COL866: Foundations of Data Science

Ragesh Jaiswal, IITD

Algorithms for Massive Data Problems: Streaming algorithm

Problem

Design an algorithm for finding the majority element (in case there exists one).

- We want a time the element that appears in the stream more than $n/2$ times.
- Claim Any deterministic algorithm requires $\Omega(\min(n, m))$ space.
- We can do better if we relax our requirement in the following manner:
  - In case there is a majority element, then the algorithm should output this element.
  - In case there is no majority element, the algorithm is allowed to output any element.

Problem

Design an algorithm for finding the majority element (in case there exists one).

- Claim Any deterministic algorithm requires $\Omega(\min(n, m))$ space.
- We can do better if we relax our requirement in the following manner:
  - In case there is a majority element, then the algorithm should output this element.
  - In case there is no majority element, the algorithm is allowed to output any element.

Algorithm

Majority($a_1, ..., a_n$)
- $s \leftarrow a_1$ and $ctr \leftarrow 1$
- For $i = 2$ to n
  - if $(a_i = s) ctr \leftarrow ctr + 1$
  - elseif $(ctr \neq 0)$ $ctr \leftarrow ctr - 1$
  - else $\{s \leftarrow a_i; ctr \leftarrow 1\}$
- return(s)

Problem

Design an algorithm for finding all the elements of the steam that have frequency more than $\frac{n}{k+1}$.

- As in the case for majority, we will produce a list of elements (along with an approximate value of its frequency) such that:
  - If the frequency of an element is more than $\frac{n}{k+1}$, then this element appears in the list.

Algorithm

`Frequency`$(a_1, ..., a_n)$

- $L \leftarrow \{\}$
- For $i = 1$ to n
  - If $(a_i \in L)$ ctr_{a_i}++
  - elseif $(|L| < k)$ $\{L \leftarrow L \cup \{a_i\}; ctr_{a_i} \leftarrow 1\}$
  - else
    - decrement all the counters by 1
    - if some counter becomes 0, delete the element from the list.
- return the list L and the counters.

Problem

Design an algorithm for finding all the elements of the steam that have frequency more than $\frac{n}{k+1}$.

Algorithm

`Frequency`$(a_1, ..., a_n)$
- $L \leftarrow \{\}$
- For $i = 1$ to n
  - If $(a_i \in L)$ $ctr_{a_i}{++}$
  - elseif $(|L| < k)$ $\{L \leftarrow L \cup \{a_i\}; ctr_{a_i} \leftarrow 1\}$
  - else
    - decrement all the counters by 1
    - if some counter becomes 0, delete the element from the list.
- return the list L and the counters.

Theorem

At the end of the algorithm `Frequency`*, for each* $s \in \{1, ..., m\}$*, its counter on the list* $\tilde{f}_s$ *satisfies* $\tilde{f}_s \in [f_s - \frac{n}{k+1}, f_s]$*. If some* s *does not occur on the list, its counter is* 0 *and the theorem asserts that* $f_s \leq \frac{n}{k+1}$*. Here* f_s *denotes true frequency of elements* s *in the stream.*

Problem

Let f_s denote the frequency of a data item $s \in \{1, ..., m\}$ in the stream of data $a_1, ..., a_n$. Design an algorithm to give an estimate of $\sum_{i=1}^{m} f_s^2$. This is known as the second moment of the stream.

- Simple techniques use space that is linear either in m or n.

Problem

Let f_s denote the frequency of a data item $s \in \{1, ..., m\}$ in the stream of data $a_1, ..., a_n$. Design an algorithm to give an estimate of $\sum_{i=1}^{m} f_s^2$. This is known as the second moment of the stream.

- Simple techniques use space that is linear either in m or n.
- Here is a simple randomness based idea:
  - Before examining the stream, randomly pick $x_s \in \{+1, -1\}$ for every $s \in \{1, ..., m\}$
  - Maintain a single sum S. When a_i arrives, ddd x_s to S if $a_i = s$.
  - After the end of the stream, output S^2.

Problem

Let f_s denote the frequency of a data item $s \in \{1, ..., m\}$ in the stream of data $a_1, ..., a_n$. Design an algorithm to give an estimate of $\sum_{i=1}^{m} f_s^2$. This is known as the second moment of the stream.

