Emre Oytun
2001040040 99
E. Oytun

1.

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

$x^2 = 3x - 2$

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

$=2$ $\quad =1$

$x_1 = 2, \quad x_2 = 1$

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

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

$T(2) = 2 = 4c_1 + c_2$ $\quad -$

$1 = 2c_1$

$c_1 = 1/2$

$c_2 = 0$

$\Rightarrow \boxed{T(n) = \frac{1}{2} 2^n}$

Since, $\lim_{n\to\infty} \frac{T(n)}{2^n} = 1$, $\boxed{T(n) \in \Theta(2^n)}$

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

$T(1) = 1$
$T(2) = 2$
$T(4) = 3$
$T(8) = 4$

The guess: $T(n) = \log n + 1$

i) Try with base case:

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

ii) Proof by induction:

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

$\log(n+1) + \cancel{1} \stackrel{?}{=} \log\left(\frac{n+1}{2}\right) + 1 + \cancel{1}$

$\log(n+1) = \log(n+1) - \underbrace{\log 2}_{1} + 1$

$\log(n+1) = \log(n+1)$ ✓

So, $\boxed{T(n) = \log n + 1}$ and $\boxed{T(n) \in \Theta(\log n)}$

Since $\lim_{n\to\infty} \frac{T(n)}{\log n} = 1$.

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

$a_n^g = a_n^h + a_n^p$

a_n^h: $\quad x^2 = 4x - 4$

$x^2 - 4x + 4 = 0$

$(x-2)^2 = 0$

$\boxed{x_1 = x_2 = 2}$

$\boxed{a_n^h = c_1 2^n + c_2 n 2^n}$

a_n^p: $3n$ is in form $T(n) = An + B$: $\quad (a_n^p)$

$An + B = 4(A(n-1) + B) - 4(A(n-2) + B) + 3n$

$An + B = (4A - \cancel{4A})n - 4A + \cancel{4B} + 8A - \cancel{4B} + 3n$

$An + B = 3n + 4A$

$A = 3$
$B = 12$ $\Rightarrow \boxed{a_n^p = 3n + 12}$

$a_n^3 = c_1 2^n + c_2 n 2^n + 3n + 12$

$T(n) = \underbrace{c_1 2^n}_{\Theta(2^n)} + \underbrace{c_2 n 2^n}_{\Theta(n2^n)} + \underbrace{3n+12}_{\Theta(n)} \Longrightarrow$ **$T_1(n) \in \Theta(n 2^n)$**

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

$= 4^2 T(n/4) + 4 \cdot (\frac{n}{2})^2 + n^2$

$= 4^3 T(n/8) + 4^2 (\frac{n}{4})^2 + 4 (\frac{n}{2})^2 + n^2$

$\vdots$

$= 4^k T(n/2^k) + 4^{k-1} (\frac{n}{2^{k-1}})^2 + \cdots + n^2$

$= 4^k T(n/2^k) + \left(\sum_{i=1}^{k-1} \underbrace{\frac{4^i}{(2^i)^2}}_{1} \cdot n^2 \right) + n^2 = 4^k T(n/2^k) + (k-1)n^2 + n^2$

$= 4^k T(n/2^k) + kn^2$

Let $T(1) = c$: $k = \log_2 n$

$T(n) = \underbrace{n^2 \cdot c}_{\Theta(n^2)} + \underbrace{\log_2 n \cdot n^2}_{\Theta(n^2 \log n)}$

Thus, **$T(n) \in \Theta(n^2 \log n)$**

e) $T(n) = 2T(n/2) + O(n)$ $\quad (T(n/2) = 2T(n/4) + O(n/2))$

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

$= 2^3 T(n/2^3) + \underbrace{2^2 O(n/2^2)}_{O(n)} + \underbrace{2 O(n/2)}_{O(n)} + O(n)$

$\vdots$

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

Let $T(1) = c$: $n = 2^k$, $k = \log_2 n$ $\Rightarrow$ **$T(n) = nc + \log_2 n \, O(n)$**

By Masters Theorem:

$a = 2$, $b = 2$, $f(n) \in O(n)$, $d = 1$

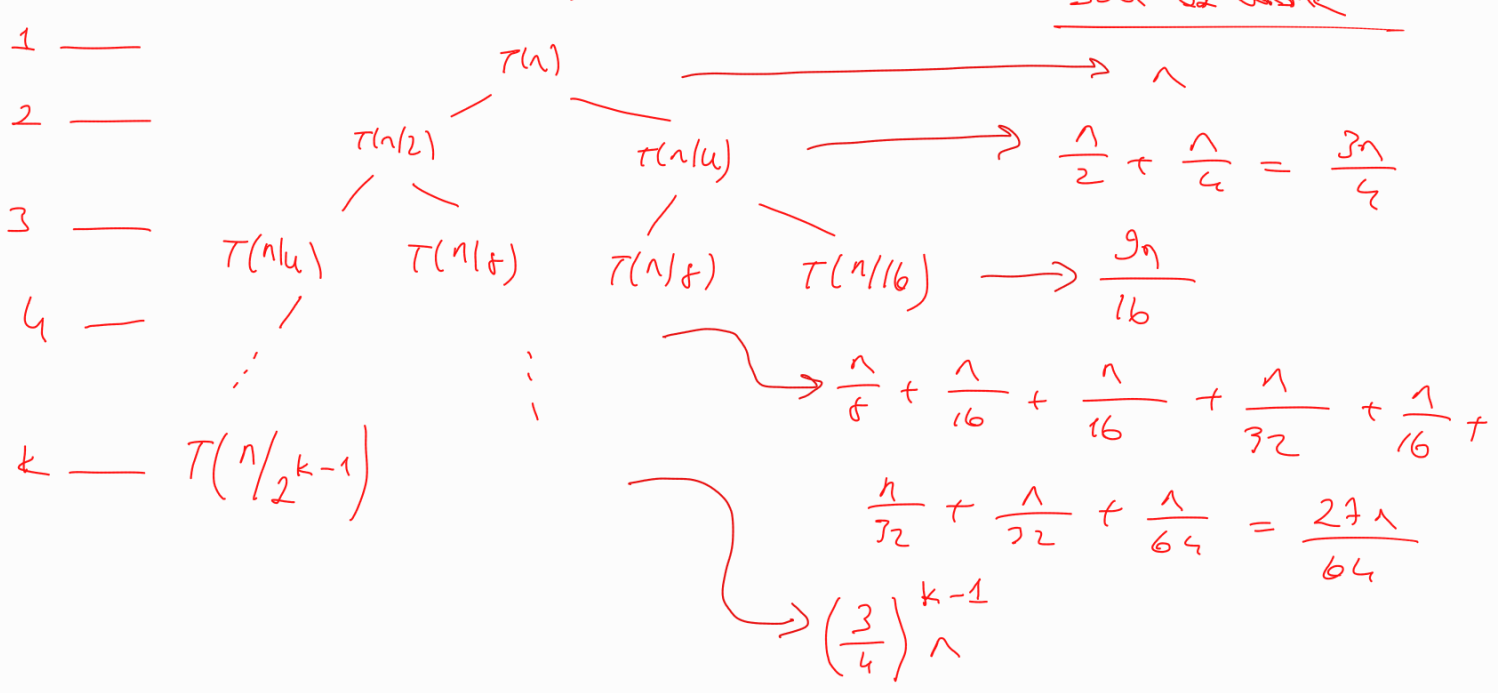
$\log_b a = \log_2 2 = 1$

$n^{\log_b a} = n^1 \in O(n)$

$f(n) \in O(n^{\log_b a}) \in O(n)$ (Their growth rate are same)

$\Rightarrow$ **$T(n) \in \Theta(n \log n)$**

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



Let $T(1) = 1$:

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

$$2^{k-1} = n$$

$$k-1 = \log_2 n \Rightarrow \boxed{k = \log_2 n + 1}$$

$$\sum_{i=1}^{k} \left(\frac{3}{4}\right)^{i-1} n$$ } Sum of the work at level k.

$$= n \sum_{i=1-1}^{k-1} \left(\frac{3}{4}\right)^{i+1-1} = n \sum_{i=0}^{k-1} \left(\frac{3}{4}\right)^{i}$$

$$= n \frac{1-\left(\frac{3}{4}\right)^k}{1-\frac{3}{4}}$$

$$= 4n\left(1-\left(\frac{3}{4}\right)^k\right) = 4n - 4n\left(\frac{3}{4}\right)^k$$

Assume $T(1) = c$.

