CS 430 HW1 **GANESH PRASAD CHANDRA SHEKAR, A20557831**

1. Compute n^{15} using 6 multiplications.

- For computing, let us take 6 variables, 5 variables to store the intermediate results and 1 to store the total result.
  We shall name the variables as *a, b, c, d, e* and final result as *Total*

  Step 1: First, we shall calculate n^2 i.e., n*n and store in variable *a*
  a = $\mathbf{n^2}$ (Multiplication count: **1**)
  Step 2: Now, use the variable *a* and multiply with n and store the result in variable *b*
  $b = a * n = (n^2) * n$ which is equivalent to $\mathbf{n^3}$ (Multiplication count: **1**)
  Step 3: Again, square the value stored in variable *b* and store result in *c*, which results in n^6
  $c = b*b = (n^3)^2 = \mathbf{n^6}$ (Multiplication count: **1**)
  Step 4: Square variable *c* and then store the result in variable *d*
  $d = c^2 = (n^6)^2$, which is equivalent to $\mathbf{n^{12}}$ (Multiplication count: **1**)
  Step 5: Multiply the value of *d* with *a* and store the result in variable *e*
  $e = d * a = n^{12} * n^2$, which is equivalent to $\mathbf{n^{14}}$ (Multiplication count: **1**)
  Step 6: In the final step, multiply variable *e* with n and store the result in *Total* variable.
  Total = variable *e* * variable *c* = $n^{14} * n = \mathbf{n^{15}}$ (Multiplication count: **1**)

  Thus with **6** multiplications, we have obtained the value of n^{15}.

2. Sort 5 distinct integers using at most 7 comparisons.

- For this arrangement it would be suitable to use "Bubble sort" algorithm.
  Let the integers to sort be a_1, a_2, a_3, a_4, a_5.

  Step 1: Compare a_1 and a_2 and, if $a_1 > a_2$ swap the position

a_1	a_2	a_3	a_4	a_5

  Step 2: Compare a_3 and a_4 and, if $a_3 > a_4$ swap the position

a_1	a_2	a_3	a_4	a_5

  Step 3: Compare a_2 and a_3 and, if $a_2 > a_3$ swap the position

a_1	a_2	a_3	a_4	a_5

  Step 4: Compare a_4 and a_5 and, if $a_4 > a_5$ swap the position

a_1	a_2	a_3	a_4	a_5

  Step 5: Compare a_2 and a_4 and, if $a_2 > a_4$ swap the position

a_1	a_2	a_3	a_4	a_5

  Step 6: Compare a_1 and a_4 and, if $a_1 > a_4$ swap the position

a_1	a_2	a_3	a_4	a_5

  Step 7: Compare a_1 and a_3 and, if $a_1 > a_3$ swap the position

a_1	a_2	a_3	a_4	a_5

We have sorted 5 distinct numbers in 7 comparisons above. With Bubble sort, we start by sorting the first adjacent elements by comparing (if the value of the first element is greater than the second then swap their positions) and continue arranging the next adjacent pair of elements until the end of the array. After the first iteration the biggest element will be present at the end of the array. Then except the last element in the array we shall again compare and sort the array elements. These steps will be done until no swaps are required. The array/list will now be arranged in ascending order.

3. Suppose T (0) = 0 and T (n) <= $\sqrt{n}$T ($\sqrt{n}$) + n for each n >= 1. Show that T (n) = O (n log log n).

- This Recurrence relation can be solved using Master Theorem,

Taking, T(n) equation as T(n) = $\sqrt{n}$T ($\sqrt{n}$) + n

Assuming the value of n= 2^m which equates to $\sqrt{n}$=$2^{m/2}$ and also m= log n,
Substituting the values in the above equation, the result will be:
$T(2^m) = 2^{m/2}T(2^{m/2}) + 2^m$……(1)
Dividing the above equation by 2^m, the resulting equation would be:
$T(2^m)/2^m = (2^{m/2}\ T(2^{m/2})/2^m)+ 1$……..(2)

Upon simplifying,
$T(2^m)/2^m = T(2^{m/2})/2^{m/2} + 1$……(3)

Let $z(m)= T(2^m)/2^m$……..…(4)
then (3) becomes,
z(m) = z(m/2)+1……..(5)

Equation (5) is in the form of Master Theorem,
$T(n)= aT(n/b)+n^d$
The values for a, b and d will be 1, 2 and 0 respectively.

Since the value d is, $d= \log_a b$ i.e., $0= \log_2 1$. We can go with ***Case 2*** of the Master Theorem.
Case 2 states that, if f(n)= θ(n^($\log_b a$)) then T(n)= $\theta(n^d * \log n)$

$z(m)= m^d \log m$, since d=0, then z(m) will be z(m)= log m……(6)

We know that from equation (4), $T(2^m)= 2^m\ z(m)$
Then substituting value of z(m) from (6),
$T(2^m) = 2^m \log m$

Since $n = 2^m$ and m = log n,
We get, **T(n) = θ(n log log n)**

4.

The following recurrence relation arises from the average-case analysis of Quicksort: $T(0) = T(1) = 0$; and for each $n \geq 2$,

$$T(n) = n - 1 + \frac{1}{n}\sum_{i=1}^{n}[T(i-1) + T(n-i)]$$

Show that for each $n \geq 1$, $T(n) = 2(n+1)H(n) - 4n$ where $H(n) = \sum_{i=1}^{n}\frac{1}{k}$ is the n-th harmonic number.

