Problem 1

In these solutions, I am assuming that we are sorting an array of n distinct elements. Note that the problem should have specified this fact, as an array with repeating elements has a different probability of being sorted.

Let A_i be the event that bogosort finds the correct solution on the 4th shuffle. We are interested in $P(A_4) = P(A_1^c \wedge A_2^c \wedge A_3^c \wedge A_4) = (1 - P(A_1))(1 - P(A_2))(1 - P(A_3))P(A_4)$. For any i, $P(A_i) = 1/n!$ because there are $n!$ possible arrangements of the n elements in an array and only one of these arrangements is the sorted one. Thus, $P(A_4) = \boxed{(1 - 1/n!)^3 1/n!}$

Let $E(T(n))$ be the expected[1] number of comparisons necessary to sort an array of n elements. $E(T(n)) = E(2T(n/2) + n * n!) = 2E(T(n/2)) + n * n!$ for $n > 1$ and 0 otherwise. The closed form solution to this recurrence is

$$\boxed{E(T(n)) = \sum_0^{\log n} n(n/2^i)!}$$

. Let's prove its correctness.

Proof. I will prove that $E(T(n)) = \sum_0^{\log n} n(n/2^i)!$ by induction.

(a) Base case: $n = 1$. $E(T(1)) = 1 * 1!$ by plugging 1 into the recurrence. $\sum_0^{\log 1} (1)(1/2^i)! = 1$ so the statement holds for $n = 1$.

(b) Inductive hypothesis: Assume that $E(T(k)) = \sum_0^{\log k} k(k/2^i)!$ for all $k < n$.

(c) Inductive step: Show that the statement holds for $k = n$. $T(n) = 2T(n/2) + n * n!$ by the given recurrence relation. $T(n/2) = \sum_0^{\log(n/2)} n/2(n/2^{i+1})!$ by the inductive hypothesis. Plugging this in, we see that $T(n) = n * n! + \sum_0^{\log(n/2)} n(n/2^{i+1})! = n * n! + \sum_1^{\log n} n * (n/2^i)! = \sum_0^{\log n} n(n/2^i)!$ and the statement is true by the principle of mathematical induction.

□

You may be wondering why the hint is correct. Let T be a random variable that represents number of comparisions[2] necessary for bogosort to sort an array of n elements. Let I be a random variable that represents the number of iterations necessary to finish sorting the array. Each iteration of bogosort takes n comparisons (shuffling the array and checking if it is sorted), so $T = nI$. We are interested in $E(T) = E(nI)$. By linearity of expectation, $E(nI) = nE(I)$ (since n is a constant, not a random variable).

$E(I) = 1 * 1/n! + (1 + E(I))(1 - 1/n!)$. If the first iteration succeeds, then $E(I) = 1$, and this event occurs with probability $1/n!$. If the first iteraton fails, then we need $1 + E(I)$ more

[1] Note that I say expected because bogosort is a randomized algorithm and may require a different number of iterations each time it is called. We will explore randomized algorithms towards the end of this course.

[2] The shuffle step in bogosort actually does more than just comparing integers. This quantity really represents the number of "basic steps" that bogosort takes, where a "basic step" is a constant-time operation.

iterations because we are back at square one (bogosort's future behavior is independent of its past behavior[3]). This occurs with probability $1 - 1/n!$. From here, algebra will show that $E(I) = n!$, so $E(T) = nE(I) = n * n!$. [4]

[3] If you've taken Stat 110, this is known as the memoryless property of the geometric distribution.

[4] If you want an alternate proof of this fact, recognize that I is a geometric random variable. The proof for expectation of a geometric random variable can be found in Chapter 4 of this textbook.

Problem 2

Proof. I will show that the closed form solution of the given recurrence is $O(n \log n)$ by induction.

(a) Base case: $n = 1$. $T(n) = c * n$ by the recurrence relation for mergesort. $c = 1 \log 1 + 1 = O(n \log n)$ so the statement is true for $n = 1$.

(b) Inductive hypothesis: Assume that $T(k) = O(k \log k)$ for all $k < n$.

