Reversal algorithm for array rotation

Write a function rotate(arr[], d, n) that rotates arr[] of size n by d elements.

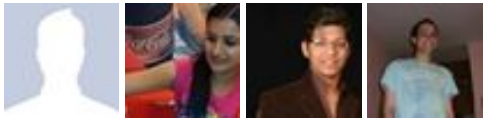
Rotation of the above array by 2 will make array

Method 4(The Reversal Algorithm)
Please read this for first three methods of array rotation.

Algorithm:

```
rotate(arr[], d, n)
  reverse(arr[], 1, d) ;
  reverse(arr[], d + 1, n);
  reverse(arr[], l, n);
```

Let AB are the two parts of the input array where A = arr[0..d-1] and B = arr[d..n-1]. The idea of the algorithm is:
Reverse A to get ArB. /* Ar is reverse of A */
Reverse B to get ArBr. /* Br is reverse of B */



Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion
Geometric Algorithms

Reverse all to get (ArBr) r = BA.

For arr[] = [1, 2, 3, 4, 5, 6, 7], d =2 and n = 7
A = [1, 2] and B = [3, 4, 5, 6, 7]
Reverse A, we get ArB = [2, 1, 3, 4, 5, 6, 7]
Reverse B, we get ArBr = [2, 1, 7, 6, 5, 4, 3]
Reverse all, we get (ArBr)r = [3, 4, 5, 6, 7, 1, 2]

Implementation:

```
/*Utility function to print an array */
void printArray(int arr[], int size);

/* Utility function to reverse arr[] from start to end */
void rvereseArray(int arr[], int start, int end);

/* Function to left rotate arr[] of size n by d */
void leftRotate(int arr[], int d, int n)
{
  rvereseArray(arr, 0, d-1);
  rvereseArray(arr, d, n-1);
  rvereseArray(arr, 0, n-1);
}

/*UTILITY FUNCTIONS*/
/* function to print an array */
void printArray(int arr[], int size)
{
  int i;
  for(i = 0; i < size; i++)
    printf("%d ", arr[i]);
  printf("%\n ");
}

/*Function to reverse arr[] from index start to end*/
void rvereseArray(int arr[], int start, int end)
{
  int i;
  int temp;
  while(start < end)
  {
    temp = arr[start];
    arr[start] = arr[end];
    arr[end] = temp;
    start++;
```


Popular Posts

```
    end--;
  }
}

/* Driver program to test above functions */
int main()
{
   int arr[] = {1, 2, 3, 4, 5, 6, 7};
   leftRotate(arr, 2, 7);
   printArray(arr, 7);
   getchar();
   return 0;
}
```

Time Complexity: O(n)

References:
http://www.cs.bell-labs.com/cm/cs/pearls/s02b.pdf

Related Tpoics:

- Remove minimum elements from either side such that 2*min becomes more than max
- Divide and Conquer | Set 6 (Search in a Row-wise and Column-wise Sorted 2D Array)

- Bucket Sort
- Kth smallest element in a row-wise and column-wise sorted 2D array | Set 1
- Find the number of zeroes
- Find if there is a subarray with 0 sum
- Divide and Conquer | Set 5 (Strassen's Matrix Multiplication)
- Count all possible groups of size 2 or 3 that have sum as multiple of 3

Writing code in comment? Please use **ideone.com** and share the link here.

16 Comments GeeksforGeeks

Sort by Newest

shashi jey • 3 months ago
its a clockwise rotation of array,,ther is not any code for anticlockwise rotation
• Reply • Share ›

Vijay Apurva • 9 months ago
if we try to rotate by 10 then this algo gives wrong answer..

you should add these lines in leftrotate function.

```
k=k%n;
if(k==0)
return;
```

1 • Reply • Share ›

rajx • 11 months ago
Above algorithm iterates Array two time. Here is solution that will do the same

Recent Comments

Aman Hi, Why arent we checking for conditions...
Write a C program to Delete a Tree. · 21 minutes ago

kzs please provide solution for the problem...
Backtracking | Set 2 (Rat in a Maze) · 24 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...
Root to leaf path sum equal to a given number · 49 minutes ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...
Count trailing zeroes in factorial of a number · 51 minutes ago

newCoder3006 If the array contains negative numbers also. We...
Find subarray with given sum · 1 hour ago

newCoder3006 Code without using while loop. We can do it...
Find subarray with given sum · 1 hour ago

is moved more than once.). See the code...

```
void rotateArray(int *arr, int size, int d)
{
int count=size, start = 0, i = 0, j;
int old = arr[0], tmp;
while(count)
{
j = (i-d < 0)?i-d+size:i-d;
tmp = arr[j];
arr[j] = old;
old = tmp;
// to avoid shifting same element more than once
if(j == start){
i += 1;
start = i;
old = arr[start];
}else
i=j;
count--;
}
}
```


