**905. Sort Array By Parity**

Given an array A of non-negative integers, return an array consisting of all the even elements of A, followed by all the odd elements of A.

You may return any answer array that satisfies this condition.

我的解决方案：

|  |
| --- |
| class Solution {  public:  vector<int> sortArrayByParity(vector<int>& A) {  vector<int> B(A.size(),0);  int beg = 0, end = A.size() - 1;  for(vector<int>::iterator i = A.begin(); i < A.end(); i++) {  if(\*i % 2 == 1) {  B[end] = \*i;  end--;  }  else {  B[beg] = \*i;  beg++;  }  }  return B;  }  }; |

C++最快的解决方案；

|  |
| --- |
| class Solution {  public:  vector<int> sortArrayByParity(vector<int>& A) {  int left = 0;  int right = A.size()-1;  while (left <= right) {  // If odd, we want to swap  if (A[left]%2 != 0) {  while(A[right]%2 != 0) {  --right;  if (right < left) return A;  }    int temp = A[right];  A[right] = A[left];  A[left] = temp;  }  ++left;  }  return A;  }  }; |

官方的解决思路：

Approach 1: Sort

Use a custom comparator when sorting, to sort by parity.

|  |
| --- |
| class Solution(object):  def sortArrayByParity(self, A):  A.sort(key = lambda x: x % 2)  return A |

#### Approach 2: Two Pass

Write all the even elements first, then write all the odd elements.

|  |
| --- |
| class Solution(object):  def sortArrayByParity(self, A):  return ([x for x in A if x % 2 == 0] +  [x for x in A if x % 2 == 1]) |

#### Approach 3: In-Place

**Intuition**

If we want to do the sort in-place, we can use quicksort, a standard textbook algorithm.

**Algorithm**

We'll maintain two pointers i and j. The loop invariant is everything below ihas parity 0 (ie. A[k] % 2 == 0 when k < i), and everything above j has parity 1.

Then, there are 4 cases for (A[i] % 2, A[j] % 2):

* If it is (0, 1), then everything is correct: i++ and j--.
* If it is (1, 0), we swap them so they are correct, then continue.
* If it is (0, 0), only the i place is correct, so we i++ and continue.
* If it is (1, 1), only the j place is correct, so we j-- and continue.

Throughout all 4 cases, the loop invariant is maintained, and j-i is getting smaller. So eventually we will be done with the array sorted as desired.

|  |
| --- |
| class Solution(object):  def sortArrayByParity(self, A):  i, j = 0, len(A) - 1  while i < j:  if A[i] % 2 > A[j] % 2:  A[i], A[j] = A[j], A[i]  if A[i] % 2 == 0: i += 1  if A[j] % 2 == 1: j -= 1  return A |