Kevin San Gabriel
301342241
CMPT 307 Assignment 2

2.1) $a = 10011011 = 1001 \cdot 2^4 + 1011$
$b = 10111010 = 1011 \cdot 2^4 + 1010$

$a_L = 1001$
$a_R = 1011$
$b_L = 1011$
$b_R = 1010$

$a_L + a_R = 1001 + 1011 = 10100$
$b_L + b_R = 1011 + 1010 = 10101$

$(a_L + a_R)(b_L + b_R) = (10100)(10101) = 110100100$

$(a_L)(b_L) = (1001)(1011) = 1100011$
$(b_R)(a_R) = (1010)(1011) = 1101110$

$\ddot{a}_L b_R + a_R b_L = (a_L + a_R)(b_L + b_R) - a_L b_L - a_R b_R$

$= 110100100 - 1100011 - 1101110 = 11010011$

$ab = 2^8 \cdot 1100011 + 2^4 \cdot 11010011 + 1101110$

$= 111000010011110$

2.2 We use induction to prove this

Base: n is a positive integer and b is a base, let $n=1$ and base $= b$. Then we have that there exists a power of b in $[1, b]$

IH: Assume for $n=k$ and base b, there exists a power of b that lies in $[k, bk]$

IS: Prove statement holds for $n=k+1$

Let b^p be a power of b. We have that $b^p > k$ by our IH and $b^p \geq (k+1)$...

Also, by our IH, $b^p = k$. Multiplying both sides by k we get $(b)(b^p) = bk$

~~then~~ $bk < b(k+1)$ so

$bb^p < b(k+1)$
$\Rightarrow b^{p+1} < b(k+1)$
$\Rightarrow (k+1) \leq b^p \leq b^{p+1} < b(k+1)$
$\Rightarrow (k+1) \leq b^p < b(k+1)$.

so, for any pos. integer $k+1$, there is a power of b, b^p, which is in the range $[(k+1), b(k+1)]$

2.4) Algorithm A:

Can be expressed as

$T(n) = 5T(\frac{n}{2}) + O(n)$

Recurrence relation is $T(n) = aT(\frac{n}{b}) + O(n^c)$,

a is 5, b is 2, c is 1.

By masters theorem, $T(n) = O(n^{\log_b a})$

Checking if $c < \log_b a$

$1 < \log_2 5$
$1 < 2.32$

So $T(n) = O(n^{\log_b a})$
$= O(n^{\log_2 5})$
$= O(n^{2.32})$

Algorithm B: Can be expressed as

$T(n) = 2T(n-1) + O(1)$

Letting $O(n) \leq cn$,

$T(1) = 2T(0) \ldots$
$T(2) = 2T(2-1)$
$= 2T(1)$

$$T(3) = 2 \cdot T(3-1)$$
$$= 2T(2)$$

so $T(n) = c \sum_{i=0}^{n-1} 2^i + 2^n T(0)$

Which is $O(2^n)$

$$T(n) = O(2^n)$$

Algorithm C: Can be expressed as

$$T(n) = 9T\left(\frac{n}{3}\right) + O(n^2)$$

$$T(n) = aT\left(\frac{n}{b}\right) + O(n^c)$$

Check: $c = \log_b a$

$$2 = \log_3 9$$

$$2 = 2 \checkmark$$

$$T(n) = O(n^2 \log n)$$

2.5 a) $T(n) = 2T(\frac{n}{3}) + 1$

$a = 2 \quad b = 3 \quad c = 0$

$\log_b a = \log_3 2 = 0.63$

$c < \log_b a$ so $T(n) = \Theta(n^{\log_3 2})$

b) $T(n) = 5T(\frac{n}{4}) + n$

$a = 5, b = 4, c = 1$

$\log_b a > c$ so $T(n) = \Theta(n^{\log_4 5})$

c) $T(n) = 7T(\frac{n}{7}) + n$

$a = 7, b = 7, c = 1$

$\log_b a = \log_7 7 = 1$

$c = \log_b a$ so $T(n) = \Theta(n \log n)$

d) $T(n) = 9T(\frac{n}{3}) + n^2$

$a = 9, b = 3, c = 2$

$c = \log_b a$ so $T(n) = \Theta(n^2 \log n)$

e) $T(n) = 8T\left(\frac{n}{2}\right) + n^3$

$a = 8,\ b = 2,\ c = 3$

$\log_b a = \log_2 8 = 3 = c$

so $T(n) = \Theta(n^3 \log n)$

f) $T(n) = 49\,T\left(\frac{n}{25}\right) + n^{\frac{3}{2}} \log n$

$a = 49,\ b = 25,\ c = \frac{3}{2}$

~~so T(n)~~ $\log_b a = \log_{25} 49 = 1.209\ldots$

so $c > \log_b a$ so $T(n) = \Theta(n^{1.209\ldots})$

k) $T(n) = T(\sqrt{n}) + 1$
$\quad = T(n^{\frac{1}{2}}) + 1$

$T(n^{\frac{1}{2}}) = T(n^{\frac{1}{2}})^{\frac{1}{2}} + 1$
$\quad = T(n^{\frac{1}{4}}) + 1$

