Hash Tables

Dr. Mattox Beckman

Illinois Institute of Technology
Department of Computer Science

Objectives

You should be able to...

- Explain the operations of a hash table: insert, find, delete.
- Explain the following collision resolution methods:
  - Chaining
  - Open Addressing
    - Linear Probing
    - Quadratic Probing
    - Double Hashing

Motivation

- What are the advantages of arrays?
  - They are fast ($\mathcal{O}(1)$)—if you know where the data is.
- What are the disadvantages of arrays?
  - They are fixed size, it takes a long time ($\mathcal{O}(n)$)to find something we put into it (unless we sort it), we can only index using an integer.

Motivation

- What are the advantages of arrays?
  - They are fast ($\mathcal{O}(1)$)—if you know where the data is.
- What are the disadvantages of arrays?
  - They are fixed size, it takes a long time ($\mathcal{O}(n)$)to find something we put into it (unless we sort it), we can only index using an integer.

Motivation

- What are the advantages of arrays?
  - They are fast ($\mathcal{O}(1)$)—if you know where the data is.
- What are the disadvantages of arrays?
  - They are fixed size, it takes a long time ($\mathcal{O}(n)$)to find something we put into it (unless we sort it), we can only index using an integer.

Introduction Motivation

Motivation

- What are the advantages of arrays?
  - They are fast ($\mathcal{O}(1)$)—if you know where the data is.
- What are the disadvantages of arrays?
  - They are fixed size, it takes a long time ($\mathcal{O}(n)$)to find something we put into it (unless we sort it), we can only index using an integer.

What we want: $\mathcal{O}(1)$ access to add and find data, not have to worry about the size, not have to know the location of the data.

Definition

- A *hash table* is an *array* t together with a *hashing function* h.
- The *hash function* $h(x)$ takes our data x and converts it into an integer i.
- Then $t[i] \leftarrow x$.
- Remember linear-time sorting? We can tell where the data should go simply by looking at it, no need for comparisons.
- We assume that $h(x)$ runs quickly ($\mathcal{O}(1)$).
- We'll talk about good hashing functions next time.

Hash Tables General Principles

Example

Define a Hash Function

x	$h(x)$
apple	24
banana	35
cherry	18
durian	75
eggplant	44
fig	52
grape	6

i	$t[i]$
0	
1	
2	
3	apple
4	
5	
6	

Insert apple:
$h(\text{apple})\ mod\ 7 = 3$

Example

Define a Hash Function

x	$h(x)$
apple	24
banana	35
cherry	18
durian	75
eggplant	44
fig	52
grape	6

i	$t[i]$
0	banana
1	
2	
3	apple
4	
5	
6	

Insert banana:
$h(\text{banana}) \ mod\, 7 = 0$

Example

Define a Hash Function

x	$h(x)$
apple	24
banana	35
cherry	18
durian	75
eggplant	44
fig	52
grape	6

i	$t[i]$
0	banana
1	
2	
3	apple
4	cherry
5	
6	

Insert cherry:
$h(\text{cherry}) \ mod\, 7 = 4$

Example

Define a Hash Function

x	$h(x)$
apple	24
banana	35
cherry	18
durian	75
eggplant	44
fig	52
grape	6

i	$t[i]$
0	banana
1	
2	
3	apple
4	cherry
5	durian
6	

Insert durian:
$h(\text{durian}) \ mod\, 7 = 5$

Example

Define a Hash Function

x	$h(x)$
apple	24
banana	35
cherry	18
durian	75
eggplant	44
fig	52
grape	6

i	$t[i]$
0	banana
1	
2	eggplant
3	apple
4	cherry
5	durian
6	

Insert eggplant:
$h(\text{eggplant}) \ mod\, 7 = 2$

Example

Define a Hash Function

x	$h(x)$
apple	24
banana	35
cherry	18
durian	75
eggplant	44
fig	52
grape	6

i	$t[i]$
0	banana
1	
2	eggplant
3	apple
4	cherry
5	durian
6	

Insert fig:
$h(\text{fig})\ mod\ 7 = 3$... problem....

Collision Handling

Two things you can do...

There are two things you can do with a collision.

