Q: What is an algorithm?

iClicker

A. A series of steps written to solve a specific problem with specific inputs

B. A series of steps written to solve a specific problem with non-specific inputs

C. A series of steps written to solve a non-specific problem with specific inputs

D. A series of steps written to solve a non-specific problem with non-specific inputs

CPSC 100

Computational Thinking

Algorithm, Classifiers and Trees!

Instructor: Parsa Rajabi
Department of Computer Science
University of British Columbia

Agenda

- Course Admin
- Classifiers
- Intro to Decision Trees
- Activity
  - In-class (if time allows), if not, take home!

Learning Goals

Learning Goals

After this lecture, you should be able to:

- Describe the **classification** steps.
- Explain the concept of a **rooted tree** and **decision tree**.
- Describe what the **general decisions** are in building a decision tree.

Course Admin

Course Admin

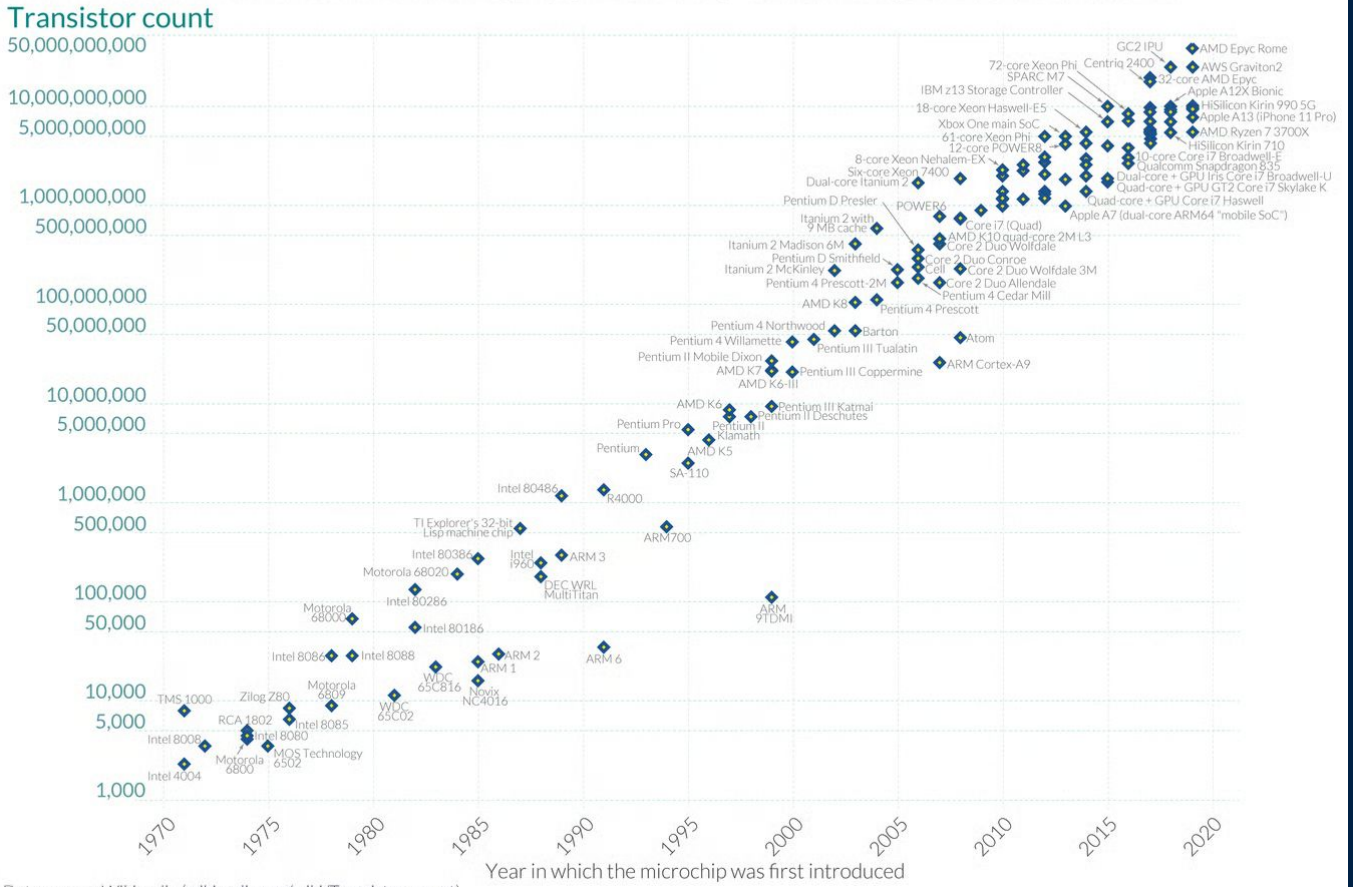
- **Lab #2**
  - Due on Friday, Jan 24 at 11:59pm
- **Post-Class (PC) Quiz #1**
  - Only 1 attempt, 60 minutes
  - Open book, open-AI* (*you must disclosure your usage)
  - Due on Sunday, Jan 26 at 11:59pm
- **Group Contracts**
  - Extended to Monday, Jan 27 at 11:59pm
- **PC Quiz #2**
  - To be released next week!

