601.433/633 Introduction to Algorithms **Fall 2025**
Homework #1 **Due:** September 15, 2025, 11:59pm

Remember: you may work in groups of up to three people, but must write up your solution entirely on your own. Solutions must by typeset (LaTeX preferred but not required). Collaboration is limited to discussing the problems – you may not look at, compare, reuse, etc. any text from anyone else in the class. Please include your list of collaborators on the first page of your submission. You may use the internet to look up formulas, definitions, etc., but may not simply look up the answers online.

Please include proofs with all of your answers, unless stated otherwise.

1 Asymptotic Notation (25 points)

For each of the following statements say if it true or false and prove your answer. The base of log is 2 unless otherwise specified, and ln is $\log_e$. Assume the domain of the functions are the positive reals.

(a) $2^n = \Omega(3^n)$

Solution: False.

(b) $n \sin n = O(n)$

Solution: True. In order to prove that $n \sin n = O(n)$, we need to show that there exists constants c and n_0 such that $n \sin n \leq cn$ for all $n > n_0$.

We know that $n \sin n \leq 1 \cdot n$ for all $n > 1$ because $\sin n \leq 1$. Thus, if we choose $c = 1$ and $n_0 = 1$, these functions satisfy the definition of big O.

(c) $e^n = O(e^{(n-5)})$

Solution: True. In order to prove that $e^n = O(e^{(n-5)})$, we need to show that there exists constants c and n_0 such that $e^n \leq c \cdot e^{(n-5)}$ for all $n > n_0$. We know that $e^5 \leq 150$, and hence $e^{n-5} = e^n/e^5 \geq e^n/150$. So if we set $n_0 = 1$ and $c = 150$, we have that $e^n \leq c \cdot e^{n-5}$ for all $n > n_0$ as required.

(d) $\frac{n}{\log^2 n} = \Omega(n^{0.9})$

Solution: True. To show this, set $c = 1$. Suppose that $\frac{n}{\log^2 n} < n^{0.9}$. Then $n^{0.05} < \log n$. But this is only true for $2.05136 < n < 1.4 \times 10^{43}$ (using Wolfram Alpha). So if we set $n_0 = 1.4 \times 10^{43}$, we have that $\frac{n}{\log^2 n} \geq n^{0.9}$ for all $n > n_0$, as claimed.

(e) $(\log(n^{1/5}))^{3/2} = \Theta(\log(n^3))$

Solution: False

(f) $2^{(9/4)\log n} = O(n^2)$

Solution: False. Observe that $2^{(9/4)\log n} = (2^{\log n})^{9/4} = n^{9/4}$. Suppose $2^{(9/4)\log n} = n^{9/4} = O(n^2)$. That is, suppose there exists some c and n_0 such that, for all $n > n_0$, $n^{9/4} \leq cn^2$. But clearly this is impossible, because $n^{9/4} \geq cn^2$ when $n > c^4$. Thus we have a contradiction! $2^{(9/4)\log n}$ is not $O(n^2)$.

(g) Let f, g be positive functions. Then $f(n) + g(n) = \Omega(\max(f(n), g(n)))$

Solution: True. In order to prove that $f(n)+g(n) = \Omega(\max(f(n), g(n)))$, we need to show that there exists constants c and n_0 such that $f(n)+g(n) \geq c \cdot \max(f(n), g(n))$ for all $n > n_0$.

We know that for all $n > 1$,

$$f(n) + g(n) \geq 1 \cdot \max(f(n), g(n))$$

because $f(n), g(n)$ are positive functions. Thus, if we choose $c = 1$ and $n_0 = 1$, then these functions satisfy the definition of big Ω.

