Running Time of a Recursive Function

- An alternative to using derivation is to model the derivation with trees

PowerOf2A and PowerOf2B

- $TA(n) = 2\ TA(n - 1) + C$

- $TB(n) = TB(n - 1) + D$

- Let's build Trees to drill down
- Start with TB (easier)

PowerOf2B

- $TB(n) = TB(n-1) + D$

$$\begin{array}{c} D \\ | \\ TB(n-1) \end{array}$$

PowerOf2B

- TB(n - 1) = TB(n - 2) + D

$$
\begin{array}{c}
D \\
| \\
D \\
| \\
TB(n-2)
\end{array}
$$

PowerOf2B

- $TB(n - 2) = TB(n - 3) + D$

$$\begin{array}{c} D \\ | \\ D \\ | \\ D \\ | \\ TB(n-3) \end{array}$$

PowerOf2B

- This tree looks like a Linked list
- When do we stop?
- What is the running time? TB(n) = ??

PowerOf2B

- This tree looks like a Linked list
- When do we stop? At TB(0)
- What is the running time?
- At each level, we spend D time
- At the bottom level, we spend TB(0) time
- TB(n) = D + D + D + …. + D + TB(0)
- How many Ds are there?

PowerOf2B

- At each level, we spend D time
- At the bottom level, we spend TB(0) time
- TB(n) = D + D + D + …. + D + TB(0)
- How many Ds are there? n

- TB(n) = D n + TB(0) = Θ(n)