Worst Case: It goes through the longest path which is the leftmost path.

The last element at level k: $T\left(n/2^{k-1}\right)$

$$\frac{n}{2^{k-1}} = 1 \Rightarrow \boxed{n = 2^{k-1}} \quad \boxed{k = \log_2 n + 1}$$

$$W(n) = 4n - 4n\left(\frac{3}{4}\right)^{\log_2 n + 1}$$

$$\boxed{W(n) \in O(n)}$$

Best Case: It goes through the shortest path which is the rightmost part.

The last element at level k: $T\left(n/4^{k-1}\right)$

$$\frac{n}{4^{k-1}} = 1 \Rightarrow \boxed{n = 4^{k-1}} \quad \boxed{k = \log_4 n + 1}$$

$$B(n) = 4n - 4n\left(\frac{3}{4}\right)^{\log_4 n + 1}$$

$$\boxed{B(n) \in O(n)}$$

Since $B(n) \in O(n)$ and $W(n) \in O(n)$

$$\Leftrightarrow$$

$$\boxed{T(n) \in \Theta(n)}$$

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

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

$\vdots$

$= T(n/2^k) + \left(n + \frac{n}{2} + \cdots + \frac{n}{2^{k-1}}\right)$

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

Let $T(a) = c$: $\frac{n}{2^k} = a \Rightarrow k = \log_2(n/a)$

$T(n) = T(a) + n\left[\frac{1}{2^{k-1}} + \frac{1}{2^{k-2}} + \cdots + \frac{1}{2} + 1\right]$

$T(n) = c + n \sum_{i=0}^{\log_2(n/a)-1} \frac{1}{2^i}$

Since, $T(1) = 1$ is given (in terms):

$$T(n) = 1 + n \sum_{i=0}^{\log_2 n - 1} \frac{1}{2^i}$$

By Master's Theorem:

$a = 1$, $b = 2$, $f(n) = n$ $\Rightarrow d = \log_b a = \log_2 1 = 0$ $\Rightarrow$ Thus, $T(n) \in \Theta(n)$

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

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

$= 2^3 T(n^{1/8}) + 2^2 + 2 + 1$

$\vdots$

$= 2^k T(n^{1/2^k}) + \underbrace{2^{k-1} + 2^{k-2} + \cdots + 1}$

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

$T(4) = 3$:

$n^{1/2^k} = 4$

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

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

$T(n) = \frac{\log n}{2} T(4) + \frac{\log n}{2} - 1$

$T(n) = 2\log n - 1$ for n is perfect Square.

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

Since, $\lim_{n \to \infty} \frac{T(n)}{\log n} = 2$

2.

a) is_balanced_bst calls is_balanced_bst_helper and makes constant operations, so is_balanced_bst_helper needs to be analyzed.

* is_balanced_helper method:

→ Returns 0 if the BST is empty.

→ Calls itself recursively two times, one with left subtree, other with right subtree. And gets the results of these.

→ If one of the results are -1 meaning it's inbalanced, then returns -1.

→ Otherwise, calculates the |left_result − right_result|. If it's greater than 1, it returns -1.

→ Otherwise, it returns left_result + right_result + 1 as total number of nodes.

Best Case: Each time number of nodes are exactly divided by 2 and the result is combined in constant time c.

$B(n) = 2B(n/2) + c$ ⇒ By Master's Theorem: $a=2, b=2, f(n)=c \Big\rangle n^{\log_b a} > f(n)$

$$\boxed{B(n) \in \Theta(n)}$$

Worst Case: Tree is like a linked list. Each time number of nodes decreases only 1 and the result is combined in constant time c.

$W(n) = W(n-1) + \underbrace{W(0)}_{1} + c \Rightarrow W(n) = W(n-1) + c_2$

$= W(n-2) + 2c_2$

$= W(n-3) + 3c_2$

$\vdots$

$= W(n-k) + kc_2$

$n-k = 0$

$k = n$

$W(n) = W(0) + nc_2$

$= 1 + nc_2$

$$\boxed{W(n) \in \Theta(n)}$$

* Since $W(n) \in \Theta(n)$ and $B(n) \in \Theta(n)$, $\boxed{T(n) \in \Theta(n)}$

b) Height of the tree is calculated as the max. number of edges from root node to leaf node.

