Problem 1

Answer: (a) Consider the following algorithm for RobustBisect\((x, L, H)\) with \(t = 100\) times:

1. Call Less\((x, L)\) \(t\) times. If the majority is 1, return OUTOF RANGE.
2. Call Less\((x, \frac{L+H}{2})\) \(t\) times. If the majority is 1, return Low.
3. Call Less\((x, H)\) \(t\) times. If the majority is 1, return HIGH. Otherwise return OUTOF RANGE.

The running time of this algorithm is \(O(T)\). We prove the answer of each majority is correct with probability at least \(1 - 0.01\).

Consider the first majority function. We use \(X_i = 1\) to indicate whether the \(i\)-th call of Less\((x, L)\) is correct or not. So \(X_1, \ldots, X_t\) are independent; and \(E[X_i] = \frac{3}{4}\). From the Chernoff bound, the probability of the 1st majority is wrong is at most \(\Pr[\sum_{i \leq t} X_i \leq t/2] \leq e^{-(1/3)^2/2} \leq e^{-t/18} \leq e^{-5} < 0.01\).

(b) Consider the following procedure to find Opt\((x)\):

1. \(H(1) = 0, T(1) = n, P(1) = 1, i = 1\)
2. while \((H(i) + 1 < T(i))\)
3. \(\text{Let } mid = \frac{H(i)+T(i)}{2}\) and res=RobustBisect\((x, H(i), T(i))\).
4. If res=Low Then \(H(i+1) = H(i), T(i+1) = mid, P(i+1) = i\).
5. else if res=High Then \(H(i+1) = mid, T(i+1) = T(i), P(i+1) = i\).
6. else \(H(i+1) = H(P(i)), T(i+1) = T(P(i)), P(i+1) = P(P(i))\).
7. \(i = i + 1\)
8. END WHILE
9. return \(H(i)\)

We consider the first moment \(i\) when RobustBisect is correct at least \(\log n\) more times than it is incorrect in the first \(i\) callings. Let \(a_i\) be the number of Low and High that are correct from RobustBisect and \(b_i\) be the number of OUTRANGE that are correct. At the same time, we use \(c_i\) to denote the number of Low and High that are incorrect from RobustBisect and \(d_i\) to denote the number of OUTRANGE that are incorrect. Without loss of generality, we assume \(n = 2^l\) and neglect the useless OUTRANGE when \(H(i) = 0\) and \(T(i) = 2^l\).

We know \(a_i + b_i \geq c_i + d_i + l\). However we always have \(b_i \leq c_i\). Hence \(a_i + c_i - b_i - d_i \geq a_i + b_i - c_i - d_i \geq l\). We have \(T(i) - H(i) = 2^{l-(a_i+c_i-b_i-d_i)} = 2^{l-l} = 1\) such that the algorithm will output some element.
Next we argue it is the correct answer because the last calling of \textbf{RobustBisect} is correct. Otherwise, in the \((i-1)\)st callings of \textbf{RobustBisect}, \textbf{RobustBisect} is correct at least \(\log n + 1\) more times than it is incorrect such that there is a moment \(i' < i\) such that \textbf{RobustBisect} is correct at least \(\log n\) more times than it is incorrect, which is contradicted to our assumption of \(i\).

(c) We add a count \(t\) in the algorithm of (b) to make sure it finds the correct answer with high probability.

1. \(H(1) = 0, T(1) = n, P(1) = 1, i = 1, t = 0, m = 100(\log n + \alpha \log n), \alpha = c\)
2. while \((t \leq \alpha \log n \text{ and } i \leq m)\)
3. \hspace{1em} If \(T(i) = H(i) + 1\) then
4. \hspace{2em} Ask \textbf{RobustBisect}(\(x, H(i) - 1, T(i)\)) or \textbf{RobustBisect}(\(x, H(i), T(i) + 1\)) arbitrarily.
5. \hspace{2em} If the answer indicates \(T(i) = \text{opt}(x)\) then \(t = t + 1, H(i + 1) = H(i), T(i + 1) = T(i), P(i + 1) = P(i)\).
6. \hspace{2em} else if \(t > 0\) then \(t = t - 1, H(i + 1) = H(i), T(i + 1) = T(i), P(i + 1) = P(i)\).
7. \hspace{2em} else \(H(i + 1) = H(P(i)), T(i + 1) = T(P(i)), P(i + 1) = P(P(i))\).
8. \hspace{1em} End If
9. \hspace{1em} ELSE
10. \hspace{2em} Let \(mid = \frac{H(i) + T(i)}{2}\) and res=\textbf{RobustBisect}(\(x, H(i), T(i)\)).
11. \hspace{2em} If res=Low Then \(H(i + 1) = H(i), T(i + 1) = mid, P(i + 1) = i\).
12. \hspace{2em} else if res=High Then \(H(i + 1) = mid, T(i + 1) = T(i), P(i + 1) = i\).
13. \hspace{2em} else \(H(i + 1) = H(P(i)), T(i + 1) = T(P(i)), P(i + 1) = P(P(i))\).
14. \hspace{1em} END else
15. \hspace{1em} \(i = i + 1\)
16. END WHILE
17. return \(H(i)\)

From the same analysis, the algorithm will output an element when \textbf{RobustBisect} is correct at least \(\log n + \alpha \log n\) more times than it is incorrect. From this property and the Chernoff bound, the algorithm will end after \(m\) times callings to \textbf{RobustBisect} with probability at least \(1 - e^{-\frac{\alpha^2}{2}} \geq 1 - e^{-\log n - \alpha \log n} \geq 1 - n^{-c - 1}\).

