Asymptotics

1. **(Fun with asymptotics)** In this exercise we used the definitions of Big-O, Big-Ω, and Big-Θ to prove the following statements.

   (a) We prove that $\log\log(n) = O(\log^2(n))$. Consider the values $c = 1$ and $n_0 = 2$. Let $x(n) = \log(n)$, which by definition of log is a positive and increasing function of n. For all $x \geq 1$, we know that $\log(x) \leq x \leq 1 \cdot x^2 = c \cdot x^2$. Therefore, for all $n \geq 2$, we know that $\log(\log(n)) \leq c \cdot \log^2(n)$.

   (b) We prove that $n^2 = \Omega(6n\sqrt{n} + 4n)$. Consider the values $c = 1/20$ and $n_0 = 2$. For all $n \geq n_0 = 2$:

   $$\begin{aligned} n^2 &\geq (1/2)n^2 = (1/20)(10n^2) \\ &\geq (1/20)(10n\sqrt{n}) = (1/20)(6n\sqrt{n} + 4n\sqrt{n}) \\ &\geq (1/20)(6n\sqrt{n} + 4n) = c\dot{(}6n\sqrt{n} + 4n) \end{aligned}$$

   Thus, there exists $c = 1/20$ and $n_0 = 2$ for which $n^2 \geq c\dot{(}6n\sqrt{n} + 4n)$ for all $n \geq n_0$.
   A general approach you can take to prove these types of bounds: simplify the given expression into the product of c and a single term that's a function of n, using the given expression as a lower bound if you're trying to prove Big-Ω or an upper bound if you're trying to prove Big-O. In this example, we notice that $6n\sqrt{n} + 4n \leq 6n\sqrt{n} + 4n\sqrt{n}$ and the latter expression can be combined into a single term $10n\sqrt{n}$. Continue to simplify the expression to something trivial like $n^2 \geq (1/2)n^2$, then write your proof performing the algebra in the reverse order, as above.

   (c) We prove that $n^2 + 5n\sqrt{n}$ is **not** $\Theta(n^3)$. We hypothesize this might be the case because $n^2 + 5n\sqrt{n}$ is **not** $\Omega(n^3)$. Proving the latter statement proves the original statement by definition of Big-Θ. To prove $n^2 + 5n\sqrt{n}$ is **not** $\Omega(n^3)$, we proceed by contradiction. Suppose there exists a c and n_0 such that for all $n \geq n_0$, $n^2 + 5n\sqrt{n} \geq c \cdot n^3$. Now, consider $n = \max\{6/c, n_0\} + 1$. By construction, we have $n \geq n_0$ and $n > 6/c$. The latter expression implies that $cn^3 > 6n^2 = n^2 + 5n^2 > n^2 + 5n\sqrt{n}$, which contradicts our earlier assumption.

   (d) We prove that $\log(n!) = \Theta(n\log(n))$.
   First, we prove that $\log(n!) = O(n\log(n))$. Notice that

   $$\log(n!) = \log\left(\prod_{i=1}^{n} i\right) = \sum_{i=1}^{n} \log(i) \leq \sum_{i=1}^{n} \log(n) = n\log(n)$$

   Thus, for $c = 1$ and $n_0 = 1$, we have $\log(n!) \leq n\log(n)$ for all $n \geq n_0 = 1$.
   Next, we prove that $\log(n!) = \Omega(n\log(n))$. Notice that

   $$\log(n!) = \sum_{i=1}^{n} \log(i) \geq \sum_{i=n/2}^{n} \log(i) \geq \sum_{i=n/2}^{n} \log(n/2) = (n/2)\log(n/2)$$

   Consider the values $c = 1/4$ and $n_0 = 4$. Since $(n/2)\log(n/2) = (n/2)\log(n) - (n/2) \geq (1/4) \cdot n\log(n) = c \cdot n\log(n)$, we have $\log(n!) \geq (1/4)n\log(n)$ for all $n \geq n_0 = 4$.
   Since $\log(n!) = O(n\log(n))$ and $\log(n!) = \Omega(n\log(n))$, then $\log(n!) = \Theta(n\log(n))$.
   The result that $\log(n!) = \Theta(n\log(n))$ might be helpful to prove Homework 1, Exercise 3(a).