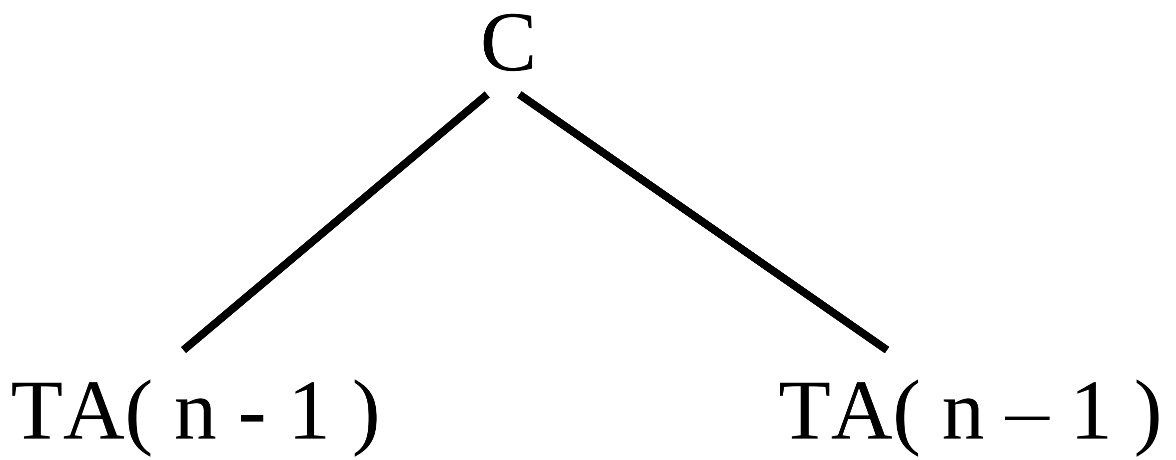
PowerOf2A

- TA(n) = 2 TA(n -1) + C

- Let's do TA
- Let's build a Tree to drill down

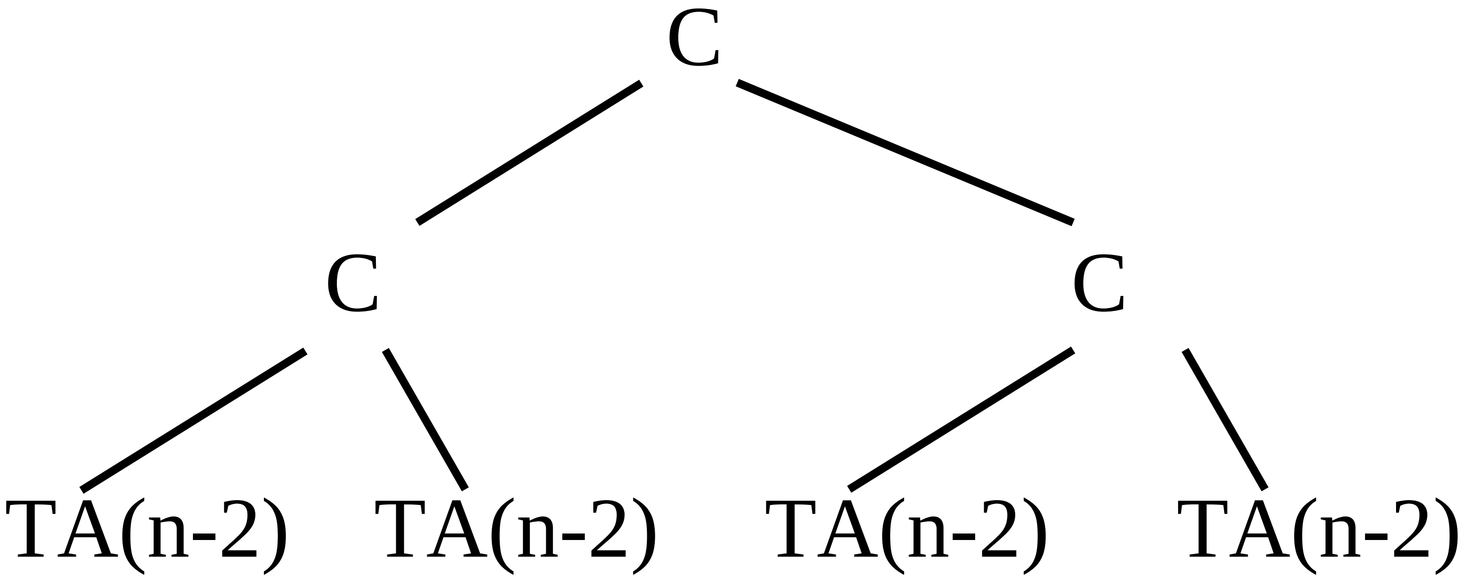
PowerOf2A

- $TA(n) = 2\,TA(n - 1) + C$
- Here is the corresponding tree



PowerOf2A

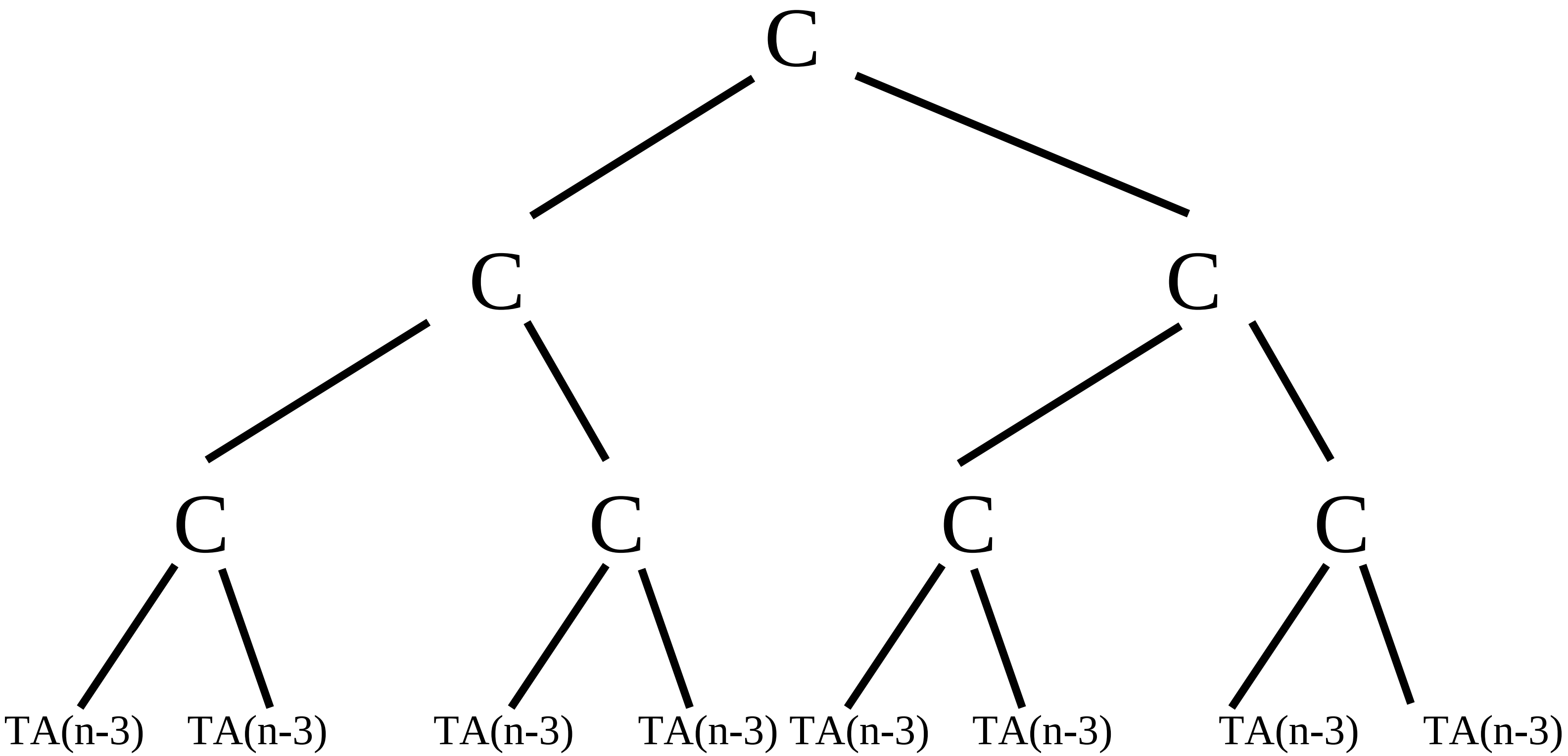
- $TA(n - 1) = 2\,TA(n - 2) + C$



- $TA(n) = C + 2C + 4TA(n - 2)$

PowerOf2A

- TA(n - 2) = 2 TA(n - 3) + C



- TA(n) = C + 2C + 4C + 8TA(n – 3)

PowerOf2A

Level	Number of Cs
0	1
1	2
2	4
3	??
L	??
What is the last level??	???

PowerOf2A

Level	Number of Cs
0	1
1	2
2	4
3	$8 = 2^3$
L	2^L
