Program 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

METHOD 1 (Use temp array)

```
Input arr[] = [1, 2, 3, 4, 5, 6, 7], d = 2, n =7
1) Store d elements in a temp array
   temp[] = [1, 2]
2) Shift rest of the arr[]
   arr[] = [3, 4, 5, 6, 7, 6, 7]
3) Store back the d elements
   arr[] = [3, 4, 5, 6, 7, 1, 2]
```

Time complexity O(n)

Auxiliary Space: O(d)

METHOD 2 (Rotate one by one)

```
leftRotate(arr[], d, n)
start
  For i = 0 to i < d
    Left rotate all elements of arr[] by one
end
```

To rotate by one, store arr[0] in a temporary variable temp, move arr[1] to arr[0], arr[2] to arr[1] …and finally temp to arr[n-1]

Let us take the same example arr[] = [1, 2, 3, 4, 5, 6, 7], d = 2

Rotate arr[] by one 2 times

We get [2, 3, 4, 5, 6, 7, 1] after first rotation and [3, 4, 5, 6, 7, 1, 2] after second rotation.

```
/*Function to left Rotate arr[] of size n by 1*/
void leftRotatebyOne(int arr[], int n);

/*Function to left rotate arr[] of size n by d*/
void leftRotate(int arr[], int d, int n)
{
  int i;
  for (i = 0; i < d; i++)
    leftRotatebyOne(arr, n);
}

void leftRotatebyOne(int arr[], int n)
{
  int i, temp;
  temp = arr[0];
  for (i = 0; i < n-1; i++)
     arr[i] = arr[i+1];
  arr[i] = temp;
}

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

/* 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;
}
```

Run on IDE

Time complexity: O(n*d)

Auxiliary Space: O(1)

METHOD 3 (A Juggling Algorithm)

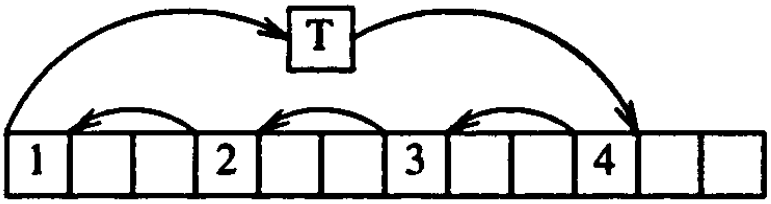
This is an extension of method 2. Instead of moving one by one, divide the array in different sets where number of sets is equal to GCD of n and d and move the elements within sets.

If GCD is 1 as is for the above example array (n = 7 and d =2), then elements will be moved within one set only, we just start with temp = arr[0] and keep moving arr[I+d] to arr[I] and finally store temp at the right place.

Here is an example for n =12 and d = 3. GCD is 3 and

```
Let arr[] be {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}

a)      Elements are first moved in first set – (See below diagram for this movement)
```




```
          arr[] after this step --> {4 2 3 7 5 6 10 8 9 1 11 12}

b)      Then in second set.
          arr[] after this step --> {4 5 3 7 8 6 10 11 9 1 2 12}

c)      Finally in third set.
          arr[] after this step --> {4 5 6 7 8 9 10 11 12 1 2 3}
```

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

/*Fuction to get gcd of a and b*/
int gcd(int a,int b);

/*Function to left rotate arr[] of siz n by d*/
void leftRotate(int arr[], int d, int n)
{
  int i, j, k, temp;
  for (i = 0; i < gcd(d, n); i++)
  {
    /* move i-th values of blocks */
    temp = arr[i];
    j = i;
    while(1)
    {
      k = j + d;
      if (k >= n)
        k = k - n;
      if (k == i)
        break;
      arr[j] = arr[k];
      j = k;
    }
    arr[j] = temp;
  }
}

/*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]);
}

/*Fuction to get gcd of a and b*/
int gcd(int a,int b)
{
   if(b==0)
     return a;
   else
     return gcd(b, a%b);
}

/* 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;
}
```

Run on IDE

Time complexity: O(n)

Auxiliary Space: O(1)

Please see following posts for other methods of array rotation:

Block swap algorithm for array rotation

Reversal algorithm for array rotation

References:

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

Please write comments if you find any bug in above programs/algorithms.

182 Comments Category: Arrays Tags: array

Related Posts:

- Longest Span with same Sum in two Binary arrays
- Count Inversions of size three in a give array
- Find the subarray with least average
- Count triplets with sum smaller than a given value
- Find zeroes to be flipped so that number of consecutive 1's is maximized
- Reorder an array according to given indexes
- Find maximum value of Sum(i*arr[i]) with only rotations on given array allowed
- Find maximum average subarray of k length