Data Structures Overview

CS 374

Why are we doing this?

Why are we doing this?

- We can't do algorithms without data.
- Different kinds of data requires different formats.

Unit Outline

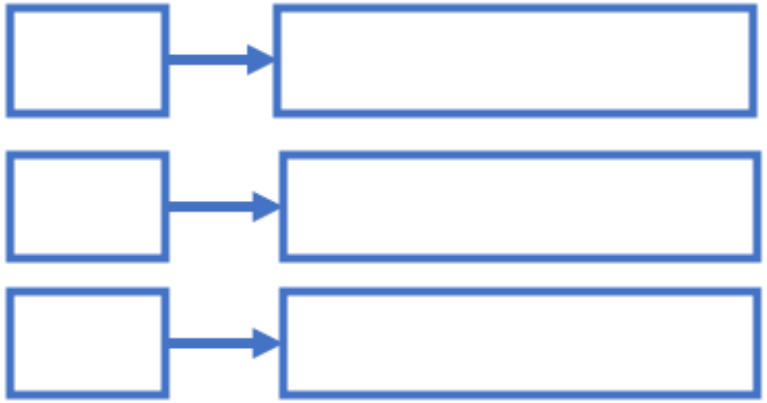
- Today – General overview. Project assigned.
- Week 4 – Specific structures
- Week 6 – Graphs, transition to graph algorithms.

Outline

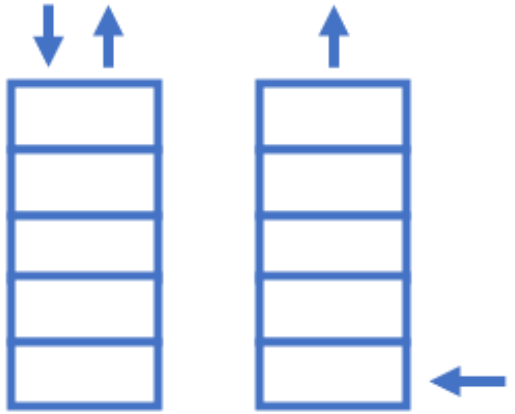
- Structures you've seen before
  - Contiguous
  - Linked
  - Stacks/Queues
- Structures you may not have seen before
  - Hashing
  - Graphs
    - Definitions, round 1
  - Binary search trees
  - Heaps (ft. a very hand-wavy explanation)
  - Sets, UnionFind (ft. a very hand-wavy explanation)
- Project

Array

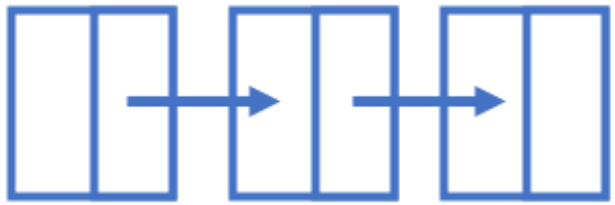
Hash Table



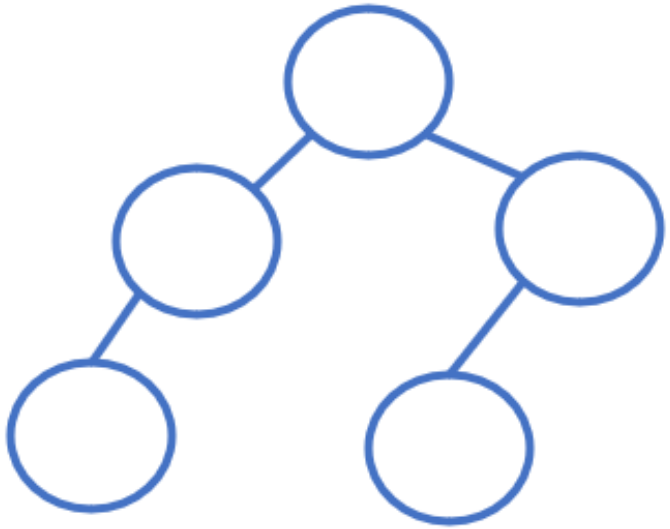
Stack & Queue



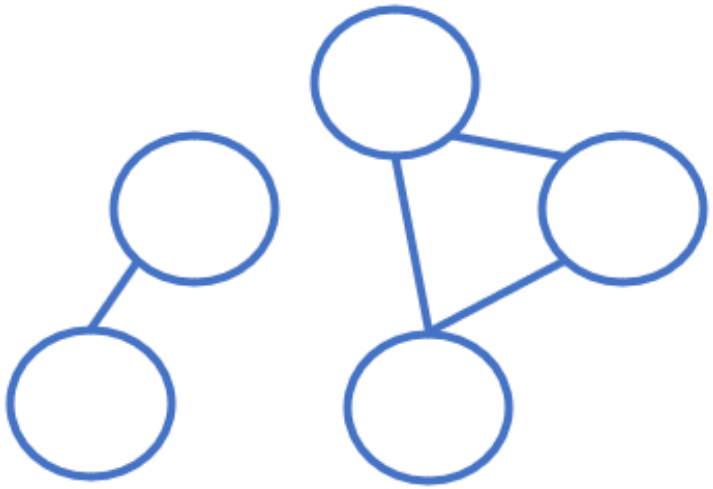
Linked List



Tree



Graph



Contiguous Structures

Contiguous

- "sharing a common border; touching."
- "next or together in sequence."
- Synonyms: adjacent, neighboring

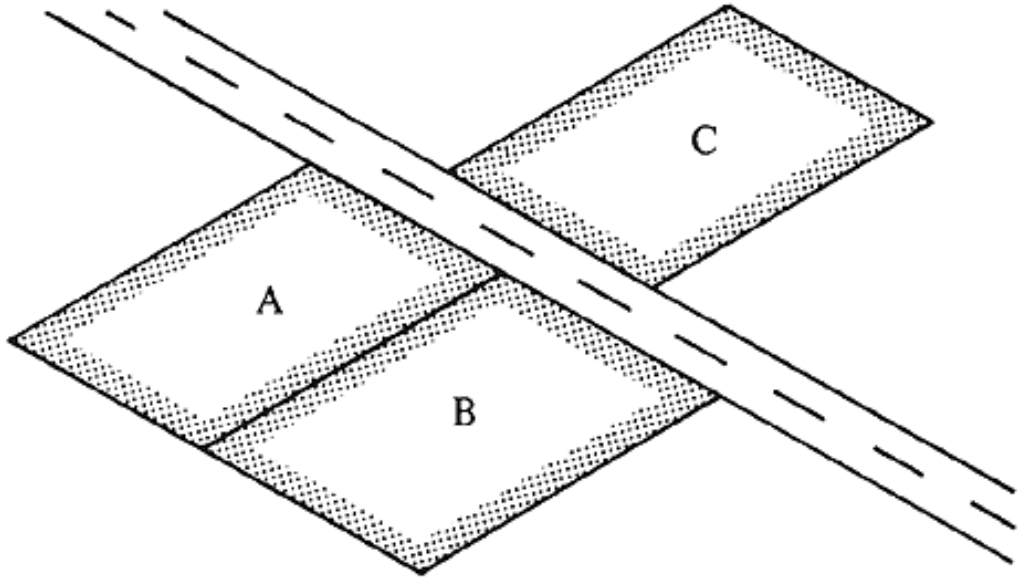
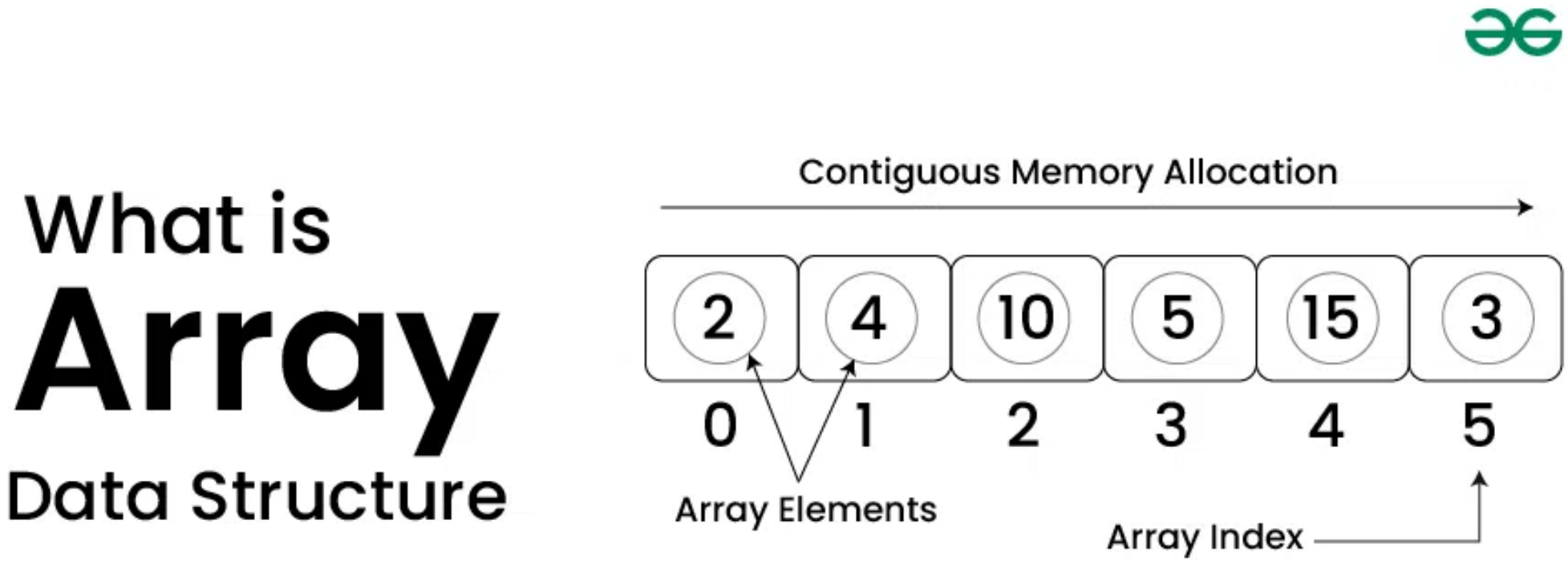


FIG. 42. CONTIGUOUS PROPERTY

Arrays

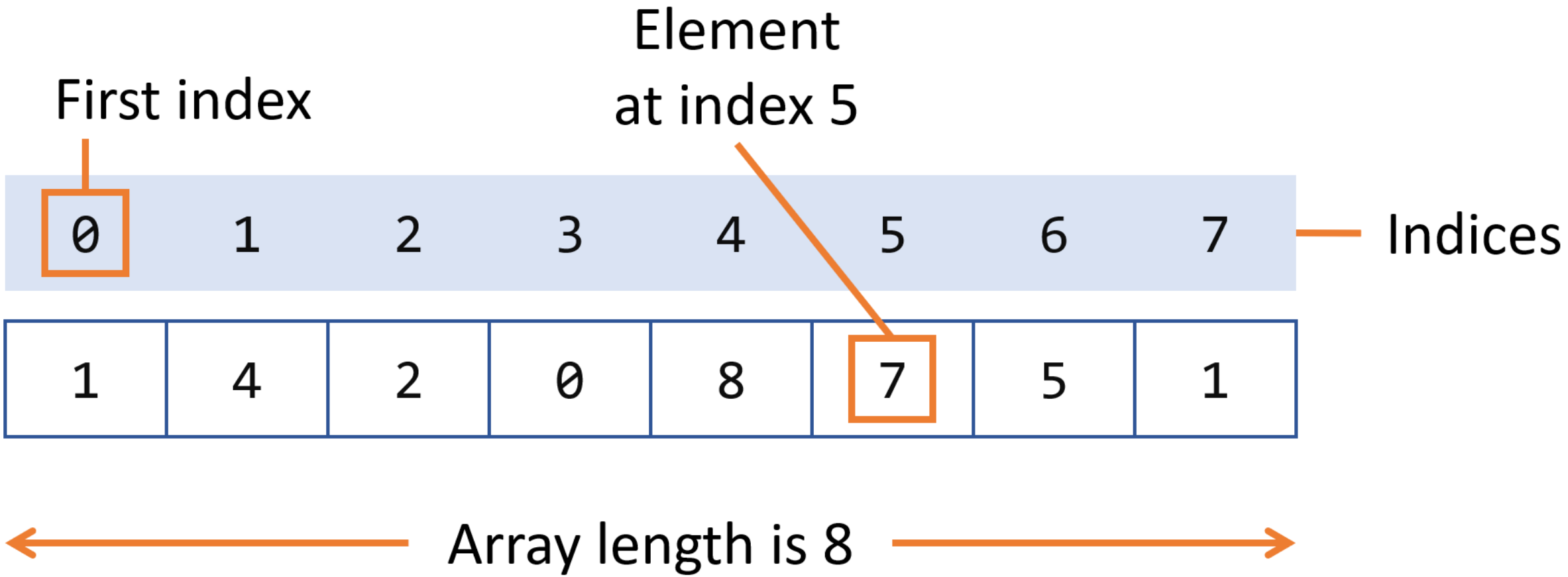

What is
Array
Data Structure
Contiguous Memory Allocation
2
4
10
5
15
3
0
1
2
3
4
5
Array Elements
Array Index

Arrays

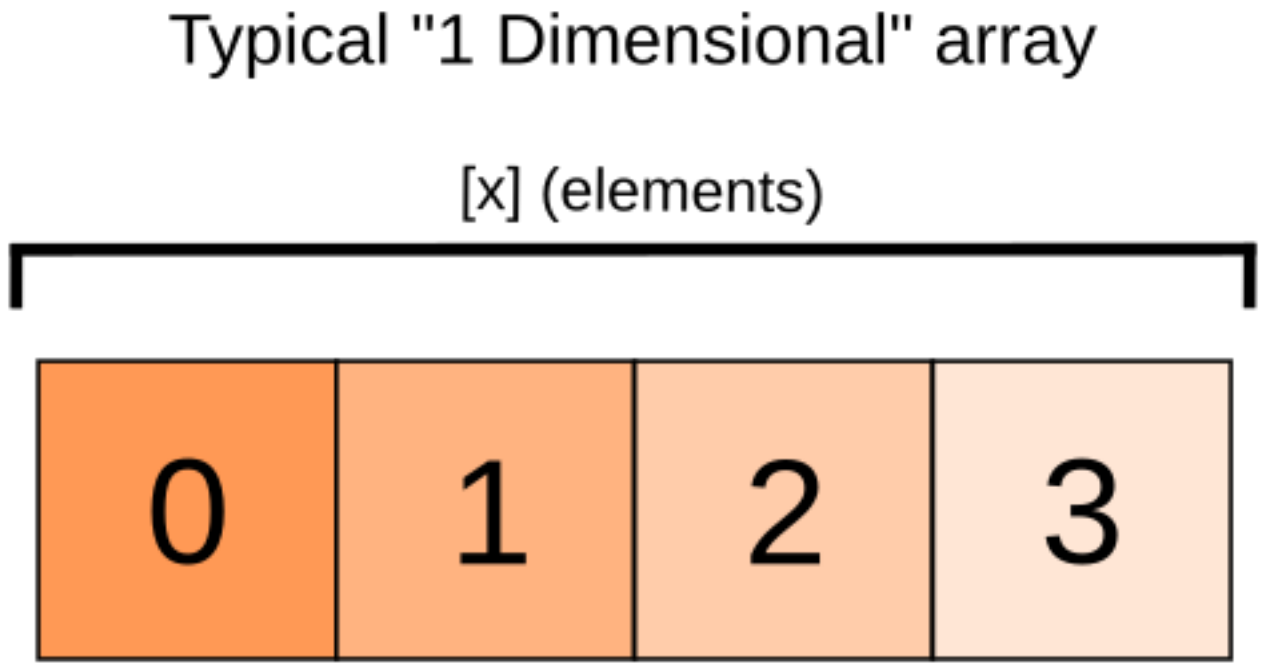
- Fixed-size
- Elements located by index/address
- *Typically* lives all together in memory

