Code Fellows

Class 30
DSA Review

seattle-javascript-401n14

Vocab Review!

What is Big-O?

What is a Data Structure?

What is a linked-list?

What is a stack?

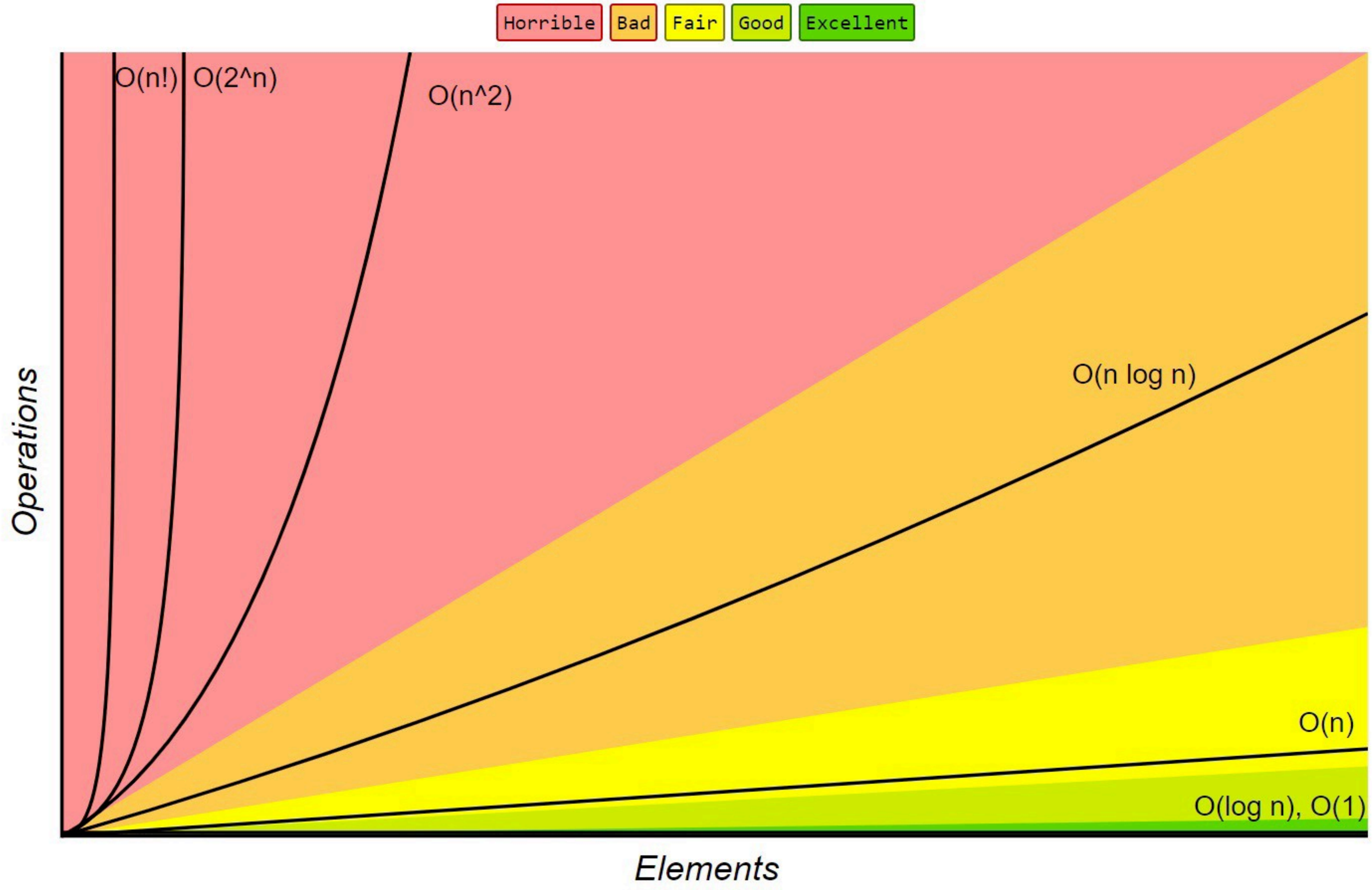
What is a queue?

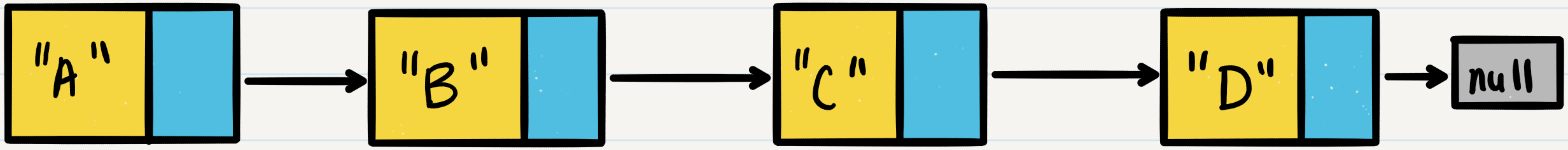
What is a tree?

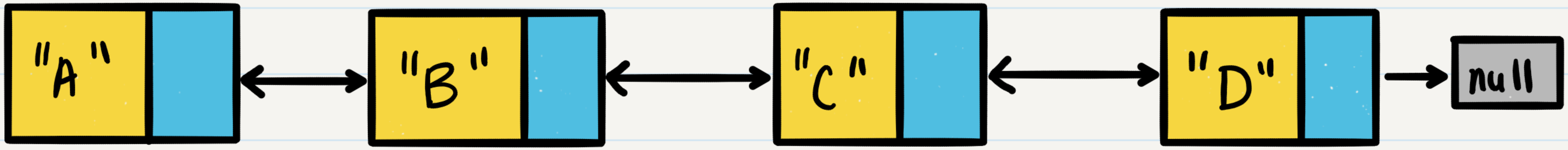
What is a sorting algorithm?

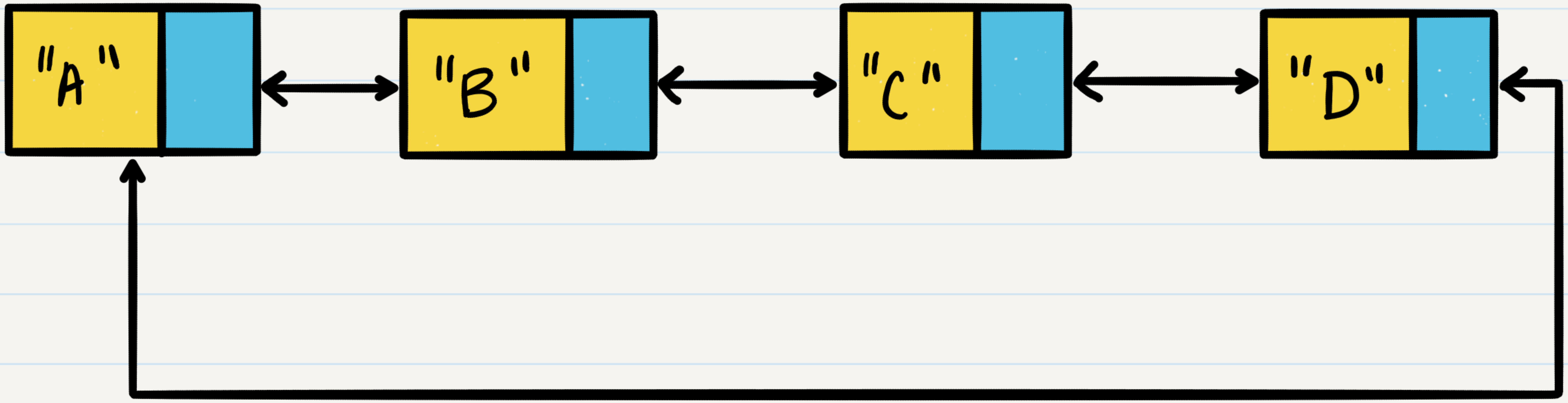
What is a hash table?

