CSE 551
Assignment 2

Arun Kumar Kumarasamy

October 4, 2020

Q1 Solution:

Algorithm for finding second smallest number:
Recursively find the smallest value and return back a list of numbers that were directly compared with the smallest number. Now find the smallest from this list.
Algorithm:
FindSecondSmallest(InputArray[ ]**):**
begin
    {min, ComparisonsMade} ← FindSmallest(0,InputArray.size(), InputArray[ ]);
    {secondMin, ComparisonsMade} ← FindSmallest(0,ComparisonsMade.size(), ComparisonsMade[]);
    return secondMin
end

FindSmallest(i, j, InputArray[ ]**):**
begin
    if (i == j):   #recursion base condition
        min ← InputArray[i];
        comparisionsMade = new Array[ ];
        return {min, comparisonsMade[ ]}

    {min1,comparisonsMade1} ← FindSmallest(i,i+(j-i)/2,InputArray[ ]);
    {min2,comparisonsMade2} ← FindSmallest(i+(j-i)/2,j,InputArray[ ]);

    if (min1 < min2):
        min ← min1;
        comparisonsMade1.add(min2);
        return { min, comparisonsMade1}
    else:
        min ← min2;
        comparisonsMade2.add(min1);
        return { min, comparisonsMade2}
end

The above algorithm finds the smallest element in the first FindSmallest call, which along with the smallest number returns the list of numbers that the smallest number was compared with.
Since all the numbers are compared in the first call,
we have $\boldsymbol{N-1}$ comparisions ——————- (1)
Now, Since the FindSmallest returns a list of elements to which the smallest was compared with, and since the FindSmallest function is a recursive function with total number of recursive calls: $\boldsymbol{\log_2(N)}$, each recursive call makes only one comparision. So the maximum no. of elements that can be added

to the "comparisonsMade" list is $\boldsymbol{\log_2(N)}$.
The second call to FindSmallest would find the smallest in the "comparisonsMade" list which would make $\boldsymbol{\log_2(N) - 1}$ comparisions. ————————(2)
Hence the total number of comparisons made by this algorithm is (1) + (2), which is

$$N - 1 + \log_2(N) - 1$$

$$\boldsymbol{N + \log_2(N) - 2}$$

Q2 Solution:

we know that Conventional matrix multiplication method makes 27 multiplications.
By using Strassen's algorithm for matrix multiplication, we can get the computation time to $O(n^{2..81})$
Strassen's algorithm is used for matrices of power of 2, i.e., a recursive approach to multiply any 2x2 matrices with recursive equation:

$$T(n) = 7T(n/2) + O(n^2)$$

Since we need the number of multiplications for a 3x3 matrix multiplication, we have

$$T(n) = kT(n/3) + O(n^2)$$

where k is the no. of mulitiplication
So, from the above recurrence relation, we need to solve for k such that

$$O(n^{\log_3(k)}) = O(n^{2.81})$$

$$n^{\log_3(k)} = n^{2.81}$$

$$\log_3(k) = 2.81$$

$$k = 21.91344$$

the largest k value for 3x3 matrix for which the computation time is approximately equal to $O(n^{2.81})$ is $\mathbf{k = 21}$
Doing the same for 4x4 matrices,

$$T(n) = kT(n/4) + O(n^2)$$

Equating,

$$O(n^{\log_4(k)}) = O(n^{2.81})$$

$$n^{\log_4(k)} = n^{2.81}$$

$$\log_4(k) = 2.81$$

$$k = 49.18$$

Hence for 4x4 matrices, the **no.of multiplications = 49**

Q3 Solution:

Proof by Induction
We need to prove that the given recurrence has the following solution,

$$T(n) = 3kn^{\log_2(3)} - 2kn$$

given: n is a power of 2 so let $n = 2$ Substituting it in the given equation, we get

$$T(2) = 3T(2/2) + 2k$$

$$T(2) = 3T(1) + 2k$$

given, $T(1) = k$ hence,

$$T(2) = 5k$$

Substituting the same n=2 in solution equation, we get,

$$T(n) = 3k2^{\log_2(3)} - 2k2$$

$$T(n) = 5k$$

Hence, solution holds for the base case (n=2)
Now let $n = 2n$
Substituting $n = 2n$ we get,

$$T(2n) = 3T(2n/2) + k(2n)$$

$$T(2n) = 3T(n) + k(2n)$$

By substituting $T(n) = 3kn^{\log_2(3)} - 2kn$, we get,

$$T(2n) = 3(3kn^{\log_2(3)} - 2kn) + k(2n)$$

$$T(2n) = 3 * 3kn^{\log_2(3)} - 6kn + 2kn)$$

$$T(2n) = 3 * 3kn^{\log_2(3)} - 4kn$$

Substituting 3 with $2^{\log_2(3)}$, we get,

$$T(2n) = 3 * (2^{\log_2(3)})kn^{\log_2(3)} - 2k(2n)$$

$$T(\mathbf{2n}) = 3k(\mathbf{2n})^{\log_2(3)} - 2k(\mathbf{2n})$$

Hence the above equation is equal to $T(n) = 3kn^{\log_2(3)} - 2kn$ where $n = 2n$
Hence proved.

Q4 Solution:

Let's represent the matrix as

$$V = \begin{bmatrix} v1 & v2 \\ v3 & v4 \end{bmatrix}$$

$$W = \begin{bmatrix} w1 & w2 \\ w3 & w4 \end{bmatrix}$$

V x W =

$$C = \begin{bmatrix} c1 & c2 \\ c3 & c4 \end{bmatrix}$$

Regular matrix multiplication would multiply it this way,

$$c1 = v1w1 + v2w3$$

$$c2 = v1w2 + v2w3$$

$$c3 = v3w1 + v3w3$$

$$c4 = v3w2 + v4w4$$

Lets consider two vectors and their vector product

$$V = \begin{bmatrix} v1 & v2 \end{bmatrix}$$

$$W = \begin{bmatrix} w1 \\ w3 \end{bmatrix}$$

Applying the vector product formula: we get,

$$VW = (v1 + w3) \times (v2 + w1) - (v2 \times v1) - (w3 \times w1)$$

which happens to be c1 (Another representation of c1)

Similarly, applying the same formula for other vectors, we can represent c1, c2, c3 and c4 as following:

$$c1 = (v1 + w3) \times (v2 + w1) - (v2 \times v1) - (w3 \times w1)$$

$$c2 = (v1 + w4) \times (v2 + w2) - (v2 \times v1) - (w4 \times w2)$$
$$c3 = (v3 + w3) \times (v4 + w1) - (v3 \times v4) - (w3 \times w1)$$
$$c4 = (v3 + w4) \times (v2 + w2) - (v3 \times v4) - (w4 \times w2)$$

This way we reduce the number of multiplication required, by computing the later part of the formula once and using it many times.