- Dynamic arrays allow for automatic re-sizing

Array Visual

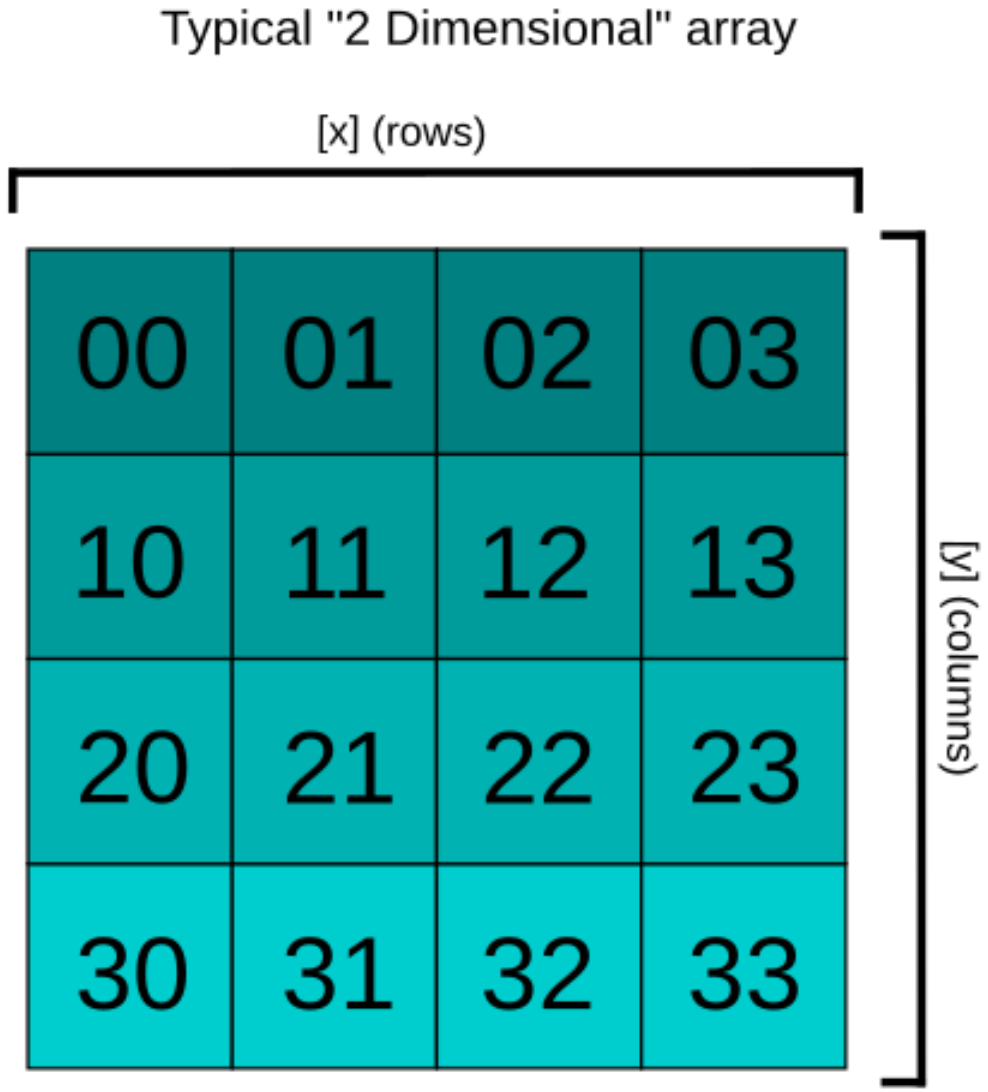


Array Visual



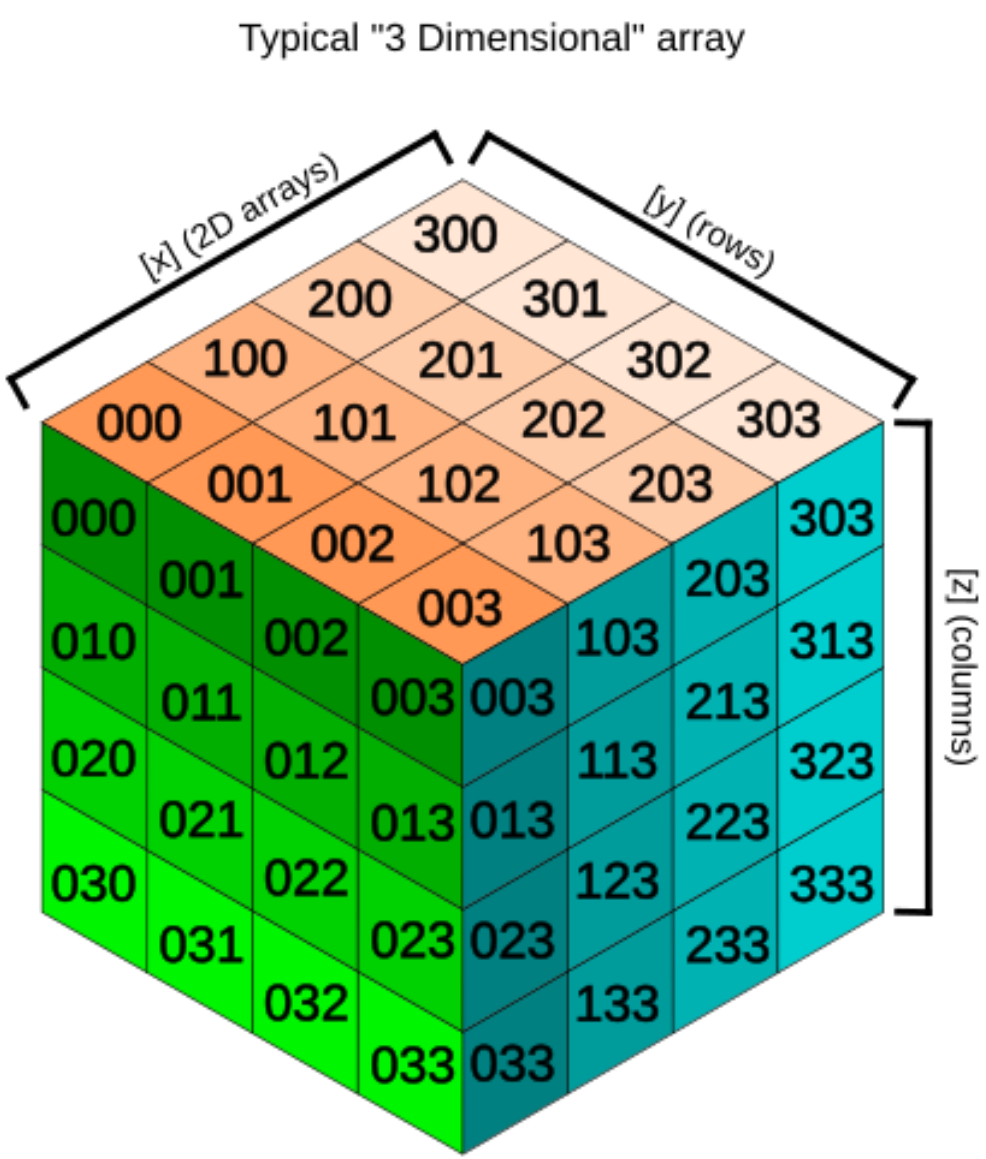
Element indexes are typically defined in the format *[x]*
[x] being the number of elements
For example: this array could be defined as *array[4]*

Array Visual



Element indexes are typically defined in the format *[x][y]*
[x] being the number of rows
[y] being the number of columns
For example: this array could be defined as *array[4][4]*

Array Visual



Element indexes are typically defined in the format *[x][y][z]*
[x] being the number of "2D" arrays
[y] being the number of rows
[z] being the number of columns
For example: this array could be defined as *array[4][4][4]*

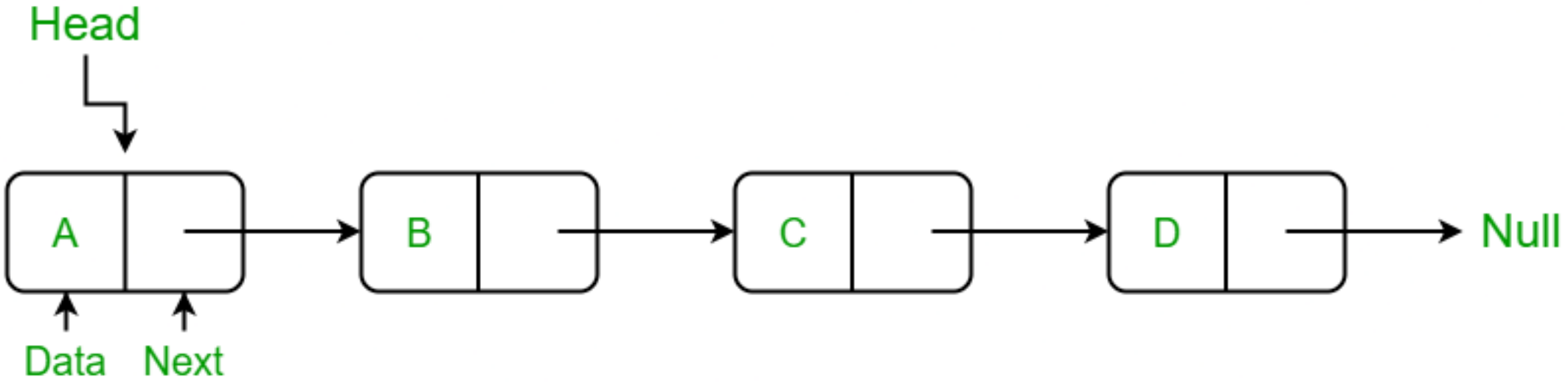
Array Operations

- Search – O(n)
- Access/Lookup – O(1)
- Delete – O(n)
- Others?

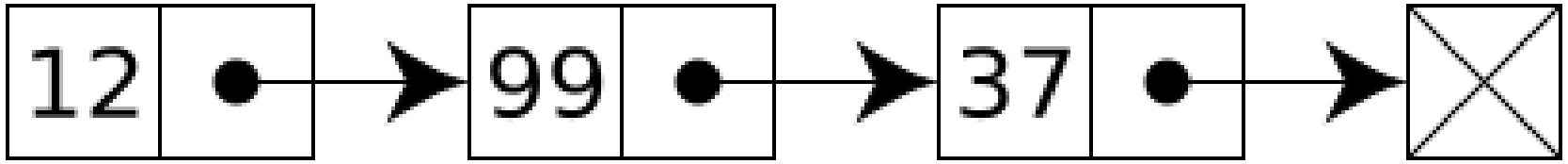
Linked Structures

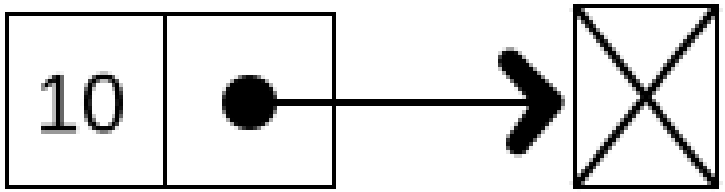
Linked Structures

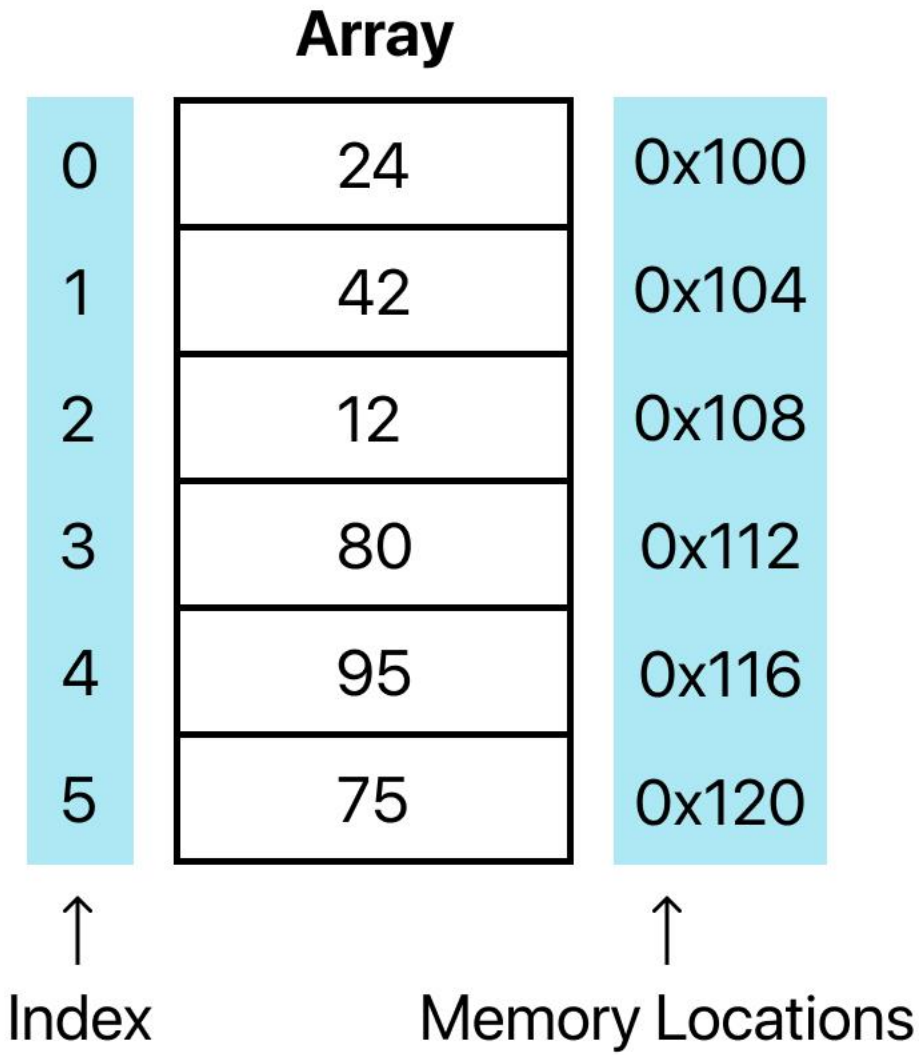
- Not contiguous in memory
- Joined by *pointers*

