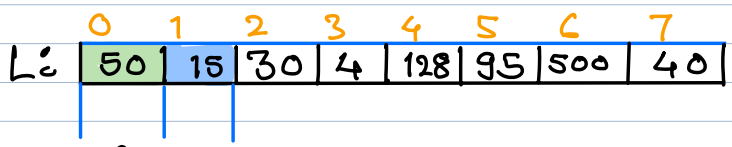
```
merge(L: list[int], p: int, q: int, r: int)

PRECONDITIONS:
    1) p <= q < r
    2) L[p...q] must be sorted.
    3) L[q+1...r] must be sorted.

POSTCONDITION:
    1) L[p...r] is sorted.
```



merge(L, 0, 0, 1)

p=0, q=0, r=1

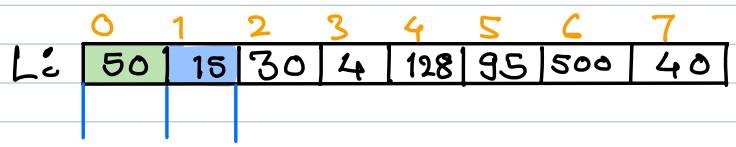
1) $0 \le p \le q < r$

$0 \le 0 \le 0 < 1$

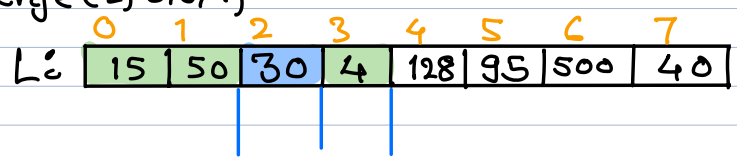
∴ $L[p \dots q] == L[0 \dots 0] \equiv L[0]$ ... sorted -- (2)

$L[q+1 \dots r] == L[1 \dots 1] \equiv L[1]$ ... sorted -- (1)

$L[p \dots r]$ is sorted. $L[0 \dots 1]$



merge(L, 0, 0, 1)



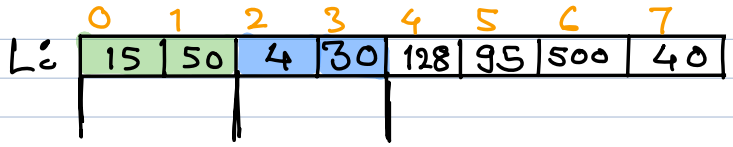
L[2..2] -- sorted

L[3..3] ... sorted

merge(L, 2, 2, 3) $p \le q < r$

p=2, q=2, r=3

p...q, 2..2, q+1...r, 3...3



merge(L, 0, 1, 3)

p = 0, q = 1, r = 3.

1) $0 \leq p \leq q < r < len(L)$

$0 \leq 0 \leq 1 < 3 < 8$ = True.

2) L[p...q] must be sorted

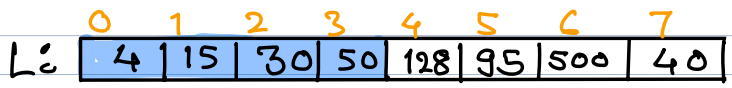
L[0...1] must be sorted. — (it is sorted)

3) L[q+1...r] must be sorted.

L[2...3] must be sorted. (it is sorted)

Post:

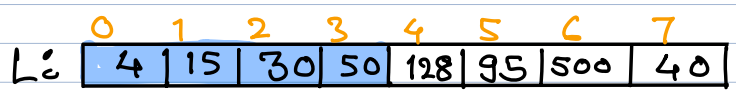
L[p...r] is sorted. L[0...3] is sorted



Recap: merge(L, 0, 0, 1) → L[0...1] sorted

merge(L, 2, 2, 3) → L[2...3] sorted

merge(L, 0, 1, 3) → L[0...3] sorted.



