# ECE345 Tutorial 3

Winston (Yuntao) Wu

Electrical & Computer Engineering

UNIVERSITY OF TORONTO

# Outline

# Recurrence

Recurrence is usually used for divide-and-conquer type of questions.

$T(n) = \sum_{i=0}^{k} a_i T(g_i(n)) + f(n)$ for some $k$, $g_i(n) < n$.

Note: it is fine to have $\sum_i^k a_i g_i(n) \neq n$, we can have overlapping subproblems (e.g. DP), or some of the problem is not really helping us (e.g. binary search).

$\sum_{i=0}^{k} a_i T(g_i(n))$ is for divide, $a_i$ is the number of subproblems and $g_i(n)$ are the sizes of subproblems.

$f(n)$ is for conquer, *i.e.* How much work you need to do if you to solve a problem of size $n$?

# Basic examples

Find asymptotic expressions for the following $T(n)$:

e.g. $T(n) = T(n-1) + n$, $T(1) = 1$

Proof: $T(n) = T(n-1) + n = T(n-2) + (n-1) + n = \cdots = T(1) + 2 + 3 + 4 + \cdots + (n-1) + n$

$$T(n) = \sum_{i=1}^{n} = \frac{n(n+1)}{2},\ T(n) = \Theta(n^2)$$

e.g. $T(n) = T(n/2) + n$, $T(1) = 1$

Proof: $T(n) = T(n/2) + n = T(n/4) + n/2 + n = T(n/8) + n/4 + n/2 + n = \cdots =$
$T(1) + 2 + 4 + 8 + \cdots + (n/4) + (n/2) + n$

$$T(n) = \sum_{i=0}^{\log n} \frac{n}{2^i} = n\frac{1-(1/2)^{\log n + 1}}{1 - 1/2} = n(2 - 1/n) = 2n - 1,\ T(n) = \Theta(n)$$

# Basic examples

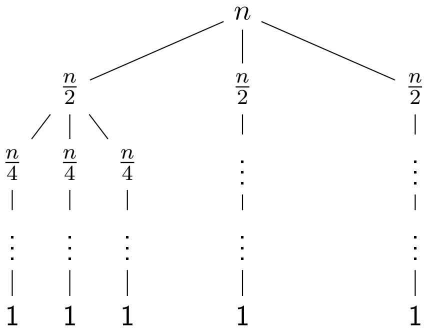
Find asymptotic expressions for the following $T(n)$:

e.g. $T(n) = 3T(n/2) + n$, $T(1) = 1$

Proof: $T(n) = 3T(n/2) + n = 3(3T(n/4) + n/2) + n = 3^2T(n/4) + 3(n/2) + n = \cdots = 3^kT(1) + 3^{k-1}(n/2^{k-1}) + \cdots + 3^2(n/2^2) + 3(n/2) + n = 3^kT(1) + \sum_{i=0}^{k-1}\left(\frac{3}{2}\right)^i n$

Assume $n = 2^k$, $k = \log n$ $T(n) = 3^{\log n} + \sum_{i=0}^{\log n - 1}\left(\frac{3}{2}\right)^i n = n^{\log 3} + n\frac{(3/2)^{\log n} - 1}{(3/2) - 1} = n^{\log 3} + 2n(n^{\log 3}/n - 1) = 3n^{\log 3} - 2n$,

$T(n) = \Theta(n^{\log 3})$



# Outline

# Master Theorem

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$, where $a \geq 1, b \geq 1$, $f(n)$ is asymptotically positive

- Case 1: $f(n) = \mathcal{O}(n^{\log_b a - \epsilon})$, $\epsilon > 0$. Then $T(n) = \Theta(n^{\log_b a})$ (cost of solving the sub-problems at each level increases by a certain factor, the last level dominates)
- Case 2: $f(n) = \Theta(n^{\log_b a})$. Then $T(n) = \Theta(n^{\log_b a} \log n)$ (cost to solve subproblem at each level is nearly equal)
- Case 3: $f(n) = \Omega(n^{\log_b a + \epsilon})$, $\epsilon > 0$ and $af\left(\frac{n}{b}\right) \leq cf(n)$ for some $c < 1$ and $n > n_0$ (regularity condition, always holds for polynomials). Then $T(n) = \Theta(f(n))$ (cost of solving the subproblems at each level decreases by a certain factor)

Method:

1. identify $a$ and $b$ and compute $\log_b a$
2. compare $n^{\log_b a}$ to $f(n)$ and decide which case applies
3. don't forget to check the regularity condition for case 3