* height_of_tree calls height_of_tree_helper and makes constant operations, so height_of_tree_helper needs to be analyzed.

* height_of_tree_helper method:

→ If BST is empty returns -1.

→ Calls itself recursively 2 times for left and right subtree.

→ Returns 1 + left_height + right_height.

Best Case: Number of nodes is exactly divided by 2 into left and right subtrees.

$B(n) = 2B(n/2) + c \Rightarrow$ By Master's Theorem: $a=2,\ b=2,\ f(n)=c \Big\rangle n^{\log_b a} > f(n)$

$$B(n) \in \Theta(n)$$

Worst Case: Tree is like a linked list. Each time number of nodes decreases only 1 and the result is combined in constant time c.

$$W(n) = W(n-1) + \underbrace{W(0)}_{1} + c \Rightarrow W(n) = W(n-1) + c_2$$

$$= W(n-2) + 2c_2$$

$$= W(n-3) + 3c_2$$

$$\vdots$$

$$= W(n-k) + kc_2$$

$n-k=0 \Rightarrow k=n$

$W(n) = W(0) + nc_2$

$= 1 + nc_2$

$W(n) \in \Theta(n)$

* Since $W(n) \in \Theta(n)$ and $B(n) \in \Theta(n)$, $T(n) \in \Theta(n)$.

3.

a) $T(n) = 5\,T(n/2) + n^3$

By Master's Theorem:

$a = 5$, $b = 2$, $f(n) = n^3$

$\log_b a = \log_2 5 = 2,\ldots$

$3 > \log_2 5$

$f(n) = n^3 > n^{\log_2 5}$

So, $T(n) \in \Theta(n^3) \Rightarrow T(n) \in O(n^3)$

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

By Master's Theorem:

$a = 2$, $b = 2$ $\Rightarrow$ $d = \log_2 1 = 1$

$f(n) = n$

Since $a > 1$, $T(n) \in O(n\,2^{n/2})$

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

By Master's Theorem:

$a = 3$, $b = 2$, $f(n) \in O(n^2)$

$n^{\log_b a} = n^{\log_2 3} = n^{1,\ldots}$

$O(n^2) \supset O(n^{\log_b a + \varepsilon}) = O(n^{1,\ldots})$

$f(n) \in \Omega(n^{1,\ldots})$

So, $T(n) \in \Theta(n^2) \Rightarrow T(n) \in O(n^2)$

* I'd choose the algorithm in option "C". Because:

$n^2 < n^3 < n\,2^{n/2}$

$O(n^2) \subset O(n^3) \subset O(n\,2^{n/2})$

} Order of growth

$$\lim_{n\to\infty} \frac{n^2}{n^3} = \lim_{n\to\infty} \frac{1}{n} = 0$$

$$\lim_{n\to\infty} \frac{n^3}{n\,2^{n/2}} = \lim_{n\to\infty} \frac{n^2}{2^{n/2}} \overset{L}{=} \lim_{n\to\infty} \frac{2n}{2^{n/2}\cdot \frac{1}{2}\ln 2}$$

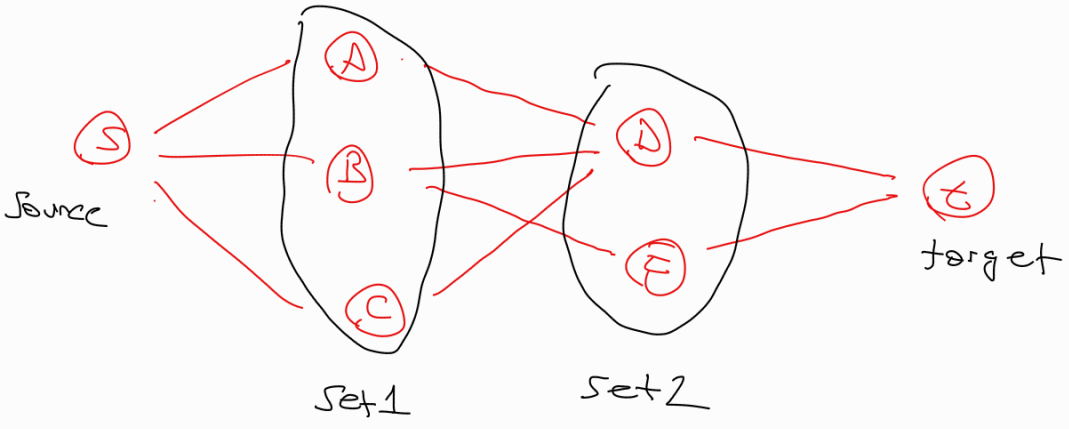
$$\overset{L'}{=} \lim_{n\to\infty} \frac{2}{2^{n/2}\left(\frac{1}{2}\right)^2 \ln^2 2} = 0 .$$

