CSE 101 Homework 0

Winter 2023

This homework is due on gradescope Friday January 13th at 11:59pm on gradescope. Remember to justify your work even if the problem does not explicitly say so. Writing your solutions in LaTeXis recommend though not required.

Question 1 (Program Runtimes, 20 points)**.** *Consider the following two programs:*

```
Alg1(n)
  for i = 1 to n^3
    for j = 1 to n
      Print(j)
```

and

```
Alg2(n)
  for i = 1 to n^3
    if i <= n
      for j = 1 to n
        Print(j)
```

For each of these programs give the asymptotic runtime as $\Theta(f(n))$ for some function f and justify your work.

Runtime for Alg1: The outside loop runs for n^3 times and the inner loop runs for n times. Multiplying them together, we get the total runtime is $\Theta(n^4)$.

Runtime for Alg2: In the first n iterations of the outside loop, the inner loop will be executed, contributing $\Theta(n \cdot n) = \Theta(n^2)$ to the runtime. In the remaining $n^3 - n$ iterations of the outside loop, only the if-statement is executed (the inner loop will be skipped), contributing $\Theta((n^3 - n) \cdot 1) = \Theta(n^3 - n)$ to the runtime. The total runtime is therefore $\Theta(n^2 + (n^3 - n)) = \Theta(n^3 + n^2 - n) = \Theta(n^3)$.

Question 2 (Asymptotic Comparisons, 20 points)**.** *Sort the following functions of n in terms of their asymptotic growth rates. In particular, ones should go later in the list if they are larger when sufficiently large values of n are used as inputs. Which of these functions have polynomial growth rates? Remember to justify your answers.*

- $a(n) = 2^{\sqrt{\log(n)}}$
- $b(n) = 2^{\sqrt{n}}$
- $c(n) = 10^{10^{10}} n^{0.01}$
- $d(n) = 6^{\log_2(n)}$

- $e(n) = n(1000 + \sqrt{n})(1000 + n)$

We will assume log is base 2 (whether it is base 10, e, or 2 will not affect the final answer). Observe for any $x = \Theta(n^k)$ with constant $k > 0$ (i.e. x is a polynomial), we have $\log(x) = \Theta(\log(n^k)) = \Theta(k \log(n)) = \Theta(\log(n))$, regardless of k's size. Since $\log(a(n)) = \sqrt{\log(n)}$, $a(n)$ must grow slower than any polynomial. Similarly, $b(n)$ must grow faster than any polynomial, since $\log(b(n)) = \sqrt{n}$.

It then remains to show the order of $c(n), d(n), e(n)$.

$$\begin{aligned} c(n) &= \Theta(n^{0.01}) \\ d(n) &= 2^{\log_2(6^{\log_2 n})} = 2^{\log_2 6 \cdot \log_2 n} = n^{\log_2 6} \approx n^{2.58} \\ e(n) &= \Theta(n) \cdot \Theta(\sqrt{n}) \cdot \Theta(n) = \Theta(n^{2.5}), \end{aligned}$$

Now, we use the fact that n^a grows slower than n^b asymptotically if $a < b$.

Therefore, the order should be $a(n) \ll c(n) \ll e(n) \ll d(n) \ll b(n)$. Among them, $b(n)$ grows faster than any polynomial function and $a(n)$ grows slower than any polynomial function. The rest all have polynomial growth rates.

Note that asymptotic comparisons ignore outer constants. While $c(n)$ is asymptotically smaller than almost all other given functions, it is actually the largest value until n gets quite large, on account of the leading $10^{10^{10}}$ constant.

Question 3 (Graph Coloring, 30 points)**.** *Let G be a finite graph with maximum degree at most d (that is no vertex is connected to more than d other vertices). Show that each vertex of G can be assigned an integer in $\{1, 2, \ldots, d+1\}$ so that no two adjacent vertices are assigned the same integer. Hint: Use induction on the number of vertices.*

Suppose the vertices are labeled from $1, \ldots, n$. We will denote by G_i as the subgraph formed by vertices from $1, \ldots, i$. We prove by induction on i that there is a valid coloring of G_i.

Base Case: When $i = 1$, it is trivial to color G_i, since the subgraph is simply a single vertex that can be assigned any color. (We could alternatively use $i = 0$ (the graph on zero vertices) for the base case if we want to ensure our statement is proven for that graph.)

Inductive Step: Assume that we have a conflict-free assignment for G_{i-1}. Then, we claim we can find a conflict-free assignment for G_i. We first give numbers to vertices $1, \ldots, i-1$ according to the assignment of G_{i-1}. For the i-th vertex, since it has at most d neighbors, we can always find some number that has not been assigned to its neighbors yet. Thus, we arrive at a conflict-free assignment for G_i.

By induction, in the end, we have a conflict-free assignment for G_n, which is exactly G.

Question 4 (Recurrence Relation, 30 points)**.** *Suppose that you have a function $T(n)$ defined by $T(1) = 1$ and*

$$T(n) = T(n-1) + n$$

for $n > 1$.

(a) *Prove by induction that $T(n) = n(n+1)/2$. [15 points]*

(b) *Consider the following "proof" that $T(n) = O(n)$ (note that this contradicts part (a)):*

We proceed by strong induction on n. Clearly $T(1) = O(1)$, which gives us our base case. If we assume that $T(n) = O(n)$, then $T(n+1) = T(n) + (n+1) = O(n) + O(n) = O(n)$. This completes our inductive step and proves that $T(n) = O(n)$ for all n.

What is wrong with the above proof? (Hint: Consider what the implied constant in the O term would be.) [15 points]

Part(a). We prove $T(n) = n(n+1)/2$ by induction on n.

Base Case: When $n = 2$, we have $T(2) = T(1) + 2 = 1 + 2 = 3$. Since $2(3)/2 = 3$, the base case holds.

Inductive Step: Assume that

$$T(n-1) = \frac{(n-1) \cdot n}{2}.$$

And we try to prove that

$$T(n) = \frac{n \cdot (n+1)}{2}.$$

This is true since

$$T(n) = T(n-1) + n = \frac{(n-1) \cdot n}{2} + \frac{2n}{2} = \frac{n \cdot (n+1)}{2}.$$

By induction, we complete the proof.

Part(b). The statement $T(n) = O(n)$ for a particular value of n is meaningless. If we instead try to prove the induction by using the definition of big-O—that $T(n) \leq C \cdot n$ for all n and an absolute constant C—we run into issues in the inductive step. There, we can only say $T(n+1) = T(n) + (n+1) \leq (C+1)n + 1$. And one can see that the "constant" in the bound becomes larger than C. This makes it no longer an absolute constant independent of n.