k such that n – k = 0	0 but we have a lot of TA(0)s

PowerOf2A

Level	Number of Cs
0	1
1	2
2	4
3	$8 = 2^{3}$
L	2^{L}
n - 1	2^{n-1}
n	0 (but we have 2^{n} TA(0))

PowerOf2A

- At level L, we spend $C * 2^L$ time
- At the bottom level, we spend $TA(0) * 2^n$
- $TA(n) = \sum_{L=0}^{n-1} C * 2^L + TA(0) * 2^n$

PowerOf2A

- $TA(n) = \sum_{L=0}^{n-1} C * 2^{L} + TA(0) * 2^{n}$

- $TA(n) = C \sum_{L=0}^{n-1} 2^{L} + TA(0) * 2^{n}$

PowerOf2A

- $TA(n) = C \sum_{L=0}^{n-1} 2^{L} + TA(0) * 2^{n}$

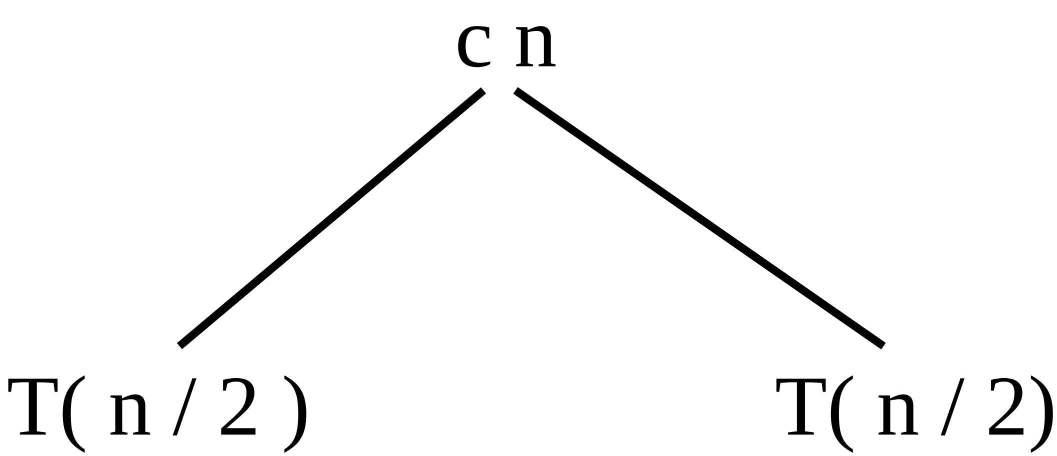
- $TA(n) = C * (2^{n} - 1) + TA(0) * 2^{n}$
- $TA(n) = \Theta(2^{n})$

