How to reverse a Array?

This is rookie level question. We are expected to reverse a array which holds elements.

There are two solutions for it. We will see both of the solutions are compare them as against time and space complexity.

Start iterating array from end and copy values to another array. The resulting array with have elements in reverse order as compared to first array. Mission accomplished. But for this problem we are not supposed to take another array. Let’s say our original array is of length 1,000,000 i.e. 1 million. Then for this solution we are taking another auxiliary array of 1million length and copying values into it. This is not efficient.

Time Complexity is O(n) as we are iterating entire array.

Space Complexity is O(n) as we are taking new array to store the result.

Reverse using extra space code:

/\*\*

\* Reverse array by auxiliary array.

\* \*/

**public** **static** **int**[] reverse(**int**[] a) {

**int**[] result = **new** **int**[a.length];

**int** k = 0;

**for** (**int** i = a.length - 1; i >= 0; i--) {

result[k++] = a[i];

}

**return** result;

}

/\*\*

\* Reverse array by auxiliary array.

\* \*/

**public** **static** **short**[] reverse(**short**[] a) {

**short**[] result = **new** **short**[a.length];

**int** k = 0;

**for** (**int** i = a.length - 1; i >= 0; i--) {

result[k++] = a[i];

}

**return** result;

}

/\*\*

\* Reverse array by auxiliary array.

\* \*/

**public** **static** **byte**[] reverse(**byte**[] a) {

**byte**[] result = **new** **byte**[a.length];

**int** k = 0;

**for** (**int** i = a.length - 1; i >= 0; i--) {

result[k++] = a[i];

}

**return** result;

}

/\*\*

\* Reverse array by auxiliary array.

\* \*/

**public** **static** **char**[] reverse(**char**[] a) {

**char**[] result = **new** **char**[a.length];

**int** k = 0;

**for** (**int** i = a.length - 1; i >= 0; i--) {

result[k++] = a[i];

}

**return** result;

}

/\*\*

\* Reverse array by auxiliary array.

\* \*/

**public** **static** **long**[] reverse(**long**[] a) {

**long**[] result = **new** **long**[a.length];

**int** k = 0;

**for** (**int** i = a.length - 1; i >= 0; i--) {

result[k++] = a[i];

}

**return** result;

}

/\*\*

\* Reverse array by auxiliary array.

\* \*/

**public** **static** **double**[] reverse(**double**[] a) {

**double**[] result = **new** **double**[a.length];

**int** k = 0;

**for** (**int** i = a.length - 1; i >= 0; i--) {

result[k++] = a[i];

}

**return** result;

}

/\*\*

\* Reverse array by auxiliary array.

\* \*/

**public** **static** **float**[] reverse(**float**[] a) {

**float**[] result = **new** **float**[a.length];

**int** k = 0;

**for** (**int** i = a.length - 1; i >= 0; i--) {

result[k++] = a[i];

}

**return** result;

}

/\*\*

\* Reverse array by auxiliary array.

\* \*/

**public** **static** String[] reverse(String[] a) {

String[] result = **new** String[a.length];

**int** k = 0;

**for** (**int** i = a.length - 1; i >= 0; i--) {

result[k++] = a[i];

}

**return** result;

}

We can reverse the array in-place. How to do that? Well it is simple thing to do. Swap first and last element. In this method when we will be half way to array we would have reversed entire array. Pretty neat, fast and efficient solution.

Time Complexity is O(n) as we are iterating entire array.

Space Complexity is O(1) as we are not taking new array to store the result.

I have written code for byte[], char[], int[], long[], doube[], float[], short[] and String[] for Solution 2.

/\*\*

\* Reverse Array in-place.

\* Swap first and last element in array.

\* Works for ever and odd length arrays.

\* \*/

**public** **static** **int**[] reverseInPlace(**int**[] a) {

**int** start = 0;

**int** end = a.length - 1;

**while** (start < end) {

**int** temp = a[start];

a[start] = a[end];

a[end] = temp;

start++;

end--;

}

**return** a;

}

/\*\*

\* Reverse Array in-place.

\* Swap first and last element in array.

\* Works for ever and odd length arrays.

