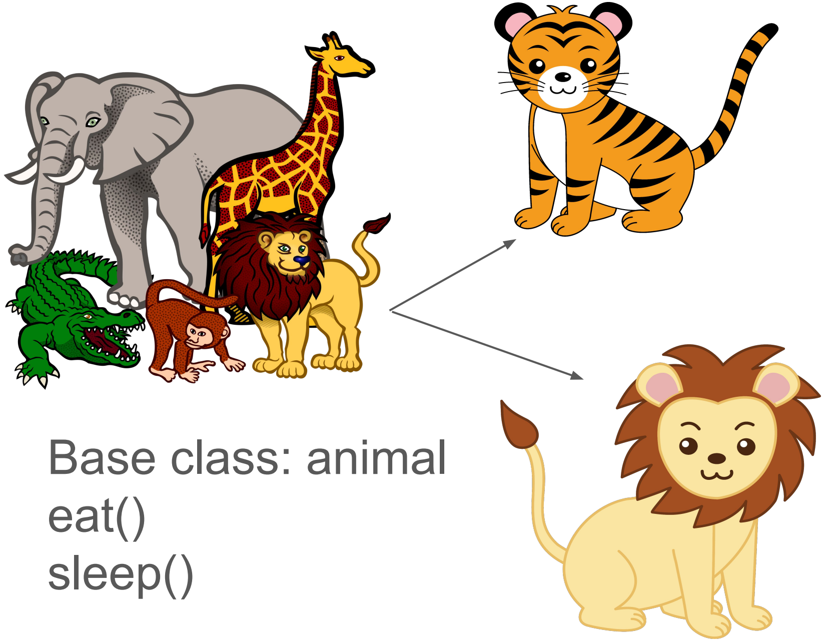
OOP, classes, inheritance

```
class Animal:
    def eat(self): pass

class Tiger(Animal):
    def meow(self):
        print("meow")

class Lion(Animal):
    def rawr(self):
        print("rawr")
```



Base class: animal
eat()
sleep()

Subclass: tiger
meow()
Also has: eat() and sleep()

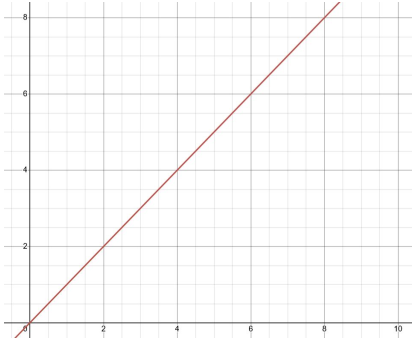
Subclass: lion
rawr()
Also has: eat() and sleep()

Organize code into reusable objects (classes) that bundle data + methods

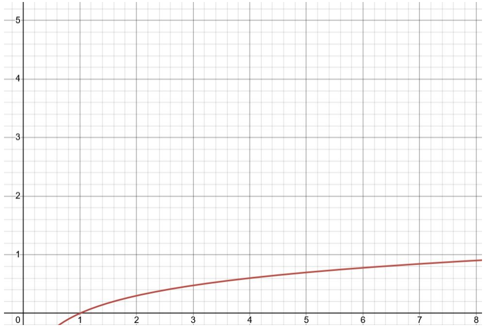
Encapsulation: Hide internal state (use private attributes).

Polymorphism: One interface, multiple implementations

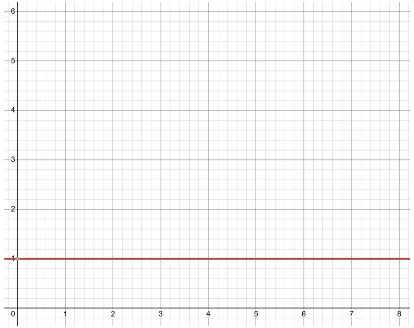
Big-Oh/Algorithm Analysis



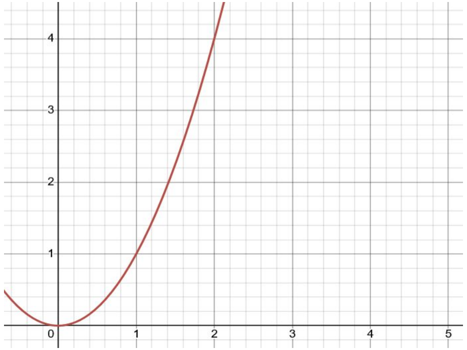
O(n):
Linear time
Iterating through list



