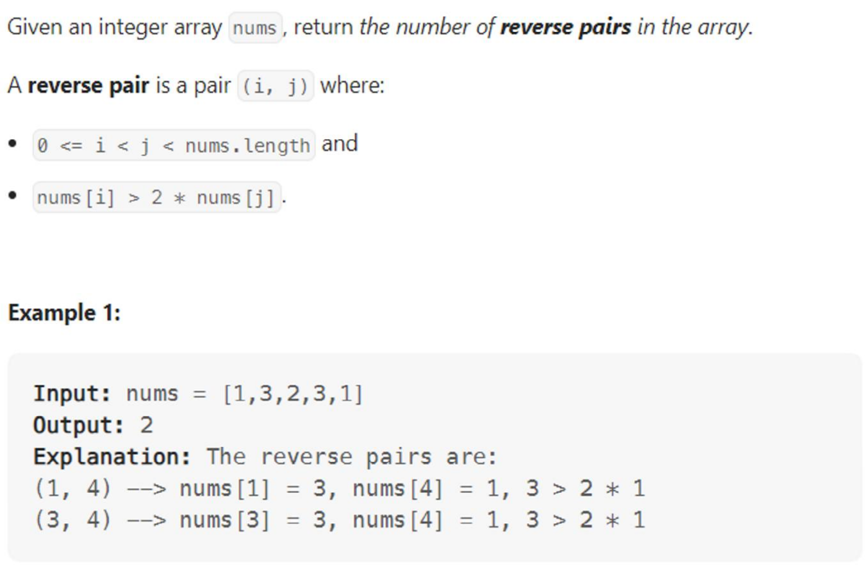
41. Reverse Pairs | Hard Interview Question



Given an integer array `nums`, return *the number of* ***reverse pairs*** *in the array*.

A **reverse pair** is a pair `(i, j)` where:

- `0 <= i < j < nums.length` and
- `nums[i] > 2 * nums[j]`.

Example 1:

```
Input: nums = [1,3,2,3,1]
Output: 2
Explanation: The reverse pairs are:
(1, 4) --> nums[1] = 3, nums[4] = 1, 3 > 2 * 1
(3, 4) --> nums[3] = 3, nums[4] = 1, 3 > 2 * 1
```

arr [] = [40 25 19 12 9 6 2]

find the no of pairs

i < j && a[i] > 2 * arr[j]

left element should be > 2 * right element

(6, 2) 6 > 2 * 2

(9, 2)

(12, 2) , (19, 2) (25, 2) (40, 2)

no of pairs = 15

1) Brute Force :

```
            i
arr [] = [40  25  19  12  9  6  2]
```

```
cnt=0
for ( i = 0 ⟶ n-1)
{
    for ( j= i+1 ⟶ n-1)
    {
        if ( a[i] > 2 * a[j])
            cnt ++;
    }
}
```

T.C ⇒ $O(n^2)$ S.C ⇒ $O(1)$

2) Optimal Solution :

$\boxed{arr[i] > 2 * arr[j]}$

[6 13 21 25]

↑ ↑ ↑ ↑

[1 2 3 4 4 5 9 11 13]

