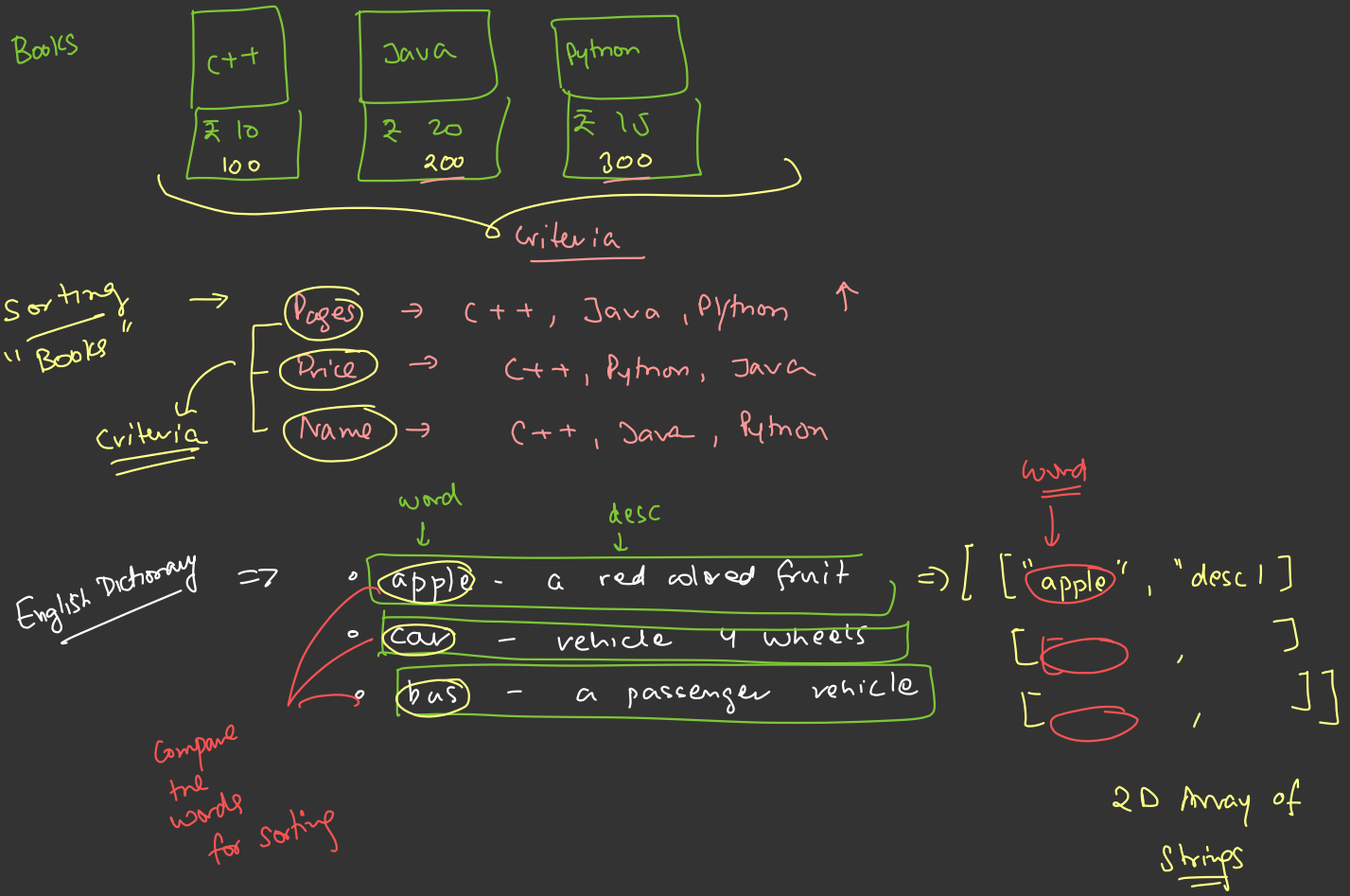
⇒ SORTING [1 class]

{more classes on Sorting in Adv Batch}

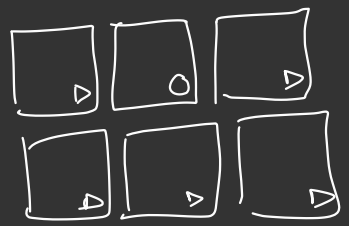
→ Arranging data in a 'particular order'

