CS 577 - Randomized Algorithms
Roots & Multiplication & Hashing

Manolis Vlatakis

Department of Computer Sciences
University of Wisconsin – Madison

Fall 2024

Freivald's Algorithm

The Problem

- Given three matrices A, B, and C of dimensions $n \times n$.
- We want to verify whether $C = A \times B$.
- **Naïve Method:** Compute $A \times B$ and compare with C.
- **Time Complexity:** $O(n^3)$.

The Problem

- Given three matrices A, B, and C of dimensions $n \times n$.
- We want to verify whether $C = A \times B$.
- **Naïve Method:** Compute $A \times B$ and compare with C.
- **Time Complexity:** $O(n^3)$.

🤔Do we know better method?

The Problem

- Given three matrices A, B, and C of dimensions $n \times n$.
- We want to verify whether $C = A \times B$.
- **Naïve Method:** Compute $A \times B$ and compare with C.
- **Time Complexity:** $O(n^3)$.

🤔Do we know better method?

- Yeap!: Divide & Conquer: Strassen's Algorithm $O(n^{2.81})$.
- Best Known Algorithm: Vassilevska (2015): $O(n^{2.373})$

The Problem

- Given three matrices A, B, and C of dimensions $n \times n$.
- We want to verify whether $C = A \times B$.
- **Naïve Method:** Compute $A \times B$ and compare with C.
- **Time Complexity:** $O(n^3)$.

🤔Do we know better method?

- Yeap!: Divide & Conquer: Strassen's Algorithm $O(n^{2.81})$.
- Best Known Algorithm: Vassilevska (2015): $O(n^{2.373})$

🤔Can we randomly check in $O(n^2)$?

Freivald's Algorithm

Freivald's Algorithm for Matrix Multiplication Verification

1. **Input:** Matrices A, B, C of dimensions $n \times n$.
2. **For** k iterations:
   - Choose a random vector $r \in \{0,1\}^n$.
   - Compute $s = C \times r$.
   - Compute $q = B \times r$.
   - Compute $t = A \times q$.
   - **If** $s \neq t$:
     - **Return** `FALSE`.
3. **Return** `TRUE`.

Freivald's Algorithm

Freivald's Algorithm for Matrix Multiplication Verification

1. **Input:** Matrices A, B, C of dimensions $n \times n$.
2. **For** k iterations:
    - Choose a random vector $r \in \{0, 1\}^n$.
    - Compute $s = C \times r$.
    - Compute $q = B \times r$.
    - Compute $t = A \times q$.
    - **If** $s \neq t$:
        - **Return** `FALSE`.
3. **Return** `TRUE`.

🤔What is the time complexity of Freivald's Algorithm?

Freivald's Algorithm

Freivald's Algorithm for Matrix Multiplication Verification

1. **Input:** Matrices A, B, C of dimensions $n \times n$.
2. **For** k iterations:
    - Compute a random vector $r \in \{0,1\}^n$. $s = C \times r$, $q = B \times r$, $t = A \times q$.
    - Check **if** $s \stackrel{?}{=} t$:

🤔What is the time complexity of Freivald's Algorithm?

Time Complexity

- Each iteration requires:
    - Matrix-vector multiplication: $O(n^2)$.
- Total time complexity for k iterations: $O(kn^2)$.
- For constant $k = \Theta(1)$, the complexity is $O(n^2)$.

Correctness of the Algorithm

- **If** $C = A \times B$:
  - The algorithm always returns `TRUE`.
- **If** $C \neq A \times B$:
  - There exists at least one r such that $A(Br) \neq Cr$.
  - The probability that the algorithm fails to detect the error is at most $\frac{1}{2}$ per iteration. (Why???)

Probabilistic Analysis

- Let $D = AB - C$ and assume $D \neq 0$.
- We want to find the probability $\Pr(Dr = 0)$.
- **Observation:** Since $D \neq 0$, there is at least one non-zero row or column.

Theorem 1

The one-sided error probability is $\leq \frac{1}{2}$

Correctness of the Algorithm

What does it mean <u>one-sided error</u>?
What kind of casino-algorithm do we have here?

is at most $\frac{1}{2}$ per iteration. (Why???)

Probabilistic Analysis

- Let $D = AB - C$ and assume $D \neq 0$.
- We want to find the probability $\Pr(Dr = 0)$.
- **Observation:** Since $D \neq 0$, there is at least one non-zero row or column.

Theorem 1

The one-sided error probability is $\leq \frac{1}{2}$

Correctness of the Algorithm

What does it mean one-sided error?
What kind of casino-algorithm do we have here?

We try to solve a decision problem.

One-sided error :

$$\begin{cases} \text{If the answer is Yes, we never fail} \\ \text{If the answer is No, there exist a failure probability} \end{cases}$$

This is a Monte-Carlo algorithm.

is

- **Observation:** Since $D \neq 0$, there is at least one non-zero row or column.

Theorem 1

The one-sided error probability is $\leq \frac{1}{2}$

Proof of the Theorem (Detailed)

- Let $D = AB - C$, and suppose $D \neq 0$.
Note: We do not compute the matrix D in the algorithm; we need it for the analysis!

$$D = \begin{bmatrix} -d_1- \\ -d_2- \\ \vdots \\ -d_n- \end{bmatrix}$$

- Since $D \neq 0$, there exists at least one non-zero row vector d in D.
- We need to compute the probability that $Dr = 0$, where $r \in \{0, 1\}^n$ is a random vector.

Proof of the Theorem (Detailed)

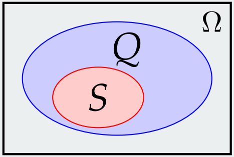
- Focus on the non-zero row d:

$$Dr = 0 \implies d \cdot r = 0 \implies \Pr[Dr = 0] \leq \Pr[d \cdot r = 0]$$

Illustration of $S \subseteq Q$ Implies $\Pr[S] \leq \Pr[Q]$

- If $S \subseteq Q$, then the event S is entirely contained within Q.
- Therefore, the probability of S cannot exceed the probability of Q:

$$\Pr[S] \leq \Pr[Q]$$



Proof of the Theorem (Detailed)

- Thus, we need to compute the probability that $dr = 0$, where $r \in \{0,1\}^n$ is a random vector.
- The dot product is given by:

$$d \cdot r = d_1 r_1 + \cdots + d_n r_n = \sum_{i=1}^{n} d_i r_i$$

- Since $d \neq 0$, $\exists$ at least one index j^* such that $d_{j^*} \neq 0$.

PROOF OF THE THEOREM (DETAILED)

- Thus, we need to compute the probability that $dr = 0$, where $r \in \{0, 1\}^n$ is a random vector.
- The dot product is given by:

$$d \cdot r = d_1 r_1 + \cdots + d_n r_n = \sum_{i=1}^{n} d_i r_i$$

- Since $d \neq 0$, $\exists$ at least one index j^* such that $d_{j^*} \neq 0$.

🤔Why do we care about that specific r_{j^*}?

Proof of the Theorem (Detailed)

- Thus, we need to compute the probability that $dr = 0$, where $r \in \{0,1\}^n$ is a random vector.
- The dot product is given by:

$$d \cdot r = d_1 r_1 + \cdots + d_n r_n = \sum_{i=1}^{n} d_i r_i$$

- Since $d \neq 0$, $\exists$ at least one index j^* such that $d_{j^*} \neq 0$.

Although unlikely, the matrix D could be such that all rows are zero except one. For example:

$D = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ In this case, only the term $d_1 r_1$ survives in $d \cdot r$.

Proof of the Theorem (Detailed)

- Thus, we need to compute the probability that $dr = 0$, where $r \in \{0,1\}^n$ is a random vector.
- The dot product is given by:

$$d \cdot r = d_1 r_1 + \cdots + d_n r_n = \sum_{i=1}^{n} d_i r_i$$

- Since $d \neq 0$, $\exists$ at least one index j^* such that $d_{j^*} \neq 0$.

Examples

Suppose the first row of D is $d = (4, -5, 1, 2)$ and $r = (1, 1, 1, 0)$. What is $d \cdot r$?

Proof of the Theorem (Detailed)

- Thus, we need to compute the probability that $dr = 0$, where $r \in \{0, 1\}^n$ is a random vector.
- The dot product is given by:

$$d \cdot r = d_1 r_1 + \cdots + d_n r_n = \sum_{i=1}^{n} d_i r_i$$

- Since $d \neq 0$, $\exists$ at least one index j^* such that $d_{j^*} \neq 0$.

