CSE 373

OCTOBER 16TH – HASHING
TODAY’S LECTURE

• Hashing
TODAY’S LECTURE

• Hashing
  • Basic Concept
TODAY’S LECTURE

• Hashing
  • Basic Concept
  • Hash functions
TODAY’S LECTURE

• Hashing
  • Basic Concept
  • Hash functions
  • Collision Resolution
TODAY’S LECTURE

• Hashing
  • Basic Concept
  • Hash functions
  • Collision Resolution
  • Runtimes
HASHING

- Introduction
HASHING

• Introduction
  • Suppose there is a set of data $M$
HASHING

• Introduction

  • Suppose there is a set of data $M$
  • Any data we might want to store is a member of this set. For example, $M$ might be the set of all strings
HASHING

• Introduction
  • Suppose there is a set of data $M$
  • Any data we might want to store is a member of this set. For example, $M$ might be the set of all strings
  • There is a set of data that we actually care about storing $D$, where $D \ll M$
HASHING

• Introduction

• Suppose there is a set of data $M$
• Any data we might want to store is a member of this set. For example, $M$ might be the set of all strings
• There is a set of data that we actually care about storing $D$, where $D << M$
• For an English Dictionary, $D$ might be the set of English words
HASHING

• What is our ideal data structure?
HASHING

• What is our ideal data structure?
  • The data structure should use $O(D)$ memory
HASHING

• What is our ideal data structure?
  • The data structure should use $O(D)$ memory
    • No extra memory is allocated
HASHING

• What is our ideal data structure?
  • The data structure should use $O(D)$ memory
    • No extra memory is allocated
  • The operation should run in $O(1)$ time
HASHING

- What is our ideal data structure?
  - The data structure should use $O(D)$ memory
    - No extra memory is allocated
  - The operation should run in $O(1)$ time
    - Accesses should be as fast as possible
HASHING

• What are some difficulties with this?
HASHING

• What are some difficulties with this?
  • Need to know the size of $D$ in advance or lose memory to pointer overhead
HASHING

• What are some difficulties with this?
  • Need to know the size of \( D \) in advance or lose memory to pointer overhead
  • Hard to go from \( M \rightarrow D \) in \( O(1) \) time
HASHING

• Memory: The Hash Table
HASHING

• Memory: The Hash Table
  • Consider an array of size $c \times D$
HASHING

• Memory: The Hash Table
  • Consider an array of size $c * D$
  • Each index in the array corresponds to some element in $M$ that we want to store.
HASHING

- Memory: The Hash Table
  - Consider an array of size $c \times D$
  - Each index in the array corresponds to some element in $M$ that we want to store.
  - The data in $D$ does not need any particular ordering.
THE HASH TABLE

• How can we do this?
THE HASH TABLE

• How can we do this?
  • Unsorted Array
THE HASH TABLE

• How can we do this?
  • Unsorted Array
THE HASH TABLE

• How can we do this?
  • Unsorted Array
THE HASH TABLE

• How can we do this?
  • Unsorted Array
THE HASH TABLE

• How can we do this?
  • Unsorted Array
THE HASH TABLE

• What is the problem here?
What is the problem here?

- Takes $O(D)$ time to find the word in the list!
THE HASH TABLE

• What is the problem here?
  • Takes O(D) time to find the word in the list
  • Same problem with sorted arrays!
THE HASH TABLE

• What is another solution?
THE HASH TABLE

• What is another solution?
  • Random mapping
THE HASH TABLE

• What’s the problem here?
THE HASH TABLE

• What’s the problem here?
  • Can’t retrieve the random variable, O(D) search!

<table>
<thead>
<tr>
<th>Kumquat</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pear</td>
<td></td>
</tr>
<tr>
<td>Durian</td>
<td></td>
</tr>
<tr>
<td>Apple</td>
<td></td>
</tr>
<tr>
<td>Orange</td>
<td></td>
</tr>
</tbody>
</table>

M
THE HASH TABLE

- What about a pseudo-random mapping?
• What about a pseudo-random mapping?
  • This is “the hash function”
The Hash Function maps the large space M to our target space D.
The Hash Function maps the large space $\textit{M}$ to our target space $\textit{D}$.

We want our hash function to do the following:
The Hash Function maps the large space $M$ to our target space $D$.

We want our hash function to do the following:

- Be repeatable: $H(x) = H(x)$ every run
HASH FUNCTIONS

• The Hash Function maps the large space $M$ to our target space $D$.

• We want our hash function to do the following:
  
  • Be repeatable: $H(x) = H(x)$ every run
  
  • Be equally distributed: For all $y,z$ in $D$, $P(H(y)) = P(H(z))$
  
  • Run in constant time: $H(x) = O(1)$
HASH EXAMPLE

• Let’s consider an example. We want to save 10 numbers from all possible Java ints
Let’s consider an example. We want to save 10 numbers from all possible Java ints.