Does Moore's Law still hold/valid?

Moore's Law: The number of transistors on microchips has doubled every two years

Our World in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years.
This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.



Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)
OurWorldinData.org – Research and data to make progress against the world's largest problems.

The Transistor Cliff

Sarah Constantin

Moore's law may be coming to an end. What happens to AI progress if it does?

How Hardware Affects AI Progress

The biggest AI models are trained on expensive, state-of-the-art microchips, or semiconductors. Only a few organizations, such as Google and OpenAI, have the budgets to train them. For years, improvements in AI performance have been driven by progress in this underlying hardware.

For most of the history of semiconductor manufacturing, steadily and predictably accelerating improvements in performance and reductions in price have been the norm. This pattern has been codified as "Moore's Law," Intel CEO Gordon Moore's observation that the number of transistors that could be placed on a chip for the same price doubled approximately every two years. That may be coming to an end. Depending on the specific semiconductor performance metric, Moore's Law has either stalled out already, or is on course to soon hit fundamental physical limits.

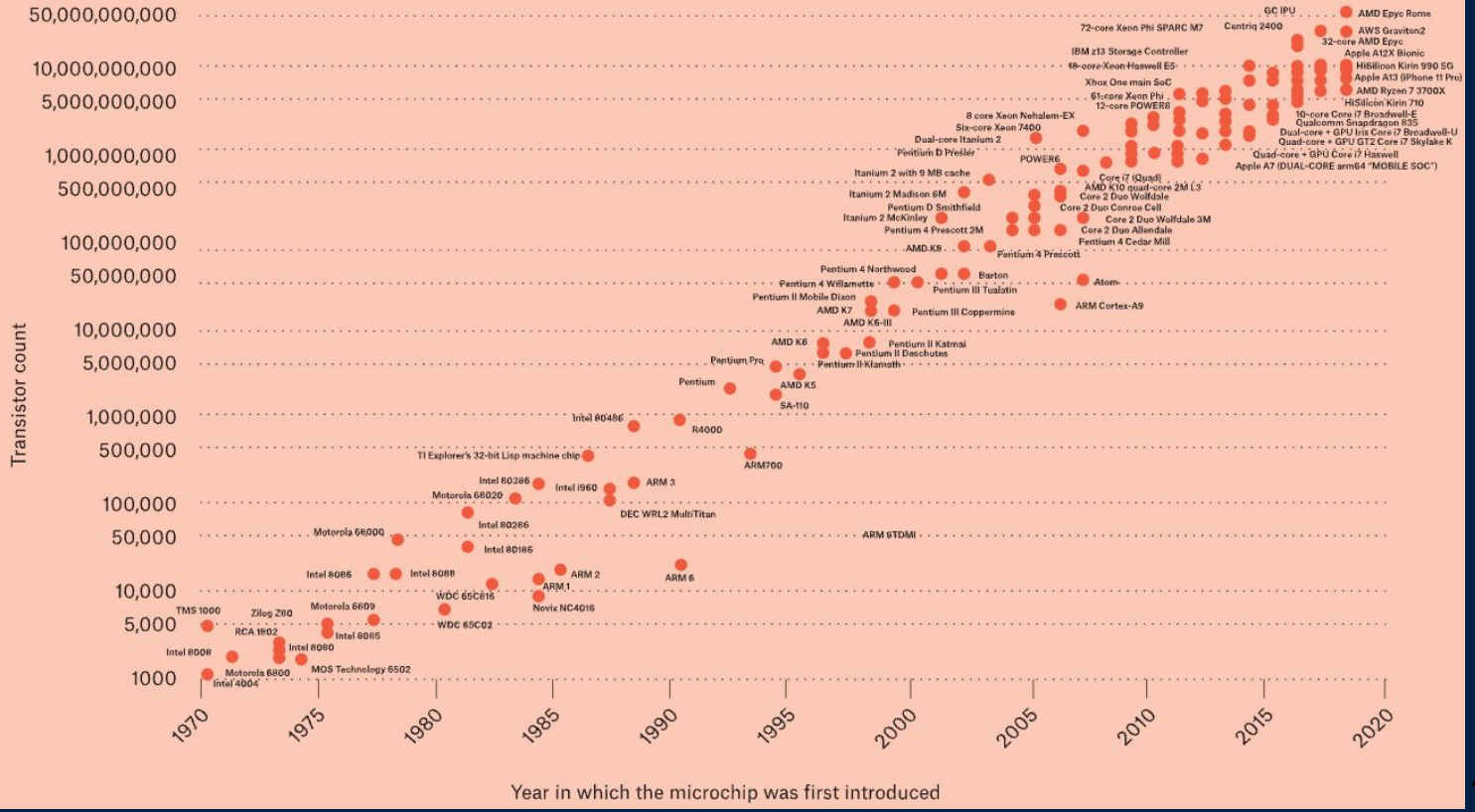
So, what could happen "after Moore's Law?" And how would that affect AI performance?

