

Trees and CFGs



Discrete Structures (CS 173) Lecture B

Gul Agha

Based on slides by Derek Hoiem, University of Illinois ¹

Last class: recursive functions

$$f(n) = \sum_{i=1}^n i = 1 + 2 + 3 + \cdots + n$$

$$f(1) = 1 \quad \leftarrow \text{base case}$$

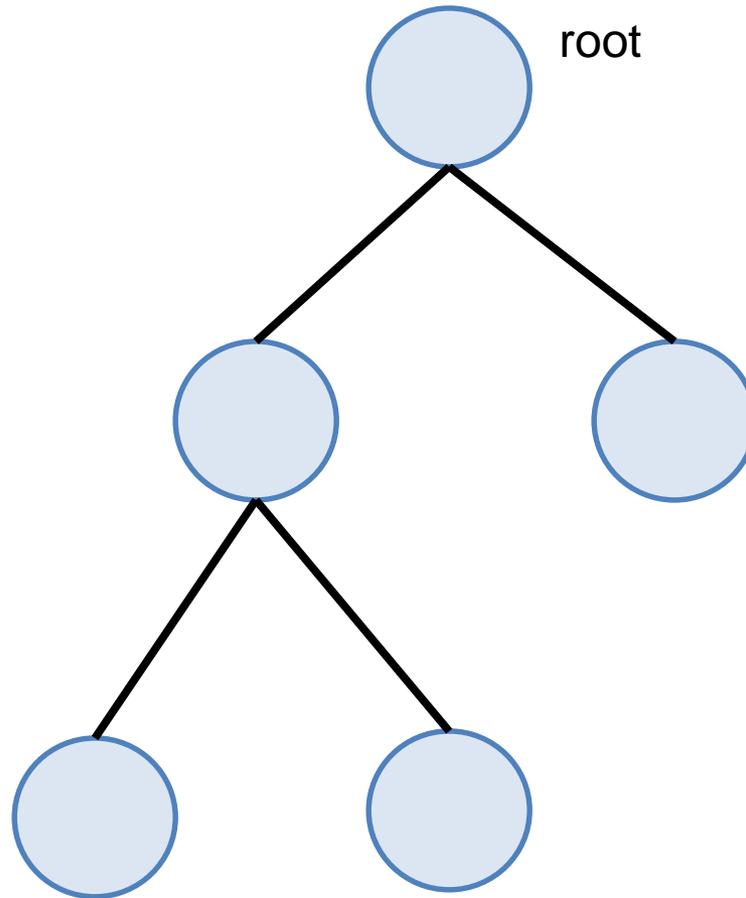
$$f(n) = f(n - 1) + n \quad \leftarrow \text{recursive formula}$$

- Process for finding closed form
 1. Unroll for several steps
 2. Write in terms of $n - k$ or n/k
 3. Substitute for value of k that is base case
 4. Substitute base case value(s) and solve
- Verifying closed form with induction

Today's lecture: Trees and CFGs

- Trees
 - Examples of uses
 - Terminology
 - Induction on trees
- Context free grammars (CFGs)
 - What they are, how they work
 - Induction on CFG

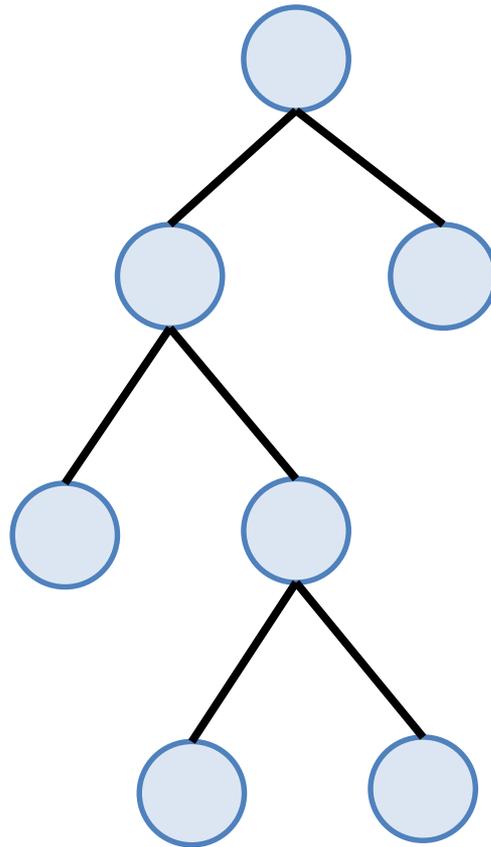
Tree: special form of graph with “root” and no cycles