- Simple techniques use space that is linear either in m or n.
- Here is a simple randomness based idea:
  - Before examining the stream, randomly pick $x_s \in \{+1, -1\}$ for every $s \in \{1, ..., m\}$
  - Maintain a single sum S. When a_i arrives, ddd x_s to S if $a_i = s$.
  - After the end of the stream, output S^2.
- Note that S is a random variable where the randomness is over the choice of x_s.
- Claim: $S = \sum_{i=1}^{m} x_s f_s$.

Problem

Let f_s denote the frequency of a data item $s \in \{1, ..., m\}$ in the stream of data $a_1, ..., a_n$. Design an algorithm to give an estimate of $\sum_{i=1}^{m} f_s^2$. This is known as the second moment of the stream.

- Simple techniques use space that is linear either in m or n.
- Here is a simple randomness based idea:
  - Before examining the stream, randomly pick $x_s \in \{+1, -1\}$ for every $s \in \{1, ..., m\}$
  - Maintain a single sum S. When a_i arrives, ddd x_s to S if $a_i = s$.
  - After the end of the stream, output S^2.
- Note that S is a random variable where the randomness is over the choice of x_s.
- Claim: $S = \sum_{i=1}^{m} x_s f_s$.
- Question: What is the expected value of S?

Problem

Let f_s denote the frequency of a data item $s \in \{1, ..., m\}$ in the stream of data $a_1, ..., a_n$. Design an algorithm to give an estimate of $\sum_{i=1}^{m} f_s^2$. This is known as the second moment of the stream.

- Simple techniques use space that is linear either in m or n.
- Here is a simple randomness based idea:
  - Before examining the stream, randomly pick $x_s \in \{+1, -1\}$ for every $s \in \{1, ..., m\}$
  - Maintain a single sum S. When a_i arrives, ddd x_s to S if $a_i = s$.
  - After the end of the stream, output S^2.
- Note that S is a random variable where the randomness is over the choice of x_s.
- Claim: $S = \sum_{i=1}^{m} x_s f_s$.
- Question: What is the expected value of S? $\mathbf{E}[S] = 0$

Problem

Let f_s denote the frequency of a data item $s \in \{1, ..., m\}$ in the stream of data $a_1, ..., a_n$. Design an algorithm to give an estimate of $\sum_{i=1}^{m} f_s^2$. This is known as the second moment of the stream.

- Simple techniques use space that is linear either in m or n.
- Here is a simple randomness based idea:
  - Before examining the stream, randomly pick $x_s \in \{+1, -1\}$ for every $s \in \{1, ..., m\}$
  - Maintain a single sum S. When a_i arrives, ddd x_s to S if $a_i = s$.
  - After the end of the stream, output S^2.
- Note that S is a random variable where the randomness is over the choice of x_s.
- Claim: $S = \sum_{i=1}^{m} x_s f_s$.
- Question: What is the expected value of S? $\mathbf{E}[S] = 0$
- Claim: $\mathbf{E}[S^2] = \sum_{i=1}^{m} f_s^2$.

Problem

Let f_s denote the frequency of a data item $s \in \{1, ..., m\}$ in the stream of data $a_1, ..., a_n$. Design an algorithm to give an estimate of $\sum_{i=1}^{m} f_s^2$. This is known as the second moment of the stream.

- Simple techniques use space that is linear either in m or n.
- Here is a simple randomness based idea:
  - Before examining the stream, randomly pick $x_s \in \{+1, -1\}$ for every $s \in \{1, ..., m\}$
  - Maintain a single sum S. When a_i arrives, ddd x_s to S if $a_i = s$.
  - After the end of the stream, output S^2.
- Note that S is a random variable where the randomness is over the choice of x_s.
- Claim: $S = \sum_{i=1}^{m} x_s f_s$.
- Question: What is the expected value of S? $\mathbf{E}[S] = 0$
- Claim: $\mathbf{E}[S^2] = \sum_{i=1}^{m} f_s^2$.

Proof sketch

$\mathbf{E}\left[\left(\sum_{s=1}^{m} x_s f_s\right)^2\right] = \mathbf{E}[\sum_{s=1}^{m} x_s^2 f_s^2] + 2\mathbf{E}[\sum_{s<t} x_s x_t f_s f_t] = \sum_{s=1}^{m} f_s^2$ (using pairwise independence).