O(log n):
Logarithmic time
Binary search



O(1):
Constant time
Array access

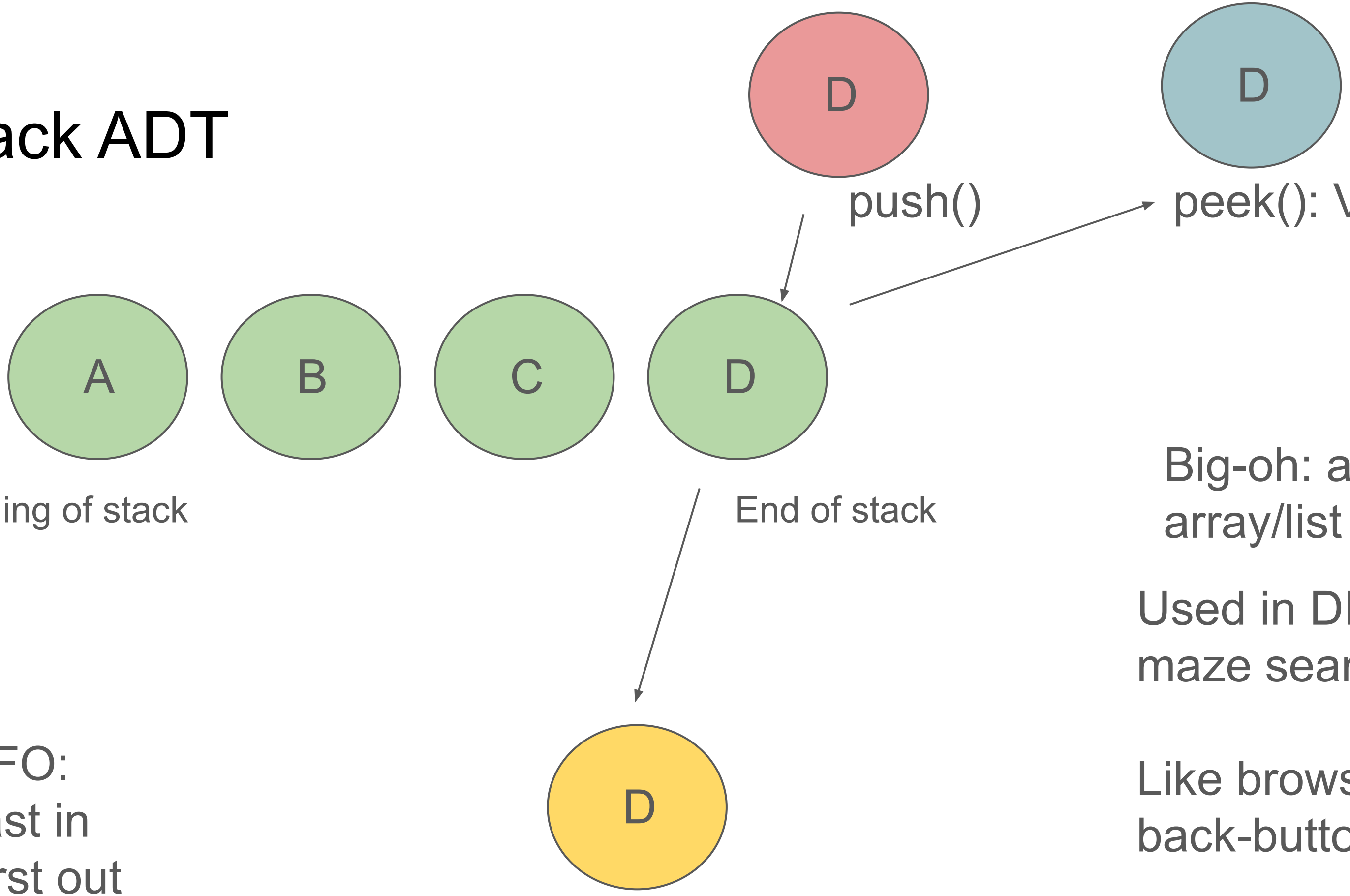


O(n^2):
Quadratic time
Nested loop

Measures how an algorithm's runtime/memory scales with input size

Example: O(n^2) means time grows quadratically (bad for large inputs)

Stack ADT



push()

peek(): View top item.