Examples

Suppose the first row of D is $d = (4, -5, 1, 2)$ and $r = (1, 1, 1, 0)$.
What is $d \cdot r$?
Answer: $d \cdot r = 4(1) + (-5)(1) + 1(1) + 2(0) = 0$

Proof of the Theorem (Detailed)

- Thus, we need to compute the probability that $dr = 0$, where $r \in \{0,1\}^n$ is a random vector.
- The dot product is given by:

$$d \cdot r = d_1 r_1 + \cdots + d_n r_n = \sum_{i=1}^{n} d_i r_i$$

- Since $d \neq 0$, $\exists$ at least one index j^* such that $d_{j^*} \neq 0$.

Examples

Suppose the first row of D is $d = (4,5,2,1)$ and $r = (0,1,1,0)$. What is $d \cdot r$?

Proof of the Theorem (Detailed)

- Thus, we need to compute the probability that $dr = 0$, where $r \in \{0,1\}^n$ is a random vector.
- The dot product is given by:

$$d \cdot r = d_1 r_1 + \cdots + d_n r_n = \sum_{i=1}^{n} d_i r_i$$

- Since $d \neq 0$, $\exists$ at least one index j^* such that $d_{j^*} \neq 0$.

Examples

Suppose the first row of D is $d = (4, 5, 2, 1)$ and $r = (0, 1, 1, 0)$.
What is $d \cdot r$?
Answer: $d \cdot r = 4(0) + 5(1) + 2(1) + 1(0) = 7$

Proof of the Theorem (Detailed)

- Thus, we need to compute the probability that $dr = 0$, where $r \in \{0,1\}^n$ is a random vector.
- The dot product is given by:

$$d \cdot r = d_1 r_1 + \cdots + d_n r_n = \sum_{i=1}^{n} d_i r_i$$

- Since $d \neq 0$, $\exists$ at least one index j^* such that $d_{j^*} \neq 0$.

Examples

Suppose the first row of D is $d = (7,0,0,0)$ and $r = (1,1,1,0)$. What is $d \cdot r$?

Proof of the Theorem (Detailed)

- Thus, we need to compute the probability that $dr = 0$, where $r \in \{0, 1\}^n$ is a random vector.
- The dot product is given by:

$$d \cdot r = d_1 r_1 + \cdots + d_n r_n = \sum_{i=1}^{n} d_i r_i$$

- Since $d \neq 0$, $\exists$ at least one index j^* such that $d_{j^*} \neq 0$.

Examples

Suppose the first row of D is $d = (7, 0, 0, 0)$ and $r = (1, 1, 1, 0)$.
What is $d \cdot r$?
Answer: $d \cdot r = 7 \cdot 1 = 7$

PROOF OF THE THEOREM (DETAILED)

- Thus, we need to compute the probability that $dr = 0$, where $r \in \{0,1\}^n$ is a random vector.
- The dot product is given by:

$$d \cdot r = d_1 r_1 + \cdots + d_n r_n = \sum_{i=1}^{n} d_i r_i$$

- Since $d \neq 0$, $\exists$ at least one index j^* such that $d_{j^*} \neq 0$.

Examples

Suppose the first row of D is $d = (7, 0, 0, 0)$ and $r = (0, 1, 1, 0)$. What is $d \cdot r$?

Proof of the Theorem (Detailed)

- Thus, we need to compute the probability that $dr = 0$, where $r \in \{0,1\}^n$ is a random vector.
- The dot product is given by:

$$d \cdot r = d_1 r_1 + \cdots + d_n r_n = \sum_{i=1}^{n} d_i r_i$$

- Since $d \neq 0$, $\exists$ at least one index j^* such that $d_{j^*} \neq 0$.

Examples

Suppose the first row of D is $d = (7,0,0,0)$ and $r = (0,1,1,0)$.
What is $d \cdot r$?
Answer: $d \cdot r = 7 \cdot 0 = 0$

Proof of the Theorem (Detailed)

- Thus, we need to compute the probability that $dr = 0$, where $r \in \{0, 1\}^n$ is a random vector.
- The dot product is given by:

$$d \cdot r = d_1 r_1 + \cdots + d_n r_n = \sum_{i=1}^{n} d_i r_i$$

- Since $d \neq 0$, $\exists$ at least one index j^* such that $d_{j^*} \neq 0$.

Examples

If $d = (12, 9, 2, 5)$ and $r = (r_1, r_2, r_3, r_4)$ with $r_i \in \{0, 1\}$, what is $\Pr[d \cdot r = 0]$?

Proof of the Theorem (Detailed)

- Thus, we need to compute the probability that $dr = 0$, where $r \in \{0, 1\}^n$ is a random vector.
- The dot product is given by:

$$d \cdot r = d_1 r_1 + \cdots + d_n r_n = \sum_{i=1}^{n} d_i r_i$$

- Since $d \neq 0$, $\exists$ at least one index j^* such that $d_{j^*} \neq 0$.

Examples

If $d = (12, 9, 2, 5)$ and $r = (r_1, r_2, r_3, r_4)$ with $r_i \in \{0, 1\}$, what is $\Pr[d \cdot r = 0]$?
Answer: Since each r_i is independent,

$$\Pr[d \cdot r = 0] = 1/2^4$$

PROOF OF THE THEOREM (DETAILED)

- Thus, we need to compute the probability that $dr = 0$, where $r \in \{0, 1\}^n$ is a random vector.
- The dot product is given by:

$$d \cdot r = d_1 r_1 + \cdots + d_n r_n = \sum_{i=1}^{n} d_i r_i$$

- Since $d \neq 0$, $\exists$ at least one index j^* such that $d_{j^*} \neq 0$.

Examples

If $d = (12, 9, 2, 5)$ and $r = (r_1, r_2, r_3, r_4)$ with $r_i \in \{0, 1\}$, what is $\Pr[d \cdot r = 0]$?
Answer: Since each r_i is independent,

$$\Pr[d \cdot r = 0] = 1/2^4$$

Proof of the Theorem (Detailed)

- Thus, we need to compute the probability that $dr = 0$, where $r \in \{0,1\}^n$ is a random vector.
- The dot product is given by:

$$d \cdot r = d_1 r_1 + \cdots + d_n r_n = \sum_{i=1}^{n} d_i r_i$$

- Since $d \neq 0$, $\exists$ at least one index j^* such that $d_{j^*} \neq 0$.

Examples

If $d = (12, 9, 2, 5)$ and $r = (r_1, r_2, r_3, r_4)$ with $r_i \in \{0,1\}$, what is $\Pr[d \cdot r = 0]$?
Answer: Since each r_i is independent,

$$\Pr[d \cdot r = 0] = 1/2^4$$

Proof of the Theorem (Detailed)

- Thus, we need to compute the probability that $dr = 0$, where $r \in \{0, 1\}^n$ is a random vector.
- The dot product is given by:

$$d \cdot r = d_1 r_1 + \cdots + d_n r_n = \sum_{i=1}^{n} d_i r_i$$

- Since $d \neq 0$, $\exists$ at least one index j^* such that $d_{j^*} \neq 0$.

Question

Does it matter which row of D is non-zero?

Proof of the Theorem (Detailed)

- Thus, we need to compute the probability that $dr = 0$, where $r \in \{0,1\}^n$ is a random vector.
- The dot product is given by:

$$d \cdot r = d_1 r_1 + \cdots + d_n r_n = \sum_{i=1}^{n} d_i r_i$$

- Since $d \neq 0$, $\exists$ at least one index j^* such that $d_{j^*} \neq 0$.

Question

Does it matter which row of D is non-zero?

Answer

No, as long as $AB \neq C$, then $\exists$ at least one non-zero row in D. We are interested in whether $d \cdot r = 0$ happens by chance due to the random choices of r ☺

Proof of the Theorem (Detailed)

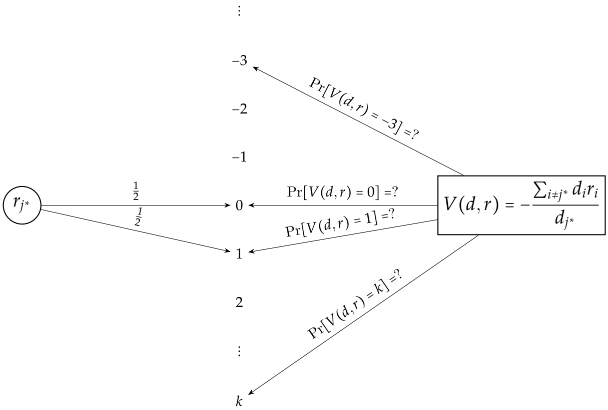
- **Case Analysis of Event** $\{d \cdot r = 0\}$**:**
  - Since $d_{j^*} \neq 0$, r_{j^*} affects $d \cdot r$.
  - We can write:

$$d \cdot r = d_{j^*} r_{j^*} + \sum_{i \neq j^*} d_i r_i = 0 \implies r_{j^*} = -\frac{\sum_{i \neq j^*} d_i r_i}{d_{j^*}}$$

  - All r_i are uniformly random in $\{0, 1\}$.

