

Day-18 Python DSA

Leetcode 493 Reverse Pairs

<https://leetcode.com/problems/reverse-pairs/description/>

Bruteforce

```
def reversePairs(nums):
```

```
    count=0
```

```
    n= len(nums)
```

```
    for i in range(0,n):
```

```
        for j in range(i+1, n):
```

```
            if nums[i] > 2* nums[j]:
```

```
                count+=1
```

```
    return count
```

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

```
reversePairs(nums)
```

TC – $O(N^2)$

SC- $O(1)$

Optimal

```
class Solution:
```

```
    def mergeList(self, arr1: List[int], arr2: List[int]) -> Tuple[List[int], int]:
```

```
        n= len(arr1)
```

```
        m= len(arr2)
```

```
        count=0
```

```

result=[]
j=0
for i in range(0,n):
    while j < m and arr1[i] > 2 * arr2[j]:
        j+=1
    count += j
i,j=0,0
while i < n and j < m:
    if arr1[i] <= arr2[j]:
        result.append(arr1[i])
        i+=1
    else:
        result.append(arr2[j])
        j+=1
while i<n:
    result.append(arr1[i])
    i+=1
while j<m:
    result.append(arr2[j])
    j+=1
return result,count
def mergeSort(self, lst: List[int]) -> Tuple[List[int], int]:
    if len(lst) <= 1:
        return lst, 0

    mid = len(lst) // 2
    first_half = lst[:mid]
    second_half = lst[mid:]

    fh, cnt1 = self.mergeSort(first_half)
    sh, cnt2 = self.mergeSort(second_half)

    merged, count = self.mergeList(fh, sh)
    return merged, cnt1 + cnt2 + count

def reversePairs(self, nums: List[int]) -> int:
    m, c = self.mergeSort(nums)
    return c

```

Dry Run

let's dry run step-by-step so you see how the reverse pair counting works for

ini

Copy

Edit

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

Step 1 – Understanding the goal

We are counting reverse pairs where:

$i < j$ and $\text{nums}[i] > 2 \cdot \text{nums}[j]$

The algorithm uses merge sort so that counting can be done efficiently during merging.

Step 2 – Merge sort splitting

Initial call:

mergeSort([1, 3, 2, 3, 1])

Split: [1, 3] and [2, 3, 1]

Left half: mergeSort([1, 3])

Split: [1] and [3]

Base case: return [1], 0 and [3], 0

MergeList([1], [3]):

Count step:

$i=0 \rightarrow \text{arr1}[0]=1, \text{arr2}[0]=3 \rightarrow 1 > 2 \cdot 3$? No $\rightarrow \text{count}=0$

Merge step: [1, 3], count=0

Result: [1, 3], count=0

Right half: mergeSort([2, 3, 1])

Split: [2] and [3, 1]

mergeSort([3, 1])

Split: [3] and [1]

Base case: [3], 0 and [1], 0

MergeList([3], [1]):

Count step:

$i=0 \rightarrow arr1[0]=3, arr2[0]=1 \rightarrow 3 > 2*1?$ Yes $\rightarrow j=1 \rightarrow count=1$

Merge step: [1, 3], count=1

Result: [1, 3], count=1

mergeSort([2]) is base case $\rightarrow [2], 0$

MergeList([2], [1, 3]):

Count step:

$i=0 \rightarrow arr1[0]=2, arr2[0]=1 \rightarrow 2 > 2*1?$ No \rightarrow count stays 0

Merge step:

Compare 2 and 1 → append 1, $j=1$

Compare 2 and 3 → append 2, $i=1$

Append 3

Result: [1, 2, 3], count=0

Total for right half:

Left half count = 0

Right half count = 1

Merge count = 0

→ Total = 0 + 1 + 0 = 1

Step 3 – Final merge

Now we merge [1, 3] (left) and [1, 2, 3] (right):

MergeList([1, 3], [1, 2, 3]):

Count step:

$i=0 \rightarrow arr1[0]=1, arr2[0]=1 \rightarrow 1 > 2*1?$ No $\rightarrow j$ stays 0 $\rightarrow count=0$

$i=1 \rightarrow arr1[1]=3, arr2[0]=1 \rightarrow 3 > 2*1?$ Yes $\rightarrow j=1$

Now $arr2[1]=2 \rightarrow 3 > 2*2?$ No \rightarrow stop $\rightarrow count += j (=1) \rightarrow count=1$

So in counting phase, we found 1 reverse pair:

$(3, 1) \rightarrow$ from left=3 and right=1

Merge step:

Compare 1 and 1 \rightarrow append 1 (left), $i=1$

Compare 3 and 1 \rightarrow append 1 (right), $j=1$

Compare 3 and 2 \rightarrow append 2 (right), $j=2$

Compare 3 and 3 \rightarrow append 3 (left), $i=2$

Append remaining 3 from right

Merged result: [1, 1, 2, 3, 3]

Step 4 – Total reverse pairs

Left half count = 0

Right half count = 1

Final merge count = 1

Total = $0 + 1 + 1 = 2$

The reverse pairs found are:

(3, 1) — left half from [3, 1] merge

(3, 1) — final merge between left 3 and right 1

Answer: `reversePairs([1, 3, 2, 3, 1]) = 2`