University at Buffalo — CSE-250 — Fall 2023

Written Assignment 2: PA1 Reflection

Due: Sunday Oct 01, 2023 before 11:59 PM

Expect this assignment to take 4-6 hours. The total point value of all 3 problems in this assignment is 100. This assignment is worth 5% of your overall grade.

Your written solution may be either handwritten and scanned or typeset. Either way, you must produce a PDF that is legible and displays reasonably on a typical PDF reader. This PDF should be submitted via autolab. You should view your submission after you upload it to make sure that it is not corrupted or malformed. **Submissions that are rotated, upside-down, or that do not load will not receive credit**. Illegible submissions may also lose credit depending on what can be read.

Problem 1: Sorted Lists (20 points)

For **PA1**, you implemented a list that stored its elements in sorted order.

a. **[5 pt.]** What is the unqualified worst-case complexity of finding an element in a regular linked list (one in which items aren't stored in sorted order)?

b. **[5 pt.]** What is the unqualified worst-case complexity of finding an element in a linked list in which items are stored in sorted order?

c. **[10 pt.]** Depending on your answers to questions 1 and 2, explain why storing elements in sorted order **does** or **does not** affect the asymptotic complexity of finding an element.

Problem 2: Hinted Search (30 points)

For **PA1**, your linked list included hinted versions of the `find` and `get` methods. These variants gave the methods a different place to start their search from rather than the `headNode`.

Imagine that we are using your `SortedList` as part of a new data structure consisting of the list and a 16-element Array called `hints`. When the `SortedList` is first initialized, we initialize `hints[0]` to the 0th element of the list, `hints[1]` to the $\frac{N}{16}$th element of the list, `hints[2]` to the $2\frac{N}{16}$th element of the list, `hints[3]` to the $3\frac{N}{16}$th element of the list, and so forth.

To find an element of the list, we first do a binary search over `hints` to find the hint with the nearest element, and use this element for the hinted `find()` method.

a. **[5 pt.]** In **no more than one sentence**, how does the sorted nature of the list affect our ability to use hinted searhces?

b. **[5 pt.]** What is the asymptotic complexity of retrieving hints from the `hints` array.

c. **[10 pt.]** What is the unqualified worst-case (Big-O) complexity of finding an element in our list when using a hint retrieved from the `hints` array?

d. **[10 pt.]** If the answer is different from the answer you gave in Problem 1.**b.**, explain how the use of hints changes the complexity class of our search. If the answer is the same as the answer you gave in Problem 1.**b.**, then you claim the runtime of the two search methods only differs by a constant factor. What is this constant factor and why?

Problem 3: Duplicate Elements (25 points)

For **PA1**, your linked list handled duplicate elements by storing them all within a single node with a count member to keep track of how many elements with that value have been added to the list.

a. **[5 pt.]** How does the sorted nature of the list affect our ability to treat duplicate elements specially?

b. **[5 pt.]** Assume we are using the `SortedList` from **PA1** to store a list of score totals from soccer[1] games; Each element is the total number of points scored during a match. The highest-scoring game ever involved 149 points; You should assume that this is an upper bound on the number of points possible in a match. What is the unqualified worst-case (Big-O) runtime to find the node for a particular score in a list storing results from N matches.

[1] Football, to our international friends.

c. **[5 pt.]** Is the runtime asymptotically different if we create a new node for every element? In at most one sentence, explain why or why not.

d. **[5 pt.]** If we use the `SortedList` from **PA1** to store a list of player names, where we can assume that there are an infinite number of possible first names, what is the unqualified worst-case runtime to find the node for a particular name?

e. **[5 pt.]** Is the runtime asymptotically different if we create a new node for every element? In at most one sentence, explain why or why not.

Problem 4: Asymptotic Complexity (25 points)

For each of the following claims, use the inequality definition of Big-O, Big-Ω, or Big-θ to either prove or disprove the claim. For each question, your answer must show, using the inequalities equivalent to the claim, and the rules for solving inequalities that we have discussed in class, one of the following:

- ... that there exists a constant (write down such a constant) for which the inequality(ies) must hold for a sufficiently large N; **OR**
- ... that the inequality(ies) does not hold for any constant and large values of N (e.g., by reducing the inequality to an invalid inequality *e.g.*, $c > N$).

$$f_1(N) = 23 \cdot N^2 + N \log(20 \cdot N)$$

$$f_2(N) = \begin{cases} N & \textbf{if } N = 0 \mod 2 \\ N^2 & \textbf{otherwise} \end{cases}$$

a. **[5 pt.]** $f_1(N) \in \Omega(N)$

b. **[5 pt.]** $f_1(N) \in O(N^2)$

c. **[5 pt.]** $f_1(N) \in \theta(N^2)$

d. **[5 pt.]** $f_2(N) \in O(N^2)$

e. **[5 pt.]** $f_2(N) \in \theta(N^2)$