Problem

Let f_s denote the frequency of a data item $s \in \{1, ..., m\}$ in the stream of data $a_1, ..., a_n$. Design an algorithm to give an estimate of $\sum_{i=1}^{m} f_s^2$. This is known as the second moment of the stream.

- Simple techniques use space that is linear either in m or n.
- Here is a simple randomness based idea:
  - Before examining the stream, randomly pick $x_s \in \{+1, -1\}$ for every $s \in \{1, ..., m\}$
  - Maintain a single sum S. When a_i arrives, ddd x_s to S if $a_i = s$.
  - After the end of the stream, output S^2.
- Note that S is a random variable where the randomness is over the choice of x_s.
- Claim: $S = \sum_{i=1}^{m} x_s f_s$.
- Question: What is the expected value of S? $\mathbf{E}[S] = 0$
- Claim: $\mathbf{E}[S^2] = \sum_{i=1}^{m} f_s^2$.
- So, S^2 is an unbiased estimate of the second moment. That is, it has the right expectation.
- In order to get a high probability statement, we would want to apply Chebychev and to be able to apply Chebychev, we would need $\mathbf{E}[S^4]$.

Problem

Let f_s denote the frequency of a data item $s \in \{1, ..., m\}$ in the stream of data $a_1, ..., a_n$. Design an algorithm to give an estimate of $\sum_{i=1}^{m} f_s^2$. This is known as the second moment of the stream.

- Here is a simple randomness based idea:
  - Before examining the stream, randomly pick $x_s \in \{+1, -1\}$ for every $s \in \{1, ..., m\}$
  - Maintain a single sum S. When a_i arrives, ddd x_s to S if $a_i = s$.
  - After the end of the stream, output S^2.
- Claim: $\mathbf{E}[S^2] = \sum_{i=1}^{m} f_s^2$.
- So, S^2 is an unbiased estimate of the second moment. That is, it has the right expectation.
- In order to get a high probability statement, we would want to apply Chebychev and to be able to apply Chebychev, we would need $\mathbf{E}[S^4]$.

Calculations

$$\begin{aligned}
\mathbf{E}[S^4] &= \mathbf{E}\left[\left(\sum_{s=1}^{m} x_s f_s\right)^4\right] = \mathbf{E}\left[\sum_{1 \leq s,t,u,v \leq m} x_s x_t x_u x_v f_s f_t f_u f_v\right] \\
&= \binom{4}{2} \mathbf{E}\left[\sum_{s=1}^{m} \sum_{t=s+1}^{m} x_s^2 x_t^2 f_s^2 f_t^2\right] + \mathbf{E}\left[\sum_{s=1}^{m} x_s^4 f_s^4\right] \\
&= 6 \sum_{s=1}^{m} \sum_{t=s+1}^{m} f_s^2 f_t^2 + \sum_{s=1}^{m} f_s^4 \leq 3\left(\sum_{s=1}^{m} f_s^2\right)^2 = 3(\mathbf{E}[S^2])^2
\end{aligned}$$

Problem

Let f_s denote the frequency of a data item $s \in \{1, ..., m\}$ in the stream of data $a_1, ..., a_n$. Design an algorithm to give an estimate of $\sum_{i=1}^{m} f_s^2$. This is known as the second moment of the stream.

- Here is a simple randomness based idea:
  - Before examining the stream, randomly pick $x_s \in \{+1, -1\}$ for every $s \in \{1, ..., m\}$
  - Maintain a single sum S. When a_i arrives, ddd x_s to S if $a_i = s$.
  - After the end of the stream, output S^2.
- Claim: $\mathbf{E}[S^2] = \sum_{i=1}^{m} f_s^2$.
- So, S^2 is an unbiased estimate of the second moment. That is, it has the right expectation.
- In order to get a high probability statement, we would want to apply Chebychev and to be able to apply Chebychev, we would need $\mathbf{E}[S^4]$.
- So, we have $\mathbf{Var}[S^2] = \mathbf{E}[S^4] - (\mathbf{E}[S^2])^2 \leq 2(\mathbf{E}[S^2])^2$.

