a.b.c.d → "a[.]b[.]c[.]d"

""

"a" 1

"a[.]" 2

"a[.]b" 3

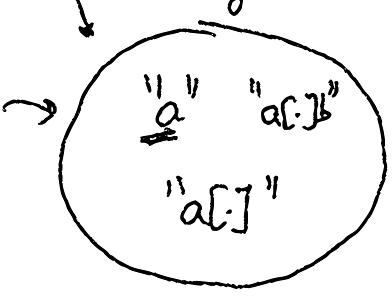
"a[.]b[.]" → $O(n(\frac{n+1}{2}))$

~ $O(n^2)$

"a[.]b[.]c" n

Java/Python

String → Immutable



C++

String → mutable

"a"

"a[.]"

```
l = ["a", ".", "b", ".", "c", ".", d]

ans = [ ]

for i, x in enumerate(l):
    if l[i] != ".":
        ans.append( .. ]
    else ..
        ans.appe("[.]")

split()

ans = ["a", "[.]", "b", "[.]", ....]

return "".join(ans)
```

$$T(n) = \underbrace{O(1) + O(1) + \dots O(1)}_{\text{"n" times}}$$

$$\simeq \boxed{O(n)}$$

```
"[.]".j([a, b, c, d])

→ "a[.]b[.]c[.]d"
```

l = [1, 2, 1]

↓

return → [1, 2, 1, 1, 2, 1]

assumption:

new list

return

ans = [1

d = [1, 2, 1]

a = return list = [1, 2, 1, 1, 2, 1]

↑ i (0, n-1) s.t. n → len(l]

lis → 0, 2n-1 s.t. 2n → len(result_l)

a[i+n] == a[i]

nums O(2n)

~ O(n)

ans = [1, 2, 1]
↑

for loop i → 0 ; n-1

ans.append (nums[i])

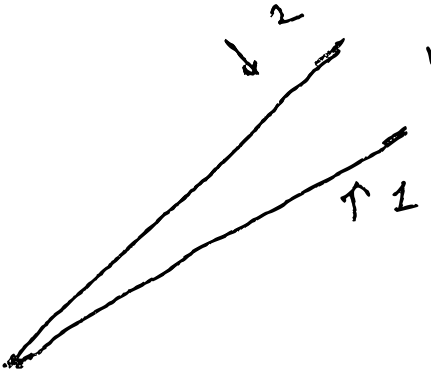
return ans.

```
for i in range(n):
    l.append(l[i])
ret l
```

← O(n)

~ 10 n

assumption: modify input



$O(2n) \sim O(n) \sim 20ms$

$O(n) \sim 10ms$

0, 1, 2, 3, 4, 5

nums = [1, 2, 3, 1, 1, 3]

good → (i, j) s.t i<j

nums[i] == nums[j]

(0, 3)
(0, 4) } 1==1
(3, 4)

(2, 5) 3==3

n → len(num)

i<j

```
→ ans = 0
→ for j in range(n):          start end
→     for i in range(0, j):
→         if nums[i] == nums[j]:
→             ans += 1
```

Start ↓ end-1

0+1+2 - - - - n-1

retun ans

→ $O(n^2)$

{1, 2, 3, 1, 1, 3]

↑ ↑

j=0 j=1

→ [ ]

0 $\underline{j=0}$

j=

```
→ for i in range(0, n):
→     for j in range(i+1, n):
          if ~~ans~~ -- -- :
              ans += 1.
```

n-1

+

n-2

+

.

.

3

+

2

+1

≈ O(n^2)

$[1, 1, 1, 1]$ "K"

(i, j)

3

2

1

$3+2+1 = 6$

"K"

$K-1$

$K-2$

$\vdots$

3

2

1

$$\sum_{i=1}^{K-1} i = \frac{K(K-1)}{2}$$

$=$

$1 \,|\, 2 \,|\, 3$

$\downarrow \quad \downarrow \quad \downarrow$

$3 \,|\, 1 \,|\, 2$

$\frac{3\times 2}{2} + 0 + 1 \rightarrow 4$

$= 3$

"1"

```
for i in rang (n):
```

d = { }

nums = [1,2,3,1,1,3]

ordering

Search/get : O(1) avg

insert : O(1)

remove : O(1)

update : O(1)

"n"

O(n)

```
for num in nums:
    d[num] = d.get(num, 0) + 1
```

O(1)

O(n)

d= {1 : 3,
2 : 1,
3 : 2}

{2: 1,
1:3,
3: 2}

```
ans = 0
for key in d :
    value = d[key]
    ans += value (value - 1)
return ans
```

$O(1)$

$d[c]$

$d \to$ "K"

$O(K)$

$O(n^2)$

$$O(n) + O(k)$$

$$= O(n+k)$$

$$\sim O(n+n)$$

$$\sim O(n)$$