Beginning of stack

End of stack

pop()

Big-oh: all O(1) with array/list

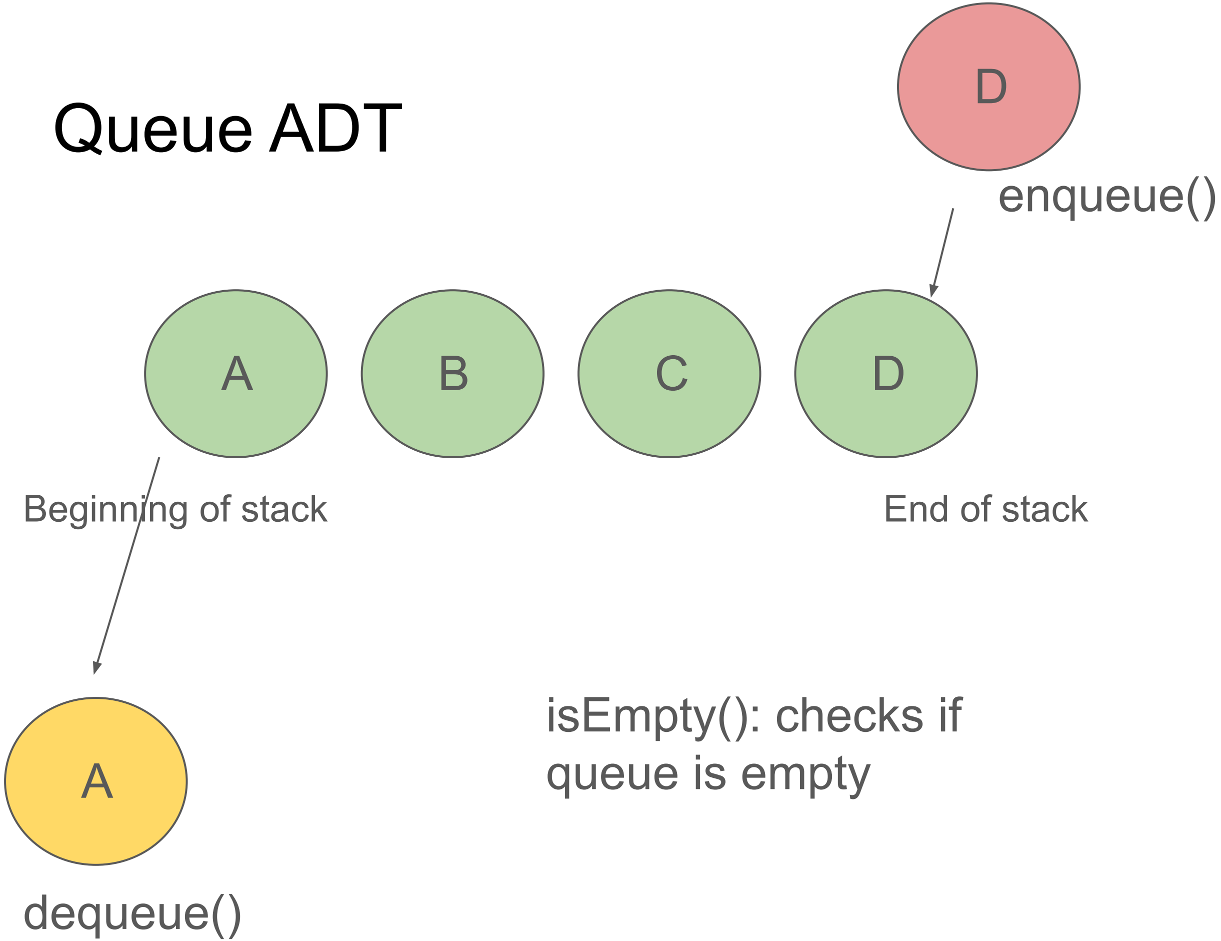
Used in DFS for maze searching

LIFO:
Last in
First out

Like browser back-button history

isEmpty(): checks if stack is empty

Queue ADT



enqueue()