Let's zoom in and look at the details.

What Does Scaling Mean?

Source

Moore's Law: The number of transistors on microchips has doubled every two years



[Source]

Algorithms

Algorithms

An ***algorithm*** describes a sequence of steps that is:

1. Unambiguous
   - No “assumptions” are required to execute the algorithm
   - The algorithm uses precise instructions
2. Executable
   - The algorithm can be carried out in practice
3. Terminating
   - The algorithm will eventually come to an end, or halt

Classifiers

Classifier

- A **classifier** is an **algorithm** that maps the input data to a specific category
  - Classifiers are derived from patterns or correlations from data.

Classifier: Training vs Test Data

- The data that classifiers learn the patterns has the "answer"
  - This data is called **training data**.

- Some of the training data is held back to check and see if the classifier works.
  - This is called **test data.**

Classifiers + Data

Classifiers then apply these patterns to new data with no “answer”

- **Input:** Digital image
- **Output:** Cat/not a cat
- **Training data:**
  Labeled images of cats and images that are not cats

Classification Task: Loans

Classification Task - Cancer Treatment

Input: Genome sequence from cancerous biopsy tissue

- Address, age, gender, credit rating, etc.

Output: Which cancer treatment is likely to work best

Training data:

Labeled genome sequences and which treatments were successful from both cancerous tissue

Classification Task - Loan Applications

Input: Individual's loan application

- Address, age, gender, credit rating, etc.

Output: Acceptance/Rejection

Training data:

List of loan apps, decisions made, and for those who were approved, whether they repaid the loan or not

Building a Classifier: Loans

Building a Classifier: Loans

You want to create a classifier to help you decide whether or not to give people loans

Here is your past (training) data on some loans

Applicant	Income	Gave loan?
#1	$50,005	Yes
#2	$25,004	No
#3	$75,005	Yes
#4	$95,005	Yes
#5	$45,007	No

***Task**: Create an algorithm to decide what your classifier does: i.e., when will you give a loan, and when will you not give a loan?*

Clicker Checkpoint

Q: Would your classifier give a loan if an applicant's income was $78,000?

A. Yes

B. No

Applicant	Income	Gave loan?
#1	$50,005	Yes
#2	$25,004	No
#3	$75,005	Yes
#4	$95,005	Yes
#5	$45,007	No

Classification

Classification

The idea behind classification is that we want to use **patterns** and/or **correlations** to make decisions

Classification happens all the time in real life

- The doctor uses your symptoms and other measurements like weight/blood pressure/etc. to help make a diagnosis
- Google uses classification to determine what an image is

Classification is a **general class** of algorithms.

Classifiers

Classifiers are algorithms that perform classification

They are specific

- e.g. - we don't give loans to anyone an income of less than $50,000 per year

The algorithm you come up with is no different than the other algorithms you've come up with so far

You still need to state the steps you need to take to come up with the solution

Steps to do Classification

Step 1: Start with the data you have

Applicant	Annual Income	Loan Approved?
#1	26 000	No
#2	60 000	Yes
#3	50 000	Yes
#4	47 000	No
#5	12 000	No
#6	108 000	Yes

Step 2: Split data into training and test sets

We chose a 50/50 split for our demo but you could do other splits like 60/40, 70/30. For large datasets, 80/20 split is used

Applicant	Annual Income	Loan Approved?
#1	26 000	No
#2	60 000	Yes
#3	50 000	Yes

} **Training Data**

Applicant	Annual Income	Loan Approved?
#4	47 000	No
#5	12 000	No
#6	108 000	Yes

} **Test Data**

Step 3: Build classifier

(i.e., Find pattern in training set)

Given your training data, can you find a pattern that can tell you when to approve a loan?

Earlier, we decided an annual income of ~$50,000 seemed Like a good cut off point. **That was a classifier!**

Applicant	Annual Income	Loan Approved?
#1	26 000	No
#2	60 000	Yes
#3	50 000	Yes

} **Training Data**

Step 4: Use classifier on test data

Applicant	Annual Income	Loan Approved?
#4	47 000	?
#5	12 000	?
#6	108 000	?