Reply • Share ›

shek8034 • 11 months ago

Best solution .... :)

Reply • Share ›

Nirdesh • a year ago

GeeksforGeeks team,you are missing another algorightm named "Gries-Mills" performance than Juggling and Reversal.

The algorithm swaps the largest equal-sized non-overlapping blocks available
swap as an operation as follows :

* A is the left subarray, B is the right subarray — that is, the starting point is AE
* If A is shorter, divide B into BL and BR, such that length of BR equals the len
* Swap A and BR to change ABLBR into BRBLA
* Recur on the two pieces of B
* Once A and B are of equal lengths, swap A and B

One can find in comparasion of above 3 algorithem in :

http://www.drdobbs.com/paralle...

1 ^ | v • Reply • Share ›

Nirdesh • a year ago

This is really a simple and great solution with O(n) time.

^ | v • Reply • Share ›

Chinmaya • a year ago

Wow...Mazza aagaya....elegant solution.

Thanks GeekforGeeks...

^ | v • Reply • Share ›

moonlight • a year ago

this algorithm is o(lg n)..correct me if I am wrong..

```
void reverse(int arr[],int first,int last)
{
int temp=0;
for(int i=first;i<last/2;i++)
{
temp=arr[i];
```

```
arr[i]=arr[last-i-1];
arr[last-i-1]=temp;
}

}
void arrayRotation3(int arr[],int size,int shifts)
{
reverse(arr,0,shifts);
reverse(arr,shifts,size+shifts);
reverse(arr,0,size);
}
```


alien ➚ moonlight • a year ago

It is the same algorithm written here in the post.

```
/* Paste your code here (You may delete these lines if not wri
```


Ganesh • a year ago

You can find java code here:

```
[sourcecode language="JAVA"]
/**
* Write a function to rotate arr[] of size n by d elements.
*
* @author GAPIITD
*
*/
public class ArrayRotation {
```

```
int arr[] = {1, 2, 3, 4, 5, 6, 7}, toBeShifted = 2;
rotateArray(arr, toBeShifted);
}

private static void rotateArray(int[] arr, int toBeShifted) {
if (toBeShifted > arr.length) {
System.out.println(" NO OF ROTATIONS IS GREATER THAN THE ARRAY S
```

see more

• Reply • Share ›

kirubakaran • 3 years ago

I have some thought about another algorithm.
current = start_of_array
next = d-1
while (next < n-d)
swap array[current], array[next]
current = current + 1
next = next + 1
Correct me if something is wrong.

• Reply • Share ›

alien ↗ kirubakaran • a year ago

your algorithm does not work for following case:

arr[] = {1,2,3,4,5,6}
d=4

Tracing:

initially current = 1, next = arr[3], which means next = 4
first pass:
4,2,3,1,5,6

second pass:
4,5,3,1,2,6
third pass:
4,5,6,1,2,3

which is incorrect.
Answer should be 3,4,5,6,1,2

```
/* Paste your code here (You may delete these lines if not wri
```


Venki • 4 years ago

Assume the string S is composed of strings M and N, and also reversal of rev

S = MN
S = R(R(S))

R(S) = R(N)R(M) = N'M' (new names)
Rotate(S) = R(N')R(M')

manjusha chava • 4 years ago

V can use reversing in an another way to achieve rotation.

first v hav to reverse the whole string
then reverse the n-d and d elements seperately.
eg: I/p {1,2,3,4,5,6,7} d=2

first reversing whole array ----> {7,6,5,4,3,2,1}
then reverse n-d i.e, from 7 to 3--->{3,4,5,6,7,2,1}
finally reverse last d elements 2 to 1---->{3,4,5,6,7,1,2}
which is our desired o/p

Jagdish • 4 years ago

It can be solved in O(n)-

1) put the first element arr[0] in a temp
2) identify its destination index (n +(currentIndex - d) %7 3) one by one swap a

Gaurav Kishan • 4 years ago

Its superb.So simple to conceive the solution.
Thank you geeksforgeeks.