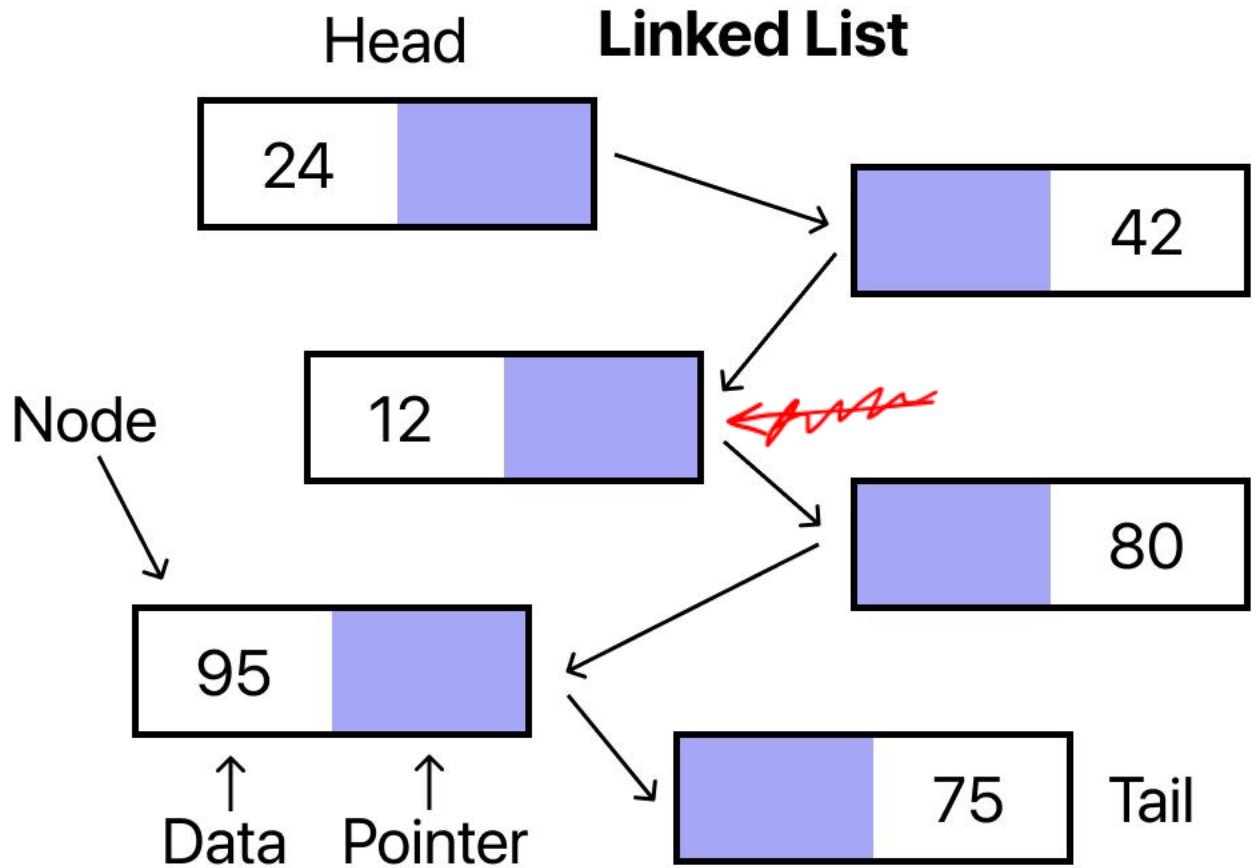


Linked Structures Visual

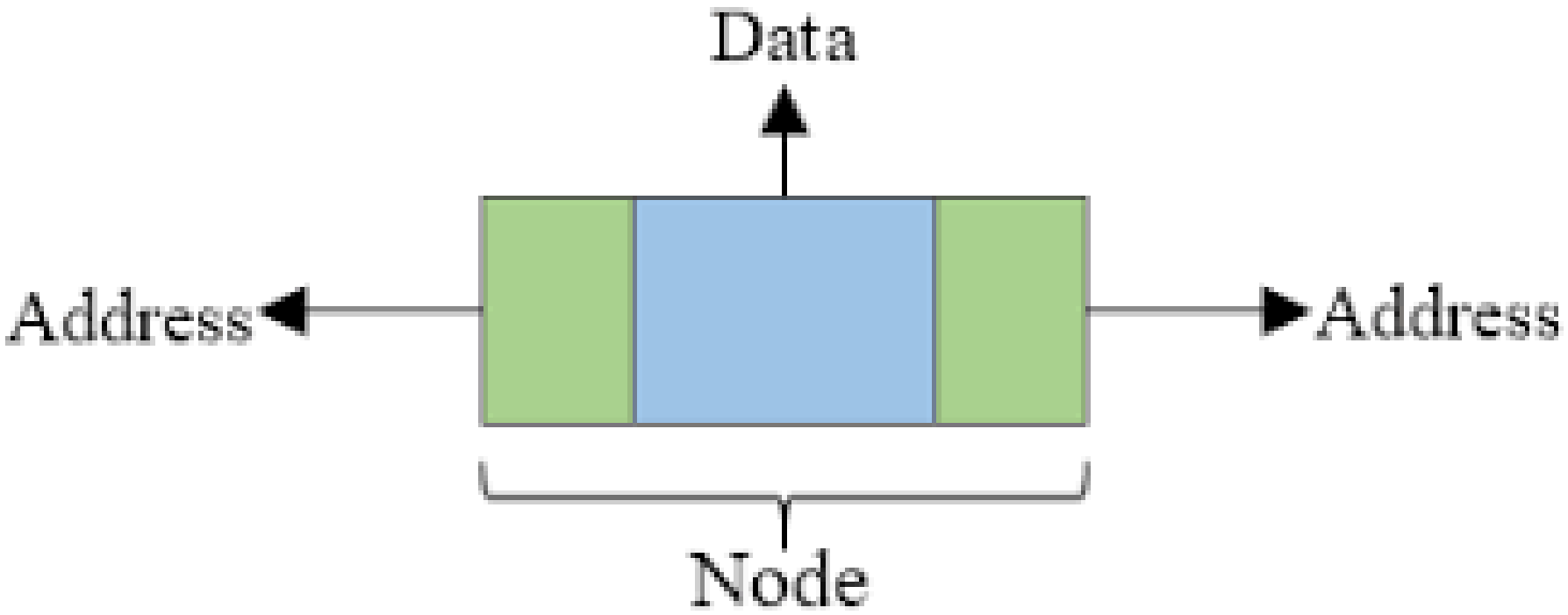





Array
0
1
2
3
4
5
24
42
12
80
95
75
0x100
0x104
0x108
0x112
0x116
0x120
Index
Memory Locations


Head
Linked List
24
42
12
80
95
75
Node
Data
Pointer
Tail

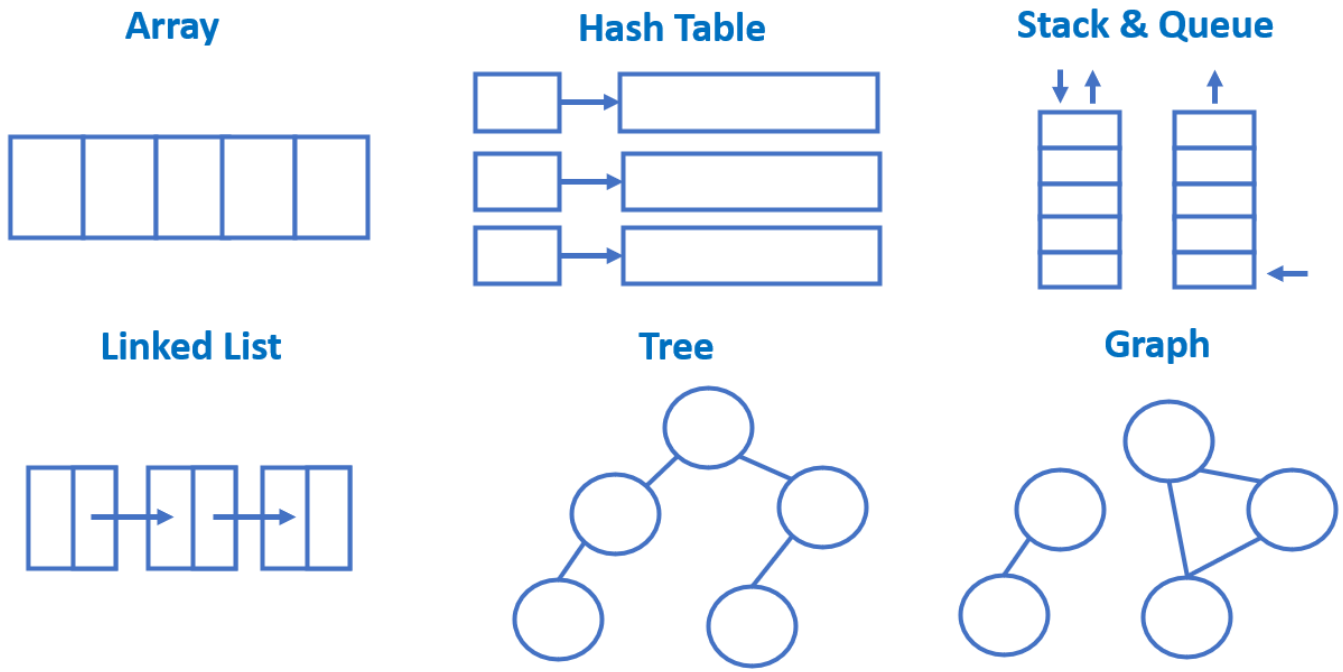
Linked Structure Visual



Linked List Operations

- Search – O(n)
- Access – O(n)
- Insert – O(n)
  - At beginning: O(1)
  - At end (if "tail"): O(1)
- Delete – O(n)
  - At beginning: O(1)
  - At end if tail & doubly linked: O(1)