$T(n^{\frac{1}{4}}) = T(n^{\frac{1}{4}})^{\frac{1}{2}} + 1$
$\quad = T(n^{\frac{1}{8}}) + 1$

~~so T(n) = T(n)~~

so $T(n) = T(n^{\frac{1}{2^k}}) + k$

k is $\Theta(\log\log n)$ so $T(n) = \Theta(\log\log n)$

2.11) $X = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$ $Y = \begin{bmatrix} E & F \\ G & H \end{bmatrix}$

the product matrix Z, is in the form

$$Z_{ij} = \sum_{k=1}^{n} X_{ik} Y_{kj} \quad \text{where } 1 \le i, j < n$$

$$Z_{ij} = (XY)_{ij}$$

$$= \sum_{k=1}^{n} X_{ik} Y_{kj}$$

$$= \sum_{k=1}^{\frac{n}{2}} X_{ik} Y_{kJ} + \sum_{k=\frac{n}{2}+1}^{n} X_{ik} Y_{kj}$$

$$= \sum_{k=1}^{\frac{n}{2}} A_{ik} E_{kj} + \sum_{k=1}^{\frac{n}{2}} B_{ik} G_{kj}$$

$$= [AE + BG]_{ij} \quad \text{for } i,j \le \frac{n}{2}$$

$$Z_{ij} = (XY)_{ij}$$

$$= \sum_{k=1}^{n} X_{ik} Y_{kj}$$

$$= \sum_{k=1}^{\frac{n}{2}} A_{ik} F_{kj} + \sum_{k=1}^{\frac{n}{2}} B_{ik} H_{kj} = [AF + BH]_{ij}$$

for $i \le \frac{n}{2}$ and $\frac{n}{2} < j \le n$

For $\frac{n}{2} < i \leq n$ and $j \leq \frac{n}{2}$,

$$Z_{ij} = (XY)_{ij}$$
$$= [CE + DG]_{ij}$$

For $\frac{n}{2} < i, j \leq n$:

$$Z_{ij} = (XY)_{ij}$$
$$= [CF + DH]_{ij}$$

$$Z = \begin{bmatrix} Z_{ij} \text{ where } i,j \leq \frac{n}{2} & \text{where } i \leq \frac{n}{2} \text{ and } \frac{n}{2} < j \leq n \\ \text{where } \frac{n}{2} < i \leq n \text{ and } j \leq \frac{n}{2} & \text{whee } \frac{n}{2} < i,j \leq n \end{bmatrix}$$

$$= \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

2.12) $T(n) = 2T(\frac{n}{2}) + O(1)$

$T(1) = 0$ and $T(2) = 1$

Using master theorem

$a = 2,\ b = 2\ \ c = 0$

$c < \log_b a = \log_2 2$ so

$T(n) = \Theta(n^{\log_2 2}) = \Theta(n)$

lines printed by program $= \Theta(n)$

2.14) We use mergesort to remove duplicates because it runs in $O(n\log n)$

Firstly, sort array with mergesort

Secondly, create second temp array and move all values from first array to second taking $O(n)$ time.

then, in the first original array, we check if for all k values in the array

if $(k-1 == k)$
copy value from first array to second.

we do this after moving the first value in the original array to ~~a~~ temp array to avoid index out of bounds errors. Then, we return second array.

This takes $O(n)$ time so in total, removing duplicates takes $O(n \log n)$

2.19) When merging k arrays one by one, when each array has 1 to n elements:

time taken to merge all arrays is

$$= O(2n + 3n + \ldots + kn)$$
$$= O(k^2 n)$$

because merging two arrays of size n is 2n time. Merging that array with another is 3n time and so on.

b) If each array is already sorted, if we merge each array repeatedly we get $O(kn)$ work to merge k arrays ~~and~~ of size n into k/2 arrays of size 2n and then we keep merging to do $O(kn)$ work $O(\log k)$ time until we have a single array. So, running time $= O(k \log k n)$

2.27 a) $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$

$$A \times A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$= \begin{bmatrix} (a \cdot a) + (b \cdot c) & b \cdot (a + d) \\ c \cdot (a + d) & (b \cdot c) + (d \cdot d) \end{bmatrix}$$

thus, there are 5 multiplications to get a 2x2 matrix

b) The recursion cannot happen because the subproblems ~~of multiply~~ are not squaring matrices, rather they are just 5 multiplications. so

$$T = 5T(n/2)$$