Assignment 3: CSE548: Analysis of Algorithms

Shivasagar Boraiah — #112077826

October 23, 2018

Problem 1(a). Textbook [Kleinberg & Tardos] Chapter 5, page 246, problem #3.

Proof. ...
Reduce the problem into n/2 equal sub-parts. Let, B1, B2....Bj denote the equivalence blocks of cards such that if two cards are equal, they belong to the same block, Bi or Bi=Bj. Using divide-and-conquer approach, lets find the pivot. 'p' recursively from each subpart such that atleast n/2 cards are equivalent to p. There may exist an equivalence block with majority in one subpart with index equal to p. If found, return the card by testing it with other block of cards for the majority. As the problem can be divided into two subproblems and run recursively in finding the card with majority equivalence doing atmost of 2n comparisons, we get the recurrence equation to be, $T(n) = 2T(n/2) + 2n$ which gives us the complexity of $O(nlogn)$ using Master's method.

Divide the cards recursively into two sub-problems ;
For set1, compare recursively to find p equal indices;
    If the block with majority equivalence found, compare with all other blocks;
If no card with majority found;
    compare recursively to find p equal indices;
        If the block with majority equivalence found, compare all other blocks;
return the card from majority block;

The existence of majority equivalence block in either set constitute for the majority equivalence block of at least one of the block proves the correctness of an algorithm. □

Problem 1(b). Textbook [Kleinberg & Tardos] Chapter 5, page 246, problem #5.

Proof. ...
I have discussed this problem with Shivanshi Gupta.
Given the set of lines, begin by sorting them in increasing order. Then, using divide-and-conquer approach, recursively split the lines into 2 equal halves and calculate the 'VisibilityPortion' for each subproblem and combine the two blocks such that the slopes of the 'VisibilityPortion' in later block is greater than that of the first due to the increasing order of sorting. So, the VisibilityPortion should intersect while merging.

Sort the lines in the increasing order of slopes, (L_1, L_2, L_3....L_n);
return the VisibilityPoints and Intersection points in increasing order for block one, L=(L_{i1},L_{i2},...L_{im});
recursively find the visibilitypoints and intersection points for later half of the problem, L'=(L_{j1},L_{j2},...L_{jm});
Let (x',y') denote the point the plane at which L_{is} and L'_{jt} intersect. Therefore, L_{is} will be left to x' and L'_{jt} will be right to x'. Therefore, the lines which become available are, (L_{i1},L_{i2},...L_{is}), (L_{jt},L_{j2},...L_{jm}) with intersection points (p_{i1}, p_{i2}....p_{is-1}) (x',y'),(q_{jt}, q_{i2}....$q_{i(m-1)}$)

as we found the VisibilityPoints and Intersection points which is the termination of the problem, the algorithm is correct and complete. □

Problem 1(c). Textbook [Kleinberg & Tardos] Chapter 5, page 246, problem #6.

Proof. ...
Givena tree, T a local minimum can be found by traversing from root to node. We return the local minimum node, n it is less than its parent value and less than its children value.

```
LocalMinimum(T) ;
    return NULL if root is null;
    return the root if it is lesser than its children;
    compare recursively to find p equal indices;
    Find minimum(root− >children) and check the condition for local minimum;
        If satisfies return node;
    Else LocalMinimum(minimum(root− >children));
```

As we dig dive only upto the height of the tree in finding localminimum, worst case the time complexity of the algorithm will be, $O(logn)$.

While the Tree traversal, if we reach root, then possibly it is the last node and it has to be local minimum of all. So, this constitute for the correctness of an algorithm. □

Problem 3. Square of a Matrix. The square of a matrix A is its product with itself, AA.