VISUALIZATION OF r_{j^*} AND $V(d,r)$

- We illustrate the relationship between r_{j^*} and $V(d,r) = -\frac{\sum_{i \neq j^*} d_i r_i}{d_{j^*}}$.



Proof of the Theorem (Finished)

- Let $D = AB - C$, and suppose $D \neq 0$.

$$\text{One-sided error: } D \neq 0 \text{ but } Dr = 0$$

$$\Downarrow$$

Proof of the Theorem (Finished)

- Let $D = AB - C$, and suppose $D \neq 0$.

One-sided error: $D \neq 0$ but $Dr = 0$

$$\Downarrow$$

- We need to compute the probability that $Dr = 0$, where $r \in \{0, 1\}^n$ is a random vector.
- Since $D \neq 0$, there exists at least one non-zero row vector d in D.

$$\Downarrow$$

Proof of the Theorem (Finished)

- We need to compute the probability that $Dr = 0$, where $r \in \{0,1\}^n$ is a random vector.
- Since $D \neq 0$, there exists at least one non-zero row vector d in D.

$$\Downarrow$$

- We can focus on the non-zero row d:

$$Dr = 0 \implies \exists d : d \cdot r = 0 \implies \Pr[Dr = 0] \leq \Pr[d \cdot r = 0]$$

$$\Downarrow$$

Proof of the Theorem (Finished)

- We can focus on the non-zero row d:

$$Dr = 0 \implies \exists d : d \cdot r = 0 \implies \Pr[Dr = 0] \leq \Pr[d \cdot r = 0]$$

$$\Downarrow$$

- Since r_{j^*} is independent of the other r_i and takes values 0 or 1 with equal probability, the probability that $d \cdot r = 0$ is at most $\frac{1}{2}$:

$$\Pr(d \cdot r = 0) \leq \frac{1}{2}$$

Proof of the Theorem (Finished)

- Since r_{j^*} is independent of the other r_i and takes values 0 or 1 with equal probability, the probability that $d \cdot r = 0$ is at most $\frac{1}{2}$:

$$\Pr(d \cdot r = 0) \leq \frac{1}{2}$$

🤔Can we improve the probability?

Reducing the Error Probability

- By running the algorithm k times:

$$\Pr(\text{Failure}) \leq \left(\frac{1}{2}\right)^k$$

- Choosing an appropriate k, the error probability becomes negligible.

Reducing the Error Probability

- By running the algorithm k times:

$$\Pr(\text{Failure}) \le \left(\frac{1}{2}\right)^k$$

- Choosing an appropriate k, the error probability becomes negligible.

🤔Homework: Assume that you have two super-large polynomials $d = 10^{10}$ in the forms:

$$\begin{cases} P(x) = a_d x^d + a_{d-1} + \cdots + a_1 x + a_0 \\ Q(x) = (x - r_1)\ldots(x - r_d) \end{cases}$$

- Give me a deterministic algorithm for verifying $P(x) = Q(x)$ with minimal evaluations of P, Q.

Reducing the Error Probability

- By running the algorithm k times:

$$\Pr(\text{Failure}) \leq \left(\frac{1}{2}\right)^k$$

- Choosing an appropriate k, the error probability becomes negligible.

Homework: Assume that you have two super-large polynomials $d = 10^{10}$ in the forms:

$$\begin{cases} P(x) = a_d x^d + a_{d-1} + \cdots + a_1 x + a_0 \\ Q(x) = (x - r_1) \ldots (x - r_d) \end{cases}$$

- Give me a deterministic algorithm for verifying $P(x) = Q(x)$ with minimal evaluations of P, Q.
- Give me a super simple randomized algorithm ☺!!!

The Success Story of Python Dictionaries

Python Dictionaries: Items as (key, value) pairs

Python dictionaries: items are (key, value) pairs
Example:

```
d = {'algorithms': 5, 'cool': 42}
```

Operations:

```
d.items()     # [('algorithms', 5), ('cool', 42)]
d['cool']     # 42
d[42]         # KeyError
'cool' in d   # True
42 in d       # False
```

Note:

```
# Python set is really dict where items are keys
```

Dictionary Data Structure

Dictionary

Given a universe U of possible elements, maintain a subset $S \subseteq U$ so that inserting, deleting, and searching in S is efficient.

Dictionary Data Structure

Dictionary

Given a universe U of possible elements, maintain a subset $S \subseteq U$ so that inserting, deleting, and searching in S is efficient.

Dictionary Operations

DICTIONARY DATA STRUCTURE

Dictionary

Given a universe U of possible elements, maintain a subset $S \subseteq U$ so that inserting, deleting, and searching in S is efficient.

Dictionary Operations

- CREATE: Initializes a fresh dictionary that can maintain a subset S of U that is initially empty.

Dictionary Data Structure

Dictionary

Given a universe U of possible elements, maintain a subset $S \subseteq U$ so that inserting, deleting, and searching in S is efficient.

Dictionary Operations

- Create: Initializes a fresh dictionary that can maintain a subset S of U that is initially empty.
- Insert(u): Adds $u \in U$ to the dictionary (S).

Dictionary Data Structure

Dictionary

Given a universe U of possible elements, maintain a subset $S \subseteq U$ so that inserting, deleting, and searching in S is efficient.

Dictionary Operations

- Create: Initializes a fresh dictionary that can maintain a subset S of U that is initially empty.
- Insert(u): Adds $u \in U$ to the dictionary (S).
- Delete(u): Remove u from S.

Dictionary Data Structure

Dictionary

Given a universe U of possible elements, maintain a subset $S \subseteq U$ so that inserting, deleting, and searching in S is efficient.

Dictionary Operations

- Create: Initializes a fresh dictionary that can maintain a subset S of U that is initially empty.
- Insert(u): Adds $u \in U$ to the dictionary (S).
- Delete(u): Remove u from S.
- Lookup(u): Determine if u is in S; if so retrieve u.

Dictionary Data Structure

Dictionary Interface.

- Create(): initialize a dictionary with $S = \emptyset$.
- Insert(u): add element $u \in U$ to S.
- Delete(u): delete u from S (if u is currently in S).
- Lookup(u): is u in S?

Challenge. Universe U can be extremely large, so defining an array of size $|U|$ is infeasible.

Dictionary Data Structure

Dictionary Interface.

- Create(): initialize a dictionary with $S = \emptyset$.
- Insert(u): add element $u \in U$ to S.
- Delete(u): delete u from S (if u is currently in S).
- Lookup(u): is u in S?

Challenge. Universe U can be extremely large, so defining an array of size $|U|$ is infeasible.

Applications.

DICTIONARY DATA STRUCTURE

DICTIONARY INTERFACE.

- CREATE(): INITIALIZE A DICTIONARY WITH $S = \varnothing$.
- INSERT(U): ADD ELEMENT $u \in U$ TO S.
- DELETE(U): DELETE u FROM S (IF u IS CURRENTLY IN S).
- LOOKUP(U): IS u IN S?

Challenge. Universe U can be extremely large, so defining an array of size $|U|$ is infeasible.

Applications. File systems,

Dictionary Data Structure

Dictionary Interface.

- Create(): initialize a dictionary with $S = \varnothing$.
- Insert(u): add element $u \in U$ to S.
- Delete(u): delete u from S (if u is currently in S).
- Lookup(u): is u in S?

Challenge. Universe U can be extremely large, so defining an array of size $|U|$ is infeasible.

Applications. File systems, databases,

Dictionary Data Structure

Dictionary Interface.

- Create(): initialize a dictionary with $S = \varnothing$.
- Insert(u): add element $u \in U$ to S.
- Delete(u): delete u from S (if u is currently in S).
- Lookup(u): is u in S?

Challenge. Universe U can be extremely large, so defining an array of size $|U|$ is infeasible.