Tree terminology

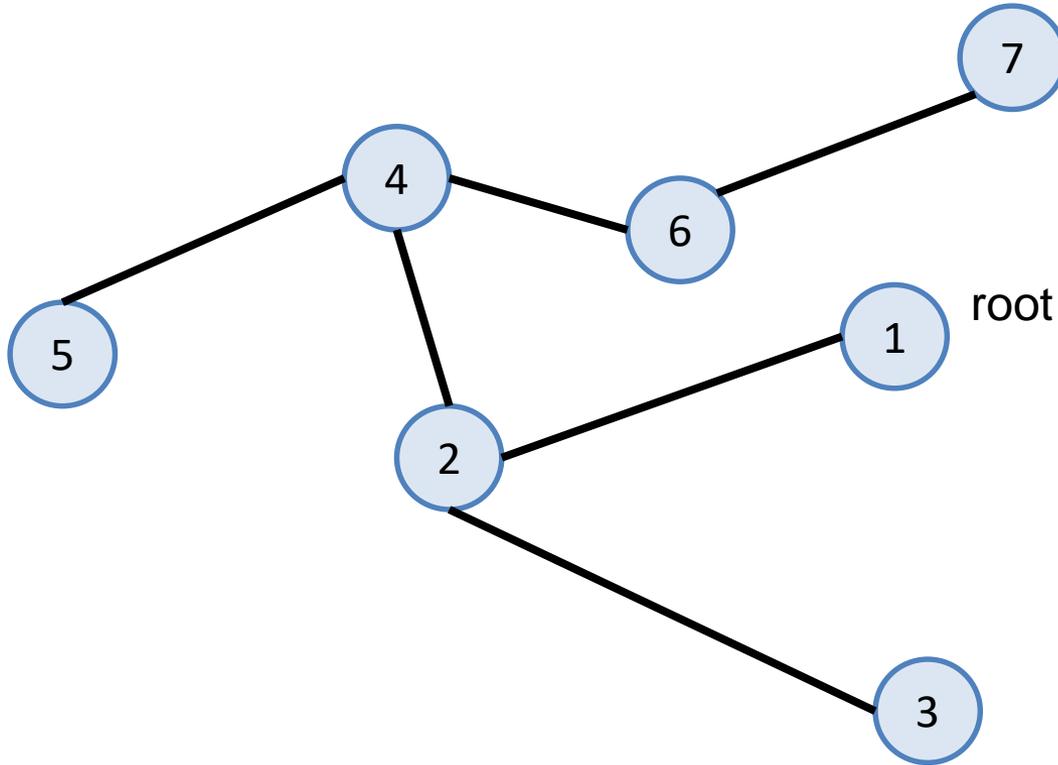
Nodes: root, internal, leaf, level, tree height