- English Dictionary
- Roll No
- Contact List in Phone
- Books in Library
- Google web Pages
  - ↳ Ranked by Relevance (to search query)

} Easy to search


Books
C++
₹ 10
100
Java
₹ 20
200
Python
₹ 15
300
Criteria
Sorting "Books"
Criteria
Pages → C++, Java, Python
Price → C++, Python, Java
Name → C++, Java, Python
English Dictionary =>
word
desc
apple - a red colored fruit
car - vehicle 4 wheels
bus - a passenger vehicle
Compare the words for sorting
=> [["apple", "desc1"]
[,]
[,]]
word
2D Array of Strings

Instagram → #tag Reels



⇒ ♡ Liked on that #song

⇒ on that timestamp

→ Liked 3 days, ♡ likes

Make My Trip

→ Early Flights.

→ Cheaper Flights low – high

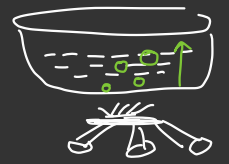
→ Duration 8 ~~hrs~~ 3 hrs

Preference

Amazon → Sort A–Z Price on Mobiles.

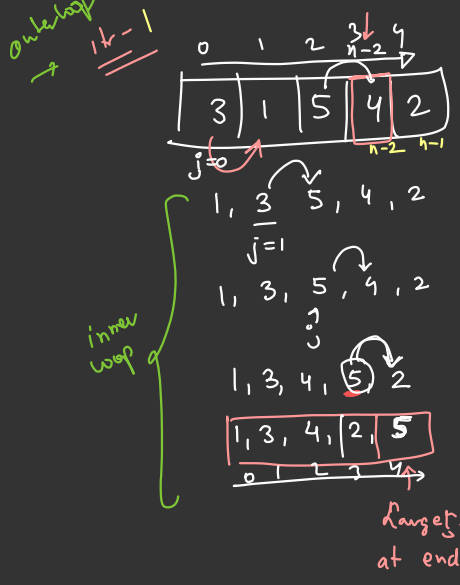
Bubble Sort

Unsorted



Large Bubbles move to Top

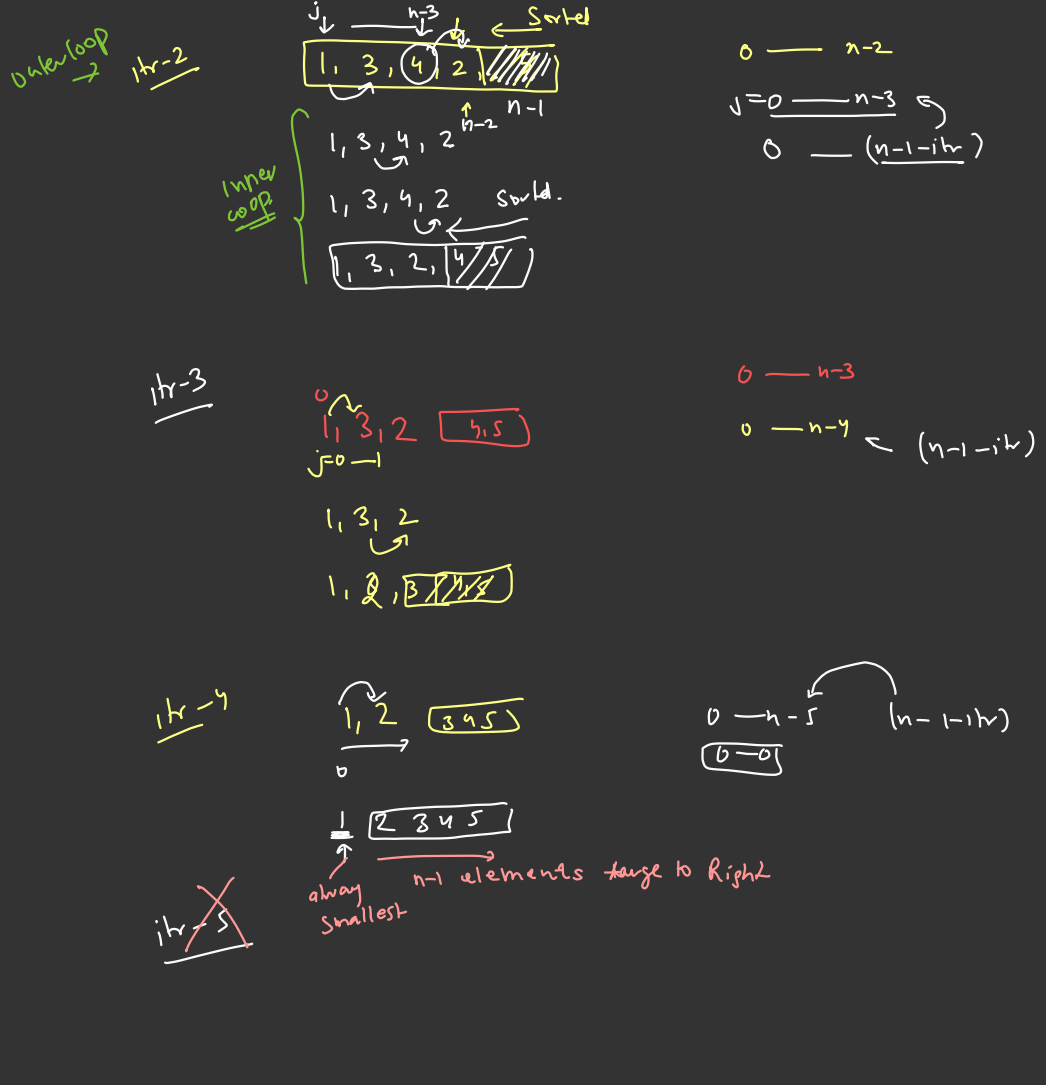
In each itr it moves the larger element to the end of array by Comparing & Swapping two Consecutive elements.