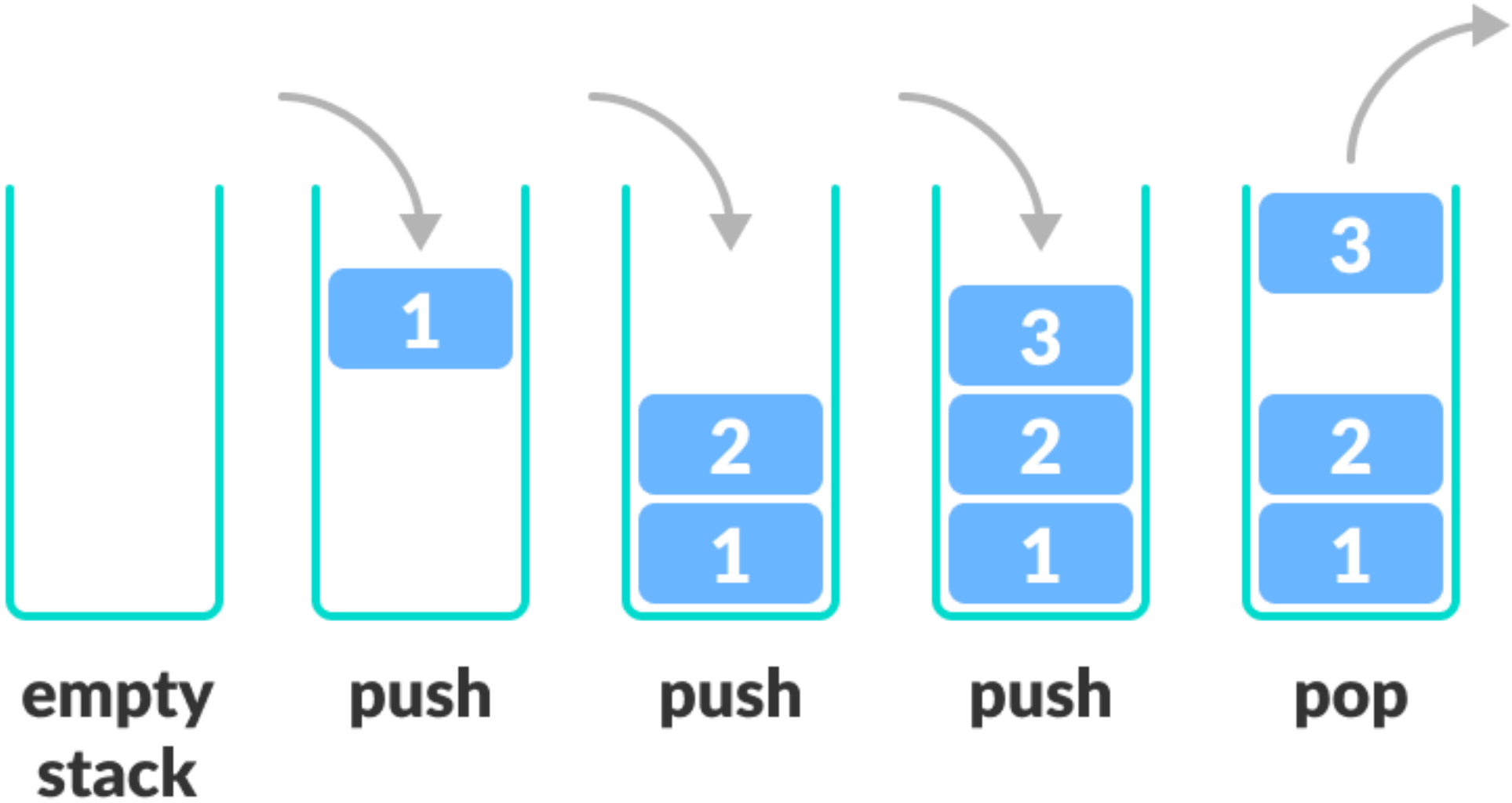
Stacks & Queues

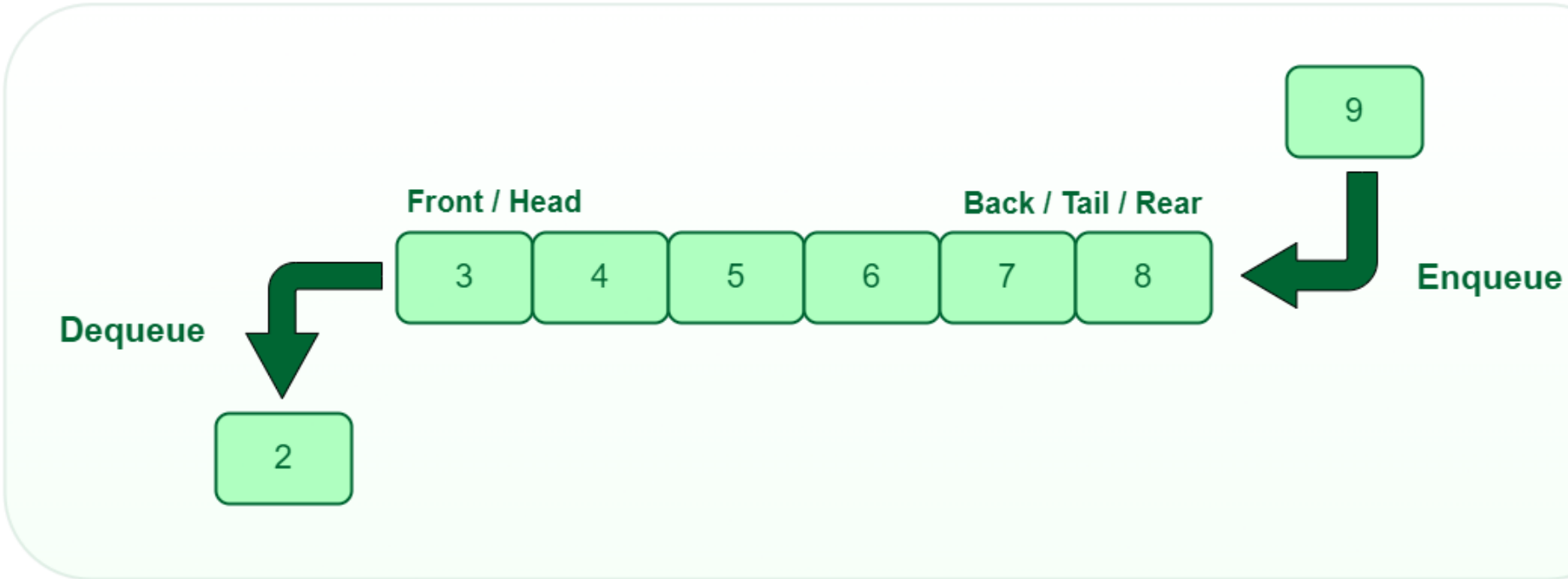


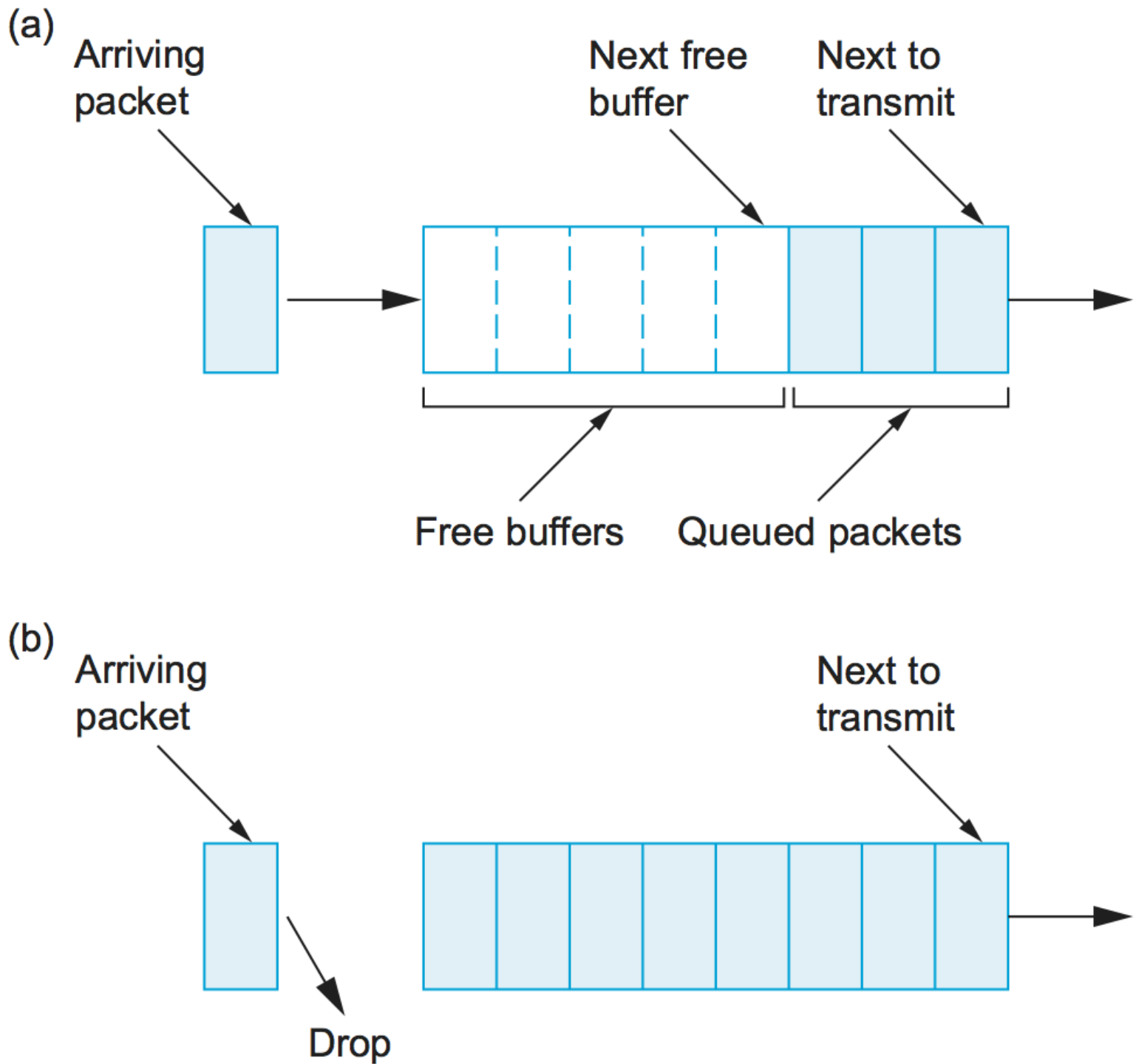
Stacks & Queues

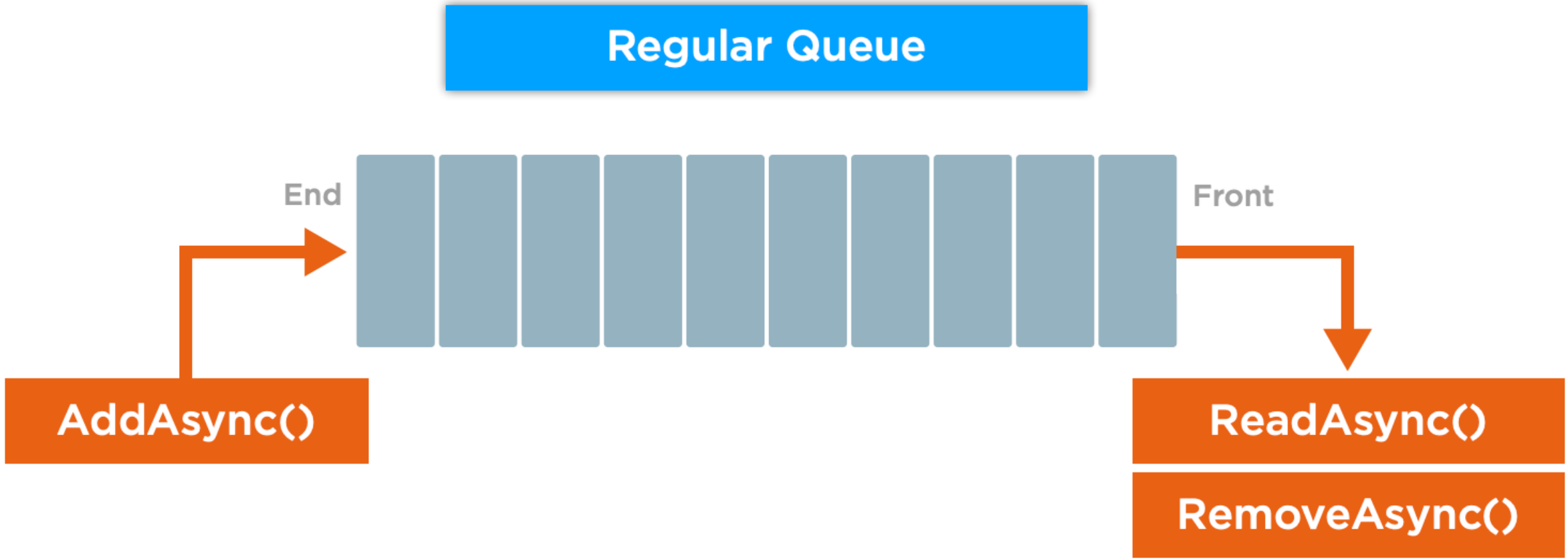
- Containers
  - Store data in a particular order
  - Retrieval depends on insertion order

- FIFO vs LIFO
  - F=First
  - L=Last
  - I=In
  - O=Out


1
2
1
3
2
1
3
2
1
empty stack
push
push
push
pop


9
Front / Head
Back / Tail / Rear
3
4
5
6
7
8
Enqueue
Dequeue
2


(a)
Arriving packet
Next free buffer
Next to transmit
Free buffers
Queued packets
(b)
Arriving packet
Next to transmit
Drop


Regular Queue
End
Front
AddAsync()
ReadAsync()
RemoveAsync()

Stack Operations

- Push (insert) – O(1)
- Pop (delete) – O(1)
- Access (top only, peek) – O(1)

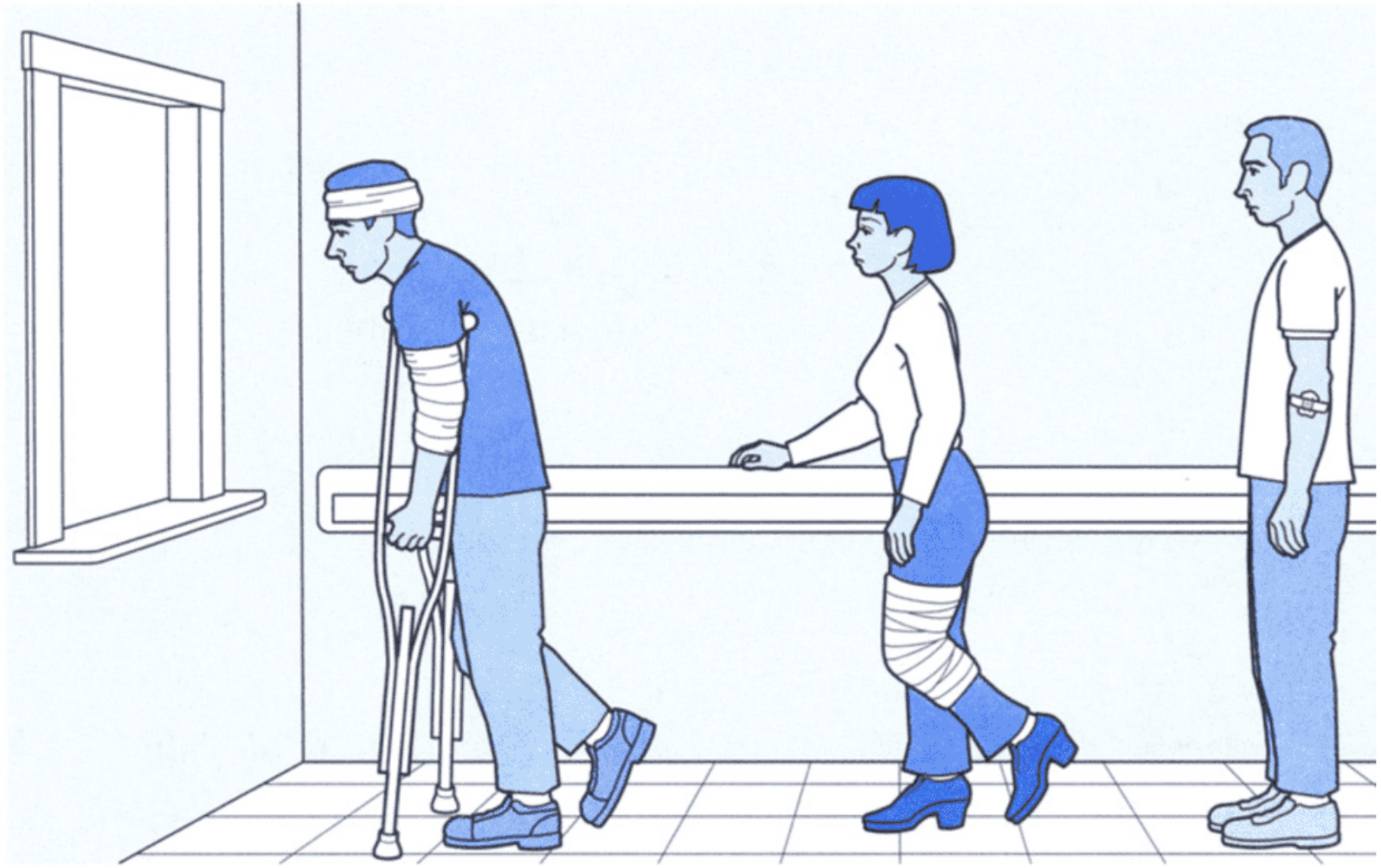
- "Not allowed":
  - Search
  - Lookup

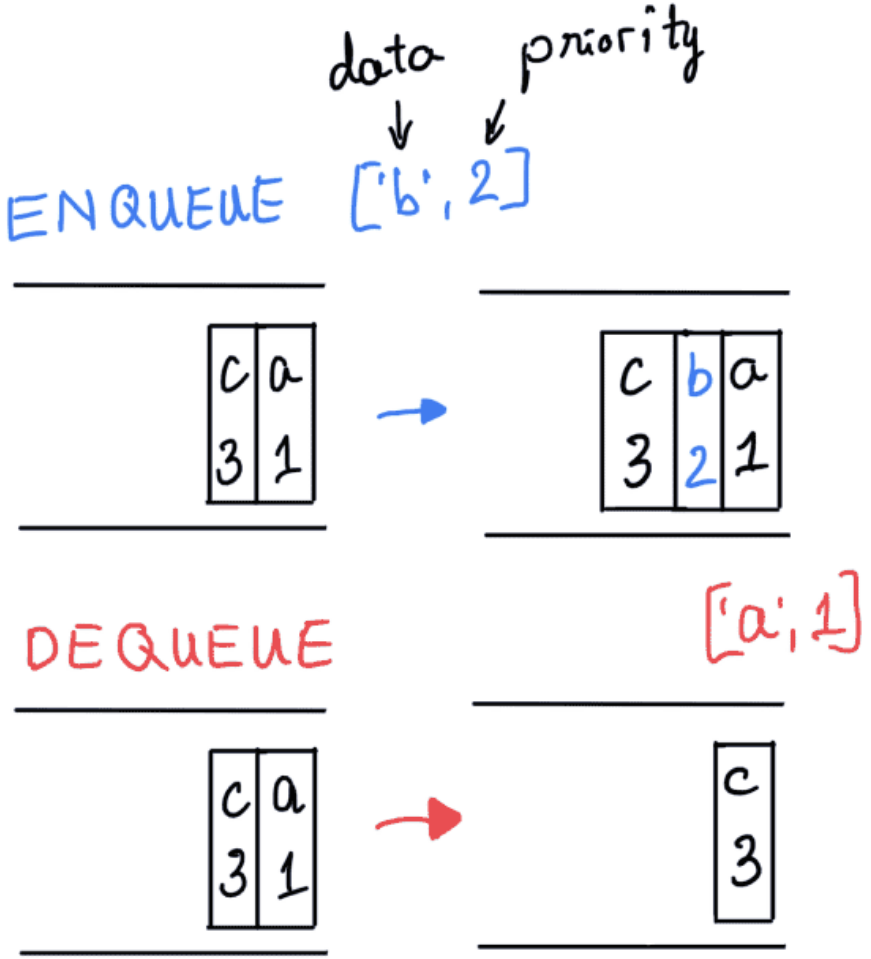
Queue Operations

- Enqueue (insert) – O(1)
- Dequeue (delete) – O(1)
- Double-ended also an option (deque)