} **Test Data**

After you come up with a classifier that seems to do okay with your training data, you use it on your test data to see what kinds of decisions it makes.

Step 5: Calculate Accuracy

Applicant	Annual Income	Loan Approved?	Classifier said to…
#4	47 000	No	No
#5	12 000	No	No
#6	108 000	Yes	Yes

} **Test Data**

If the results of your classifier match up with the decisions you've made in your test data, it's looking good.

You can start trying to use it on data that you haven't made any decisions on yet.

Seems Straightforward

- What happens when we have more than one attribute?
- In the example before, we only had to consider annual income
- But what would happen if we had multiple attributes, like 5 or 10 or 100?
- **How do we decide which attribute to use?**

Seems Straightforward

- What happens when we have more than one attribute?
- In the example before, we only had to consider annual income
- But what would happen if we had multiple attributes, like 5 or 10 or 100?
- **How do we decide which attribute to use?**

Decision Trees!

Trees & Decisions Trees

Regular Trees

Trees in Computer Science

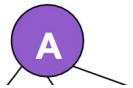
Regular Trees

Trees in Computer Science

- A Decision Tree is a way for a computer to make decisions based on a series of questions.

Trees in Computer Science

- A Decision Tree is a way for a computer to make decisions based on a series of questions.



A **tree** is a **collection of nodes** such that

- One node is the designated ***root***.

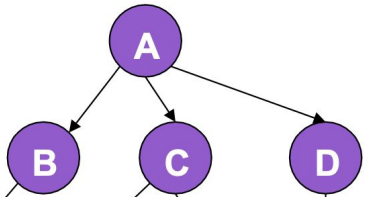
A is the ***root***

Trees in Computer Science

- A Decision Tree is a way for a computer to make decisions based on a series of questions.

A **tree** is a **collection of nodes** such that

- One node is the designated ***root***.
- A node can have zero or more *children*;



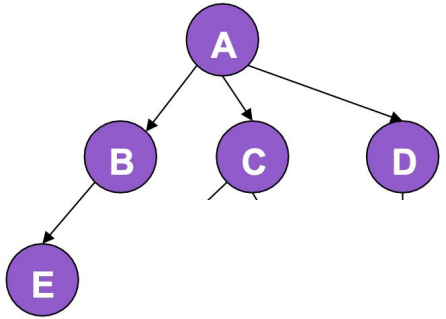
B, C and D are A's children

Trees in Computer Science

- A Decision Tree is a way for a computer to make decisions based on a series of questions.

A **tree** is a **collection of nodes** such that

- One node is the designated ***root***.
- A node can have zero or more *children*;
- a node with zero children is a *leaf*.



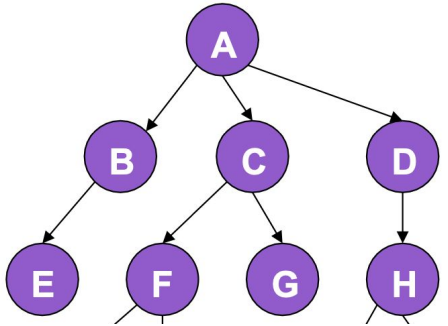
E is a leaf

Trees in Computer Science

- A Decision Tree is a way for a computer to make decisions based on a series of questions.

A **tree** is a **collection of nodes** such that

- One node is the designated ***root***.
- A node can have zero or more *children*;
- a node with zero children is a *leaf*.
- All non-root nodes have a single *parent*.



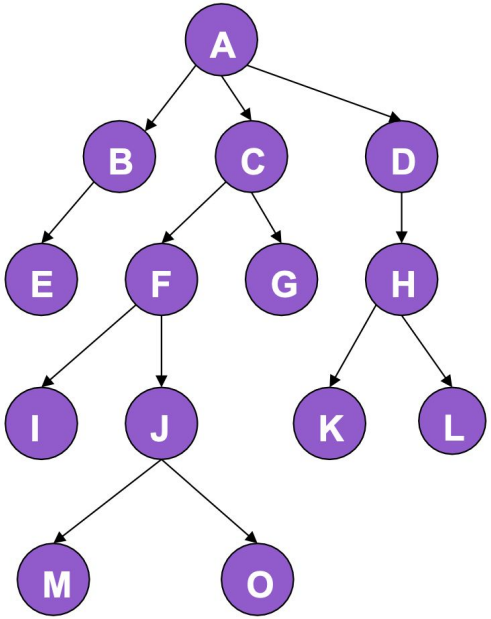
C is a parent to F + G

Trees in Computer Science

- A Decision Tree is a way for a computer to make decisions based on a series of questions.

