First, a cool program
By the end of today I am going to show you how this works.
We’ve Gotten Ahead of Ourselves

Source: The Hobbit
Start at the Beginning

Source: The Hobbit
Hashing: The Heart of the Hash Map
Today’s Goals

1. Understand how the HashMap works
2. Add hashing to your algorithmic toolbox
3. Practice linked lists
Today’s Route

Hashing

Collisions

Real Hash fns

Code

Other Applications

(eg HashMap<string, string>)
First Objective

(eg HashMap<string, string>)
// adds name / text pair to dataset
put(string articleName, string articleHTML)

// returns corresponding articleHTML
get(string articleName)

// removes the article
remove(string articleName)
Wikipedia is Big!

5 million articles and growing

I'm ruined!
Where did it all go wrong?

Uh, I'm guessing the part at the beginning where you decided to print out Wikipedia.

Source: xkcd
**Key Value Pairs**

**Key = Title, Value = Article**

### Key: “Mantis Shrimp”

#### Value:

**Mantis shrimp**

From Wikipedia, the free encyclopedia

Mantis shrimp or stomatopods are marine crustaceans, the members of the order Stomatopoda. Most species can grow to around 10 centimetres (3.9 in) in length, though a few species reach up to 38 cm (15 in). The largest ever caught has a length of 46 cm (18 in) in the ocean near Fort Pierce, Florida, USA. The carapace of mantis shrimp covers only the rear part of the head and the first four segments of the thorax. There are more than 400 species of mantis shrimp. Varieties range from shades of brown to vivid colours, and are among the most important predators in many shallow, tropical and subtropical marine habitats. Despite being common, they are poorly understood as many species spend most of their time tucked away in burrows and holes.

![Mantis shrimp](image)

**Scientific classification**

- **Kingdom:** Animalia
- **Phylum:** Arthropoda
- **Subphylum:** Crustacea
- **Class:** Malacostraca
- **Subclass:** Hoplocarida
- **Order:** Stomatopoda
- **Family:** Spiny lobster

![Superfamilies and families](image)

### Key: “Antelope Canyon”

#### Value:

**Antelope Canyon**

Antelope Canyon is a slot canyon in the American Southwest. It is located on Navajo land east of Page, Arizona. Antelope Canyon includes two separate, photogenic slot canyon sections, referred to individually as Upper Antelope Canyon or the Crack, and Antelope Canyon: The Corkscrew. The Navajo name for Upper Antelope Canyon is Tsi biiʼnáhí, which means “the place where water runs through rocks.” Lower Antelope Canyon is Heiśídzáin, (advertised as “Horseshoe” by the Navajo Parks and Recreation Department), or “spiral rock arches.” Both are located within the LeChee Chapter of the Navajo Nation.

![Antelope Canyon](image)

**Contents**

1. Geology
2. Tourism and photography
   2.1 Upper Antelope Canyon

![A beam of light in Upper Antelope Canyon](image)
**Key:** "Mantis Shrimp"

**Value:**
"<p><b>Mantis shrimp</b> or <b>stomatopods</b> are marine <a href="/wiki/Crustacean" title="Crustacean">crustaceans</a>, the members of the <a href="/wiki/Order_(biology)" title="Order (biology)">order</a> <b>Stomatopoda</b>. though a few species reach up to 38..."
Review

Binary Tree Search
Could Use Binary Search Tree

Except the values are strings and we use string comparison when making the tree.
Could Use Binary Search Tree

Key's are Strings
$\log_2 5 \text{ million} = 22$
Today we will see something even better.
Thought Experiment

What if: we store articles in one gigantic vector

Wikipedia:

0  Antelope Canyon
1  Stanford Band
2  John Coltrane
3  Cuckoo Hashing
4  Methuselah
Thought Experiment

What if: we want to get(“Mantis shrimp”)?

