COL202: Discrete Mathematical Structures

Ragesh Jaiswal, CSE, IIT Delhi

Solve the recurrence relation $a_k = 3a_{k-1}$ for $k = 1, 2, ...$ and initial condition $a_0 = 2$.

- Let $G(x)$ be the generating function for the sequence $\{a_k\}$.
- Claim 1: $xG(x) = \sum_{k=1}^{\infty} a_{k-1}x^k$.
- Claim 2: $G(x) - 3xG(x) = a_0$.
- Claim 3: $G(x) = \sum_{k=0}^{\infty} 2 \cdot 3^k \cdot x^k$.

Data Structures: Universal Hashing

- How do we design a good hash function?
- A set S of keys from a universe $U = \{0, 1, ..., m-1\}$ is supposed to be stored in a table of size n with indices $T = \{0, 1, ..., n-1\}$.
  - Assume collisions are resolved using auxiliary data structure.
- What we need is a hash function $h : U \rightarrow T$ with the following main requirements:
  1. The hash function should minimize the number of collisions.
  2. The space used should be proportional to the number of keys stored. (i.e., $n \approx |S|$)

- How do we design a good hash function?
- A set S of keys from a universe $U = \{0, 1, ..., m-1\}$ is supposed to be stored in a table of size n with indices $T = \{0, 1, ..., n-1\}$.
  - Assume collisions are resolved using auxiliary data structure.
- What we need is a hash function $h : U \rightarrow T$ with the following main requirements:
  1. The hash function should minimize the number of collisions.
  2. The space used should be proportional to the number of keys stored. (i.e., $n \approx |S|$)
- Claim 1: If $m > n$, then for any h there exists a key set S such that h has collision w.r.t. S (i.e., $\exists x, y \in S, h(x) = h(y)$)

- How do we design a good hash function?
- A set S of keys from a universe $U = \{0, 1, ..., m-1\}$ is supposed to be stored in a table of size n with indices $T = \{0, 1, ..., n-1\}$.
  - Assume collisions are resolved using auxiliary data structure.
- What we need is a hash function $h : U \rightarrow T$ with the following main requirements:
  1. The hash function should minimize the number of collisions.
  2. The space used should be proportional to the number of keys stored. (i.e., $n \approx |S|$)
- Claim 1: If $m > n$, then for any h there exists a key set S such that h has collision w.r.t. S (i.e., $\exists x, y \in S, h(x) = h(y)$)
  - Claim 1.1: Any fixed hash function $h : U \rightarrow T$, must map at least $\lceil \frac{m}{n} \rceil$ elements of U to some index in the set T.

- How do we design a good hash function?
- A set S of keys from a universe $U = \{0, 1, ..., m-1\}$ is supposed to be stored in a table of size n with indices $T = \{0, 1, ..., n-1\}$.
  - Assume collisions are resolved using auxiliary data structure.
- What we need is a hash function $h : U \rightarrow T$ with the following main requirements:
  1. The hash function should minimize the number of collisions.
  2. The space used should be proportional to the number of keys stored. (i.e., $n \approx |S|$)
- Claim 1: If $m > n$, then for any h there exists a key set S such that h has collision w.r.t. S (i.e., $\exists x, y \in S, h(x) = h(y)$)
- Claim 2: For any fixed key set S such that $|S| \leq n$, there exists a hash function such that h has no collisions w.r.t. S.

- How do we design a good hash function?
- A set S of keys from a universe $U = \{0, 1, ..., m-1\}$ is supposed to be stored in a table of size n with indices $T = \{0, 1, ..., n-1\}$.
  - Collisions are resolved using auxiliary data structure.
- What we need is a hash function $h : U \rightarrow T$ with the following main requirements:
  1. The hash function should minimize the number of collisions.
  2. The space used should be proportional to the number of keys stored. (i.e., $n \approx |S|$)
- Claim 1: If $m > n$, then for any h there exists a key set S such that h has collision w.r.t. S (i.e., $\exists x, y \in S, h(x) = h(y)$)
- Claim 2: For any fixed key set S such that $|S| \leq n$, there exists a hash function such that h has no collisions w.r.t. S.
- The issue is that the key set S is not known a-priori. That is, before using the data structure.
- Question: How do we solve this problem then?