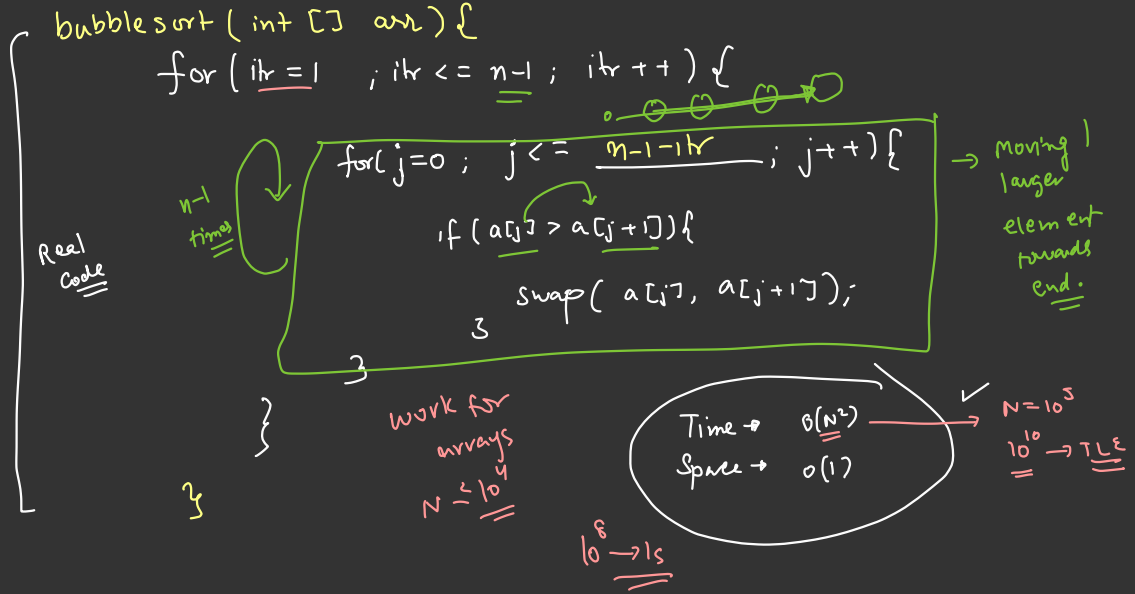



```
for(j=0 ——→ n-2)
    if (a[j] > a[j+1])
      { swap(a[j], a[j+1]); }
    j++
```

itr1 ⇒ j=0 — (n-2)

n-1-itr


Outer loop
itr-2
Sorted
n-3
n-2
n-1
0 —— n-2
j=0 —— n-3
0 —— (n-1-itr)
Inner loop
Sorted.
itr-3
0 —— n-3
0 —— n-4 < (n-1-itr)
itr-4
0 —— n-5
(n-1-itr)
0 —— 0
n-1 elements have to Right
always Smallest
itr-5


bubblesort (int [] arr) {
for (itr = 1 ; itr <= n-1 ; itr ++) {
for(j=0 ; j <= n-1-itr ; j++) {
if (a[j] > a[j+1]) {
swap(a[j], a[j+1]);
}
}
}
}
Real Code
n-1 times
Moving larger element towards end.
Work for arrays N <= 10^4
Time → O(N^2)
Space → O(1)
N = 10^5
10^10 → TLE
10^8 → 1s