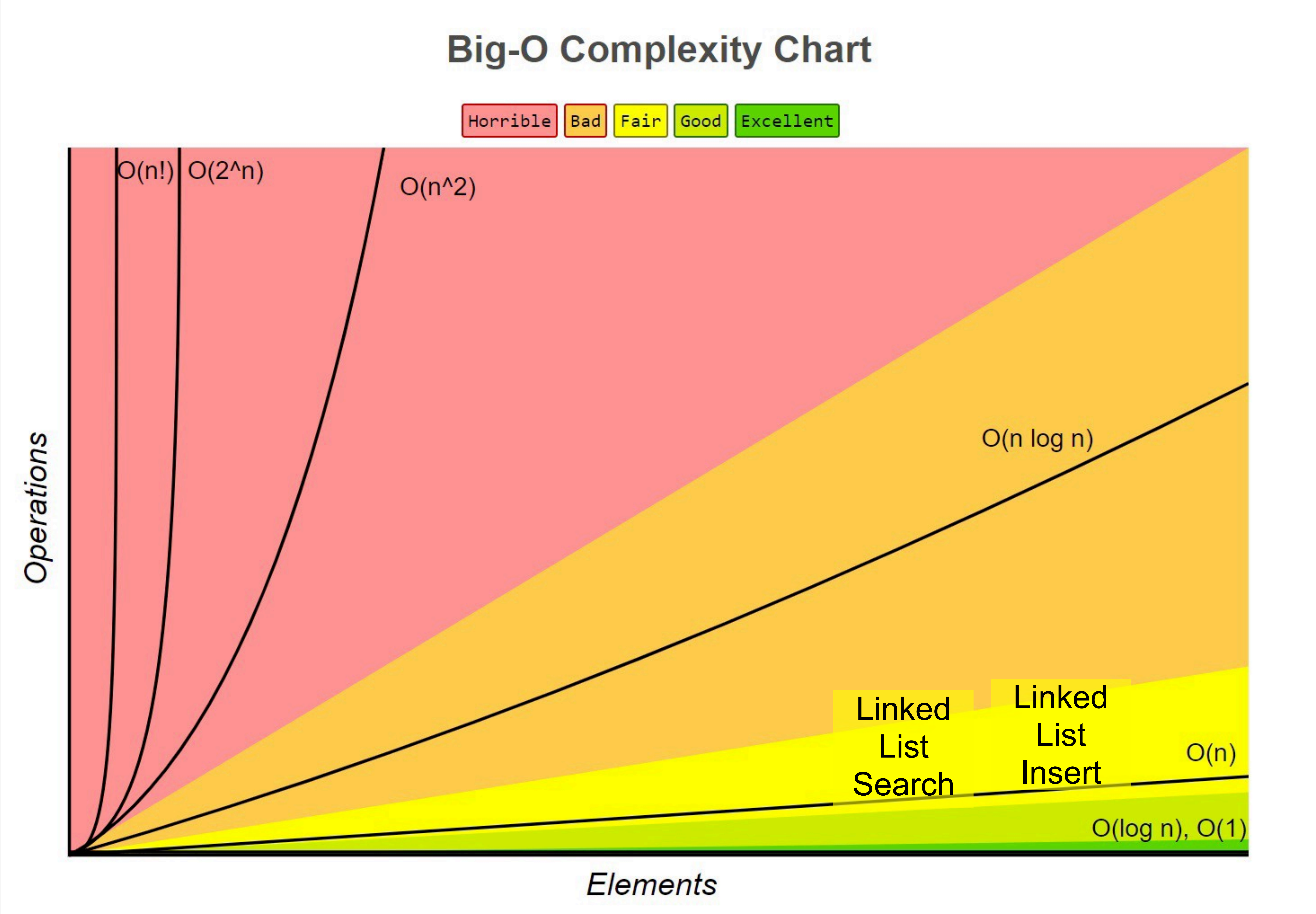
Big-O Complexity Chart




"A"
"B"
"C"
"D"
null

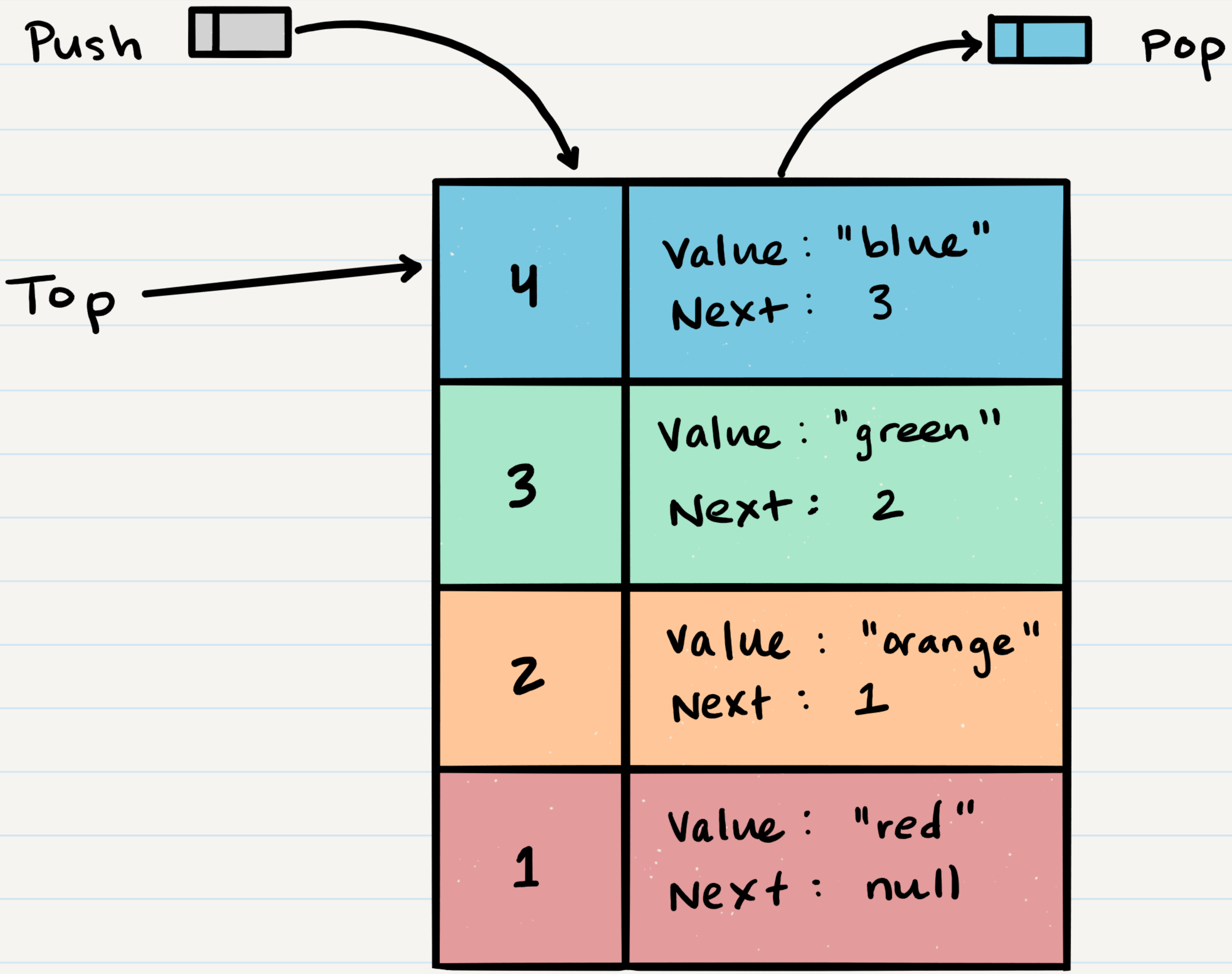

"A"
"B"
"C"
"D"
null

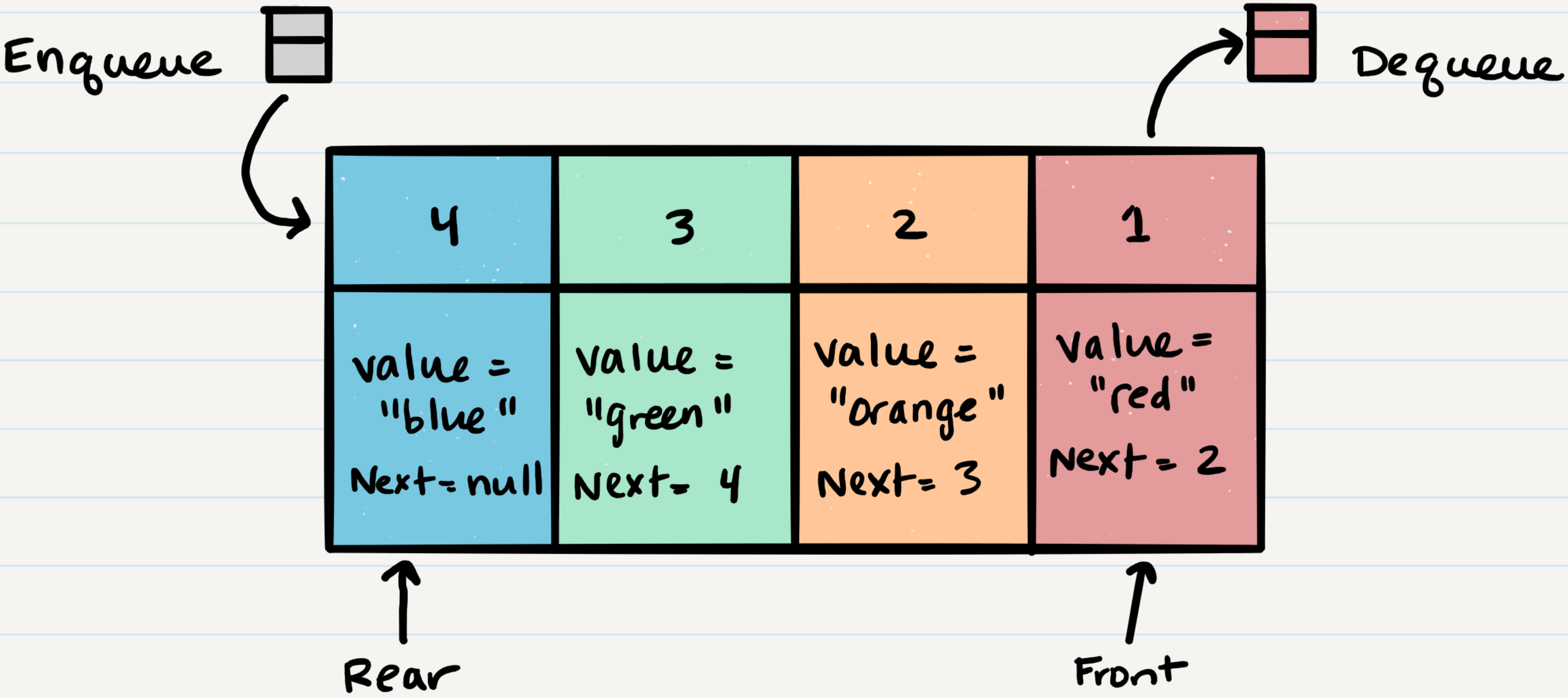

"A"
"B"
"C"
"D"

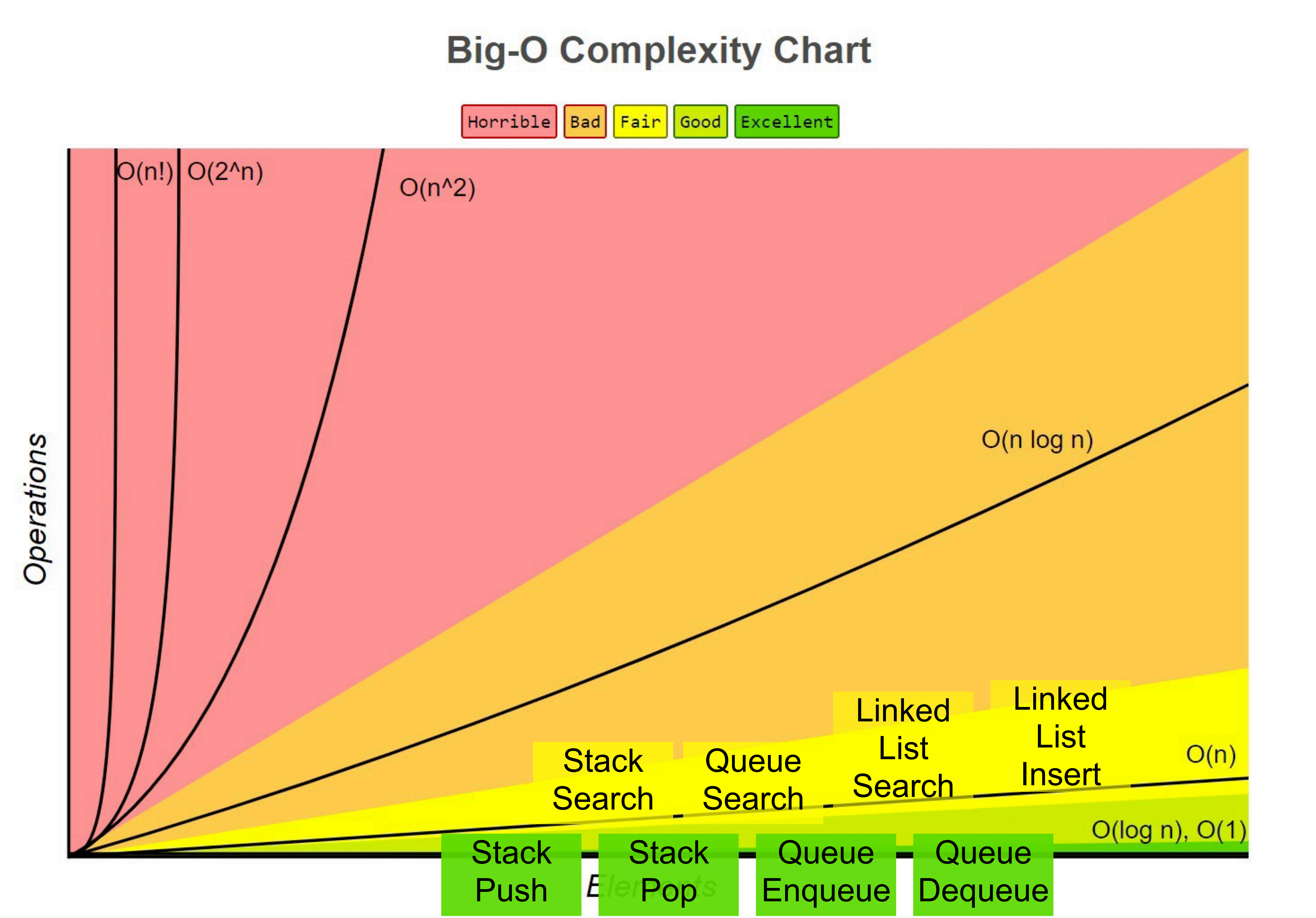

Big-O Complexity Chart
Horrible
Bad
Fair
Good
Excellent
O(n!)
O(2^n)
O(n^2)
O(n log n)
Linked List Search
Linked List Insert
O(n)
O(log n), O(1)
Operations
Elements

Linked List Questions

- Determine if a given linked list has a cycle in it
- Remove duplicates from a given linked list
- Determine if a given linked list is sorted
- Given a linked list that represents a long number (where each digit in the number is a single node), increment that number by one
- Zip together two linked lists
- Given the value of a node to delete, remove it from a linked list


Push
Pop
Top
4
Value : "blue"
Next : 3
3
Value : "green"
Next : 2
2
Value : "orange"
Next : 1
1
Value : "red"
Next : null


Enqueue
Dequeue
4
3
2
1
value = "blue"
Next = null
value = "green"
Next = 4
value = "orange"
Next = 3
value = "red"
Next = 2
Rear
Front