Relations: parent/child/sibling, ancestor/descendant

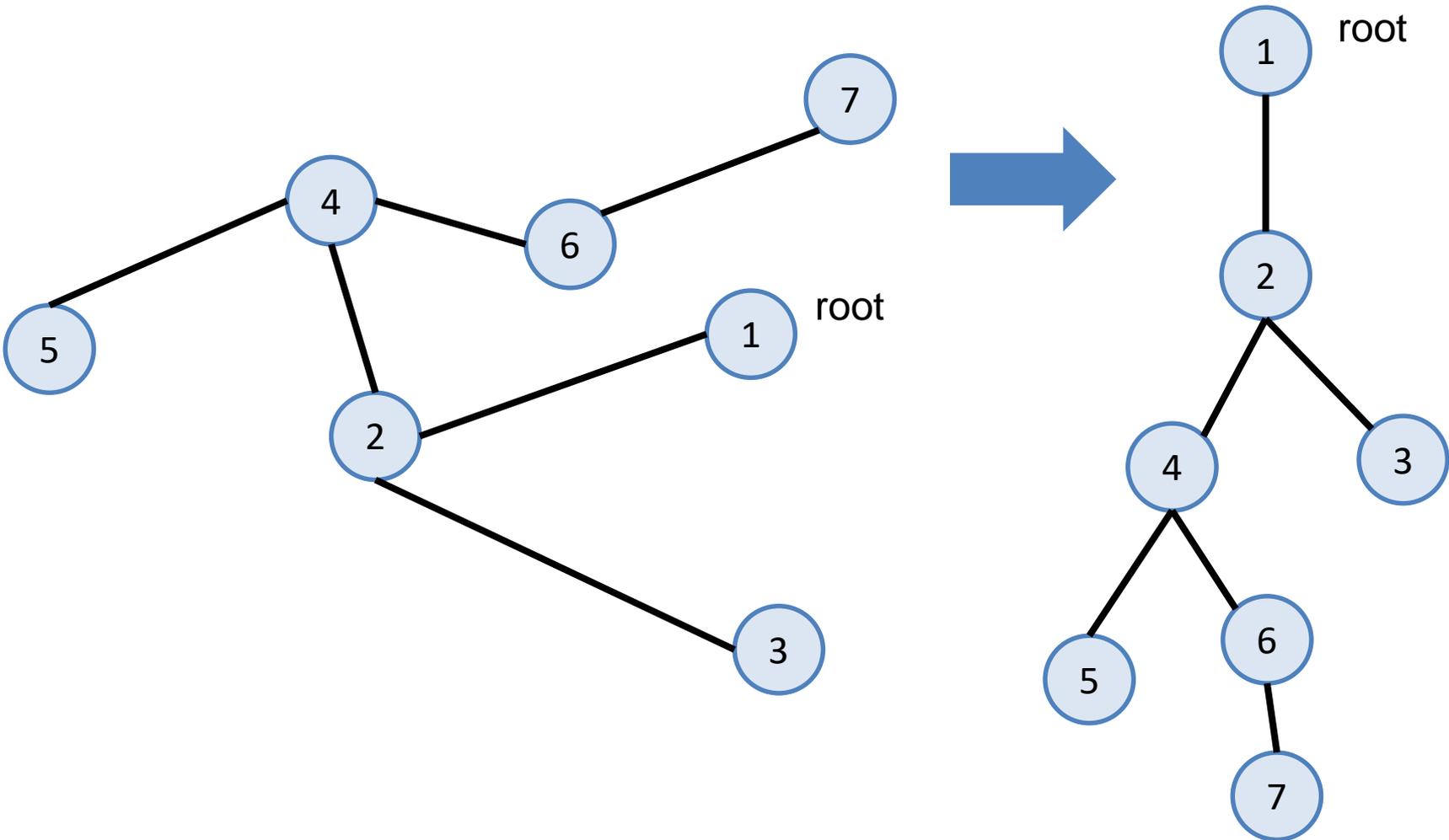


overhead

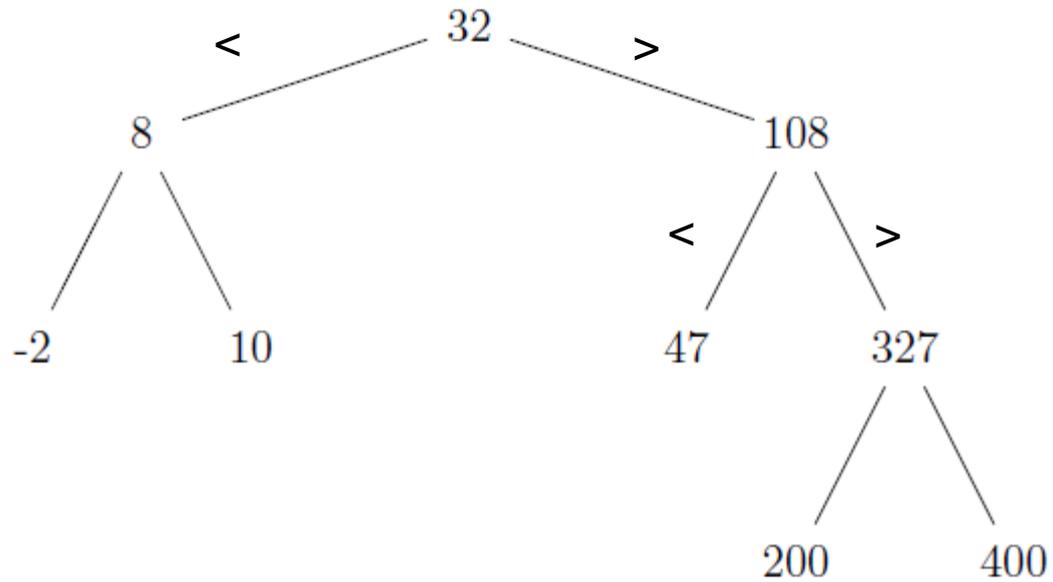
Another example



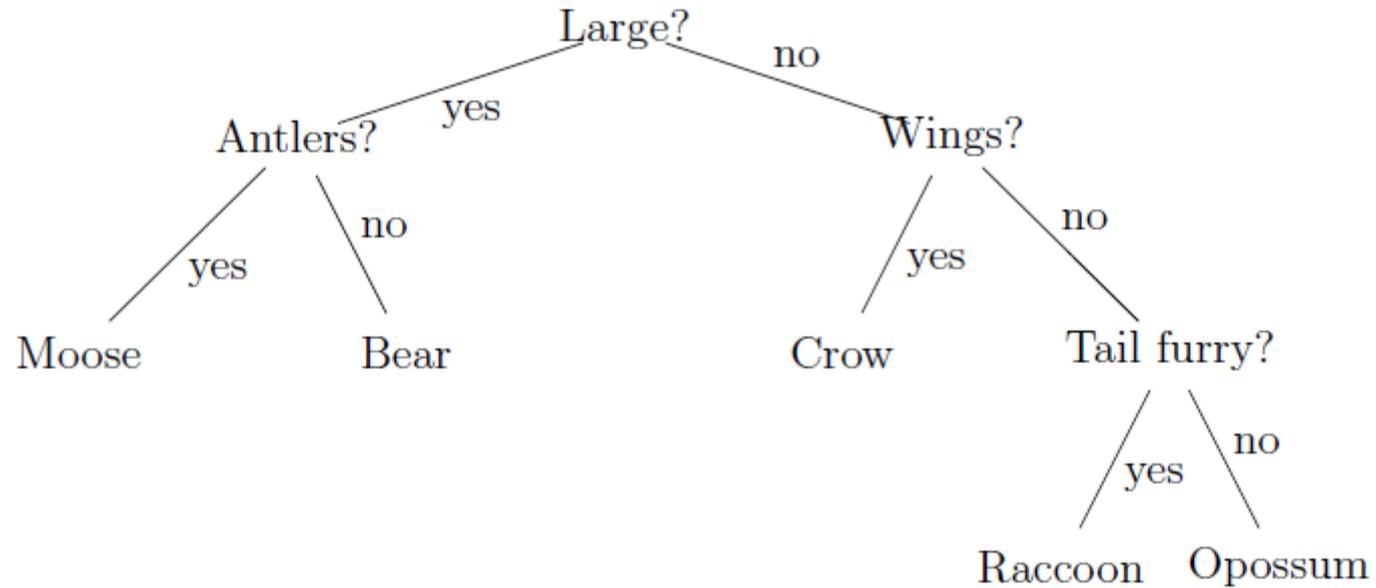
Another example



Trees for sorting



Decision trees



Hierarchical data structure

The screenshot displays the WebMD Symptom Checker interface, which is organized into three main steps: 1. Choose Symptom(s), 2. Your Choices, and 3. Possible Conditions. The user is currently in Step 1, where they have selected 'Cough' from a list of chest symptoms. A pop-up window titled 'Refine Your Symptoms.' is overlaid on the interface, allowing the user to further specify the type of cough. The pop-up includes a question 'Q: Cough:' and a list of possible answers (A:) with checkboxes for selection. The interface also includes a body map, a search bar, and navigation buttons like 'Skip Questions' and 'Next'.

WebMD symptom checker

Gender: male
Age: 25-34 years

Start Over Print Save Symptoms Take the Tour

1 Choose Symptom(s)

Body Map

Chest Symptoms:

- Bleeding
- Bleeding from nipple
- Broken bone (single fracture)
- Broken bones (multiple fractures)
- Bruising or discoloration
- Cough
- Difficulty talking
- Discharge from nipple
- Drainage or pus
- Episodes of not breathing during sleep
- Feeling of not being able to get enough air
- Food getting stuck (swallowing)
- Hoarseness

Back View Zoom Out

Don't know where to point?
[More symptoms here](#)

Search Symptoms

2 Your Choices

3 Possible Conditions

1 Choose Symptom(s)

Refine Your Symptoms.