2. **(A hairy proof)** In the step that applies log to both sides, it must be applied to the c as well; otherwise, we're no longer comparing the desired functions!

   Consider the proof written with this operation applied correctly. First, we apply log to both sides of $n^3 \leq c \cdot n^2$, which gives: $3\log(n) \leq \log(c) + 2\log(n)$. Again, consider the values $c = 2$ and $n_0 = 1$. For the chosen value of c, notice $3\log(n) \geq \log(2) + 2\log(n)$ for all $n \geq 2$.

   An exercise left to the reader: prove n^3 is **not** $O(n^2)$.

3. **(n-naught not needed?)** Let

$$c' = \max\left(c, \frac{T(1)}{1^d}, \frac{T(2)}{2^d}, \ldots, \frac{T(n_0-1)}{(n_0-1)^d}\right)$$

   Then

$$T(n) \leq \begin{cases} T(n) & n < n_0 \\ cn^d & n \geq n_0 \end{cases} \leq \begin{cases} c'n^d & n < n_0 \\ c'n^d & n \geq n_0 \end{cases} \leq c'n^d$$

   Notice that the first inequality follows from the definition of Big-O and that $T(n) = O(n^d)$ (given in the problem statement). The second inequality follows by construction of c'. The third inequality summarizes what we wanted to prove.

   This problem is challenging. It helps to consider the premise of the problem statement: you've been taught that by definition of Big-O, $T(n) = O(n^d)$ iff there exists *both* c *and* n_0 such that $T(n) \leq c \cdot n^d$ for all $n \geq n_0$. To prove that n_0 isn't actually necessary, given arbitrary $T(n)$ and n^d where $T(n) = O(n^d)$, you need to prove there exists some c' such that $T(n) \leq c'n^d$ for all $n \geq n_0 = 1$. Thus, the solution constructs this c' explicitly from values c and n_0, which must exist due to the original definition of Big-O.

4. Here's the table.

	`linear_search`	`binary_search`
Lower-bound of best-case	$\Omega(1)$	$\Omega(1)$
Upper-bound of best-case	$O(1)$	$O(1)$
Lower-bound of worst-case	$\Omega(n)$	$\Omega(\log(n))$
Upper-bound of worst-case	$O(n)$	$O(\log(n))$

   The main takeaway here: understand the difference between worst-case/best-case and lower-bound/upper-bound. The worst-case/best-case describe a *specific* runtime (e.g. $6n\log(n)$ or $3.25n^2$) which can then be lower, upper, or tightly bounded.

Recurrences

---

1. **(Fun with recurrences)** In this exercise we solved the following recurrences.

   (a) $T(n) = O(n^{\log_4 3})$, using the Master Theorem with $a = 3$, $b = 4$, $d = 1/2$. We have $a > b^d$, so the runtime is $O(n^{\log_b(a)})$.

   (b) $T(n) = O(n^3)$, using the Master Theorem with $a = 7$, $b = 2$, $d = 3$. (Notice that the $\Theta(n^3)$ expression is $O(n^3)$ as well, as per the statement in class.) Then $a < b^d$, so the runtime is $O(n^d) = O(n^3)$.

   (c) To get intuition, we begin by iteratively plugging in the recurrence relation:

$$\begin{aligned} T(n) &= 2T(\sqrt{n}) + 1 \\ &= 2(2T(n^{1/4}) + 1) + 1 \\ &= 4T(n^{1/4}) + 2 + 1 \\ &= 4(2T(n^{1/8}) + 1) + 2 + 1 \\ &= 8T(n^{1/8}) + 4 + 2 + 1 \end{aligned}$$

   Carrying on this way, we see that for $t \geq \log\log(n)$,

$$T(n) = 2^t T(n^{1/2^t}) + \sum_{i=0}^{t-1} 2^i.$$

   (To formally prove this, we may do a proof by induction; this is not required for credit for problems like this one if asked on a homework assignment or exam.) Now, for $t = \log\log(n)$, we have $n^{1/2^t} = 2$, and so plugging in $t = \log\log(n)$ we see

$$T(n) = 2^{\log\log(n)} T(2) + \sum_{i=0}^{\log\log(n)-1} 2^{\log\log(n)} = O(\log(n)).$$

   An exercise left to the reader: Prove this bound with substitution method.

