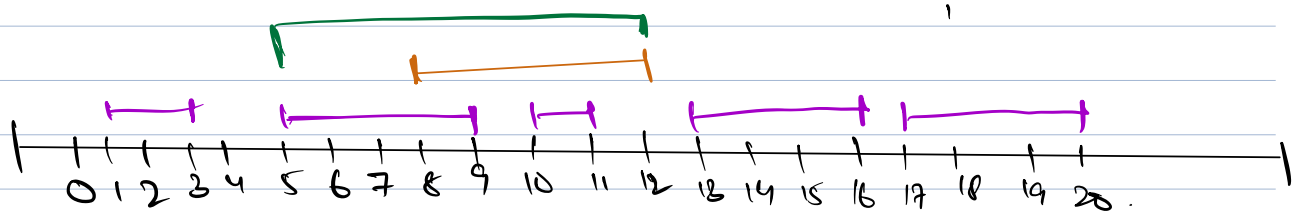
Merging intervals

Given a sorted set of non-overlapping intervals. Insert a new interval. & resulting interval is also sorted & non-overlapping

ex: s → [1 5 10 13 17]
e → [3 9 11 16 20]



Insert → [8, 12]

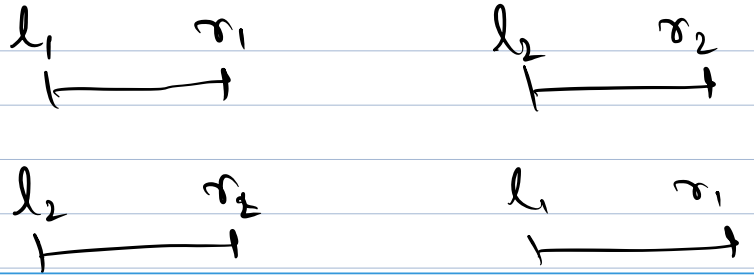
s → [1 5 13 17]
e → [3 12 16 20]

l_1 r_1 l_2

Non-overlapping.

$(r_1 < l_2$ or $r_2 < l_1)$

Non-overlapping



Overlapping

$!(r_1 < l_2$ or $r_2 < l_1)$

Traverse intervals from left to right & check for overlapping with the new interval (we want to insert)

i → first overlapping idx (interval)

j → last overlapping idx / interval.

→ Non-overlapping → put directly in ans.

start → min(s[i], s_{new})

end → max(e[j], e_{new})

TC = O(N)
SC = O(1)

arr = [[1,2] [3,6] [5,10] [2,15]]
0 1 2 3

[1, 3, 5, 2]
[2, 6, 10, 15]

s_{new} s_{end}

```
for(i=0 ; i< N ; i++) {
    // overlap check.
    if !( (arr[i][1] < s_new) || (s_end < arr[i][0]) )
```

Q Given N distinct arr[] ; when 0 ≤ arr[i] ≤ N

replace arr[i] = arr[arr[i]]

[0, N-1]

ex: [3 1 2 4 4 0 1 0]

0 1 2 3 4

arr[arr[0]]
arr[3]
= 1

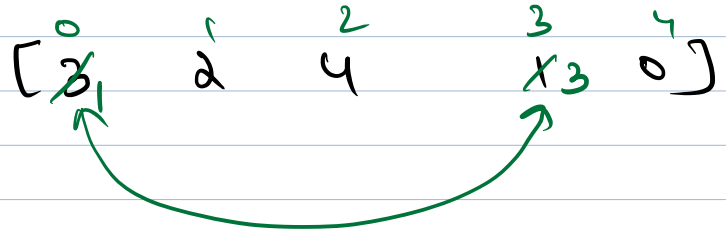
arr[i] = arr[arr[i]]

[1, 4, 0, 2, 3]

arr[1] = arr[arr[1]] = arr[2] = 4
arr[2] = arr[arr[2]] = arr[4] = 0
arr[3] = arr[arr[3]] = arr[1] = 2
arr[4] = arr[arr[4]] = arr[0] = 3

Do you think I will let you use an extra Array ?

→ Cant use an extra array



arr[4] = arr[arr[4]] = arr[0] = ~~3~~

→ Cannot swap

ex: arr[7] = [3 1 4 6 5 0 2]
0 1 2 3 4 5 6

Ans- = [6 1 5 2 0 3 4]

ex: arr[7] = [1 6 3 5 4 2 0]
0 1 2 3 4 5 6

Hint: Somehow at a[i], we need to store both new & old value.

Big Bang Theory	Day	Hour
	0	
23	0	23
40	1	16
100	4	4
125	5	5
200	8	8
x	x/24	x%24

↳ Aren't we storing 2 values in a single variable

$\frac{x}{24}$ → Quotient → days.

x%24 → Remainder → hour.

$0 \le x < 24$

$0 \le a[i] < N.$ ←

↳ [0, N-1]

All values in A[i] will be % N.

x → /24, %24.

arr[i] → /N → old
arr[i] → %N → new

old | new
new | old

↳ To DO.

x → /N → old
x → %N → New

$x = (old * N) + new$

$x/N = old$

$x\%N = new$

$\frac{old * N + new}{N} < N$

$\frac{old * N}{N} + \frac{new}{N}$ → [0, N-1]

$old + \frac{new}{N}$ → greater than 1 or lesser than 1 → 0

$\frac{x}{N} = old$

$x = old * N$

$\frac{N-1}{N}$

$\frac{old * N}{N}$

$\frac{old}{N} = 0$ → [0, N-1]

* $x\%N = new\%N$

$(old * N + new)\%N = new$

$(N + new)\%N = old$

$(old * N + new) \% N = new$

$(old * N) \% N + (new \% N)$

$(old * N) \% N = 0$

$new \% N = new$

$(4 \times 10) \% 10 \quad 0$

$40 \% 10 = 0 \quad =$

$x = (old * N) + new$

	0	1	2	3	4	5	6
arr[7] =	3*7	1*7	4*7	6*7	5*7	0*7	2*7
	+6	+1	+5	+2	+0	+3	+4

→ Multiply all values by 7.
→ old value → /7
→ New value → %7.

	idx.	value
arr[0] = arr[arr[0]]	$\frac{21}{7} = 3$ arr[3]	arr[3] = 6
arr[1] = arr[arr[1]]	arr[1] = 1	arr[1] = 1
arr[2] = arr[arr[2]] =	arr[2] = 4	arr[4] = 5

$21 + 6 = 27 \% 7 = 6$

$\frac{27}{7} = 3$

TC → O(N)
SC → O(1)

Break = 11:03

Q → Given a string of distinct chars. Find the sorted permutation rank of that string.

ex: BCDA → 18

A B C D

1. A B C D
2. A B D C
3. A C B D
4. A C D B
5. A D B C
6. A D C B
7. B A C D
8. B A D C
9. B C A D
10. B C D A

A _ _ _ → 3 × 2 × 1 = 3!

B C D A → 2! = 11!

3! + 2! + 1! + 1
6 + 2 + 1 + 1 = 10

sorted → R S T

str → S T R R

R _ _ _ → 2!

S R _ _ → 1!

S T R _ → +1

1 R S T
2 R T S
3 S R T
4 S T R

2! + 1! + 1 = 2 + 1 + 1 = 4

① R _ _ _ → (3!/2!)

% 3000

Inverse Modulo

② S _ _ _