Beginning of stack

End of stack

dequeue()

isEmpty(): checks if queue is empty

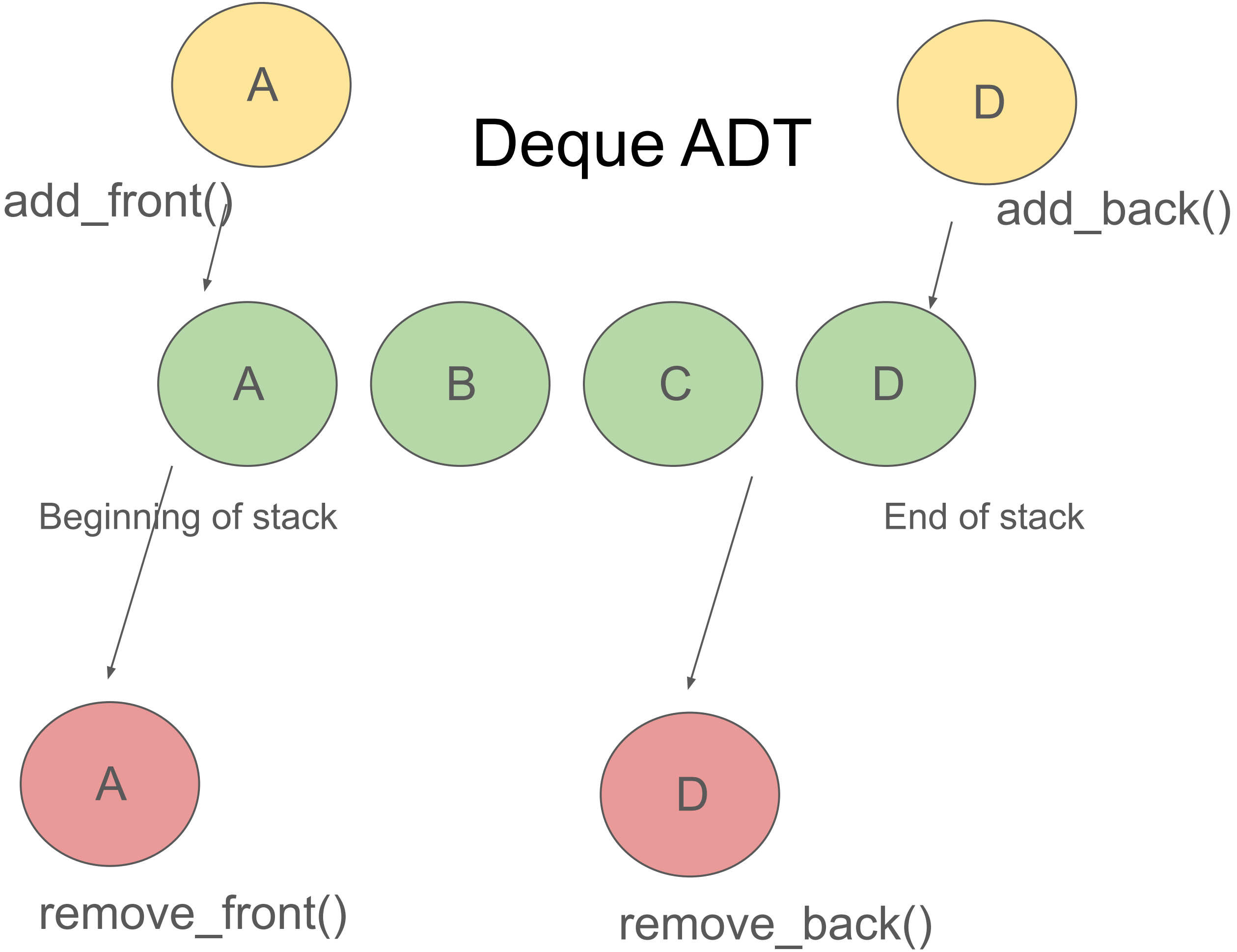
FIFO:
First in
First out

Big-oh: all O(1)* with array/list

Used in DFS for maze searching

Like browser back-button history

Deque ADT



isEmpty(): checks if queue is empty

You can add from either side of the doubly-linked list

Big-oh: all O(1) with doubly-linked list

Can be used to check for palindromes

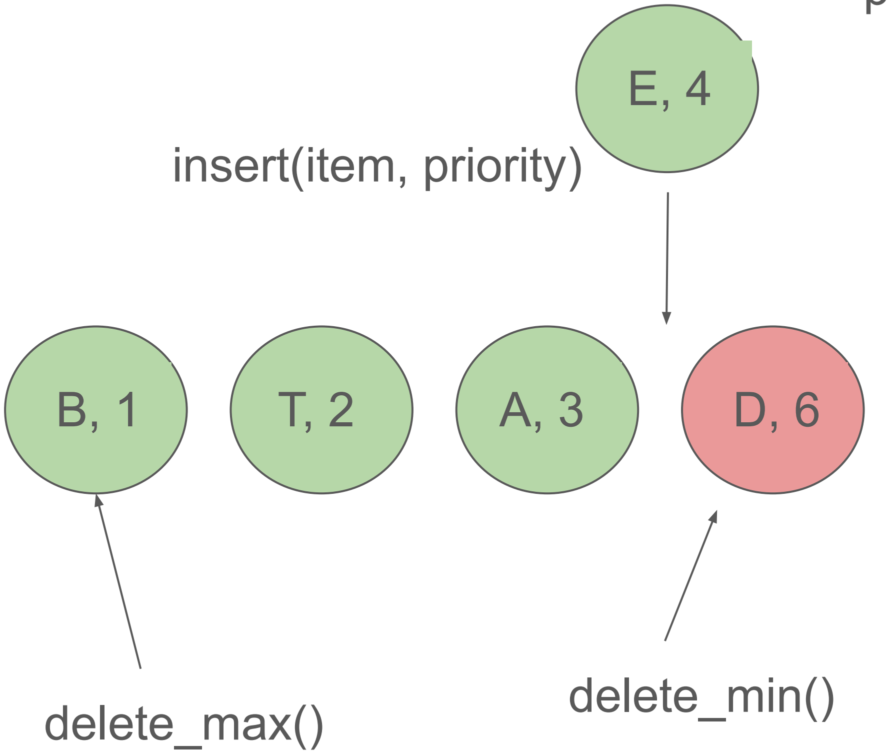
Like undo/redo for text editors

Priority Queue

Dijkstra’s algorithm (prioritizes shortest paths in graphs).

Stores items in order of priority

Big-oh: o*(log n) for insert/delete (heap)

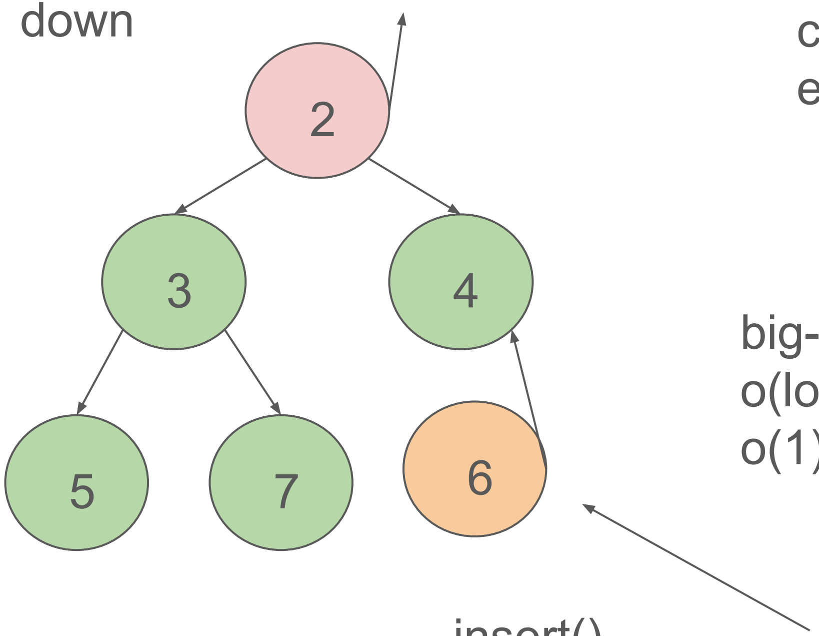


Example: calendar (priority is time)