1 of 4 [Skip Questions](#) [Next](#)

Q: Cough:

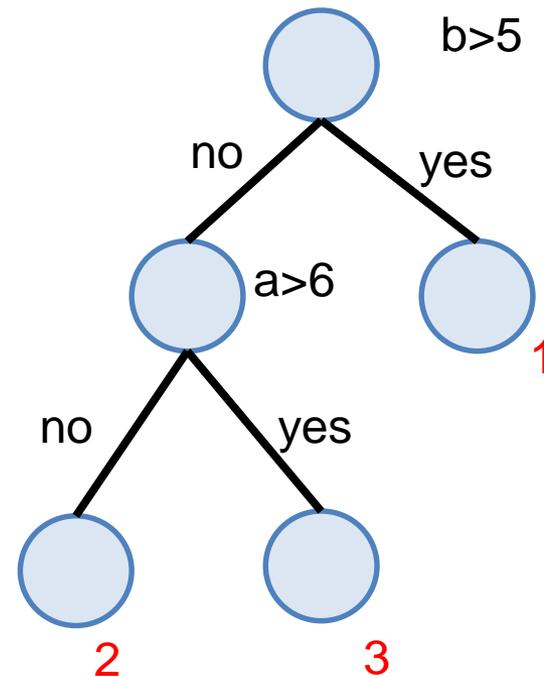
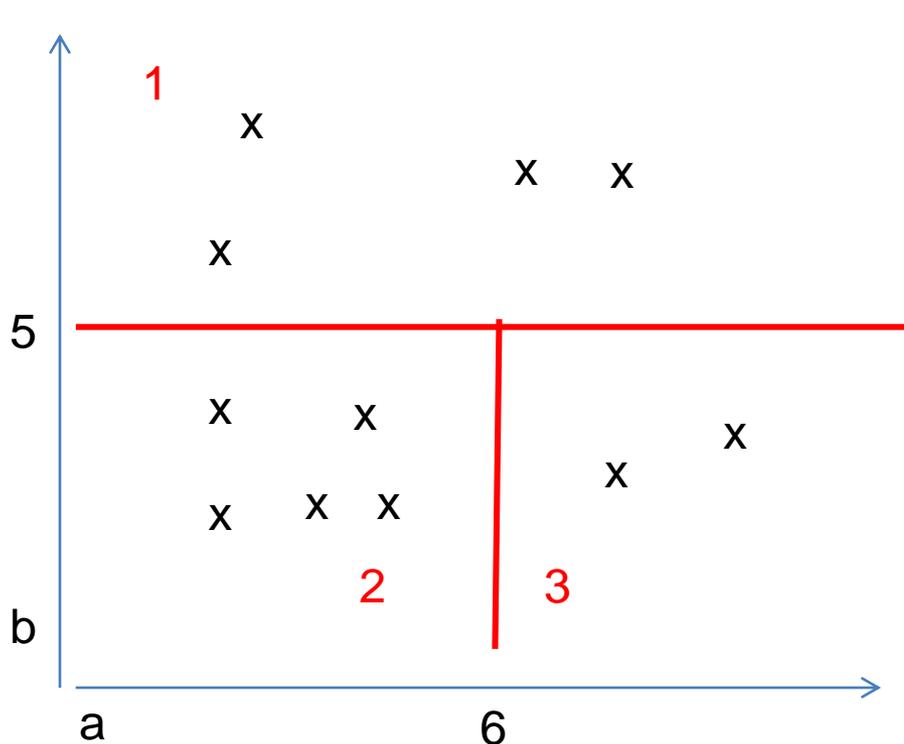
A:

- high pitched or barking
- coughing up blood or blood in sputum
- hacking
- white or pink sputum
- whooping noise when inhaling
- yellow or green sputum
- no sputum (non-productive)
- None of the above

1 of 4 [Skip Questions](#) [Next](#)

Trees for clustering

- Goal: create a function that maps from R^N to Z^+ such that nearby N-dimensional points are mapped to the same integer



More terminology

- m -ary tree: each node can split into m subtrees
- full: each node splits 0 or m times
- complete: all leaves have the same height
- Balanced: all leaves are “approximately” the same height
- How many nodes in a full m -ary tree with i internal nodes?
- What are the lower/upper bounds on the number of nodes n of an m -ary tree with height h ?

Induction proof on trees

Claim: In a binary tree of height h , the number of nodes $n \leq 2^{h+1} - 1$.

Context-free Grammars

A **context-free grammar (CFG)** is a set of rules that defines a set of possible **parse trees**.

A CFG specifies a set of rules, valid start symbols, and valid terminals.

Example:

$$S \rightarrow S a$$

$$S \rightarrow a \mid b \mid c$$

Start symbol: S

Terminals: a, b, c

CFG Example

Example:

$$S \rightarrow b S a$$

$$S \rightarrow a \mid b \mid c a$$

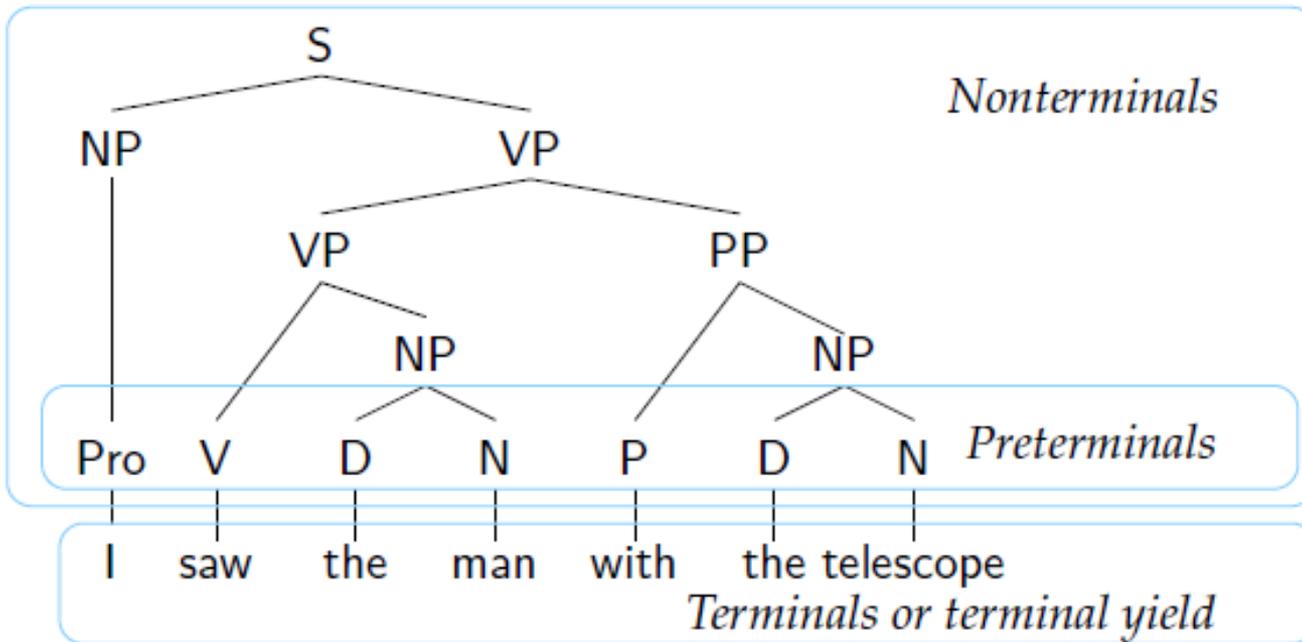
Which of these strings can be generated by the grammar above?