Applications. File systems, databases, Google,

Dictionary Data Structure

Dictionary Interface.

- Create(): initialize a dictionary with $S = \varnothing$.
- Insert(u): add element $u \in U$ to S.
- Delete(u): delete u from S (if u is currently in S).
- Lookup(u): is u in S?

Challenge. Universe U can be extremely large, so defining an array of size $|U|$ is infeasible.

Applications. File systems, databases, Google, compilers,

DICTIONARY DATA STRUCTURE

DICTIONARY INTERFACE.

- CREATE(): INITIALIZE A DICTIONARY WITH $S = \emptyset$.
- INSERT(U): ADD ELEMENT $u \in U$ TO S.
- DELETE(U): DELETE u FROM S (IF u IS CURRENTLY IN S).
- LOOKUP(U): IS u IN S?

Challenge. Universe U can be extremely large, so defining an array of size $|U|$ is infeasible.

Applications. File systems, databases, Google, compilers, checksums,

Dictionary Data Structure

Dictionary Interface.

- Create(): initialize a dictionary with $S = \emptyset$.
- Insert(u): add element $u \in U$ to S.
- Delete(u): delete u from S (if u is currently in S).
- Lookup(u): is u in S?

Challenge. Universe U can be extremely large, so defining an array of size $|U|$ is infeasible.

Applications. File systems, databases, Google, compilers, checksums, P2P networks,

Dictionary Data Structure

Dictionary Interface.

- Create(): initialize a dictionary with $S = \varnothing$.
- Insert(u): add element $u \in U$ to S.
- Delete(u): delete u from S (if u is currently in S).
- Lookup(u): is u in S?

Challenge. Universe U can be extremely large, so defining an array of size $|U|$ is infeasible.

Applications. File systems, databases, Google, compilers, checksums, P2P networks, cryptography,

DICTIONARY DATA STRUCTURE

DICTIONARY INTERFACE.

- CREATE(): INITIALIZE A DICTIONARY WITH $S = \varnothing$.
- INSERT(U): ADD ELEMENT $u \in U$ TO S.
- DELETE(U): DELETE u FROM S (IF u IS CURRENTLY IN S).
- LOOKUP(U): IS u IN S?

Challenge. Universe U can be extremely large, so defining an array of size $|U|$ is infeasible.

Applications. File systems, databases, Google, compilers, checksums, P2P networks, cryptography, web caching, etc.

Dictionary Data Structure

Dictionary Interface.

- Create(): initialize a dictionary with $S = \varnothing$.
- Insert(u): add element $u \in U$ to S.
- Delete(u): delete u from S (if u is currently in S).
- Lookup(u): is u in S?

Challenge. Universe U can be extremely large, so defining an array of size $|U|$ is infeasible.

Applications. File systems, databases, Google, compilers, checksums, P2P networks, cryptography, web caching, etc.

🤔Can we implement Create-Insert-Delete-Lookup in $\Theta(1)$ on expectation?

The Solution of Day :

Hashing
A Randomized Data Type

Hashing

Definition

A function that converts some input value into a hash value.

- Input: A large universe of values U. Typically, assume $|U| \gg n$.
- Output: A hash value for $u \in U$ to $\{0, 1, 2, \ldots, n-1\}$.

Hashing

Definition

A function that converts some input value into a hash value.

- Input: A large universe of values U. Typically, assume $|U| \gg n$.
- Output: A hash value for $u \in U$ to $\{0, 1, 2, \ldots, n-1\}$.

Why?

Typically used to generate keys for a dictionary data structure.

Hashing

Motivation

- The values in U may be huge. Ex: Blog posts.

Hashing

Motivation

- The values in U may be huge. Ex: Blog posts.
- Take a value $u \in U$ and build a smaller key.

Hashing

Motivation

- The values in U may be huge. Ex: Blog posts.
- Take a value $u \in U$ and build a smaller key.

Hashing

- Hash Table: a n-length array H to store the values.
- Hash Function: Map $u \in U$ to an index in H; $h : U \to [0..n-1]$

Hashing

Motivation

- The values in U may be huge. Ex: Blog posts.
- Take a value $u \in U$ and build a smaller key.

Hashing

- Hash Table: a n-length array H to store the values.
- Hash Function: Map $u \in U$ to an index in H; $h : U \to [0..n-1]$

Dictionary Hashing

- 🤔Let $u, v \in U$. Say $|U| \gg n$, can $h(u) = h(v)$?

Hashing

Motivation

- The values in U may be huge. Ex: Blog posts.
- Take a value $u \in U$ and build a smaller key.

Hashing

- Hash Table: a n-length array H to store the values.
- Hash Function: Map $u \in U$ to an index in H; $h : U \to [0..n-1]$

Dictionary Hashing

- Let $u, v \in U$. Say $|U| \gg n$, can $h(u) = h(v)$? Yes.
- Collision: $h(u) = h(v)$ – At $H[i]$ is a linked-list (bucket) to store any values where $h(u) = i$.

Hashing

Motivation

- The values in U may be huge. Ex: Blog posts.
- Take a value $u \in U$ and build a smaller key.

Hashing

- Hash Table: a n-length array H to store the values.
- Hash Function: Map $u \in U$ to an index in H; $h : U \to [0..n-1]$

Dictionary Hashing

- Collision: $h(u) = h(v)$ – At $H[i]$ is a linked-list (bucket) to store any values where $h(u) = i$.
- 🤔Say $|S| = n$, what is the worst-case number of comparisons to $\textsc{Lookup}(u)$?

Hashing

Motivation

- The values in U may be huge. Ex: Blog posts.
- Take a value $u \in U$ and build a smaller key.

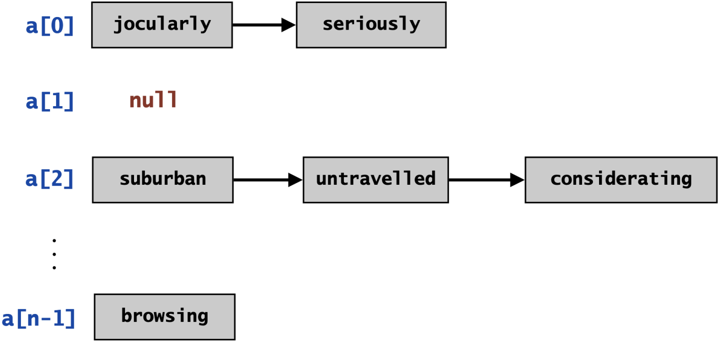
Hashing

- Hash Table: a n-length array H to store the values.
- Hash Function: Map $u \in U$ to an index in H; $h : U \rightarrow [0..n-1]$

Dictionary Hashing

- Collision: $h(u) = h(v)$ – At $H[i]$ is a linked-list (bucket) to store any values where $h(u) = i$.
- 🤔Say $|S| = n$, what is the worst-case number of comparisons to $\textsc{Lookup}(u)$? $O(n)$

Example of Bucketing



Birthday Paradox for Random hashing

How probable is to have the first collision after n elements

- **Problem:** In a group of n elements, what is the probability that at least two items map to the same bucket?

Birthday Paradox for Random hashing

How probable is to have the first collision after n elements

- **Problem:** In a group of n elements, what is the probability that at least two items map to the same bucket?
- **Restatement:** In a group of n people, what is the probability that at least two people share the same birthday?

BIRTHDAY PARADOX FOR RANDOM HASHING

HOW PROBABLE IS TO HAVE THE FIRST COLLISION AFTER n ELEMENTS

- **Problem:** In a group of n elements, what is the probability that at least two items map to the same bucket?
- **Restatement:** In a group of n people, what is the probability that at least two people share the same birthday?
- **Assumption:** Each bucket is equally likely.

Birthday Paradox for Random hashing

How probable is to have the first collision after n elements

- **Problem:** In a group of n elements, what is the probability that at least two items map to the same bucket?
- **Restatement:** In a group of n people, what is the probability that at least two people share the same birthday?
- **Assumption:** Each bucket is equally likely.
- **Restatement:** Each day of the year is equally likely for a birthday (ignoring leap years).

Birthday Paradox for Random hashing

How probable is to have the first collision after n elements

- **Problem:** In a group of n elements, what is the probability that at least two items map to the same bucket?
- **Restatement:** In a group of n people, what is the probability that at least two people share the same birthday?
- **Assumption:** Each bucket is equally likely.
- **Restatement:** Each day of the year is equally likely for a birthday (ignoring leap years).
- **Paradox:** Even with just $\Theta(\sqrt{n})$ people, the probability of a shared birthday exceeds 50%.

