DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING
CONCORDIA UNIVERSITY
COMP 6651 Algorithm Design Techniques
Fall 2024
ASSIGNMENT 1
Due: Monday, September 23

---

1. **[20 marks]** Let $f(n)$ and $g(n)$ be asymptotically positive functions. Prove or disprove each of the following conjectures.
   (a) $f(n) = O(g(n))$ implies $g(n) = O(f(n))$.
   (b) $f(n) = O((f(n))^2)$.
   (c) $f(n) = \Theta(f(n/2))$.
   (d) $f(n) + o(f(n)) = \Theta(f(n))$.
2. **[10 marks]** The solution to the recurrence $T(n) = 4T(n/2)+n$ turns out to be $T(n) = \Theta(n^2)$.
   (a) Show that a substitution proof with the assumption $T(n) \leq cn^2$ fails.
   (b) Then show how to subtract a lower-order term to make a substitution proof work.
3. **[10 marks]** For each of the following recurrences, sketch its recursion tree, and guess a good asymptotic upper bound on its solution. Then use the substitution method to verify your answer.
   (a) $T(n) = T(n/2) + n^3$.
   (b) $T(n) = 3T(n–1) + 1$.
4. **[10 marks]** Use the Akra-Bazzi method to solve the following recurrences.
   (a) $T(n) = T(n/2) + T(n/3) + T(n/6) + n \lg n$.
   (b) $T(n) = (1/3)T(n/3) + 1/n$.
5. **[20 marks]** Suppose you are consulting for a bank that is concerned about fraud detection, and they come to you with the following problem. They have a collection of n bank cards. Each bank card is a small plastic object, containing a smart chip with some encrypted data, and it corresponds to a unique account in the bank. Each bank can have many bank cards corresponding to it, and we will say that two bank cards are *equivalent* if they correspond to the same account.

   It is very difficult to read the account number off a bank card directly, but the bank has a high-tech "equivalence tester" that takes two bank cards and, after performing some computations, determines whether they are equivalent.

   Their question is the following: among the collection of n cards, is there a set of more than $n/2$ of them that are all equivalent to one another? Assume that the only feasible operations you can do with the cards are to pick two of them and plug them into the equivalence tester.

   (a) **[10 marks]** Give an algorithm to decide the answer to their question with only $O(n \log n)$ invocations of the equivalence tester.
   (b) Prove the correctness of your algorithm.

(c) Using a recurrence relation to define the worst-case running time in terms of the number of invocations of the equivalence tester, and prove the running time is $O(n \log n)$.

6. **[15 marks]** Professors Howard, Fine, and Howard have proposed a deceptively simple sorting algorithm, named stooge sort in their honor, appearing on the following page.

   (a) Argue that the call **STOOGE-SORT**$(A, 1, n)$ correctly sorts the array $A[1..n]$.

   (b) Give a recurrence for the worst-case running time of STOOGE-SORT and a tight asymptotic (Θ-notation) bound on the worst-case running time.

   (c) Compare the worst-case running time of **STOOGE-SORT** with that of insertion sort, merge sort, and quicksort. Do the professors deserve tenure?

```
STOOGE-SORT(A, p, r)
  if A[p] > A[r] then
     exchange A[p] with A[r]
  if p + 1 < r then
     k ← ⌊(r − p + 1)/3⌋                          ▷ round down
     STOOGE-SORT(A, p, r − k)                     ▷ first two-thirds
     STOOGE-SORT(A, p + k, r)                     ▷ last two-thirds
     STOOGE-SORT(A, p, r − k)                     ▷ first two-thirds again
```

7. **[15 marks]** You are interested in analyzing some hard-to-obtain data from two separate databases. Each database contains n numerical values - so there are $2n$ values total - and you may assume that no two values are the same. You would like to determine the median of this set of $2n$ values, which we will define to be the n^{th} smallest value.

   However, the only way you can access these values is through queries to the databases. In a single query, you can specify a value k to one of the databases and the chosen database will return the k^{th} smallest value that it contains. Since queries are expensive, you would like to compute the median using as few queries as possible.

   Given an algorithm that finds the median value using at most $O(\lg n)$ queries.