Input:

js
CopyEdit

```js
mergeSort([8, 3, 5, 4, 7, 6, 1, 2])
```

📊 Goal:

Fully sort this array using merge sort.

🔁 DETAILED RECURSIVE CALL TREE

scss
CopyEdit

```scss
mergeSort([8,3,5,4,7,6,1,2])
├─ mergeSort([8,3,5,4])                    ⬅ LEFT HALF
│   ├─ mergeSort([8,3])
│   │   ├─ mergeSort([8])                  ⬅ base case → returns [8]
│   │   └─ mergeSort([3])                  ⬅ base case → returns [3]
│   │   └─ merge([8], [3])                 → [3, 8]
│   └─ mergeSort([5,4])
│       ├─ mergeSort([5])                  ⬅ base case → returns [5]
│       └─ mergeSort([4])                  ⬅ base case → returns [4]
│       └─ merge([5], [4])                 → [4, 5]
│   └─ merge([3,8], [4,5])                 → [3, 4, 5, 8]
└─ mergeSort([7,6,1,2])                    ⬅ RIGHT HALF
    ├─ mergeSort([7,6])
    │   ├─ mergeSort([7])                  ⬅ base case → returns [7]
    │   └─ mergeSort([6])                  ⬅ base case → returns [6]
    │   └─ merge([7], [6])                 → [6, 7]
    └─ mergeSort([1,2])
        ├─ mergeSort([1])                  ⬅ base case → returns [1]
```

```
      └─ mergeSort([2])                       ⬅ base case → returns
[2]
      └─ merge([1], [2])                      → [1, 2]
   └─ merge([6,7], [1,2])                     → [1, 2, 6, 7]

FINAL merge([3, 4, 5, 8], [1, 2, 6, 7])        → [1, 2, 3, 4, 5, 6,
7, 8]
```

🧠 What Happens at Each Stage (with Stack-Like Behavior)

1. Call: `mergeSort([8,3,5,4,7,6,1,2])`

- Split into `left = [8,3,5,4]`, `right = [7,6,1,2]`

---

2. Call: `mergeSort([8,3,5,4])`

- Split into `[8,3]` and `[5,4]`

`mergeSort([8,3])`

- `[8]` → base
- `[3]` → base
- merge: `[3,8]`

`mergeSort([5,4])`

- `[5]` → base
- `[4]` → base
- merge: `[4,5]`

merge: `[3,8]` and `[4,5]` → `[3,4,5,8]`

---

3. Call: `mergeSort([7,6,1,2])`

- Split into `[7,6]` and `[1,2]`

`mergeSort([7,6])`

- `[7]` → base
- `[6]` → base
- merge: `[6,7]`

`mergeSort([1,2])`

- `[1]` → base
- `[2]` → base
- merge: `[1,2]`

merge: `[6,7]` and `[1,2]` → `[1,2,6,7]`

✅ FINAL MERGE:

js
CopyEdit
```
merge([3, 4, 5, 8], [1, 2, 6, 7]) → [1, 2, 3, 4, 5, 6, 7, 8]
```

🖥️ Console Logs (if you added them):

sql
CopyEdit
```
merge([8], [3])       => [3, 8]
merge([5], [4])       => [4, 5]
merge([3,8], [4,5])   => [3, 4, 5, 8]

merge([7], [6])       => [6, 7]
merge([1], [2])       => [1, 2]
merge([6,7], [1,2])   => [1, 2, 6, 7]

merge([3,4,5,8], [1,2,6,7]) => [1, 2, 3, 4, 5, 6, 7, 8]
```

📌 Final Notes:

- ✅ **Each mergeSort() call works on its own mini-array.**
- ✅ The **left is always resolved fully first** (goes all the way to base case).
- ✅ Only after left is returned, the **right is processed**.
- ✅ Then the `merge()` happens.

Would you like me to **generate an image** of this full merge tree with arrows and boxes like a diagram for better memory?

You said:
also show the call stack diagram

ChatGPT said:

Awesome! Let's now visualize the **Call Stack diagram step-by-step** for the input:

scss
CopyEdit
```
mergeSort([8, 3, 5, 4, 7, 6, 1, 2])
```

We'll simulate how the **JavaScript call stack** grows and shrinks as recursive calls go deeper and return. This is what happens *internally in memory*.

🧠 Reminder:

In **JavaScript**, the **Call Stack** is LIFO (Last In, First Out).
Each time `mergeSort()` is called, it is **pushed onto the stack**.
When it **returns**, it is **popped off the stack**.