Birthday Paradox for Random hashing

How probable is to have the first collision after n elements

- **Problem:** In a group of n elements, what is the probability that at least two items map to the same bucket?
- **Restatement:** In a group of n people, what is the probability that at least two people share the same birthday?
- **Assumption:** Each bucket is equally likely.
- **Restatement:** Each day of the year is equally likely for a birthday (ignoring leap years).
- **Paradox:** Even with just $\Theta(\sqrt{n})$ people, the probability of a shared birthday exceeds 50%.
- **Restatement:** Even with just 23 people, the probability of a shared birthday exceeds 50%.

CALCULATING THE PROBABILITY

- **Probability of no collision:** It's easier to compute the probability that **all** birthdays are different.
- **For n people:**

$$\Pr[\text{all different}] = \frac{365}{365} \times \frac{364}{365} \times \cdots \times \frac{365 - n + 1}{365} = \frac{365!}{(365 - n)! \times 365^n}$$

- **Probability of at least one collision:**

$$\Pr[\text{at least one shared birthday}] = 1 - P[\text{all different}]$$

- **Example with $m = 23$:**

$$\Pr[\text{at least one shared birthday}] \approx 1 - 0.493 = 0.507$$

- **Conclusion:** With just 23 people, there is over a 50% chance that two people share a birthday.

Calculating the Probability

- **Probability of no collision:** It's easier to compute the probability that **all** birthdays are different.
- **For m items to n buckets:**

$$\Pr[\text{all different}] = \frac{n}{n} \times \frac{n-1}{n} \times \cdots \times \frac{n-m+1}{n} = \prod_{k=1}^{m-1} (1 - \frac{k}{n})$$

Calculating the Probability

- **Probability of no collision:** It's easier to compute the probability that **all** birthdays are different.
- **For m items to n buckets:**

$$\Pr[\text{all different}] = \frac{n}{n} \times \frac{n-1}{n} \times \cdots \times \frac{n-m+1}{n} = \prod_{k=1}^{m-1} (1 - \frac{k}{n})$$

The only inequality that you must know: $1 - x \leq e^{-x}$

Calculating the Probability

- **Probability of no collision:** It's easier to compute the probability that **all** birthdays are different.
- **For m items to n buckets:**

$$\Pr[\text{all different}] = \frac{n}{n} \times \frac{n-1}{n} \times \cdots \times \frac{n-m+1}{n} = \prod_{k=1}^{m-1}(1 - \frac{k}{n})$$

The only inequality that you must know: $1 - x \leq e^{-x}$

$$\Pr[\text{all different}] = \prod_{k=1}^{m-1}(1 - \frac{k}{n})$$

Calculating the Probability

- **Probability of no collision:** It's easier to compute the probability that **all** birthdays are different.
- **For m items to n buckets:**

$$\Pr[\text{all different}] = \frac{n}{n} \times \frac{n-1}{n} \times \cdots \times \frac{n-m+1}{n} = \prod_{k=1}^{m-1}(1-\frac{k}{n})$$

The only inequality that you must know: $1 - x \le e^{-x}$

$$\Pr[\text{all different}] = \prod_{k=1}^{m-1}(1-\frac{k}{n}) \le \prod_{k=1}^{m-1}(e^{-\frac{k}{n}})$$

Calculating the Probability

- **Probability of no collision:** It's easier to compute the probability that **all** birthdays are different.
- **For m items to n buckets:**

$$\Pr[\text{all different}] = \frac{n}{n} \times \frac{n-1}{n} \times \cdots \times \frac{n-m+1}{n} = \prod_{k=1}^{m-1}(1-\frac{k}{n})$$

The only inequality that you must know: $1 - x \leq e^{-x}$

$$\Pr[\text{all different}] = \prod_{k=1}^{m-1}(1-\frac{k}{n}) \leq \prod_{k=1}^{m-1}(e^{-\frac{k}{n}}) = (e^{-\frac{\sum_{k=1}^{m-1} k}{n}}) = e^{-\frac{\binom{m}{2}}{n}}$$

Calculating the Probability

- **Probability of no collision:** It's easier to compute the probability that **all** birthdays are different.
- **For m items to n buckets:**

$$\Pr[\text{all different}] = \frac{n}{n} \times \frac{n-1}{n} \times \cdots \times \frac{n-m+1}{n} = \prod_{k=1}^{m-1} (1 - \frac{k}{n})$$

The only inequality that you must know: $1 - x \leq e^{-x}$

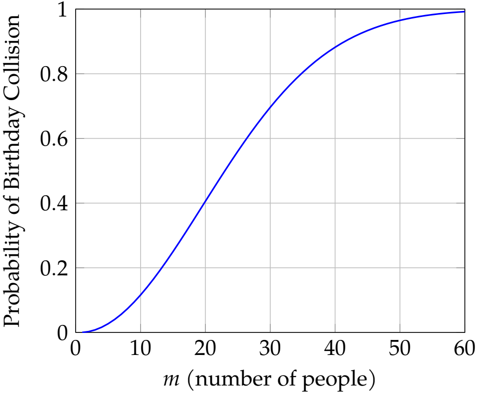
$$\Pr[\text{all different}] = \prod_{k=1}^{m-1} (1 - \frac{k}{n}) \leq \prod_{k=1}^{m-1} (e^{-\frac{k}{n}}) = (e^{-\frac{\sum_{k=1}^{m-1} k}{n}}) = e^{-\frac{\binom{m}{2}}{n}}$$

- **Probability of at least one collision:**
  $p = \Pr[\text{at least one shared birthday}] = 1 - P[\text{all different}]$

$$p \geq 1 - e^{-\frac{\binom{m}{2}}{n}} \geq \frac{1}{2} + \epsilon \Rightarrow m \geq \frac{1+\sqrt{1+8n\left[-\ln\left(\frac{1}{2}-\epsilon\right)\right]}}{2} \approx \Omega(\sqrt{n+\epsilon})$$

Probability of Collision vs. Number of People

- **Conclusion:** With at least $\Omega(\sqrt{n+\epsilon})$ items, there is over a $1/2+\epsilon$ chance that two items maps to same bucket.



Hash Function Design

Good Hash Function

- Compact and efficient.
- Minimize the collisions.

Hash Function Design

Good Hash Function

- Compact and efficient.
- Minimize the collisions.

Some ideas for hash functions

- Hash as a prefix: Collisions can result from similar prefixes. E.g. many phrases in English start with “The”.

Hash Function Design

Good Hash Function

- Compact and efficient.
- Minimize the collisions.

Some ideas for hash functions

- Hash as a prefix: Collisions can result from similar prefixes. E.g. many phrases in English start with “The”.
- $u \mod n$: Risk of collision can be large especially if say n is a power of 2.

Hash Function Design

Good Hash Function

- Compact and efficient.
- Minimize the collisions.

Some ideas for hash functions

- Hash as a prefix: Collisions can result from similar prefixes. E.g. many phrases in English start with "The".
- $u \mod n$: Risk of collision can be large especially if say n is a power of 2.
- $u \mod p$, where p is a prime: Less risk than n especially if p is not tiny, but $p \approx n$.

Random Hash Function

$h(x)$: Return a value from 0 to $n-1$ UAR.

Lemma 2

Given $h(x)$, the probability that $h(u) = h(v)$ for any $u, v \in U$ is 🤔*?*

Random Hash Function

$h(x)$: Return a value from 0 to $n-1$ UAR.

Lemma 2

Given $h(x)$, the probability that $h(u) = h(v)$ for any $u, v \in U$ is $1/n$.

Random Hash Function

$h(x)$: Return a value from 0 to $n-1$ UAR.

Lemma 2

Given $h(x)$, the probability that $h(u) = h(v)$ for any $u, v \in U$ is $1/n$.

Proof.

- There are n^2 possible pairs of values $(h(u), h(v))$. Exactly n of them have $h(u) = h(v)$, Hence, $\Pr[h(u) = h(v)] = \frac{n}{n^2} = \frac{1}{n}$.

Random Hash Function

$h(x)$: Return a value from 0 to $n-1$ UAR.

Lemma 2

Given $h(x)$, the probability that $h(u) = h(v)$ for any $u, v \in U$ is $1/n$.

Proof.