(c) Inductive step: I will show that $T(n) = O(n \log n)$. Using the recurrence for mergesort, $T(n) = 2T(n/2) + cn$. The inductive hypothesis gives us that $T(n/2) = O(n/2 \log(n/2))$. By the definition of big-O, $\exists N, a$ such that $\forall n > N$, $T(n/2) \leq a * n/2 \log(n/2)$.

Plugging this in, we get that $T(n) \leq an \log(n/2) + cn = an(\log n - 1) + cn = an \log n - an + cn$. Since this inequality holds for all $n > N$, $T(n) = O(n \log n)$ by the definition of big-O. Thus, the statement is true by the principle of mathematical induction.

$\square$

The general recurrence relation describing the number of comparisons even when n is not a power of two is

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n - 1$$

The closed form solution to this recurrence is

$$T(n) = n\lceil \log n \rceil - 2^{\lceil \log n \rceil} + 1$$

Notice that this is still $O(n \log n)$. To prove the correctness of this recurrence, I recommend a proof by cases and induction. This was not required for this section.

Proof. I will do a proof by induction. The formula holds for base case n=1. Plugging 1 into the recurrence yields $T(1) = T(1) + T(0) + 1 - 1$. Plugging 1 into the formula yields $0 = T(0) + 1 - 1$, so the statement is true for $n = 1$.

Assume as the inductive hypothesis that the formula holds true for everything less than n. Now, we want to prove its correctness for n. This requires showing that:

$$T(n) = n\lceil \log n \rceil - 2^{\lceil \log n \rceil} + 1 \tag{1}$$

(a) Case 1: n is even. If n is even, $\frac{n}{2}$ is an integer so $\lceil \frac{n}{2} \rceil = \lfloor \frac{n}{2} \rfloor = \frac{n}{2}$. The recurrence gives us that $T(n) = 2T(\frac{n}{2}) + n - 1$. By the inductive hypothesis, $T(\frac{n}{2}) = 2(\frac{n}{2} \log\lceil \frac{n}{2} \rceil - 2^{\log\lceil \frac{n}{2} \rceil} + 1) + n - 1$. Thus, $T(n) = 2(\frac{n}{2} \log \frac{n}{2} - 2^{\log \frac{n}{2}} + 1) + n - 1 = n\lceil \log n \rceil - 2^{\lceil \log n \rceil} + 1$. Therefore, the formula holds if n is even.

(b) Case 2: n-1 is even, n is odd, and $(n-1)/2$ is a power of 2. In this case, $\lceil \frac{n}{2} \rceil = \frac{n-1}{2} + 1$ and $\lfloor \frac{n}{2} \rfloor = \frac{n-1}{2}$. Therefore, using the recurrence, $T(n) = T(\frac{n+1}{2}) + T(\frac{n-1}{2}) + n - 1$. Now, I can plug in the inductive hypothesis to get:

$$T(n) = (\frac{n+1}{2})\lceil \log(\frac{n+1}{2})\rceil - 2^{\lceil \log(\frac{n+1}{2})\rceil} + \frac{n-1}{2}\lceil \log\frac{n-1}{2}\rceil - 2^{\lceil \log\frac{n-1}{2}\rceil} + 2 \qquad (2)$$

Because $(n-1)/2$ is a power of two, this can be simplified to $T(n) = \frac{n+1}{2}(\log(\frac{n-1}{2}) + 1) - 2^{\log(\frac{n-1}{2})+1} + \frac{n-1}{2}\log\frac{n-1}{2} - 2^{\log\frac{n-1}{2}} + 2$. With some algebra and application of properties of logarithms, this becomes $T(n) = n\lceil \log n\rceil - 2^{\lceil \log n\rceil} + 1$, so the statement holds in this case.