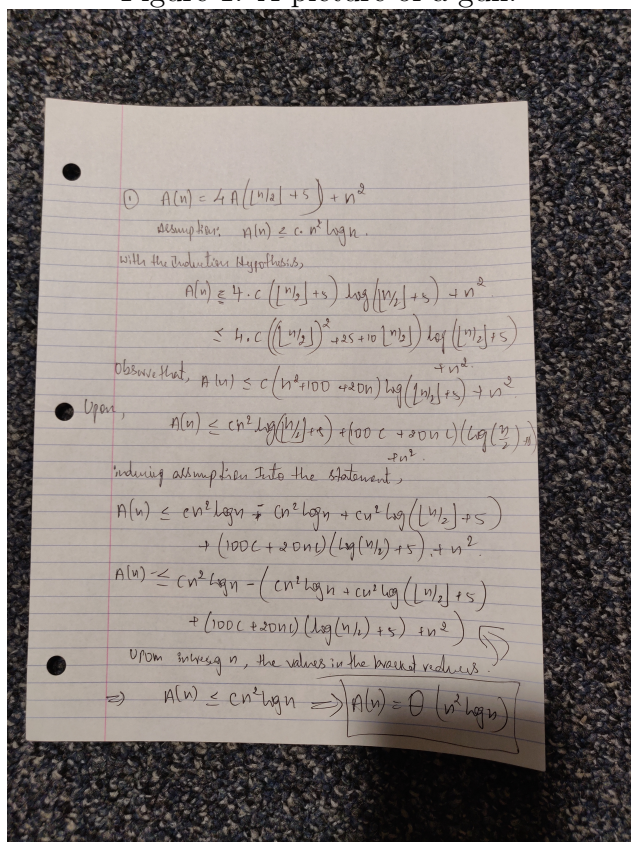
3(a)

Proof. ..
Problem Statement:
5 multiplications are sufficient to compute a product of 2 x 2 Matrix. Consider a matrix A with elements {a, b, c, d} and let us try to deduce the required solution with an example.

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} * \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} a^2 + b * c & b * (a + d) \\ c * (a + d) & b * c + d^2 \end{bmatrix}$$

$$A = \begin{bmatrix} a^2 + B & a * (b + d) \\ a * (c + d) & B + d^2 \end{bmatrix} \text{ where B = b*c}$$

Figure 1: A picture of a gull.



① $A(n) = 4A(\lfloor n/2 \rfloor + 5) + n^2$

Assumption: $A(n) \le c \cdot n^2 \log n$.

With the Induction Hypothesis,

$$A(n) \le 4 \cdot c(\lfloor n/2 \rfloor + 5) \log(\lfloor n/2 \rfloor + 5) + n^2$$

$$\le 4 \cdot c((\lfloor n/2 \rfloor)^2 + 25 + 10\lfloor n/2 \rfloor) \log(\lfloor n/2 \rfloor + 5) + n^2.$$

Observe that, $A(n) \le c(n^2 + 100 + 20n) \log(\lfloor n/2 \rfloor + 5) + n^2$.

Upon,

$$A(n) \le cn^2 \log(\lfloor n/2 \rfloor + 5) + (100c + 20nc)(\log(\tfrac{n}{2}) + 5) + n^2.$$

inducing assumption into the statement,

$$A(n) \le cn^2 \log n - cn^2 \log n + cn^2 \log(\lfloor n/2 \rfloor + 5) + (100c + 20nc)(\log(n/2) + 5) + n^2.$$

$$A(n) \le cn^2 \log n - (cn^2 \log n + cn^2 \log(\lfloor n/2 \rfloor + 5) + (100c + 20nc)(\log(n/2) + 5) + n^2)$$

Upon increasing n, the values in the bracket reduces.

$$\Rightarrow A(n) \le cn^2 \log n \Rightarrow \boxed{A(n) = \Theta(n^2 \log n)}$$

from the above derivation, 5 multiplications { a*a, b*c, d*d, b*(a+d), c*(a+d)} are sufficient to compute 2 x 2 Matrix square. Now the problem reduced to five sub-problems and each sub-problem takes a time of $O(n^2)$ for every n*n matrix.
Hence the recurrence for the above problem can deduced to $T(n) = 5T(n/2) + O(n^2)$ which gives the time complexity of $O(n^{\log_2 5})$ □

3(b)

