CSC265 Fall 2020 Homework Assignment 5

Solutions

The list of people with whom I discussed this homework assignment: Kunal Chawla

A family of hash functions $\mathcal{H}$ from U to $\{0,\ldots,m-1\}$ is *pairwise independent* if for every two distinct keys $x_1, x_2 \in U$ and for every $y_1, y_2 \in \{0,\ldots,m-1\}$,

$$\operatorname*{Prob}_{h\in\mathcal{H}}[h(x_1) = y_1 \text{ and } h(x_2) = y_2] = 1/m^2.$$

Let $\mathcal{H}$ be a pairwise independent family of hash functions from U to $\{0,\ldots,m-1\}$.

1. Prove that $\mathcal{H}$ is universal.

   Solution: Let $x, y \in U$ be arbitrary. We will show that $\operatorname*{Prob}_{h\in\mathcal{H}}[h(x) = h(y)] = 1/m$. Note that we can partition the set $\mathcal{H}' = \{h \in \mathcal{H} : h(x) = h(y)\}$ into union of $\mathcal{H}_i = \{h \in \mathcal{H} : h(x) = h(y) = i\}$, for $i \in \{0,\ldots,m-1\}$. So $\mathcal{H}' = \bigsqcup_{i=0}^{m-1} \mathcal{H}_i$. And we know by $\mathcal{H}$ being pairwise independent that $\operatorname{Prob}\mathcal{H}_i = \operatorname*{Prob}_{h\in\mathcal{H}}[h(x) = i \text{ and } h(y) = i] = 1/m^2$. Therefore, we can compute

   $$\operatorname*{Prob}_{h\in\mathcal{H}}[h(y) = h(x)] = \operatorname{Prob}\mathcal{H}' = \operatorname{Prob}\bigsqcup_{i=0}^{m-1}\mathcal{H}_i = \sum_{i=0}^{m-1}\operatorname{Prob}\mathcal{H}_i = m(1/m^2) = 1/m.$$

2. Let $u = |U|$ and let $m = u^3$. Prove that $\operatorname*{Prob}_{h\in\mathcal{H}}[h \text{ is perfect for } U] > 1 - 1/u$.

   Solution: In this question, let A^c denote $\mathcal{H} \setminus A$ for $A \subseteq \mathcal{H}$. Define $\mathcal{H}_{x,y} = \{h \in \mathcal{H} : h(x) \neq h(y)\}$, for $x, y \in U$. Notice that the set of hash functions in $\mathcal{H}$ that are perfect for U is exactly $\bigcap_{x,y\in U} \mathcal{H}_{x,y}$. Thus, we have

   $$\begin{aligned}\operatorname*{Prob}_{h\in\mathcal{H}}[h \text{ is perfect for } U] &= \operatorname{Prob}\bigcap_{x,y\in U}\mathcal{H}_{x,y} \\ &= \operatorname{Prob}\Big(\bigcap_{\substack{x,y\in U\\ x\neq y}}\mathcal{H}_{x,y}\Big)^{c^c} \\ &= 1 - \operatorname{Prob}\Big(\bigcap_{\substack{x,y\in U\\ x\neq y}}\mathcal{H}_{x,y}\Big)^{c} \\ &= 1 - \operatorname{Prob}\bigcup_{\substack{x,y\in U\\ x\neq y}}(\mathcal{H}_{x,y})^c.\end{aligned}$$

We know

$$\begin{aligned}
\operatorname{Prob} \bigcup_{\substack{x,y \in U \\ x \neq y}} (\mathcal{H} \setminus \mathcal{H}_{x,y}) &\leq \sum_{\substack{x,y \in U \\ x \neq y}} \operatorname{Prob} {\mathcal{H}_{x,y}}^c \\
&= \sum_{\substack{x,y \in U \\ x \neq y}} \frac{1}{m} \\
&= u(u-1)\frac{1}{u^3} \\
&< \frac{1}{u},
\end{aligned}$$

where the first inequality follows from probability, the second follows because ${\mathcal{H}_{x,y}}^c$ are those that hash x, y to the same slot, which has probability $1/m$ by Q1. The equality follows since there are $u(u-1)$ possible (x, y) pairs from U^2 where $x \neq y$.

Thus, combining the results from above, we get that $\operatorname{Prob}_{h \in \mathcal{H}}[h \text{ is perfect for } U] > 1 - 1/u$.

3. Let $k \in \{0, \ldots, m-1\}$. For any value $a \in U$, let $X_a : \mathcal{H} \to \{0, 1\}$ be the indicator variable such that $X_a(h) = 1$ if and only if $h(a) < k$.
   Let $S \subseteq U$ and let $Y = \sum\{X_a \mid a \in S\}$. Prove that

   $$\operatorname{var}[Y] \leq \operatorname*{E}_{h \in \mathcal{H}}[Y] = |S|k/m.$$

   You may use without proof any property of variance given in CLRS section C.3.

