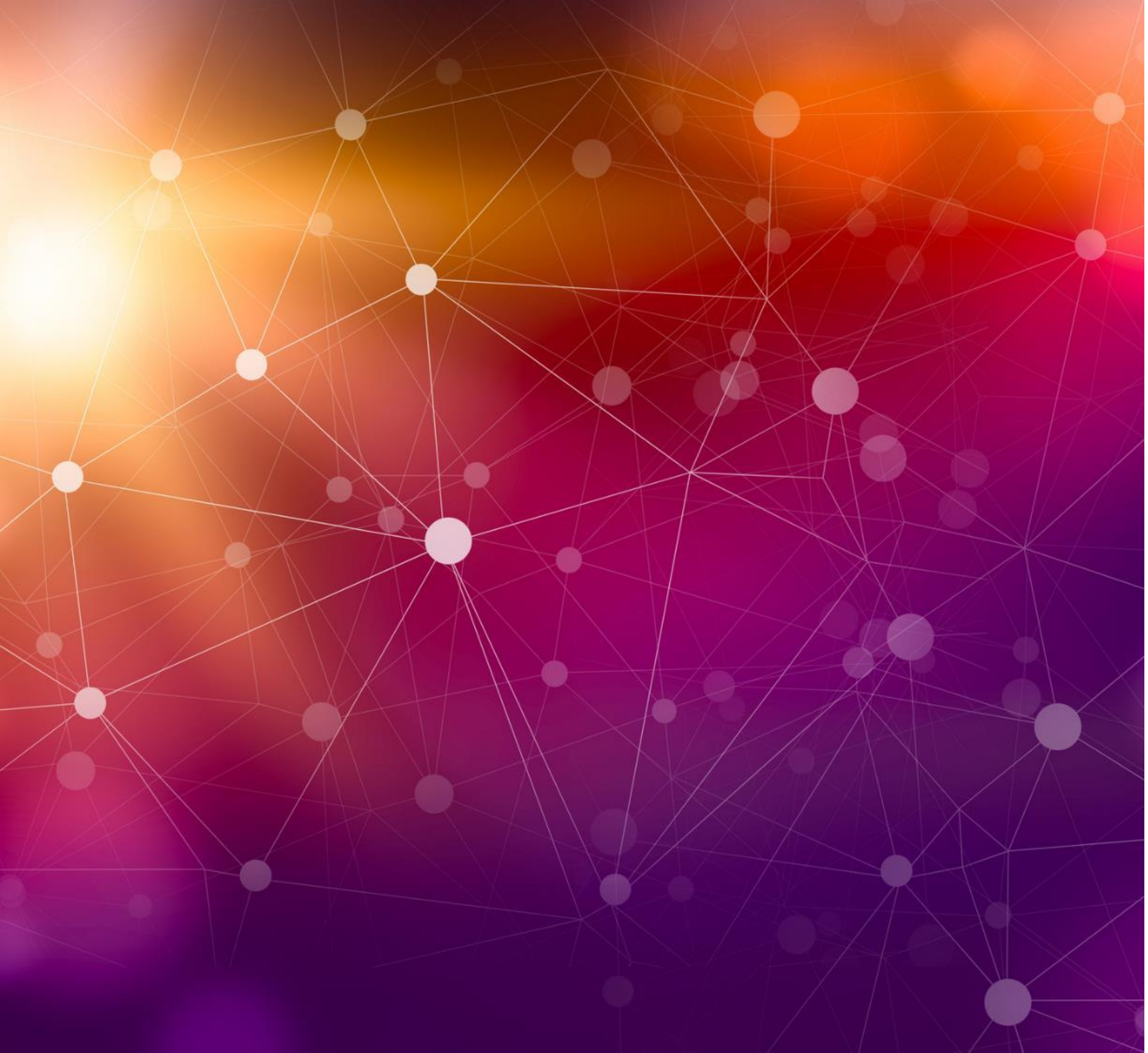

Foundations of Algorithm SCS1308

Dr. Dinuni Fernando PhD

Senior Lecturer

UCSC

Solving recurrence equations

- Techniques for solving recurrence equations:
  - *Recursion tree method*
  - *Substitution method*
  - *Iteration method*
  - *Master Theorem*
- We discuss these methods with examples.