MCS Divide and Conquer

- We have
- $T(n) = 2\,T(n/2) + c\,n$

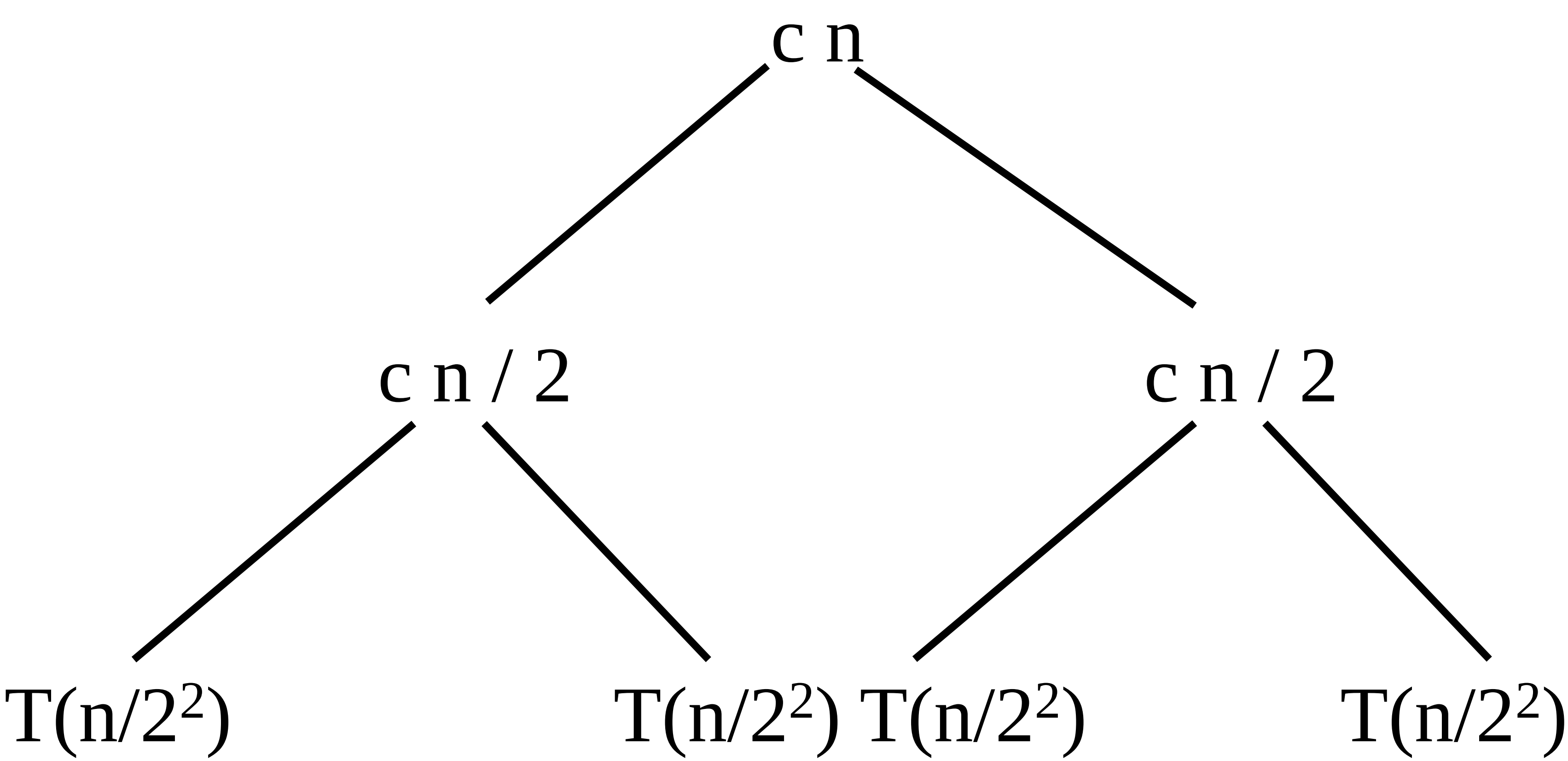
Recursion Trees

- $T(n) = 2\,T(n/2) + cn$
- Here is the corresponding tree



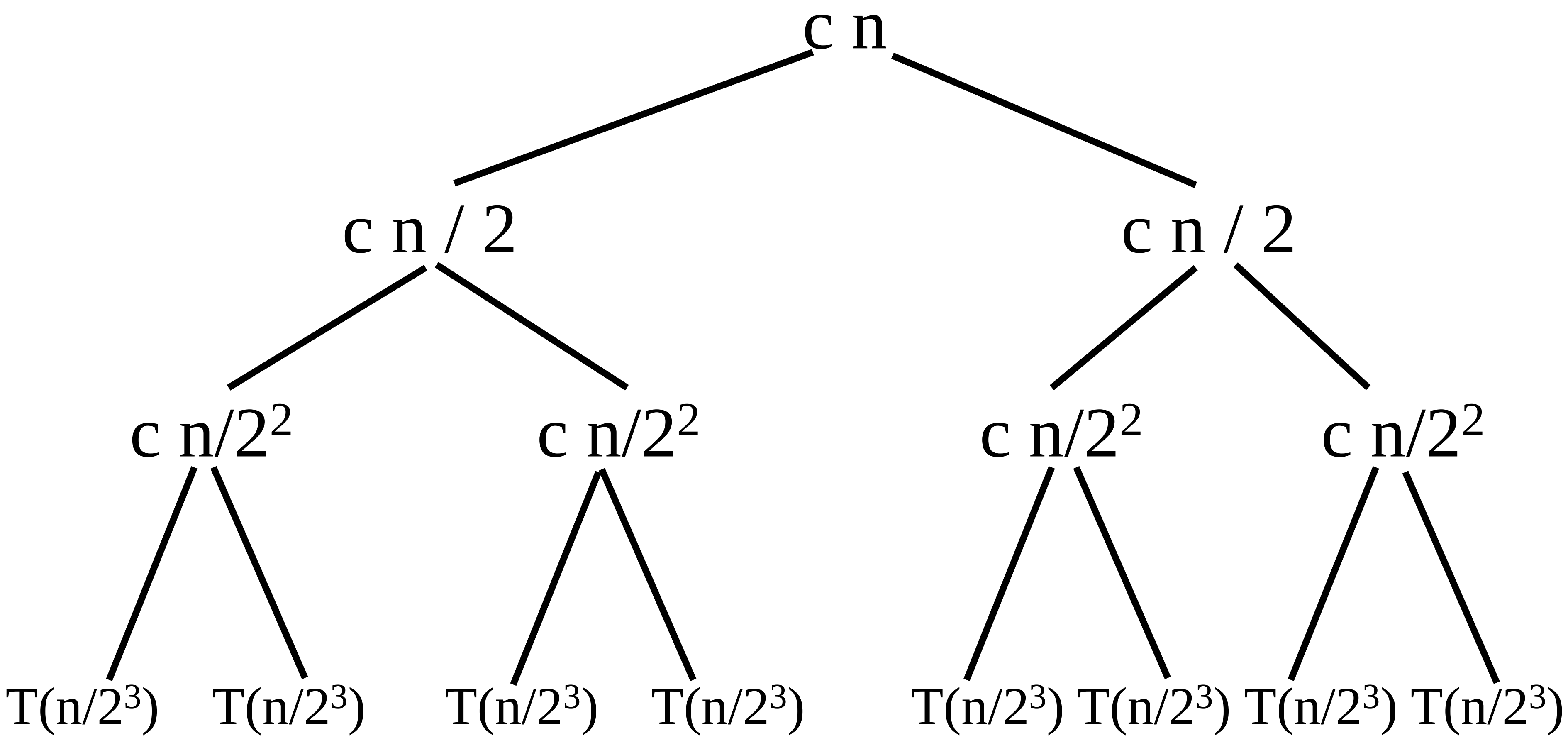
Recursion Trees

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



Recursion Trees

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



Recursion Trees

- At each level, it costs ???
- How many levels do we have ???
- How many leaves in the last level ???
- What is inside the leaves at the last level ???

