CSE 321 - HW2

1-)

a-) $T(n) = 3T(n-1) - 2T(n-2)$

$a^2 = 3a - 2$

$a^2 - 3a + 2 = 0$

$(a-2)(a-1)$

$a^+ = 2 \quad a^- = 1$

$f(n) = c_1 (a^+)^n + c_2 (a^-)^n$

$f(n) = c_1 2^n + c_2 \qquad T(1) = 1 \quad T(2) = 2$

$f(1) = c_1 2 + c_2 = 1$

$f(2) = c_1 4 + c_2 = 2 \qquad 2c_1 = 1 \quad c_2 = 0$

$c_1 = 1/2$

$\boxed{f(n) = \frac{2^n}{2}} \quad T(n) = O(2^n)$

b-) $T(n) = T(n/2) + 1$

=> Master theorem

$T(n) = a\,T(n/b) + f(n)$

$a = 1$

$b = 2 \qquad f(n) = 1$

this is case two

$T(n) = \Theta(\log n)$

c-) $T(n) = 4T(n-1) + 4T(n-2) + 3n$

$a^2 = 4a + 4 + \frac{3n}{a^2}$

$a^2 - 4a - 4 = \frac{3n}{a^2}$

$(a-2)^2 = 0$

$a^+ = a^- = 2$

$f(n) = c_1 (a^+)^n + c_2 n (a^-)^n$

$f(n) = 2^n (c_1 + c_2 n)$

d-) $T(n) = 4T(n/2) + n^2$

-> Master Theorm

$T(n) = aT(n/b) + f(n)$

$a = 4 \qquad f(n) = n^2 = \in \Theta(n^d) \qquad d = 2$

$b = 2$

case 2:

$\Theta(n^d \log n) = \Theta(n^2 \log n)$

e-) $T(n) = 2T(n/2) + O(n^2) \qquad a = 2 \quad b = 2 \quad d = 1$

$T(n) = 2T(n/2) + cn$

2- $T(n) = 2[2T(\frac{n}{4}) + c(\frac{n}{2})] + cn$

simplfy

$T(n) = 2^2 T(\frac{n}{4}) + 2c(\frac{n}{2}) + cn$

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

4- $\frac{n}{2^k} = 1 \qquad \log_2 n = k$

5- $T(n) = n\,T(1) + O(n^2)$

f-) $T(n) = T(n/2) + T(n/4) + n$

$\Theta(n) = \Theta(\frac{n}{2}) + \Theta(\frac{n}{4}) + n$

$\Theta(n) = 2\Theta(n) + n$

$0 = \Theta(n) + n$ => That is impossible

so

$T(n) = \Theta(n \log n)$

$\Theta(n\log n) = \Theta(\frac{n}{2}\log\frac{n}{2}) + \Theta(\frac{n}{4}\log\frac{n}{4}) + n$

simplfy

$\Theta(n\log n) = \Theta(n\log n) + \Theta(\frac{n}{2}\log\frac{n}{2}) + n$

$\frac{1}{2}\Theta(n\log n) = n$

$\Theta(n\log n) = 2n$

$T(n) = \Theta(n\log n)$

g-) $T(n) = T(n/2) + n$

=> master

$a = 1 \quad b = 2 \qquad f(n) = n \qquad d = 1$

=> case 1

$\Theta(n^d) = \Theta(n)$

h-) $T(n) = 2T(\sqrt{n}) + 1$

$T(\sqrt{n}) = 2T(n^{1/4}) + 1$

$T(n) = 2(2T(n^{1/4}) + 1) + 1$

$T(n) = 2^2 T(n^{1/4}) + 2 + 1$

$T(n) = 2^k T(n^{1/2^k}) + \sum_{i=0}^{k} 2^i$

$\log_2 n^{\frac{1}{2^k}} = 1$

$2^k = \sqrt{n}$

$k = \log_2 \sqrt{n} = \frac{1}{2}\log(n)$

$T(n) = \underbrace{2^{\frac{1}{2}\log(n)}}_{\text{cons}} \underbrace{T(1)}_{\text{cons}} + \sum_{i=0}^{\frac{1}{2}\log_2 n - 1} 2^i \Rightarrow \Theta(\sqrt{n})$