Problems

---

1. **(Selection sort)**

   (a) At the beginning of iteration i (the iteration where we try to select the element to occupy `A[i]`), `A[:i]` contains the i smallest elements from the original list A, in sorted order.

   (b) At the beginning of iteration j, `min_idx` contains the index of the minimum element in the sublist `A[i:j]`.

   (c)
   i. The loop invariant holds at the beginning of iteration i of the outer loop i.e. `A[:i]` contains the i smallest elements from the original list A, in sorted order.
   ii. The loop invariant holds before the algorithm starts when `i=0` i.e. `A[:0]` contains 0 elements, which are trivially the smallest elements in sorted order.
   iii. Suppose the loop invariant holds at the beginning of iteration i. We prove it holds at the beginning of iteration $i+1$. If the inner loop invariant holds at the end of the inner loop, then `min_idx` contains the index of the smallest element in `A[i:n]`. Calling `swap` swaps this smallest element into position `A[i]`. According to the inductive hypothesis, `A[0]` $\leq$ `A[1]` $\leq \ldots \leq$ `A[i-1]`; also, `A[i]` is at least as large as `A[i-1]` since `A[:i]` contained the i smallest elements from the original list A. Therefore, after the swap, `A[:i+1]` contains the $i+1$ smallest elements from the original list A, in sorted order, completing the induction.
   iv. At the beginning of iteration $n-1$ of the outer loop, `A[:n-1]` contains the $n-1$ smallest elements from the original list A, in sorted order. If `A[:n-1]` contains the $n-1$ smallest elements from the original list A, then `A[n-1]` must be at least as large as all elements in `A[:n-1]`; therefore, `A[:n]` is sorted. Since `A[:n]` is the whole list A, selection sort is correct.

   (d)
   i. The loop invariant holds at the beginning of iteration j of the inner loop i.e. `min_idx` contains the index of the minimum element in the sublist `A[i:j]`
   ii. The loop invariant holds before the loop starts when `j=i+1` i.e. `min_idx` is set to `i` and `A[i:i+1]` contains only one element, element `i`.
   iii. Suppose the loop invariant holds at the beginning of iteration j. We prove it holds at the beginning of iteration $j+1$. According to the inductive hypothesis, `min_idx` contains the index of the minimum element in the sublist `A[i:j]`. If `A[j]` $<$ `A[min_idx]`, then `A[j]` is the minimum element in `A[i:j+1]`, so setting `min_idx = j` is correct. Otherwise, `A[min_idx]` is the minimum element in `A[i:j+1]`, so leaving it alone is correct. This completes the induction.
   iv. At the beginning of iteration n of the inner loop, `min_idx` contains the index of the minimum element in the sublist `A[i:n]`. This is the condition required by the inductive step of outer loop.

A few general tips for finding the loop invariant:

- First, determine the "protagonist" of the loop invariant by asking the following question: What data changes as a function of i? In insertion sort, the so-called "protagonist" of the loop invariant is the sublist `A[:i]`. Same goes for selection sort.
- Then consider: What conditions are true about the "protagonist" for each iteration? These conditions form the initial candidate for your inductive hypothesis.
- Next be skeptical: Can you think of a counterexample where a partial solution satisfies this hypothesis but doesn't produce a correct answer? For example, if the initial candidate for your inductive hypothesis for the outer loop of selection sort were: "`A[:i]` is sorted," you could think of the following partial solution that satisfies this hypothesis but will produce an incorrect answer: `[2, 3, 5, 1, 4]` at the start of iteration $i = 3$.