- How do we design a good hash function?
- A set S of keys from a universe $U = \{0, 1, ..., m-1\}$ is supposed to be stored in a table of size n with indices $T = \{0, 1, ..., n-1\}$.
  - Collisions are resolved using auxiliary data structure.
- What we need is a hash function $h : U \to T$ with the following main requirements:
  1. The hash function should minimize the number of collisions.
  2. The space used should be proportional to the number of keys stored. (i.e., $n \approx |S|$)
- Claim 1: If $m > n$, then for any h there exists a key set S such that h has collision w.r.t. S (i.e., $\exists x, y \in S, h(x) = h(y)$)
- Claim 2: For any fixed key set S such that $|S| \leq n$, there exists a hash function such that h has no collisions w.r.t. S.
- The issue is that the key set S is not known a-priori. That is, before using the data structure.
- Question: How do we solve this problem then?
  - Randomly select a hash function from a family H of hash functions.

- How do we design a good hash function?
- A set S of keys from a universe $U = \{0, 1, ..., m-1\}$ is supposed to be stored in a table of size n with indices $T = \{0, 1, ..., n-1\}$.
  - Collisions are resolved using auxiliary data structure.
- What we need is a hash function $h : U \rightarrow T$ with the following main requirements:
  1. The hash function should minimize the number of collisions.
  2. The space used should be proportional to the number of keys stored. (i.e., $n \approx |S|$)
- The issue is that the key set S is not known a-priori. That is, before using the data structure.
- Question: How do we solve this problem then?
  - Randomly select a hash function from a family H of hash functions.

Definition (2-universality)

A hash function family H is said to be 2-universal iff:

$$\forall x, y \in U, x \neq y, \mathbf{Pr}_{h \leftarrow H}[h(x) = h(y)] \leq \frac{1}{n}.$$

Definition (2-universality)

A hash function family H is said to be 2-universal iff:

$$\forall x, y \in U, x \neq y, \mathbf{Pr}_{h \leftarrow H}[h(x) = h(y)] \leq \frac{1}{n}.$$

- Theorem: Consider hashing using a 2-universal hash function family. Consider t insert operations, the expected cost of each operation is at most $(1 + t/n)$.

Definition (2-universality)

A hash function family H is said to be 2-universal iff:

$$\forall x, y \in U, x \neq y, \mathbf{Pr}_{h \leftarrow H}[h(x) = h(y)] \leq \frac{1}{n}.$$

- Theorem: Consider hashing using a 2-universal hash function family. Consider t insert operations, the expected cost of each operation is at most $(1 + t/n)$.
  - Proof sketch: Consider any key x. The expected number of keys in location $h(x)$ is at most t/n.
- Question: Can you think of a 2-universal hash function family?

Definition (2-universality)

A hash function family H is said to be 2-universal iff:

$$\forall x, y \in U, x \neq y, \mathbf{Pr}_{h \leftarrow H}[h(x) = h(y)] \leq \frac{1}{n}.$$

- Theorem: Consider hashing using a 2-universal hash function family. Consider t insert operations, the expected cost of each operation is at most $(1 + t/n)$.
  - Proof sketch: Consider any key x. The expected number of keys in location $h(x)$ is at most t/n.
- Question: Can you think of a 2-universal hash function family?
  - Simple answer: The set of all functions from U to T.
  - Do you see any issues with using this hash function family?

Definition (2-universality)

A hash function family H is said to be 2-universal iff:

$$\forall x, y \in U, x \neq y, \mathbf{Pr}_{h \leftarrow H}[h(x) = h(y)] \leq \frac{1}{n}.$$

- Theorem: Consider hashing using a 2-universal hash function family. Consider t insert operations, the expected cost of each operation is at most $(1 + t/n)$.
  - Proof sketch: Consider any key x. The expected number of keys in location $h(x)$ is at most t/n.
- Question: Can you think of a 2-universal hash function family?
  - Simple answer: The set of all functions from U to T.
  - Do you see any issues with using this hash function family? The description of any hash function from this family is large.
  - Question: Can we design a more compact hash function family?

Definition (2-universality)

A hash function family H is said to be 2-universal iff:

$$\forall x, y \in U, x \neq y, \mathbf{Pr}_{h \leftarrow H}[h(x) = h(y)] \leq \frac{1}{n}.$$

- Theorem: Consider hashing using a 2-universal hash function family. Consider t insert operations, the expected cost of each operation is at most $(1 + t/n)$.
- A compact 2-universal hash function family:
  - Let $m \leq p \leq 2m$.
  - $H = \{h_{a,b} | a \in \{1, ..., p-1\}, b \in \{0, ..., p-1\}\}$ and $h_{a,b}(x) = ((ax + b) \mod p) \mod n$.
  - How many functions does H have?