Wikipedia:

0  1  2  3  4

Antelope Canyon  Stanford Band  John Coltrane  Cuckoo Hashing  Methuselah
Thought Experiment

What if: we want to get ("Mantis shrimp")?

Wikipedia:

0 1 2 3 4

Antelope Canyon  Stanford Band  John Coltrane  Cuckoo Hashing  Methuselah
Thought Experiment

What if: we want to get ("Mantis shrimp")?

Wikipedia:
Thought Experiment

What if: we want to get ("Mantis shrimp")?

Wikipedia:

0 1 2 3 4

Antelope Canyon Stanford Band John Coltrane Cuckoo Hashing Methuselah
Thought Experiment

What if: we want to get("Mantis shrimp")?

Wikipedia:

0  Antelope Canyon
1  Stanford Band
2  John Coltrane
3  Cuckoo Hashing
4  Methuselah

Thought Experiment
Has to be a better way!
What if: there was a way to know where our article belonged just by looking at the title?
Hash Function

\[
\text{key} \rightarrow \text{Hash Fn} \rightarrow \text{array index}
\]

\[
\text{int hash(string key);}\]
Hash Function

\[
\text{title} \rightarrow \text{Hash Fn} \rightarrow \text{array index}
\]

\[
\text{int hash(string title);}
\]
Wikipedia with Hashing

Wikipedia:

39  40  41  42  43

Hash Fn

?
put("mantis shrimp", html)

Wikipedia:

39  40  41  42  43
put("mantis shrimp", html)

Wikipedia:

```
39 40 41 42 43
```

"mantis shrimp" → Hash Fn
put("mantis shrimp", html)
put("mantis shrimp", html)
put("mantis shrimp", html)

Wikipedia:
Wonderful!
get("mantis shrimp")

Wikipedia:

39  40  41  42  43

Mantid shrimp

Hash Fn
get("mantis shrimp")

Wikipedia:
get("mantis shrimp")
get("mantis shrimp")

Wikipedia:

```
39  40  41  42  43
```

Mantis shrimp

"mantis shrimp" → Hash Fn → 42
Property #1: Consistent

If you pass in the same input, you will *always* get the same index.

```
"stanford band" \rightarrow \text{Hash Fn} \rightarrow 314

"stanford band" \rightarrow \text{Hash Fn} \rightarrow 314

"mantis shrimp" \rightarrow \text{Hash Fn} \rightarrow 42
```
Property #2: Well Distributed

If you pass in the different inputs, a hash function will return different indices as often as possible.

Diagram:
- "cuckoo hashing" maps to 8
- "john coltrane" maps to 42
- "mantis shrimp" maps to 77
- "stanford band" maps to 314
What Does a Hash Function Look Like?
The Hashing Hat

The sorting hat looks at its “key” and chooses a “bucket” to put it into.
Volunteer
Today’s Route

(eg HashMap<string, string>)

Hashing

Collisions

Real Hash fns

Code Wikipedia

HashMaps River

Other Applications
Today’s Route

(eg HashMap<string, string>)
**HashMap**

```java
Article ** data = new Article* [NUM_BUCKETS];
```

where the node type is `Article`
Wikipedia:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

49,999
HashMap

Wikipedia:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

\[ \text{Hash Fn} \]

49,999
Our old friend, linked lists!
HashMap

Wikipedia:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 49,999

Article:

Key
Value
Next

Hash Fn
get("Antelope Canyon")

Wikipedia:

49,999
get("Antelope Canyon")

Wikipedia:

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
```

"Antelope Canyon" → Hash Fn
get("Antelope Canyon")

Wikipedia:

```
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
</table>
```

```
| 49,999 |
```

```
"Antelope Canyon" → Hash Fn → 14
```
get("Antelope Canyon")
get("Antelope Canyon")
get("Antelope Canyon")

Wikipedia:

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
```

```
["Antelope Canyon"]
```

```
Hash Fn -> 14
```

```
49,999
```
get("Antelope Canyon")
HashMap

Wikipedia:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 49,999

HashFn
put("John Coltrane", html)