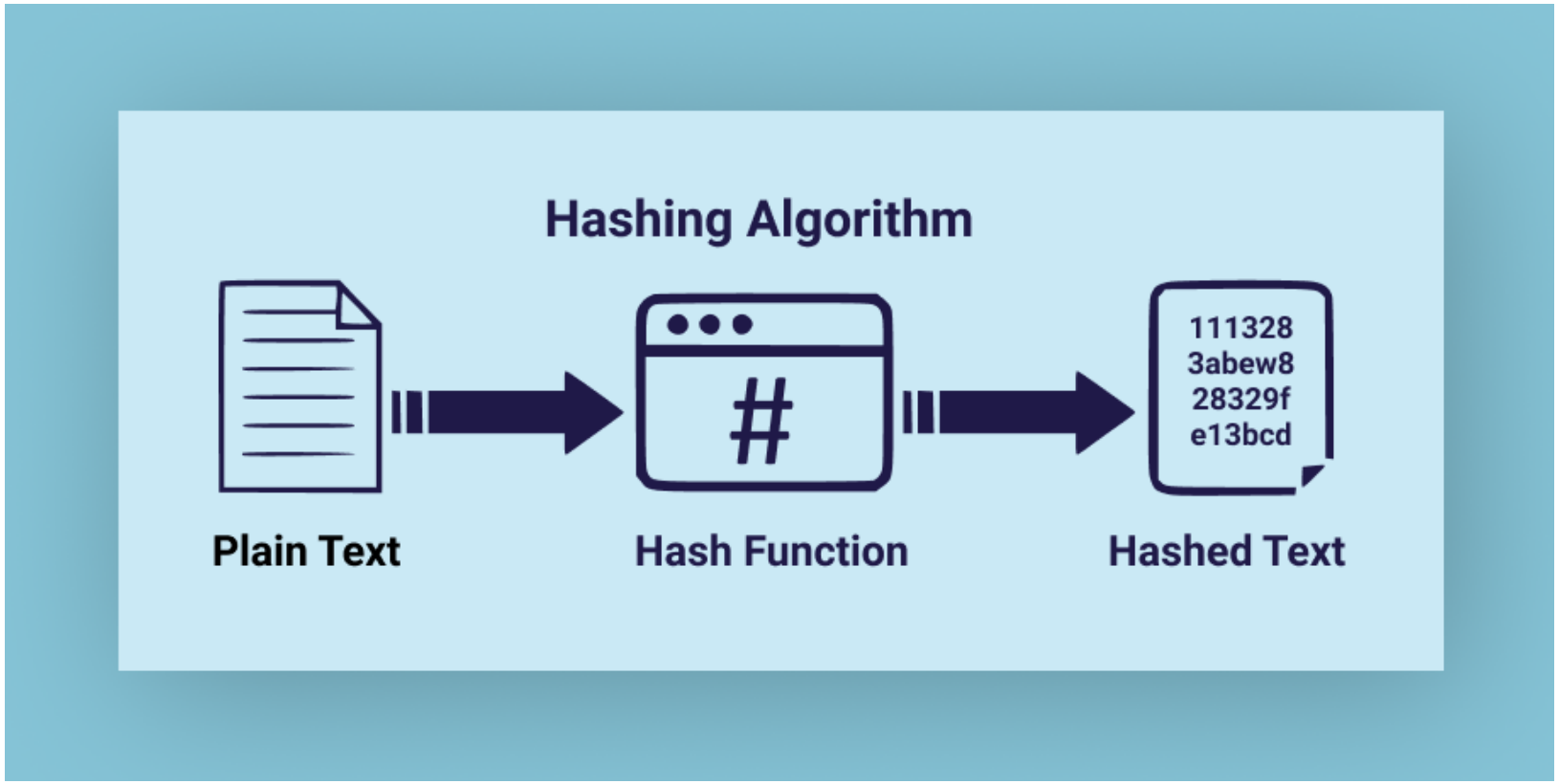
- Same "not allowed"s as Stacks

Priority Queue





Hashing/Dictionaries

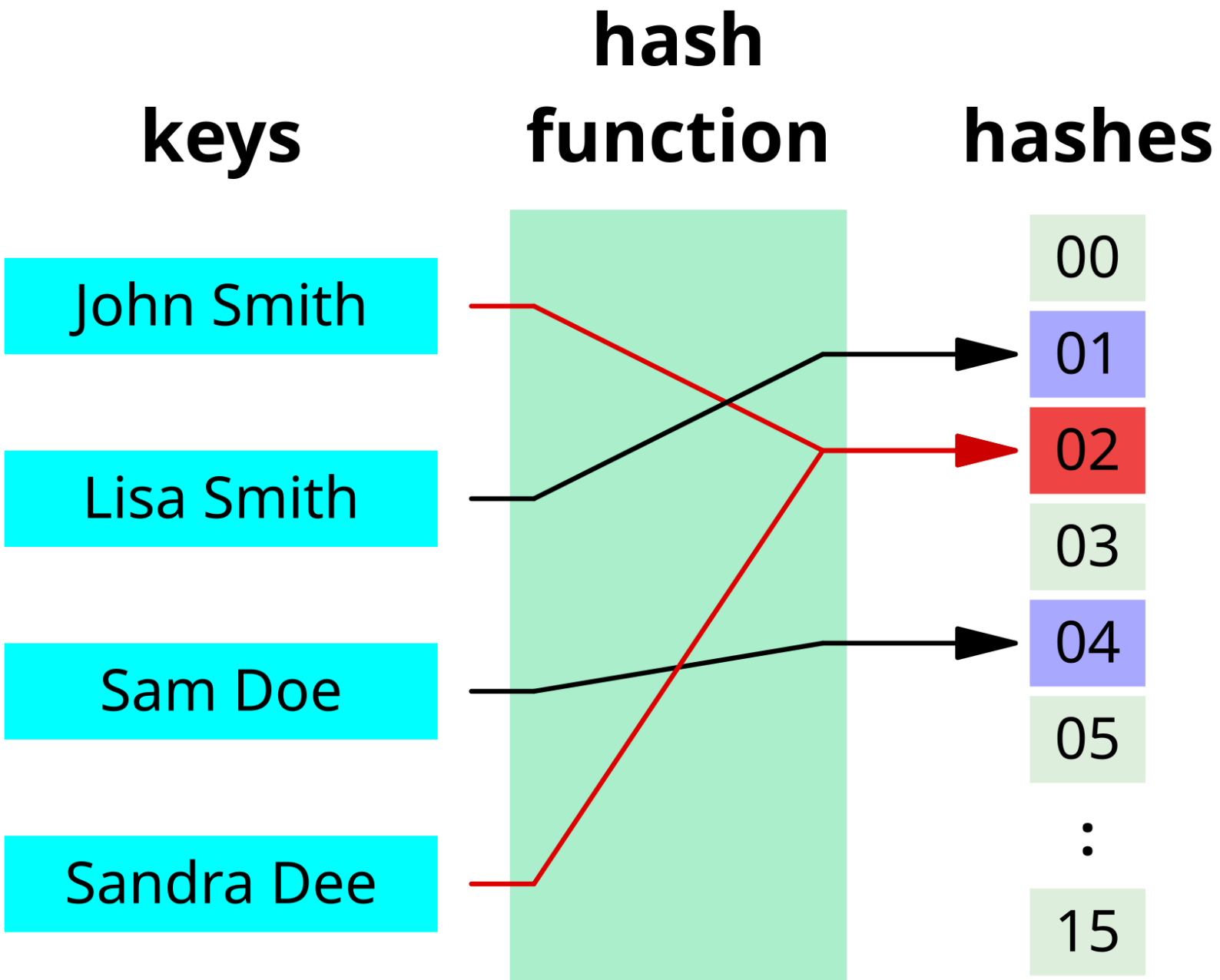

Hashing Algorithm
#
111328
3abew8
28329f
e13bcd
Plain Text
Hash Function
Hashed Text

Hashing

- A *hash* is a function that maps keys to an integer.
- Used in dictionaries, etc. to help you find it
- Also used for cryptography

- Have a function, then mod by m to stick it into spot m in the structure

- Collisions are an issue – what if two things have same hash?


keys
hash function
hashes
John Smith
Lisa Smith
Sam Doe
Sandra Dee
00
01
02
03
04
05
:
15

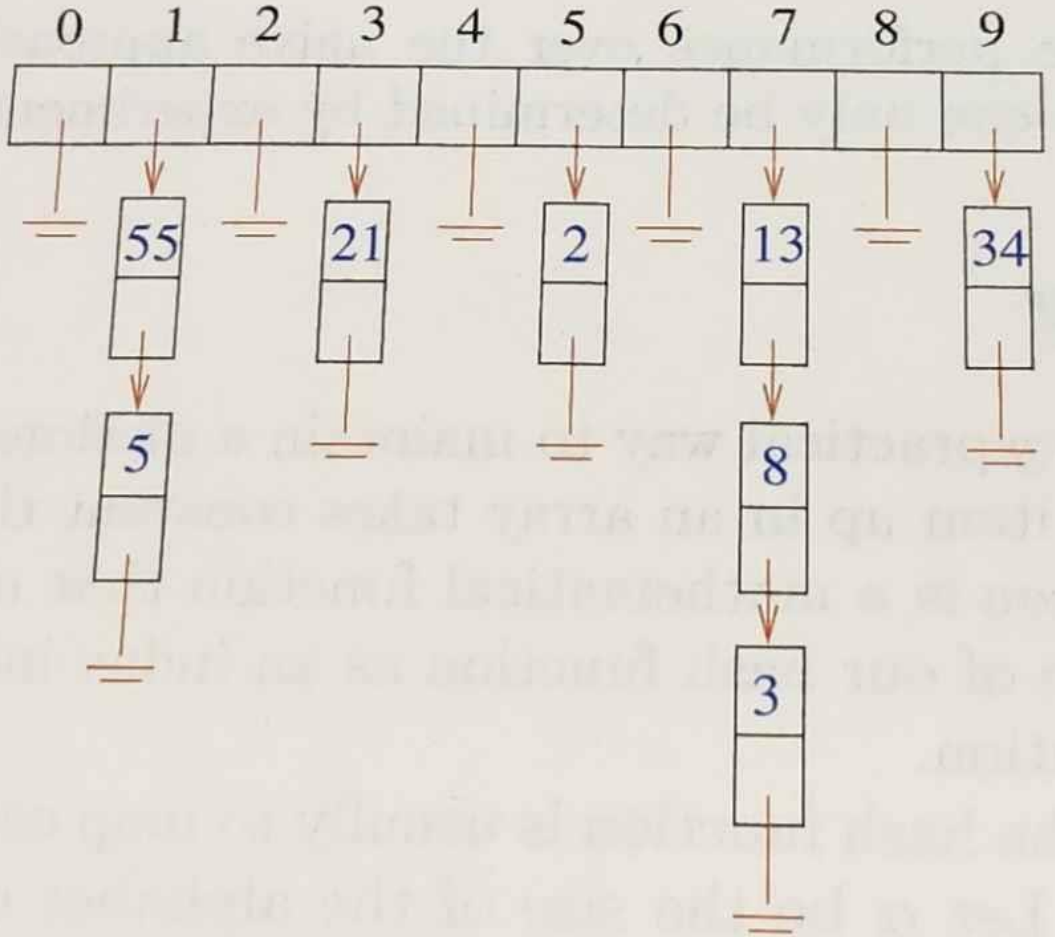


Figure 3.10: Collision resolution by chaining, after hashing the first eight Fibonacci numbers in increasing order, with hash function $H(x) = (2x + 1) \bmod 10$. *Insertions occur at the head of each list in this figure.*

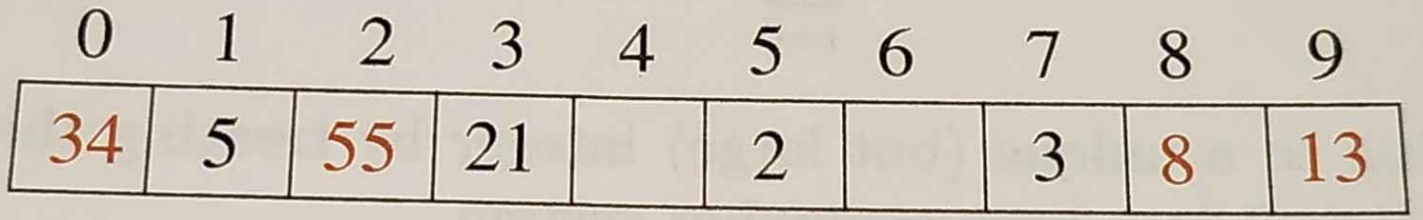
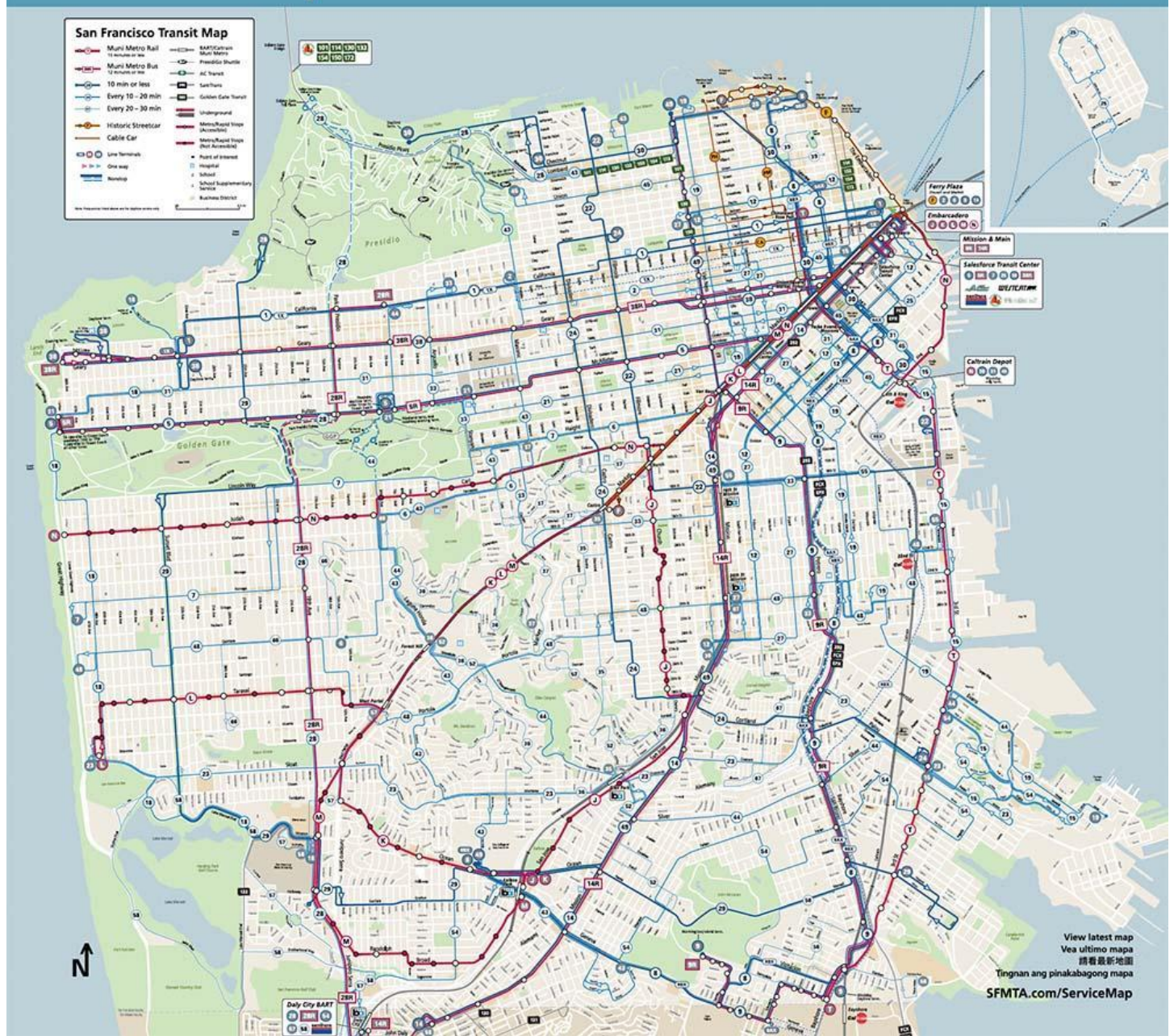


Figure 3.11: Collision resolution by open addressing and sequential probing, after inserting the first eight Fibonacci numbers in increasing order with $H(x) = (2x + 1) \mod 10$. *The red elements have been bumped to the first open slot after the desired location.*

