## Merge Sort:

Divide and Conquer Method:

1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_ the problem in smaller problems
2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_ these problems
3. \_\_\_\_\_\_\_\_\_\_\_\_\_\_ the answers

How does it relate to Merge Sort?

Merge Sort:

1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_ the problems into 2 halves
2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_ each half
3. \_\_\_\_\_\_\_\_\_\_\_\_\_\_ the sorted halves

Ex:

Array: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Iterations: |  |  |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |  |  |

Code:

TC:

Recurrence Formula and Recursion Tree:

……………….

……

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Level | Arg/Problem Size | Nodes per Level | 1 Node TC | Level TC |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

SC:

Stable? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Adaptive? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Usable Data Types:

Variations:

int binary\_search**(**int A**[],** int left**,** int right**,** int v**){**

int m **=** left**+(**right**-**left**)/**2**;**

**if** **(**left **>** right**)** **return** **-**1**;**

**if** **(**v **==** A**[**m**])** **return** m**;**

**if** **(**v **<** A**[**m**])**

**return** binary\_search**(**A**,** left**,** m**-**1**,** v**);**

**else**

**return** binary\_search**(**A**,** m**+**1**,** right**,** v**);**

**}**

Merge\_sort**(**A**,** le**,** r**)** //N = ri-le+1

**if** **(**le**>=**ri**)** **return**

**else**

m **=** floor**(**le**+(**ri**-**le**)/**2**)**

Merge sort**(**A**,** le**,** m**);**

Merge\_sort**(**A**,** m**+**1**,** ri**);**

Merge**(**A**,** le**,** m**,** ri**);**

Merge**(**A**,** le**,** m**,** ri**)**

1 n1**=**m**-**le**+**1**+**1 // +1 for inf

2 n2**=**ri**-**m**+**1 // +1 for inf

3 let L**[**n1**],** R**[**n2**]** be arrays

4 **for** j**=**0 to n1**-**2

5 L**[**j**]=**A**[**le**+**j**]**

6 **for** j**=**0 to n2**-**2

7 R**[**j**]=**A**[**m**+**1**+**j**]**

8 L**[**n1**]** **=** inf

9 R**[**n2**]** **=** inf

10 j**=**0**,**

11 i**=**0

12 **for** k**=**le to ri

13 **if** L**[**i**]** ≤ R**[**i**]**

14 A**[**k**]=**L**[**i**]**

15 i**++**

16 **else**

17 A**[**k**]** **=** R**[**j**]**

18 j**++**

// pseudocode

// - indentation => instruction group in {}

// - loops:

// for k=le to ri

// for(k=le; k<=ri; k++)