A **tree** is a **collection of nodes** such that

- One node is the designated ***root***.
- A node can have zero or more *children*;
- a node with zero children is a *leaf*.
- All non-root nodes have a single *parent*.
- *Edges* denote parent-child relationships.
  - Example: The arrows between F → I

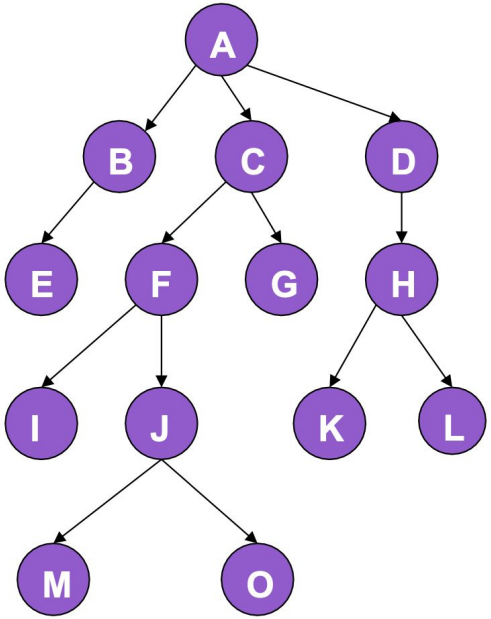


Trees in Computer Science

- A Decision Tree is a way for a computer to make decisions based on a series of questions.

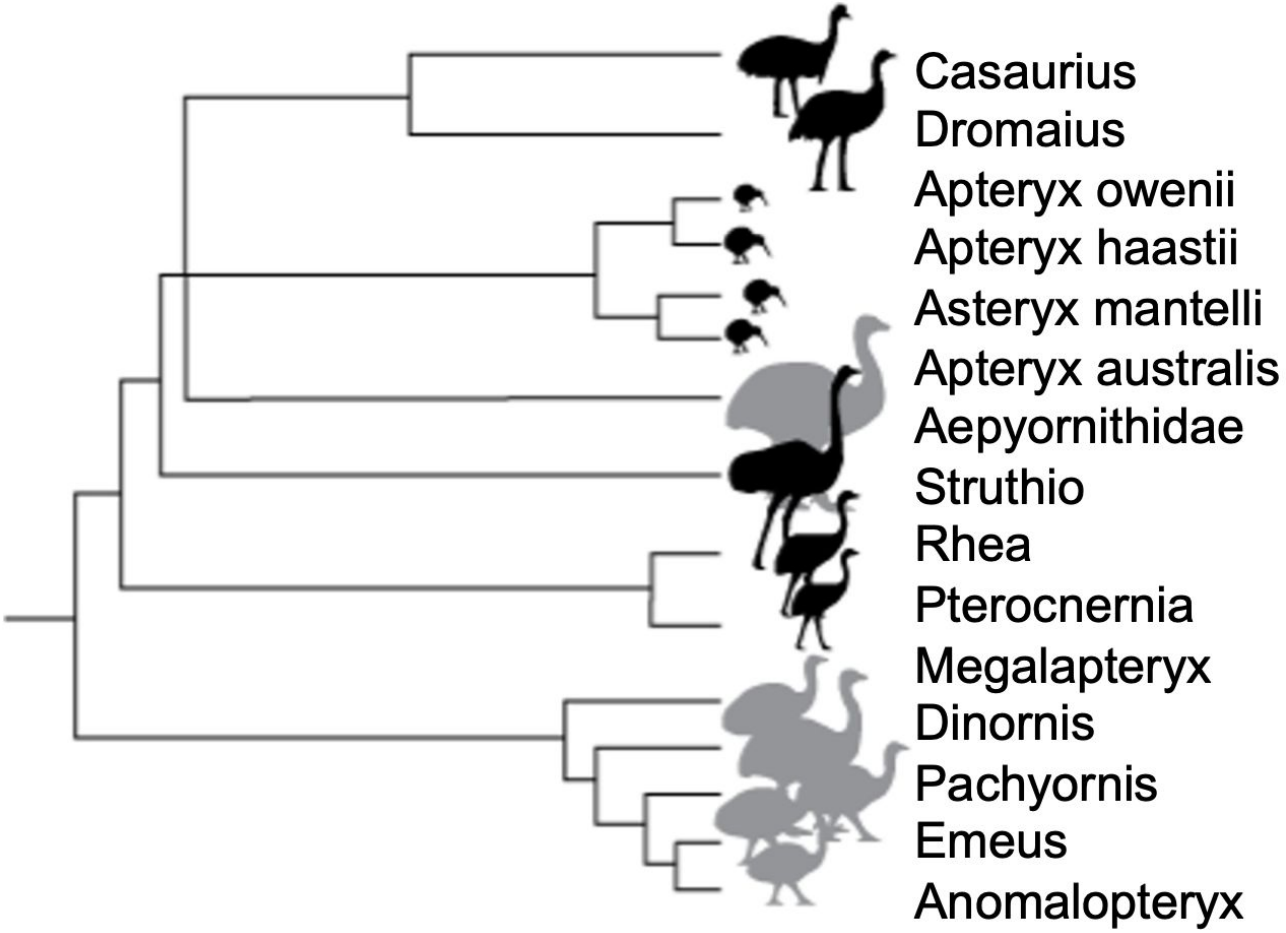
A **tree** is a **collection of nodes** such that