Calculations

$$\begin{aligned}
\mathbf{E}[S^4] &= \mathbf{E}\left[\left(\sum_{s=1}^{m} x_s f_s\right)^4\right] = \mathbf{E}\left[\sum_{1 \leq s,t,u,v \leq m} x_s x_t x_u x_v f_s f_t f_u f_v\right] \\
&= \binom{4}{2} \mathbf{E}\left[\sum_{s=1}^{m} \sum_{t=s+1}^{m} x_s^2 x_t^2 f_s^2 f_t^2\right] + \mathbf{E}\left[\sum_{s=1}^{m} x_s^4 f_s^4\right] \\
&= 6 \sum_{s=1}^{m} \sum_{t=s+1}^{m} f_s^2 f_t^2 + \sum_{s=1}^{m} f_s^4 \leq 3\left(\sum_{s=1}^{m} f_s^2\right)^2 = 3(\mathbf{E}[S^2])^2
\end{aligned}$$

Problem

Let f_s denote the frequency of a data item $s \in \{1, ..., m\}$ in the stream of data $a_1, ..., a_n$. Design an algorithm to give an estimate of $\sum_{i=1}^{m} f_s^2$. This is known as the second moment of the stream.

- Here is a simple randomness based idea:
  - Before examining the stream, randomly pick $x_s \in \{+1, -1\}$ for every $s \in \{1, ..., m\}$
  - Maintain a single sum S. When a_i arrives, ddd x_s to S if $a_i = s$.
  - After the end of the stream, output S^2.
- Claim: $\mathbf{E}[S^2] = \sum_{i=1}^{m} f_s^2$.
- So, S^2 is an unbiased estimate of the second moment. That is, it has the right expectation.
- In order to get a high probability statement, we would want to apply Chebychev and to be able to apply Chebychev, we would need $\mathbf{E}[S^4]$.
- So, we have $\mathbf{Var}[S^2] = \mathbf{E}[S^4] - (\mathbf{E}[S^2])^2 \leq 2(\mathbf{E}[S^2])^2$.
- Question: How do we utilise the above inequality to get a good estimate on the second moment?

Problem

Let f_s denote the frequency of a data item $s \in \{1, ..., m\}$ in the stream of data $a_1, ..., a_n$. Design an algorithm to give an estimate of $\sum_{i=1}^{m} f_s^2$. This is known as the second moment of the stream.

- Here is a simple randomness based idea:
  - Before examining the stream, randomly pick $x_s \in \{+1, -1\}$ for every $s \in \{1, ..., m\}$
  - Maintain a single sum S. When a_i arrives, ddd x_s to S if $a_i = s$.
  - After the end of the stream, output S^2.
- Claim: $\mathbf{E}[S^2] = \sum_{i=1}^{m} f_s^2$.
- So, S^2 is an unbiased estimate of the second moment. That is, it has the right expectation.
- In order to get a high probability statement, we would want to apply Chebychev and to be able to apply Chebychev, we would need $\mathbf{E}[S^4]$.
- So, we have $\mathbf{Var}[S^2] = \mathbf{E}[S^4] - (\mathbf{E}[S^2])^2 \leq 2(\mathbf{E}[S^2])^2$.
- Question: How do we utilise the above inequality to get a good estimate on the second moment?
  - Instead of maintaining one S, maintain $r = \frac{2}{\varepsilon^2 \delta}$ independent $S_1, ..., S_r$ and then output $A = \frac{S_1^2 + ... + S_r^2}{r}$ at the end.
  - Claim: $\mathbf{Pr}[|A - \sum_{s=1}^{m} f_s^2| > \varepsilon \sum_{s=1}^{m} f_s^2] \leq \delta$.

Problem

Let f_s denote the frequency of a data item $s \in \{1, ..., m\}$ in the stream of data $a_1, ..., a_n$. Design an algorithm to give an estimate of $\sum_{i=1}^{m} f_s^2$. This is known as the second moment of the stream.