Solve the following recurrence using Recursion Tree method

- $T(n) = T(n-1) + 1 : n > 0$
- $T(n) = 1 : n = 0$

Solving using Recursion Tree method

- T (n) = T (n-1) + 1 : n > 0
- T(n) = 1 : n = 0

- T(n) = T(n-1) +1

	depths	
1	0	O(1)
T(n-1)	1	O(1)
T(n-2)	2	O(1)
T(n-3)	3	O(1)
	⋮	⋮
		(n-1)

$$\underbrace{[0 \ldots\ldots (n-1)]}_{n} * O(1)$$

$\underline{\underline{O(n)}}$

Solving using Iteration method

- $T(n) = T(n-1) + 1 : n > 0$
- $T(n) = 1 : n = 0$

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

$$T(n) = T(n-1) + 1$$
$$= T(n-2) + 1 + 1$$
$$= T(n-2) + 2$$
$$\vdots \quad k$$
$$= T(n-k) + k$$

assume $n - k = 0$;

$$= T(0) + n = 1 + n = \Theta(n)$$

Solving using Iteration method

- Unroll the recurrence step by step.

$$
\begin{aligned}
T(n) &= T(n-1) + 1 \\
&= (T(n-2) + 1) + 1 = T(n-2) + 2 \\
&= T(n-3) + 3 \\
&\vdots \\
&= T(n-k) + k.
\end{aligned}
$$

- Choose $k = n$ so the argument becomes $T(0)$:
- $T(n) = T(0) + n = 1 + n.$
- So by iteration/unrolling the solution is $\boxed{T(n) = n + 1}$.

Solving using substitution method (induction proof)

- Prove by induction that $T(n) = n + 1$.
- **Base.** $n = 0$: $T(0) = 1$and formula gives $0 + 1 = 1$. True.
- **Inductive step.** Assume for some $m \geq 0$ that $T(m) = m + 1$ .Then for $n = m + 1$,

$$T(m+1) = T(m) + 1 = (m+1) + 1 = (m+1) + 1.$$

So $T(m+1) = (m+1) + 1$ .Thus by induction $T(n) = n + 1$ for all $n \geq 0$.

- Therefore O(n)

Masters Theorem

- Let T(n) be <u>a monotonically increasing</u> function that satisfies

  $$T(n) = a\,T(n/b) + f(n)$$
  $$T(1) = c$$

  where $a \geq 1$, $b \geq 2$, $c>0$. If $f(n)$ is $\Theta(n^d)$ where $d \geq 0$ then

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log n) & \text{If } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

When Master’s Theorem Does not Apply

- Master’s Theorem cannot be used in certain cases:

1. If f(n) is not polynomially related to $n^{log_b^a}$ $(\, if\ f(n) involves\ with\ irregular\ functions\ like\ logarithms\ or\ exponential\ terms)$
2. If the recurrence relation does not fit the required form.

Other methods such as recursion tree, substitution method can be used.

Master method examples

- Case 1:
  - $T(n) = 8T(n/4) + 5n^2$ for $n>1$, n is a power of 4
  - $T(1) = 3$

→ $a=8$, $b=4$, $d=2$

→ As $a < b^d$ (i.e., $8 < 4^2$), $T(n) = \Theta(n^2)$

Master method examples

- Case 2:
  - $T(n) = 8T(n/2) + 5n^3$ for $n>64$, n is a power of 2
  - $T(64) = 200$

→As $a = b^k$ (i.e., $8 = 2^3$), $T(n) = \Theta(n^3 \lg n)$

Master method examples

- Case 3:
  - T(n) = 9T(n/3) + 5n for n > 1, n is a power of 3
  - T(1) = 7

→a = 9, b = 3, d = 1

→Since a > b^d ,

$$T(n) = \Theta(n^{\log_3 9}) = \Theta(n^2)$$

Thank you