Elifnur Kabalcı
1801042617

1)

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

$\alpha^2 = 3\alpha - 2$

$\alpha^2 - 3\alpha + 2 = 0$ $\quad (\alpha-2)(\alpha-1)$

$\alpha \quad -2$

$\alpha \quad -1$

$\alpha_1 = 2 \quad \alpha_2 = 1$

$x(n) = c_1 \cdot 2^n + c_2 \cdot 1^n$

$n=1 \quad c_1 \cdot 2 + c_2 = 1$
$n=2 \quad c_1 \cdot 4 + c_2 = 2$

$c_1 = 1/2$
$c_2 = 0$

$x(n) = 2^{n-1} \in \Theta(2^n)$

b) $T(n) = T(n/2) + 1$ $\quad T(1) = 1$, $T(2) = 2$

$n=2 \to T(2) = T(1) + 1$
$n=4 \to T(4) = T(2) + 1$
$n=8 \to T(8) = T(4) + 1$
$\vdots$
$T(n) = T(n/2) + 1$

$2^x = n$
$\log \quad \log$

$x = \log_2^n$ times

$\to T(n) = T(1) + \log_2^n$

$T(n) = 1 + \log_2^n \in \Theta(\log n)$

c) $T(n) = 4T(n-1) - 4T(n-2) + 3n$
$= 4(4T(n-2) - 4T(n-3) + 3(n-1)) - 4T(n-2) + 3n$
$= 4^2 \cdot T(n-2) - 4^2 T(n-3) + 15n - 12$
$= 4^3 \cdot T(n-3) - 4^3 T(n-4) + 15 \cdot 4(n-2) - 12$
$= 4^3 \cdot T(n-3) - 4^3 T(n-4) + 60n - 108$

$T(n) = 4^k \cdot T(n-k) - 4^k \cdot T(n-k-1) + 3k \cdot 4^{k-1} - 12(1 + \dots + k-1)$

$n = 4^k \to k = \log_4^n$

$T(n) = 4^{\log_4^n} \cdot T(1) - 4^{\log_4^n} T(0) + 3\log_4^n \cdot 4^{\log_4^n - 1} - 12(1 + \dots + \log_4^n - 1)$

$T(n) = n \cdot T(1) + n \cdot T(0) + 3n \cdot \log_4^n - 12 \quad \in \Theta(n\log n)$

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

$a=4$, $b=2$, $d=2$ $\to$ Master teorem

$a = b^d \Rightarrow x(n) = \Theta(n^2 \cdot \log n)$

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

$a=2$, $b=2$, $f(n) = O(n) \to$ master teorem

$T(n) = \Theta(n^{\log_b^a}) = \Theta(n^{\log_2^2}) \Rightarrow O(n)$

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

$= (T(n/4) + T(n/8) + n/2) + (T(n/8) + T(n/16) + n/4) + n$

$= T(n/4) + 2 \cdot T(n/8) + T(n/16) + n/2 + n/4 + n$

$\vdots$

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

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

$T(n) = T(1) + \log_2^n \left(\frac{n}{2^{\log_2^n}} + \cdots + 1\right)$

$\Rightarrow T(1) + \log_2^n \cdot O(\log n) \Rightarrow \Theta(\log^2 n)$

g) $T(n) = T(n/2) + n, \quad T(1) = 1, \quad T(2) = 3$

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

$T(n/4) = T(n/8) + n/4$

$\vdots$

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

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

$T(n) = T(1) + (2^1 + 2^2 + \cdots + n) \qquad 2^k = n$

$= T(1) + (2^1 + 2^2 + \cdots + 2^k)$

$= T(1) + \sum_{i=1}^{k} 2^i \Rightarrow T(1) + 2^{k+1} - 1$

$= 1 + \underbrace{2^k}_{n} \cdot 2 - 1$

$T(n) = 2n \rightarrow \in \Theta(n)$

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

$2n = \frac{2n}{2} + n$ ✓

h) $T(n) = 2\,T(\sqrt{n}) + 1, \quad T(1) = 1, \quad T(4) = 3$

$n = 2^m \rightarrow$ perfect squares.

