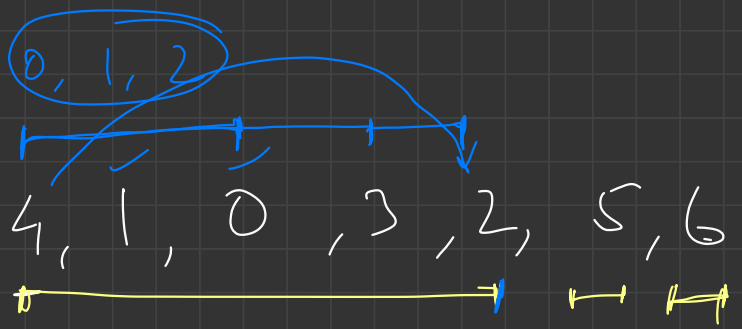
n = 7



sort 0, 1, 2, 3, 4, 5, 6.

index 0 1 2 3 4 5 6

→ go from left to right and keep track of max. value

→ if max value = i (index) ⇒ we can split at this point

4 digits

↳ possible arrangements

4! (24)

a, b, c, d.

a, b, d, c

a, c, b, d

a, c, d, b

a, d, b c

a, d, c b

5.

⋮

① create all possible orderings of the 4 digits

② check whether it is a valid time ?? ⇒ if valid

③ compare it with maximum time

④ output the largest time

$h_1 h_2$ ⇓ <=23

$m_1 m_2$ ⇓ <=60

14 : 58 < 20 : 11

compare hour, then compare minute

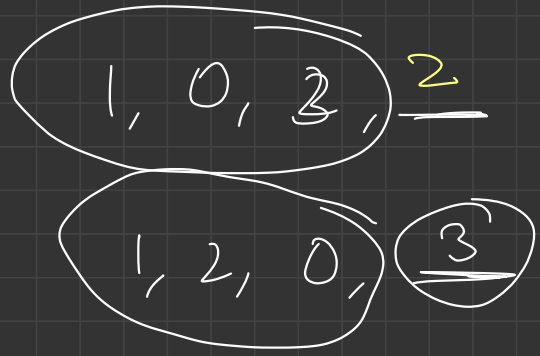
① how to create all possible times ??

[0, 1, 2, 3]

⇓

all possible orderings

choose first 3 we can get the 4th index automatically



```
for(i=0; i<=3; i++){
    for(j=0; j<=3; j++){
        if(i==j) continue;
        for(k=0; k<=3; k++){
            if(j==k or i==k)
                continue;
        }
    }
}
```

i, j, k

0-3

6-i-j-k

i=0, j=2, k=1

0, 1, 2, 6-0-1-2 ③

0, 1, 3, 6-0-1-3 ②

0, 2, 1, 6-0-2-1 ③

0, 1, ✗

✗

2

0, 1, 3, 2

i = 3

j = 1

k = 0

3, 1, 0, 2

0 → 3 → all possible nos. with no duplicate digits

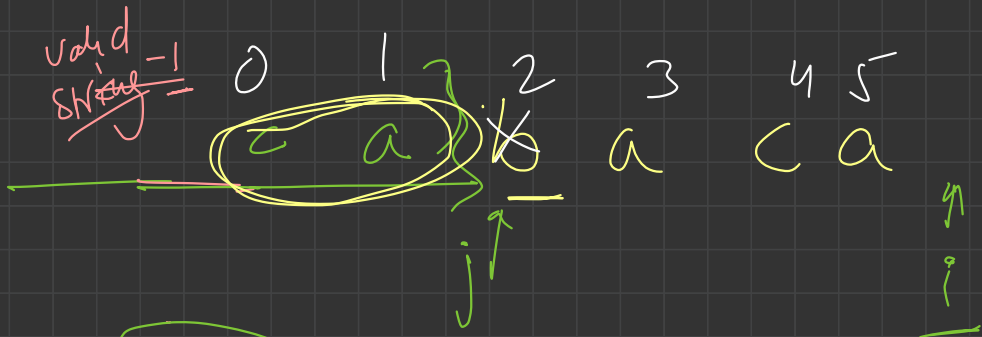
S → a b b a c a

push_back()

pop_back()

