Block swap algorithm for array rotation

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

Rotation of the above array by 2 will make array

Algorithm:

```
Initialize A = arr[0..d-1] and B = arr[d..n-1]
1) Do following until size of A is equal to size of B

  a)  If A is shorter, divide B into Bl and Br such that Br is of same
       length as A. Swap A and Br to change ABlBr into BrBlA. Now A
       is at its final place, so recur on pieces of B.

   b)  If A is longer, divide A into Al and Ar such that Al is of same
       length as B Swap Al and B to change AlArB into BArAl. Now B
       is at its final place, so recur on pieces of A.

2)  Finally when A and B are of equal size, block swap them.
```


Recursive Implementation:

```
#include<stdio.h>

/*Prototype for utility functions */
void printArray(int arr[], int size);
void swap(int arr[], int fi, int si, int d);

void leftRotate(int arr[], int d, int n)
{
  /* Return If number of elements to be rotated is
     zero or equal to array size */
  if(d == 0 || d == n)
    return;

  /*If number of elements to be rotated is exactly
     half of array size */
  if(n-d == d)
  {
    swap(arr, 0, n-d, d);
    return;
  }

 /* If A is shorter*/
  if(d < n-d)
  {
    swap(arr, 0, n-d, d);
    leftRotate(arr, d, n-d);
  }
  else /* If B is shorter*/
  {
    swap(arr, 0, d, n-d);
    leftRotate(arr+n-d, 2*d-n, d); /*This is tricky*/
  }
}

/*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 ");
}
```


Popular Posts

```
/*This function swaps d elements starting at index fi
  with d elements starting at index si */
void swap(int arr[], int fi, int si, int d)
{
   int i, temp;
   for(i = 0; i<d; i++)
   {
     temp = arr[fi + i];
     arr[fi + i] = arr[si + i];
     arr[si + i] = temp;
   }
}

/* 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;
}
```

Iterative Implementation:

Here is iterative implementation of the same algorithm. Same utility function swap() is used here.

```
void leftRotate(int arr[], int d, int n)
{
  int i, j;
  if(d == 0 || d == n)
    return;
  i = d;
  j = n - d;
  while (i != j)
  {
    if(i < j) /*A is shorter*/
    {
      swap(arr, d-i, d+j-i, i);
      j -= i;
    }
    else /*B is shorter*/
    {
      swap(arr, d-i, d, j);
      i -= j;
    }
```

```
    // printArray(arr, 7);
  }
  /*Finally, block swap A and B*/
  swap(arr, d-i, d, i);
}
```

Time Complexity: O(n)

Please see following posts for other methods of array rotation:
http://geeksforgeeks.org/?p=2398
http://geeksforgeeks.org/?p=2838

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

Please write comments if you find any bug in the above programs/algorithms or want to share any additional information about the block swap algorithm.

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)

Recent Comments

- 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.

12 Comments GeeksforGeeks

Sort by Newest

cool • 9 months ago
wats d logic in '2d-n'??explain pls!!
• Reply • Share ›

venkat cool • 9 months ago
there "among the first (d) elements last (n-d) elements are replaced so in the remaining elements the number of elements to be rotated is (d-(n-d)) i.e., (2d-n)...
Eg: arr[] = {1, 2, 3, 4, 5, 6, 7, 8, 9} and rotate by (d= 6).

after first block swap {7, 8, 9, 4, 5, 6, 1, 2, 3}

now we have to rotate array {4, 5, 6, 1, 2, 3} by 2*6 - 9 = 3

I hope understood.....:)
3 • Reply • Share ›

Anonymous • 10 months ago

The following code takes O(d)/O(n-d) time whichever is larger. But has a spac
The following code stores the array the array to be swapped in another array.

```
 #include<stdio.h>
#include<stdlib.h>

void rotateArray(int a[],int d,int n)
{
    d%=n;
    int i,j=0;
    int *b=(int *) calloc (d,sizeof(int));
    for(i=0;i<d;i++)
        b[i]=a[i];
    for(i=d;i<n;i++)
        a[i-d]=a[i];
    for(i=n-d;i<n;i++)
        a[i]=b[i-n+d];
}
```

see more

• Reply • Share ›

Vanathi • a year ago

Algo:
Rotate array(arr[] , d, length)
1.Initialize A[] of given input array size.
2.store j = d ,i=0
3.Iterate the array til i < array.length
a. if j < array.length
add the elements array[j] into A[i];
increment i;

increment j;
b. else
assign j=0;
iterate till j!= d
add the element array[j] into A[i];
increment i;
increment j;

return A;

[sourcecode language="JAVA"]

see more

• Reply • Share ›

naresh • a year ago

Here is the block-swap implementation in java (block swap can be rep

```
package com.nbethi;

import java.util.Arrays;

public class BlockSwapArray {

    public static void main(String[] args) {
        int[] array;
        int size = 17;
        array = new int[size];
        for (int i = 0; i < size; i++) {
            array[i] = i;
        }
        System.out.println(Arrays.toString(array));
        rotateLeft(array, 7);
```

```
        System.out.println(Arrays.toString(array));
```

see more

• Reply • Share ›

ar • 2 years ago

The algorithm does not handle the above example: [1,2,3,4,5,6,7] with d = 2

Because at final step you will have [3,5,4,6,7,1,2]

And A = [5,4] and B=EMPTY

So if d==n, then we should swap the numbers. This case should be corrected

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

• Reply • Share ›

red • 3 years ago

swap the first d elements with the next d elements (swapping one element at a
auxiliary varaible).
continue this until the no of elements left in the array is less than d. in that mov
required.

time complexity O(n)
space complexity O(1)

• Reply • Share ›

KK123 • 3 years ago

Here my recursive implementation its pretty easy than above one :)

```
#include<iostream>
using namespace std;
```

```
void rotate(int *arr, int d, int n, int index);

int main()
{
    int n, d, arr[100];
    cin >> n;
    for(int i=0; i> arr[i];

    cout <> d;

    rotate(arr, d, n, 0);

    for(int i=0; i<n; i++)
```

see more

• Reply • Share ›

GeeksforGeeks • 4 years ago

@Jagdish: This post only explains Block Swap Algorithm for array rotation. We of array rotation. Please see below

http://geeksforgeeks.org/?p=23...
http://geeksforgeeks.org/?p=28...

Also, please see http://www.cs.bell-labs.com/cm... for comparison of the stan

• Reply • Share ›

Jagdish • 4 years ago

this is so complex logic, here is a simple one

Reverse the two array [0,d] and [d,n] == { 2,1,7,6,5,4,3}
Reverse whole array = {3,4,5,6,7,1,2}

Asit ➚ Jagdish • 4 years ago

good logic!

ranga ➚ Asit • a year ago

Too good...Fantastic...Thanks alot.