Stack Search

Queue Search

Linked List Search

Linked List Insert

Stack Push

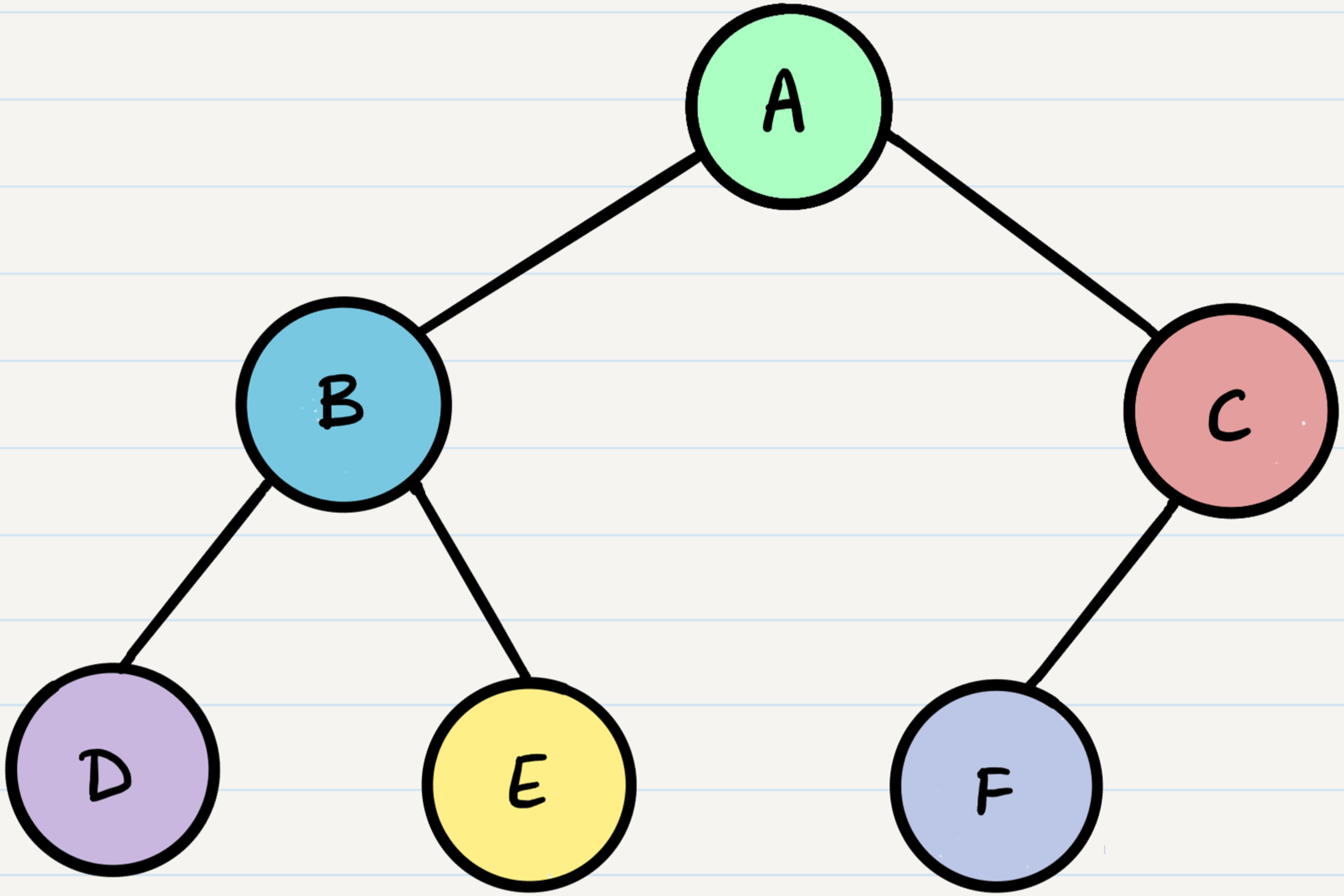
Stack Pop

Queue Enqueue

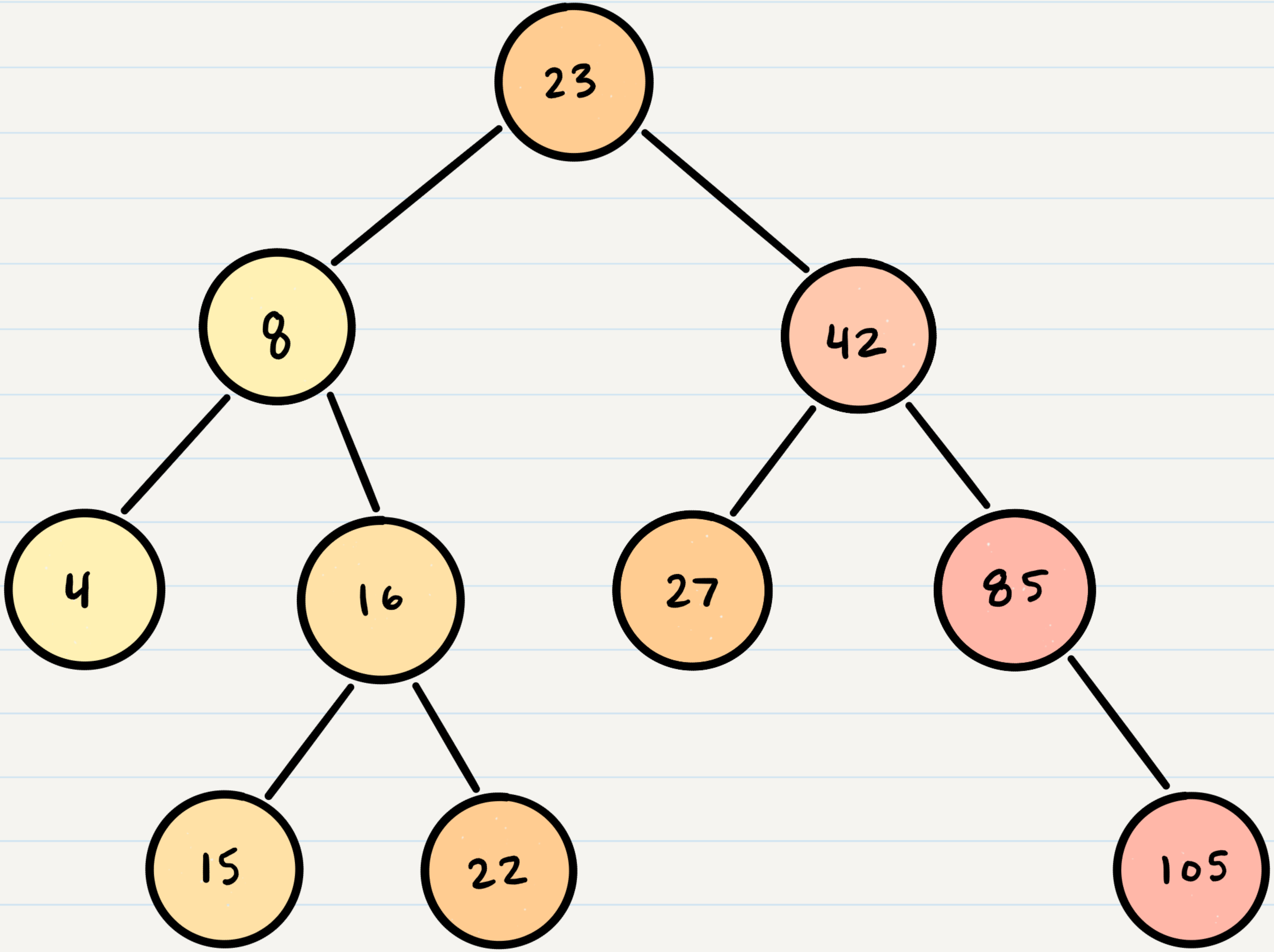
Queue Dequeue

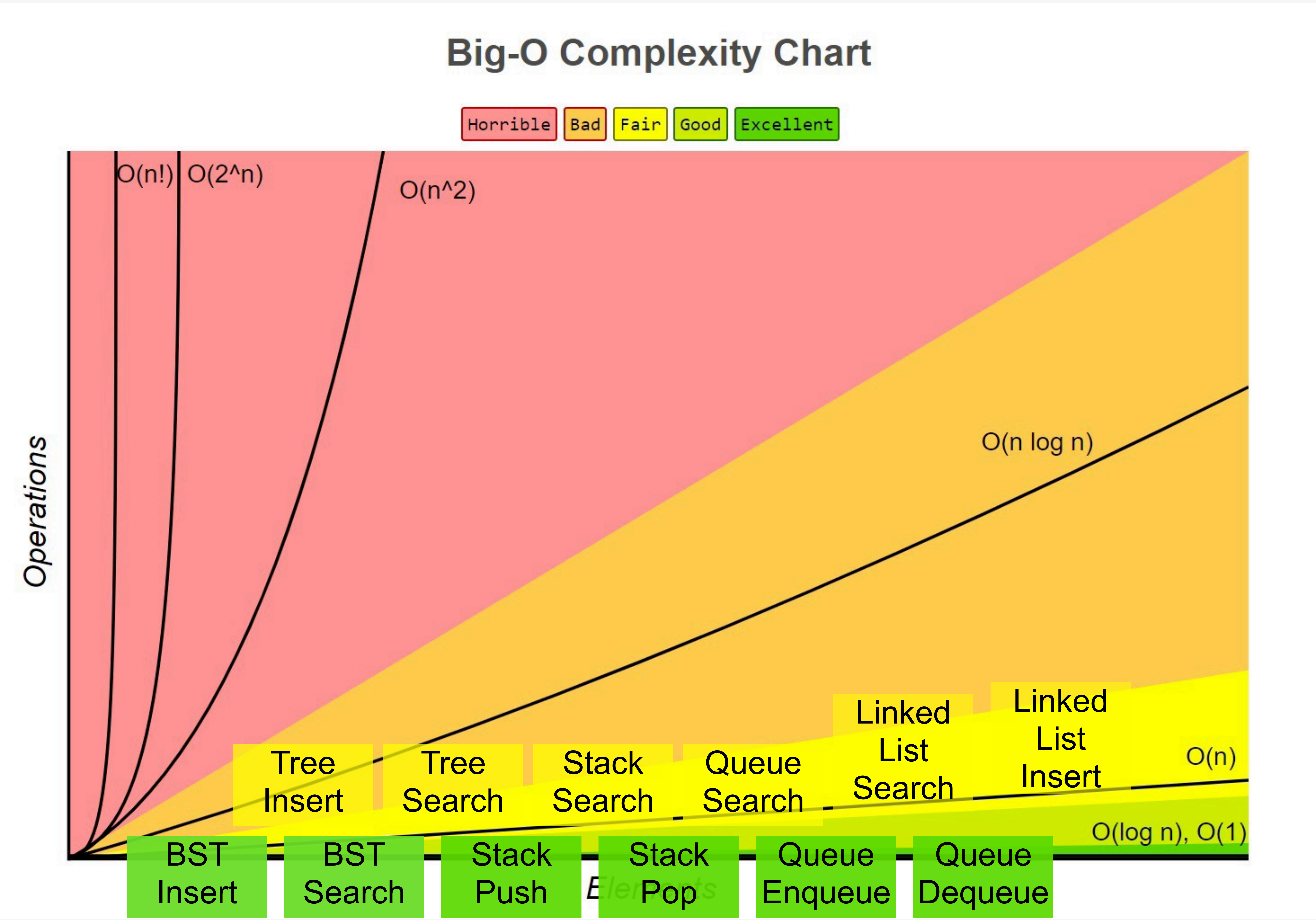
Stacks and Queues Questions

- Given an Animal Shelter with Cats and Dogs, write methods that dequeue any animal, enqueue any animal, dequeue the next Dog, and dequeue the next Cat
- Towers of Hanoi
  - Three poles, leftmost has rings sorted from smallest to largest at the bottom
  - Get the same sort order on the third rightmost ring
- Is string a palindrome
- Are two strings with backspace characters ‘#’ equal
- Determine if there are valid parentheses in a string


A
B
C
D
E
F

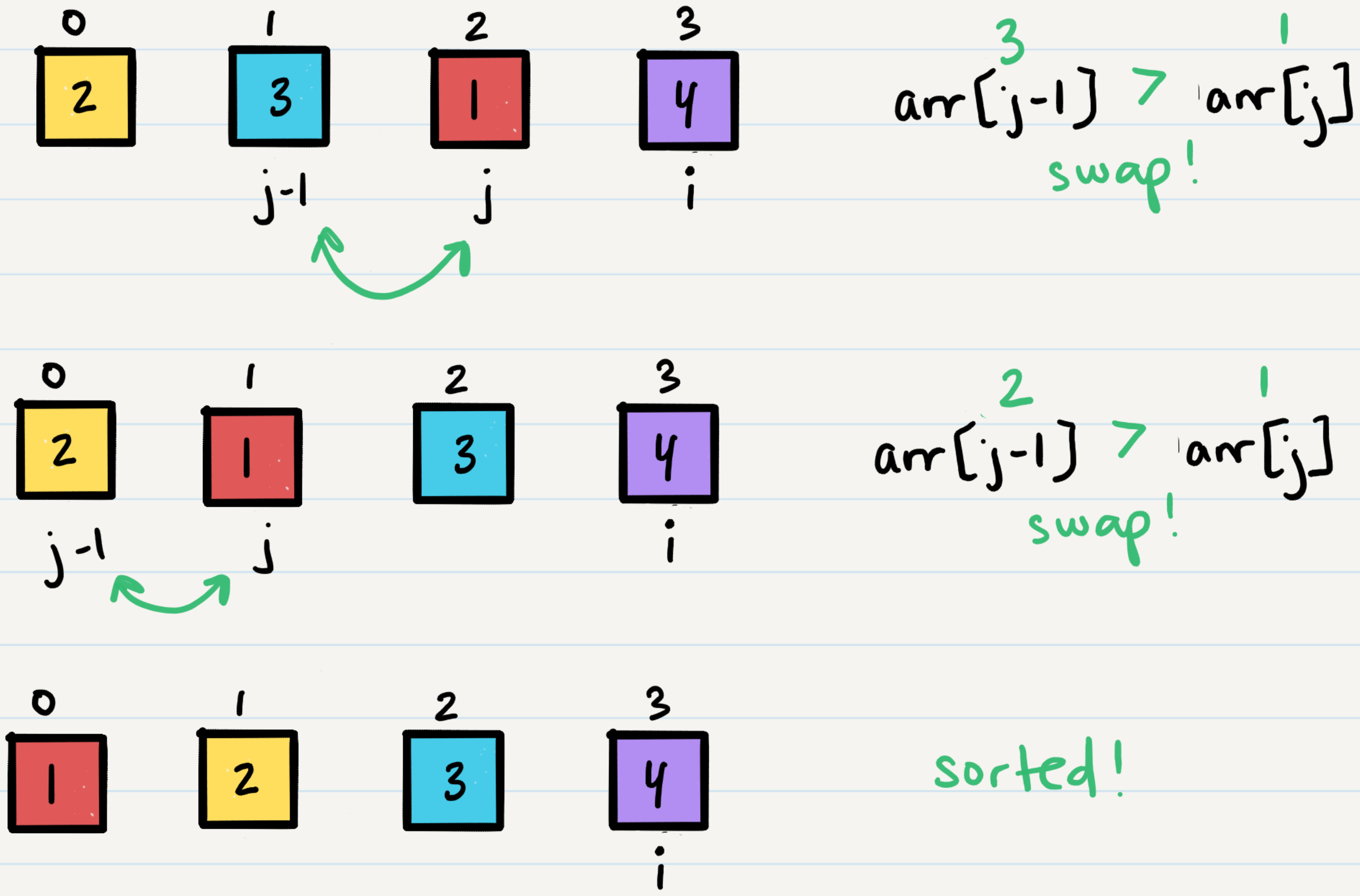
Pre-order: `root >> left >> right`
In-order: `left >> root >> right`
Post-order: `left >> right >> root`