merge(L, 4, 4, 5)
p q r

p...q : 4..4 | q+1...r | 5...5 | p...r | 4...5

$0 \leq p \leq q < r < 8$

↓

	0	1	2	3	4	5	6	7
L:	4	15	30	50	95	128	500	40

merge (L, 6, 6, 7) | p..q → 6..6 | q+1..r → 7..7 | p..r | 6..7

$0 \le p \le q < r < 8$ | True. | L[6].. Sorted L[7] -- sorted

L[6..7] → sorted

	0	1	2	3	4	5	6	7
L:	4	15	30	50	95	128	40	500

merge(L, 4, 5, 7)

$p=4, q=5, r=7$ | $0 \le p \le q < r < 8$ | $0 \le 4 \le 5 < 7 < 8$

L[p..q) | L[4..5] sorted

L[q+1..r] | L[6...7] sorted

L[p... r] | L[4...7) sorted

	0	1	2	3	4	5	6	7
L:	4	15	30	50	40	95	128	500

merge(L, 0, 3, 7).

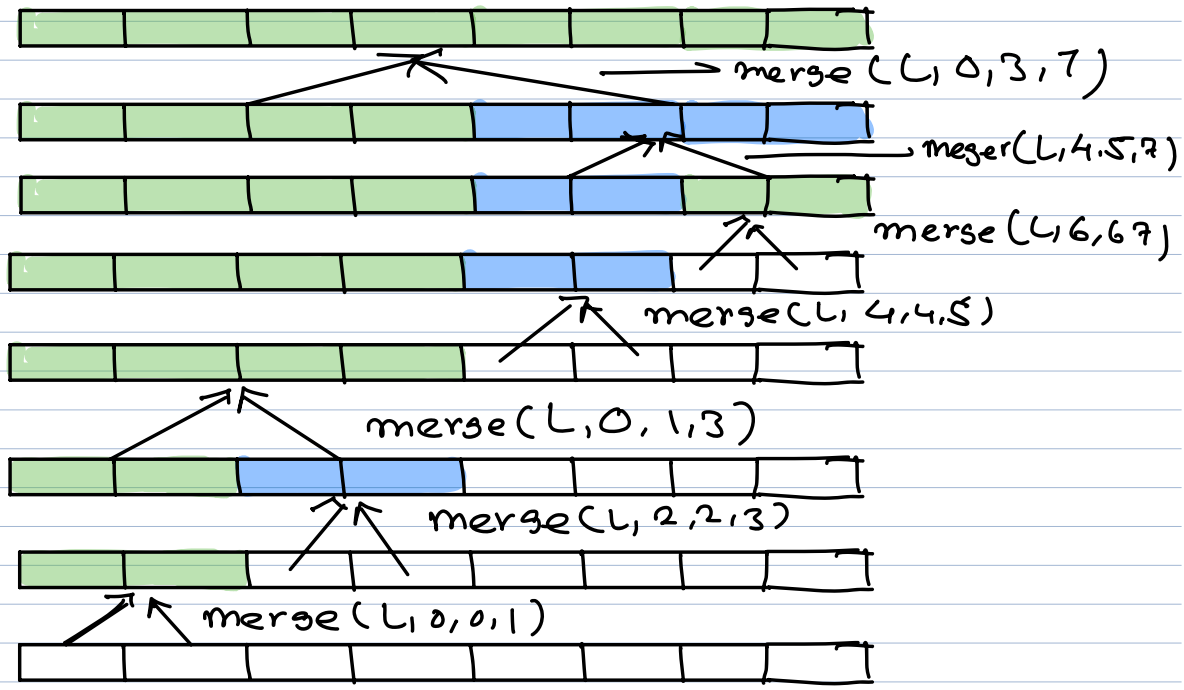
$p=0, q=3, r=7$ | $0 \le p \le q < r < len(L)$

$0 \le 0 \le 3 < 7 < 8$ — true.

L[p..q) == L[0...3] Sorted ✓.

L[q+1..r) == L[4...7] sorted ✓.

L[p...r] == L[0...7) sorted.




```
for i in range(?):
    merge(L, ?, ?, ?)
```

there is no general formula.

Iterative Solution ⟶ Recursive Solution.

Recursive Solve ⟶✗ Iterative Solution alone.