Recursion Trees

- At each level, it costs c n
- How many levels do we have ? k
- How many leaves in the last level? 2^k where k is the level number (top level is level 0)
- What is inside the leaves at the last level ? $T(n / 2^k) = T(1)$ // base case

Recursion Trees

- $T(n) = c\, n * k +$ total costs of last level
- Total cost of last level $= T(1) * 2^{k}$
- $T(n / 2^{k}) = T(1) \Rightarrow n / 2^{k} = 1$
- $\Rightarrow k = \log n$
- $\Rightarrow T(n) = c\, n \log n + n\, T(1)$

- SAME equation as we generated using derivation $\Rightarrow \Theta(n \log n)$

Divide and Conquer

- Generally, we could have
- $T(n) = a T(n / b) + f(n)$
- Where $a >= 1$ and $b > 1$

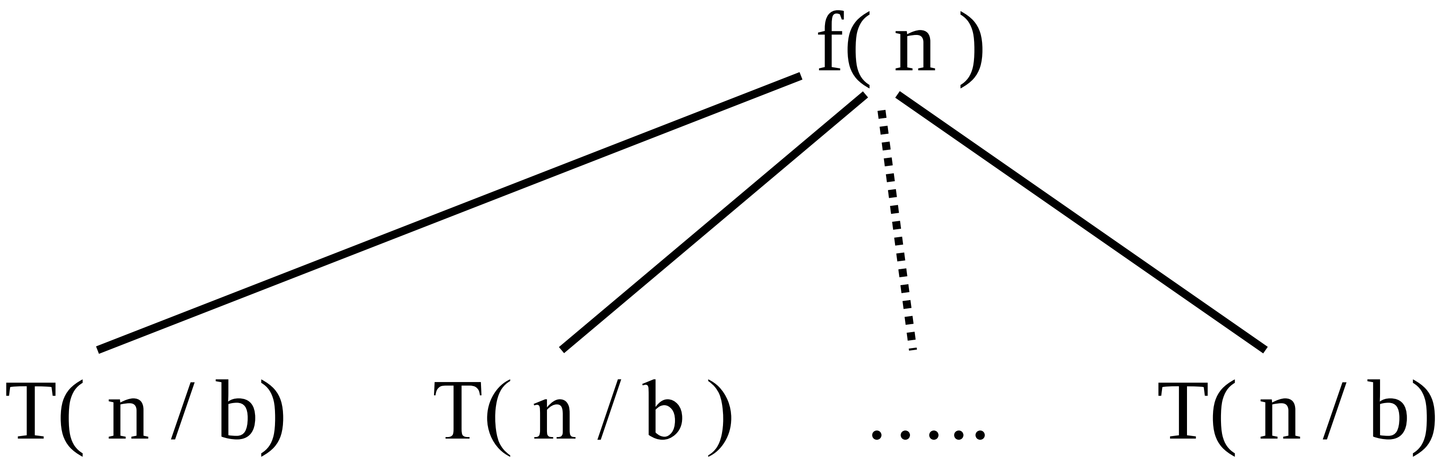
Divide and Conquer

- $T(n) = a\,T(n / b) + f(n)$
- Where $a >= 1$ and $b > 1$

- Note 1: if $b < 1$, the problem gets bigger, not smaller
- Note 2: if $b = 1$, the size of the problem would stay the same