- One node is the designated ***root***.
- A node can have zero or more *children*;
- a node with zero children is a *leaf*.
- All non-root nodes have a single *parent*.
- *Edges* denote parent-child relationships.
- Nodes and/or edges may be labeled by data.
  - Each node on this tree is labeled by a letter



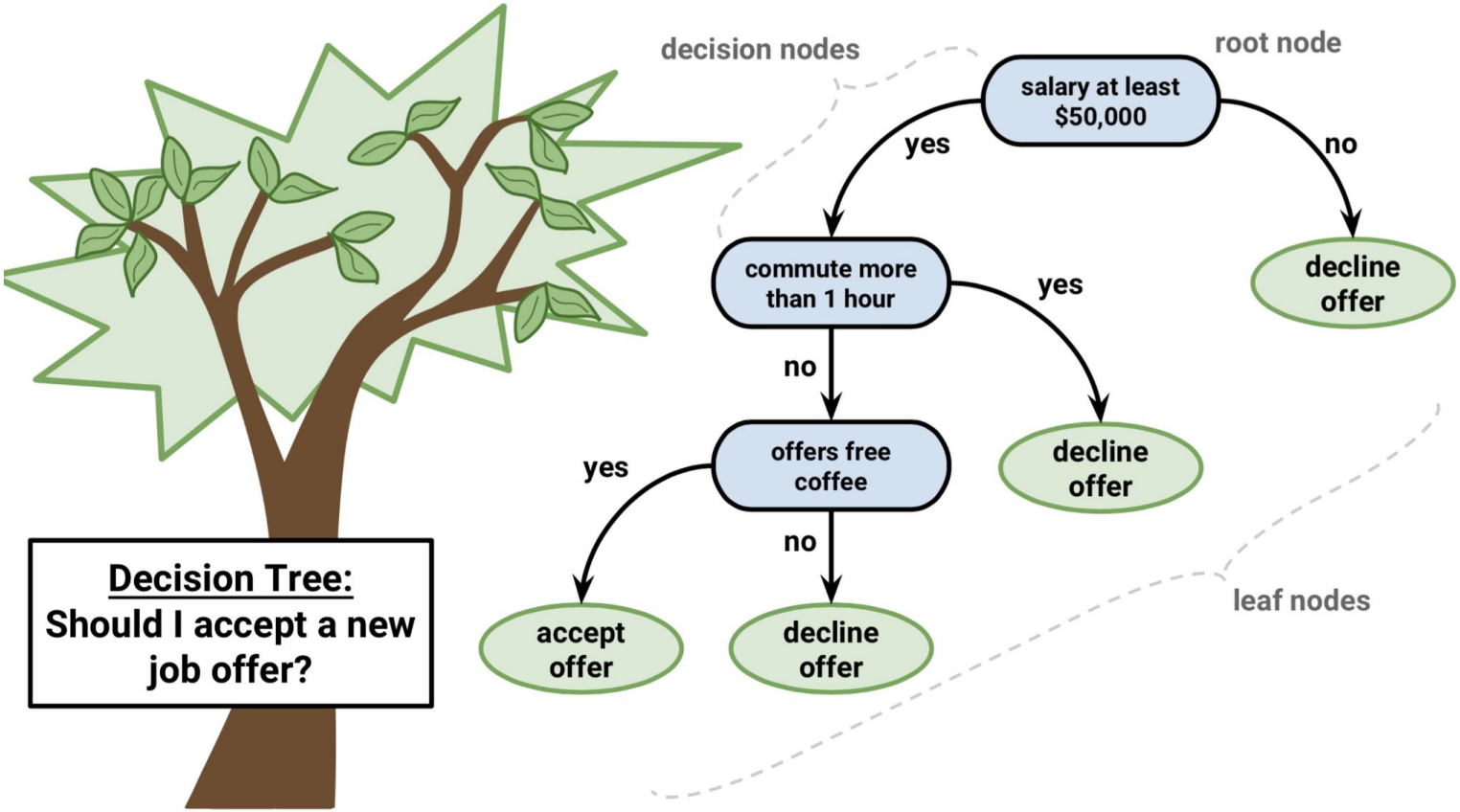
Non-CS Decisions Trees

Rooted trees in CS often (but not always) drawn with root on top



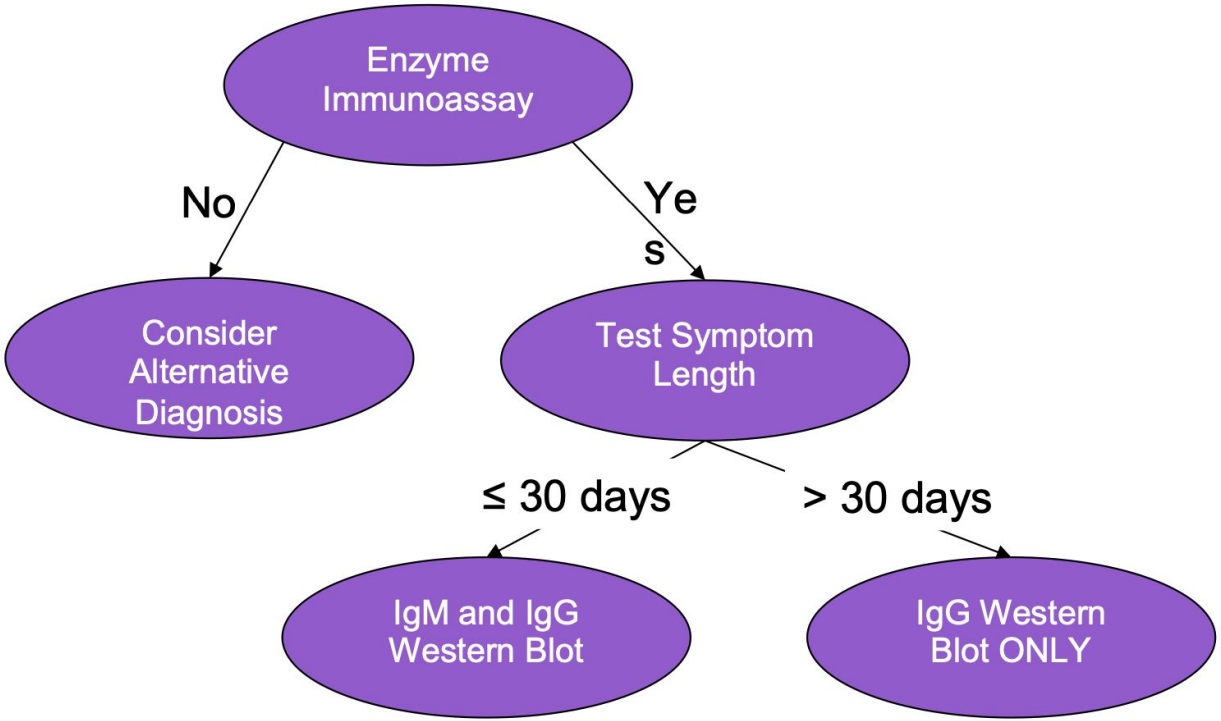
