(1)

a. We know that any node v will have d adjacent nodes. We also know that since the probability for any node to be included in U is p, the probability for any node to *not* be included in U is $(1-p)$. Now, for a node v to be not covered, neither v nor its d neighbors can be in U. The probability of this happening is $(1-p)^{d+1}$.

b. Let $X_i = 1$ if the i^{th} node is uncovered. From part a, we know that $\mathbb{E}(X_i) = (1-p)^{d+1}$. The expected number of uncovered nodes can be obtained by summing over this expected value for each node. Plugging in the value of p gives:

$$\begin{aligned}
\mathbb{E}(\text{Num. uncovered nodes}) &= n(1-p)^{d+1} \\
&\approx n\exp\left[-p(d+1)\right] \\
&= n\exp\left[-\left(\frac{\ln 2n}{d+1}\right)(d+1)\right] \\
&= n\exp\left[-\ln 2n\right] \\
&= \frac{n}{2n} = \frac{1}{2}
\end{aligned}$$

c. Each node has probability p of being included in U. So, the expected number of nodes in U should be np. Plugging in the value of P gives:

$$\mathbb{E}(\text{Num. nodes in } U) = np = \frac{n\ln 2n}{d+1}$$

e. We can call the algorithm k times; out of the k calls, we return the output with the smallest dominating set. Failure, in this context, means that out of the k calls, not once does it output the true smallest dominant set. Since the probability of failure for each individual call is $1/2$, the probability of failure after k calls is $(1/2)^k$.

To limit the probability of failure, we solve for k such that probability of failure after k calls is less than some small fraction, δ.

$$\begin{aligned}
\frac{1}{2^k} &\leq \delta \\
2^{-k} &\leq \delta \\
k &\geq -\log_2 \delta \\
k &\geq \log_2\left(\frac{1}{\delta}\right)
\end{aligned}$$

(3)

a. The algorithm is as follows:

Run $\mathcal{A}(x)$ once.
If it outputs "not prime", return "not prime".
If it outputs "prime", run $\mathcal{B}(x)$, and return its output.

We return immediately if $\mathcal{A}(x)$ outputs "not prime" because the only time it does so is if the true input was not prime. Thus, it must be the correct answer, if it outputs "not prime." On the other hand, it may output "prime", on both "prime" and "not prime" inputs. Thus, we turn to the deterministic algorithm for the correct answer.

b. If the input is prime, the running time is deterministic: the algorithm will run $\mathcal{A}(x)$ once; the answer will always be "prime", forcing us to run $\mathcal{B}(x)$. The total running time is $T(n)+100T(n) = 101T(n)$.

c. If the input is not prime, the running time will vary. We can think of the expected total running time as:

$$\mathbb{E}(T) = \frac{1}{2}T(n) + \frac{1}{2}101T(n) = 51\ T(n)$$

The first part of this equation allots for the time spent if $\mathcal{A}(x)$ outputs "prime" multiplied by the probability that $\mathcal{A}(x)$ outputs "prime". The second part allots for the time spent if $\mathcal{A}(x)$ outputs "not prime", multiplied by the probability that $\mathcal{A}(x)$ outputs "not prime".

(5)

a. We want to know the chance that the algorithm will mistakenly return a number that is not prime (due to the fact that $\mathcal{A}(x)$ is not perfect at returning prime numbers). We can express this as $Pr(x \text{ is not prime} \mid \mathcal{A}(x) \text{ outputs "prime"})$. For shorthand, let: $P =$ [x is not prime], $R =$ [x is prime], and $Q = [\mathcal{A}(x)$ outputs "prime"]. Bayes rule can be applied as follows:

$$Pr(P|Q) = \frac{Pr(Q|P)Pr(P)}{Pr(Q)} = \frac{Pr(Q|P)Pr(P)}{Pr(Q \cap P) + Pr(Q \cap R)}$$

Plugging in:

$$\begin{aligned} Pr(Q|P) &= \frac{1}{10^6 n} \\ Pr(P) &= \frac{n-1}{n} \\ Pr(Q \cap P) &= Pr(Q|P)Pr(P) = \left(\frac{1}{10^6 n}\right)\left(\frac{n-1}{n}\right) \\ Pr(Q \cap R) &= Pr(Q|R)Pr(R) = \frac{1}{n} \end{aligned}$$

we get roughly an upperbound of $\frac{1}{10^6 n}$

(6)

a. Note that since we have no knowledge of the first nine cards, our scenario is the same as if the tenth card was drawn first (without knowledge of any cars). There are four aces in a deck of 52 cards; the probability is $4/52 = 1/13$.

b. Now, the only other card we know about is the third card; an ace. Now we have only have 3 aces to choose from 51; the probability is $3/51 = 1/17$.

c. Now, the only other card we know about is the tenth card; an ace. Since we could have drawn the third and tenth cards in any order, this number is the same as part b: $1/17$.