- There are n^2 possible pairs of values $(h(u), h(v))$. Exactly n of them have $h(u) = h(v)$, Hence, $\Pr[h(u) = h(v)] = \frac{n}{n^2} = \frac{1}{n}$.
- Alternate proof: Since $h(u)$ and $h(v)$ are independent:
  - Fix $h(u)$. What is the probability that $h(u) = h(v)$?

Random Hash Function

$h(x)$: Return a value from 0 to $n-1$ UAR.

Lemma 2

Given $h(x)$, the probability that $h(u) = h(v)$ for any $u, v \in U$ is $1/n$.

Proof.

- There are n^2 possible pairs of values $(h(u), h(v))$. Exactly n of them have $h(u) = h(v)$, Hence, $\Pr[h(u) = h(v)] = \frac{n}{n^2} = \frac{1}{n}$.
- Alternate proof: Since $h(u)$ and $h(v)$ are independent:
  - Fix $h(u)$. What is the probability that $h(u) = h(v)$?
  - $\Pr[h(v) = x | h(u) = x] = \frac{1}{n}$.

$\square$

RANDOM HASH FUNCTION

$h(x)$: Return a value from 0 to $n-1$ UAR.

Lemma 2

Given $h(x)$, the probability that $h(u) = h(v)$ for any $u, v \in U$ is $1/n$.

Proof.

- There are n^2 possible pairs of values $(h(u), h(v))$. Exactly n of them have $h(u) = h(v)$, Hence, $\Pr[h(u) = h(v)] = \frac{n}{n^2} = \frac{1}{n}$.
- Alternate proof: Since $h(u)$ and $h(v)$ are independent:
  - Fix $h(u)$. What is the probability that $h(u) = h(v)$?
  - $\Pr[h(v) = x | h(u) = x] = \frac{1}{n}$.

□

What is the problem with this random hash function?

Random Hash Function

$h(x)$: Return a value from 0 to $n-1$ UAR.

Lemma 2

Given $h(x)$, the probability that $h(u) = h(v)$ for any $u, v \in U$ is $1/n$.

Proof.

- There are n^2 possible pairs of values $(h(u), h(v))$. Exactly n of them have $h(u) = h(v)$, Hence, $\Pr[h(u) = h(v)] = \frac{n}{n^2} = \frac{1}{n}$.
- Alternate proof: Since $h(u)$ and $h(v)$ are independent:
  - Fix $h(u)$. What is the probability that $h(u) = h(v)$?
  - $\Pr[h(v) = x | h(u) = x] = \frac{1}{n}$.

□

What is the problem with this random hash function?
For a dictionary, Delete(u) and Lookup(u) won't work since $h(u)$ returns a random value!

Universal Class of Hash Functions

Randomly choosing a hash function

Definition

Let $\mathcal{H}$ be a class of functions such that:

- Universal property of $\mathcal{H}$: For any pair of values $u, v \in U$, the probability that a randomly chosen h has a collision for any u, v is at most $1/n$

$$\Pr_{h \sim \mathcal{H}}[h(u) = h(v)] \leq \frac{1}{n}.$$

Universal Class of Hash Functions

Randomly choosing a hash function

Definition

Let $\mathcal{H}$ be a class of functions such that:

- Universal property of $\mathcal{H}$: For any pair of values $u, v \in U$, the probability that a randomly chosen h has a collision for any u, v is at most $1/n$

$$\Pr_{h \sim \mathcal{H}}[h(u) = h(v)] \leq \frac{1}{n}.$$

- Each $h \in \mathcal{H}$ is represented compactly and can be computed efficiently.

Universal Class of Hash Functions

Randomly choosing a hash function

Definition

Let $\mathcal{H}$ be a class of functions such that:

- Universal property of $\mathcal{H}$: For any pair of values $u, v \in U$, the probability that a randomly chosen h has a collision for any u, v is at most $1/n$

$$\Pr_{h \sim \mathcal{H}}[h(u) = h(v)] \leq \frac{1}{n}.$$

- Each $h \in \mathcal{H}$ is represented compactly and can be computed efficiently.

MakeDictionary: Given $\mathcal{H}$, choose h from $\mathcal{H}$ UAR for the dictionary.

Universal Class of Hash Functions

Randomly choosing a hash function

Definition

Let $\mathcal{H}$ be a class of functions such that:

- Universal property of $\mathcal{H}$: For any pair of values $u, v \in U$, the probability that a randomly chosen h has a collision for any u, v is at most $1/n$

$$\Pr_{h \sim \mathcal{H}}[h(u) = h(v)] \leq \frac{1}{n}.$$

- Each $h \in \mathcal{H}$ is represented compactly and can be computed efficiently.

🤔What is the expected number of collisions???

Universal Class of Hash Functions

Randomly choosing a hash function

Theorem 3

Let $\mathcal{H}$ be a universal class of hash functions mapping U to $[0..n-1]$. Let $S \subseteq U$ be of size $\leq n$. The expected number of elements $s \in S$ where $h(s) = h(u)$ for any $u \in U$ when h is chosen UAR from $\mathcal{H}$ is ≤ 1.

Proof.

- Fix $u \in U$. Let X_s be a random variable that is 1 if $h(s) = h(u)$; 0 otherwise.

Universal Class of Hash Functions

Randomly choosing a hash function

Theorem 3

Let $\mathcal{H}$ be a universal class of hash functions mapping U to $[0..n-1]$. Let $S \subseteq U$ be of size $\leq n$. The expected number of elements $s \in S$ where $h(s) = h(u)$ for any $u \in U$ when h is chosen UAR from $\mathcal{H}$ is ≤ 1.

Proof.

- Fix $u \in U$. Let X_s be a random variable that is 1 if $h(s) = h(u)$; 0 otherwise.
- Let $X = \sum_{s \in S} X_s$.

UNIVERSAL CLASS OF HASH FUNCTIONS

RANDOMLY CHOOSING A HASH FUNCTION

Theorem 3

Let $\mathcal{H}$ be a universal class of hash functions mapping U to $[0..n-1]$. Let $S \subseteq U$ be of size $\leq n$. The expected number of elements $s \in S$ where $h(s) = h(u)$ for any $u \in U$ when h is chosen UAR from $\mathcal{H}$ is ≤ 1.

Proof.

- Fix $u \in U$. Let X_s be a random variable that is 1 if $h(s) = h(u)$; 0 otherwise.
- Let $X = \sum_{s \in S} X_s$.

$$\mathbb{E}[X] = \mathbb{E}\left[\sum_{s \in S} X_s\right] = \sum_{s \in S} \mathbb{E}[X_s]$$

Universal Class of Hash Functions

Randomly choosing a hash function

Theorem 3

Let $\mathcal{H}$ be a universal class of hash functions mapping U to $[0..n-1]$. Let $S \subseteq U$ be of size $\leq n$. The expected number of elements $s \in S$ where $h(s) = h(u)$ for any $u \in U$ when h is chosen UAR from $\mathcal{H}$ is ≤ 1.

Proof.

- Fix $u \in U$. Let X_s be a random variable that is 1 if $h(s) = h(u)$; 0 otherwise.
- Let $X = \sum_{s \in S} X_s$.
- By linearity of expectation:

$$\mathbb{E}[X] = \mathbb{E}\left[\sum_{s \in S} X_s\right] = \sum_{s \in S} \mathbb{E}[X_s]$$

Universal Class of Hash Functions

Randomly choosing a hash function

Theorem 3

Let $\mathcal{H}$ be a universal class of hash functions mapping U to $[0..n-1]$. Let $S \subseteq U$ be of size $\leq n$. The expected number of elements $s \in S$ where $h(s) = h(u)$ for any $u \in U$ when h is chosen UAR from $\mathcal{H}$ is ≤ 1.

Proof.

- Fix $u \in U$. Let X_s be a random variable that is 1 if $h(s) = h(u)$; 0 otherwise.
- Let $X = \sum_{s \in S} X_s$.
- By linearity of expectation:

$$\mathbb{E}[X] = \mathbb{E}\left[\sum_{s \in S} X_s\right] = \sum_{s \in S} \mathbb{E}[X_s] \leq |S| \cdot \frac{1}{n} \leq 1 .$$

□

Designing a Universal Class of Hash Functions

🤔Can we implement Create-Insert-Delete-Lookup in
$$\Theta(1) = 1 + \alpha = 1 + \frac{|\text{universe}|}{|\text{buckets}|}$$
on expectation?

Defining $\mathcal{H}$

- Choose a prime $p \approx n$.

Designing a Universal Class of Hash Functions

🤔Can we implement Create-Insert-Delete-Lookup in

$$\Theta(1) = 1 + \alpha = 1 + \frac{|\text{universe}|}{|\text{buckets}|}$$

on expectation?