What is a simple hash function?
HASH EXAMPLE

• Let’s consider an example. We want to save 10 numbers from all possible Java ints
  • Just use the number, but we need to mod by the table size to prevent OOB

\[ h(x) = \text{key} \% 10 \]
HASH EXAMPLE

• Let’s insert(519) table

\[ h(x) = \text{key}\%10 \]
HASH EXAMPLE

• Let’s insert(519) table
  • Where does it go?

\[
h(x) = \text{key}\%10
\]
HASH EXAMPLE

• Let’s insert(519) table
  • Where does it go?
  • $519 \% 10 =$

\[ h(x) = \text{key}\%10 \]
HASH EXAMPLE

• Let’s insert(519) table
  • Where does it go?
  • $519 \mod 10 = 9$
HASH EXAMPLE

• Insert(214)

\[ h(x) = \text{key}\%10 \]
HASH EXAMPLE

• Insert(214)

\[ h(x) = \text{key}\%10 \]
HASH EXAMPLE

- `insert(1001)`

```
h(x) = key % 10
```

```
214
519
1001
```
```
0
1
2
3
4: 214
5
6
7
8
9: 519
```
HASH EXAMPLE

• insert(1001)

\[ h(x) = key \% 10 \]
HASH EXAMPLE

- Is there a problem here?

$h(x) = \text{key} \% 10$
HASH EXAMPLE

• Is there a problem here?
  • insert(3744)
HASH EXAMPLE

• Is there a problem here?
  • insert(3744)
HASH EXAMPLE

- Is there a problem here?
  - insert(3744)
  - This is called a collision!

\[ h(x) = \text{key} \% 10 \]
HASH FUNCTION

• In reality, good hash functions are difficult to produce
  • We want a hash that distributes our data evenly throughout the space
HASH FUNCTION

• In reality, good hash functions are difficult to produce
  • We want a hash that distributes our data evenly throughout the space
  • Usually, our hash function returns some integer, which must then be modded to our table size
HASH FUNCTION

• In reality, good hash functions are difficult to produce
  • We want a hash that distributes our data evenly throughout the space
  • Usually, our hash function returns some integer, which must then be modded to our table size
  • Needs to incorporate all the data in the keys
HASH FUNCTION

• You will not have to produce hash functions, but you should recognize good ones
HASH FUNCTION

- You will not have to produce hash functions, but you should recognize good ones
  - They run in constant time
HASH FUNCTION

• You will not have to produce hash functions, but you should recognize good ones
  • They run in constant time
  • They evenly distribute the data
HASH FUNCTION

- You will not have to produce hash functions, but you should recognize good ones
  - They run in constant time
  - They evenly distribute the data
  - They return an integer
HASH EXAMPLE

• How to rectify collisions?
  • Think of a strategy for a few minutes
HASH EXAMPLE

• How to rectify collisions?
  • Think of a strategy for a few minutes
• Possible solutions:
  • Store in the next available space
How to rectify collisions?
  - Think of a strategy for a few minutes

Possible solutions:
  - Store in the next available space
  - Store both in the same space
• How to rectify collisions?
  • Think of a strategy for a few minutes
• Possible solutions:
  • Store in the next available space
  • Store both in the same space
  • Try a different hash
HASH EXAMPLE

• How to rectify collisions?
  • Think of a strategy for a few minutes

• Possible solutions:
  • Store in the next available space
  • Store both in the same space
  • Try a different hash
  • Resize the array
HASH EXAMPLE

• Consider the simplest solution
HASH EXAMPLE

• Consider the simplest solution
  • Find the next available spot in the array
LINEAR PROBING

• Consider the simplest solution
  • Find the next available spot in the array
  • This solution is called *linear probing*
LINEAR PROBING

- Consider the simplest solution
  - Find the next available spot in the array
  - This solution is called linear probing

\[ h(x) = \text{key}\%10 \]
LINEAR PROBING

• What are the problems with this?
LINEAR PROBING

• What are the problems with this?
  • How do we search for 3744?
LINEAR PROBING

• What are the problems with this?
  • How do we search for 3744?
    • Need to go to 4, and then cycle through all of the entries until--
LINEAR PROBING

• What are the problems with this?
  • How do we search for 3744?
    • Need to go to 4, and then cycle through all of the entries until we find the element or find a blank space
LINEAR PROBING

• What are the problems with this?
  • How do we search for 3744?
    • Need to go to 4, and then cycle through all of the entries until we find the element or find a blank space
  • What if we need to add something that ends in 5?
LINEAR PROBING

• What are the problems with this?
  • How do we search for 3744?
    • Need to go to 4, and then cycle through all of the entries until we find the element or find a blank space
  • What if we need to add something that ends in 5?
    • It also ends up in this problem area
LINEAR PROBING

• What are the problems with this?
  • How do we search for 3744?
    • Need to go to 4, and then cycle through all of the entries until we find the element or find a blank space
  • What if we need to add something that ends in 5?
    • It also ends up in this problem area
    • This is called clustering
CLUSTERING

• What are the negative effects of clustering?
CLUSTERING

• What are the negative effects of clustering?
  • If the cluster becomes too large, two things happen:
CLUSTERING

• What are the negative effects of clustering?
  • If the cluster becomes too large, two things happen:
    • The chances of colliding with the cluster increase
What are the negative effects of clustering?

