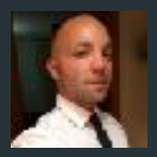
Why every developer should be a data scientist

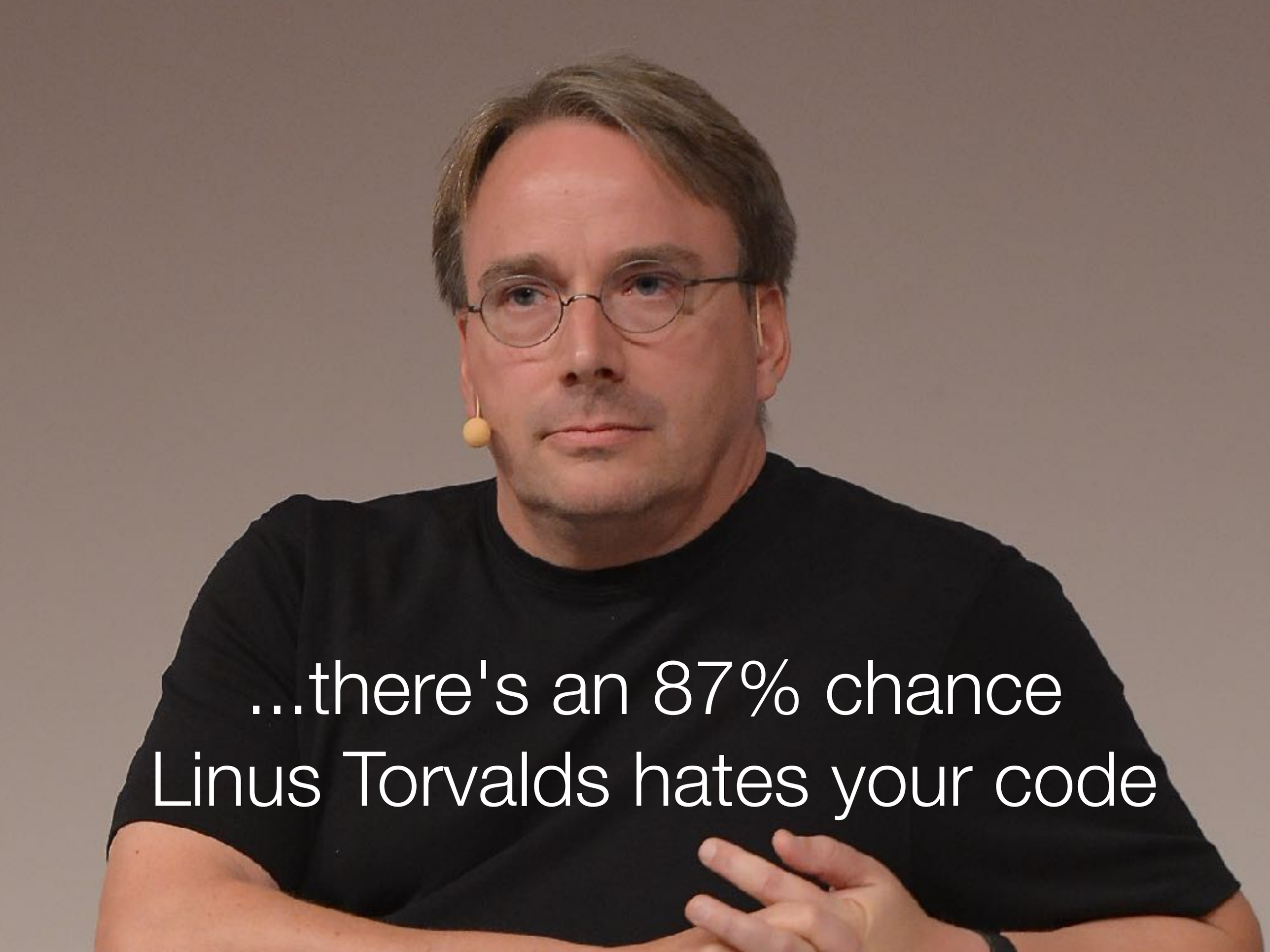


Ivan Danyliuk
Software Developer @ Status.im

GDG DevFest Ukraine 2017

The recent study from MIT has found...

#dfua


...there's an 87% chance
Linus Torvalds hates your code

"Bad programmers **worry about the code**.
Good programmers **worry about data structures** and their relationships."

"Show me your [code] and conceal your [data structures], and **I shall continue to be mystified**.

Show me your [data structures], and I won't usually need your [code]; **it'll be obvious**."

Fred Brooks

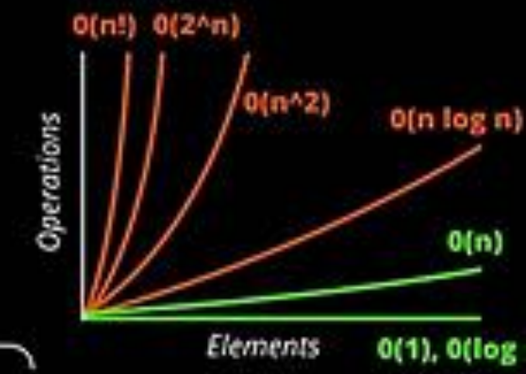
design programs around the data

LEGEND

TIME Complexity vs. **SPACE** Complexity

Good Fair Bad

Good Fair Bad



<BIG-O-CHEATSHEET>

</>

www.bigocheatsheet.com

DATA STRUCTURE Operations

DATA Structure | TIME Complexity | SPACE Complexity

TIME Complexity: Average | Worst

Average: Access | Search | Insertion | Deletion

Worst: Access | Search | Insertion | Deletion

Data Structure	Time Complexity (Average)				Time Complexity (Worst)				Space Complexity (Worst)
Array	O(1)	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	N/A	O(1)	O(1)	O(1)	N/A	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	N/A	O(log(n))	O(log(n))	O(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
B-Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	N/A	O(log(n))	O(log(n))	O(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
KD Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