(h) Let f, g be positive functions. Then $f(n) + g(n) = O(\max(f(n), g(n)))$

True. In order to prove that $f(n)+g(n) = O(\max(f(n), g(n)))$, we need to show that there exists constant $c, n_0 > 0$ such that $f(n)+g(n) \leq c \cdot \max(f(n), g(n))$ for all $n > n_0$. We know that for all $n > 1$, both $f(n) \leq \max(f(n), g(n))$ and $g(n) \leq \max(f(n), g(n))$ and hence,

$$f(n) + g(n) \leq 2 \cdot \max(f(n), g(n))$$

because $f(n), g(n)$ are positive functions. Thus, if we choose $c = 2$ and $n_0 = 1$, then these functions satisfy the definition of big O.

2 Recurrences (25 pts)

Solve the following recurrences, giving your answer in Θ notation (so prove both an upper bound and a lower bound). For each of them you may assume $T(x) = 1$ for $x \leq 5$. Justify your answer (formal proof not necessary, but recommended).

(a) $T(n) = 6T(n/5) + n^2$

Solution: We can use the Master Theorem, which implies that $T(n) = \Theta(n^2)$.

(b) $T(n) = 10T(n - 5)$

Solution: We can find the answer using the unrolling method:

$$\begin{aligned} T(n) &= 10T(n-5) \\ &= 10^2 T(n-10) \\ &= 10^3 T(n-15) \\ &\vdots \\ &= 10^{\lfloor n/5 \rfloor} T(n - 5\lfloor \tfrac{n}{5} \rfloor) \\ &= 10^{\lfloor n/5 \rfloor} \end{aligned}$$

The last step works because $n - 5\lfloor \frac{n}{5} \rfloor \leq 5$.

Based on these calculations, we can conclude that $\boxed{T(n) = \Theta(10^{\frac{n}{5}})}$.

(c) $T(n) = 2T(n/2) + \log n$

Solution: $\Theta(n)$.

3 Basic Proofs (25 pts)

(a) Prove **by induction** that $\sum_{i=1}^{n} i^2 = \frac{1}{6}n(n+1)(2n+1)$

Solution: For the base case ($n = 1$), we have that $\sum_{i=1}^{n} i^2 = \sum_{i=1}^{1} i^2 = 1 = \frac{1}{6}(1 \cdot 2 \cdot 3) = \frac{1}{6}n(n+1)(2n+1)$ as required.

For the inductive step, suppose that $\sum_{i=1}^{n-1} i^2 = \frac{1}{6}(n-1)n(2(n-1)+1) = \frac{1}{6}(n-1)n(2n-1)$. Then

$$\begin{aligned} \sum_{i=1}^{n} i^2 &= n^2 + \sum_{i=1}^{n-1} i^2 = n^2 + \frac{1}{6}(n-1)n(2n-1) \\ &= n^2 + \frac{1}{6}(2n^3 - 3n^2 + n) \\ &= \frac{1}{6}(2n^3 + 3n^2 + n) \\ &= \frac{1}{6}n(n+1)(2n+1) \end{aligned}$$

as required.

(b) Consider a polynomial $P(x) = \sum_{k=0}^{n} a_k x^k$, and consider the following algorithm:

```
y = 0;
for i = n down to 0 do
    y = a_i + (x · y);
end
return y;
```

Prove that this algorithm correctly computes $P(x)$ when called on input x.
Hint: Think of an appropriate "loop invariant" / "induction hypothesis" for a proof by induction.

Solution: We will prove by induction that at the end of each iteration of the for loop, $y = \sum_{k=0}^{n-i} a_{k+i}x^k$ (where i is the value of i in the loop). If we can prove this then we are finished, since it implies that when $i = 0$, at the end of the loop we have that $y = \sum_{k=0}^{n} a_k x^k$, and this is the value that is finally returned.

So to prove that at the end of each iteration $y = \sum_{k=0}^{n-i} a_{k+i}x^k$, first consider the base case of $i = n$. This is the very first iteration of the for loop, so at the end of the iteration we have that $y = a_n + (x \cdot 0) = a_n x^0 = \sum_{k=0}^{n-i} a_{k+i}x^k$ as claimed.