•back()

ans → c a



$s[j] = s[i]$

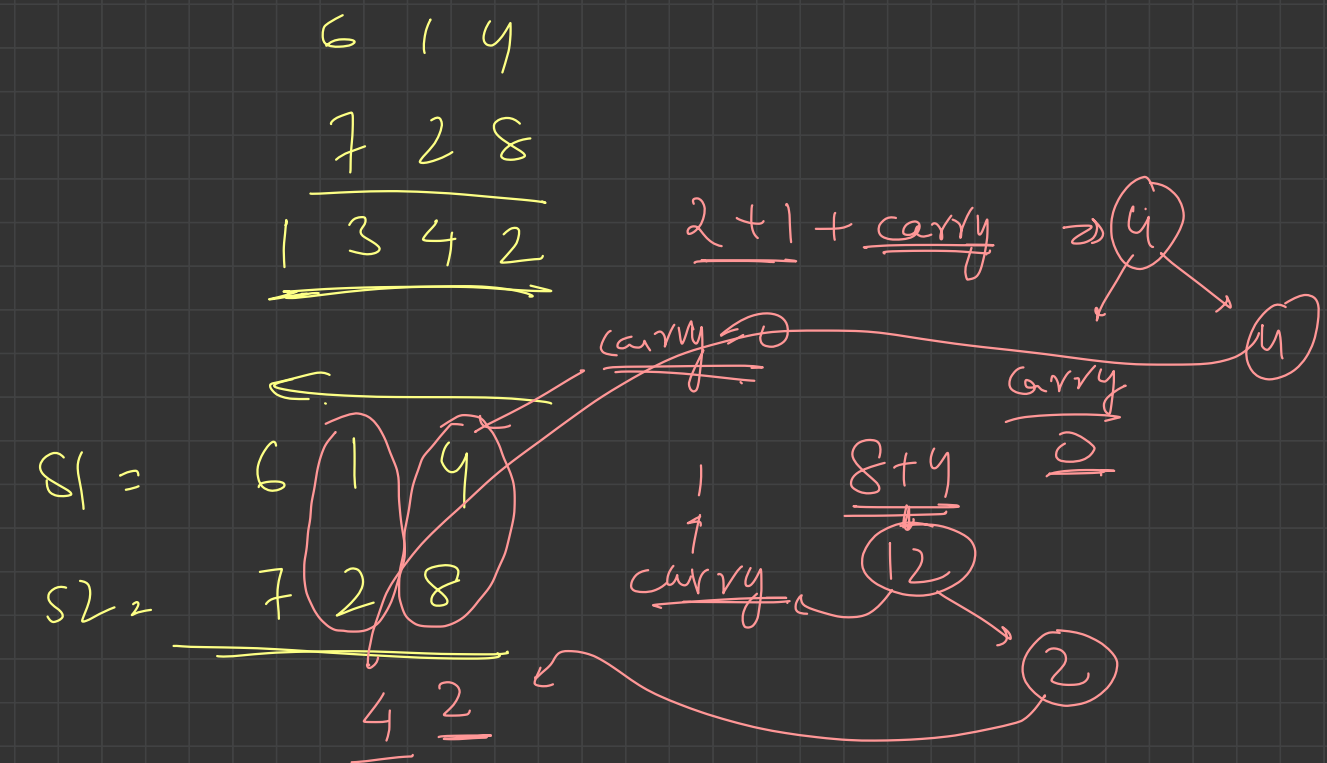
```
if( j>0 and s[j] == s[j-1] ){
    j -= 2;
```

j → string length

3

```
j++;  i++;
```

j → represents the valid part of the string


6 1 4
7 2 8
1 3 4 2
2+1+carry
4
carry=0
4
carry
0
S1 =
S2 =
6 1 4
7 2 8
8+4
1
carry
12
2
4 2