a *bba* *ba* *abbcaa*

bca *bbbcaaaa*

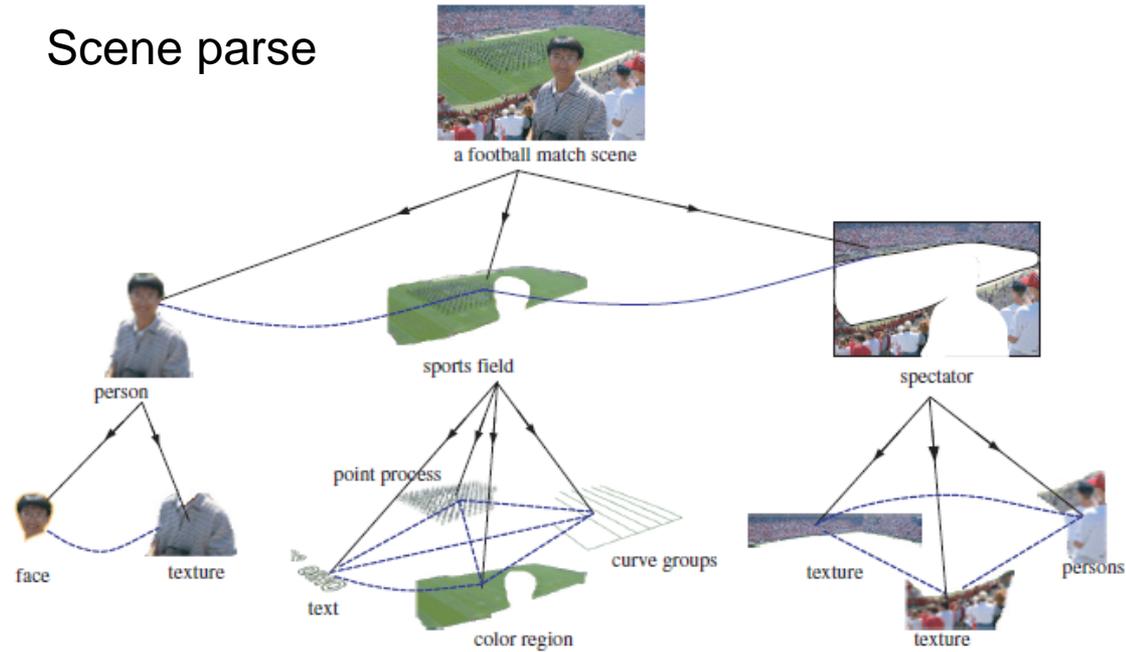
Examples of parse trees

Language

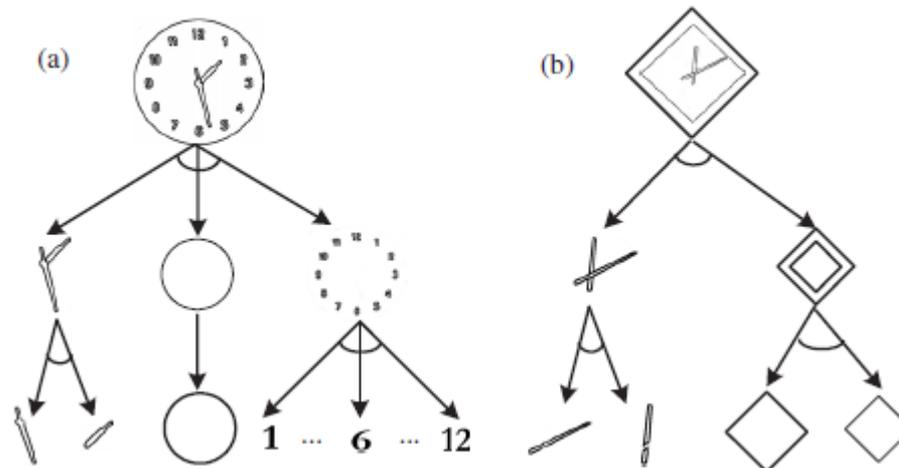


Examples of parse trees

Scene parse



Object Parse



Examples of parse trees

Stochastic CFG for blackjack actions

Production Rules		Description
S	$\rightarrow AB$	[1.0] Blackjack \rightarrow "play game" "determine winner"
A	$\rightarrow CD$	[1.0] play game \rightarrow "setup game" "implement strategy"
B	$\rightarrow EF$	[1.0] determine winner \rightarrow "evaluate strategy" "cleanup"
C	$\rightarrow HI$	[1.0] setup game \rightarrow "place bets" "deal card pairs"
D	$\rightarrow GK$	[1.0] implement strategy \rightarrow "player strategy"
E	$\rightarrow LKM$	[0.6] evaluate strategy \rightarrow "flip dealer down-card" "dealer hits" "flip player down-card"
	$\rightarrow LM$	[0.4] evaluate strategy \rightarrow "flip dealer down-card" "flip player down-card"
F	$\rightarrow NO$	[0.5] cleanup \rightarrow "settle bet" "recover card"
	$\rightarrow ON$	[0.5] \rightarrow "recover card" "settle bet"
G	$\rightarrow J$	[0.8] player strategy \rightarrow "Basic Strategy"
	$\rightarrow Hf$	[0.1] \rightarrow "Splitting Pair"
	$\rightarrow bffjH$	[0.1] \rightarrow "Doubling Down"
H	$\rightarrow l$	[0.5] place bets
	$\rightarrow lH$	[0.5]
I	$\rightarrow ffi$	[0.5] deal card pairs
	$\rightarrow ee$	[0.5]
J	$\rightarrow f$	[0.8] Basic strategy
	$\rightarrow fJ$	[0.2]
K	$\rightarrow e$	[0.6] house hits