- Here is a simple randomness based idea:
  - Before examining the stream, randomly pick $x_s^t \in \{+1, -1\}$ for every $s \in \{1, ..., m\}$ and every $t \in \{1, ..., r\}$, where $r = \frac{2}{\varepsilon^2 \delta}$.
  - Maintain sums $S_1, ..., S_r$. When a_i arrives, ddd x_s^t to S_t if $a_i = s$.
  - After the end of the stream, output $\frac{S_1^2 + ... + S_r^2}{r}$.
- Claim: $\mathbf{E}[S^2] = \sum_{i=1}^{m} f_s^2$.
- So, S^2 is an unbiased estimate of the second moment. That is, it has the right expectation.
- In order to get a high probability statement, we would want to apply Chebychev and to be able to apply Chebychev, we would need $\mathbf{E}[S^4]$.
- So, we have $\mathbf{Var}[S^2] = \mathbf{E}[S^4] - (\mathbf{E}[S^2])^2 \leq 2(\mathbf{E}[S^2])^2$.
- Question: How do we utilise the above inequality to get a good estimate on the second moment?
  - Instead of maintaining one S, maintain $r = \frac{2}{\varepsilon^2 \delta}$ independent $S_1, ..., S_r$ and then output $A = \frac{S_1^2 + ... + S_r^2}{r}$ at the end.
  - Claim: $\mathbf{Pr}[|A - \sum_{s=1}^{m} f_s^2| > \varepsilon \sum_{s=1}^{m} f_s^2] \leq \delta$.

Problem

Let f_s denote the frequency of a data item $s \in \{1, ..., m\}$ in the stream of data $a_1, ..., a_n$. Design an algorithm to give an estimate of $\sum_{i=1}^{m} f_s^2$. This is known as the second moment of the stream.

- Here is a simple randomness based idea:
  - Before examining the stream, randomly pick $x_s^t \in \{+1, -1\}$ for every $s \in \{1, ..., m\}$ and every $t \in \{1, ..., r\}$, where $r = \frac{2}{\varepsilon^2 \delta}$.
  - Maintain sums $S_1, ..., S_r$. When a_i arrives, ddd x_s^t to S_t if $a_i = s$.
  - After the end of the stream, output $\frac{S_1^2 + ... + S_r^2}{r}$.
- Question How much space does the above algorithm require?

Problem

Let f_s denote the frequency of a data item $s \in \{1, ..., m\}$ in the stream of data $a_1, ..., a_n$. Design an algorithm to give an estimate of $\sum_{i=1}^{m} f_s^2$. This is known as the second moment of the stream.

- Here is a simple randomness based idea:
  - Before examining the stream, randomly pick $x_s^t \in \{+1, -1\}$ for every $s \in \{1, ..., m\}$ and every $t \in \{1, ..., r\}$, where $r = \frac{2}{\varepsilon^2 \delta}$.
  - Maintain sums $S_1, ..., S_r$. When a_i arrives, ddd x_s^t to S_t if $a_i = s$.
  - After the end of the stream, output $\frac{S_1^2 + ... + S_r^2}{r}$.
- Question: How much space does the above algorithm require? $O(rm)$
- Idea to save space: Use hash function $h : \{1, ..., m\} \rightarrow \{+1, -1\}$.
- Question: Suppose we use the random hash function family. Then how much space do we require? $O(rm)$ since describing a hash function uses $O(m)$ space and we need r of them
- Question: So, how do we get a reduction on the space?

Problem

Let f_s denote the frequency of a data item $s \in \{1, ..., m\}$ in the stream of data $a_1, ..., a_n$. Design an algorithm to give an estimate of $\sum_{i=1}^{m} f_s^2$. This is known as the second moment of the stream.

- Here is a simple randomness based idea:
  - Before examining the stream, randomly pick $x_s^t \in \{+1, -1\}$ for every $s \in \{1, ..., m\}$ and every $t \in \{1, ..., r\}$, where $r = \frac{2}{\varepsilon^2 \delta}$.
  - Maintain sums $S_1, ..., S_r$. When a_i arrives, ddd x_s^t to S_t if $a_i = s$.
  - After the end of the stream, output $\frac{S_1^2 + ... + S_r^2}{r}$.
- Question: How much space does the above algorithm require? $O(rm)$
- Idea to save space: Use hash function $h : \{1, ..., m\} \to \{+1, -1\}$.
- Question: Suppose we use the random hash function family. Then how much space do we require? $O(rm)$ since describing a hash function uses $O(m)$ space and we need r of them
- Question: So, how do we get a reduction on the space?
  - An important thing to notice in the analysis is that we did not require full independence property of variables x_s but only 4-wise independence.
  - Main idea: Use a 4-wise independent hash function family.