1. You can put both values into the same bucket.
2. You can put the collided value into a different bucket.

Separate Chaining

With separate chaining, the buckets are linked lists.

x	$h(x)$
apple	24
banana	35
cherry	18
durian	75
eggplant	44
fig	52
grape	5

i	$t[i]$
0	banana
1	
2	eggplant
3	fig $\longrightarrow$ apple
4	cherry
5	durian
6	

Insert fig:
$h(\text{fig})\ mod\ 7 = 3$

Separate Chaining

With separate chaining, the buckets are linked lists.

x	$h(x)$
apple	24
banana	35
cherry	18
durian	75
eggplant	44
fig	52
grape	5

i	$t[i]$
0	banana
1	
2	eggplant
3	fig $\longrightarrow$ apple
4	cherry
5	grape $\longrightarrow$ durian
6	

Insert grape:
$h(\text{grape})\ mod\ 7 = 5$

Performance of Separate Chaining

- What do you think will happen to the performance of the hash table as more data is inserted?

Linear Probing

- Linked lists sometimes behave badly with memory: we can't tell from which page the next element will be...
- Solution: put the element in a different, empty section of the hash table.
- Several ways to pick the next element. Linear probing technique: move forward until an empty spot is found. So, if $t[i = h(x)]$ is full, try $t[i+1]$, $t[i+2]$, ...

Linear Probing Example

x	$h(x)$
apple	1
banana	3
cherry	1
durian	2
eggplant	4
fig	1

i	$t[i]$
0	
1	apple
2	
3	
4	
5	
6	

Insert apple...

Linear Probing Example

x	$h(x)$
apple	1
banana	3
cherry	1
durian	2
eggplant	4
fig	1

i	$t[i]$
0	
1	apple
2	
3	banana
4	
5	
6	

Insert banana...

Linear Probing Example

x	$h(x)$
apple	1
banana	3
cherry	1
durian	2
eggplant	4
fig	1

i	$t[i]$
0	
1	apple
2	cherry
3	banana
4	
5	
6	

Insert cherry...

Linear Probing Example

x	$h(x)$
apple	1
banana	3
cherry	1
durian	2
eggplant	4
fig	1

i	$t[i]$
0	
1	apple
2	cherry
3	banana
4	durian
5	
6	

Insert durian...

Linear Probing Example

x	$h(x)$
apple	1
banana	3
cherry	1
durian	2
eggplant	4
fig	1

i	$t[i]$
0	
1	apple
2	cherry
3	banana
4	durian
5	eggplant
6	

Insert eggplant...

Linear Probing Example

x	$h(x)$
apple	1
banana	3
cherry	1
durian	2
eggplant	4
fig	1

i	$t[i]$
0	
1	apple
2	cherry
3	banana
4	durian
5	eggplant
6	fig

Insert fig...

Primary Clustering

x	$h(x)$
apple	1
banana	3
cherry	1
durian	2
eggplant	4
fig	1

i	$t[i]$
0	
1	apple
2	cherry
3	banana
4	durian
5	eggplant
6	fig

Note what happens when “fig” is inserted.
A collision tends to create a *cluster* that will make it more likely for a collision in the future.

Deletion

x	$h(x)$
apple	1
banana	3
cherry	1
durian	2
eggplant	4
fig	1

i	$t[i]$
0	
1	apple
2	cherry
3	banana
4	durian
5	eggplant
6	fig

Deletion must be handled carefully. Suppose we delete “banana” now...

Deletion

x	$h(x)$
apple	1
banana	3
cherry	1
durian	2
eggplant	4
fig	1

i	$t[i]$
0	
1	apple
2	cherry
3	
4	durian
5	eggplant
6	fig

Deletion must be handled carefully. Suppose we delete “banana” now... and then try to find “fig”. What happens?

Deletion

x	$h(x)$
apple	1
banana	3
cherry	1
durian	2
eggplant	4
fig	1

i	$t[i]$
0	
1	apple
2	cherry
3	X
4	durian
5	eggplant
6	fig

We need to put a placeholder in spots containing deleted elements.

Quadratic Probing

- Linear probing causes clustering. Perhaps this can be avoided by picking a different collision resolution method.
- Quadratic probing technique: move forward by squares until an empty spot is found. So, if $t[i = h(x)]$ is full, try $t[i+1^2]$, $t[i+2^2]$, $t[i+3^2]$,...