	$\rightarrow eK$	[0.4]
L	$\rightarrow ae$	[1.0] Dealer downcard
M	$\rightarrow dh$	[1.0] Player downcard
N	$\rightarrow k$	[0.16] settle bet
	$\rightarrow kN$	[0.16]
	$\rightarrow j$	[0.16]
	$\rightarrow jN$	[0.16]
	$\rightarrow i$	[0.18]
	$\rightarrow iN$	[0.18]
O	$\rightarrow a$	[0.25] recover card
	$\rightarrow aO$	[0.25]
	$\rightarrow b$	[0.25]
	$\rightarrow bO$	[0.25]

Symbol	Domain-Specific Events (Terminals)
a	dealer removed card from house
b	dealer removed card from player
c	player removed card from house
d	player removed card from player
e	dealer added card to house
f	dealer dealt card to player
g	player added card to house
h	player added card to player
i	dealer removed chip
j	player removed chip
k	dealer pays player chip
l	player bets chip

CFG example

$*S \rightarrow NP VP$

$VP \rightarrow V NP \mid V VP \mid to V \mid VP NP \mid \epsilon$

$NP \rightarrow NP \text{ and } NP \mid NP NP \mid N \mid \epsilon$

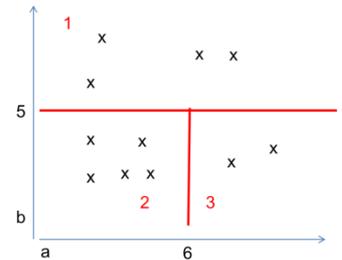
$V \rightarrow like \mid eat \mid drink \mid hate$

$N \rightarrow I \mid apples \mid bananas \mid coconuts \mid you$

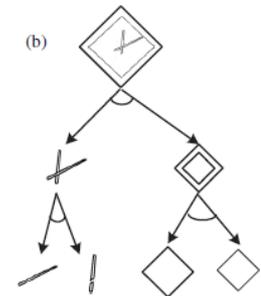
I like to eat apples and bananas

Things to remember

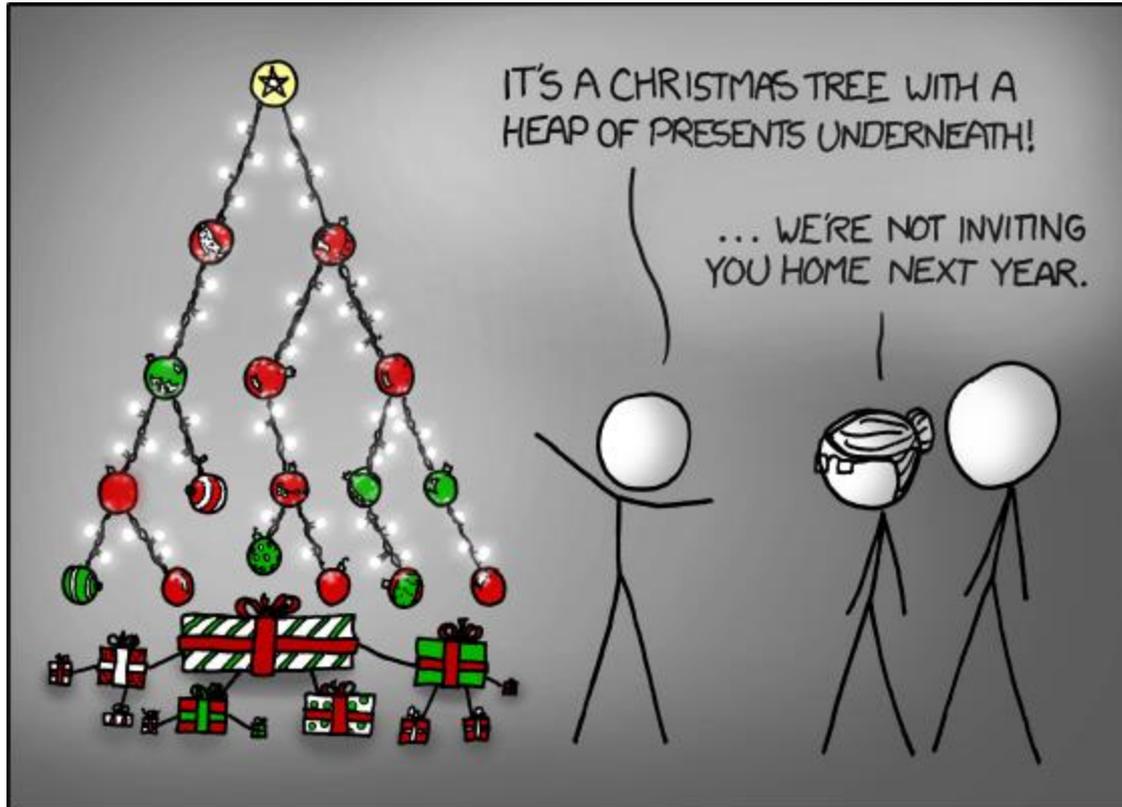
- Trees are a special graph with root and no cycles, with many uses
 - Sorting, clustering, finding similar values
 - Decision tree: machine learning, modeling choices
 - Parse trees: representing hierarchical structures



