COL106: Data Structures and Algorithms

Ragesh Jaiswal, IIT Delhi

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 $h : U \to T$ 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., $\exists x, y \in S, h(x) = h(y)$)

- 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 $h : U \rightarrow T$ 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?
  - 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?

End