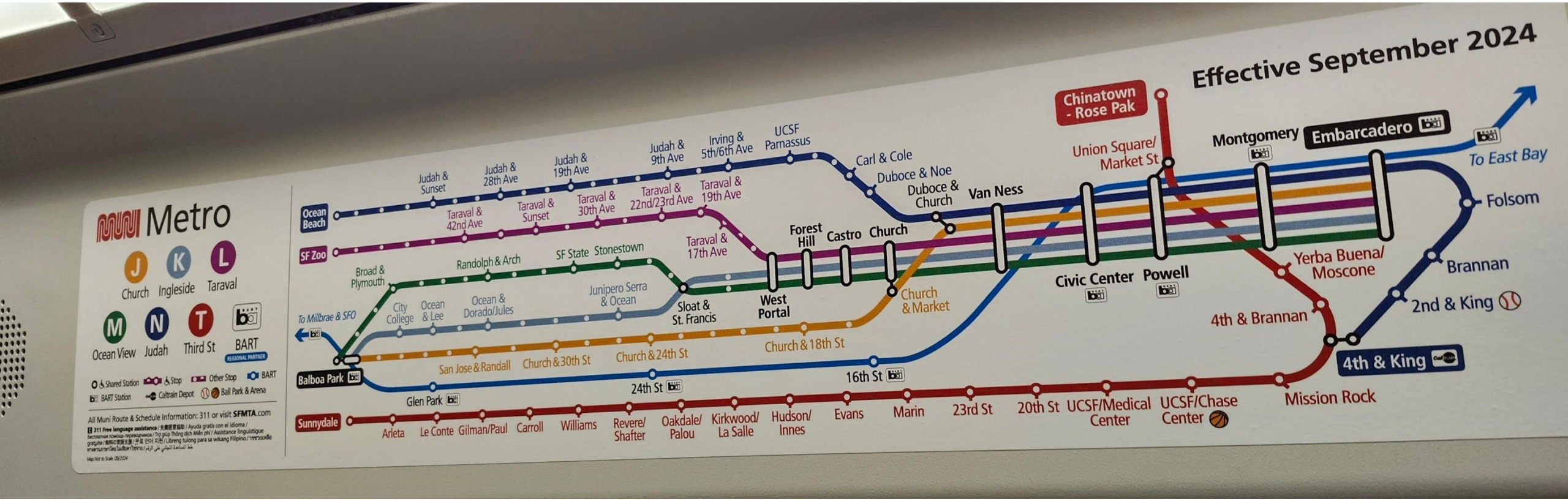
Graphs

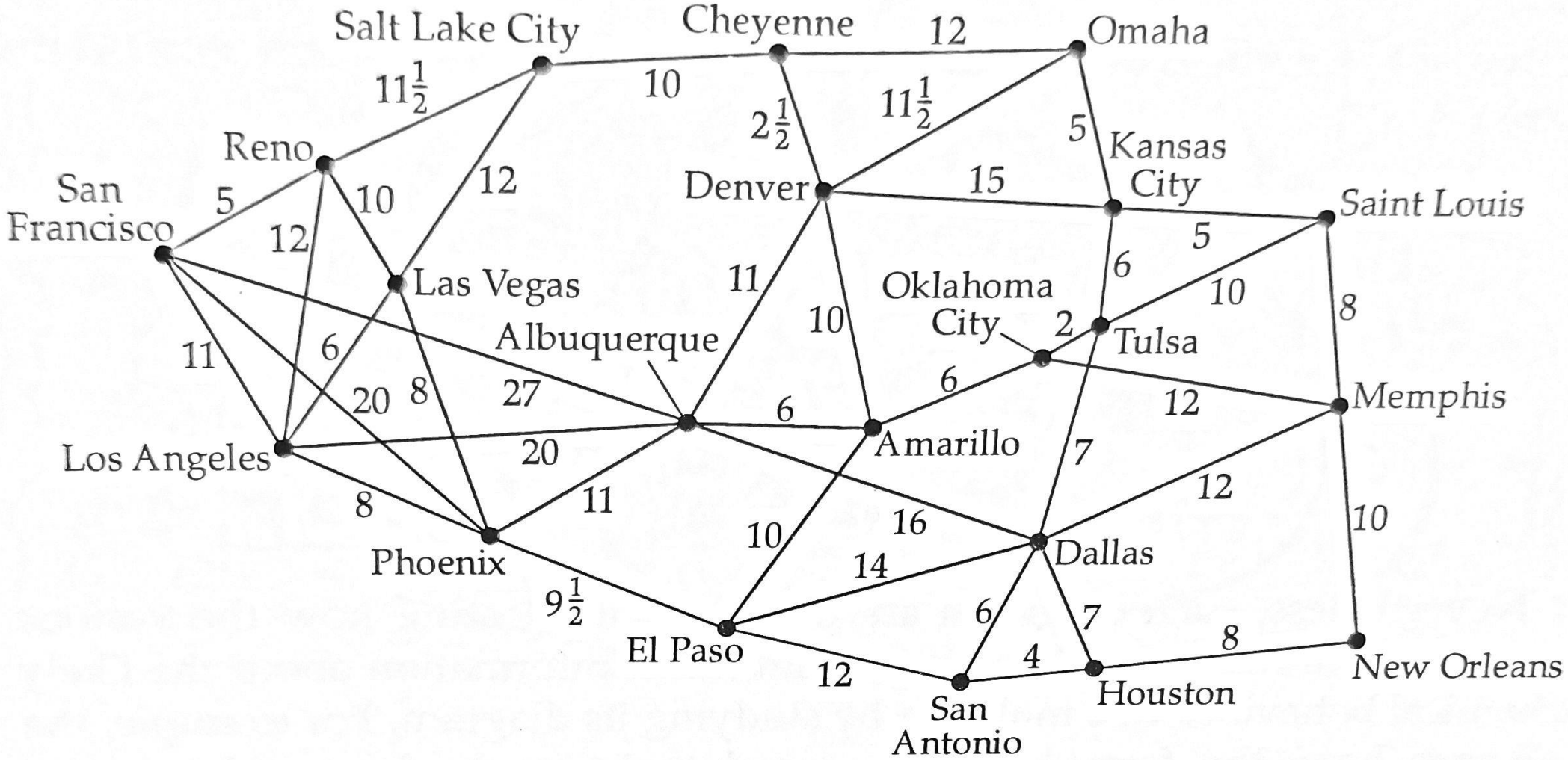
Muni Map

Effective September 2024

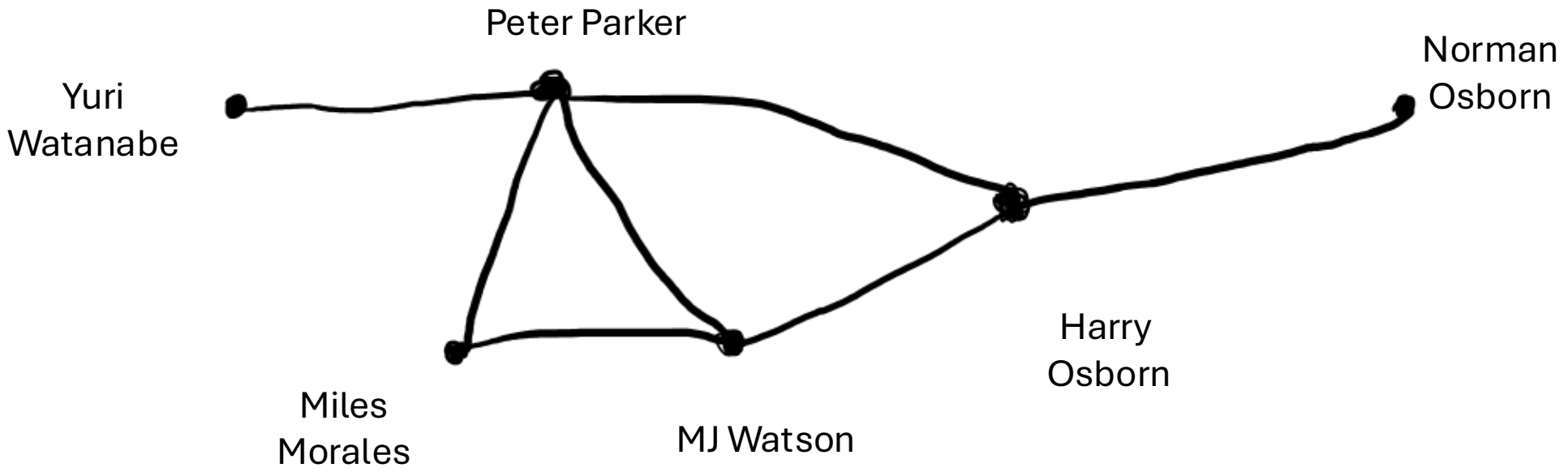



Muni Metro
Effective September 2024
Embarcadero
Montgomery
Chinatown - Rose Pak
Union Square/ Market St
Powell
Civic Center
Van Ness
Duboce & Church
Duboce & Noe
Carl & Cole
UCSF Parnassus
Irving & 5th/6th Ave
Judah & 9th Ave
Judah & 19th Ave
Judah & 28th Ave
Judah & Sunset
Ocean Beach
Church
Castro
Forest Hill
Taraval & 17th Ave
Taraval & 19th Ave
Taraval & 22nd/23rd Ave
Taraval & 30th Ave
Taraval & Sunset
Taraval & 42nd Ave
SF Zoo
Sloat & St. Francis
West Portal
Stonestown
SF State
Randolph & Arch
Junipero Serra & Ocean
Dorado & Jules
Ocean & Lee
City College
Broad & Plymouth
Church & Market
Church & 18th St
Church & 24th St
Church & 30th St
Glen Park
San Jose & Randall
Balboa Park
To Millbrae & SFO
16th St/ Mission
24th St/ Mission
To East Bay
Folsom
Brannan
Yerba Buena/ Moscone
2nd & King
4th & Brannan
4th & King
Mission Rock
UCSF/Chase Center
UCSF Medical Center
20th St
23rd St
Marin St
Evans
Hudson/ Innes
Kirkwood/ La Salle
Oakdale/ Palou
Revere/ Shafter
Williams
Carroll
Gilman/Paul
Le Conte
Arleta
Sunnydale
J Church
K Ingleside
L Taraval
M Ocean View
N Judah
T Third St
BART
REGIONAL PARTNER
Shared Station
Stop
Other Stop
BART Station & Line
Caltrain Depot
Ball Park & Arena


Effective September 2024
Metro
J Church
K Ingleside
L Taraval
M Ocean View
N Judah
T Third St
BART
REGIONAL PARTNER
Shared Station
Stop
Other Stop
BART
BART Station
Caltrain Depot
Ball Park & Arena
All Muni Route & Schedule Information: 311 or visit SFMTA.com
311 Free language assistance
Map Not to Scale 09/2024
Ocean Beach
SF Zoo
Balboa Park
To Millbrae & SFO
Sunnydale
Judah & Sunset
Judah & 28th Ave
Judah & 19th Ave
Judah & 9th Ave
Irving & 5th/6th Ave
UCSF Parnassus
Carl & Cole
Duboce & Noe
Duboce & Church
Van Ness
Taraval & 42nd Ave
Taraval & Sunset
Taraval & 30th Ave
Taraval & 22nd/23rd Ave
Taraval & 19th Ave
Taraval & 17th Ave
Broad & Plymouth
Randolph & Arch
SF State
Stonestown
City College
Ocean & Lee
Ocean & Dorado/Jules
Junipero Serra & Ocean
Sloat & St. Francis
Forest Hill
Castro
Church
West Portal
Church & Market
San Jose & Randall
Church & 30th St
Church & 24th St
Church & 18th St
Glen Park
24th St
16th St
Chinatown - Rose Pak
Union Square/ Market St
Montgomery
Embarcadero
Civic Center
Powell
Yerba Buena/ Moscone
4th & Brannan
Mission Rock
4th & King
To East Bay
Folsom
Brannan
2nd & King
Arleta
Le Conte
Gilman/Paul
Carroll
Williams
Revere/ Shafter
Oakdale/ Palou
Kirkwood/ La Salle
Hudson/ Innes
Evans
Marin
23rd St
20th St
UCSF/Medical Center
UCSF/Chase Center


Salt Lake City
Cheyenne
12
Omaha
$11\frac{1}{2}$
10
$2\frac{1}{2}$
$11\frac{1}{2}$
5
Reno
Kansas
City
San
Francisco
5
10
12
Denver
15
Saint Louis
12
5
6
Las Vegas
11
Oklahoma
City
10
2
Tulsa
8
Albuquerque
10
11
6
6
20
8
27
6
12
Memphis
Amarillo
Los Angeles
20
7
12
8
11
16
10
10
Dallas
Phoenix
14
$9\frac{1}{2}$
6
7
8
El Paso
4
12
New Orleans
San
Antonio
Houston

"The Friendship Graph" – Spider-Man edition



Vertices =

Edges =

Questions we can ask

- If I'm friends with you, are you friends with me?
  - Directed/undirected
- How close are we?
  - Weight
- Am I friends with myself?
  - Simple?
- How many friends do you have?
  - Degree/valence
- Are you an individual, or part of the crowd?
  - Labeled vs Unlabeled

Questions we can ask

- Do they live close by?
  - Embedded graph/data on edges
- Oh, you know them too?
  - Erdős number / Bacon number
  - Six degrees of separation

How should we represent a graph?

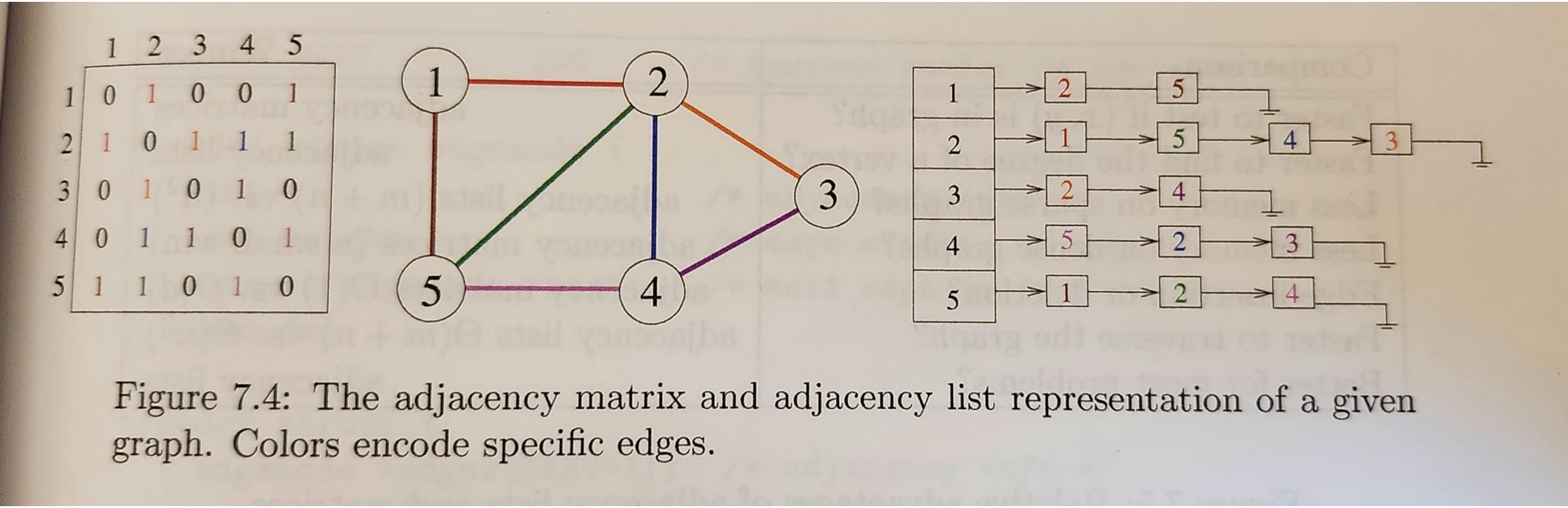


Figure 7.4: The adjacency matrix and adjacency list representation of a given graph. Colors encode specific edges.

