Maximum circular subarray sum

Given n numbers (both +ve and -ve), arranged in a circle, fnd the maximum sum of consecutive number.

Examples:

```
Input: a[] = {8, -8, 9, -9, 10, -11, 12}
Output: 22 (12 + 8 - 8 + 9 - 9 + 10)

Input: a[] = {10, -3, -4, 7, 6, 5, -4, -1}
Output:  23 (7 + 6 + 5 - 4 -1 + 10)

Input: a[] = {-1, 40, -14, 7, 6, 5, -4, -1}
Output: 52 (7 + 6 + 5 - 4 - 1 - 1 + 40)
```

There can be two cases for the maximum sum:

Case 1: The elements that contribute to the maximum sum are arranged such that no wrapping is there. Examples: {-10, 2, -1, 5}, {-2, 4, -1, 4, -1}. In this case, Kadane's algorithm will produce the result.

Case 2: The elements which contribute to the maximum sum are arranged such that wrapping is there. Examples: {10, -12, 11}, {12, -5, 4, -8, 11}. In this case, we change wrapping to non-wrapping. Let us see how. Wrapping of contributing elements implies non wrapping of non contributing elements, so find out the sum of non contributing elements and subtract this sum from the total sum. To find out the sum of non contributing, invert sign of each element and then run Kadane's algorithm.
Our array is like a ring and we have to eliminate the maximum continuous negative that implies maximum continuous positive in the inverted arrays.

Finally we compare the sum obtained by both cases, and return the maximum of the two sums.

Thanks to ashishdey0 for suggesting this solution. Following are C/C++, Java and Python implementations of the above method.

C/C++

```
// C/C++ program for maximum contiguous circular sum problem
#include<stdio.h>
```

```
// Standard Kadane's algorithm to find maximum subarray
// sum
int kadane(int a[], int n);

// The function returns maximum circular contiguous sum
// in a[]
int maxCircularSum(int a[], int n)
{
   // Case 1: get the maximum sum using standard kadane'
   // s algorithm
   int max_kadane = kadane(a, n);

   // Case 2: Now find the maximum sum that includes
   // corner elements.
   int max_wrap = 0, i;
   for (i=0; i<n; i++)
   {
        max_wrap += a[i]; // Calculate array-sum
        a[i] = -a[i];  // invert the array (change sign)
   }

   // max sum with corner elements will be:
   // array-sum - (-max subarray sum of inverted array)
   max_wrap = max_wrap + kadane(a, n);

   // The maximum circular sum will be maximum of two sums
   return (max_wrap > max_kadane)? max_wrap: max_kadane;
}

// Standard Kadane's algorithm to find maximum subarray sum
// See http://www.geeksforgeeks.org/archives/576 for details
int kadane(int a[], int n)
{
    int max_so_far = 0, max_ending_here = 0;
    int i;
    for (i = 0; i < n; i++)
    {
        max_ending_here = max_ending_here + a[i];
        if (max_ending_here < 0)
            max_ending_here = 0;
        if (max_so_far < max_ending_here)
            max_so_far = max_ending_here;
    }
    return max_so_far;
}

/* Driver program to test maxCircularSum() */
int main()
{
    int a[] =  {11, 10, -20, 5, -3, -5, 8, -13, 10};
    int n = sizeof(a)/sizeof(a[0]);
    printf("Maximum circular sum is %d\n",
                          maxCircularSum(a, n));
    return 0;
}
```

Java

```
// Java program for maximum contiguous circular sum problem
import java.io.*;
import java.util.*;

class MaxCircularSum
{
    // The function returns maximum circular contiguous sum
    // in a[]
    static int maxCircularSum (int a[])
    {
        int n = a.length;

        // Case 1: get the maximum sum using standard kadane'
        // s algorithm
        int max_kadane = kadane(a);

        // Case 2: Now find the maximum sum that includes
        // corner elements.
        int max_wrap  =  0;
        for (int i = 0; i < n; i++)
        {
            max_wrap += a[i]; // Calculate array-sum
            a[i] = -a[i];  // invert the array (change sign)
        }

        // max sum with corner elements will be:
        // array-sum - (-max subarray sum of inverted array)
        max_wrap = max_wrap + kadane(a);

        // The maximum circular sum will be maximum of two sums
        return (max_wrap > max_kadane)? max_wrap: max_kadane;
    }

    // Standard Kadane's algorithm to find maximum subarray sum
    // See http://www.geeksforgeeks.org/archives/576 for details
    static int kadane (int a[])
    {
        int n = a.length;
        int max_so_far = 0, max_ending_here = 0;
        for (int i = 0; i < n; i++)
        {
            max_ending_here = max_ending_here + a[i];
            if (max_ending_here < 0)
                max_ending_here = 0;
            if (max_so_far < max_ending_here)
                max_so_far = max_ending_here;
        }
        return max_so_far;
    }

        public static void main (String[] args)
        {
            int a[] =  {11, 10, -20, 5, -3, -5, 8, -13, 10};
            System.out.println("Maximum circular sum is " +
                            maxCircularSum(a));
        }
} /* This code is contributed by Devesh Agrawal*/
```

Python

```
# Python program for maximum contiguous circular sum problem

# Standard Kadane's algorithm to find maximum subarray sum
def kadane(a):
    n = len(a)
    max_so_far = 0
    max_ending_here = 0
    for i in range(0, n):
        max_ending_here = max_ending_here + a[i]
        if (max_ending_here < 0):
            max_ending_here = 0
        if (max_so_far < max_ending_here):
            max_so_far = max_ending_here
    return max_so_far

# The function returns maximum circular contiguous sum in
# a[]
def maxCircularSum(a):

    n = len(a)

    # Case 1: get the maximum sum using standard kadane's
    # algorithm
    max_kadane = kadane(a)

    # Case 2: Now find the maximum sum that includes corner
    # elements.
    max_wrap = 0
    for i in range(0,n):
        max_wrap += a[i]
        a[i] = -a[i]

    # Max sum with corner elements will be:
    # array-sum - (-max subarray sum of inverted array)
    max_wrap = max_wrap + kadane(a)

    # The maximum circular sum will be maximum of two sums
    if max_wrap > max_kadane:
        return max_wrap
    else:
        return max_kadane

# Driver function to test above function
a = [11, 10, -20, 5, -3, -5, 8, -13, 10]
print "Maximum circular sum is", maxCircularSum(a)

# This code is contributed by Devesh Agrawal
```

Output:

```
Maximum circular sum is 31
```

Time Complexity: O(n) where n is the number of elements in input array.

Note that the above algorithm doesn't work if all numbers are negative e.g., {-1, -2, -3}. It returns 0 in this case. This case can be handled by adding a pre-check to see if all the numbers are negative before running the above algorithm.

Please write comments if you find any of the above codes/algorithms incorrect, or find other ways to solve the same problem.

Ramraj
Cotton
Now Visit us at Malleswaram. Showroom Opened. Visit Us Today.
www.ramrajcotton.in

113 Comments Category: Arrays

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

(Login to Rate and Mark)

3 Average Difficulty : **3/5.0**
Based on **5** vote(s)

☐ Add to TODO List
☐ Mark as DONE

Like Share 7 people like this. Be the first of your friends.

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.