Heaps

Stored into
[2, 3, 4, 5, 7, 6*]
For easy access

find_min(): remove root and replace with last element, bubble down



Binary tree:
Every node, value of its children is greater than or equal to its own value

big-io:
o(log n) insert
o(1) find-min

Example: HeapSort instead of O(n log n) srting

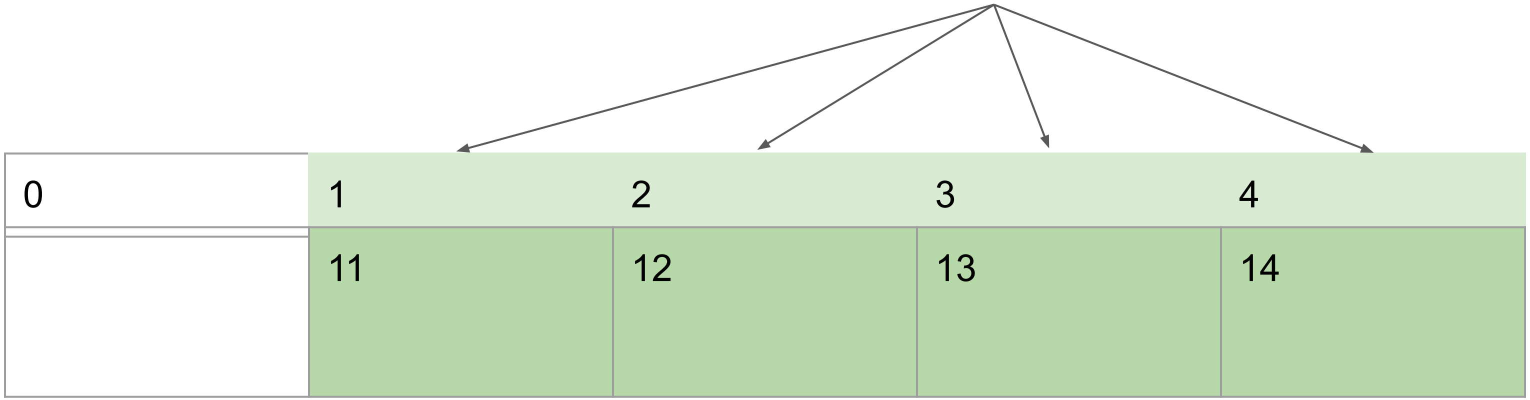
insert()

Bubble up and compare to sort

Hashing

Mapping data to specific index in hash table using hash function

List = [11, 12, 13, 14] ⟶ H(x) = [x % 10]

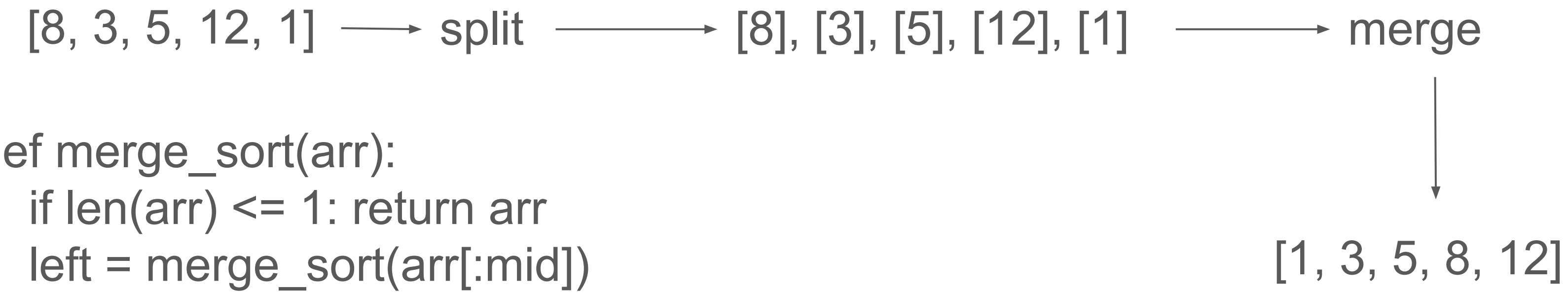


0	1	2	3	4
	11	12	13	14

All operations (search, insert, and delete) are o(1) on average

Example: python dict
Password manager (key-value)

Recursion and Merge Sort



[8, 3, 5, 12, 1] → split → [8], [3], [5], [12], [1] → merge → [1, 3, 5, 8, 12]

```
def merge_sort(arr):
   if len(arr) <= 1: return arr
   left = merge_sort(arr[:mid])
   right = merge_sort(arr[mid:])
   return merge(left, right)
```

Example: sorting large datasets (stable and predictable).

Divide: split array into halves.
Conquer: recursively sort halves.
Combine: merge sorted halves (O(n) per merge).