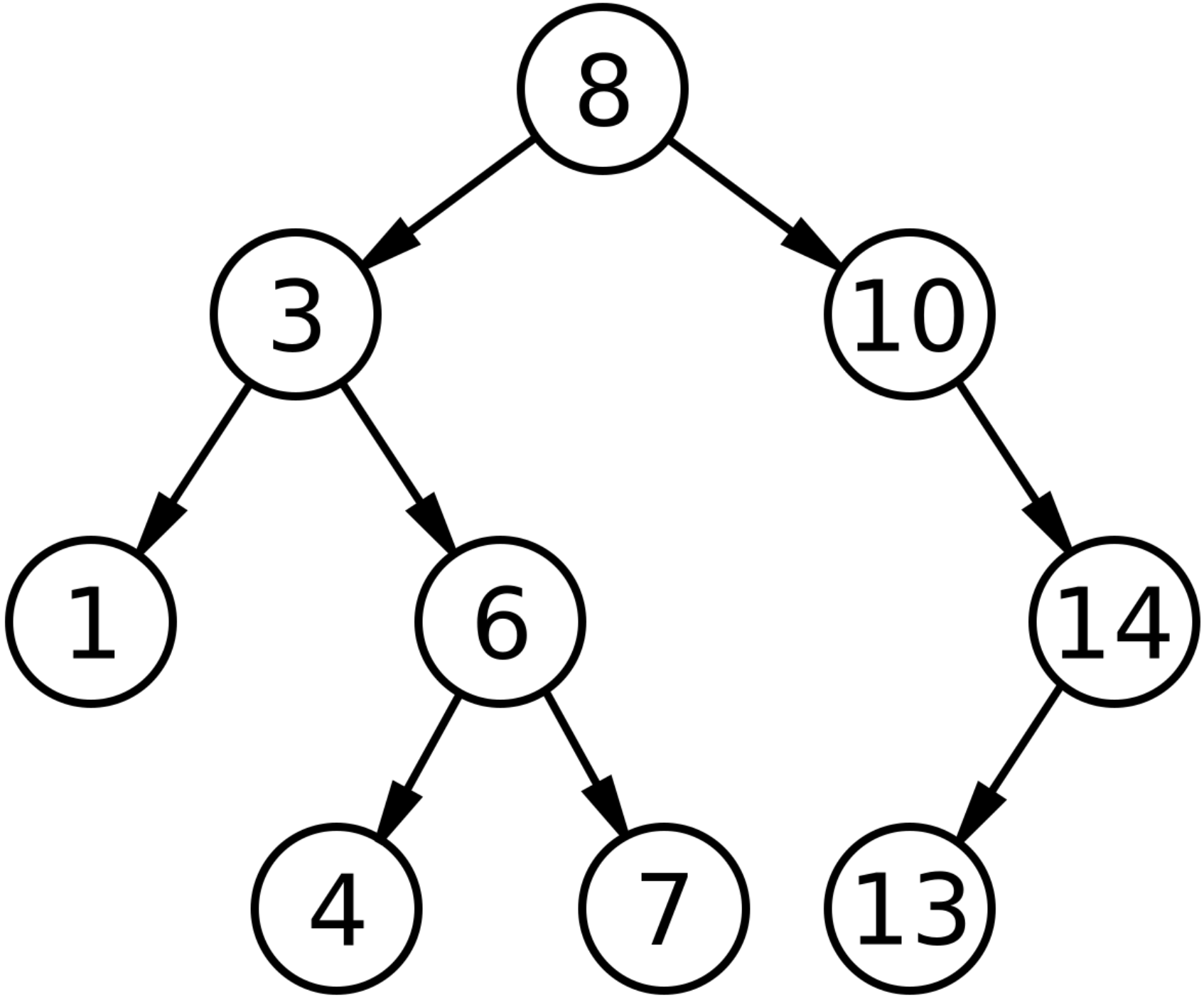
(Not) coming back around

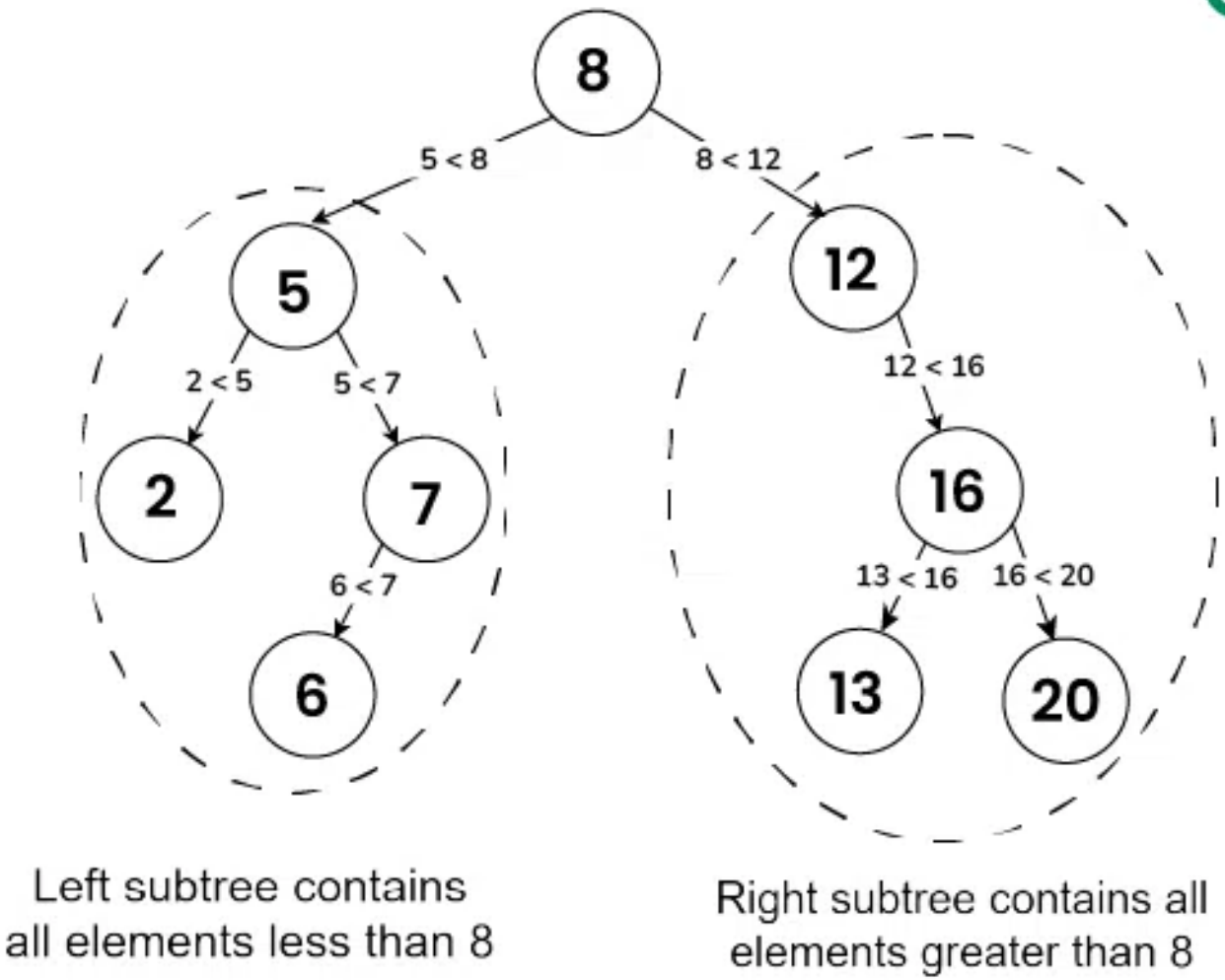
- A **tree** is a graph with no **cycles**.

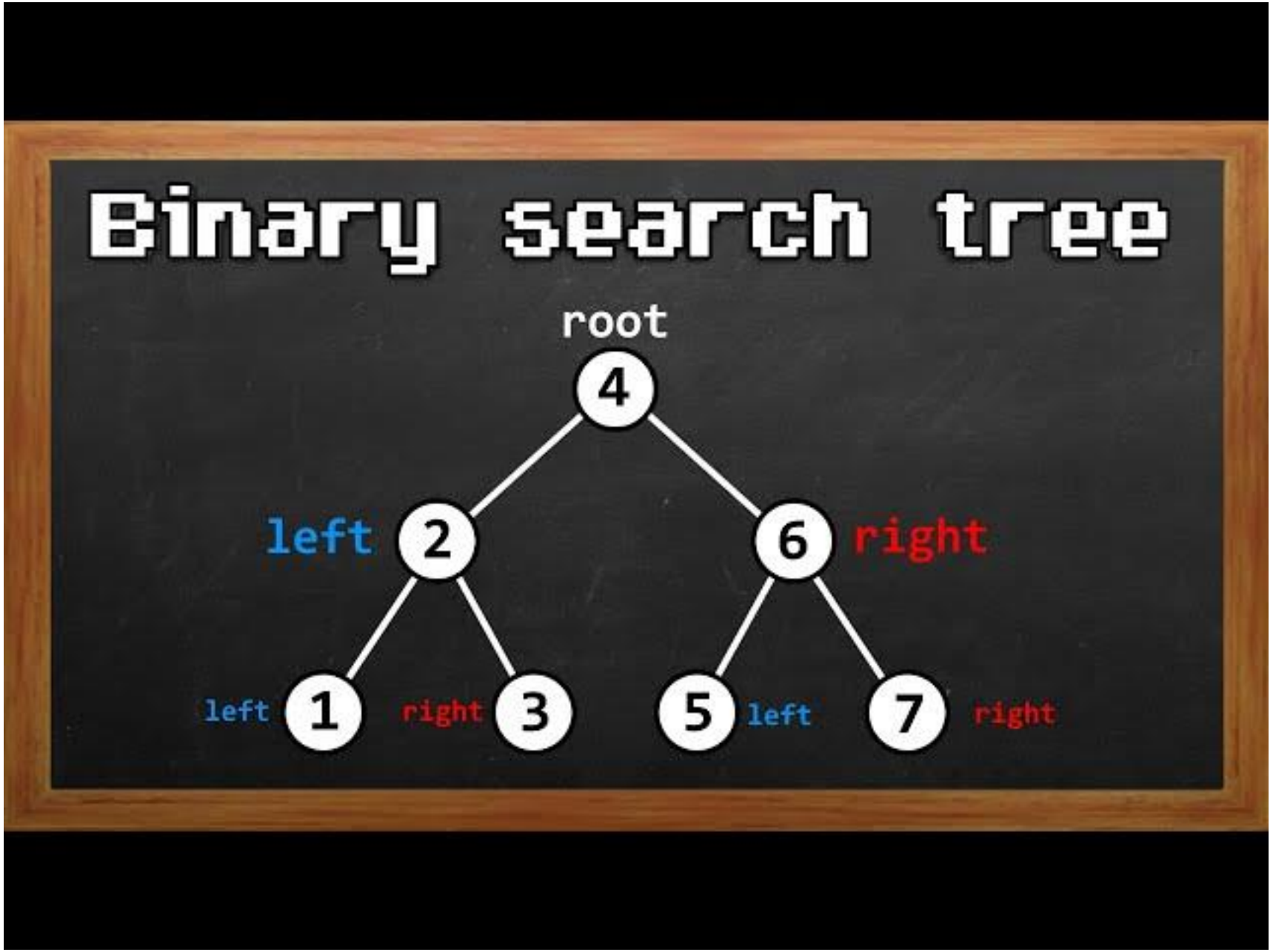
Binary Search Trees

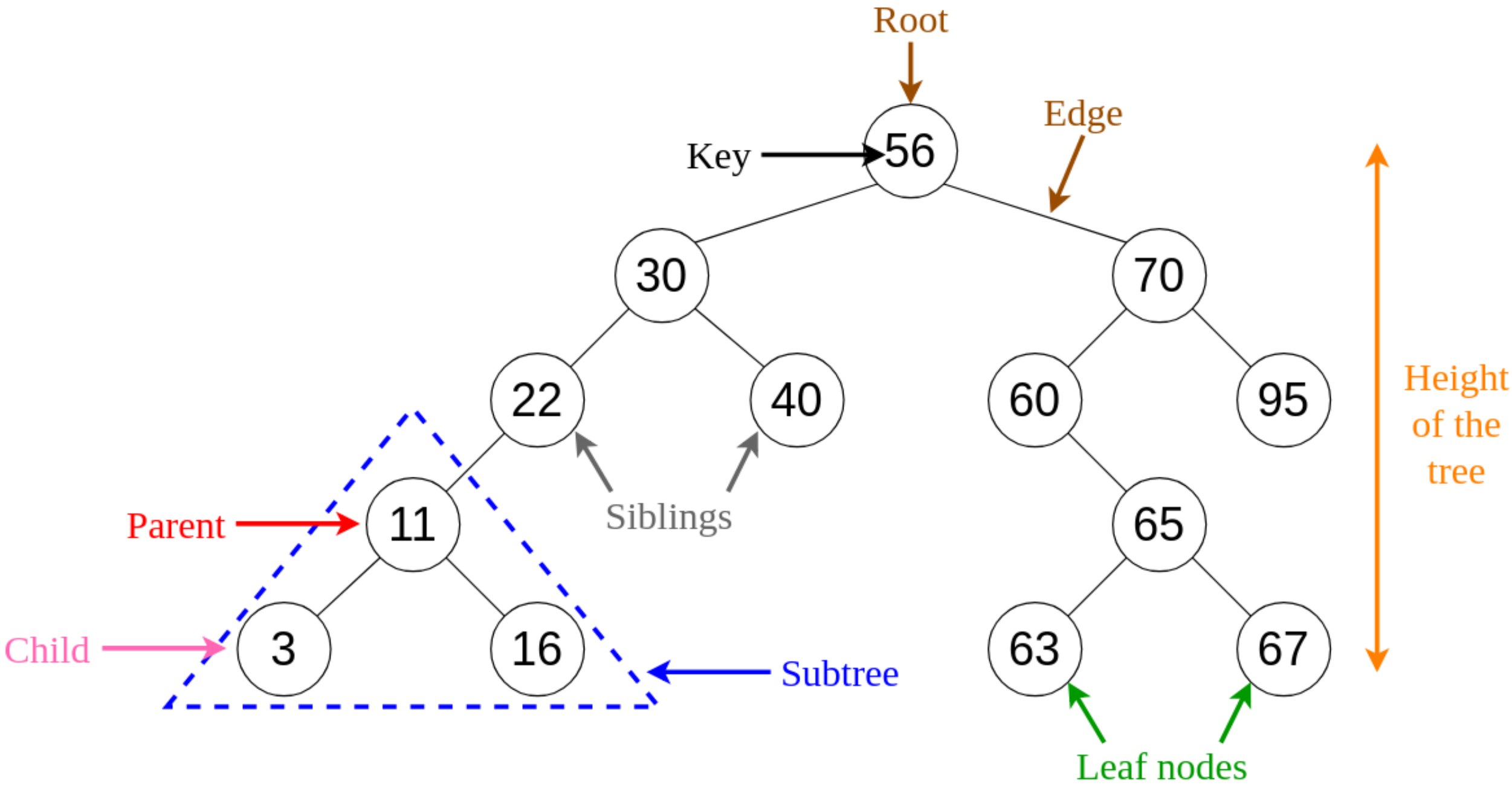
- Fast search
- Flexible update

- Root – holds the top of tree (or is empty)
- Child(ren) – node(s) below this node
- Subtree – the BST starting from this node
- BST property is: for any node x,
  - All left children of x are < x.
  - All right children of x are > x.