📘 Legend:

- 📥 = Call pushed on stack
- ✅ = Returned and popped from stack

🟢 START

Initial call:

📥 `mergeSort([8, 3, 5, 4, 7, 6, 1, 2])`

lua
CopyEdit

```
|-------------------------------|
| mergeSort([8,3,5,4,7,6,1,2])  |
|-------------------------------|
```

🔽 Go Left: `[8,3,5,4]`

📥 `mergeSort([8,3,5,4])`

lua
CopyEdit

```
|-------------------------------|
| mergeSort([8,3,5,4])          |
| mergeSort([8,3,5,4,7,6,1,2])  |
|-------------------------------|
```

📥 `mergeSort([8,3])`

📥 `mergeSort([8])` → ✅ base case

📥 `mergeSort([3])` → ✅ base case

📥 `merge([8], [3])` → ✅ returns `[3, 8]`

lua
CopyEdit

```
|-------------------------------|
| mergeSort([8,3,5,4])          |
| mergeSort([8,3,5,4,7,6,1,2])  |
|-------------------------------|
```

📥 `mergeSort([5,4])`

📥 `mergeSort([5])` → ✅ base case
 📥 `mergeSort([4])` → ✅ base case
 📥 `merge([5], [4])` → ✅ returns `[4,5]`

📥 `merge([3,8], [4,5])` → ✅ returns `[3,4,5,8]`

lua
CopyEdit

```
|-------------------------------|
| mergeSort([8,3,5,4,7,6,1,2])  |
|-------------------------------|
```

🔽 Now Go Right: `[7,6,1,2]`

📥 `mergeSort([7,6,1,2])`

lua
CopyEdit

```
|-------------------------------|
| mergeSort([7,6,1,2])          |
| mergeSort([8,3,5,4,7,6,1,2])  |
|-------------------------------|
```

📥 `mergeSort([7,6])`

📥 `mergeSort([7])` → ✅ base case
 📥 `mergeSort([6])` → ✅ base case
 📥 `merge([7], [6])` → ✅ returns `[6,7]`

📥 `mergeSort([1,2])`

📥 `mergeSort([1])` → ✅ base case
 📥 `mergeSort([2])` → ✅ base case
 📥 `merge([1], [2])` → ✅ returns `[1,2]`

📥 `merge([6,7], [1,2])` → ✅ returns `[1,2,6,7]`

lua
CopyEdit

```
|-------------------------------|
| mergeSort([8,3,5,4,7,6,1,2])  |
|-------------------------------|
```

📥 `merge([3,4,5,8], [1,2,6,7])` → ✅ returns `[1,2,3,4,5,6,7,8]`

🎉 Call stack is now **empty** → Final result is returned!

🧱 Call Stack Growth and Shrink (Simplified Steps)

css
CopyEdit

```
➡️ Pushed: mergeSort([8,3,5,4,7,6,1,2])
➡️ Pushed: mergeSort([8,3,5,4])
➡️ Pushed: mergeSort([8,3])
➡️ Pushed: mergeSort([8]) → returned
➡️ Pushed: mergeSort([3]) → returned
➡️ merge([8],[3]) → returned [3,8]
➡️ Pushed: mergeSort([5,4])
➡️ Pushed: mergeSort([5]) → returned
➡️ Pushed: mergeSort([4]) → returned
➡️ merge([5],[4]) → returned [4,5]
➡️ merge([3,8],[4,5]) → returned [3,4,5,8]
⬅️ Left done!

➡️ Pushed: mergeSort([7,6,1,2])
➡️ Pushed: mergeSort([7,6])
➡️ Pushed: mergeSort([7]) → returned
➡️ Pushed: mergeSort([6]) → returned
➡️ merge([7],[6]) → returned [6,7]
➡️ Pushed: mergeSort([1,2])
➡️ Pushed: mergeSort([1]) → returned
➡️ Pushed: mergeSort([2]) → returned
➡️ merge([1],[2]) → returned [1,2]
➡️ merge([6,7],[1,2]) → returned [1,2,6,7]
⬅️ Right done!

➡️ Final merge([3,4,5,8], [1,2,6,7]) → returned [1,2,3,4,5,6,7,8]
✅ Done!
```

✅ Final Output:

js

CopyEdit

```
[1, 2, 3, 4, 5, 6, 7, 8]
```