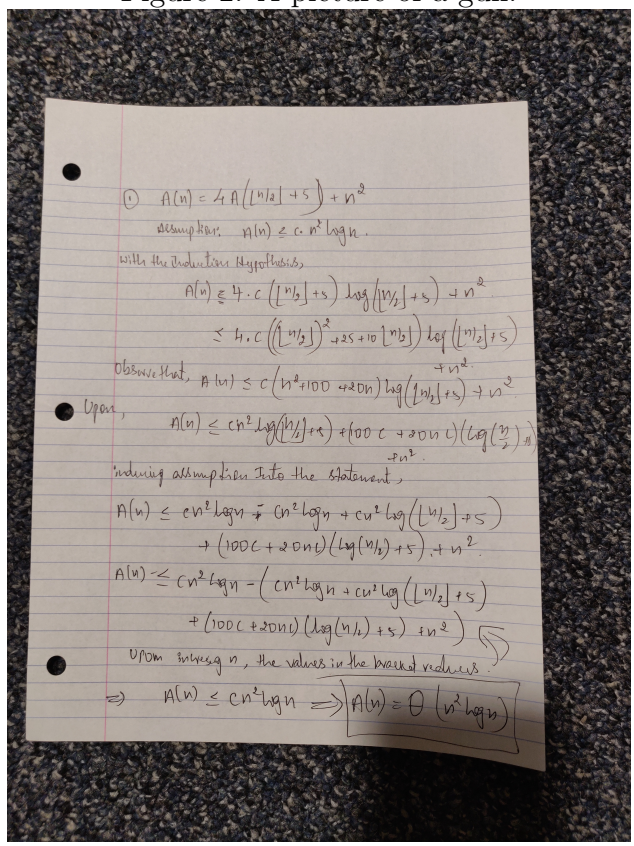
Proof. ..

Using divide and conquer algorithm, we divide a nxn matrix into 4 blocks, and recursively calculate n times until the problem reduced to 2x2 or 1x1. Such small size problems (2x2), can be calculated using 5 sub-problems itself as of 3(a). But, as we proceed in combining such small problems, we cannot assume that we can calculate in 5 sub-problems because the combined sub-problems might not be of same type as the original sub-problems and require to calculate in $O(n^{\log_2 7})$ time. Also, In Matrix multiplication, the commutativte property is not true, so we cannot reuse, AB as BA, given A and B are elements if Matrix. i, e.

$$L = \begin{bmatrix} a & b \\ c & d \end{bmatrix} * \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} a*a+b*c & a*c+b*d \\ c*a+c*d & c*d+d*d \end{bmatrix} ! = \begin{bmatrix} a^2+b*c & a*(b+d) \\ a*(c+d) & b*c+d^2 \end{bmatrix}$$ □

3(c)

Figure 2: A picture of a gull.



① $A(n) = 4A(\lfloor n/2 \rfloor + 5) + n^2$

assumption: $A(n) \le c \cdot n^2 \log n$.

With the Induction Hypothesis,

$$A(n) \le 4 \cdot c(\lfloor n/2 \rfloor + 5) \log(\lfloor n/2 \rfloor + 5) + n^2$$

$$\le 4 \cdot c((\lfloor n/2 \rfloor)^2 + 25 + 10\lfloor n/2 \rfloor) \log(\lfloor n/2 \rfloor + 5) + n^2$$

Observe that, $A(n) \le c(n^2 + 100 + 20n) \log(\lfloor n/2 \rfloor + 5) + n^2$

Upon,

$$A(n) \le cn^2 \log(\lfloor n/2 \rfloor + 5) + (100c + 20nc)(\log(\tfrac{n}{2}) + 5) + n^2$$

inducing assumption into the statement,

$$A(n) \le cn^2 \log n - cn^2 \log n + cn^2 \log(\lfloor n/2 \rfloor + 5) + (100c + 20nc)(\log(n/2) + 5) + n^2$$

$$A(n) \le cn^2 \log n - (cn^2 \log n + cn^2 \log(\lfloor n/2 \rfloor + 5) + (100c + 20nc)(\log(n/2) + 5) + n^2)$$

Upon increasing n, the values in the bracket reduces.

$$\Rightarrow A(n) \le cn^2 \log n \Rightarrow \boxed{A(n) = \Theta(n^2 \log n)}$$

Proof. ..
consider two nxn matrices A and B and compute 2n x 2n matrix multiplication to form matrix C as:

$C = \begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix}$ If we conpute C^2, it will contain AB in its diagonal block as, $C^2 = \begin{bmatrix} AB & 0 \\ 0 & BA \end{bmatrix}$

The product AB can be calculated in the time, $O(S(2n))$ time, if squaring a matrix, S(n) takes $O(n^c)$ time, then $O(S(2n))$ takes same time too. □