- If the cluster becomes too large, two things happen:
  - The chances of colliding with the cluster increase
  - The time it takes to find something in the cluster increases
What are the negative effects of clustering?

- If the cluster becomes too large, two things happen:
  - The chances of colliding with the cluster increase
  - The time it takes to find something in the cluster increases. This isn’t $O(1)$ time!
How can we solve this problem?
How can we solve this problem?

- Resize the array
CLUSTERING

• How can we solve this problem?
  • Resize the array
    • Give the elements more space to avoid clusters
CLUSTERING

- How can we solve this problem?
  - Resize the array
    - Give the elements more space to avoid clusters. *How long does this take?*
CLUSTERING

• How can we solve this problem?
  • Resize the array
    • Give the elements more space to avoid clusters. *How long does this take?* $O(n)$ all of the elements need to be rehashed.
CLUSTERING

• How can we solve this problem?
  • Resize the array
    • Give the elements more space to avoid clusters. *How long does this take? O(n) all of the elements need to be rehashed.*
  • Store multiple items in one location
CLUSTERING

• How can we solve this problem?
  • Resize the array
    • Give the elements more space to avoid clusters. How long does this take? \( O(n) \) all of the elements need to be rehashed.
  • Store multiple items in one location
    • This is called chaining
CLUSTERING

• How can we solve this problem?
  • Resize the array
    • Give the elements more space to avoid clusters. How long does this take? $O(n)$ all of the elements need to be rehashed.
  • Store multiple items in one location
    • This is called chaining
    • We’ll discuss it later
COLLISIONS

• Hash table methods are defined by how they handle collisions
COLLISIONS

• Hash table methods are defined by how they handle collisions
• Two main approaches
COLLISIONS

• Hash table methods are defined by how they handle collisions
• Two main approaches
  • Probing
COLLISIONS

• Hash table methods are defined by how they handle collisions

• Two main approaches
  • Probing
  • Chaining
COLLISIONS

• Probing
COLLISIONS

• Probing
  • Linear probing
COLLISIONS

• Probing
  • Linear probing
    • Try the appropriate hash table row first
COLLISIONS

• Probing
  • Linear probing
    • Try the appropriate hash table row first
    • Increase the index by one until a spot is found
COLLISIONS

• Probing
  • Linear probing
    • Try the appropriate hash table row first
    • Increase the index by one until a spot is found
    • Guaranteed to find a spot if it is available
COLUMNS

• Probing
  • Linear probing
    • Try the appropriate hash table row first
    • Increase the index by one until a spot is found
    • Guaranteed to find a spot if it is available
    • If the array is too full, its operations reach $O(n)$ time
COLLISIONS

• Probing
  • Quadratic Probing
COLLISIONS

- Probing
  - Quadratic Probing
    - Rather than increasing by one each time, we increase by the squares
COLLISIONS

• Probing
  • Quadratic Probing
    • Rather than increasing by one each time, we increase by the squares
    • k+1, k+4, k+9, k+16, k+25
COLLISIONS

• Probing
  • Quadratic Probing
    • Rather than increasing by one each time, we increase by the squares
    • $k+1, k+4, k+9, k+16, k+25$
    • Certain tables can cause secondary clustering
COLLISIONS

• Probing
  • Quadratic Probing
    • Rather than increasing by one each time, we increase by the squares
    • $k+1, k+4, k+9, k+16, k+25$
    • Certain tables can cause secondary clustering
COLLISIONS

• Probing
  • Quadratic Probing
    • Rather than increasing by one each time, we increase by the squares
    • $k+1$, $k+4$, $k+9$, $k+16$, $k+25$
    • Certain tables can cause secondary clustering
    • Can fail to insert if the table is over half full
COLLISIONS

• Probing
  • Secondary Hashing
COLLISIONS

• Probing
  • Secondary Hashing
    • If two keys collide in the hash table, then a secondary hash indicates the probing size
C O L L I S I O N S

• Probing
  • Secondary Hashing
    • If two keys collide in the hash table, then a secondary hash indicates the probing size
    • Need to be careful, possible for infinite loops with a very empty array