CSCB63 Assignment 1- Summer 2022

60 Points

Due Date: 10th June 2022 11:59 pm

- Only a subset of the questions will be graded however you are responsible for all the material on this assignment.
- Read the guidelines for academic integrity here:
  - https://www.utsc.utoronto.ca/aacc/sites/utsc.utoronto.ca.aacc/files/tipsheets/AIM%20-%20Tipsheet%20oct%202015.pdf
- Please note that all your answers and code will be thoroughly checked for any similarity with other students both present and past.

Question 1: Asymptotic Growth (7 points)

Sort all the functions below in increasing order of asymptotic $(big - O)$ growth. If some have the same asymptotic growth, then be sure to indicate that. lg means base 2.

1. $5n$
2. $4\ lg\ n$
3. $4 \lg \lg n$
4. n^4
5. $(lg\ n)^{5lg\ n}$
6. $n^{\lg n}$
7. 5^n
8. 4^{n^4}
9. 4^{4^n}
10. 5^{5^n}
11. 5^{5n}
12. $n^{\,n^{1/5}}$
13. $n^{\,n/4}$
14. $(n/4)^{n/4}$

Question 2: (8 points)

We often see time (or space) complexity written as a recurrence relation (think of any recursive algorithm).

To determine the complexity in asymptotic notation $(O(\cdot), \Omega(\cdot), \Theta(\cdot))$ we solve the recurrence relation as a closed form function. A recurrence relation is a function $f(n)$ defined in terms of itself. A closed form function $f(n)$ is defined in terms of n. There are two common ways to do this; repeated back substitution (which we saw in lecture) or apply the Master Theorem. Solving complexity recurrence relations is an important skill you will need in CSCC73.

We will illustrate both methods for the time complexity recurrence of Merge Sort, $T(n)$, which can be expressed as

$T(n) = 2T(\frac{n}{2}) + n \; and \; T(1) = 0$

Repeated back substitution is the process of repeatedly substituting into the recurrence. For simplicity assume n is a power of 2.

$T(n) = 2T(\frac{n}{2}) + n$ and observe that $T\left(\frac{n}{2}\right) = 2T\left(\frac{n}{4}\right) + \frac{n}{2}$ so we can replace the $T\left(\frac{n}{2}\right)$ with $2T\left(\frac{n}{4}\right) + \frac{n}{2}$

$T(n) = 2\left(2T\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n$ and applying the same concept again gives:

$T(n) = 2(2(2T(\frac{n}{8}) + \frac{n}{4}) + \frac{n}{2}) + n$. Repeating until $\frac{n}{2^i} = 1$ i.e., until $\mathrm{i} = log_2 \mathrm{n}$.

$$T(n) = \sum_{i=0}^{log_2 n} 2^i \frac{n}{2^i}$$

Simplifying the summation gives

$$T(n) = \sum_{i=0}^{log_2 n} n = n \, log_2 n$$

So Merge Sort's asymptotic complexity $T(n) \in \Theta(n \, log_2 n)$.

If your recurrence has the correct format, you can apply the Master Theorem to find the closed form:

Generalized Master Theorem

Let a and b be positive real numbers with $a \geq 1 \; and \; b \geq 2$.

Let $T(n)$ be defined by

$$T(n) = \begin{cases} aT\left(\left\lceil \frac{n}{b} \right\rceil\right) + f(n) & if \; n > 1 \\ d & if \; n = 1 \end{cases}$$

Then

$(a) \; if \; f(n) = \Theta(n^c) \; where \; log_b a < c, then \; T(n) = \Theta(n^c) = \Theta(f(n))$.

$(b) \; if \; f(n) = \Theta(n^c), where \; log_b a = c, then \; T(n) = \Theta(n^{log_b a} \; log_b n)$.

$(c) \; if \; f(n) = \Theta(n^c), where \; log_b a > c, then \; T(n) = \Theta(n^{log_b a})$.

$The \; same \; results \; apply \; with \; ceilings \; replaced \; by \; floors$

For example, the run time of Mergesort, $T(n)$, can be expressed as

$$T(n) = 2T(\frac{n}{2}) + n$$

Applying the Master Theorem produces

$$T(n) = = \Theta\left(n^{\log_b a} \log n\right) = \Theta\left(n^{\log_1 1} \log n\right)$$

For each of the following determine the asymptotic bound $\Theta(n)$ for the recurrence relation.

