

```
def find_Max_Num(arr):
```

```
'''
```

Write a python function to find the largest number that can be formed with the given list of digits.

```
'''
```

Using 3 models from WizardCoder-Python-V1.0 Family, with  $\theta = 0.5$

$k_2 = 1, l_2 = 2, \text{temperature} = 0$

Test Case 1:

```
assert find_Max_Num([1, 2, 3, 4, 5]) == 54321
assert find_Max_Num([1, 2, 3, 4]) == 4321
```

Code Solution 1:

```
if not arr:
    return ""

arr.sort(reverse=True)
result = []

while arr:
    num = arr.pop()
    result.append(str(num))
    if len(result) > 1:
        if int("".join(result)) >
int("".join(result[:-1])):
            result.pop()
        else:
            result.pop(0)

return int("".join(result))
```

The best pair is  $(s_1, t_2)$ , where  $n_s=2, n_t=1$ . We have  $N_{t_1} = 2$  valid test lines. The minimum product to accept the output is  $\theta \times N_{t_1} \times k_1 = 1.0$ . The product of  $n_s \times n_t$  is greater, thus we accept  $s_1$ .

$t_1$   
 $t_2$  →  $s_1$

Return  $s_1$  as final output for this question and exit here

34B

$k_3 = 5, l_3 = 4, \text{temperature} = 0.8$   
(skipped)

7B

$k_1 = 3, l_1 = 2, \text{temperature} = 0.8$

Code Solution 1:

```
if not arr:
    return ""

arr.sort(reverse=True)
result = []

while arr:
    num = arr.pop()
    result.append(str(num))
    if len(result) > 1:
        if int("".join(result)) >
int("".join(result[:-1])):
            result.pop()
        else:
            result.pop(0)

return int("".join(result))
```

Code Solution 2:

```
arr.sort(reverse=True)
result = ""
for num in arr:
    result += num
    if int(result) > int(result[:-1]):
        return int(result)
return 0
```

$t_1$   
 $t_2$   
 $t_3$   
 $t_4$   
 $t_5$   
 $t_6$  →  $s_1$   
 $s_2$   
 $s_3$

Test Case 1:

```
assert find_Max_Num([9, 9, 9, 9, 9]) == 99999
assert find_Max_Num([1, 2, 3, 4, 5]) == 54321
```

Test Case 2:

```
assert find_Max_Num([9, 9, 9]) == 999 # Test 1
assert find_Max_Num([9, 9, 8]) == 989 # Test 2
```

Test Case 3:

```
assert find_Max_Num([1,2,3,4,5]) == 55555
assert find_Max_Num([9,9,9]) == 999
```

Code Solution 3:

```
n = len(arr)
max_num = ''

def backtrack(start):
    nonlocal max_num
    if start == 0 and int(max_num) >
int(max_num[-1]):
        return

    for i in range(start, n):
        if arr[start] == '0' and start > 0:
            continue
        if max_num == '':
            max_num = arr[start]
        else:
            max_num = max_num + arr[start]
            if int(max_num) > int(max_num[-1]):
                backtrack(i+1)
            max_num = max_num[:-1]

backtrack(0)
return max_num
```

The best pair is  $(s_3, t_6)$ , where  $n_s=0, n_t=0$ . We have  $N_{t_1} = 6$  valid test lines. The minimum product to accept the output is  $\theta \times N_{t_1} \times k_1 = 9.0$ . The product of  $n_s \times n_t$  is smaller, thus we reject this model's output and use the next model.