---

4) Pseudocode and algorithm explanation:

i) Graph is colored and vertexes are splitted into 2 sets. (coloring process)
ii) Source and target vertexes added.
iii) There are always an edge between source and a vertex in set1.
iv) There are always an edge between a vertex in set2 and target.
v) Each time, it tries to find a path from source to target by running BFS.
vi) If it's found, then it decrements the residual flow from the edges in this path, and increments the max flow by 1. At the end, returns to step 5.
vii) If it's not found, the algorithm terminates.

exp: An example of bipartite graph, source, target:



Coloring Process:

0 → blue
1 → red

i) It initializes all colors as −1 (uncolorful).
ii) It iterates through all vertexes. If the vertex is not colored, then it's colored and a BFS is executed starting from this node while coloring the nodes each time.

Coloring Process Analysis:

Since the BFS is executed with no pre-terminator condition, it always iterates through all vertices and the edges of these vertices.

Best-case: $B(n) = \Theta(V+E)$

Worst-case: $B(n) = \Theta(V+E)$

Average-case: $B(n) = \Theta(V+E)$

Source-Target Vertices Addition Analysis:

An edge between source and all vertices in set1, en " " target " " " " set2 is created.

Therefore, it always runs in $\Theta(V)$ time.

Best-case: $B(n) = \Theta(V)$

Worst-case: $B(n) = \Theta(V)$

Average-case: $B(n) = \Theta(V)$

Augmating Path Finding Analysis:

Best-case: It finds the augmenting path in at most 3-depth.

So, $\Theta(1)$.

Worst-case: It cannot find a path so it traverses all vertexes and edges meanwhile.

So, $\Theta(V+E)$

Decreasing Residual Flow Analysis:

Best-case; There is only one edge between two vertexes from set1 and set2 except source and target.

So, iterating this path takes $\Theta(1)$.

Worst-case: The path can include all vertexes, so it takes $\Theta(V)$ time.

$V = |V|$ (# of vertexes)

$f \rightarrow$ max flow

Overall Algorithm Analysis:

	Best	Worst	Average
Coloring	$\Theta(V+E)$	$\Theta(V+E)$	$\Theta(V+E)$
Adding S and T	$\Theta(V)$	$\Theta(V)$	$\Theta(V)$
Finding Augmenting Path	$\Theta(1).1$	$\Theta(V+E).f$	$\Theta(V+E).f$
Decreasing Residual Workflow	$\Theta(1).1$	$\Theta(V).f$	$\Theta(V).f$

(f: max. flow = max. cardinality)

As a result:

If we consider finding the sets in bipartite graph and adding source and target :

Best-case: $O(V+E)$
Worst-case: $O((V+E).f)$
Average-case: $O((V+E).f)$

If we only consider the algorithm after necessary properties are done :

Best-case: $O(1)$.
Worst-case: $O((V+E).f)$
Average-case: $O((V+E).f)$

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

$$\begin{cases} T(n) = 0, & \text{if } n \leq 1 \\ T(n) = 2T(n/2) + n, & \text{if } n > 1 \end{cases}$$

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

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

$= 2^3 T(n/8) + n + n + n$

$\vdots$

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

$\frac{n}{2^k} = 0 \Rightarrow k = \log_2 n$

$\rightarrow T(n) = nT(0) + \log_2 n \cdot n$

$T(n) = n \log_2 n$