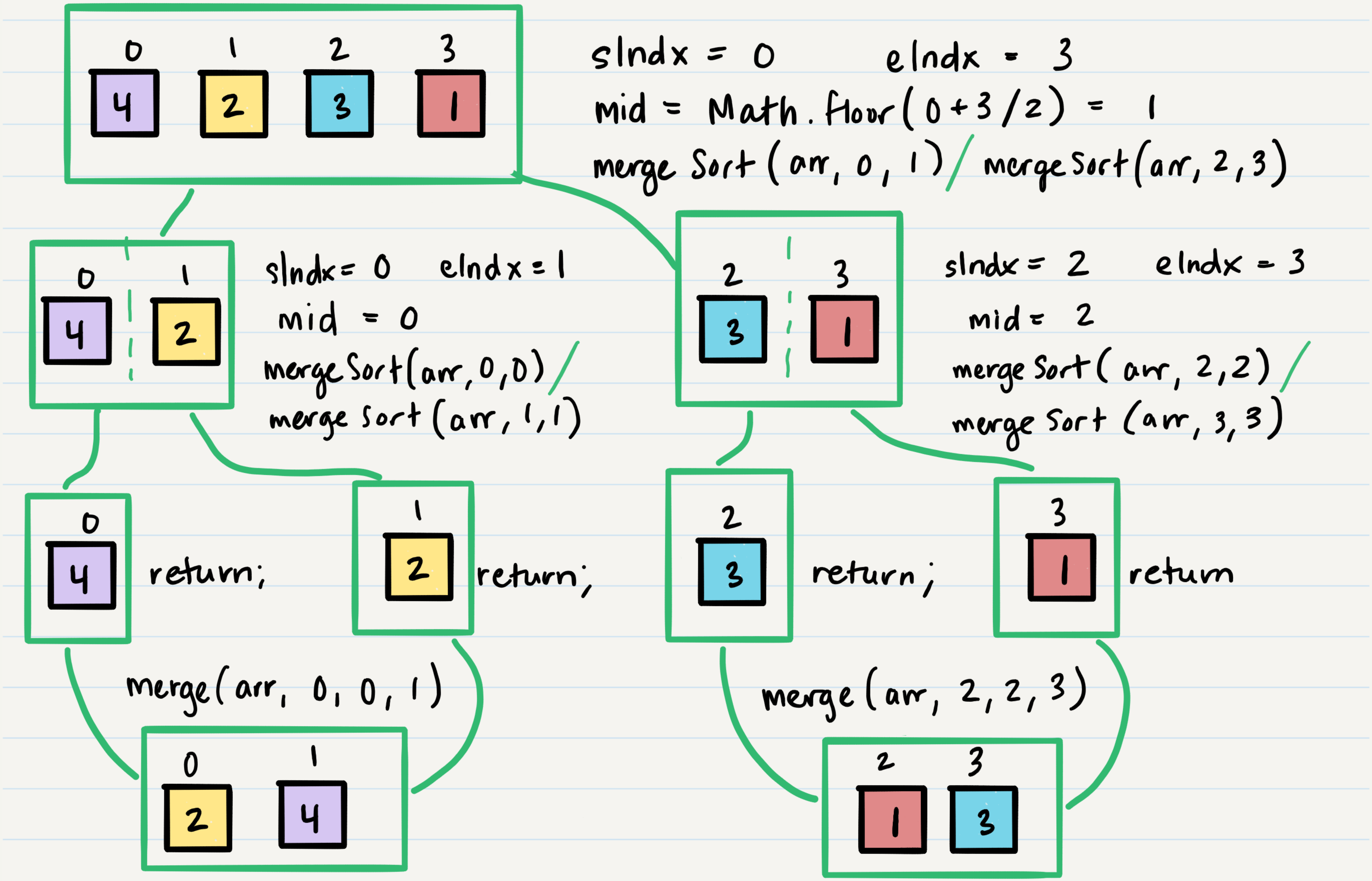

23
8
42
4
16
27
85
15
22
105


Big-O Complexity Chart
Horrible
Bad
Fair
Good
Excellent
Operations
Elements
O(n!)
O(2^n)
O(n^2)
O(n log n)
O(n)
O(log n), O(1)
Tree Insert
Tree Search
Stack Search
Queue Search
Linked List Search
Linked List Insert
BST Insert
BST Search
Stack Push
Stack Pop
Queue Enqueue
Queue Dequeue

Tree Questions

- Merge two binary trees, making the resulting tree nodes a sum of the original tree nodes
- Determine if two binary trees have matching leaves
- Determine if a binary tree is balanced
- Determine if a binary tree has a path where the sum of each node in that path equals a given number
- Determine if two nodes in a binary tree are cousins
- Implement Breadth-First, Post-Order, Pre-Order and In-Order traversals
- Find the lowest common ancestor for two given nodes in a binary tree


0 1 2 3
2 3 1 4
j-1 j i
3 1
arr[j-1] > arr[j]
swap!
0 1 2 3
2 1 3 4
j-1 j i
2 1
arr[j-1] > arr[j]
swap!
0 1 2 3
1 2 3 4
i
sorted!


0 1 2 3
4 2 3 1
sIndx = 0 eIndx = 3
mid = Math.floor(0+3/2) = 1
merge Sort (arr, 0, 1)/ mergeSort(arr, 2,3)
0 1
4 2
sIndx= 0 eIndx=1
mid = 0
mergeSort(arr,0,0)/
merge sort (arr, 1,1)
2 3
3 1
sIndx = 2 eIndx = 3
mid = 2
merge Sort (arr, 2,2)/
merge Sort (arr, 3, 3)
0
4
return;
1
2
return;
2
3
return;
3
1
return
merge(arr, 0, 0, 1)
0 1
2 4
merge(arr, 2, 2, 3)
2 3
1 3