Iteration + Auxillary User Defined Stack ≡ Recursion

p : None
q : None
r : None
} — ?

i = 0

while i < 7:

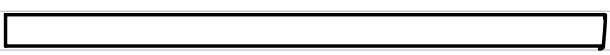
merge(L, p, q, r)

p
q
r
} modify ??

i = i+1

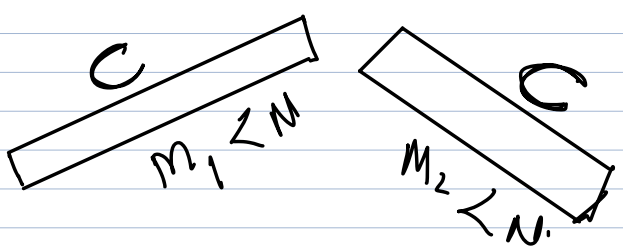
---

1) Divide And Conquer.



Input Data : Size = N.

Computational Problem – C Solve.

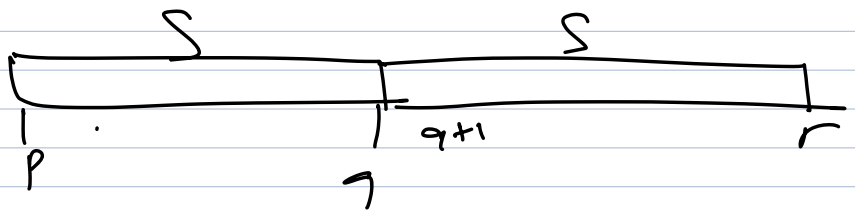


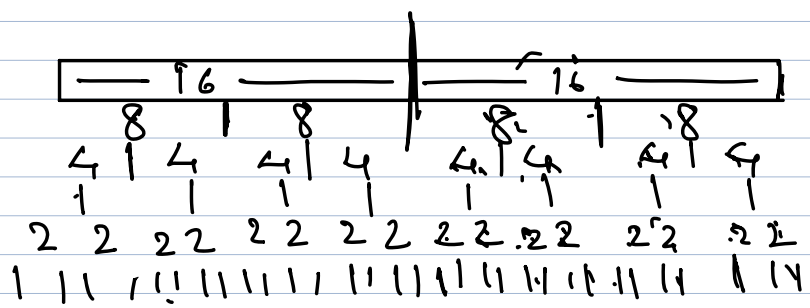
$M1 + M2 = N$

C is solved on M_1

C is solved on M2.

$(M1 + M2) \rightarrow C$ – solve.