Recommended : in-built sort function (N Log N)

Java (Randomised Quicksort) ← Arrays.sort(---)
↑ cross verify. Collections.sort(--)

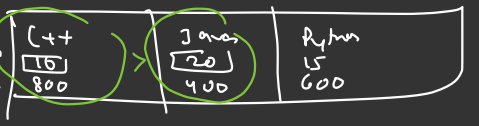
TODO. Refer sorting syntax from web

Book

Book[] arr;

bubbleSort(Obj[] arr){

if(book[j] > book[j+1]){



Can't directly compare books.

}

}

Price → ✓ book[j].price > book[j+1].price →

✓ book[j].name > book[j+1].name

"java" ≯ "Python"

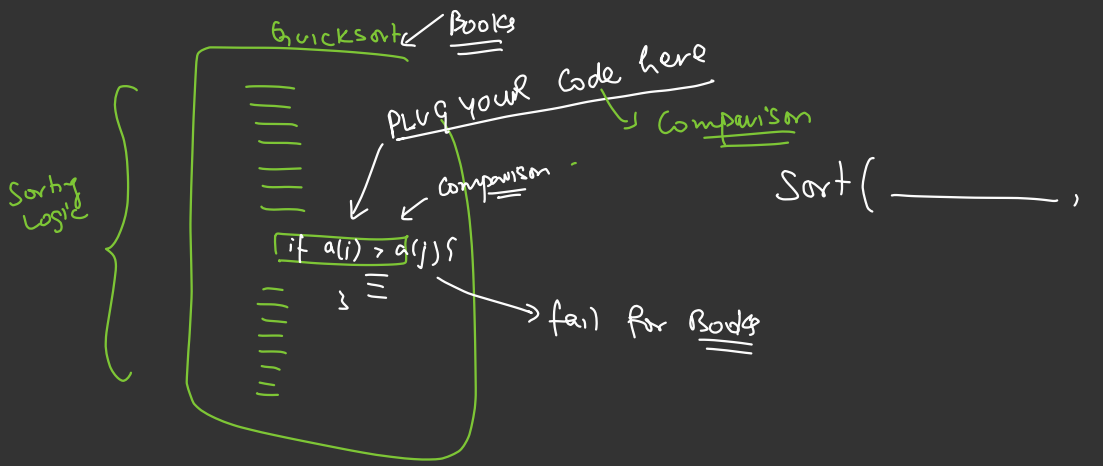
Strict order

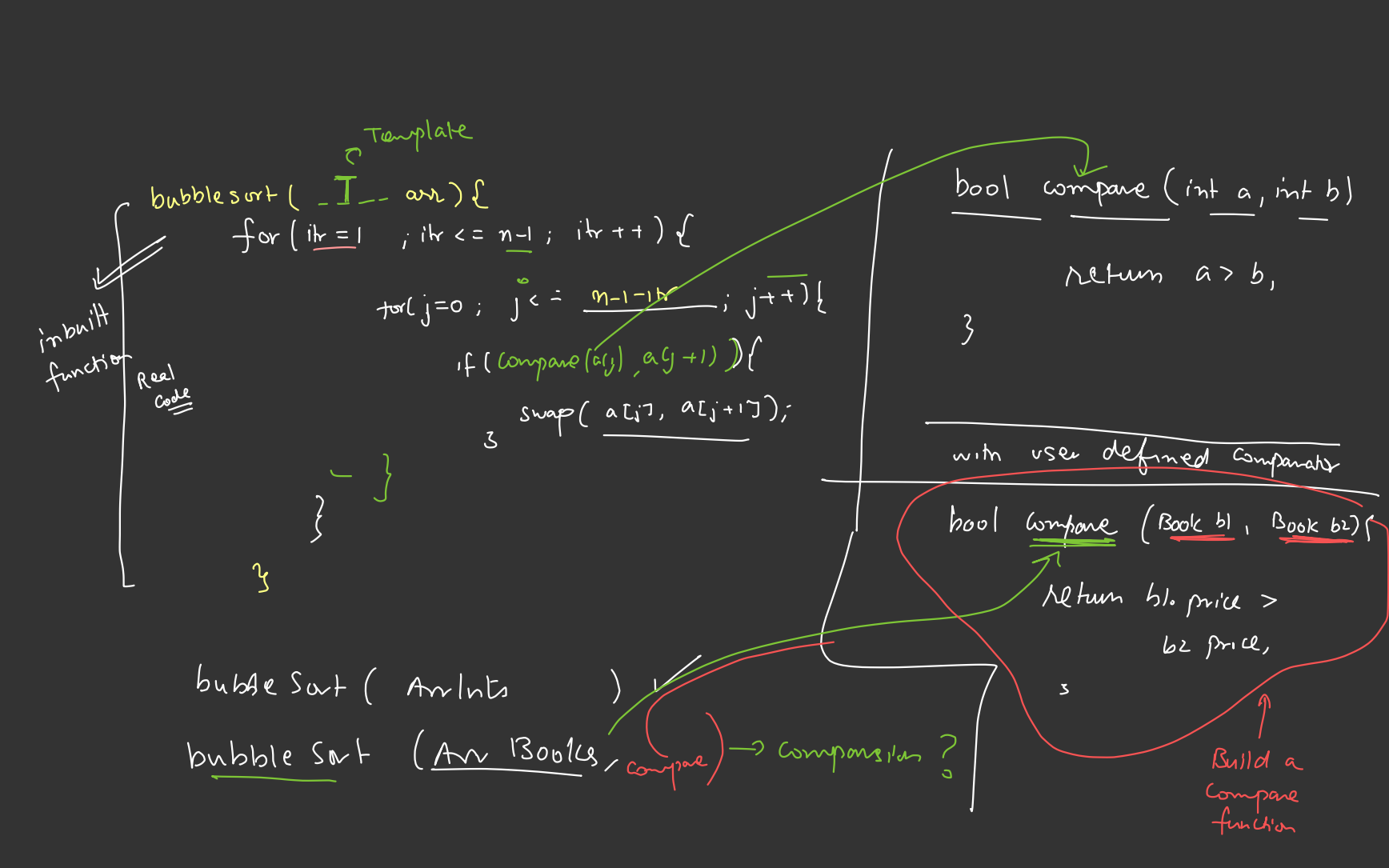
int1 > int 2

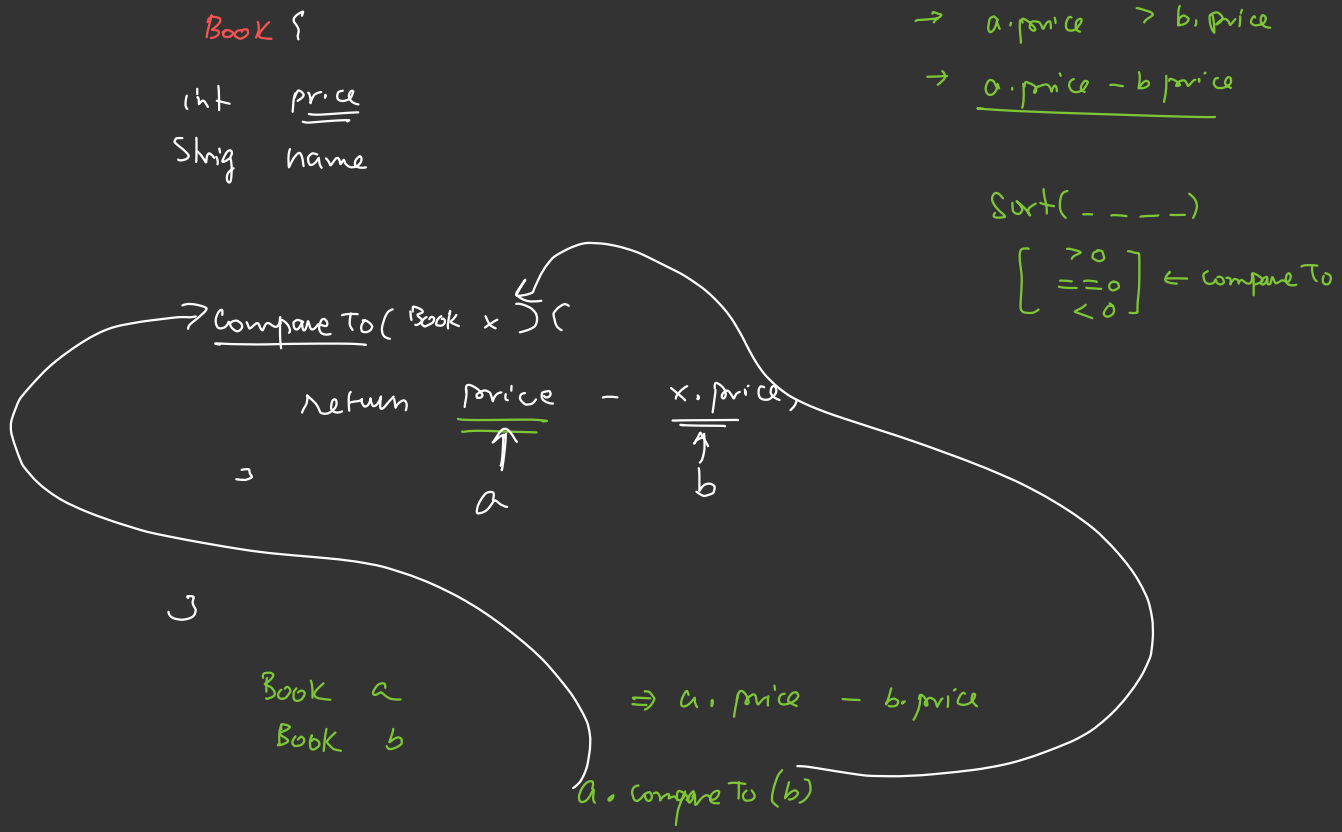
string1 > string 2

book1 > book 2

arrays of
For custom objects, we need to define criteria for "sorting"




Template
bubblesort (_I_ arr) {
for (itr = 1 ; itr <= n-1 ; itr++) {
for(j=0 ; j <= n-1-itr ; j++) {
if (compare (a[j] , a[j+1])){
swap(a[j], a[j+1]);
}
}
}
inbuilt function
Real Code
