CS251 Homework 2: Sorting and Hashing

Out: Feb 2, 2018 @ 9:00 PM

Due: Feb 9, 2018 @ 11:59 PM

Submission Instructions: Please submit a typeset PDF on blackboard.

Short-Answer Questions:

1. What are the worst-case time complexities of 'insert' and 'find' operations in a dictionary ADT, where n is the size of the dictionary?

a) O(1) & O(1)

b) O(n) & O(n)

c) O(1) & O(n)

d) O(n) & O(1)

The answer is C. In an unsorted list the insert function would be of constant time because it would simply insert the entry at the end of the list. The find method would take O(n) in the worst case because you would need to do a sequential search.

2. Which of the following sorting algorithms has the lowest worst-case time complexity?

a) Selection sort

b) Insertion sort

c) Bubble sort

d) Heap sort

The answer is D. A selection sort algorithm has worst case $O(n^2)$ time complexity. An insertion sort algorithm has worst case $O(n^2)$ time complexity. A bubble sort algorithm has worst case $O(n^2)$ time complexity. And heap sort algorithm has worst case O(n logn) time complexity, which is the lowest worst case time complexity.

3. Which of the following sorting algorithms is not stable (i.e., the initial order of equal keys is maintained)?

a) Selection sort

b) Insertion sort

c) Bubble sort

d) All sorting algorithms are stable

The answer is A. This is because in a Selection sort it works by finding the least value in a set of values and swapping it with the first value. So if there are two values that are the same then it will change the order of elements.

4. Suppose you have the following hash table, implemented using linear probing. The hash function we are using is the identity function, h(x) = x % 7.

0	1	2	3	4	5	6
14	21	9	3	4	12	18

Which of the following is/are a possible order of insertion into the hash table? There may be multiple correct answers. Select all that apply.

a) 9, 14, 4, 18, 12, 3, 21

b) 12, 3, 14, 18, 4, 9, 21

c) 12, 14, 3, 9, 4, 18, 21

d) 9, 12, 14, 3, 4, 21, 18

e) 12, 9, 18, 3, 14, 21, 4

The answer is C and D. For A it would not work because 18 would go into index 5 instead of 6. B would not work because 18 would go into index 4 instead of 6. E would also not work because 18 would go into index 4 instead of 6.

5. What is the worst case running time of search with linear probing?

a) O(1)

b) O(n)

c) O(log n)

d) O($\sqrt{n}$)

The answer is B. This is because the average case is constant at O(1) but in the worst case it will have to go over every element which would be O(n).

6. What are the ‘expected running time of insertion’ and ‘worst case running time of deletion’, in a hash table with quadratic probing to resolve collisions?

a) O(log n) and O($\sqrt{n}$)

b) O(1) and O(n)

c) O($\sqrt{n}$) and O($\sqrt{n}$)

d) O($\sqrt{n}$) and O(n)

The answer is A. This is because the expected runtime of insertion is O(log n) and the worst case of deletion is O($\sqrt{n}$).

7. Using alphabet position, encode 'science' with the following hash map

$p(z) = a_0 + a_1 z + a_2 z^2 + \ldots + a_{n-1} z^{n-1}$ with z=19

$p(z) = 19 + (3)(19) + (8)(19)^2 + (5)(19)^3 + (14)(19)^4 + (3)(19)^5 + (5)(19)^6$

$= 19 + 57 + 2{,}888 + 34{,}295 + 1{,}824{,}494 + 7{,}428{,}297 + 141{,}137{,}643$

$= 150{,}427{,}693$

8. Which of the following sorting algorithms takes linear time in the best case?

a. Selection sort

b. Insertion sort

c. Bubble sort

d. Heap sort

The answer is B. This is because while even though both Insertion sort and bubble sort both provide linear time in the best case, but in insertion sort there are less iterations through the list which results in a faster algorithm.

9. A heap with n keys has a height of:

a. Θ (n)

b. Θ (log n)

c. Θ (n log n)

d. (log2 Θ n)

The answer is B. We can prove this is the case by letting a variable h be the height of the heap and having at least one key at depth h. With that we get $n \geq 1 + 2 + 4 + \ldots + 2^{h-1} + 1$. As a result we get $n \geq 2^h$ which is $h \leq \log n$.

10. Time complexity of getMin() and removeMin() operations on a min-heap are respectively

a. Θ (1) & Θ (log n)

b. Θ (log n) & Θ (log n)

c. Θ (log n) & Θ (1)

d. Θ (1) & Θ (n)

The answer is A. This is because get min takes constant time to go through a heap and remove min take O(log n) time because it only has to go through part of a heap and not through all the elements.

Long-Answer Questions:

11. Given two sets A and B represented as sorted sequences, describe an efficient algorithm for computing A$\oplus$B, which is the set of elements that are in A or B, but not in both.

Since the two sets are already sorted then first we must go through A and B separately and remove all duplicates. Then we can merge the two sets into one and then do a linear scan for duplicates, and if there are duplicates then the A$\oplus$B is false.

12. Let S be a random permutation of n distinct integers. Argue that the expected running time of insertion-sort on S is Ω (n^2). (Hint: Note that half of the elements ranked in the top half of a sorted version of S are expected to be in the first half of S.)

The expected running time for insertion-sort on S would be Ω (n^2). This is due to the fact that the average case for insertion-sort is $\Theta(n^2)$ and the worst case is $O(n^2)$. Taking those into consideration and the fact that Ω is a lower bound then we can see that the running time must be at least Ω (n^2).

13. Come up a compression map (function) that results in no collisions for the following data for a 7-entry hash table: 55, 45, 12, 1, 4, 23, 7. Draw the resulting hash table.

h(i) = i % 7

55%7 = 6

45%7 = 3

12%7 = 5

1%7 = 1

4%7 = 4

23%7 = 2

7%7 = 0

0	1	2	3	4	5	6
7	1	23	45	4	12	55

14.Draw the 7-entry hash table that results from hash function, h(i) = (2i+5)%7, to hash the keys 12, 34, 13, 78, 21, 91, 11, 39, 10, 6, and 15, assuming collisions are handled by chaining.

(2(12) + 5)%7 = 1

(2(34) + 5)%7 = 3

(2(13) + 5)%7 = 3 goes to next available slot which is 4

(2(78) + 5)%7 = 0

(2(21) + 5)%7 = 5

(2(91) + 5)%7 = 5 goes to next available slot which is 6

(2(11) + 5)%7 = 6 goes to next available slot which is 7

(2(39) + 5)%7 = 6 goes to next available slot which is 8

(2(10) + 5)%7 = 4 goes to next available slot which is 9

(2(6) + 5)%7 = 3 goes to next available slot which is 10

(2(15) + 5)%7 = 0 goes to next available slot which is 2

0	1	2	3	4	5	6	7	8	9	10
78	12	15	34	13	21	91	11	11	10	6

15. Does a preorder traversal of a heap list its keys in non-decreasing order? Argue that it is right or give a counterexample to disprove.

Imagine a heap represented by a list [1,3,2,6,7,5,4]. This heap does not produce keys in non-decreasing order.