8
3
10
1
6
14
4
7
13

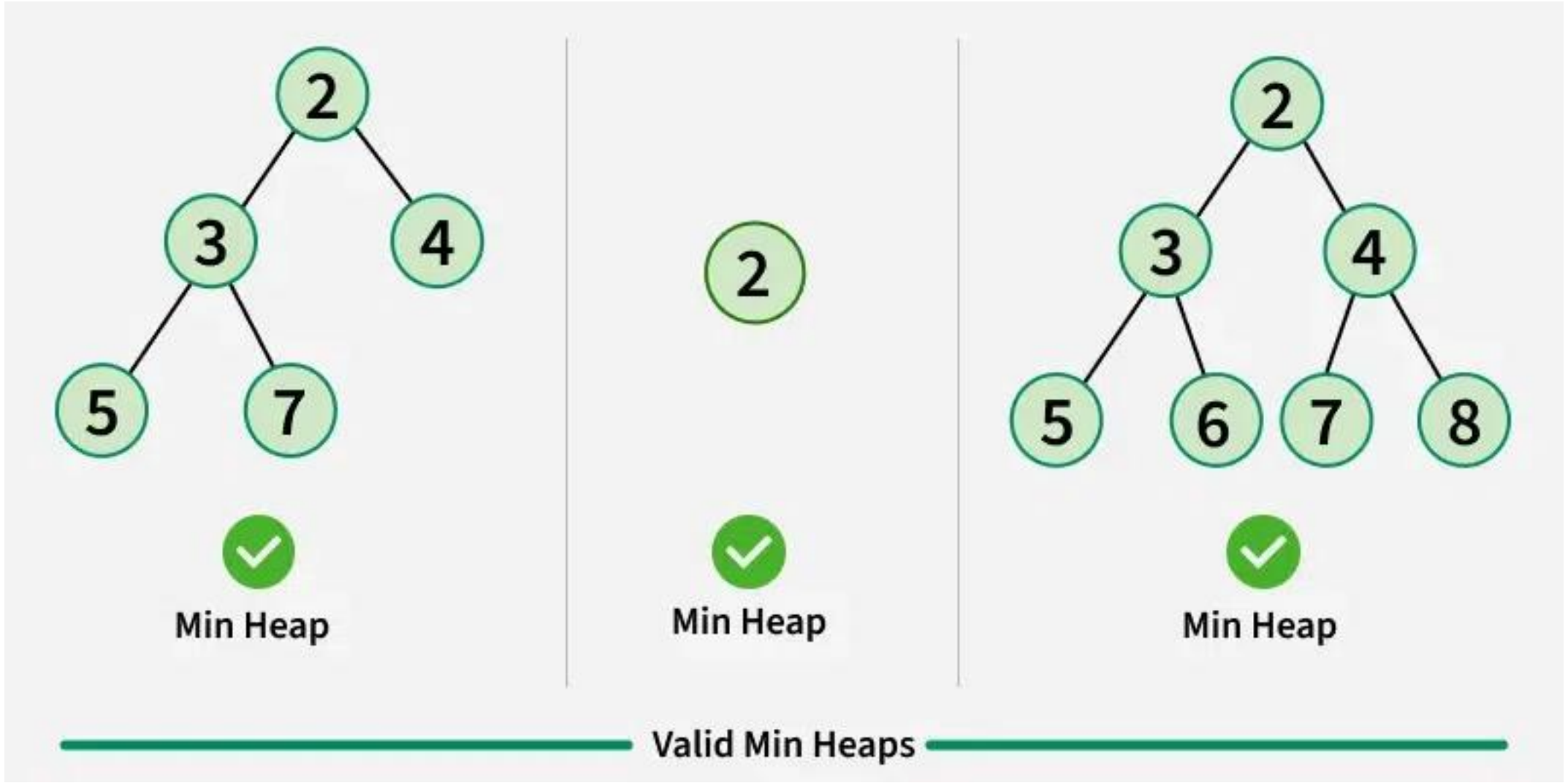

8
5 < 8
8 < 12
5
12
2 < 5
5 < 7
12 < 16
2
7
16
6 < 7
13 < 16
16 < 20
6
13
20
Left subtree contains all elements less than 8
Right subtree contains all elements greater than 8

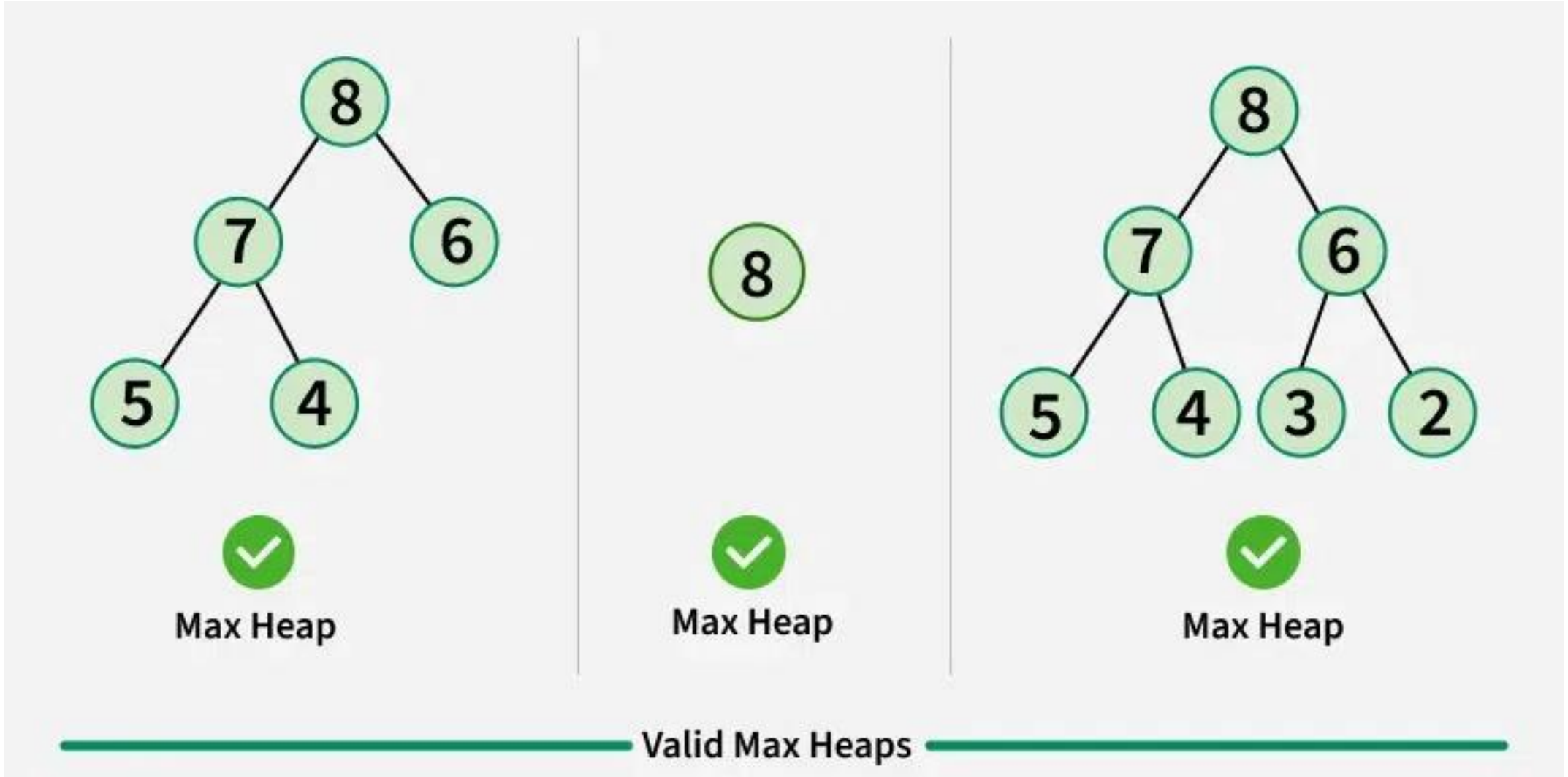

Binary search tree
root
4
left
2
6
right
left
1
right
3
5
left
7
right

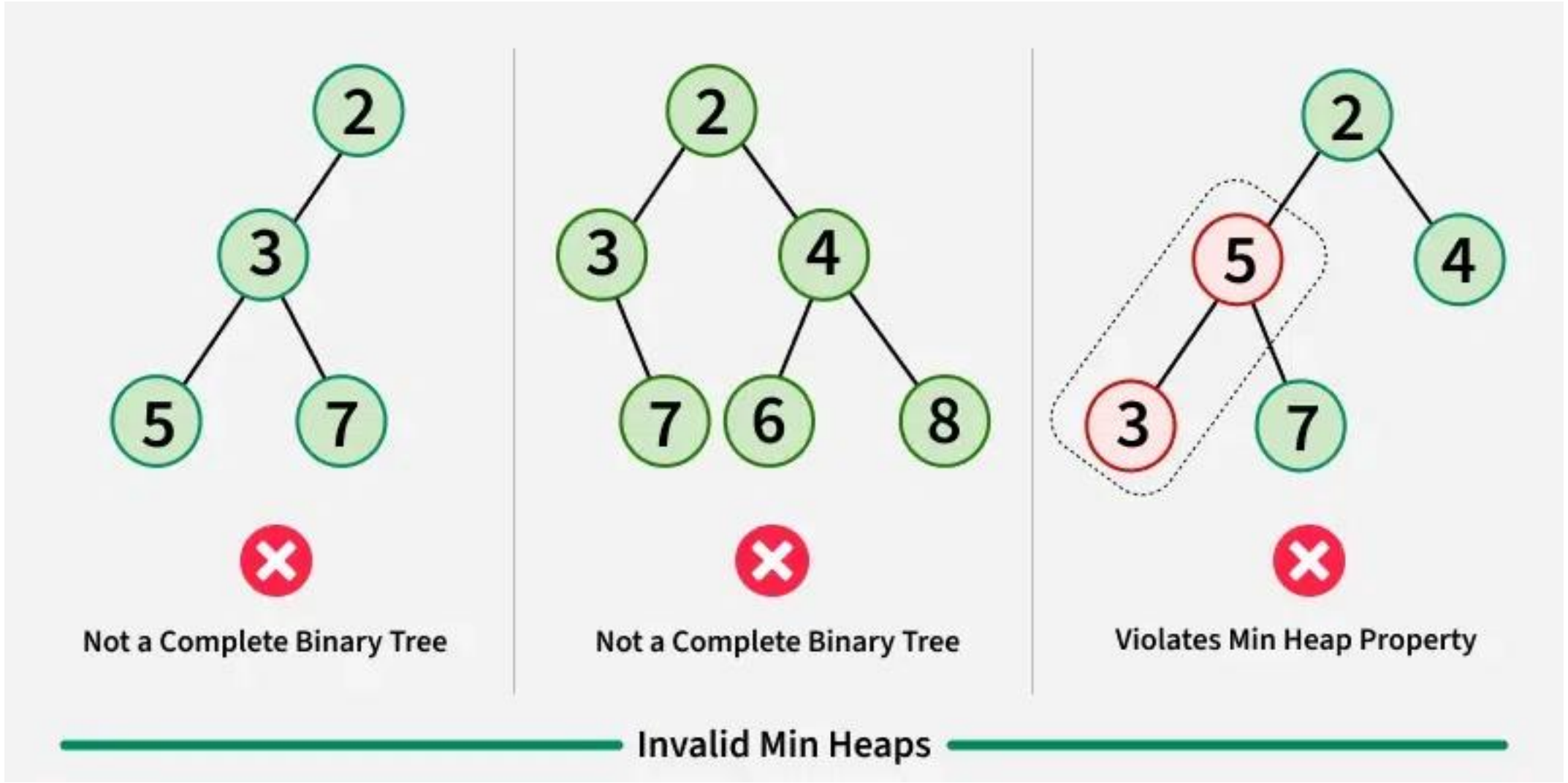

Root
Edge
Key
56
30
70
22
40
60
95
65
11
3
16
63
67
Parent
Siblings
Child
Subtree
Height of the tree
Leaf nodes

Graphs - Heaps

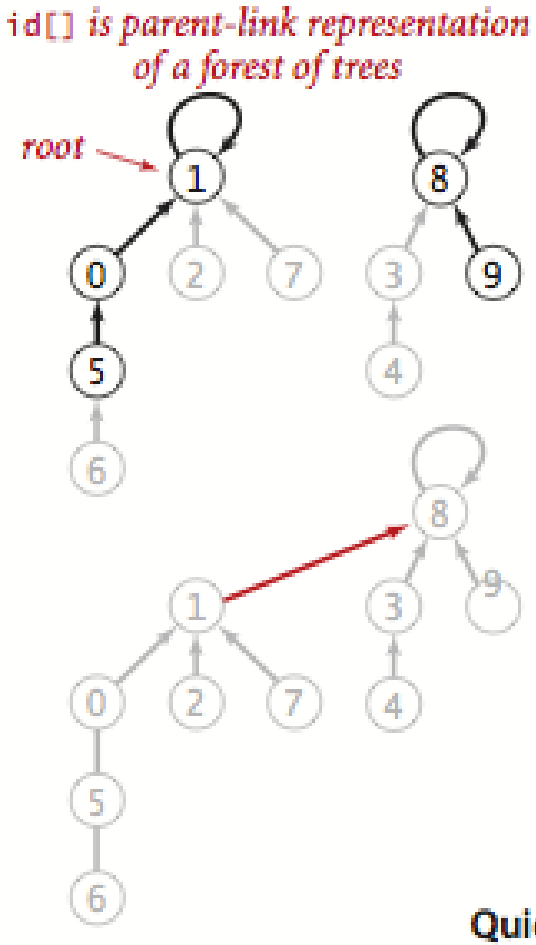
- A special kind of tree:
- Complete binary search tree
  - Complete=every node has the most possible number of children
- (Min-)Heap: The value of each of a node's children is >= its own
- (Max-)Heap: The value of each of a node's children is <= its own


2
3
4
5
7
Min Heap
2
Min Heap
2
3
4
5
6
7
8
Min Heap
Valid Min Heaps


8
7
6
5
4
Max Heap
8
Max Heap
8
7
6
5
4
3
2
Max Heap
Valid Max Heaps


2
3
5
7
Not a Complete Binary Tree
2
3
4
7
6
8
Not a Complete Binary Tree
2
5
4
3
7
Violates Min Heap Property
Invalid Min Heaps

Sets - UnionFind



find has to follow links to the root

p	q	0	1	2	3	4	5	6	7	8	9
5	9	1	1	1	8	3	0	5	1	8	8

`find(5)` *is* `id[id[id[5]]]`

`find(9)` *is* `id[id[9]]`

union changes just one link

p	q	0	1	2	3	4	5	6	7	8	9
5	9	1	1	1	8	3	0	5	1	8	8
		1	8	1	8	3	0	5	1	8	8

Quick-union overview

Explain the project