Definition (2-universality)

A hash function family H is said to be 2-universal iff:

$$\forall x, y \in U, x \neq y, \mathbf{Pr}_{h \leftarrow H}[h(x) = h(y)] \leq \frac{1}{n}.$$

- Theorem: Consider hashing using a 2-universal hash function family. Consider t insert operations, the expected cost of each operation is at most $(1 + t/n)$.
- A compact 2-universal hash function family:
  - Let $m \leq p \leq 2m$.
  - $H = \{h_{a,b} | a \in \{1, ..., p-1\}, b \in \{0, ..., p-1\}\}$ and $h_{a,b}(x) = ((ax + b) \mod p) \mod n$.
  - How many functions does H have? $p(p-1)$

Definition (2-universality)

A hash function family H is said to be 2-universal iff:

$$\forall x, y \in U, x \neq y, \mathbf{Pr}_{h \leftarrow H}[h(x) = h(y)] \leq \frac{1}{n}.$$

- Theorem: Consider hashing using a 2-universal hash function family. Consider t insert operations, the expected cost of each operation is at most $(1 + t/n)$.
- A compact 2-universal hash function family:
  - Let $m \leq p \leq 2m$.
  - $H = \{h_{a,b} | a \in \{1, ..., p-1\}, b \in \{0, ..., p-1\}\}$ and $h_{a,b}(x) = ((ax + b) \mod p) \mod n$.
  - How many functions does H have? $p(p-1)$
  - Theorem: H is 2-universal.

Definition (2-universality)

A hash function family H is said to be 2-universal iff:

$$\forall x, y \in U, x \neq y, \mathbf{Pr}_{h \leftarrow H}[h(x) = h(y)] \leq \frac{1}{n}.$$

- Theorem: Consider hashing using a 2-universal hash function family. Consider t insert operations, the expected cost of each operation is at most $(1 + t/n)$.
- A compact 2-universal hash function family:
  - Let $m \leq p \leq 2m$.
  - $H = \{h_{a,b} | a \in \{1, ..., p-1\}, b \in \{0, ..., p-1\}\}$ and $h_{a,b}(x) = ((ax + b) \mod p) \mod n$.
  - Theorem: H is 2-universal.

- Theorem: H is 2-universal.

Proof sketch

- Let $g_{a,b}(x) = (ax + b) \mod p$. So, $h_{a,b}(x) = g_{a,b}(x) \mod n$.
- Consider any $x, y \in \{0, ..., p-1\}$ such that $x \neq y$.
- Claim 1: If $h_{a,b}(x) = h_{a,b}(y)$, then $g_{a,b}(x) = g_{a,b}(y) \mod n$.

- Theorem: H is 2-universal.

Proof sketch

- Let $g_{a,b}(x) = (ax + b) \mod p$. So, $h_{a,b}(x) = g_{a,b}(x) \mod n$.
- Consider any $x, y \in \{0, ..., p-1\}$ such that $x \neq y$.
- Claim 1: If $h_{a,b}(x) = h_{a,b}(y)$, then $g_{a,b}(x) = g_{a,b}(y) \mod n$.
- Claim 2: For all $\alpha, \beta \in \{0, ..., p-1\}$:

$$\mathbf{Pr}[g_{a,b}(x) = \alpha \text{ and } g_{a,b}(y) = \beta] = \begin{cases} 0 & \text{if } \alpha = \beta \\ \frac{1}{p(p-1)} & \text{otherwise} \end{cases}$$

- Theorem: H is 2-universal.

Proof sketch

- Let $g_{a,b}(x) = (ax + b) \mod p$. So, $h_{a,b}(x) = g_{a,b}(x) \mod n$.
- Consider any $x, y \in \{0, ..., p-1\}$ such that $x \neq y$.
- Claim 1: If $h_{a,b}(x) = h_{a,b}(y)$, then $g_{a,b}(x) = g_{a,b}(y) \mod n$.
- Claim 2: For all $\alpha, \beta \in \{0, ..., p-1\}$:

$$\mathbf{Pr}[g_{a,b}(x) = \alpha \text{ and } g_{a,b}(y) = \beta] = \begin{cases} 0 & \text{if } \alpha = \beta \\ \frac{1}{p(p-1)} & \text{otherwise} \end{cases}$$

- Claim 3: We have:

$$\mathbf{Pr}[h_{a,b}(x) = h_{a,b}(y)] = \frac{|\{(\alpha, \beta) : \alpha \neq \beta \text{ and } \alpha \equiv \beta \mod n\}|}{p(p-1)} \leq \frac{1}{n}.$$

End