6 → 1, 2

13 → 1, 2, 3, 4, 4, 5

21 → 1, 2, 3, 4, 4, 5, 9

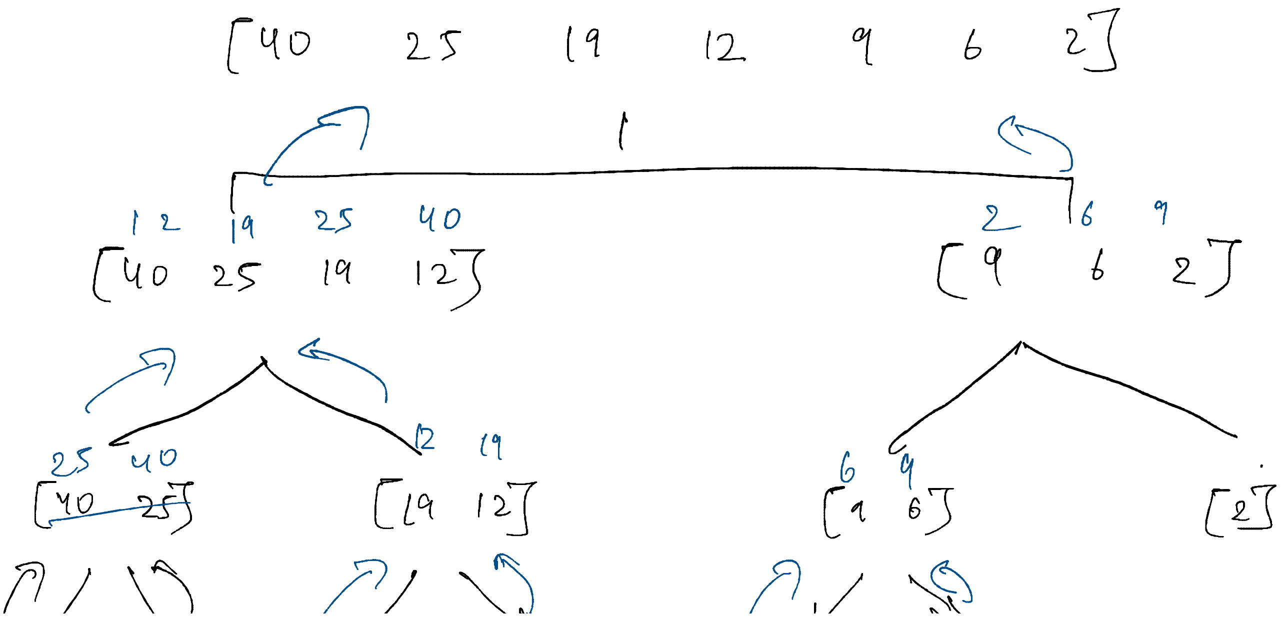
25 → 1, 2, 3, 4, 4, 5, 9, 11

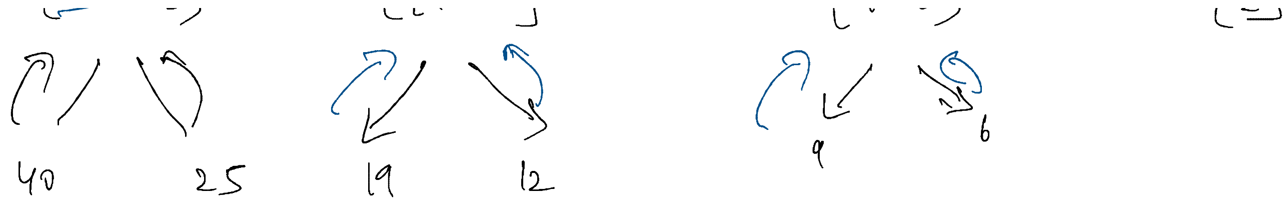
✗ You can see a pattern here, if 1, 2 can form a pair with 6 then it can form pair with 13, 21, 25 the reason is 13 is greater than 6.

As it stored, we will iterative.

$$+2 + 6 + 7 + 8 = \boxed{23}$$

this we will implement in mergesort.





{40} {25} {19} {12}

$40 > 25*2$ ✗ $19 > 12*2$ ✗

[25 40] [12 19]

↑ ↑

$25 > 12 * 2$ ✓ (+1) so everything before 12 is possible

$25 > 19 * 2$ ✗

$40 > 19 * 2$ ✓

[6 9] [2]

↑

$6 > 2 * 2$ ✓

$9 >$ ✓

[12 19 25 40] [2 6 9]

