

Assignment 1 (due 09/23/2026)

*Instructor: Shikha Singh**Solution template: [Overleaf link](#)*

Note. This is a **partner assignment**. You can find your assigned partner by going on GLOW → People → Groups → Assignment 1 Pairs. You and your partner must complete a joint write up and submit the PDF on Gradescope. When submitting, make sure to add your partner's name to the submission, as well as assign pages to each problem.

Problem 1 (15 points). Take the following list of functions and arrange them in ascending order of growth rate. That is, if function $f(n)$ and $g(n)$ are such that $f(n)$ is $O(g(n))$, then $g(n)$ must appear **after** $f(n)$ in your ordering. Separate functions in your ordering with a comma. If two functions have asymptotically the same order, that is, $f(n) = \Theta(g(n))$, then indicate that explicitly and place them at that same location in your ordering. No justification is required.

For example, suppose we are sorting the functions $\log_{100} n$, n^2 , $n \log n$ and $\log_2 n$ in ascending order of growth, then the final ordering must be $\boxed{\log_{100} n = \Theta(\log_2 n), n \log n, n^2}$.

Note. All log's are base 2.

(a) $\log n$ (d) $4^{\log n}$ (b) $\log \sqrt{n}$ (e) 5^n (c) $\sqrt{\log n}$ (f) n^2

Solution.

□

Problem 2. Prove the following statement using the definition of Big $O(\cdot)$ and $\Omega(\cdot)$.

Statement 1. A function $f(n) = O(g(n))$ if and only if $g(n) = \Omega(f(n))$.

Solution.

□

Problem 3. Consider the following insertion sort algorithm. The array A contains $n \geq 1$ numbers, indexed from 0 to $n - 1$. Assume each comparison, array access, and assignment takes constant time.

```
insertionSort(A):  
  for i = 1 to |A| - 1:  
    j = i  
    while j > 0 and A[j] < A[j-1]:  
      swap(A[j-1], A[j]) # swaps A[j-1] and A[j]  
      j = j - 1
```

- (a) Prove that the algorithm correctly sorts A by showing that it maintains the following invariant. (*Hint.* Prove this by induction on i .)

Claim 1. *The items $A[0]$ through $A[i]$ are sorted in non-decreasing order after i iterations of the for loop.*

- (b) Analyze the algorithm's worst-case running time using big- O notation. Bound the number of steps during iteration i , then sum over all iterations.
- (c) Give an input array A on which the algorithm has running time $\Omega(n^2)$.

Problem 4. (Stable Matchings)

- (a) For any n , consider the following input: For all $1 \leq i \leq n$, the preferences for h_i are $s_1, s_2, \dots, s_n$. For all $1 \leq i \leq n$, the preferences for s_i are $h_1, h_2, \dots, h_n$.

Show that for any n , under the above input, the **while** loop of Gale-Shapley iterates $\Omega(n^2)$ times (in other words, the hospitals make $\Omega(n^2)$ offers). You are not required to give a formal inductive proof—a clear English explanation suffices. Assume that the list of free hospitals is implemented using a queue, exactly as we saw in class, and assume that the queue begins with all hospitals in order $h_1, \dots, h_n$ (so h_1 will be the first hospital removed from the queue).

Solution.

□

- (b) Decide whether you think the following statement is TRUE or FALSE. If the statement is true give a proof; if false give a counterexample.

Statement 2. Consider an instance of the Stable Matching problem in which there is a hospital h and a student s such that h is ranked first on the preference list of s , and s is ranked first on the preference list of h . Then h and s must be matched to each other in every stable matching for the instance.

Solution.

□