Now we do the inductive step. Suppose that the claim is true at the end of some iteration $i + 1$, and consider the next iteration i (note that the indices are going down since the for loop counts down). Then

$$\begin{aligned} a_i + (x \cdot y) &= a_i + x \cdot \sum_{k=0}^{n-(i+1)} a_{k+(i+1)}x^k = a_i + \sum_{k=0}^{n-i-1} a_{k+1+i}x^{k+1} \\ &= a_i + \sum_{k=1}^{n-i} a_{k+i}x^k = \sum_{k=0}^{n-i} a_{k+1}x^k \end{aligned}$$

(c) I have a bucket with 23 balls, 13 of which are white and 10 of which are black. If I draw 8 balls at random from the bucket (all at one time), what is the probability that exactly five of them are white?

Solution: For this problem, the sample space consists of all the different ways of drawing 8 balls from the bucket. There are $\binom{23}{8}$ ways, and they each have equal probability.

Now consider all the outcomes that have exactly 5 white balls and 3 black balls. There are $\binom{13}{5}$ ways of selecting 5 white balls from the bucket, and $\binom{10}{3}$ ways of selecting 3 black balls. So there are $\binom{13}{5}\binom{10}{3}$ ways, in total, of selecting 8 balls such that exactly 5 of them are white.

This means that, if you draw 8 balls total, there is a $\frac{\binom{13}{5}\binom{10}{3}}{\binom{23}{8}}$ probability that exactly 5 will be white.

4 Mistakes and Insertion Sort (25 pts)

Given an array $[a_0, a_1, \ldots, a_{n-1}]$, a *mistake* is a pair (i, j) such that $i < j$ but $a_i > a_j$. For example, in the array $[5, 3, 2, 10]$ there are are three mistakes $((0, 1), (0, 2), (1, 2))$. Note that the array has no mistakes if and only if it is sorted, so the number of mistakes can be thought of as a measure of how well-sorted an array is. For this problem, assume that all elements in an array are distinct.

(a) What is the expected number of mistakes in a random array? More formally, consider a random permutation π of n distinct elements $a_0, \ldots, a_{n-1}$: what is the expected number of mistakes in the resulting array?

Solution: Let X_{ij} be an indicator random variable for the event that (i, j) is a mistake. Since the permutation is random and all elements are distinct, we know that $\Pr[X_{ij} = 1] = 1/2$

and $\Pr[X_{ij} = 0] = 1/2$ and hence $\mathrm{E}[X_{ij}] = 1/2$. By linearity of expectations, the total expected number of mistakes is

$$\mathrm{E}\left[\sum_{i=0}^{n-1}\sum_{j=i+1}^{n} X_{ij}\right] = \sum_{i=0}^{n-1}\sum_{j=i+1}^{n} \mathrm{E}[X_{ij}] = \sum_{i=0}^{n-1}\sum_{j=i+1}^{n} \frac{1}{2} = \frac{n(n-1)}{4}$$

(b) Recall the insertion sort algorithm:

```
for i = 1 to n − 1 do
    j = i;
    while j > 0 and A[j − 1] > A[j] do
        Swap A[j] and A[j − 1];
        j = j − 1;
    end
end
```

Suppose that our array has d mistakes. Prove a lower bound on the running time of insertion sort in terms of d (asymptotic notation OK).

Solution: Each iteration of the while loop decreases the number of mistakes by exactly 1. Hence the running time must be at least $\Omega(d)$.

(c) Prove an upper bound on the running time of insertion sort in terms of n and d (asymptotic notation OK).

Solution: Each iteration of the while loop decreases the number of mistakes by exactly 1, and hence the while loop is executed precisely d times. Each iteration of the while loop takes only $O(1)$ time, and hence the total time spent inside the while loop is $O(d)$. There is an additional $O(n)$ time spend outside of the while loop, and hence the total running time is $O(n + d)$.