~~↑~~ ~~↑~~ ~~↑~~ ~~↑~~ ~~↑~~ ~~↑~~ ~~↑~~

$12 > 2 * 2$ ✓ +1

12 > 2 * 2 ✓ +1

12 > 2 * 6 ✗

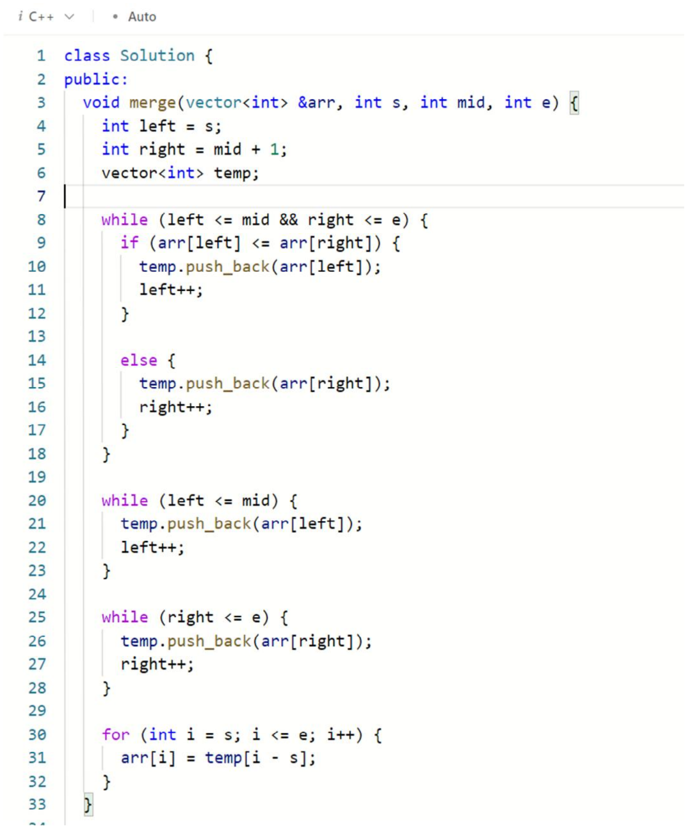
19 > 2 * 6 ✓

19 > 2 * 9 ✓

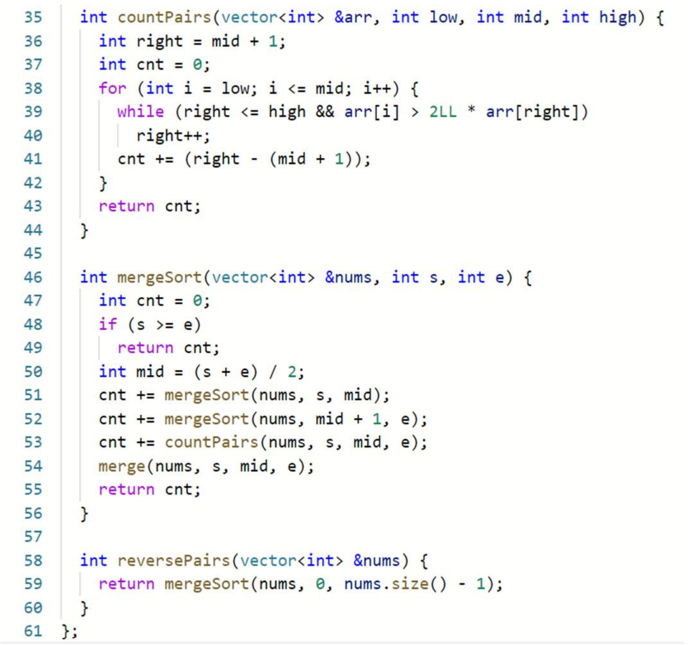
Pseudo Code

```
[6   13   21   25]        [1   2   3   4   4   5   9   11   13]
 ↑              ↑           ↑                                 ↑
low            mid        mid+1                              low
```

```
cnt = 0, right = mid+1
for (i = low ──> mid)
{
    while (right <= high && a[i] > 2 * a[right])    ──> if the cond. is crct
    {
        right ++;
    }
    cnt = cnt + (right - (mid+1))  // will give nos of
                                       element
}
```



```
class Solution {
public:
  void merge(vector<int> &arr, int s, int mid, int e) {
    int left = s;
    int right = mid + 1;
    vector<int> temp;

    while (left <= mid && right <= e) {
      if (arr[left] <= arr[right]) {
        temp.push_back(arr[left]);
        left++;
      }

      else {
        temp.push_back(arr[right]);
        right++;
      }
    }

    while (left <= mid) {
      temp.push_back(arr[left]);
      left++;
    }

    while (right <= e) {
      temp.push_back(arr[right]);
      right++;
    }

    for (int i = s; i <= e; i++) {
      arr[i] = temp[i - s];
    }
  }
```



```
  int countPairs(vector<int> &arr, int low, int mid, int high) {
    int right = mid + 1;
    int cnt = 0;
    for (int i = low; i <= mid; i++) {
      while (right <= high && arr[i] > 2LL * arr[right])
        right++;
      cnt += (right - (mid + 1));
    }
    return cnt;
  }

  int mergeSort(vector<int> &nums, int s, int e) {
    int cnt = 0;
    if (s >= e)
      return cnt;
    int mid = (s + e) / 2;
    cnt += mergeSort(nums, s, mid);
    cnt += mergeSort(nums, mid + 1, e);
    cnt += countPairs(nums, s, mid, e);
    merge(nums, s, mid, e);
    return cnt;
  }

  int reversePairs(vector<int> &nums) {
    return mergeSort(nums, 0, nums.size() - 1);
  }
};
```

T.C ⇒ $O(\log n \times (n+n))$

— $O(2n\log n)$

S.C ⇒ $O(n)$