Kyle Ryan
CMPT435
31 March 2021

Assignment 6

1. Using the master theorem discussed in class, find a tight bound for the solution of the following recurrence equation (3 points each).

a. T(n) = 2T(n/2) + n3
3 > log(2)2 → 3 > 1 → O(n^3)

b. T(n) = T(9n/10) + n
1 > log(10)1 → 1 > 0 → O(n)

c. T(n) = 16T(n/4) + n2
2 = log(4)16 → 2 = 2 → O(n^2logn)

d. T(n) = 7T(n/3) + n2
2 > log(3)7 → 2 > 1.7 → O(n^2)

e. T(n) = 2T(n/4) + sqrt(n)
½ = log(4)2 → ½ = ½ => O(SQRT(n)logn)

3. We are given an array of n numbers A in an arbitrary order. Design an algorithm to find the largest and second largest number in A using at most 3n/2 -2 comparisons.

(i) describe the idea behind your algorithm in English (1 point);
The idea behind my algorithm is to divide the problem into subproblems and compare the max values to get the max and the second max. It will continue to break down the problem until I am able to find the max and second max in the entire given array.

(ii) provide pseudocode (4 points);

```
Max2ndMax pair = new Max2ndMax()
if(j == i)
    pair.max = A[i]
    pair.max2nd = -1
    return pair
end if

int mid = (i + j)/2;
Max2ndMax pair1 = dcfindmax2ndmax(A, i, mid)
Max2ndMax pair2 = dcfindmax2ndmax(A, mid+1, j)

if (pair1.max > pair2.max)
    pair.max = pair1.max
    pair.max2nd = Math.max(pair2.max, pair1.max2nd)
end if

else if (pair2.max > pair1.max)
    pair.max = pair2.max
    pair.max2nd = Math.max(pair1.max, pair2.max2nd)
end else if

else
    pair.max = pair1.max
    pair.max2nd = Math.max(pair.max2nd, pair2.max2nd)

return pair;
```

(iii) analyze the number of comparisons used in your algorithm (2 points).
The number of comparisons in my algorithm is 3n/2-2 because it is built on the concept of the divide-conquer Minmax approach.

Note: The divide-conquer Minmax algorithm makes 3n/2-2 comparisons to find min and max in an array – we will discuss the proof next week. For this problem, you are expected to design an algorithm similar to Minmax to find max and 2nd max in an array.

Note: Full credit (7 points) will be awarded for an algorithm that uses at most 3n/2 -2 comparisons. Algorithms that make more than 3n/2 -2 comparisons will be scored out of 2 points.