Problem

Let f_s denote the frequency of a data item $s \in \{1, ..., m\}$ in the stream of data $a_1, ..., a_n$. Design an algorithm to give an estimate of $\sum_{i=1}^{m} f_s^2$. This is known as the second moment of the stream.

- Here is a simple randomness based idea:
  - Before examining the stream, randomly pick $x_s^t \in \{+1, -1\}$ for every $s \in \{1, ..., m\}$ and every $t \in \{1, ..., r\}$, where $r = \frac{2}{\varepsilon^2 \delta}$.
  - Maintain sums $S_1, ..., S_r$. When a_i arrives, ddd x_s^t to S_t if $a_i = s$.
  - After the end of the stream, output $\frac{S_1^2 + ... + S_r^2}{r}$.
- Question: How much space does the above algorithm require? $O(rm)$
- Idea to save space: Use hash function $h : \{1, ..., m\} \to \{+1, -1\}$.
- Question: Suppose we use the random hash function family. Then how much space do we require? $O(rm)$ since describing a hash function uses $O(m)$ space and we need r of them
- Question: So, how do we get a reduction on the space?
  - An important thing to notice in the analysis is that we did not require full independence property of variables x_s but only 4-wise independence.
  - Main idea: Use a 4-wise independent hash function family.
  - There exists a 4-wise independent hash function family such that describing a hash function from this family takes $O(\log m)$ bits.

Problem

Let f_s denote the frequency of a data item $s \in \{1, ..., m\}$ in the stream of data $a_1, ..., a_n$. Design an algorithm to give an estimate of $\sum_{i=1}^{m} f_s^2$. This is known as the second moment of the stream.

- Here is a streaming algorithm that uses $O(\log m)$ space:
  - Before examining the stream, pick hash functions $h_1, ..., h_r$ independently and at random from a 4-wise independent hash function family $\mathcal{H}$, where $r = \frac{2}{\varepsilon^2 \delta}$.
  - Maintain sums $S_1, ..., S_r$. When a_i arrives, add $h_t(a_i)$ to S_t.
  - After the end of the stream, output $\frac{S_1^2 + ... + S_r^2}{r}$.
- Question: How much space does the above algorithm require? $O(r \log m)$
- Main idea:
  - An important thing to notice in the analysis is that we did not require full independence property of variables x_s but only 4-wise independence.
  - Main idea: Use a 4-wise independent hash function family.
  - There exists a 4-wise independent hash function family such that describing a hash function from this family takes $O(\log m)$ bits.

Problem

Let f_s denote the frequency of a data item $s \in \{1, ..., m\}$ in the stream of data $a_1, ..., a_n$. Design an algorithm to give an estimate of $\sum_{i=1}^{m} f_s^2$. This is known as the second moment of the stream.

- Here is a streaming algorithm that uses $O(\log m)$ space:
  - Before examining the stream, pick hash functions $h_1, ..., h_r$ independently and at random from a 4-wise independent hash function family $\mathcal{H}$, where $r = \frac{2}{\varepsilon^2 \delta}$.
  - Maintain sums $S_1, ..., S_r$. When a_i arrives, add $h_t(a_i)$ to S_t.
  - After the end of the stream, output $\frac{S_1^2 + ... + S_r^2}{r}$.
- Question: How much space does the above algorithm require? $O(r \log m)$
- Main idea:
  - An important thing to notice in the analysis is that we did not require full independence property of variables x_s but only 4-wise independence.
  - Main idea: Use a 4-wise independent hash function family.
  - There exists a 4-wise independent hash function family such that describing a hash function from this family takes $O(\log m)$ bits.

Problem

There exists a 4-wise independent hash function family consisting of functions mapping $\{1, ..., m\}$ to $\{+1, -1\}$ such that describing a hash function from this family takes $O(\log m)$ bits.

- Lert $m = 2^k$. (*The arguments can be made to work even if m is not a power of* 2).
- Fact: There is a finite field F with $2^k = m$ elements. The elements of the field may be represented using k bits.
- Polynomial interpolation: For any four distinct points $a_1, a_2, a_3, a_4 \in F$ and any four points (not necessarily unique) $b_1, b_2, b_3, b_4 \in F$, there is a unique polynomial $f(x) = f_0 + f_1 x + f_2 x^2 + f_3 x^3$ of degree at most 3 such that $f(a_1) = b_1; f(a_2) = b_2; f(a_3) = b_3; f(a_4) = b_4$.