Then we prove it outputs a wrong element \(x'\) with very small probability. For any wrong element that is outputted by the algorithm, it has to reach \(t = \alpha \log n\). Suppose the algorithm enters \(x' = T(i') = H(i')\) at moment \(i'\), let \(X_1, X_2, \cdots, X_i\) denote the result of \textbf{RobustBisect} is incorrect or not. We know \(E[X_j] = 1/4\) and \(\sum_{j=i'}^{i} X_j \geq \frac{i - i' + 1}{2}\). From the Chernoff bound, this probability is at most

\[
\Pr \left[ \sum_j X_j \geq (1 + 1) \sum_j E[X_j] \right] \leq e^{-\frac{\alpha}{64}} \leq n^{-c - 2}.
\]

Then we apply an union bound on \(i, i', x'\), which indicates the the algorithm outputs the correct element with probability at least \(1 - n^{-c - 1} - n \cdot m^2 \cdot n^{-c - 2} \geq 1 - n^{-c}\).
Problem 3 continued on next page...
Problem 4

**Answer:** (a) For any deterministic algorithm, we apply induction to prove there is one instance $T^h$ with height $h$ such that the algorithm reads all $n = 3^h$ leaves and outputs the label of the last leaf in the visit.

Base Case $h = 0$: We have to read the leaf to determine the value.

Induction $h = k + 1$: let $i_1$ be the first child of root the deterministic algorithm visited. We set the subtree of $i_1$ to be $T^h_k$, which is the instance the sub-algorithm on $i_1$ reads all leaves and outputs $0$. Let $i_2$ be the second child of root the algorithm visited. We set the subtree of $i_2$ to be $T^h_k$ and set the subtree of the last child of root to be $T^h_k$.

Let $j_1$ be the first subtree of $T^h_k, T^h_k, T^h_k$ that the deterministic algorithm determines the value and $j_2, j_3$ be the 2nd and 3rd subtree. We set the last visited leaf of $T^h_k$ to be $0$ and the last visited leaf of $T^h_k$ to be $1$ such that the deterministic algorithm has to read the whole subtree of $T^h_k$.

Hence for any deterministic, there is an instance that forces it to read all leaves.

(b) Let $f(n)$ be the number of leaves that the algorithm needs to visit. Because the oracle can tell the algorithm two subtrees with the same label, the algorithm only needs to visit two subtree to determine their value. Hence $f(n) = 2f(n/3)$ and $f(1) = 1$. So $f(n) = 2\log_3 n = n\log_3 2$.

(c) Consider the randomized algorithm that we random permute the three child of each node, and visit the subtrees of them until we can determine the value of the node. Observe that with probability $1/3$, we only need to visit two subtrees if the first two children provide the same answer. Hence $f(n) = \frac{1}{3} \cdot 2f(n/3) + \frac{2}{3} \cdot 3f(n/3) = 8/3 \cdot f(n/3)$ and $f(1) = 1$. Therefore $f(n) = (8/3)^{\log_3 n} = n^{\log_3(8/3)} \leq n^9$.

Problem 5

**Answer:** (a) Suppose the hash function uniform map “a” into $2^{64}$. Because they are pairwise independent, $Pr[h("a") = h("b")] = 2^{-64}$. For each initialization $i$ and $j(j \neq i)$, the probability that they are the same is $2^{-64}$. From a union bound, the probability that same value appears is at most $\binom{2000}{2} 2^{-64}$.

(b) Without randomization, we see 2000 repeats. In Python2 with randomization, we see 10-20 pairs repeat and 1-2 triples. In Python3 with randomization, we see 0 repeats.

(c) Python2 without randomization only has one hash function. Python3 with randomization should have at least $C \cdot \binom{2000}{2}$ hash functions with a huge $C$.

Let $m$ be the number of hash functions in Python2 with randomization. We estimate $m$ now and assume there are 15 pairs and 1 triples. Hence $\binom{2000}{2}/m \approx 15 \cdot \binom{2}{2} + 1 \cdot \binom{3}{2}$, which indicates $m \approx 10^5$. 

Problem 5 continued on next page...
(d) With $10^6$ experiments, there are roughly 2000 collisions. Hence the fraction is about $p = 0.002$. Let $m$ be the number of experiments to guarantee the error is small with high probability. Let $X_i$ the result of one experiment is collided or not and $X = \sum_{i=1}^{m} X_i$. So $E[X] = pm$ and $Var[X] = mp(1 - p)$. We use Chebyshev’s inequality:

$$\Pr[|X - E[X]| \geq 2E[X]] \leq \frac{Var[X]}{(2E[X])^2} = \frac{mp(1 - p)}{0.04p^2m^2}.$$ 

Setting $m = 20,000 \geq 12,500$, we know this probability is less than 0.01.

There are multiple ways to attack a website such as submitting lots of information/data with colliding hash value to make the hash-based data structure unresponsive.