bool compare (int a, int b)
return a > b,
}
with user defined comparator
bool compare (Book b1 , Book b2){
return b1. price > b2 price,
}
bubble Sort (Arr Ints)
bubble Sort (Arr Books, compare) → Comparision ?
Build a Compare function


Book {
int price
String name
→ a.price > b.price
→ a.price - b.price
Sort(_ _ _ _)
[>0 ==0 <0] ← compareTo
compareTo(Book x){
return price - x.price;
a
b
}
}
Book a
Book b
⇒ a.price - b.price
a.compareTo(b)

Book {

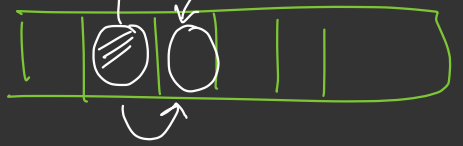
compareTo(Book x){
--- x.pages - pages,
}

if(x.pages == pages){
return price - x.price,
}

objects

Sort(Arr) {

Knows how to compare with other objects



object

Arr[j].compareTo(Arr[j+1])

must have name compareTo

→ Packing the comparison logic in object itself

}

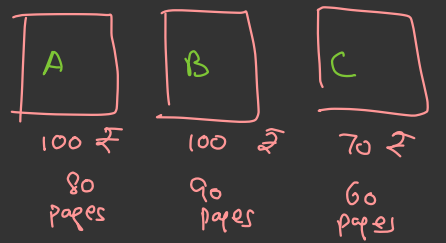
→ You can supply the ~~sorting~~ comparison logic

