Kyle Ryan
CMPT435
5 April 2021

Assignment 7

1. Given an array A of n elements, find out the maximum difference between any two elements such that larger element appears after the smaller element in A. In other words, find a pair of elements A[p], A[q] with q>p such that (A[q] – A[p]) is the maximum among all such pairs in A.

For example, if array is [2, 3, 10, 6, 4, 8, 1] then the maximum difference should be 8 (Diff between 10 and 2).

If array is [7, 9, 1, 6, 3, 2] then the maximum difference should be 5 (Diff between 1 and 6).

Design a divide-and-conquer algorithm of O(n) time complexity to solve this problem.

(i) describe the idea behind your algorithm in English (1 point);
My algorithm is based on the divide-and-conquer strategy. We will divide the input array in half, left and right, and keep track of the maximum and minimum values. From there, the only thing left to do is to return the difference between the two.

(ii) provide pseudocode (4 points);

```
mid = (minDiff + maxDiff) / 2

if (maxDiff != (minDiff + 1)
        if (maxDiff == minDiff)
                diffTemp.diffNum = 0
                diffTemp.minNum = A[minDiff]
                diffTemp.MaxNum = A[minDiff]
                return diffTemp
        end if

        lSplit = findmaxdiff(A, minDiff, mid)
        rSplit = findmaxdiff(A, mid + 1, maxDiff)

        diffTemp.MaxNum = rSplit.MaxNum
        diffTemp.minNum = lSplit.minNum
        diffTemp.diffNum = diffTemp.MaxNum - diffTemp.minNum

        if (lSplit.diffNum <= diffTemp.diffNum)
                if (rSplit.diffNum > diffTemp.diffNum)
                        diffTemp.diffNum = rSplit.diffNum
                end if
```

```
    end if
    else
            diffTemp.diffNum = Math.max(lSplit.diffNum, rSplit.diffNum)
    end else

    diffTemp.MaxNum = Math.max(rSplit.MaxNum, lSplit.MaxNum)
    diffTemp.minNum = Math.min(rSplit.minNum, lSplit.minNum)

    return diffTemp
  end if

    else
    if (A[minDiff] >= A[maxDiff])
            diffTemp.diffNum = 0
            diffTemp.MaxNum = A[minDiff]
            diffTemp.minNum = A[maxDiff]

    end if
    else
            diffTemp.diffNum = A[maxDiff] - A[minDiff]
            diffTemp.MaxNum = A[maxDiff]
            diffTemp.minNum = A[minDiff]
    end else
    return diffTemp;
    end else
```

(iii) analyze its running time (2 points).

The runtime of this algorithm, being a divide and conquer algorithm, is O(n). The divide part of the algorithm, the recursive step, splits the search space in half each time.

Regarding requirement (iii): Unless otherwise specified, show the steps of your analysis and present your result using big-O. Note: Full credit (7 points) will be awarded for a divide-and-conquer algorithm that is O(n). Algorithms that are NOT divide-and-conquer or slower than O(n) will be scored out of 2 points.