- With the help of Induction, we can prove T(n) = 2(n+1) H(n) – 4n

Base Case:
For n=1,
T(1)=0; equation becomes 2(1+1) H(1)- 4, Where H(1)= 1
T(0) = T(1) = 4 - 4 = 0
The base case holds true.

Inductive Hypothesis:
Assuming the statement is true for n=k, then
T(k) = 2(k+1)H(k) – 4k

Inductive Step:
Using the recurrence relation we have to prove that for n=k+1,
T(k+1) = 2(k+2)H(k+1)- 4(k+1)

Recurrence relation states that,
$T(k+1) = (k+1-1) + 1/(k+1) \sum^{k+1}_{i=1} [T(i-1)+ T(k+1-i)]$
Expanding the summation on RHS the above equation will turn out be as below,
T(k+1) = k+ (1/(k+1))*(T(0)+T(1)+ T(2)+……+ T(k-1)+ T(k-1)+ T(k-2)+.. T(0))

We know that, T(0) = T(1) =0, then the above equation becomes,
T(k+1) = k+ (1/(k+1)) * 2(T(k-1) + T(0))

Rearranging the equation which we got in the previous step,
T(k+1) = (k+2) + 2 /(k+1) * T(k-1)

By Inductive Hypothesis, we have T(k-1) = 2(k)H(k) – 4k, using the values in the above step,
T(k+1) = (k + 2) + 2/ (k+1) * (2(k) H(k) – 4k)

Which can be further simplified as,
T(k+1) = 2(k+1)H(k+1) – 4(k+1)

Which is precisely, T(k+1) = 2(k+2) H(k) – 4(k+1), therefore it is proven by mathematical induction that if T(k) = 2(k+1) H(k) – 4(k), then precisely T(k+1) = 2(k+2) H(k+1) – 4(k+1) holds true for all n>= 1.

5. The Fibonacci numbers are denoted by the recurrence relation: $F_0 = 0$, $F_1 = 1$; and for $n > 1$, $F_n = F_{n-1} + F_{n-2}$. Suppose the Euclidean algorithm for the positive integer pair $a > b$ takes n iterations. Prove by induction on n that $a >= F_{n+2}$ and $b >= F_{n+1}$. Show that $n <= 5 \log_{10} b$

- *Base Case:*
  For n=0, the value of the integers a & b will be equal with value of 1 and with zero iterations.
  Hence $a >= F_2$ & $b >= F_3 = 1$.
  For $n <= 5 \log_{10} b$, taking n=0 will result in $0 <= 5\log_{10} b$ which is true

  Inductive Hypothesis:
  Assuming for k>=n; $a >= F_{k+2}$ and $b >= F_{k+1}$ and $k <= 5 \log_{10} b$ holds true

  Inductive Step:
  To prove for n= k+1, the values $a >= F_{k+3}$ and $b >= F_{k+2}$ and $k+1 <= 5 \log b$.

  In every iteration of Euclidean algorithm, a will be replaced by b and b will be replaced by the value of r(reminder), so *r* is a mod b.

  With the Fibonacci recurrence, we can see that,
  $F_{k+3} = F_{k+2} + F_{k+1}$
  Since we know that, $a >= F_{k+2}$ and $b >= F_{k+1}$. Hence,
  $a >= F_{k+2} + F_{k+1}$
  Using the relation, we have $a >= F_{k+3} = F_{k+2} + F_{k+1}$
  During our next iteration, *a* will be $>= F_{k+3}$ and similarly *b* will be $>= F_{k+2}$

  Hence by induction it is proved that k>=0, a>= F(n+2) and b >= F(n+1)

  Now to prove for, $k+1 <= 5 \log b$,
  From inductive hypothesis we are aware that $k <= 5 \log b$, so to obtain k+1,
  We shall add 1 on both sides of the equation, $k+1 <= 5 \log b + 1$

  To prove the above step, the equation can also be written as shown below,
  $5 \log b <= 5 \log b + 1$

  This proves the inequality since the value of RHS is greater than the value on the left hand side, therefore $k+1 <= 5 \log b + 1$ mathematical induction holds true.

  Hence, we have proven that that $a >= F_{n+2}$,$b >= F_{n+1}$ and $n <= 5 \log_{10} b$

6. Given a sequence of n integers, deciding whether all integers in the sequence are unique is known as Element Uniqueness. Suppose any algorithm for Element Uniqueness requires $\Omega(n \log n)$ time. Show that computing the closest pair of n points in a Euclidean plane requires $\Omega (n \log n)$

- We shall prove this assumption with the help of reduction argument. Assuming that there exists an algorithm to find the closest pair of n points in a Euclidian plane in O(n log n) time. Consider a line on the Euclidean plane, with n unique points. Then, find the closest pair of points on this line using the assumed algorithm. After calculating the closest pair of points, if the distance between the closest pair of points is greater than 0 then all elements are unique otherwise the elements are not unique, that is they have the same coordinates, this entire procedure will take O(n log n) time for finding Element Uniqueness. Since, we made an assumption that the closest pair can be got in O(n log n) time then Elemental Uniqueness

can also be solved in O(n log n) time. However, it is given that Elemental Uniqueness takes Ω(n log n) amount of time, leading to contradiction. Therefore, initial assumption is incorrect. Consequently, it is proved that computing the closest pair of points in Euclidean plane also takes Ω(n log n) time.