\* \*/

**public** **static** **short**[] reverseInPlace(**short**[] a) {

**int** start = 0;

**int** end = a.length - 1;

**while** (start < end) {

**short** temp = a[start];

a[start] = a[end];

a[end] = temp;

start++;

end--;

}

**return** a;

}

/\*\*

\* Reverse Array in-place.

\* Swap first and last element in array.

\* Works for ever and odd length arrays.

\* \*/

**public** **static** **long**[] reverseInPlace(**long**[] a) {

**int** start = 0;

**int** end = a.length - 1;

**while** (start < end) {

**long** temp = a[start];

a[start] = a[end];

a[end] = temp;

start++;

end--;

}

**return** a;

}

/\*\*

\* Reverse Array in-place.

\* Swap first and last element in array.

\* Works for ever and odd length arrays.

\* \*/

**public** **static** **float**[] reverseInPlace(**float**[] a) {

**int** start = 0;

**int** end = a.length - 1;

**while** (start < end) {

**float** temp = a[start];

a[start] = a[end];

a[end] = temp;

start++;

end--;

}

**return** a;

}

/\*\*

\* Reverse Array in-place.

\* Swap first and last element in array.

\* Works for ever and odd length arrays.

\* \*/

**public** **static** **byte**[] reverseInPlace(**byte**[] a) {

**int** start = 0;

**int** end = a.length - 1;

**while** (start < end) {

**byte** temp = a[start];

a[start] = a[end];

a[end] = temp;

start++;

end--;

}

**return** a;

}

/\*\*

\* Reverse Array in-place.

\* Swap first and last element in array.

\* Works for ever and odd length arrays.

\* \*/

**public** **static** **char**[] reverseInPlace(**char**[] a) {

**int** start = 0;

**int** end = a.length - 1;

**while** (start < end) {

**char** temp = a[start];

a[start] = a[end];

a[end] = temp;

start++;

end--;

}

**return** a;

}

/\*\*

\* Reverse Array in-place.

\* Swap first and last element in array.

\* Works for ever and odd length arrays.

\* \*/

**public** **static** **double**[] reverseInPlace(**double**[] a) {

**int** start = 0;

**int** end = a.length - 1;

**while** (start < end) {

**double** temp = a[start];

a[start] = a[end];

a[end] = temp;

start++;

end--;

}

**return** a;

}

/\*\*

\* Reverse Array in-place.

\* Swap first and last element in array.

\* Works for ever and odd length arrays.

\* \*/

**public** **static** String[] reverseInPlace(String[] a) {

**int** start = 0;

**int** end = a.length - 1;

**while** (start < end) {

String temp = a[start];

a[start] = a[end];

a[end] = temp;

start++;

end--;

}

**return** a;

}

Below is final program, test cases and output of test cases:

**Program:**

**package** arrays;

**public** **class** ArrayReverse {

**public** **static** **void** main(String[] args) {

**float**[] b = *reverseInPlace*(**new** **float**[] { 1, 2, 3, 4, 5 });

**for** (**int** i = 0; i < b.length; i++) {

System.***out***.println(b[i]);

}

}

/\*\*

\* Reverse array by auxiliary array.

\* \*/

**public** **static** **int**[] reverse(**int**[] a) {

**int**[] result = **new** **int**[a.length];

**int** k = 0;

**for** (**int** i = a.length - 1; i >= 0; i--) {

result[k++] = a[i];

}

**return** result;

}

/\*\*

\* Reverse array by auxiliary array.

\* \*/

**public** **static** **short**[] reverse(**short**[] a) {

**short**[] result = **new** **short**[a.length];

**int** k = 0;

**for** (**int** i = a.length - 1; i >= 0; i--) {

result[k++] = a[i];

}

**return** result;

}

/\*\*

\* Reverse array by auxiliary array.

\* \*/

**public** **static** **byte**[] reverse(**byte**[] a) {

**byte**[] result = **new** **byte**[a.length];

**int** k = 0;

**for** (**int** i = a.length - 1; i >= 0; i--) {

result[k++] = a[i];

}

**return** result;

}

/\*\*

\* Reverse array by auxiliary array.

\* \*/

**public** **static** **char**[] reverse(**char**[] a) {

**char**[] result = **new** **char**[a.length];

**int** k = 0;

**for** (**int** i = a.length - 1; i >= 0; i--) {

result[k++] = a[i];

}

**return** result;

}

/\*\*

\* Reverse array by auxiliary array.