(c) Case 3: n-1 is even, n is odd, and $(n-1)/2$ is **not** a power of 2. Equation 2 still holds, so we can start from there. Because $(n-1)/2$ is not a power of 2, $\lceil \log(\frac{n+1}{2})\rceil = \lceil \log\frac{n-1}{2}\rceil = \lceil \log\frac{n}{2}\rceil = \lceil \log(n)\rceil - 1$ by properties of the log and ceiling functions. Substituting those into equation 2, we get that $T(n) = \frac{n+1}{2}\lceil \log n\rceil - 2^{\lceil \log n\rceil} + \frac{n-1}{2}\lceil \log n\rceil - 2^{\lceil \log n\rceil} + 2$.

$$T(n+1) = (\frac{n}{2}+1)(\lceil \log_2(n+1)\rceil - 1) - 2^{(\lceil \log_2(n+1)\rceil - 1)} + \\ (\frac{n}{2})(\lceil \log_2(n+1)\rceil - 1) - 2^{(\lceil \log_2(n+1)\rceil - 1)} + 2 + n$$

$$T(n+1) = (n+1)\lceil \log_2(n+1)\rceil - 2^{\lceil \log_2(n+1)\rceil} + 1.$$

Therefore, the formula holds when n+1 is a power of two as well. By the principle of mathematical induction, we have proved:

$$T(n) = n\lceil \log_2 n\rceil - 2^{\lceil \log_2 n\rceil} + 1$$

□

Problem 3

(a) Consider $f_1(n) = n$.

Proof. By definition, if $f_1(2n) = O(f_1(n))$, then $\exists$ c and $\exists$ N such that $\forall n > N$, $f_1(2n) < cf_1(n)$. Let c = 5 and N = 1. $\forall n > N$, $f_1(2n) = 2n < 5n = 5f_1(n) = cf_1(n)$. Therefore, $f_1(2n) < cf_1(n)$ so $f_1(2n) = O(f_1(n))$ □

(b) Consider $f_2(n) = n^n$.

Proof. $f_2(2n)$ is not $O(f_2(n))$ iff $\forall$ c, $\exists$ N such that $\forall n > N$, $f_2(2n) >= f_2(n)$, which implies that $\forall$ c, $\lim_{x\to\infty} \frac{f_2(2n)}{cf_2(n)} > 0$.

$$\lim_{n\to\infty} \frac{f_2(2n)}{f_2(n)} = \lim_{n\to\infty} \frac{(2n)^{2n}}{cn^n}$$

$$= \frac{1}{c} \lim_{n\to\infty} 2^{2n} n^n = \infty$$

Therefore, $\lim_{x\to\infty} \frac{f_2(2n)}{f_2(n)} \neq 0$, so $f_2(2n)$ is not $O(f_2(n))$ □

(c) *Proof.* Apply the definition of big-O. If $f(n)$ is $O(g(n))$, then $\exists c_1$ and $\exists N_1$ such that $\forall n > N_1$, $f(n) \leq c_1 g(n)$ If $g(n)$ is O($h(n)$), then $\exists c_2$ and $\exists N_2$ such that $\forall n > N_2$, $g(n) \leq c_2 h(n)$.

Let N = maximumN_1, N_2. Then, $\forall n > N$, $f(n) \leq c_1 g(n)$ so by substitution, $f(n) \leq c_1 c_2 h(n)$.

Therefore, $\exists c = c_1 c_2$ such that $\forall n > N$, $f(n) \leq c(h(n))$, so $f(n)$ is $O(h(n))$ by the definition of O(n). □

(d) Here is a counterexample. Consider

$$f(n) \begin{cases} n & \text{n is even} \\ 1 & \text{n is odd} \end{cases}$$

$$g(n) \begin{cases} 1 & \text{n is even} \\ n & \text{n is odd} \end{cases}$$

$f(n) \neq O(g(n))$. For any constants c and N, $\exists n > N$ such that n is even and $n > c$. Therefore, $f(n) = n > c = cg(n)$, so $f(n) \neq O(g(n))$.

$g(n) \neq O(f(n))$. For any constants c and N, $\exists n > N$ such that n is odd and $n > c$. Therefore, $g(n) = n > c = cf(n)$, so $g(n) \neq O(f(n))$.