[CS202]-Algorithms I **Fall 2022-23**

HOMEWORK I

Due Date: **23:59, March 13, 2023**.

Instructions.

1. Submissions will not be accepted after the due date. Discussion is encouraged but you need to write out your solutions independently.
2. The answers must be typed in a given LaTeX template and submit as pdf (or ZIP) file on canvas. File name should be Coursecode-HW1-Rollno, for example if the rollno of a student is 11840170 then file name should be CS202-HW1-11840170.
3. Submissions not following the guidelines will NOT be evaluated.
4. Homework will be evaluated based on (a) HW submission and (b) a short class quiz, giving equal weightage to both of them.
5. Most of the problems below appear on many online sites and text books. Copying from any sources without attribution is a violation of academic honesty.

---

1. Define the following terms with an example (a) Time complexity of an algorithm (b) Worst case running time of an algorithm (c) Best case running time of an algorithm.
2. For the following pseudo-code, calculate the running time by drawing a table (consisting costs and number of times a line is executed) and finding an exact expression for the running time using the table. Finally compute the the value returned by the pseudo code (Give an exact expression).

```
1: SAMPLE(n)
2: s = 3
3: for i = 1 to n do
4:     for j = i to n do
5:         s = s^3
6:     end for
7: end for
8: return s
```

3. Sort the following functions in increasing order of asymptotic (big-O) growth. If some have the same asymptotic growth, then indicate that.

   1.01^n, $4n$, 5^n, $4\log n$, $4n\log n$, $(\frac{3}{2})^n$, $n^{\log n}$, $\sqrt{n}\log n$, n^2, $2^{\log n}$, $n!$

4. Consider each of the following statements, assuming that all functions are non-negative. For each statement, if the statement is true then provide a proof from the scratch. If the statement is false then provide a counter example and demonstrate why the statement is false.

   (a) If $f_1(n) = \Theta(g(n))$ and $f_2(n) = \Theta(g(n))$, then $f_1(n) - f_2(n) = O(1)$.

(b) If $f(n) = O(g(n))$ then $f(n) + g(n) = \Theta(g(n))$.

(c) If $f(n) = O(g(n))$ then $g(n) = \Omega(f(n))$.

(d) For any constant $b > 1$ the function $f(n) = 1 + b + b^2 + \ldots + b^n$ is in $\Theta(b^n)$.

5. Consider the function h given by $h(n) = \log(n!) = \sum_{i=1}^{n} \log(i)$.

   (a) TRUE/FALSE: $h(n) = O(\log(n))$.

   (b) TRUE/FALSE: $h(n) = \Omega(n \log(n))$.

   (c) TRUE/FALSE: $h(n) = o(n^2)$

   Justify your answers.

6. Solve the following recurrence relations using Master theorem

   (a) $T(n) = 16T(n/2) + n^3$

   (b) $T(n) = 4T(n/2) + n^2$

   (c) $T(n) = 3T(n/2) + n^2$

   (d) $T(n) = 2T(n/4) + n^{0.58}$

7. (a) Solve the following recurrence relation using the recursion tree method.

$$T(n) = 3T(n/4) + \Theta(n^2)$$

   (b) Can the master method be applied to the recurrence $T(n) = 8T(n/2) + n^3 \log n$? Why or why not? Give an asymptotic upper bound for this recurrence.

8. One way to rank researchers is by their "h- index". A researcher's h-index is the maximum integer k such that the researcher has at least k papers that have been cited at least k times each. Suppose a researcher X has written n papers and paper i has been cited c_i times. Suppose you have these sorted in an array C with $c_1 \geq c_2 \geq \ldots \geq c_n$. Give a divide-and-conquer algorithm to find researcher X's h-index. Your algorithm must run in time $\Theta(\log n)$.

9. Suppose you are given two arrays $A[1 \ldots n]$, $B[1 \ldots n]$ of elements. You are required to find the pair of elements $A[i]$, $B[j]$ such that their absolute difference is the smallest over all such pairs.

   (a) Describe briefly the brute force algorithm and state and justify informally its running time.

   (b) A better solution is to use a variant of the Merge algorithm from MergeSort. This solution is avilable on many web sites, but try to not look at those solutions.

   Sort the arrays in increasing order. Compare the two first elements and note the difference. Then, decide which list to "remove" the minimum element from. Try some examples and come up with a strategy that you think works. The algorithm should be to keep applying this strategy and always compare the absolute difference between the two minimum values in A, B to a running minimum difference and updating the minimum difference if necessary until all elements in A and B have been "removed". Write the above algorithm in pseudocode and prove its correctness. Also state and justify informally the running time of this algorithm.

10. In your favorite programming language, implement:

    (a) the Insertion Sort algorithm.

    (b) the Merge Sort algorithm.

    (c) the Quick Sort algorithm.

(d) the HeapSort algorithm (use Max-Heap)

Run the above four sorting algorithms on input of sizes 10, 20, 50, 100, 200, 500, 800, 1000, 5000 and 10000.

For every size run your algorithm at least three times and take the average time.

You should plot results as follows: on the X-axis is the length of the array input. The Y-axis is the measured runtime (average) in milliseconds.

For all the four sorting algorithms, briefly discuss whether their running times grow at the rate you would expect, given their theoretical asymptotic complexity, or grow faster/slower.