

**CMPS 201**  
**Midterm 1**  
**Review Problems**

1. Let  $T(n)$  satisfy the recurrence  $T(n) = aT(n/b) + f(n)$ , where  $a \geq 1$ ,  $b > 1$  and  $f(n)$  is a polynomial satisfying  $\deg(f) > \log_b(a)$ . Prove that case (3) of the Master Theorem applies, and in particular, prove that the regularity condition necessarily holds.

2. The  $n^{\text{th}}$  harmonic number is defined to be  $H_n = \sum_{k=1}^n \left(\frac{1}{k}\right) = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n-1} + \frac{1}{n}$ . Use induction to prove that

$$\sum_{k=1}^n H_k = (n+1)H_n - n$$

for all  $n \geq 1$ . (Hint: Use the fact that  $H_n = H_{n-1} + \frac{1}{n}$ .)

3. Define the sequence  $S_n$  by the recurrence  $S_n = (n-1) + \frac{n-1}{n^2} \cdot \sum_{k=1}^{n-1} S_k$ . Use induction to prove  $S_n \leq 2n$  for all  $n \geq 1$ .

4. The following sorting algorithm, called `BadSort()` is a modified version of `StoogeSort()` from the 2<sup>nd</sup> edition of CLRS, which seems to have been left out of the 3<sup>rd</sup> edition.

BadSort( $A, p, r$ ) pre:  $p \leq r$

1. if  $A[p] > A[r]$
2.      $A[p] \leftrightarrow A[r]$  (swap)
3. if  $p + 1 \geq r$
4.     return
5. else
6.      $q = \lfloor (r - p + 1)/3 \rfloor$
7.     BadSort( $A, p, r - q$ )
8.     BadSort( $A, p + q, r$ )
9.     BadSort( $A, p, r - q$ )

- a. Use induction on the length  $m = r - p + 1$  of  $A[p \dots r]$  to prove the correctness of `BadSort()`.
- b. Write a recurrence relation for the number of array comparisons performed by `BadSort()` on an array of length  $n$ .
- c. Use the Master Theorem to find an asymptotic solution to this recurrence, and explain what is bad about `BadSort()`.

5. Simplify the recurrence for `MergeSort()` by assuming that  $n$  is an exact power of 2;  $n = 2^k$  for some integer  $k \geq 0$ .

$$T(n) = \begin{cases} 0 & n = 1 \\ 2T\left(\frac{n}{2}\right) + (n-1) & n \geq 2, n = 2^k \end{cases}$$

Use the iteration method to find an exact solution to this recurrence.

6. Write a recursive algorithm (modeled on MergeSort()) that determines if an array is sorted, i.e. given an array  $A = (A_1, A_2, \dots, A_n)$  as input, return TRUE/FALSE iff  $A$  is/is-not arranged in increasing order. Prove the correctness of your algorithm. Write a recurrence for the number  $T(n)$  of array comparisons performed by your algorithm. Check that  $T(n) = n - 1$  is the exact solution to your recurrence.
7. Given  $A = (A_1, A_2, \dots, A_n)$ , a pair of indices  $(i, j)$  is called an *inversion* iff both  $i < j$  and  $A_i > A_j$ . Write a recursive algorithm that determines the number of inversions in its input array  $A$ . Do this in such a way that the worst case number of comparisons performed is  $T(n) = \Theta(n \log n)$ . (Hint: modify MergeSort() so that it counts inversions as it sorts.)