0	1	2	3	4
E	D	B	C	A

Step 1: Pick a Pivot

0	1	2	3	4
E	D	B	C	A

pivot = 3

Step 2: Swap the pivot with the last item in the array

0	1	2	3	4
E	D	B	A	C

pivot = 4

Step 3: Find the first item larger than the pivot, left → right

0	1	2	3	4
E	D	B	A	C

fil = 0 pivot = 4

Step 4: Find the first item smaller than the pivot, right → left

0	1	2	3	4
E	D	B	A	C

fil = 0 pivot = 4 fis = 3

Step 5: Swap the two values

0	1	2	3	4
A	D	B	E	C

pivot = 4

Repeat steps 3-5 until fil > fis

0	1	2	3	4
A	D	B	E	C

pivot = 4 fil = 1 fis = 2

0	1	2	3	4
A	B	D	E	C

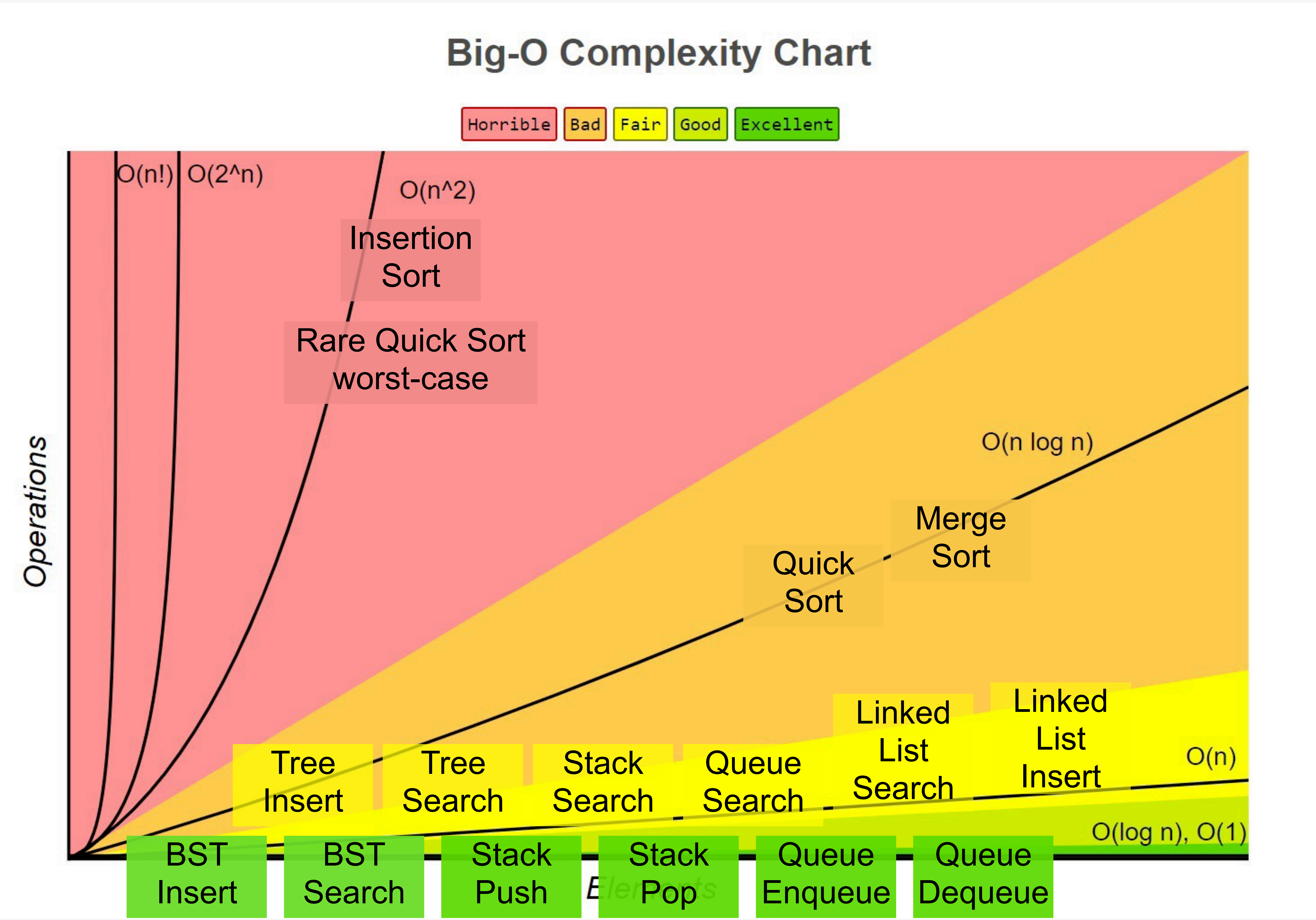
swap!

0	1	2	3	4
A	B	D	E	C

now, fil = 2 fis = 1

fil > fis!