Q2-)

→ BST = root

```
T(n) def is_balanced(root):

        if root is None:
            return True

O(n)  [ left_h = height_of_tree(root.left)
O(n)    right_h = height_of_tree(root.right)
                                                    T(n/2)                        T(n/2)
2T(n/2) if (abs(left_h - right_h) <= 1 and is_balanced(root.left) and is_balanced(
                                                                                  root.ri
            return True

        return False
```

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

Master Theorem $\Rightarrow n^{\log_2 2} = n \Rightarrow$ ... $T(n) = O(n \log n)$

```
T(n) def height_of_tree(root):

1          if root is None:
               return 0

T(n/2)     left_h = height_of_tree(root.left)
T(n/2)     left_r = height_of_tree(root.right)

1          if left_h > right_h:

                return left_h +1

           return right_h +1
```

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

Master T $\Rightarrow n^{\log_2 2} \Rightarrow O(n)$

Q3-)

a-) $T(n) = 5T(\frac{n}{2}) + O(n^3)$

$\rightarrow$ Master Theorem
$a=5 \quad b=2 \quad d=3$

$\Theta(n^3)$

Master Theorem

$\Theta(n^d)$; if $a < b^d$

$\Theta(n^d \log n)$ if $a = b^d$

$\Theta(n^{\log_b a})$; if $a > b^d$

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

$T(n) = 2T(n-2)$ $\rightarrow$ I will add $O(n)$

$T(n-2) = 2T(n-4)$ $\quad T(n-4) = 2T(n-6)$

$T(n) = 2[2T(n-4)] = 2[2[2T(n-6)]]$

$T(n) = 2^2 T(n-2k)$

$T(n) = 2^k T(n-2k)$ $\quad n = 2k$

$T(n) = 2^{n/2}$ $\quad k = \frac{n}{2}$

$= 2^n \in \Theta(2^n)$ $\quad O(n) \rightarrow$ it is to small so doesn't affect

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

$\rightarrow$ Master
$a=3 \quad b=2 \quad d=2$
$a < b^d$

$\Theta(n^2)$

$\rightarrow$ The worst is B and the best is C so I chose C bcs of running time.

Q4-)

Hopcroft-Karp Algorithm:

1-) Initialization
$\rightarrow$ Start with an empty matching.
$\rightarrow$ Initialize distance labels for vertices in sets A and B.
$\rightarrow$ Use BFS to find the shortest augmenting paths.

2-) Augmenting Paths
$\rightarrow$ While there exists an augmenting path:
- Use DFS to find an augmenting path
- Update the matching by augmenting along the path.

3-) Repeat steps 1-2 until no more augmenting paths can be found.

Worst $O(\sqrt{V}E)$

Best $O(E)$

Avg $O(\sqrt{V}E)$

Q-5)

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

a) $T(n)$

$n \quad T(n/2) \quad T(n/2)$

n

$n/2 \quad n/2$

$n/4 \quad n/4 \quad n/4 \quad n/4$

$k = h$

$\frac{n}{2^k} = 1$

$2^k = n$

$k = \log n$

$nk = (\log n)^{*} n \to O(n \log n)$

b) $T(n) = 2T(n/2) + n$

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

$\to T(n) = 2\left(2T\left(\frac{n}{2^2}\right) + \frac{n}{2}\right) + n$

$T(n) = 2^2 T\left(\frac{n}{2^2}\right) + \overbrace{n + n}^{2n}$

$T(n) = 2^k T\left(\frac{n}{2^k}\right) + kn$

$\frac{n}{2^k} = 1$

$n = 2^k$

$\boxed{\log n = k}$

$T(n) = 2^{\log n} T\left(\underbrace{\frac{n}{2^{\log n}}}_{1}\right) + \log n^{*} n$

$= 2^{\log n} T(1) + \log n^{*} n$

$= n T(1) + n \log n$

$T(n) \in O(n \log n)$

$T(n) \in O(n + n \log n)$