Wikipedia:

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
```

Hash Fn

49,999
put("John Coltrane", html)
put("John Coltrane", html)

Wikipedia:

```
49,999
```

```
put("John Coltrane", html)
```

```
"John Coltrane"
```

```
Hash Fn
```

```
\[ 8 \]
```
put("John Coltrane", html)
put("John Coltrane", html)
put("John Coltrane", html)

Wikipedia:

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
```

"John Coltrane" → Hash Fn → 8
```cpp
put("John Coltrane", html)
```
put("John Coltrane", html)
HashMap Algorithm Summary

**Initialization:**
0. Make a large array of linked lists

**Getting and Setting:**
1. Hash: key -> bucketIndex.
2. Jump directly to the bucket head.
3. Run linear search over the short list.
Today’s Route

(eg HashMap<string, string>)

Hashing

Collisions

Real Hash fns

Code Wikipedia

HashMaps River

Other Applications
Today’s Route

Hashing

Collisions

Real Hash fns

HashMaps River

Code Wikipedia

Other Applications

(eg HashMap<string, string>)
Hash Function Template

\[ \text{key} \rightarrow \text{pre hash} \rightarrow \text{hash} \]

% by num. buckets

In range \([0, \text{max})\)

Generate a really large (positive) number
Simple String Hash:

preHash = The sum of all of the characters
Given string $a$ as a key.

\[
\text{prehash} = a[0] + a[1] + \cdots + a[n - 1]
\]

\text{return (prehash \% numBuckets)}
A Basic Hash Function

```cpp
int basicHash(string key) {
    int preHash = 0;
    for(int i = 0; i < key.length(); i++) {
        preHash += key[i];
    }
    return preHash % NUM_BUCKETS;
}
```
Experiment: I hashed 50 thousand wikipedia articles into 50 thousand buckets and looked at the number of collisions in each bucket.
Judge Strings by the Contents of their Characters

preHash = Add each character in a string

preHash = Add each character in a string, weighted by $31^i$

Each pixel is one bucket

Number of collisions

- 0
- 1
- 2
- 3
- 4
- >= 5
The C++ Hash

Given string $a$ as a key.

prehash =

$$a[0] + 31 \cdot a[1] + 31^2 \cdot a[2] + \cdots + 31^n \cdot a[n]$$

return (prehash % numBuckets)

* In C++, the weights are in reverse order. That is not important :).
Why 31?
What if the constant was 2?

\[ a[0] + 2 \cdot a[1] + 2^2 \cdot a[2] + \cdots + 2^n \cdot a[n] \]
Question:
What is the big-O of the get method?
[suspense]
Answer: O(1)

* In expectation
What if nBuckets is too small?!?
HashMap Big-O

- put: $O(\text{Ign})$
- get: $O(\text{Ign})$
- remove: $O(\text{Ign})$
- hash map: $O(1)$
- binary search tree: $O(\text{Ign})$
Knead to Program It

Source: timescity
Gopher It

Source: Planet Animal Zone
Hashing

Collisions

Other Applications

Code

Wikipedia

Real Hash fns

HashMaps River

HashMap (e.g., HashMap<string, string>)

Today’s Route
Today’s Route

(eg HashMap<string, string>)

Hashing

Collisions

Real Hash fns

Code Wikipedia

HashMaps River

Other Applications

CS106B
Today’s Route

- Hashing
- Collisions
- HashMaps River
- Real Hash fns
- Code Wikipedia

(eg HashMap<string, string>)
Many Applications of Hashing

Jeff Jarvis @jeffjarvis
Hey @Snowden, for context, how long would it take the NSA to dedupe 650k emails?

Edward Snowden @Snowden
@jeffjarvis Drop non-responsive To:/CC:/BCC: hash both sets, then subtract those that match. Old laptops could do it in minutes-to-hours.