Decision trees

Trees whose node labels are attributes, edge labels are conditions



Decisions Trees in Medicine

Decision tree for Lyme Disease diagnosis



Decisions Trees in Business

GBR Graziadio Business Review

A Peer-Reviewed Journal of Relevant Information and Analysis

Home | Archives | About | Submissions | Blog

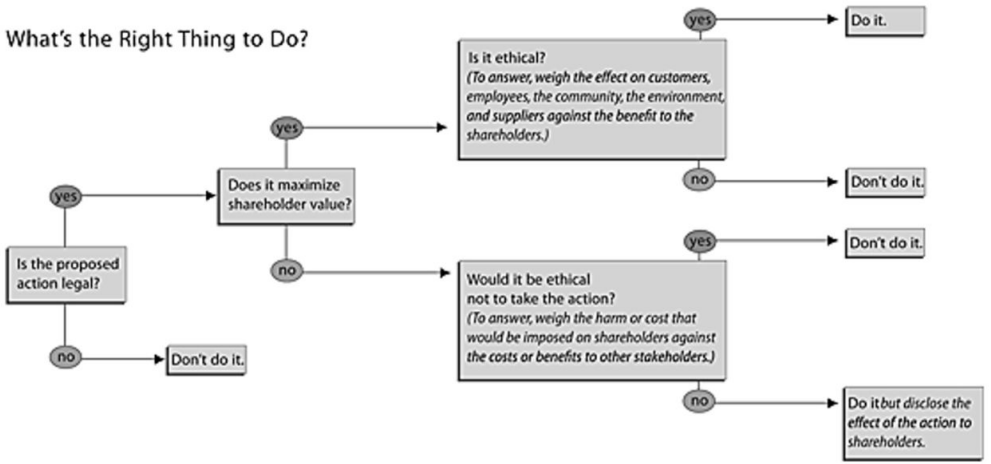
How Gerber Used a Decision Tree in Strategic Decision-Making

Possible outcomes explored in an in
Products Safety Commission.