Quadratic Probing Example

x	$h(x)$
apple	1
banana	3
cherry	1
durian	2
eggplant	4
fig	1

i	$t[i]$
0	
1	apple
2	
3	
4	
5	
6	

Insert apple...

Quadratic Probing Example

x	$h(x)$
apple	1
banana	3
cherry	1
durian	2
eggplant	4
fig	1

i	$t[i]$
0	
1	apple
2	
3	banana
4	
5	
6	

Insert banana...

Quadratic Probing Example

x	$h(x)$
apple	1
banana	3
cherry	1
durian	2
eggplant	4
fig	1

i	$t[i]$
0	
1	apple
2	cherry
3	banana
4	
5	
6	

Insert cherry...

Quadratic Probing Example

x	$h(x)$
apple	1
banana	3
cherry	1
durian	2
eggplant	4
fig	1

i	$t[i]$
0	
1	apple
2	cherry
3	banana
4	
5	
6	durian

Insert durian...

Quadratic Probing Example

x	$h(x)$
apple	1
banana	3
cherry	1
durian	2
eggplant	4
fig	1

i	$t[i]$
0	
1	apple
2	cherry
3	banana
4	eggplant
5	
6	durian

Insert eggplant... no collision this time!

Quadratic Probing Example

x	$h(x)$
apple	1
banana	3
cherry	1
durian	2
eggplant	4
fig	1

i	$t[i]$
0	
1	apple
2	cherry
3	banana
4	eggplant
5	fig
6	durian

Insert fig...

Secondary Clustering

x	$h(x)$
apple	1
banana	3
cherry	1
durian	2
eggplant	4
fig	1

i	$t[i]$
0	
1	apple
2	cherry
3	banana
4	eggplant
5	fig
6	durian

Note what happens when “fig” is inserted.
This creates a different kind of clustering pattern, called *secondary clustering*.
It only affects collisions that start in the same cell.

Double Hashing

- With both linear and quadratic probing, we have trouble when an element hashes to an occupied space: the algorithm will always retrace the same path.
- Solution: make each key do something different in the event of a collision. Use a second hash function.
- $i = h_1(x)$. If $t[i]$ is full, try $t[i + h_2(x)]$, $t[i + 2h_2(x)]$, $t[i + 3h_2(x)]$, ...

Double Hashing Probing Example

x	$h_1(x)$	$h_2(x)$
apple	1	5
banana	3	2
cherry	1	4
durian	2	4
eggplant	4	3
fig	1	3

i	$t[i]$
0	
1	apple
2	
3	
4	
5	
6	

Insert apple...

Double Hashing Probing Example

x	$h_1(x)$	$h_2(x)$
apple	1	5
banana	3	2
cherry	1	4
durian	2	4
eggplant	4	3
fig	1	3

i	$t[i]$
0	
1	apple
2	
3	banana
4	
5	
6	

Insert banana...

Double Hashing Probing Example

x	$h_1(x)$	$h_2(x)$
apple	1	5
banana	3	2
cherry	1	4
durian	2	4
eggplant	4	3
fig	1	3

i	$t[i]$
0	
1	apple
2	
3	banana
4	
5	cherry
6	

Insert cherry...

Double Hashing Probing Example

x	$h_1(x)$	$h_2(x)$
apple	1	5
banana	3	2
cherry	1	4
durian	2	4
eggplant	4	3
fig	1	3

i	$t[i]$
0	
1	apple
2	durian
3	banana
4	
5	cherry
6	

Insert durian...

Hash Tables Collisions

Double Hashing Probing Example

x	$h_1(x)$	$h_2(x)$
apple	1	5
banana	3	2
cherry	1	4
durian	2	4
eggplant	4	3
fig	1	3

i	$t[i]$
0	
1	apple
2	durian
3	banana
4	eggplant
5	cherry
6	

Insert eggplant...

Hash Tables Collisions

Double Hashing Probing Example

x	$h_1(x)$	$h_2(x)$
apple	1	5
banana	3	2
cherry	1	4
durian	2	4
eggplant	4	3
fig	1	3

i	$t[i]$
0	fig
1	apple
2	durian
3	banana
4	eggplant
5	cherry
6	

Insert fig...

Hash Tables Collisions

Performance

- What would you expect from the performance of a hash table as it becomes full?
- After about 70-80% of the slots have been filled, it is good to resize the array, and rehash all of the elements.
- The deletion markers can be omitted during rehashing.