- Refine your inductive hypothesis to handle this counterexample.

Once you decide on the loop invariant and inductive hypothesis, the rest of the proof gets reduced to an exercise in plug-and-chug, so you should mostly practice identifying the loop invariant and inductive hypothesis!

2. **(Why not Select with groups of 3 or 7?)**

   (a) Let $g = \lceil n/3 \rceil$ represent the number of groups.

$$\begin{aligned} |A| &\leq n - 1 - (2 \cdot (\lceil g/2 \rceil - 2) + 1) \\ &= n + 2 - 2 \cdot \lceil g/2 \rceil \\ &\leq n + 2 - 2g/2 \\ &= n + 2 - \lceil n/3 \rceil \\ &\leq n + 2 - n/3 \\ &= 2n/3 + 2 \end{aligned}$$

   Note that the -2 comes from discarding the group with the median of medians and the leftover group and the $+1$ comes from including the one value in the same group as the median of medians.

   (b) $T(n) \leq T(n/3) + T(2n/3 + 2) + \Theta(n)$

   (c) No, it is not $O(n)$.

   We will imagine drawing a tree. At the top level of our tree, we have only one problem of size n, and we do $\Theta(n)$ work. Then, at the next level, we do $\Theta(n/3) + \Theta(2n/3) = \Theta(n)$ work. In fact, if we look at the two children of a particular node, we notice that the amount of work done within each of those two children is exactly equal to the amount of work done in the parent! This is true all the way down the tree, so we can see that there is $\Theta(n)$ work done at every level. Moreover, since we either multiply the problem by $1/3$ or by $2/3$ each time, we can see that there are $\Theta(\log n)$ levels. (Depending what route you take through the tree, it could range from $\log_3(n) + 1$ at shortest to $\log_{3/2}(n) + 1$ at longest, but the difference between $\log_3(x)$ and $\log_{3/2}(x)$ is just a multiplicative constant, which we ignore in the Big-Θ notation. (See Homework 1, Exercise 3b for further convincing.) Then overall we can see that the runtime is $\Theta(n \log n)$, so it is not $O(n)$.

   (d) Let $g = \lceil n/7 \rceil$ represent the number of groups.

$$\begin{aligned} |A| &\leq n - 1 - (4 \cdot (\lceil g/2 \rceil - 2) + 3) \\ &= n + 4 - 4 \cdot \lceil g/2 \rceil \\ &\leq n + 4 - 4g/2 \\ &= n + 4 - 2\lceil n/7 \rceil \\ &\leq n + 4 - 2n/7 \\ &= 5n/7 + 4 \end{aligned}$$

   Note that the -2 comes from discarding the group with the median of medians and the leftover group and the $+3$ comes from including the three values in the same group as the median of medians.

   (e) $T(n) \leq T(n/7) + T(5n/7 + 4) + \Theta(n)$

   (f) Yes, it is $O(n)$. We proceed by substitution method.

   - **Inductive hypothesis** $T(k) = O(k)$ for all $k \in \{1, 2, \ldots, n-1\}$ i.e for $1 \leq k < n$, we have that $T(k) \leq \max\{7, 7d\} \cdot k$.
   - **Base case** $T(k) \leq \max\{7, 7d\} \cdot k$ for all $k \leq 100$.

- **Inductive step** Since $T(n) \leq T(n/7) + T(5n/7+4) + \Theta(n)$, there exists some d and n_0 such that $T(n) \leq T(n/7) + T(5n/7+4) + dn$.

$$\begin{aligned} T(n) &\leq T(n/7) + T(5n/7+4) + dn \\ &\leq \max\{7, 7d\} \cdot n/7 + \max\{7, 7d\} \cdot (5n/7+4) + dn \\ &= \max\{7, 7d\} \cdot (6n/7+4) + dn \\ &\leq \max\{7, 7d\} \cdot n \end{aligned}$$

- **Conclusion** For all $n > 1$, $T(n) \leq \max\{7, 7d\} \cdot n$. Therefore, $T(n) = O(n)$.