- Context free grammars: generate parse trees



- Proofs on trees: split at root, use inductive hypothesis on subtrees headed by the root's children

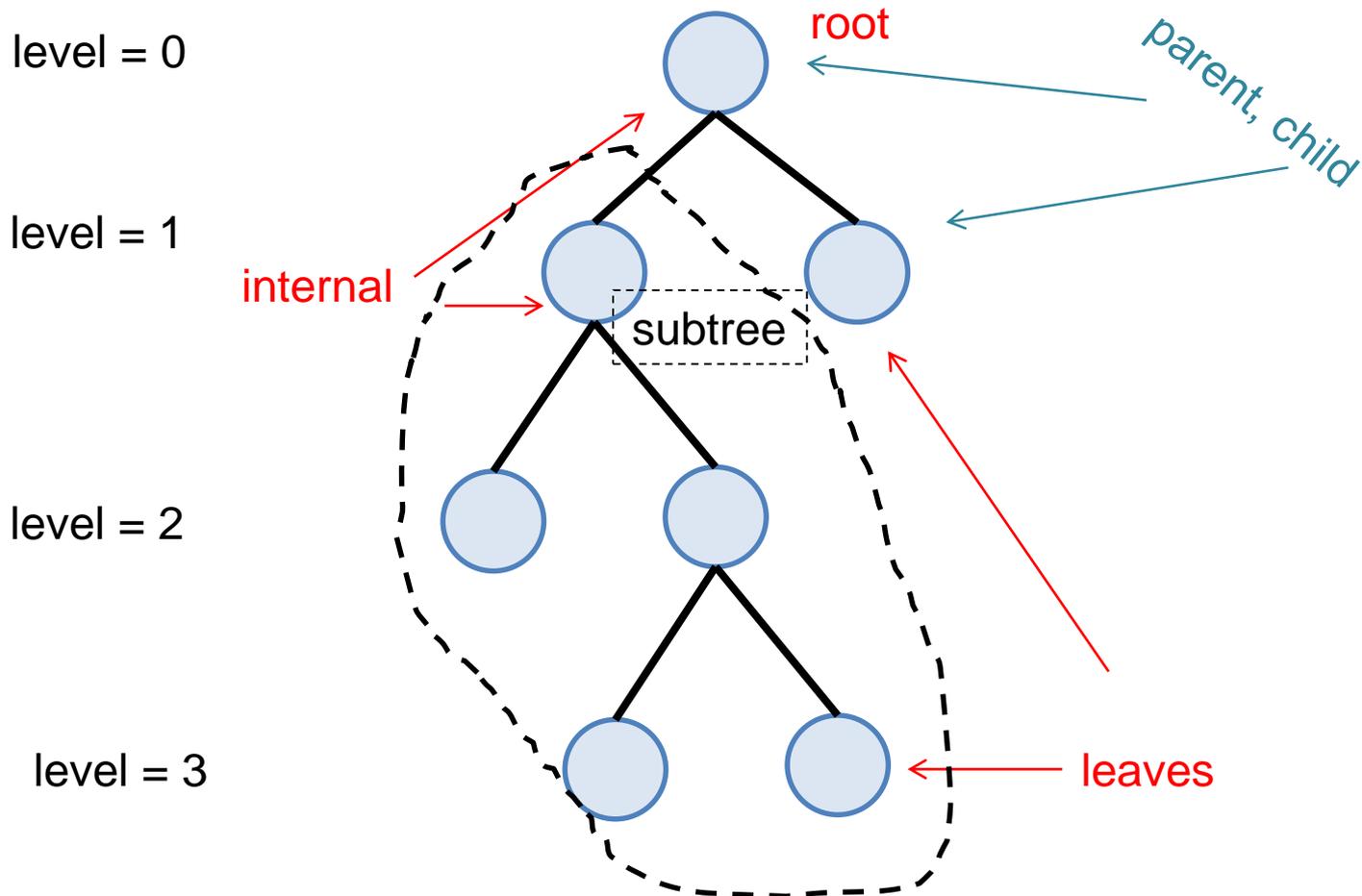


xkcd

Tree terminology

Nodes: root, internal, leaf, level, tree height

Relations: parent/child/sibling, ancestor/descendant



Induction proof on CFG

$S \rightarrow a S b \mid a S S c$

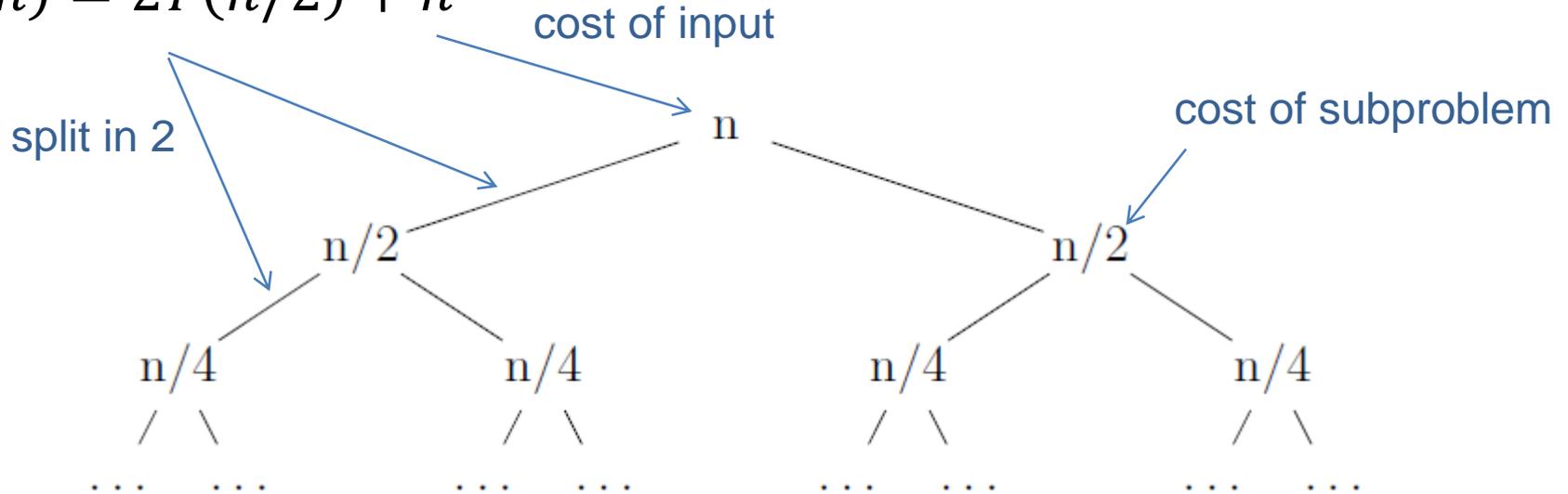
$S \rightarrow a b \mid a c$

Claim: For any string generated by the grammar G above, the number of a 's will be equal to the number of b 's plus the number of c 's ($n_a = n_b + n_c$)

Recursion trees

$$T(1) = c$$

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



Cost of each level of internal nodes?

Height of tree?

Total cost of internal nodes?

Cost of leaf nodes?

Total cost = leaf cost + internal cost:

Useful formulas

$$\sum_{k=0}^n r^k = \frac{r^{n+1} - 1}{r - 1}$$

$$\sum_{k=m}^n r^k = \frac{r^{n+1} - r^m}{r - 1}$$

$$\sum_{k=1}^n k = \frac{n(n+1)}{2}$$

$$\sum_{k=0}^n 2^k =$$

$$\sum_{k=1}^n 2^k =$$

$$\sum_{k=0}^n 2^{-k} =$$

$$\sum_{k=0}^n 2^{k+2} =$$

$$(m^a)^b = m^{ab}$$

$$m^{a+b} = m^a m^b$$

$$2^{\log_2 n} = n$$

$$\log_a(b) = \log_2(b) / \log_2(a)$$

$$\log(mn) = \log(m) + \log(n)$$

$$2^{\log_4(n)+2} =$$

Recursion trees

$$T(1) = c$$

$$T(n) = 2T(n - 1) + d$$

Recursion trees

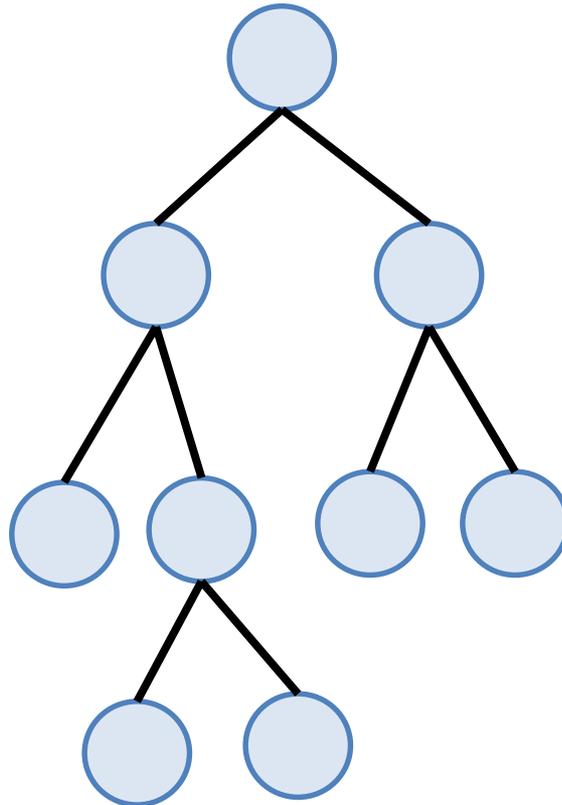
$$A(1) = c$$

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

Tree induction proof

If T is a binary tree with root r , then its **rank** is

- (a) 0 if r has no children
- (b) $1 + q$ if r has two children, both with rank q
- (c) otherwise, the maximum rank of any of the children



Tree induction proof

If T is a binary tree with root r , then its **rank** is

- (a) 0 if r has no children
- (b) $1 + q$ if r has two children, both with rank q
- (c) otherwise, the maximum rank of any of the children

Claim: A tree with rank q has at least 2^q leaves.