$(a)\ A(n) = n^2 A(n-1)\ for\ n > 1\ and\ A(1) = 1.$

$(b)\ A(n) = 8A(\lfloor n/2 \rfloor) + 2n^3 + 4n, where\ A(0) = 6.$

You can give your answer in Θ notation.

Question 3: (8 points)

Let f and g be functions such that $f(n) \geq 1$ and $lg(g(n)) \geq 1$ for all natural n.

Prove: If $f(n) \in O(g(n))$, then $lg(f(n)) \in O(lg(g(n)))$.

Question 4: (8 points)

Consider inserting the following keys in order into an AVL tree. Use the convention from lecture of calculating the balance factor = height of left subtree minus height of right subtree.

Only show your tree at the indicated times.

5, 4, 6, 9, 12, 16, 14, 2, 3

Show your tree.

Now insert:

1, 20, 19, 18, 17

Show your tree.

Now using the predecessor when there is a choice, delete in order

1, 2, 3, 4, 5, 12, 6

Show your tree.

Question 5: (10 points)

Consider Insertion Sort and Merge Sort. Insertion sort requires $c.n^2$ to sort n numbers. To merge m subarrays (which are sorted) that contain n items together you have to compare the m top items in all the subarrays to choose the current maximum item and then place it into the sorted array. You may assume that all the items need to be sorted in ascending order.

Therefore, then merging time is proportional to $(m-1).n$. Assume the time required for merging is $c.(m-1).n$ where the constant c has the same value for insertion sort. Consider following possibility to accelerate the sorting time for n items.

- The initial array of size n is split into m subarrays of size (n/m). Assume they are of equal sizes
- Each subarray of size (n/m) is sorted separately using Insertion Sort.
- Sorted subarrays are merged into final sorted array.

If you are convinced that this acceleration is possible, find the optimum value of m and compare the resulting complexity with Insertion Sort and with Merge Sort.

Question 6: (9 points)

Consider an abstract datatype RECENT that keeps track of recently accessed values.

In this ADT, we consider a set $S = \{1, \ldots, n\}$ and a subset R of S, of size m. Initially, $R = \emptyset$. The only operation is access, which takes a parameter $i \in S$. It adds i to R and, if this causes the size of R to become bigger than m, removes the element from R that was least recently the parameter of an access operation.

Give a data structure for this abstract data type that uses $O(m \log n)$ space (measured in bits) and has worst case time complexity $O(\log m)$.

Justify that your data structure is correct and satisfies the desired complexity bounds.

Question 7: Programming question (10 points)

warm up before attempting programming question.

To enable you to solve the programming question, answer the following question first.

It will ***not be graded*** but will make **coding portion (which is graded**) much easier. Starter code (in C) is available on MarkUs.

Consider an ADT consisting of a set S of distinct integers and the following operations:

INSERT (S, x): Insert the element x into the set S. This operation has no effect if x is already in S.

DELETE (S, x): Delete the element x from the set S. This operation has no effect if x is not in S.

QUERY (S, x): Return true if x is in S and return false if x is not in S.

CLOSEST-PAIR (S): Return two integers in S which are closest together in value.

In other words, if CLOSEST-PAIR(S) returns the integers a and b, then they must satisfy the condition.

$$\forall x \forall y (x \neq y \rightarrow |a - b| \leq |x - y|).$$

It is an error if S contains fewer than two elements.

Describe a data structure to implement this ADT. All the usual operations of INSERT, DELETE and QUERY must be performed in $O(\log n)$ time, where $n = |S|$ and CLOSEST-PAIR must be in $O(1)$.

Describe any new information that will be stored. Justify why your algorithms are correct and why they achieve the required time bound.

Submission Instructions:

Your assignment must be typed; handwritten assignments **will not be marked**. You may use any word-processing software you like. Many academics use LaTeX. It produces beautifully typeset text and handles mathematical notation well. If you would like to learn LaTeX, there are helpful resources online.

Whatever you choose to use, you need to produce a **final document in pdf format**. You must hand in your work electronically using the MarkUs online system.

For this assignment, hand in: **A1.pdf (questions 1 to 6)**

closest.c (question 7)

Check that you have submitted the correct version of your file by downloading it from MarkUs. Late files will be accepted with 10% penalty for each day after the due date up to 3 days. Submissions after that will not be accepted or marked.