   Solution: We first show the following lemma.

   Lemma 1. *For $a \in U$, $\operatorname{E}[X_a] = k/m$.*

   Proof. By definition of expectation, we have $\operatorname{E}[X_a] = 1 \cdot \operatorname{Prob}_{h \in \mathcal{H}}[h(a) < k] + 0 \cdot \operatorname{Prob}_{h \in \mathcal{H}}[h(a) \geq k]$. Let $b \in U$. Define $\mathcal{H}'$ to be the set of hash functions in $\mathcal{H}$ mapping a to less than k. Notice that $\mathcal{H}'$ can be partitioned in the following way as $\bigsqcup_{i=0}^{k-1} \bigsqcup_{j=0}^{m-1} \{h \in \mathcal{H} : h(a) = i \text{ and } h(b) = j\}$.

   Thus, we have

   $$\begin{aligned}
   \operatorname{Prob}_{h \in \mathcal{H}}[h(a) < k] &= \operatorname{Prob} \mathcal{H}' \\
   &= \operatorname{Prob} \bigsqcup_{i=0}^{k-1} \bigsqcup_{j=0}^{m-1} \{h \in \mathcal{H} : h(a) = i \text{ and } h(b) = j\} \\
   &= \sum_{i=0}^{k-1} \sum_{j=0}^{m-1} \operatorname{Prob}\{h \in \mathcal{H} : h(a) = i \text{ and } h(b) = j\} \\
   &= \sum_{i=0}^{k-1} \sum_{j=0}^{m-1} 1/m^2 \quad \text{since } \mathcal{H} \text{ is pairwise independent} \\
   &= \sum_{i=0}^{k-1} 1/m = k/m.
   \end{aligned}$$

$\square$

Now, since $Y = \sum_{a \in S} X_a$, we have, by linearity of expectation and the lemma, that $\mathrm{E}[Y] = \sum_{a \in S} \mathrm{E}[X_a] = |S|k/m$. Also, by CLRS C.3, we can compute $\mathrm{var}[Y] = \mathrm{E}[Y^2] - \mathrm{E}[Y]^2$.

To find $\mathrm{E}[Y^2]$, we apply Y's definition to see that $\mathrm{E}[Y^2] = \mathrm{E}[(\sum_{a \in S} X_a)^2] = \mathrm{E}[\sum_{a_i, a_j \in S} X_{a_i} X_{a_j}] = \sum_{a_i, a_j \in S} \mathrm{E}[X_{a_i} X_{a_j}]$, where the last equality is by linearity of expectation.

Looking at the summand, we know if $a_i = a_j$, then $X_{a_i} = X_{a_j}$. Thus $X_{a_i} X_{a_j} = (X_{a_i})^2 = X_{a_i}$ since $1^2 = 1$ and $0^2 = 0$. Thus, if $a_i = a_j$, then $E[X_{a_i} X_{a_j}] = E[X_{a_i}] = k/m$. On the other hand, if $a_i \neq a_j$, then the value of $X_{a_i}(h) X_{a_j}(h)$ for a hash function h is 1 exactly when $h(a_i) < k$ and $h(a_j) < k$. Thus, we have k choices for the value of $h(a_i)$ and k for $h(a_j)$, given a total of k^2 choices. Also, given any such choice $l < k$, $w < k$, we have $\underset{h \in \mathcal{H}}{\mathrm{Prob}}[h(a_i) = l, h(a_j) = w] = 1/m^2$. Thus, if $a_i \neq a_j$, we have $\mathrm{E}[X_{a_i} X_{a_j}] = 1 \cdot \underset{h \in \mathcal{H}}{\mathrm{Prob}}[X_{a_i}(h) X_{a_j}(h) = 1] = \sum_{l, w < k} \underset{h \in \mathcal{H}}{\mathrm{Prob}}[h(a_i) = l, h(a_j) = w] = k^2/m^2$.

Now, we can finally evaluate

$$\begin{aligned}
\sum_{a_i, a_j \in S} \mathrm{E}[X_{a_i} X_{a_j}] &= \sum_{\substack{a_i, a_j \in S \\ a_i \neq a_j}} \mathrm{E}[X_{a_i} X_{a_j}] + \sum_{a_i \in S} \mathrm{E}[X_{a_i} X_{a_i}] \\
&= \sum_{\substack{a_i, a_j \in S \\ a_i \neq a_j}} k^2/m^2 + \sum_{a_i \in S} k/m \\
&= \frac{(|S|^2 - |S|)k^2}{m^2} + \frac{|S|k}{m} \\
&= \frac{|S|k(|S|k - k + m)}{m^2}.
\end{aligned}$$

Thus, we have $\mathrm{var}[Y] = \mathrm{E}[Y^2] - \mathrm{E}[Y]^2 = \frac{|S|k(|S|k-k+m)}{m^2} - \frac{|S|k^2}{m^2} = \frac{|S|k(m-k)}{m^2}$. We have $\frac{|S|k(m-k)}{m^2} \leq \mathrm{E}[Y]$ iff $m - k \leq m$, which is true. Hence, we can conclude

$$\mathrm{var}[Y] \leq \underset{h \in \mathcal{H}}{\mathrm{E}}[Y] = |S|k/m.$$

4. Consider the following algorithm that takes as input a sequence $a_1, \ldots, a_n$ of n elements from U and is supposed to return an estimate of the number d of distinct elements in the sequence. Here t is a parameter of the algorithm.

