due date.

Please upload the document containing your answers. They can be handwritten and scanned, but they must be clearly legible to receive a grade on the assignment. PDF is the best format for canvas. You DO NOT Need to include this cover sheet in your upload. It is formatted for the grader to use for me, as needed.

Chapter 7: Work the following problems. Point values are provided for each problem.

Problem #	Points	Grader's Notes
Sec 7.3, #3	*1*	
Sec 7.3, #7	*2*	
Sec 7.4, #4	*2*	
Sec 7.4, #5	*1*	

Chapter 12: Work the following problems. Point values are provided for each problem.

Problem #	Points	Grader's Notes
Sec. 12.2, #1	*1*	
Sec. 12.2, #5	*3*	

End of Semester Question (10 points)

Below is a link to an article in Medium. Please read the article. Then write one to two paragraphs answering the following:

Why is lifelong learning important in the software development career field? Where does *Algorithms* fit into your professional development? How would you continue to engage in continuing to enhance and use your understanding of the topics presented in this course?

https://medium.com/better-programming/the-one-programming-language-to-rule-them-all-620366df2805

A note - please write complete paragraphs using good form (topic sentence, correct grammar, etc). Communication skills are also important to software developers.

7.3

3. Why is it not a good idea for a hash function to depend on just one letter (say, the first one) of a natural-language word?

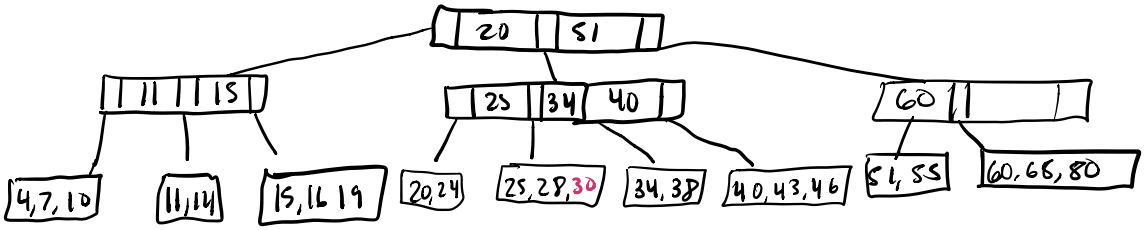
If you only used the first letter, the number of values calculated by the function would be limited. Also most words don't start with the same letter.

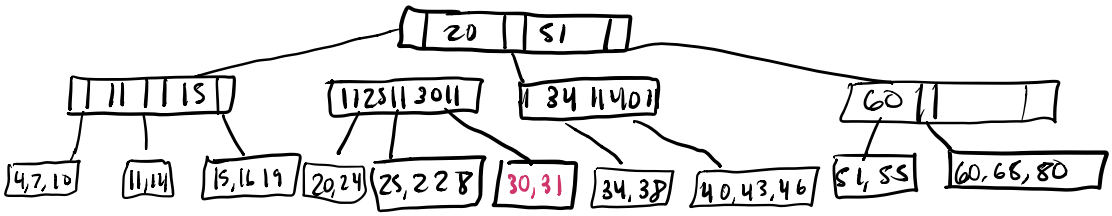
7. Explain how to use hashing to check whether all elements of a list are distinct. What is the time efficiency of this application? Compare its efficiency with that of the brute-force algorithm (Section 2.3) and of the presorting-based algorithm (Section 6.1).

- You could do this by inserting the elements of the list one by one. If there is an element that matches a previous one that was inserted, then the elements are not distinct. Run until list is empty otherwise.
- Worst case would be moving all elements to the hash or the last element is a copy. $\Theta(n^2)$. Comparisons $\Theta(n^2)$. Average case is $\Theta(n)$
- Brute force takes $\Theta(n^2)$
- Presorting $O(n \log n)$-> can be done in-place.

7.4

4. Draw the B-tree obtained after inserting 30 and then 31 in the B-tree in Figure 7.8. Assume that a leaf cannot contain more than three items.





5. Outline an algorithm for finding the largest key in a B-tree.

Step one would be to start at the root. Then all you would need to do is follow the chain of pointers on the right. When you get to the last key, that would be the max value.

12.2

1. What data structure would you use to keep track of live nodes in a best-first branch-and-bound algorithm?

For maximization use a heap.
For minimization use a min-heap.

5. Solve the following instance of the knapsack problem by the branch-and-bound algorithm:

item	weight	value	
1	10	$100	
2	7	$63	$W = 16$
3	8	$56	
4	4	$12