# Examples

$T(n) = 7T\left(\frac{n}{2}\right) + n^2$ (Works for $n^2 \log n$)

Solution: $a = 7, b = 2$, $\log_2 7 \approx 2.8$, $f(n) = n^2 = \mathcal{O}(n^{\log_2 7 - \epsilon})$ for $\epsilon \in (0, \log_2 7 - 2)$

Case 1, $T(n) = \Theta(n^{\log_2 7})$

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

Solution: $a = 4, b = 2$, $\log_2 4 = 2$, $f(n) = n^{2.5} = \Omega(n^{2+\epsilon})$ for $\epsilon \in (0, 0.5]$

Case 3, Check regularity: $af\left(\frac{n}{b}\right) = 4\left(\frac{n}{2}\right)^{2.5} \le cn^{2.5}$. Choose $c \in \left[\frac{1}{\sqrt{2}}, 1\right)$

$T(n) = \Theta(f(n)) = \Theta(n^{2.5})$

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

Solution: let $n = 2^m$, $S(m) = T(2^m) = T(n)$

Then $T(2^m) = T(2^{\frac{m}{2}}) + 1 \Leftrightarrow S(m) = S\left(\frac{m}{2}\right) + 1$

$a = 1, b = 2$, $\log_2 1 = 0$, $f(m) = 1 = m^0 = \Theta(m^0)$

Case 2, $T(2^m) = S(m) = \Theta(\log m)$, $T(n) = \Theta(\log \log n)$

# Outline



UNIVERSITY OF TORONTO

# Substitution Method

Recall the definitions:

$T(n) = \mathcal{O}(g(n)) \Leftrightarrow \exists c, n_0 > 0$ s.t. $0 \leq T(n) \leq cg(n)$, $\forall n \geq n_0$.

$T(n) = \Omega(g(n)) \Leftrightarrow \exists c, n_0 > 0$ s.t. $0 \leq cg(n) \leq T(n)$, $\forall n \geq n_0$.

Method:

1. Guess the form of the solution: $T(n) = \mathcal{O}(g(n))$ (e.g. Use Recursion Tree method)
2. Induction Hypothesis: Assume $T(k) \leq cg(k)$, $\forall k < n$ (Strong induction)
3. Induction step: show $T(n) \leq cg(n)$
4. Usually don't care about base case, because we consider the long term behavior. You can choose the best-suited base case.

# Example

$T(n) = 2T\left(\frac{n}{2}\right) + n$, $T(1) = 1$

Guess: $T(n) = \mathcal{O}(n \log n)$
**Induction Hypothesis**: $T(k) \leq ck \log k, \forall k < n$
**Induction Step**: $T(n) = 2T\left(\frac{n}{2}\right) + n$
$\leq 2\left(c\frac{n}{2}\log\frac{n}{2}\right) + n$
$= cn\log n - cn\log 2 + n = cn\log n - n(c-1)$
$\leq cn\log n, \forall c \geq 1$

Guess: $T(n) = \Omega(n \log n)$
**Induction Hypothesis**: $T(k) \geq ck \log k, \forall k < n$
**Induction Step**: $T(n) = 2T\left(\frac{n}{2}\right) + n$
$\geq 2\left(c\frac{n}{2}\log\frac{n}{2}\right) + n$
$= cn\log n - cn\log 2 + n = cn\log n + n(1-c)$
$\geq cn\log n, \forall c \in (0, 1]$

The following is wrong:

Guess: $T(n) = \mathcal{O}(n)$
**Induction Hypothesis**: $T(k) \leq ck, \forall k < n$
**Induction Step**: $T(n) = 2T\left(\frac{n}{2}\right) + n$
$\leq 2\left(c\frac{n}{2}\right) + n$
$= cn + n \not\leq cn$

Guess: $T(n) = \Omega(n)$
**Induction Hypothesis**: $T(k) \geq ck, \forall k < n$
**Induction Step**: $T(n) = 2T\left(\frac{n}{2}\right) + n$
$\geq 2\left(c\frac{n}{2}\right) + n$
$= cn + n \geq cn, \forall c > 0$

# Example

$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{3}\right) + n$, $T(1) = 1$

Guess: $T(n) = \mathcal{O}(n \log n)$

**Induction Hypothesis**: $T(k) \leq ck \log k$, $\forall k < n$

**Induction Step**: $T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{3}\right) + n \leq c\frac{n}{2} \log \frac{n}{2} + c\frac{n}{3} \log \frac{n}{3} + n$

$= c\frac{n}{2} \log n - c\frac{n}{2} + c\frac{n}{3} \log n - c\frac{n}{3} \log 3 + n = cn\left(\frac{1}{2}\log n - \frac{1}{2} + \frac{1}{3}\log n - \frac{1}{3}\log 3\right) + n$

$= \frac{5}{6}cn \log n - cn\left(\frac{1}{2} + \frac{1}{3}\log 3\right) + n = \frac{5}{6}cn\log n - n\left(c\left(\frac{1}{2} + \frac{1}{3}\log 3\right) - 1\right)$

$\leq \frac{5}{6}cn \log n \leq cn \log n$, $\forall c \geq \frac{1}{\frac{1}{2} + \frac{1}{3}\log 3}$

For $\Omega$, we need $0 < c \leq \frac{1}{\frac{1}{2} + \frac{1}{3}\log 3}$

Note: this solution is not optimal, see the next example for the tightest bound.

# Outline

# Recursion Tree

Recursion Tree helps find a good working guess for substitution

- Longest path gives upper bound
- Shortest path gives lower bound

Usually, total cost=$h(\text{tree})\times$cost per level.
More precisely, total cost=$\sum_{i=0}^{h}$ cost at level $i$.

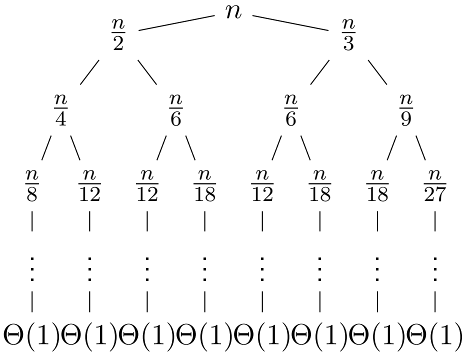
**Note:** If you use recursion tree to find the asymptotic bound, you MUST use substitution method to prove it.

# Example

$T(n) = T\left(\lceil \frac{n}{2} \rceil\right) + T\left(\lceil \frac{n}{3} \rceil\right) + n$, $T(1) = 1$

Longest path to a leaf: $\frac{n}{2^{k_1}} = 1$, $k_1 = \log n$, $h = \mathcal{O}(\log n)$
Shortest path to a leaf: $\frac{n}{3^{k_2}} = 1$, $k_2 = \log_3 n$, $h = \Omega(\log_3 n)$,
$\therefore h = \Theta(\log n)$
Sum of the work at each level $\leq n$
Total work $T(n) = \mathcal{O}(f(n)n) = \mathcal{O}(n \log n)$
More accurately, work at each level $i$ is $\leq \left(\frac{5}{6}\right)^i n$,

so the total work $T(n) \leq \sum_{i=0}^{\log n - 1} \left(\frac{5}{6}\right)^i n \leq \sum_{i=0}^{\infty} \left(\frac{5}{6}\right)^i n = 6n$, so $T(n) = \mathcal{O}(n)$ actually.
To prove it, use strong induction to get
$T(n) \leq c\frac{1}{2}n + c\frac{1}{3}n + n = (\frac{5}{6}c + 1)n < cn$ for $c > 6$



# Example

$T(n) = 2T\left(\lceil \frac{n}{2} \rceil\right) + T\left(\lceil \frac{n}{3} \rceil\right) + n$, $T(1) = 1$

Longest path to a leaf:

$\frac{n}{2^{k_1}} = 1$, $k_1 = \log n$,

$h = \mathcal{O}(\log n)$

Shortest path to a leaf:

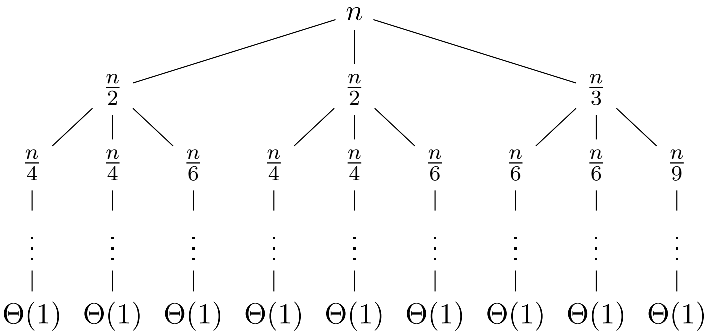
$\frac{n}{3^{k_2}} = 1$, $k_2 = \log_3 n$,

$h = \Omega(\log_3 n)$,

$\therefore h = \Theta(\log n)$

Sum of the work at each level $\geq n$, since $2T\left(\lceil \frac{n}{2} \rceil\right)$ always contributes exactly $n$ work

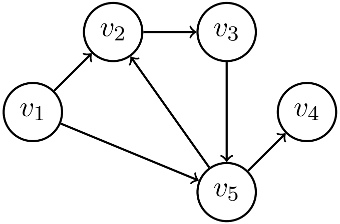
Total work $T(n) = \Omega(f(n)n) = \Omega(n \log n)$



# Outline

# Graphs

$G = (V, E)$, $V = \{\text{vertices}\}$, $E = \{\text{edges}\}$.
e.g. $V = \{v_1, v_2, v_3, v_4, v_5\}$
$E = \{(v_1, v_2), (v_1, v_5), (v_2, v_3), (v_3, v_5), (v_5, v_2), (v_5, v_4)\}$




## Representing Graphs

Adjacency List: $j \in L[i] \Leftrightarrow (v_i, v_j) \in E$
Time: $\mathcal{O}(V)$
Space: $\mathcal{O}(E) = \mathcal{O}(V^2)$ (worst case)
Good for sparse graph

| $L$ | |
|---|---|
| $v_1$ | $\{v_2, v_5\}$ |
| $v_2$ | $\{v_3\}$ |
| $v_3$ | $\{v_5\}$ |
| $v_4$ | $\{\}$ |
| $v_5$ | $\{v_2, v_4\}$ |

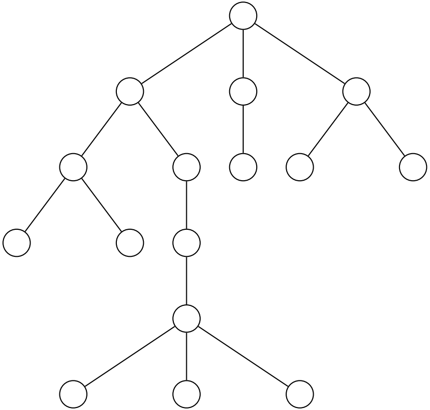
Adjacency Matrix: $M[i, j] =$ weight of edge $(v_i, v_j)$
Time: $\mathcal{O}(1)$
Space: $\mathcal{O}(V^2)$
Good for dense graph

| $M$ | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ |
|---|---|---|---|---|---|
| $v_1$ | 0 | 1 | 0 | 0 | 1 |
| $v_2$ | 0 | 0 | 1 | 0 | 0 |
| $v_3$ | 0 | 0 | 0 | 0 | 1 |
| $v_4$ | 0 | 0 | 0 | 0 | 0 |
| $v_5$ | 0 | 1 | 0 | 1 | 0 |

# Trees

A tree is a connected, undirected, acyclic graph
Sec B.5. Theorem B.2. (Properties of free trees):
Let $G = (V, E)$ be an undirected graph. The following statements are equivalent:

1. $G$ is a free tree
2. Any 2 vertices in $G$ are connected by a unique simple path
3. $G$ is connected, but if any edge is removed from $E$, the resulting graph is disconnected
4. $G$ is connected and $|E| = |V| - 1$
5. $G$ is acyclic and $|E| = |V| - 1$
6. $G$ is acyclic, but if any edge is added to $E$, the resulting graph contains a cycle



# Proof of Theorem B.2.

We can prove in a cycle of implications $1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 4 \Rightarrow 5 \Rightarrow 6 \Rightarrow 1$
$1 \Rightarrow 2$: $G$ is a free tree $\Rightarrow$ Any 2 vertices in $G$ are connected by a unique simple path
(Note on uniqueness proof: 1. prove at least one exists, 2. Assume 2 exist and reach a contradiction)
Given $G$ a free tree, i.e. connected, undirected, acyclic graph
$G$ free tree $\Rightarrow$ connected $\Rightarrow$ at least one simple path exits
Assume that a second path exists
Suppose 2 simple path $s \xrightarrow{p_1} t$ and $s \xrightarrow{p_2} t$
$\exists v$ in one of the path, but not the other. $s \xrightarrow{p_1} t \xrightarrow{p_2} s$ forms a cycle since $p_1 \neq p_2$. Contradiction.

$2 \Rightarrow 3$: Any 2 vertices in $G$ are connected by a unique simple path $\Rightarrow$ $G$ is connected, but if any edge is removed from $E$, the resulting graph is disconnected
Given any 2 vertices in $G$ are connected by a unique simple path
$G$ connected since $\exists$ path between all nodes since all vertices connected by definition of a simple path
removing one edge disconnects the nodes on the path and there is no other path to the nodes.

# Proof cont.

$3 \Rightarrow 4$: $G$ is connected, but if any edge is removed from $E$, the resulting graph is disconnected $\Rightarrow G$ is connected and $|E| = |V| - 1$

Given $G$ is connected, and removing any edge results in disconnecting the graph

often to prove equality, we have to prove both $\geq$ and $\leq$

$|E| \geq |V| - 1$:

**Base Step**: $|V| = 1 \Rightarrow |E| = 0$, $|E| \geq |V| - 1$

**Induction Hypothesis**: if $|V| = n$, then $|E| \geq n - 1$

**Induction Step**: suppose we have a connected graph $G$ with $|V| = n + 1$

Remove one vertex $v$, $V' = V - \{v\}$, $|V'| = n$ and $|E'| \geq |V'| - 1$ by IH

Now, add $v$ back, since $G$ is connected, it must add at least one edge. $|V| = |V'| + 1$ and $|E| \geq |E'| + 1$, $|E| \geq |V| - 1$

Common mistake: do not start with $G$ with $|V| = n$ and add one edge. You need to make sure the graph with $|V| = n + 1$ is connected at first.

UNIVERSITY OF TORONTO

# Proof cont.

$3 \Rightarrow 4$ (Part 2): $G$ is connected, but if any edge is removed from $E$, the resulting graph is disconnected $\Rightarrow$ $G$ is connected and $|E| = |V| - 1$

$|E| \leq |V| - 1$:

**Base Step**: $|V| = 1 \Rightarrow |E| = 0$, $|E| \leq |V| - 1$

$|V| = 2 \Rightarrow |E| = 1$, $|E| \leq |V| - 1$

**Induction Hypothesis**: if $|V| = k$, then $|E| \leq k - 1$, $\forall k \leq n$

**Induction Step**: suppose we have a graph $G$ satisfying 3 with $|V| = n + 1$

Removing an arbitrary edge separates $G$ into 2 connected components $G_1$, $G_2$

Each component satisfies $|E_1| \leq |V_1| - 1$, $|E_2| \leq |V_2| - 1$. Connect both,

$|E| = |E_1| + |E_2| + 1 \leq |V_1| + |V_2| - 1 = |V| - 1$

# Proof cont.

$4 \Rightarrow 5$: $G$ is connected and $|E| = |V| - 1 \Rightarrow G$ is acyclic and $|E| = |V| - 1$

Given $G$ is connected, $|E| = |V| - 1$

Assume that $G$ contains a cycle $v_1, ..., v_k, v_1$

WLOG, assume the cycle is simple

Let $G_k$ be the subgraph containing this cycle. $|V_k| = k$ and $|E_k| = k$

We add vertices back into $G_k$ to reconstruct $G$ and each time we must add at least one more edge since $G$ is connected

$|V_{k+i}| = k + i$ and $|E_{k+i}| \geq k + i = |V_{k+i}|, \forall i$

$|E| \geq |V|$ Contradiction.

# Proof cont.

$5 \Rightarrow 6$: $G$ is acyclic and $|E| = |V| - 1 \Rightarrow G$ is acyclic, but if any edge is added to $E$, the resulting graph contains a cycle

We actually do $5 \Rightarrow 1 \Rightarrow 2 \Rightarrow 6$

Let $k$ be the number of connected component of $G$.

Each connected component is a free tree, since $1 \Rightarrow 5$, $|E_i| = |V_i| - 1, \forall i = 1, ..., k$

$\sum_{i=1}^{k} |E_i| = \sum_{i=1}^{k} |V_i| - 1 = |V| - k$, need $k = 1$ to satisfy $|E| = |V| - 1$.

$G$ is fully connected, thus a free tree.

Since $1 \Rightarrow 2$, there must be a unique simple path connecting all vertices. Adding any edge creates a cycle.

# Proof cont.

$6 \Rightarrow 1$: $G$ is acyclic, but if any edge is added to $E$, the resulting graph contains a cycle $\Rightarrow G$ is a free tree

Given $G$ is acyclic and adding any edge creates a cycle

Suppose we add edge $(u, v)$, this creates a cycle which means removing $(u, v)$ leaves a path connecting $u$ to $v$.

$G$ is connected

$G$ is a free tree