Complex Sort

Example →

```
int Compare (Book b1, Book b2) {
    if (b1.price == b2.price) {
        return b1.pages - b2.pages;
    }
    else {
        return b1.price - b2.price
    }
}
```

Rev it



Double Criteria

1. → Cheapest Book
2. → Same price → more pages

Price A = 100
Price B = 80

100 − 80 > 0

Longer →

Smaller →

Sort( ) {

if (Compare(---) > 0)

1, 2, 4 →

4, 2, 1 →

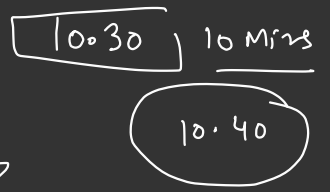
swap

Pack "compare" 'inside' object

TODO → Way 1 for Books

way-1 implementing "compareTo" method inside Book class

way-2 Build a "compare" fn outside object



Books ——
Book ——

price → User1
name → user2

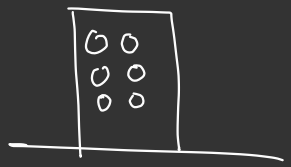
what ↓ How to compare

⇒ Sort (Books , Compare)

↑ no compare fn defined

Interface → list of methods which are empty

[Contract]



Govt

Contract {

fireSafety() {

= ? ? ?