32 element

C

N

M1

M2

C Solved

C Solved

M1+M

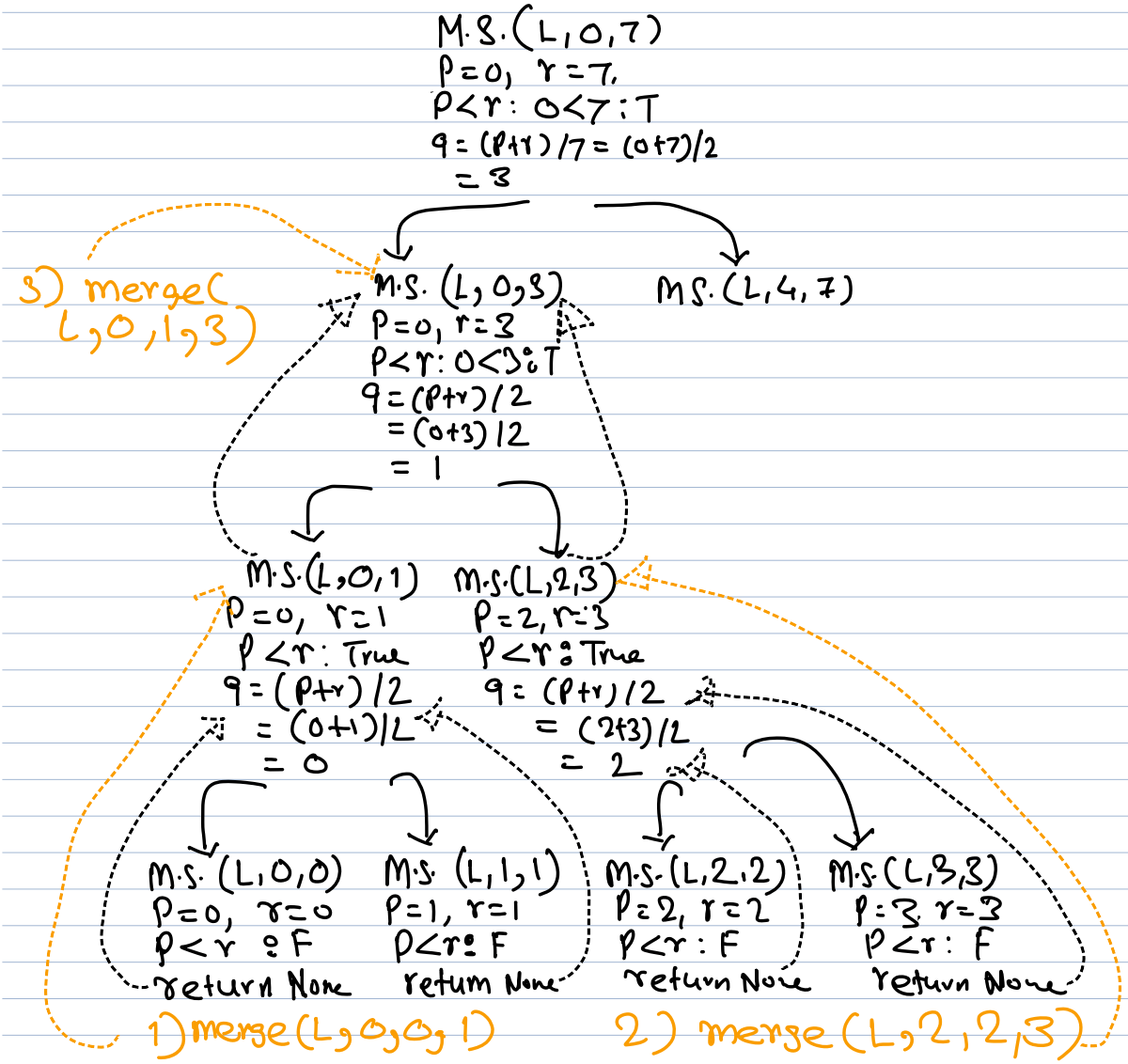
C Solved

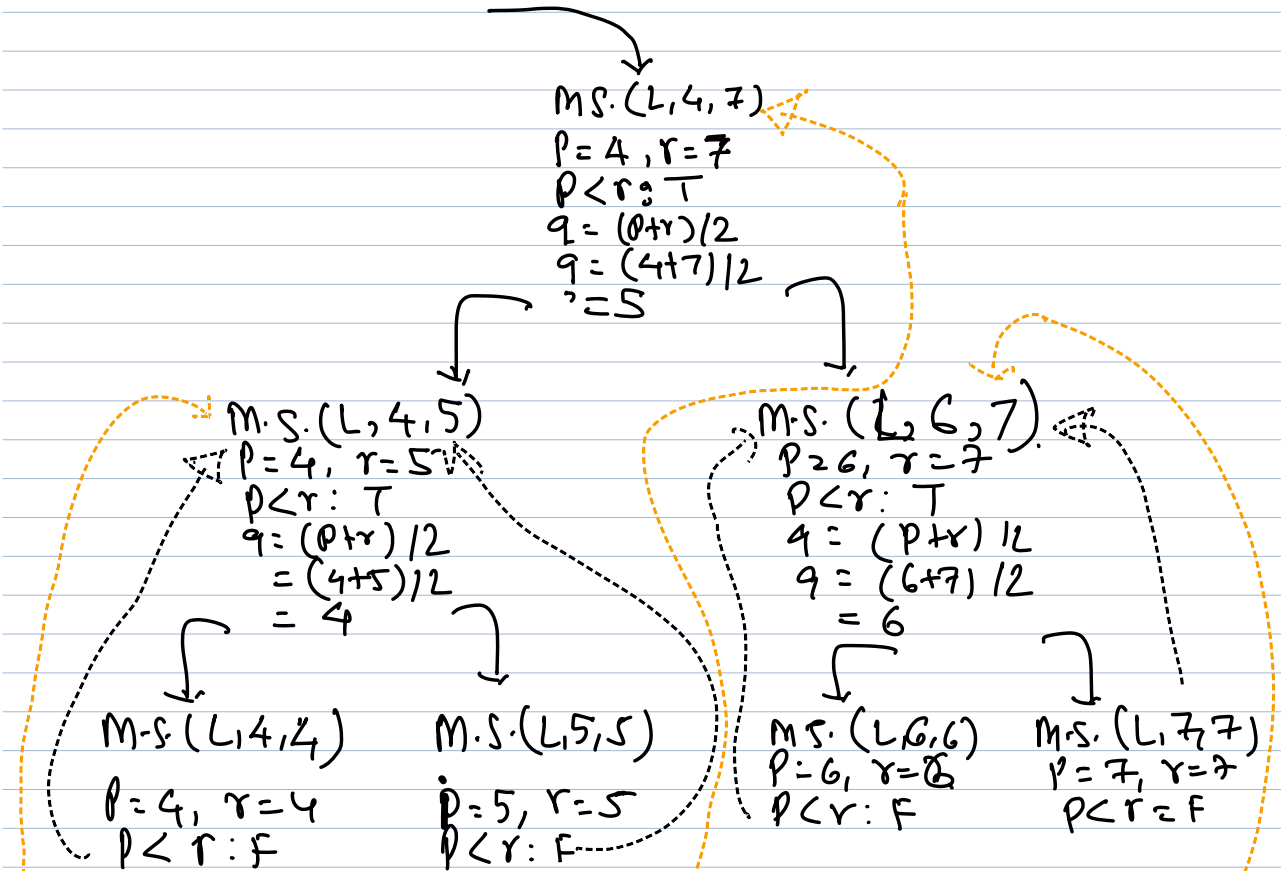
$$f(L, p, r) = \begin{cases} q = (p+q)/2 \\ f(L, p, q) \\ f(L, q+1, r) \\ M(L, p, q, r) & \text{if } p < r \\ \text{STOP} & \text{if } p \geq r. \end{cases}$$

```
def merge_sort(L: list[int], p:int, r: int):
    if p < r:
        q = (p+r) // 2
        merge_sort(L, p, q)
        merge_sort(L, q+1, r)
        merge(L, p, q, r)
```

---

merge_sort(L, 0, 7)


M.S.(L,0,7)
p=0, r=7,
p<r: 0<7: T
q=(p+r)/7 = (0+7)/2
=3
M.S.(L,0,3)
p=0, r=3
p<r: 0<3: T
q=(p+r)/2
=(0+3)/2