$T(2^m) = 2 \cdot T(2^{m/2}) + 1$

$S(m) = T(2^m) \rightarrow$ Lets assume

$S(m) = 2\,S\left(\frac{m}{2}\right) + 1 \rightarrow$ master teorem

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

$T(2^m) = \Theta((\log 2^m)^1) = \Theta(m)$

$\Downarrow$

$T(n) = \Theta(\log_2^n) \rightarrow \in \Theta(\log n)$

2)

a) is_balanced: Reccurence method. We divide into 2 parts to tree in every node. So;

$$T(n) = T(n/2) + O(1) \rightarrow \in \Theta(\log n)$$

b) height_of_tree: This method also reccurence. Algorithm is very similar to a.

$$T(n) = T(n/2) + O(1) \rightarrow \in \Theta(\log n)$$

3)

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

→ master teorem. $a=5$, $b=2$ ⇒ $O(n^{\log_2 5})$

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

→ master teorem, $a=2$, $k=1$ ⇒ $O(n^3)$

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

→ master teorem, $a=3$, $k=2$ ⇒ $O(n^{\log_2 3})$

⇒ C has the min complexity, But B is more stable. n's power is a number. So B needs to be prefered choice.

4) Question's says definition of Hopcroft-Karp algorithm. It's a polynomial-time algorithm for finding a maximum cardinality matching in bipartite graphs.

algorithm:

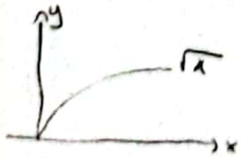
1. Initialize maximal maching M as empty.
2. While there exist an Augmenting Path P
   - Remove matching edges of P from M and add not-matching edges of P to M.
   - Increase size of M by 1 as P starts and ends with a free vertex
3. Return M.

→ n: # of vertices } in bipartite graph
  m: # of edges }

→ in worst case: It need to check every node in graph. It become $O(n \cdot m)$.
If we say $m=n$, $\in O(n^2)$

→ in best case, matching parts is very large. BFS explore small area so $\Omega(n+m)$
If we say $m=n$, $\in \Omega(2n) \rightarrow \in \Omega(n)$.

→ In average case, It depends on finding augmented path times. As we visit, fewer nodes will remain unvisited.



→

# of augmented path so $O(n \cdot \sqrt{m}) \rightarrow$ If $n=m$, $\in \Theta(n^{3/2})$

5)

```
foo(n):
  if n<=1;
    return 1
  else:
    for i in range(n):
      print("a");
  return foo(n/2) + foo(n/2);
```

→ parameter is n → # of print "a" in every calling.

→ 2 times call function itself with half of n.

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

in master teorem

$a=2$, $b=2$, $d=1$ $\Rightarrow \Theta(n^d \cdot \log n)$

$\Rightarrow \Theta(n \cdot \log n)$

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

$$= 2\left(2 \cdot T(n/4) + n/2\right) + n$$

$$= 2\left(2 \cdot \left(2\left(T(n/8) + n/4\right)\right) + n/2\right) + n$$

# of step $n = 2^x$

$x = \log_2^n$

$$\Rightarrow \underbrace{2^x}_{n} \cdot T(1) + \left[\underbrace{n + 2 \cdot \frac{n}{2} + 4 \cdot \frac{n}{4} \cdots}_{\log_2^n \text{ step}}\right]$$

$$\Rightarrow n \cdot \underbrace{T(1)}_{1} + \log_2^n$$

$$T(n) \Rightarrow n + \log_2^n$$

We have $\log_2^n$ step for calling this function. for every tour, running for loop n times. So we write "a" $n \cdot \log_2^n$ times.