Problem

There exists a 4-wise independent hash function family consisting of functions mapping $\{1, ..., m\}$ to $\{+1, -1\}$ such that describing a hash function from this family takes $O(\log m)$ bits.

- Lert $m = 2^k$. (*The arguments can be made to work even if m is not a power of* 2).
- Fact: There is a finite field F with $2^k = m$ elements. The elements of the field may be represented using k bits.
- Polynomial interpolation: For any four distinct points $a_1, a_2, a_3, a_4 \in F$ and any four points (not necessarily unique) $b_1, b_2, b_3, b_4 \in F$, there is a unique polynomial $f(x) = f_0 + f_1 x + f_2 x^2 + f_3 x^3$ of degree at most 3 such that $f(a_1) = b_1; f(a_2) = b_2; f(a_3) = b_3; f(a_4) = b_4$.
- For $f_0, f_1, f_2, f_3 \in F$ define function $h_{f_0, f_1, f_2, f_3}(s) = Lead(f_0 + f_1 s + f_2 s^2 + f_3 s^3)$, where $Lead(.)$ denotes the leading bit of input.
- $\mathcal{H} = \{h_{f_0, f_1, f_2, f_3} | f_0, f_1, f_2, f_3 \in F\}$.

Problem

There exists a 4-wise independent hash function family consisting of functions mapping $\{1, ..., m\}$ to $\{+1, -1\}$ such that describing a hash function from this family takes $O(\log m)$ bits.

- Lert $m = 2^k$. (*The arguments can be made to work even if m is not a power of* 2).
- Fact: There is a finite field F with $2^k = m$ elements. The elements of the field may be represented using k bits.
- Polynomial interpolation: For any four distinct points $a_1, a_2, a_3, a_4 \in F$ and any four points (not necessarily unique) $b_1, b_2, b_3, b_4 \in F$, there is a unique polynomial $f(x) = f_0 + f_1 x + f_2 x^2 + f_3 x^3$ of degree at most 3 such that $f(a_1) = b_1; f(a_2) = b_2; f(a_3) = b_3; f(a_4) = b_4$.
- For $f_0, f_1, f_2, f_3 \in F$ define function $h_{f_0,f_1,f_2,f_3}(s) = Lead(f_0 + f_1 s + f_2 s^2 + f_3 s^3)$, where $Lead(.)$ denotes the leading bit of input.
- $\mathcal{H} = \{h_{f_0,f_1,f_2,f_3} | f_0, f_1, f_2, f_3 \in F\}$.

- Fact: There is a finite field F with $2^k = m$ elements. The elements of the field may be represented using k bits.
- Polynomial interpolation: For any four distinct points $a_1, a_2, a_3, a_4 \in F$ and any four points (not necessarily unique) $b_1, b_2, b_3, b_4 \in F$, there is a unique polynomial $f(x) = f_0 + f_1 x + f_2 x^2 + f_3 x^3$ of degree at most 3 such that $f(a_1) = b_1; f(a_2) = b_2; f(a_3) = b_3; f(a_4) = b_4$.
- For $f_0, f_1, f_2, f_3 \in F$ define function $h_{f_0,f_1,f_2,f_3}(s) = Lead(f_0 + f_1 s + f_2 s^2 + f_3 s^3)$, where $Lead(.)$ denotes the leading bit of input.
- $\mathcal{H} = \{h_{f_0,f_1,f_2,f_3} | f_0, f_1, f_2, f_3 \in F\}$.
- Claim: $\mathcal{H}$ is a 4-wise independent hash function family.

Proof sketch

- For the proof, assume that the elements of F are represented as ± 1 strings.
- Claim For any fixed $s, t, u, v \in F$ and $\alpha, \beta, \gamma, \delta \in \{+1, -1\}$,

$$\mathbf{Pr}_{h \leftarrow \mathcal{H}}[h(s) = \alpha, h(t) = \beta, h(u) = \gamma, h(v) = \delta] = \frac{1}{16}.$$

End