(2) Next Permutation :

arrangement of the elements

arr[] = [3 1 2]

ie next element that will appear if we go by dictionary order.

1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1

nxt permutn.

for arr[] = [3 2 1].

↓ nxt permutn.
if not available
then go for smallest permutn

(M1) Brute

generate all permutn in sorted order.
locate the current permutn using linear search
ans = nxt permutn.

T.C: N!

(M2) → C++ STL → modify the array itself to nxt permuth.

Code

```
next_permutation (A.begin(), A.end());
```

(M3) Optimal :-

Observation :

(1) next/later elements in dictionary tend to have a longer prefix match.
eg: raj, ram, ran

eg: arr[] =

[2 1 5 4 3 0 0]

nxt permuth will have a longer prefix match.

↳ So fix some prefix & with later part → see if we can create a permuth greater than it

Basic Idea

q: [2 1 5 4 3 0 0]
fix

↓ cannot form smt greater

2 1 5 4 300
fixed

↳ 003
030 } X cannot create >300.

2 15 4300
fixed rem

↳ cannot create a permutn with 4300 which is > 4300

21 54300
fixed rem → cannot create >54300

2 154300 → greatest permutn can generate is:
fixed rem.

↓ 543100

001345

↓ X

↓ 453100

Bcz its not >154300.

2 (1) 5 4 3 0 0

fixed ↓ ← till here we couldn't create a permutn to create a greater no.

make change at this posn (to swt just greater than 1) → for just the nxt greater permutn.

i.e (3)

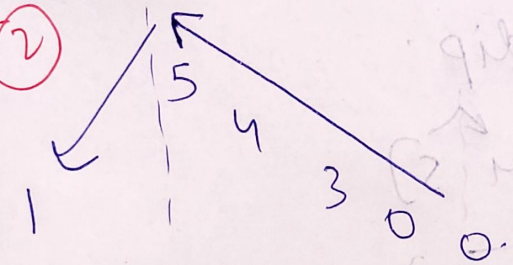
Q) why is this the soln?

↓

bcz,

at the right of 1 → I've nos > 1 so we can create a greater permutn

observn (2)



⇒ then figure out what is the nxt permutn.

→ figure out the 1st dip from the right (bcz for just greater permutn from that posn, keeping the longest prefix constant)

→ & at the right of ith elem (i.e dip elem) find > ith elem.

→ choose the smallest elem > dip elem. from the right elems.

dip.

arr[] = [2 1 | 5 4 3 0 0]

(place here some no just >1 from right for nxt permutn ie 3

2 3

observ ③

fill it with

5 4 0 0 1.

i.e smallest permutn of it

ie in sorted order.

Approach:

S1) find the dip.

[1 2 3 4 5]

No dip so lex. greatest permutn ⇐ [5 4 3 2 1]

→ so reverse it for ans.

S2) find the smallest elem greater than dip elem & since last to front in inc order

swap it ⇐

[ ]

search from back

i.e 2 1 | 5 4 3 0 0.

find the smallest no >1 & swap them.

S3) 2 3 | 5 4 1 2 0 0.

M2) sort this portion

M1) this is maintained

So just reverse it