\* \*/

**public** **static** **long**[] reverse(**long**[] a) {

**long**[] result = **new** **long**[a.length];

**int** k = 0;

**for** (**int** i = a.length - 1; i >= 0; i--) {

result[k++] = a[i];

}

**return** result;

}

/\*\*

\* Reverse array by auxiliary array.

\* \*/

**public** **static** **double**[] reverse(**double**[] a) {

**double**[] result = **new** **double**[a.length];

**int** k = 0;

**for** (**int** i = a.length - 1; i >= 0; i--) {

result[k++] = a[i];

}

**return** result;

}

/\*\*

\* Reverse array by auxiliary array.

\* \*/

**public** **static** **float**[] reverse(**float**[] a) {

**float**[] result = **new** **float**[a.length];

**int** k = 0;

**for** (**int** i = a.length - 1; i >= 0; i--) {

result[k++] = a[i];

}

**return** result;

}

/\*\*

\* Reverse array by auxiliary array.

\* \*/

**public** **static** String[] reverse(String[] a) {

String[] result = **new** String[a.length];

**int** k = 0;

**for** (**int** i = a.length - 1; i >= 0; i--) {

result[k++] = a[i];

}

**return** result;

}

/\*\*

\* Reverse Array in-place.

\* Swap first and last element in array.

\* Works for ever and odd length arrays.

\* \*/

**public** **static** **int**[] reverseInPlace(**int**[] a) {

**int** start = 0;

**int** end = a.length - 1;

**while** (start < end) {

**int** temp = a[start];

a[start] = a[end];

a[end] = temp;

start++;

end--;

}

**return** a;

}

/\*\*

\* Reverse Array in-place.

\* Swap first and last element in array.

\* Works for ever and odd length arrays.

\* \*/

**public** **static** **short**[] reverseInPlace(**short**[] a) {

**int** start = 0;

**int** end = a.length - 1;

**while** (start < end) {

**short** temp = a[start];

a[start] = a[end];

a[end] = temp;

start++;

end--;

}

**return** a;

}

/\*\*

\* Reverse Array in-place.

\* Swap first and last element in array.

\* Works for ever and odd length arrays.

\* \*/

**public** **static** **long**[] reverseInPlace(**long**[] a) {

**int** start = 0;

**int** end = a.length - 1;

**while** (start < end) {

**long** temp = a[start];

a[start] = a[end];

a[end] = temp;

start++;

end--;

}

**return** a;

}

/\*\*

\* Reverse Array in-place.

\* Swap first and last element in array.

\* Works for ever and odd length arrays.

\* \*/

**public** **static** **float**[] reverseInPlace(**float**[] a) {

**int** start = 0;

**int** end = a.length - 1;

**while** (start < end) {

**float** temp = a[start];

a[start] = a[end];

a[end] = temp;

start++;

end--;

}

**return** a;

}

/\*\*

\* Reverse Array in-place.

\* Swap first and last element in array.

\* Works for ever and odd length arrays.

\* \*/

**public** **static** **byte**[] reverseInPlace(**byte**[] a) {

**int** start = 0;

**int** end = a.length - 1;

**while** (start < end) {

**byte** temp = a[start];

a[start] = a[end];

a[end] = temp;

start++;

end--;

}

**return** a;

}

/\*\*

\* Reverse Array in-place.

\* Swap first and last element in array.

\* Works for ever and odd length arrays.

\* \*/

**public** **static** **char**[] reverseInPlace(**char**[] a) {

**int** start = 0;

**int** end = a.length - 1;

**while** (start < end) {

**char** temp = a[start];

a[start] = a[end];

a[end] = temp;

start++;

end--;

}

**return** a;

}

/\*\*

\* Reverse Array in-place.

\* Swap first and last element in array.

\* Works for ever and odd length arrays.

\* \*/

**public** **static** **double**[] reverseInPlace(**double**[] a) {

**int** start = 0;

**int** end = a.length - 1;

**while** (start < end) {

**double** temp = a[start];

a[start] = a[end];

a[end] = temp;

start++;

end--;

}

**return** a;

}

/\*\*

\* Reverse Array in-place.

