
DS ADVENTURE
LEARNING DATA STRUCTURES!
3 6 9
This is a tree.
DS
12
4 7
12
1 7 25
8
12 7 25


EPISODE 1
Sriju Enters the DATA STRUCTURES FACTORY
Whoa... where am I?
Welcome to Data Structures Factory.
WAAAH!
Data overload! Hurry!
Easy now. Panic never fixes data.
Because data without structure is just noise.
Data structures give data a proper shape.
I see! Structure makes data easier to use!

Next department: Stack Storage Tower


EPISODE 3
The
STACK STORAGE TOWER
This place looks... tall.
1
2
3
4
5
Top only!
7
5
Data is stored vertically here.


This is a stack.
1
2
3


We add data on the top.


And remove from the top.
1
pop
2
pop
3
4


I'm pushing data.
1


push
1
1
2
3
4
5
Last in, first out!
1
2
3
4
5


I'm pushing data.
1
1
3


I'm popping data.
pop
1
3
4


EPISODE 4
The QUEUE DISPATCH ZONE
I made it to the Queue Dispatch Zone!
EXIT
Data lines up in the order it arrives.
Queues follow FIFO.
First in... first out?
FITO
FIFO
First In First Out
Enqueue puts data at the back of the line.
First in, first out. Order matters.
enqueue
dequeue
Next stop: Hash Vaults


EPISODE 5
The HASH VAULTS
HASH VAULTS
Wow... this place looks important!
Each vault stores data safely.
Welcome to the Hash Vaults.
banana
How do we know which vault to open?
berry
ruby


EPISODE 6
The HASH VAULTS
FAST LOOKUP
There are so many vaults! Do we search them one by one?
We convert keys into numbers.
Keys are convert into numbers.
hash("apple")=4
Index generated
That was super fast!
THE SYSTEM JUMPS DIRECTLY TO THE VAULT.
That's why hash tables are efficient.


EPISODE 7
The HASH VAULTS
-- COLLISIONS & SOLUTIONS -
What if two keys choose the same vault?
berry
COLLISION ZONE!
0
0
5
6
8
3
Both keys map to the same index.


hash("berry") → 5
hash("ruby") → 4
Index generated
4

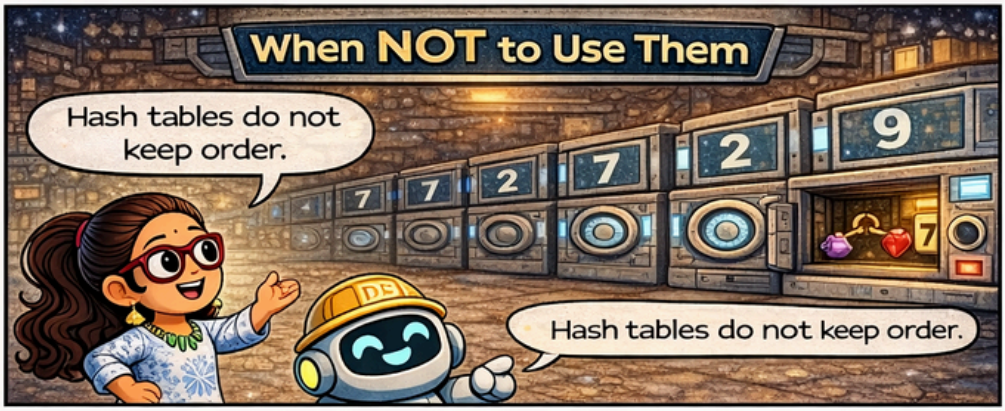

Collision detected!
berry
ruby


Both keys go to 5?!

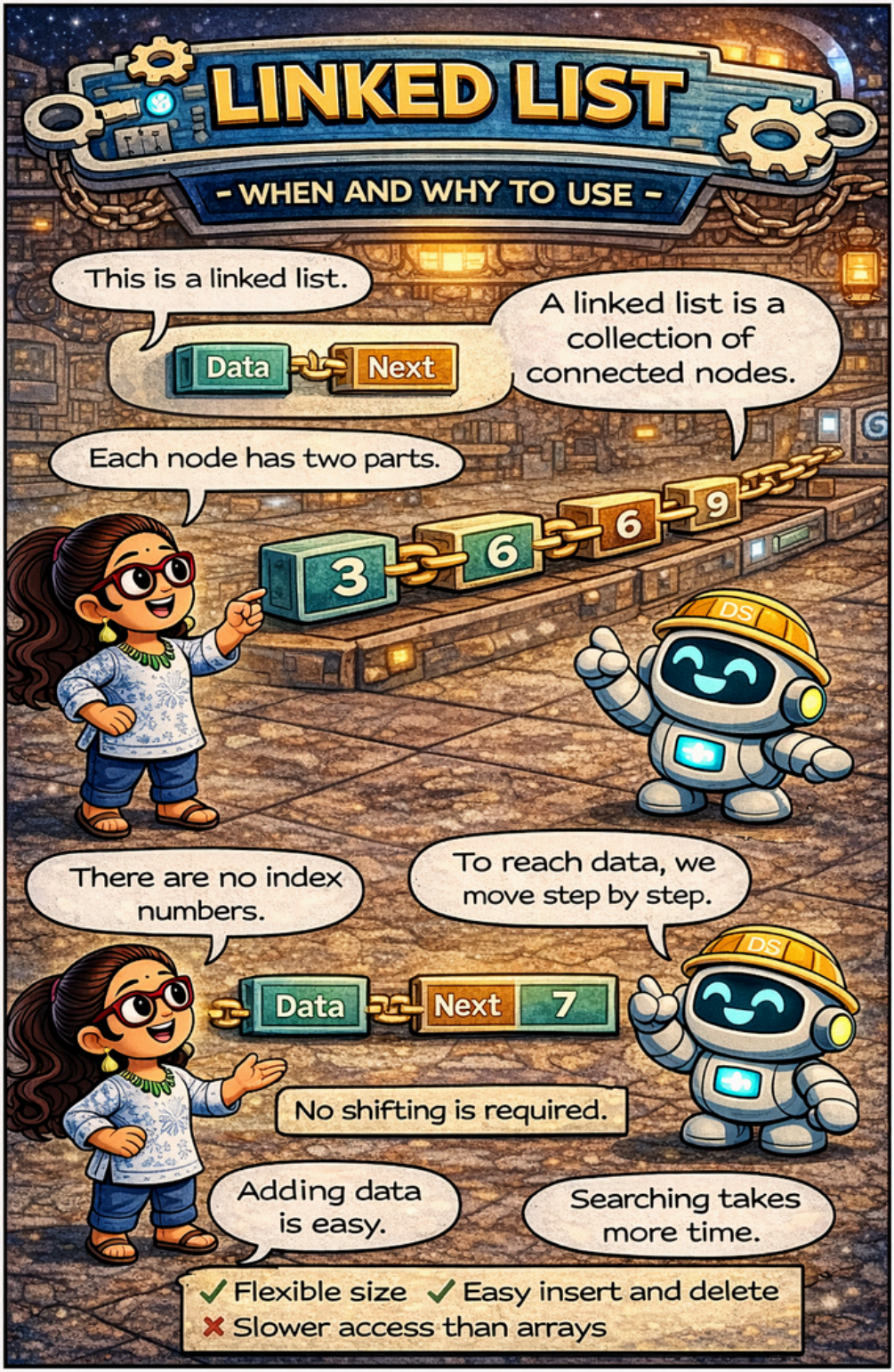

This is a colllision!!
4
Collisions can happan. We handle them.


EPISODE 8
The HASH VAULTS
REPLACING HASH TABLES
What if two keys choose the same vault?
COLLISION ZONE
5
4
5
6
8
9
berry
Links are updated.
hash("berry") → 5
hash("ruby") → 5
Index generated
4
Adding is easy.
berry
ruby
7
No shifting needed!
Collisions can happen. We handle them.
1
2
4
5
4
7
Next: Episode 7 - When to Use Hash Tables


EPISODE 9
When
USE HASH TABLES
- WHEN TO USE HASH TABLES -
Hash tables are really fast.
Keys lead directly to values.


When NOT to Use Them
Hash tables do not keep order.
Hash tables do not keep order.


Other structures handle order better.
LINKED LIST WORKSHOP
Next: Linked Lists - When and Why to Use


LINKED LIST
- WHEN AND WHY TO USE -
This is a linked list.
Data
Next
A linked list is a collection of connected nodes.
Each node has two parts.
3
6
6
9
DS
There are no index numbers.
To reach data, we move step by step.
Data
Next
7
DS
No shifting is required.
Adding data is easy.
Searching takes more time.
✓ Flexible size ✓ Easy insert and delete
✗ Slower access than arrays


TREES
– WHEN AND WHY TO USE –
This is a tree.
12
4
15
1
17
Trees have parent and child nodes.
DS
Good for sorting and hierarchies.
Root
Parent
Child
Anya
Sam
Kyle
Ivy
Adding is efficient!
Inserting data keeps order.
12
4
7
12
1
7
25
DS
✓ Flexible size ✓ Easy insert and delete
✗ Slower access than arrays