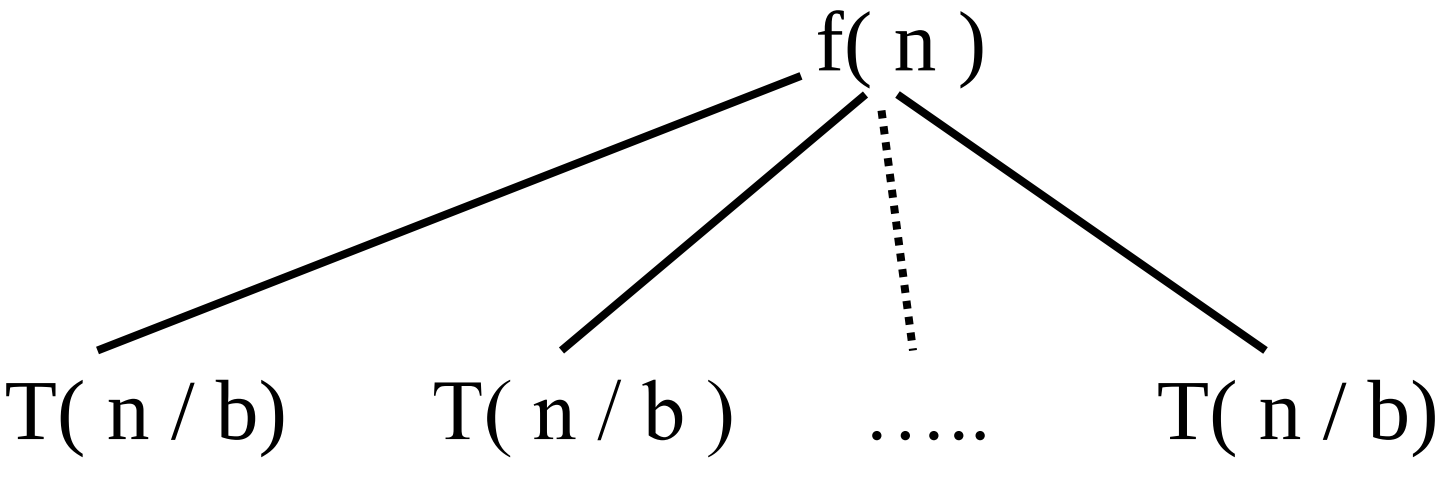
Recursion Trees

- $T(n) = a\,T(n/b) + f(n)$
- Here is the corresponding tree



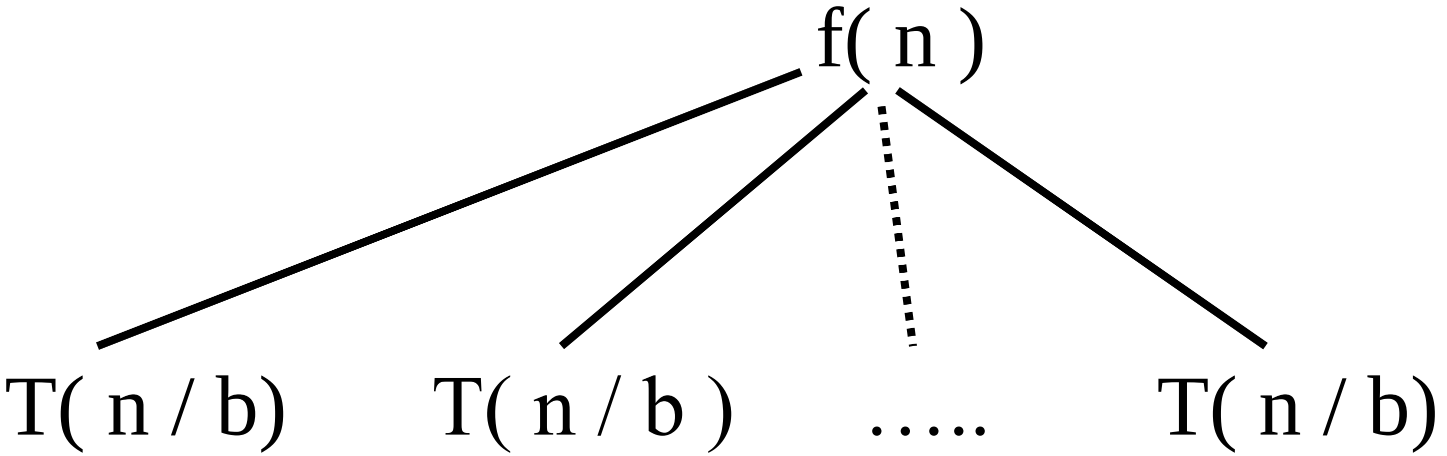
Recursion Trees

- $T(n) = a\,T(n / b) + f(n)$
- How many branches in the tree below?



Recursion Trees

- $T(n) = a\,T(n / b) + f(n)$
- How many branches in the tree below? a



Recursion Trees

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