Group -8

Assignment 3

Bratin Mondal (21CS10016)
Somya Kumar (21CS30050)
Submitted on: August 14th, 2023

Topic: Max Circular Subarray Sum

Given an integer circular array arr of size N, find the contiguous subarray which has the largest sum and return its sum.

Algorithm:

From a circular array, finding the maximum subarray can be solved using brute force by exhaustively looking over all possible subarrays, which will have exponential time complexity.

We will devise an algorithm with *linear time* complexity.
We know that for normal subarrays (not circular), we can find the maximum subarray sum and also the minimum subarray sum using Kadane's Algorithm in linear time.
For a circular subarray, the subarray with maximum sum has two possibilities:

1. It doesn't have the elements arr[0] and arr[N-1] simultaneously.
   In this case, the problem reduces to the case of normal subarrays (not circular) and finding the maximum subarray sum for it.
2. It has the elements arr[0] and arr[N-1] simultaneously and some neighboring elements.
   In this case, we can observe that the remaining of array will be the subarray with the possible minimum sum.
   So, in this case, the problem reduces to finding the minimum subarray sum for the normal array case. Then we can subtract it from the total sum of all the elements in the area and get our answer.

So our algorithm will check both cases using Kadane's algorithm and return the maximum of both the answer.

Kadane's Algorithm:

For Max Subarray Sum:

The intuition of the algorithm is not to consider the subarray as a part of the answer if its sum is less than 0. A subarray with a sum less than 0 will always reduce our answer, and so this type of subarray cannot be a part of the subarray with a maximum sum.
Here, we will iterate the given array with a single loop, and while iterating, we will add the elements in a sum variable. Now, if at any point the sum becomes less than 0, we will set the sum as 0 as we are not going to consider any subarray with a negative sum. Among all the sums calculated, we will consider the maximum one.
Thus we can solve this problem with a single loop.
Let the ans of it be max_arr

Pseudo-Code:

```
for (int i = 0; i < n; i++)
{
       sum += arr[i]

       if (sum > maxi)
       {
            maxi = sum
       }
       if (sum < 0)
       {
           sum = 0
       }
 }
```

For Min Subarray Sum:

The intuition of the algorithm is not to consider the subarray as a part of the answer if its sum is greater than 0. A subarray with a sum greater than 0 will always increase our answer, and so this type of subarray cannot be a part of the subarray with a minimum sum.
Here, we will iterate the given array with a single loop, and while iterating, we will add the elements in a sum variable. Now, if at any point the sum becomes greater than 0, we will set the sum as 0 as we are not going to consider any subarray with a positive sum. Among all the sums calculated, we will consider the minimum one.
Thus we can solve this problem with a single loop.
Let the ans of it be min_arr

Pseudo-Code:

```
for (int i = 0; i < n; i++)
{
       sum += arr[i]

       if (sum < mini)
       {
            mini = sum
       }
       if (sum > 0)
       {
           sum = 0
       }
 }
```

Also, let the sum of all the elements in the array be sum_arr
Finally, Our algorithm will return max (max_arr, sum_arr - min_arr)