Defining $\mathcal{H}$

- Choose a prime $p \approx n$.
- Bootstrapping: All values in U are associated with a vector coordinate $x = (x_1, x_2, \ldots, x_r)$ for some r, where $0 \leq x_i < p$.
  - $r \approx \frac{\log |Universe|}{\log |Buckets|}$ for unique x per item in U.

Designing a Universal Class of Hash Functions

🤔Can we implement Create-Insert-Delete-Lookup in

$$\Theta(1) = 1 + \alpha = 1 + \frac{|\text{universe}|}{|\text{buckets}|}$$

on expectation?

Defining $\mathcal{H}$

- Choose a prime $p \approx n$.
- Bootstrapping: All values in U are associated with a vector coordinate $x = (x_1, x_2, \ldots, x_r)$ for some r, where $0 \leq x_i < p$.
  - $r \approx \frac{\log|Universe|}{\log|Buckets|}$ for unique x per item in U.
- Let $\mathcal{A}$ be the set of all vectors of the form $a = (a_1, a_2, \ldots, a_r)$, where $0 \leq a_i < p$.

Designing a Universal Class of Hash Functions

Can we implement Create-Insert-Delete-Lookup in
$$\Theta(1) = 1 + \alpha = 1 + \frac{|\text{universe}|}{|\text{buckets}|}$$
on expectation?

Defining $\mathcal{H}$

- Choose a prime $p \approx n$.
- Bootstrapping: All values in U are associated with a vector coordinate $x = (x_1, x_2, \ldots, x_r)$ for some r, where $0 \leq x_i < p$.
  - $r \approx \frac{\log|Universe|}{\log|Buckets|}$ for unique x per item in U.
- Let $\mathcal{A}$ be the set of all vectors of the form $a = (a_1, a_2, \ldots, a_r)$, where $0 \leq a_i < p$.
- $\mathcal{H}$ contains $h_a(x) = \left(\sum_{i=1}^{r} a_i x_i\right) \mod p$ for all $a \in \mathcal{A}$.

Analyze our definition of $\mathcal{H}$

Lemma 4 (Technical Lemma - Inverse at modulo)

For any prime p and any integer $z \not\equiv 0 \mod p$, and any two integers α, β, if $\alpha z \equiv \beta z \mod p$, then $\alpha \equiv \beta \mod p$.

Analyze our definition of $\mathcal{H}$

Lemma 4 (Technical Lemma - Inverse at modulo)

For any prime p and any integer $z \not\equiv 0 \mod p$, and any two integers α, β, if $\alpha z \equiv \beta z \mod p$, then $\alpha \equiv \beta \mod p$.

Proof.

Suppose $\alpha z \equiv \beta z \mod p$:

- $\iff z(\alpha - \beta) \equiv 0 \mod p$

Analyze our definition of $\mathcal{H}$

Lemma 4 (Technical Lemma - Inverse at modulo)

For any prime p and any integer $z \not\equiv 0 \mod p$, and any two integers α, β, if $\alpha z \equiv \beta z \mod p$, then $\alpha \equiv \beta \mod p$.

Proof.

Suppose $\alpha z \equiv \beta z \mod p$:

- $\iff z(\alpha - \beta) \equiv 0 \mod p$
- z is not divisible by p, so $(\alpha - \beta) \equiv 0 \mod p$.

Analyze our definition of $\mathcal{H}$

Lemma 4 (Technical Lemma - Inverse at modulo)

For any prime p and any integer $z \not\equiv 0 \mod p$, and any two integers α, β, if $\alpha z \equiv \beta z \mod p$, then $\alpha \equiv \beta \mod p$.

Proof.

Suppose $\alpha z \equiv \beta z \mod p$:

- $\iff z(\alpha - \beta) \equiv 0 \mod p$
- z is not divisible by p, so $(\alpha - \beta) \equiv 0 \mod p$.
- Hence, $\alpha \equiv \beta \mod p$.

□

Analyze our definition of $\mathcal{H}$

Lemma 4 (Technical Lemma - Inverse at modulo)

For any prime p and any integer $z \not\equiv 0 \mod p$, and any two integers α, β, if $\alpha z \equiv \beta z \mod p$, then $\alpha \equiv \beta \mod p$.

Theorem 5 (Our Main Theorem)

The class of linear functions $\mathcal{H}$ as defined previously is universal.

Analyze our definition of $\mathcal{H}$

Lemma 4 (Technical Lemma - Inverse at modulo)

For any prime p and any integer $z \not\equiv 0 \mod p$, and any two integers α, β, if $\alpha z \equiv \beta z \mod p$, then $\alpha \equiv \beta \mod p$.

Theorem 5 (Our Main Theorem)

The class of linear functions $\mathcal{H}$ as defined previously is universal.

Proof.

- Let $x = (x_1, x_2, \ldots, x_r)$ and $y = (y_1, y_2, \ldots, y_r)$ be two distinct elements of U. ($r \approx \frac{\log |U|}{\log n}$)

Analyze our definition of $\mathcal{H}$

Lemma 4 (Technical Lemma - Inverse at modulo)

For any prime p and any integer $z \not\equiv 0 \mod p$, and any two integers α, β, if $\alpha z \equiv \beta z \mod p$, then $\alpha \equiv \beta \mod p$.

Theorem 5 (Our Main Theorem)

The class of linear functions $\mathcal{H}$ as defined previously is universal.

Proof.

- Let $x = (x_1, x_2, \ldots, x_r)$ and $y = (y_1, y_2, \ldots, y_r)$ be two distinct elements of U. ($r \approx \frac{\log |U|}{\log n}$)
- We need to show that $\Pr[h_a(x) = h_a(y)] \leq 1/p$ for a randomly chosen $a \in \mathcal{A}$.

Analyze our definition of $\mathcal{H}$

Theorem 4 (Our Main Theorem)

The class of linear functions $\mathcal{H}$ as defined previously is universal.

Proof.

- Let $x = (x_1, x_2, \ldots, x_r)$ and $y = (y_1, y_2, \ldots, y_r)$ be two distinct elements of U. $(r \approx \frac{\log |U|}{\log n})$
- Let j be an index such that $x_j \neq y_j$.

Analyze our definition of $\mathcal{H}$

Theorem 4 (Our Main Theorem)

The class of linear functions $\mathcal{H}$ as defined previously is universal.

Proof.

- Let $x = (x_1, x_2, \ldots, x_r)$ and $y = (y_1, y_2, \ldots, y_r)$ be two distinct elements of U. ($r \approx \frac{\log |U|}{\log n}$)
- Let j be an index such that $x_j \neq y_j$.
- Define $a := \{\text{arb fix } a_i \text{ for } i \neq j\}$, a_j defined later.

ANALYZE OUR DEFINITION OF $\mathcal{H}$

Theorem 4 (Our Main Theorem)

The class of linear functions $\mathcal{H}$ as defined previously is universal.

Proof.

- Let j be an index such that $x_j \neq y_j$.
- Define $a := \{\text{arb fix } a_i \text{ for } i \neq j\}$, a_j defined later.
- Consider $h_a(x) = h_a(y)$:

$$\sum_{i=1}^{r} a_i x_i \equiv \sum_{i=1}^{r} a_i y_i \iff a_j(x_j - y_j) \equiv \sum_{i \neq j} a_i(y_i - x_i) \mod p \quad (1)$$

Analyze our definition of $\mathcal{H}$

Theorem 4 (Our Main Theorem)

The class of linear functions $\mathcal{H}$ as defined previously is universal.

Proof.

- Let j be an index such that $x_j \neq y_j$.
- Define $a := \{\text{arb fix } a_i \text{ for } i \neq j\}$, a_j defined later.
- Consider $h_a(x) = h_a(y)$:

$$\sum_{i=1}^{r} a_i x_i \equiv \sum_{i=1}^{r} a_i y_i \iff a_j(x_j - y_j) \equiv \sum_{i \neq j} a_i(y_i - x_i) \mod p \quad (1)$$

- Lemma 4 shows there is a single value for a_j to satisfy (1).

Analyze our definition of $\mathcal{H}$

Theorem 4 (Our Main Theorem)

The class of linear functions $\mathcal{H}$ as defined previously is universal.

Proof.