=1
M.S.(L,4,7)
M.S.(L,0,1)
p=0, r=1
p<r: True
q=(p+r)/2
=(0+1)/2
=0
M.S.(L,2,3)
p=2, r=3
p<r: True
q=(p+r)/2
=(2+3)/2
=2
M.S.(L,0,0)
p=0, r=0
p<r: F
return None
M.S.(L,1,1)
p=1, r=1
p<r: F
return None
M.S.(L,2,2)
p=2, r=2
p<r: F
return None
M.S.(L,3,3)
p=3, r=3
p<r: F
return None
3) merge(L,0,1,3)
1) merge(L,0,0,1)
2) merge(L,2,2,3)

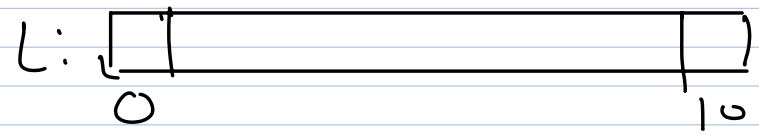


4) Merge(L, 4, 4, 5)

5) merge(L, 6, 6, 7)

6) Merge(L, 4, 5, 7)

7) Merge(L, 0, 3, 7).



M.S. (L, 0, 10)
q = 5

MS (0,5)
q = 2

MS (6, 10)

MS (0,2)
q = 1

L(3,5)
q = 4

MS (0,1)
q = 0

MS (2,2)

MS(0,0)

MS(1,1)

m(0,0,1)

m(0,1,2)