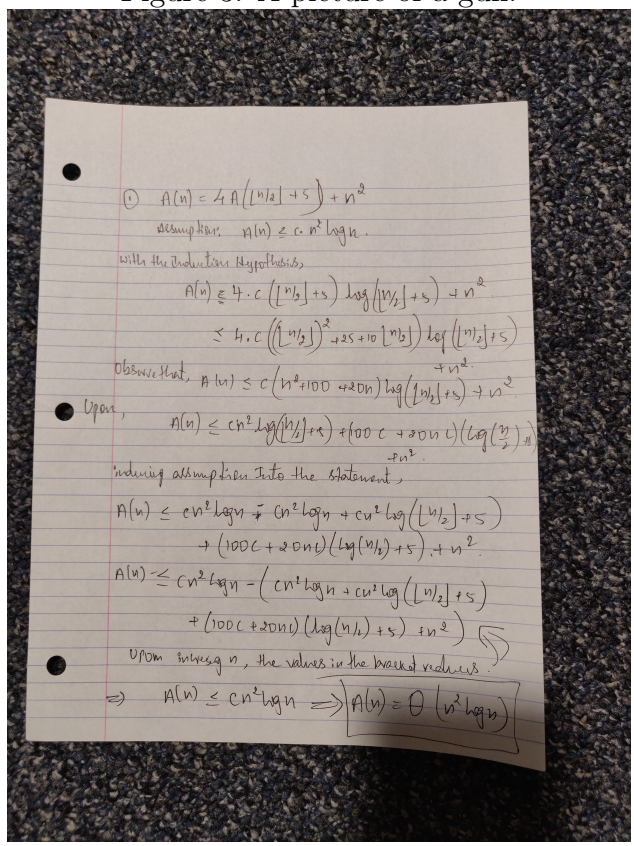
3(c)(i)

Proof. ...
Consider the equation: $(A+B)^2$=A^2+B^2+$2AB$=A^2+B^2+AB+BA
therefore, AB+BA=$(A+B)^2$-A^2-B^2, if A and B are matrices of size n, then we can make use of the above derivation to calculate AB+BA, which involves squaring of three problems of time $S(n)$ as described in 3(c). So, the overall computation comes down to $3S(n)+O(n^2)$. where $O(n^2)$ is the charge for every multiplication. □

3(c)(ii)

Proof. ..
Given two nXn matrices, X and Y, and A and B is defined, Using that need to compute AB

Figure 3: A picture of a gull.



① $A(n) = 4A(\lfloor n/2 \rfloor + 5) + n^2$

Assumption: $A(n) \le c \cdot n^2 \log n$.

With the Induction Hypothesis,

$$A(n) \le 4 \cdot c(\lfloor n/2 \rfloor + 5) \log(\lfloor n/2 \rfloor + 5) + n^2$$

$$\le 4 \cdot c\left(\left(\lfloor n/2 \rfloor\right)^2 + 25 + 10\lfloor n/2 \rfloor\right) \log(\lfloor n/2 \rfloor + 5) + n^2.$$

Observe that, $A(n) \le c(n^2 + 100 + 20n) \log(\lfloor n/2 \rfloor + 5) + n^2$.

Upon,

$$A(n) \le cn^2 \log(\lfloor n/2 \rfloor + 5) + (100c + 20nc)(\log(\tfrac{n}{2}) + 5) + n^2.$$

inducing assumption into the statement,

$$A(n) \le cn^2 \log n - cn^2 \log n + cn^2 \log(\lfloor n/2 \rfloor + 5) + (100c + 20nc)(\log(n/2) + 5) + n^2.$$

$$A(n) \le cn^2 \log n - \left(cn^2 \log n + cn^2 \log(\lfloor n/2 \rfloor + 5) + (100c + 20nc)(\log(n/2) + 5) + n^2\right)$$

Upon increasing n, the values in the bracket reduces.

$$\Rightarrow A(n) \le cn^2 \log n \Rightarrow \boxed{A(n) = \Theta(n^2 \log n)}$$

+ BA.

$$A * B = \begin{bmatrix} X & 0 \\ 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & Y \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & XY \\ 0 & 0 \end{bmatrix}$$

$$B * A = \begin{bmatrix} 0 & Y \\ 0 & 0 \end{bmatrix} * \begin{bmatrix} X & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$A * B + B * A = \begin{bmatrix} 0 & XY \\ 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & XY \\ 0 & 0 \end{bmatrix}$$

□

3(c)(iii)

Proof. ...

The product X*Y can be computed by evaluating AB*BA as per (ii). From, (i), the overall time AB+BA can be computed is $3S(2n)+O(n^2)$ for n=2, with $O(n^2)$ being the computation time required for 2x2 matrix multiplication.

with c>=2, from the above derivations we can conclude that matrix multiplication takes $O(n^c)$ with S(n)=$O(n^c)$ for c>=2.

□