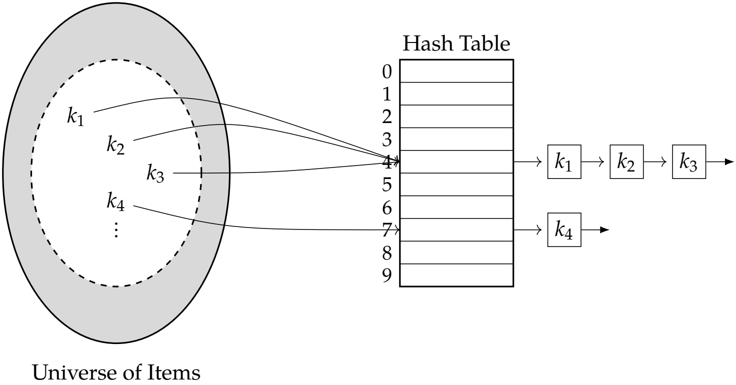
- Let j be an index such that $x_j \neq y_j$.
- Define $a := \{\text{arb fix } a_i \text{ for } i \neq j\}$, a_j defined later.
- Consider $h_a(x) = h_a(y)$:

$$\sum_{i=1}^{r} a_i x_i \equiv \sum_{i=1}^{r} a_i y_i \iff a_j(x_j - y_j) \equiv \sum_{i \neq j} a_i(y_i - x_i) \mod p \quad (1)$$

- Lemma 4 shows there is a single value for a_j to satisfy (1).
- So, $\Pr[h_a(x) = h_a(y)] \leq \frac{1}{p}$.

$\square$

Maximum Load for Universal Hash Function



🤔What is the maximum load in a bucket of the hash table on expectation?

MAXIMUM LOAD FOR UNIVERSAL HASH FUNCTION

What is the maximum load in a bucket of the hash table on expectation?

Let h be chosen from a universal hash family and let L be the maximum load of any slot. Then $\Pr\left[L > t\sqrt{\binom{m}{2}/n}\right] \le 1/t^2$ *for* $t \ge 1$.

Let $C = \sum_{x,y \in S, x \neq y} C_{x,y}$ be the total number of collisions.

1. $\mathbb{E}[C] \le \binom{m}{2}/n$.
2. **Observation:** $C \ge \binom{L}{2}$. Why?
3. $L > \rho$ implies $C > \rho^2/2$.
4. By Markov $\Pr[C > t^2\binom{m}{2}/n] \le \frac{\mathbb{E}[C]}{t^2\binom{m}{2}/n} \le \frac{1}{t^2}$.
5. Hence $\Pr\left[L > t\sqrt{\binom{m}{2}/n}\right] \le \frac{1}{t^2}$.

Maximum Load for Universal Hash Function

🤔What is the maximum load in a bucket of the hash table on expectation?

Let h be chosen from a universal hash family and let L be the maximum load of any slot. Then $\mathbb{E}[L] = O\left(\sqrt{\frac{\binom{m}{2}}{n}}\right)$.

L is a non-negative random variable in range. Hence

- Define $\beta = \sqrt{\frac{\binom{m}{2}}{n}} = \sqrt{\frac{m(m-1)}{2n}}$ and rewrite the inequality: $\Pr[L > t\beta] \le \frac{1}{t^2}$.
- Use the expectation formula:

$$\mathbb{E}[L] = \int_0^\infty \Pr[L > x]\, dx \quad \text{and} \quad \Pr[L > x] \le \min\left(1, \left(\frac{\beta}{x}\right)^2\right).$$

Maximum Load for Universal Hash Function

🤔What is the maximum load in a bucket of the hash table on expectation?

Let h be chosen from a universal hash family and let L be the maximum load of any slot. Then $\mathbb{E}[L] = O\left(\sqrt{\frac{\binom{m}{2}}{n}}\right)$.

- Split the integral:

$$\mathbb{E}[L] \le \int_0^{\beta} 1\,dx + \int_{\beta}^{m} \left(\frac{\beta}{x}\right)^2 dx.$$

- Calculate the integrals:
  - $\int_0^{\beta} 1\,dx = \beta$.
  - $\int_{\beta}^{m} \left(\frac{\beta}{x}\right)^2 dx = \beta^2 \left(\frac{1}{\beta} - \frac{1}{m}\right) = \beta\left(1 - \frac{\beta}{m}\right)$.

Maximum Load for Universal Hash Function

What is the maximum load in a bucket of the hash table on expectation?

Let h be chosen from a universal hash family and let L be the maximum load of any slot. Then $\mathbb{E}[L] = O\left(\sqrt{\frac{\binom{m}{2}}{n}}\right)$.

- Combine results:

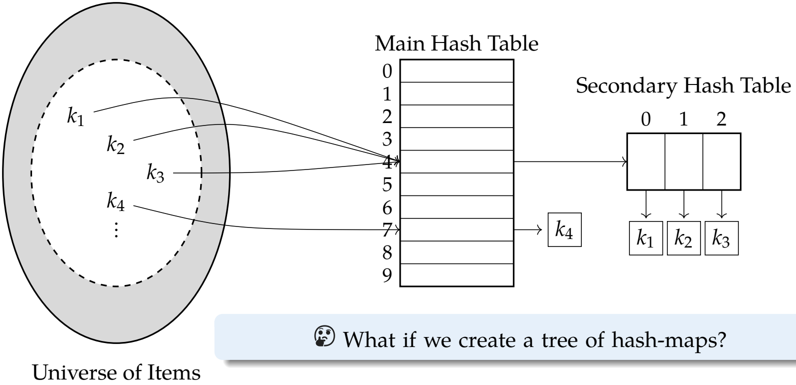
$$\mathbb{E}[L] \leq \beta + \beta\left(1 - \frac{\beta}{m}\right) = 2\beta - \frac{\beta^2}{m} \leq 2\beta.$$

Conclusion:

$$\mathbb{E}[L] \leq 2\sqrt{\frac{\binom{m}{2}}{n}}.$$

Maximum Load for Universal Hash Function

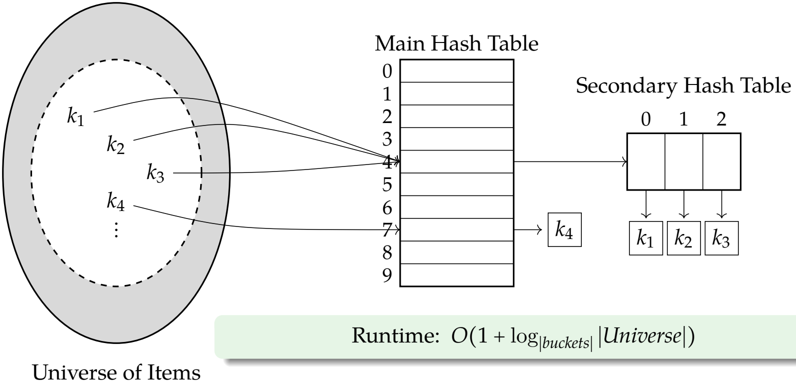
🤔What is the maximum load in a bucket of the hash table on expectation?



🤔 What if we create a tree of hash-maps?

Maximum Load for Universal Hash Function

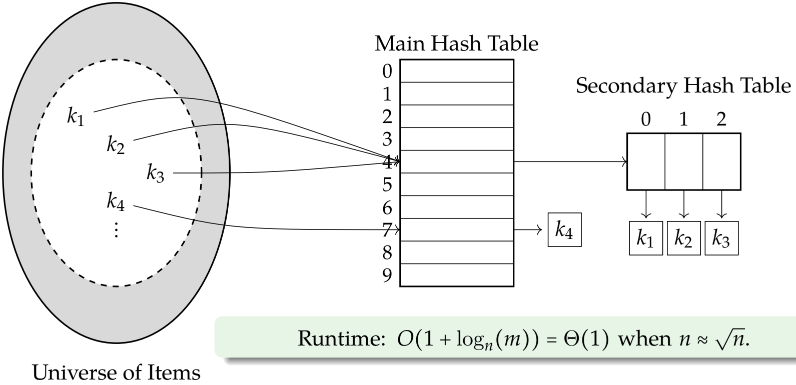
🤔What is the maximum load in a bucket of the hash table on expectation?



Runtime: $O(1 + \log_{|buckets|} |Universe|)$

Maximum Load for Universal Hash Function

What is the maximum load in a bucket of the hash table on expectation?



Runtime: $O(1 + \log_n(m)) = \Theta(1)$ when $n \approx \sqrt{n}$.


That's all Folks!

Appendix

References

Image Sources I

https://brand.wisc.edu/web/logos/