   Let $h \in \mathcal{H}$ be chosen uniformly at random.
   Determine the set T of the t smallest distinct elements in $\{h(a_i) \mid 1 \leq i \leq n\}$.
   If there are fewer than t distinct elements in $\{h(a_i) \mid 1 \leq i \leq n\}$,
   then return the size of this set;
   else let V be the largest element in T.
   Return $D = (t - \frac{1}{2})(m-1)/V$.

Explain how to implement this algorithm so that it takes $O(n \log t)$ time and uses $O(t)$ words of memory, each storing $O(\log m)$ bits.
Assume that a hash function can be stored in $O(1)$ words of memory and that it can be evaluated on an element of U in $O(1)$ time.

Solution: To implement this algorithm, we need to specify a data structure and how the algorithm operates on the data structure. Let our data structure be a red-black tree T, together with an integer counter *size* and a pointer *max* to a node in T. At the beginning of this algorithm, T is empty, $\mathit{size} = 0$, and $\mathit{max} = \text{NIL}$. We denote the key of the node *max* is pointing to by $*\mathit{max}$, and it is $-\infty$ if $\mathit{max} = \text{NIL}$. We also denote, if a key x is in T, the node corresponding to x as $n(x)$.

For every input a_i, we compute $h(a_i)$. Then, if $\mathit{size} < t$, we search to see if $h(a_i)$ is already in T, if it's not already there, we add $h(a_i)$ to T, set $\mathit{max} = n(h(a_i))$ if $h(a_i) > *\mathit{max}$, and increment *size*. If $\mathit{size} = t$, then we check whether $h(a_i) < *\mathit{max}$ (if $h(a_i)$ is at least $*\mathit{max}$, then there is no point inserting) and whether $h(a_i)$ is not in T. If the two conditions hold, we delete $n(*\mathit{max})$ from the tree, insert $h(a_i)$ into T, and set *max* to the rightmost node in T (this is done by traversing down the right, until there are no more right children).

After doing this for every input, we will have a set of at most t h-values in T. To be consistent with the algorithm, we set $V = *\mathit{max}$. Now, we do exactly what is specified: if $\mathit{size} < t$, we return *size*; otherwise, we return $D = (t - \frac{1}{2})(m-1)/V$.

To see that this algorithm runs in $O(n \log t)$, notice that for each for each a_i, $1 \leq i \leq n$, we do a constant number of RB-tree insert, delete, and search on T. More specifically, we know that: when $\mathit{size} < t$, we perform a search and insert; when $\mathit{size} = t$, we perform a search, at most one deletion, at most one insert, and at most one retrieval for the rightmost node. We know from CLRS13 that insert and delete are $O(\log t)$ (t is the maximum number of nodes in T), and, since the height of T is $O(\log t)$, binary search and retrieval of the rightmost node takes $O(\log t)$ time too. Finally, there is a constant number of comparisons between *size*, t, $h(a_i)$, and $*\mathit{max}$. Therefore, we can conclude each of the n inputs take $O(\log t)$ time to process, so the overall algorithm is $O(n \log t)$ (notice that we have a constant number of arithmetic and assignment regarding D and V after, but since there are a constant number of them, they don't contribute to the asymptotic runtime).

For space complexity, notice that T has at most size t, and each node stores a color (1 bit), an integer key ($\log_2 m$ bits). Other than T, we have an integer counter *size* which is at most t ($\log_2 m$ bits) and a pointer *max* to a tree node ($\log_2 m$ bits). Thus, the space complexity is overall $t(1 + \log_2 m) + 2\log_2 m \in O(t \log m)$.

5. Give a brief, intuitive explanation why $\mathrm{E}[D]$ is approximately d.

Solution: Notice that if we set $S = \{a_1, \ldots, a_n\}$, then $d = |S|$. For any $0 \leq V \leq m-1$, we know by Q3 that given a hash function $h \in \mathcal{H}$, we should expect h to map $|S|(V+1)/m$ elements of S to less than to $V+1$—we called this number t in Q4 (beware the $V+1$ might be m, which is not covered by Q3, however, the statement in Q3 should still hold since $V+1$ might cancel with m). So, given $|S|(V+1)/m \approx t$, solving for $|S|$ gives $|S| \approx t(m)/(V+1) \approx (t - 1/2)(m-1)/V$ when t and m are large.