}

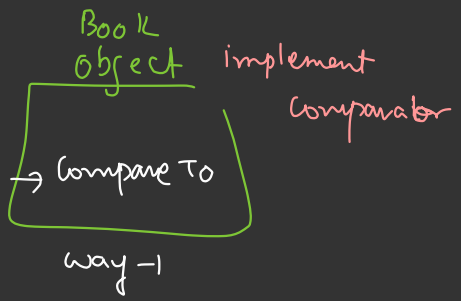
Builder implements Contract {

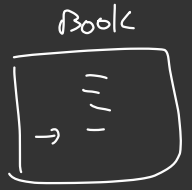
fireSafety() {

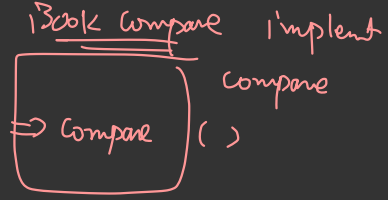
= = Put 5 extinguishers.

}

}







Sort (Book Arr)

Sort (Book Arr, BookCompare1)

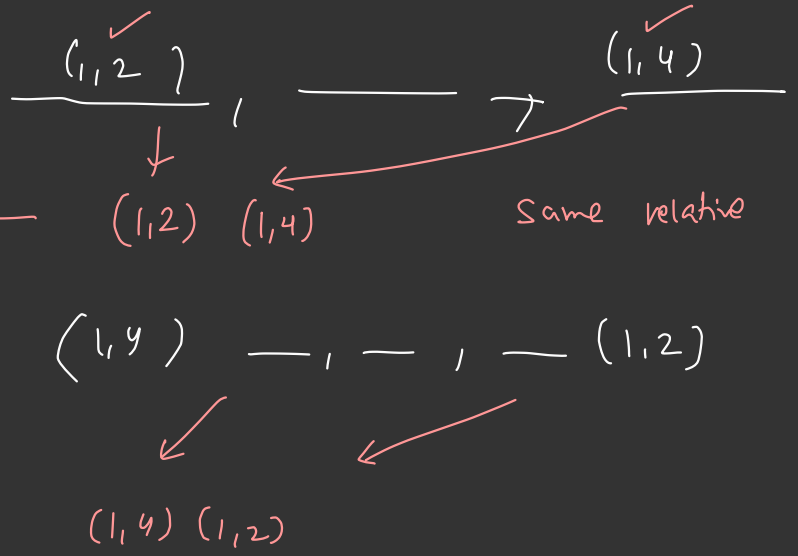
Sort (____, BookCompare2)

Sort (____, BookCompare3)

Bubble sort → Behaviour → Stable → maintain the relative ordering of same keys

Sorting → Unstable sort ⇒ change the order in any way.

Quicksort ↓

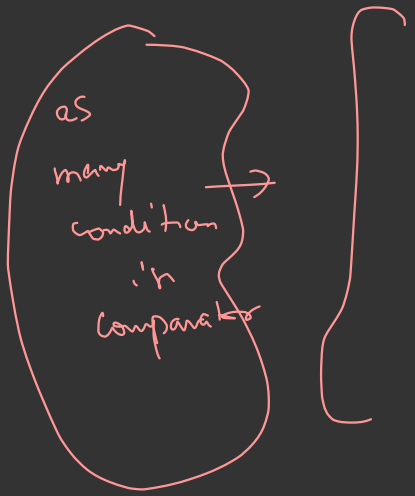


Base upon x-axis

(1,2) == (1,4)

Yes

x	y		x1	y2
(1,	2)	<	(1,	4)
p1			p2	




```
if (p1.x == p2.x) {
    return p1.y - p2.y;      // Compare y if x equal
} else {
    return p1.x - p2.x;      // Compare on x.
}
```

a fn which can be called without creating obj

Static fn

fn

Stack

main

Static data/variables (heap)

data which shared across all object of same class.

Deep Dive → OOPS