By JAY BUCKLEY and THOMAS J. DUDLEY, DBA

1999 Volume 2 Issue 3

Decision trees can assist executives in making strat



Let's build a Decision Tree

Building Decision Trees

- Should you get an ice cream?
- You might start out with the following data

Weather	Wallet	Ice Cream?
Great	Empty	No
Nasty	Empty	No
Great	Full	Yes
Okay	Full	Yes
Nasty	Full	No

Building Decision Trees

- Should you get an ice cream?
- You might start out with the following data

Attributes

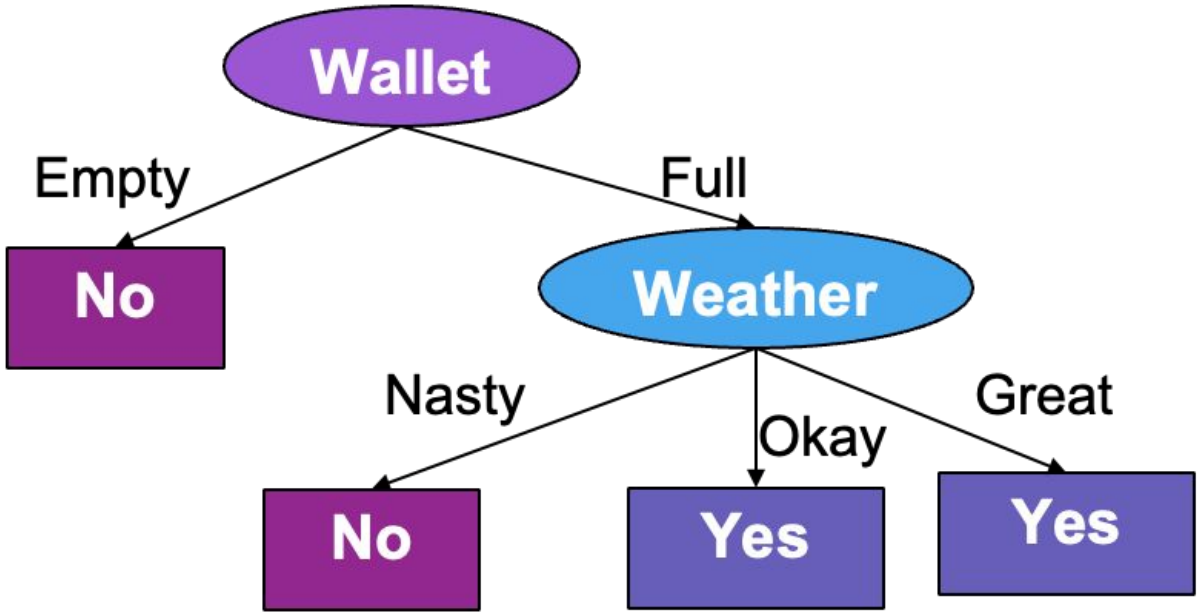
Conditions

Weather	Wallet	Ice Cream?
Great	Empty	No
Nasty	Empty	No
Great	Full	Yes
Okay	Full	Yes
Nasty	Full	No

Ice Cream Decision Tree

Should you get an ice cream?

Weather	Wallet	Ice Cream?
Great	Empty	No
Nasty	Empty	No
Great	Full	Yes
Okay	Full	Yes
Nasty	Full	No



Activity

In-class Activity: Decision Trees

Draw the decision tree for the following dataset:

Understanding the risk to prevent a heart attack

Age	Weight Range (kg)	Risk
< 18	≤ 50	Low Risk
< 18	> 50	High Risk
18–30	≤ 70	Low Risk
18–30	> 70	Low Risk
> 30	≤ 80	Low Risk
> 30	> 80	High Risk

Wrap up

Wrap Up

- **Lab #2**
  - Due on Friday, Jan 24 at 11:59pm
- **Post-Class (PC) Quiz #1**
  - Only 1 attempt, 60 minutes
  - Open book, open-AI* (*you must disclosure your usage)
  - Due on Sunday, Jan 26 at 11:59pm
- **Group Contracts**
  - Extended to Monday, Jan 27 at 11:59pm
- **PC Quiz #2**
  - To be released next week!