\* Swap first and last element in array.

\* Works for ever and odd length arrays.

\* \*/

**public** **static** String[] reverseInPlace(String[] a) {

**int** start = 0;

**int** end = a.length - 1;

**while** (start < end) {

String temp = a[start];

a[start] = a[end];

a[end] = temp;

start++;

end--;

}

**return** a;

}

}

Test Cases:

**package** arrays;

**import** org.junit.Assert;

**import** org.junit.Test;

**public** **class** ArrayReverseTest {

**byte**[] b = { 1, 2, 3, 4, 5 };

**char**[] c = { 'a', 'b', 'c', 'd', 'e' };

**int**[] i = { 1, 2, 3, 4, 5 };

**long**[] l = { 1, 2, 3, 4, 5 };

**double**[] d = { 1, 2, 3, 4, 5 };

**float**[] f = { 1, 2, 3, 4, 5 };

**short**[] s = { 1, 2, 3, 4, 5 };

String[] str={"Jan","Feb","March","April","May"};

**byte**[] bReverse = { 5, 4, 3, 2, 1 };

**char**[] cReverse = { 'e', 'd', 'c', 'b', 'a' };

**int**[] iReverse = { 5, 4, 3, 2, 1 };

**long**[] lReverse = { 5, 4, 3, 2, 1 };

**double**[] dReverse = { 5, 4, 3, 2, 1 };

**float**[] fReverse = { 5, 4, 3, 2, 1 };

**short**[] sReverse = { 5, 4, 3, 2, 1 };

String[] strReverse={"May","April","March","Feb","Jan"};

@Test

**public** **void** testByteArray() {

Assert.*assertArrayEquals*(bReverse, ArrayReverse.*reverse*(b));

}

@Test

**public** **void** testCharArray() {

Assert.*assertArrayEquals*(cReverse, ArrayReverse.*reverse*(c));

}

@Test

**public** **void** testIntArray() {

Assert.*assertArrayEquals*(iReverse, ArrayReverse.*reverse*(i));

}

@Test

**public** **void** testLongArray() {

Assert.*assertArrayEquals*(lReverse, ArrayReverse.*reverse*(l));

}

@Test

**public** **void** testDoubleArray() {

Assert.*assertArrayEquals*(dReverse, ArrayReverse.*reverse*(d), 0.00);

}

@Test

**public** **void** testFloatArray() {

Assert.*assertArrayEquals*(fReverse, ArrayReverse.*reverse*(f), 0.0f);

}

@Test

**public** **void** testShortArray() {

Assert.*assertArrayEquals*(sReverse, ArrayReverse.*reverse*(s));

}

@Test

**public** **void** testStringArray() {

Assert.*assertArrayEquals*(strReverse, ArrayReverse.*reverse*(str));

}

//-------------------Test Cases for In-Place Solution---------------------

@Test

**public** **void** testByteArrayInPlace() {

Assert.*assertArrayEquals*(bReverse, ArrayReverse.*reverseInPlace*(b));

}

@Test

**public** **void** testCharArrayInPlace() {

Assert.*assertArrayEquals*(cReverse, ArrayReverse.*reverseInPlace*(c));

}

@Test

**public** **void** testIntArrayInPlace() {

Assert.*assertArrayEquals*(iReverse, ArrayReverse.*reverseInPlace*(i));

}

@Test

**public** **void** testLongArrayInPlace() {

Assert.*assertArrayEquals*(lReverse, ArrayReverse.*reverseInPlace*(l));

}

@Test

**public** **void** testDoubleArrayInPlace() {

Assert.*assertArrayEquals*(dReverse, ArrayReverse.*reverseInPlace*(d), 0.00);

}

@Test

**public** **void** testFloatArrayInPlace() {

Assert.*assertArrayEquals*(fReverse, ArrayReverse.*reverseInPlace*(f), 0.0f);

}

@Test

**public** **void** testShortArrayInPlace() {

Assert.*assertArrayEquals*(sReverse, ArrayReverse.*reverseInPlace*(s));

}

@Test

**public** **void** testStringArrayInPlace() {

Assert.*assertArrayEquals*(strReverse, ArrayReverse.*reverseInPlace*(str));

}

}

Output of Test cases:

