$T(n) = 3T(\frac{n}{2}) + \text{constant} * n$ called recurrence equations.

Solving recurrences

- A recurrence is an equation or inequality that describes a function in terms of its value on smaller inputs.

- Analyse the running time of divide and conquer algorithm.

Methods to solve recurrences

1. Substitution method
2. Recursion tree method
3. The master method

Substitution method

This is the most general method

1. Guess the form of the solution

2. verify it by induction

3. solve some constants.

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

[Assume $T(1) =$ constant $\theta(1)$]

- Guess $O(n^3)$
- Assume that $T(k) \le ck^3$ for $k < n$

we need to prove, $T(n) \le cn^3$ by induction.

$$
\begin{aligned}
T(n) &= 4T\left(\frac{n}{2}\right) + n \\
&\le 4 \cdot c \cdot \left(\frac{n}{2}\right)^3 + n \\
&= \frac{c}{2} n^3 + n \\
&= \underbrace{cn^3}_{\text{required}} - \underbrace{\left[\frac{c}{2} n^3 - n\right]}_{\text{residual}}
\end{aligned}
$$

~~required~~

$T(n) \le cn^3$
whenever $\frac{c}{2} n^3 - n > 0$

This is true
when $c \ge 2$, $n \ge 1$

$\boxed{\text{so } T(n) = O(n^3)}$

Guess: $O(n^2)$

I.H. $T(K) \le cK^2$ for $K < n$

$$T(n) = 4T\left(\frac{n}{2}\right) + n$$
$$\le 4 \cdot c \cdot \left(\frac{n}{2}\right)^2 + n$$
$$= cn^2 + n$$
$$= \underbrace{cn^2}_{\text{required}} - \underbrace{[-n]}_{\text{residual}}$$

H.W. Try solving examples

Strengthen

I.H. $T(K) \le c_1 K^2 - c_2 K$ for $K < n$

$$T(n) = 4T\left(\frac{n}{2}\right) + n$$
$$\le 4 \cdot c_1 \left(\frac{n}{2}\right)^2 - 4c_2 \frac{n}{2} + n$$
$$= \underbrace{c_1 n^2 - c_2 n}_{\text{desired/required}} - [c_2 n - n]$$

$T(n) \le c_1 n^2 - c_2 n$ whenever $c_2 n - n \ge 0$

$\Rightarrow c_2 \ge 1$

$$T(n) = O(c_1 n^2 - c_2 n)$$
$$= O(n^2)$$

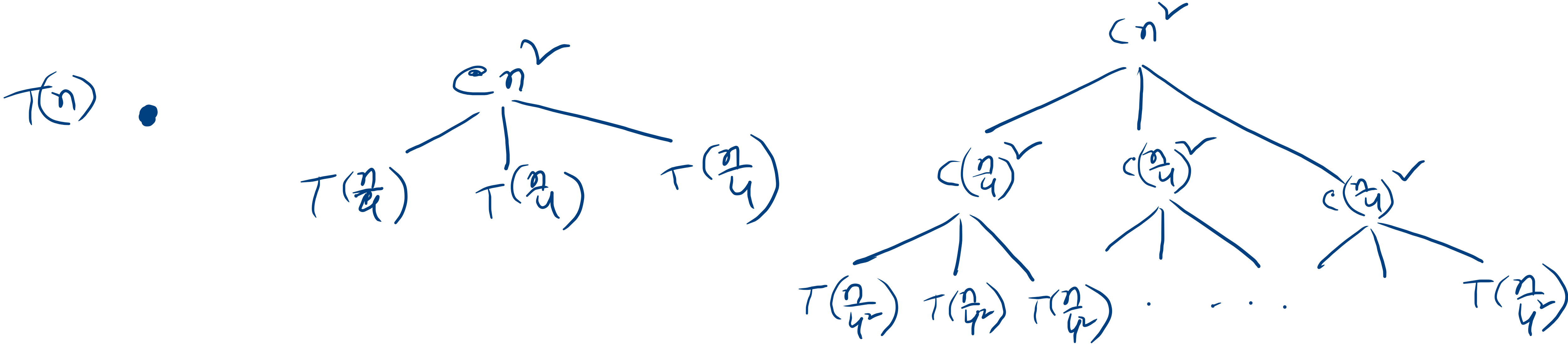
Recursion tree method

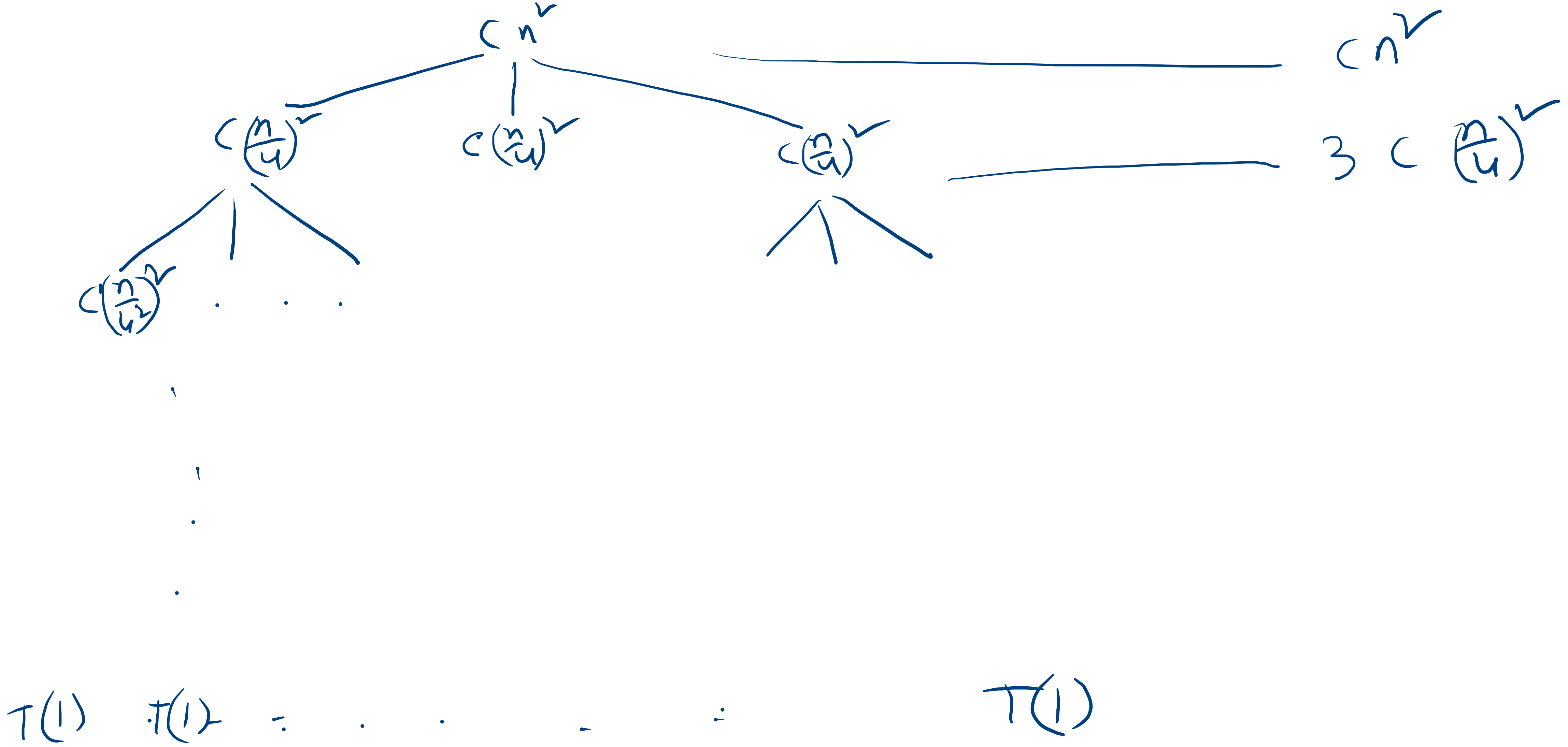
- A recursion tree models the cost (time) of a recursive execution of an algorithm.
- It is an intution of the running time of an algorithm
- It can be use as a guess for substitution method.

Guess the solution using recursion tree method.

$$T(n) = 3T\left(\frac{n}{4}\right) + \Theta(n^2)$$

$$\equiv \quad T(n) = 3T\left(\frac{n}{4}\right) + cn^2 \quad \text{for some constant } c.$$




c n^2
c(n/4)^2
c(n/4)^2
c(n/4)^2
c(n/4^2)^2
T(1)
T(1)
T(1)
c n^2
3 c (n/4)^2

cost at i-th depth

\# nodes in i-th depth $\rightarrow 3^i$

cost of each node at depth $i \rightarrow c \cdot \left(\frac{n}{4^i}\right)^2$

cost at depth $i \Rightarrow 3^i \cdot c \cdot \left(\frac{n}{4^i}\right)^2 = \left(\frac{3}{16}\right)^i c\, n^2$

Height Let i be the height

$$\frac{n}{4^i} = 1 \Rightarrow i = \log_4 n$$

cost at last level $\rightarrow 3^{\log_4 n} \cdot T(1)$

Total cost of the tree.

$$T(n) = \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i c n^2 + 3^{\log_4 n}$$

$$\leq \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i c n^2 + n^{\log_4 3}$$

$$= \frac{1}{1 - \frac{3}{16}} c n^2 + n^{\log_4 3}$$

$$= O(n^2)$$