The first laptop ever made...
And Here We Are
Similar to Wikipedia:
Huge search problem.
Spectrogram

Does anyone recognize this song?
Spectrogram

Does anyone recognize this song?
Notes Over Time

Wang, A. An Industrial-Strength Audio Search Algorithm
Hash the whole thing?
No
Find Pairs of Notes

Wang, A. An Industrial-Strength Audio Search Algorithm
Note Pairs

Wang, A. An Industrial-Strength Audio Search Algorithm
Note/Songs Pairs

**Shazam**

<table>
<thead>
<tr>
<th>Key:</th>
<th>Value:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_2$</td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td>$f_1$</td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td>$\Delta t$</td>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>

**Wikipedia**

<table>
<thead>
<tr>
<th>Key:</th>
<th>Value:</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Antelope Canyon”</td>
<td>Antelope Canyon is a slot canyon in the American Southwest. It is located on Navajo land east of Page, Arizona. Antelope Canyon includes two separate, photogenic slot canyon sections, referred to individually as Upper Antelope Canyon or The Crack, and Antelope Canyon or The Corkscrew. The Navajo name for Upper Antelope Canyon is Tsé hiihiihini, which means &quot;the place where water runs through rocks.&quot; Lower Antelope Canyon is Huddleson (advertised as &quot;Huddleson Mesa&quot; by the Navajo Parks and Recreation Department), or &quot;typical rock arches.&quot; Both are located within the LeChee Chapter of the Navajo Nation.</td>
</tr>
</tbody>
</table>

**Value:**

- You Can Call Me Al – Paul Simon. 7s,
- You Can Call Me Al – Paul Simon. 43s,
- All Right Now – Police. 18s

Wang, A. An Industrial-Strength Audio Search Algorithm
Note Pair Hashing

- Dirty Paws (94s)
- All Right (18s)
- Now (18s)
- You Can Call (7s)
- Me At
- You Can Call (43s)
- Me At
- Riptide (3s)
Note Pair Hashing

![Graph showing frequency over time and vote distribution for different songs and fragments.]
Note Pair Hashing

- Window for n

![Graph showing time vs. frequency with a window highlighted]

- Votes
  - Dirty Paws (94s)
  - All Right (18s)
  - You Can Call Me At (7s)
  - You Can Call Me At (43s)
  - Riptide (3s)
Note Pair Hashing

![Graph showing note pair hashing](image)

**Votes**
- Dirty Paws (94s)
- All Right (18s)
- You Can Call (13s)
- You Can Call (43s)
- Riptide (3s)
Note Pair Hashing

![Graph showing frequency vs. time with highlighted area]

- Votes
  - Dirty Paws (94s)
  - All Right (18s)
  - You Can Call (3s)
  - You Can Call (43s)
  - Riptide (3s)
Note Pair Hashing

[Graph showing frequency over time]

[Votes distribution for different songs]
```c
int hash(int f1, int f2, int timeDelta) {
    int p = 31;
    int pre = f1 + (p * f2) + (p * p * timeDelta);
    return pre % NUM_BUCKETS;
}
```

You Can Call Me Al – Paul Simon. 23s,
You Can Call Me Al – Paul Simon. 54s,
Message in a Bottle – Police. 92s

Wang, A. An Industrial-Strength Audio Search Algorithm
Code Available
Review
HashMap Algorithm Summary

Initialization:
0. Make a large array of linked lists

Getting and Setting:
1. Hash: key -> bucketIndex.
2. Jump directly to the bucket head.
3. Run simple search over the list.

* If ever the # elements of the hash map is much larger than the number of buckets, double size and “rehash”
The HashMap needs a Hash Function
int hash(string key);

1. Consistent

2. Well Distributed
Today’s Goals

1. Understand how the HashMap works
2. Add hashing to your algorithmic toolbox
3. Practice linked lists
1. Understand how the HashMap works
2. Add hashing to your algorithmic toolbox
3. Practice linked lists
Thank you
Life in the internet

1 feel oversimplified

-Chris