2 > 1



Insertion Sort

Rare Quick Sort worst-case

Quick Sort

Merge Sort

Tree Insert

Tree Search

Stack Search

Queue Search

Linked List Search

Linked List Insert

BST Insert

BST Search

Stack Push

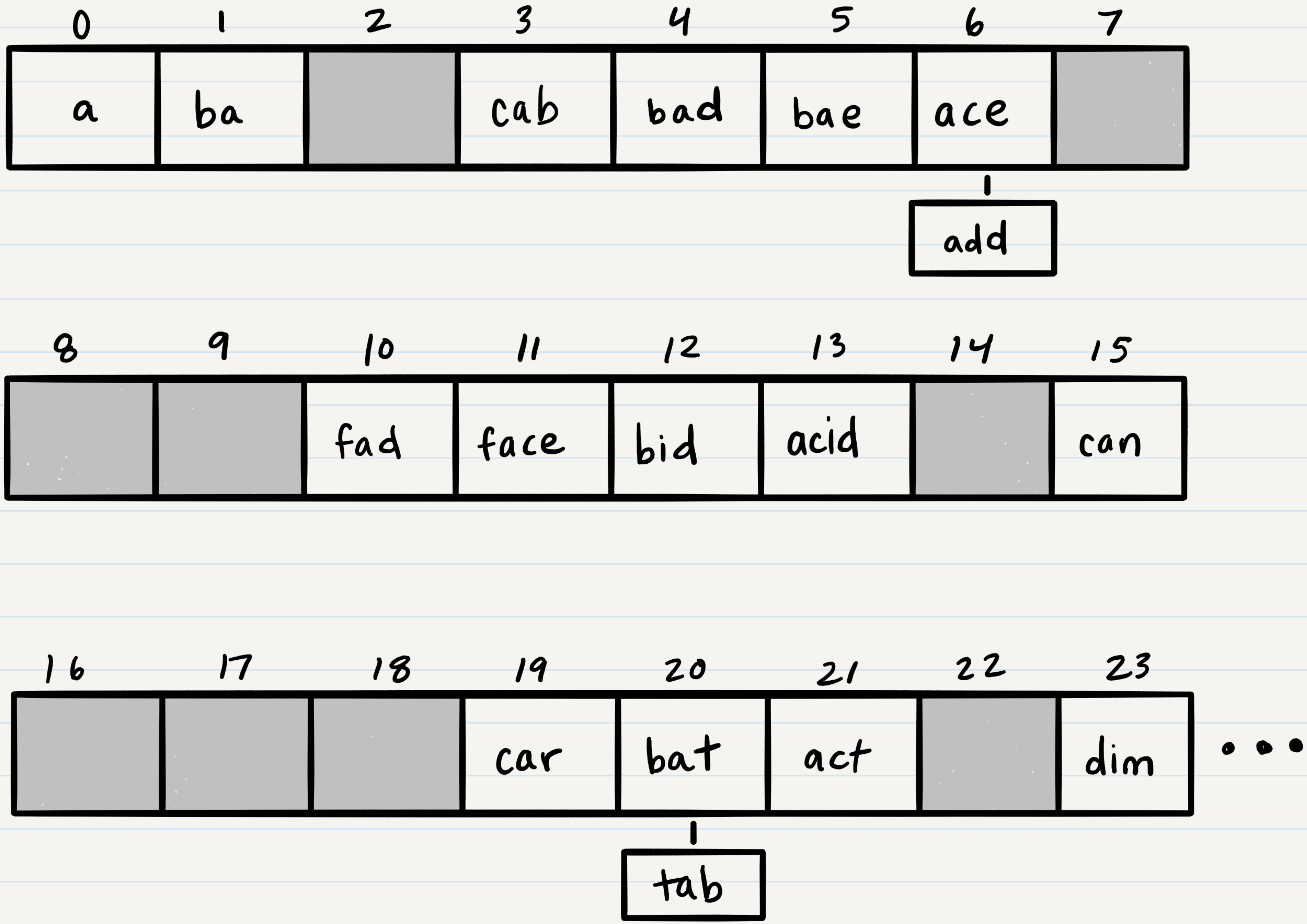
Stack Pop

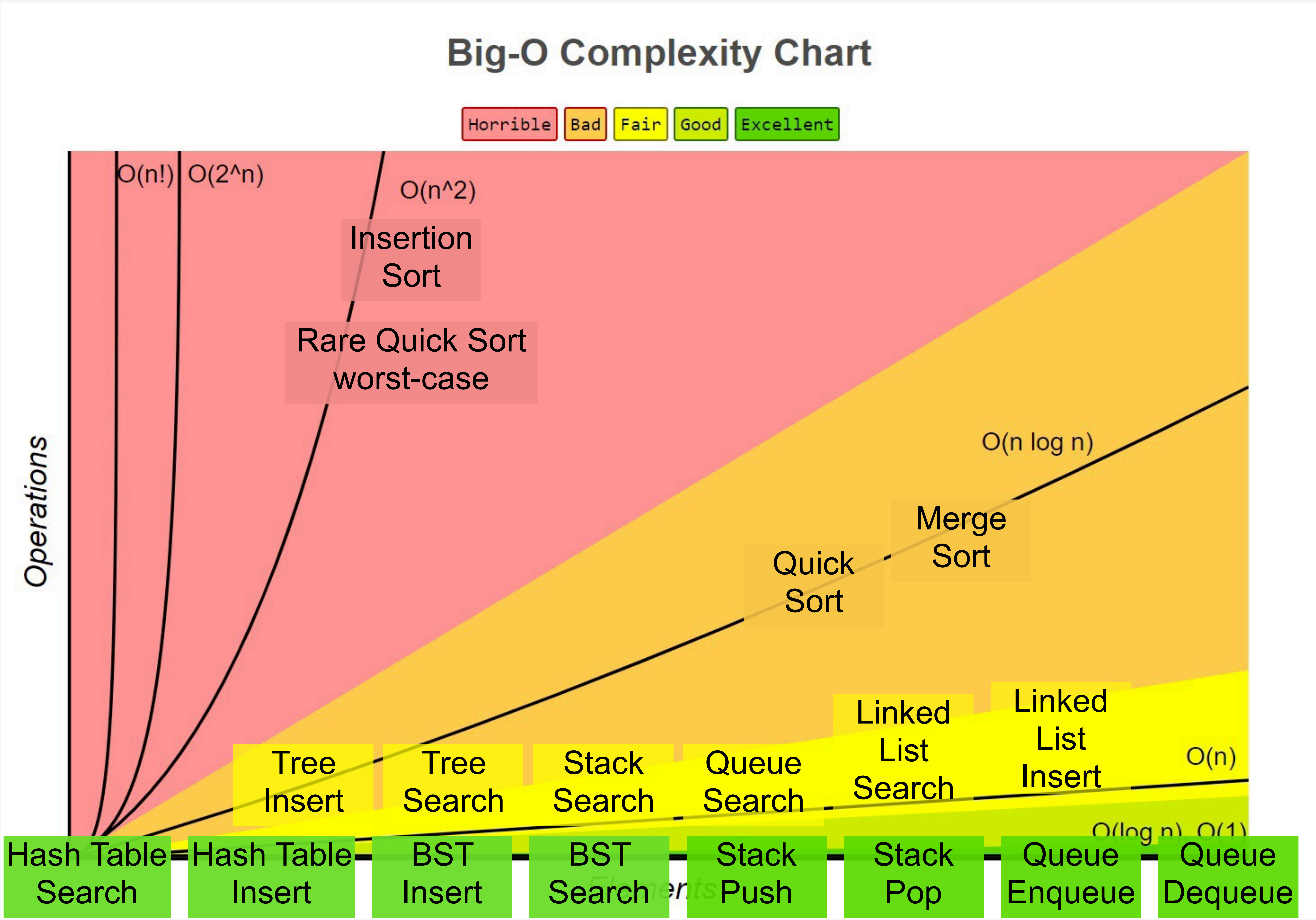
Queue Enqueue

Queue Dequeue

Sorting Questions

- Implement Insertion Sort
- Implement Merge Sort
- Implement Quick Sort


0
1
2
3
4
5
6
7
a
ba
cab
bad
bae
ace
add
8
9
10
11
12
13
14
15
fad
face
bid
acid
can
16
17
18
19
20
21
22
23
car
bat
act
dim
tab
• • •



Insertion Sort

Rare Quick Sort worst-case

Quick Sort

Merge Sort

Tree Insert

Tree Search

Stack Search

Queue Search

Linked List Search

Linked List Insert

Hash Table Search

Hash Table Insert

BST Insert

BST Search

Stack Push

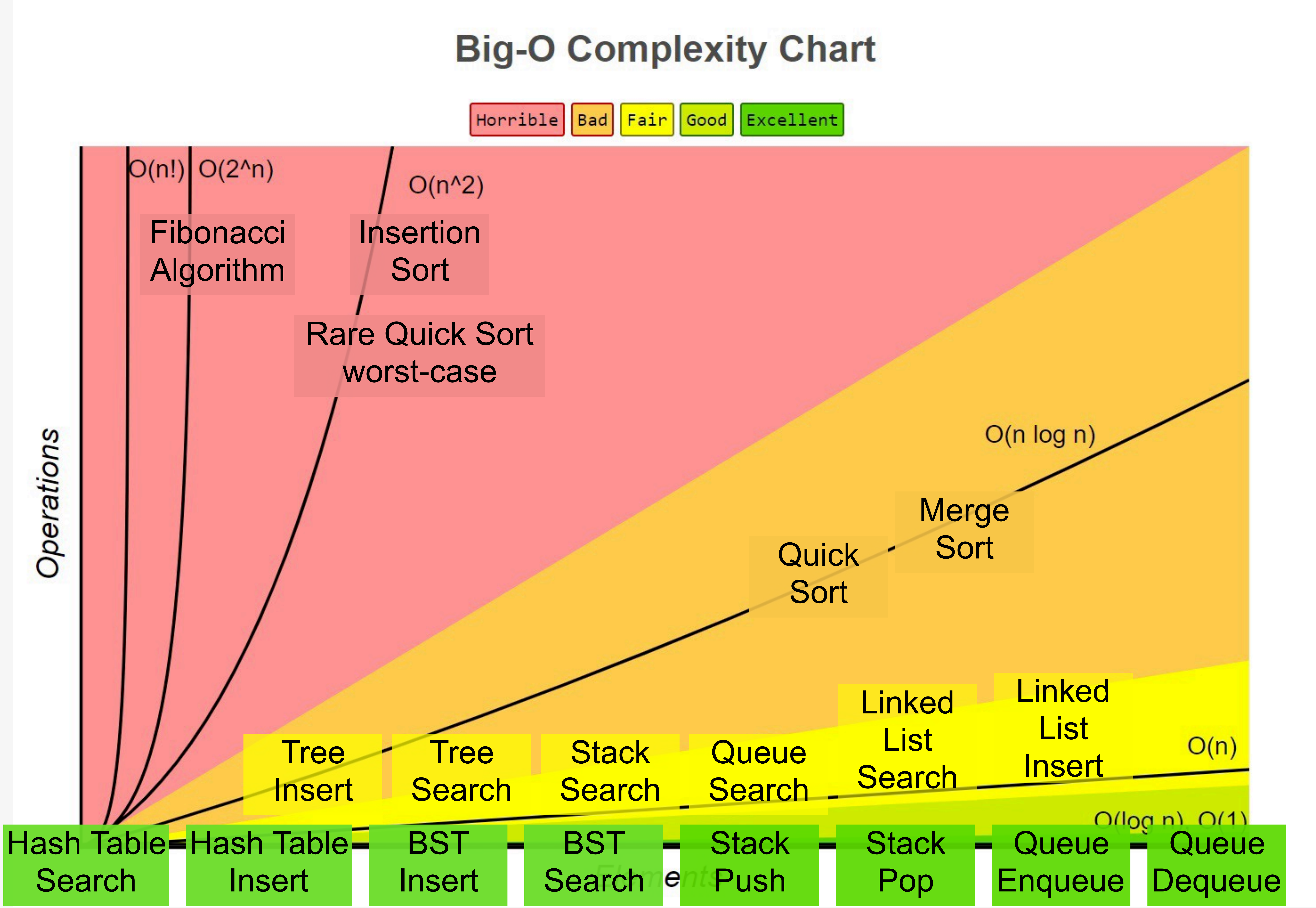
Stack Pop

Queue Enqueue

Queue Dequeue

Hash Table Questions

- Given two sentences, return the words that are not shared between the two
- Build a Hash Table from scratch, with your own hash function
- Given a string, find the first non-repeating character in it and return it's index
- Determine if two strings are anagrams of one another
- Given a pattern and a string, determine if the string follows that pattern



Fibonacci Algorithm

Insertion Sort

Rare Quick Sort worst-case

Quick Sort

Merge Sort

Tree Insert

Tree Search

Stack Search

Queue Search

Linked List Search

Linked List Insert

Hash Table Search

Hash Table Insert

BST Insert

BST Search

Stack Push

Stack Pop

Queue Enqueue

Queue Dequeue