ARRAY SORTING Algorithms

ARRAY Algorithms | TIME Complexity | SPACE Complexity

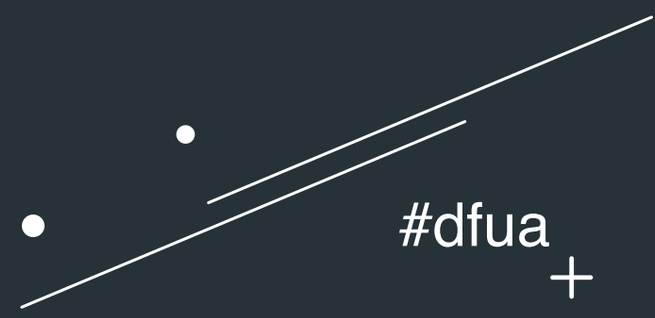
TIME Complexity: Best | Average | Worst

Array Algorithms	Time Complexity (Best)	Time Complexity (Average)	Time Complexity (Worst)	Space Complexity (Worst)
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(1)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)[1]	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)

Question

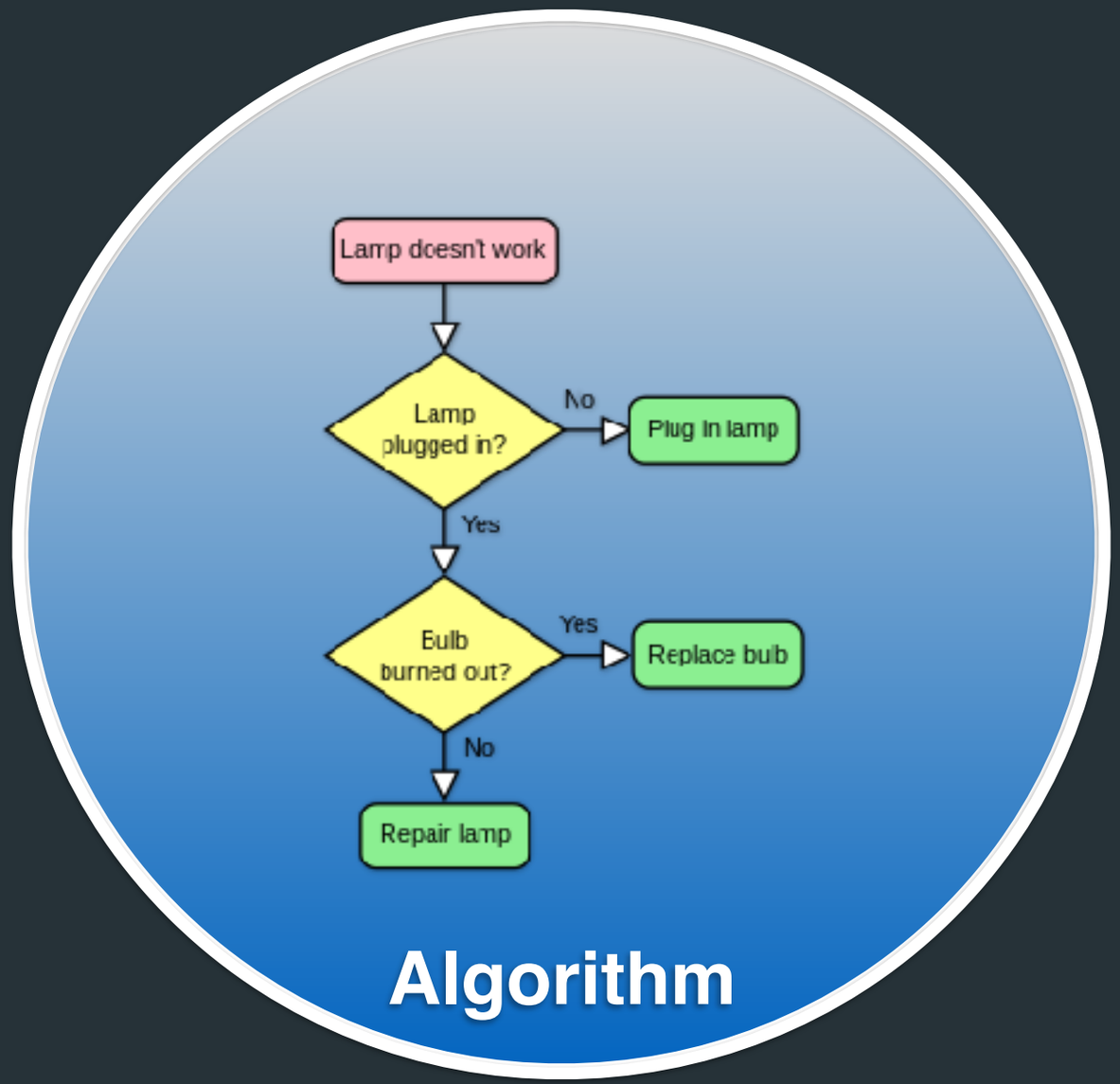
- Raise your hand if you have designed your own non-standard data-structure recently?
- Raise your hand if you have implemented your own custom algorithm recently?
- Now, raise your hand if you have written a new microservice recently?

Our "programs" now are distributed systems.



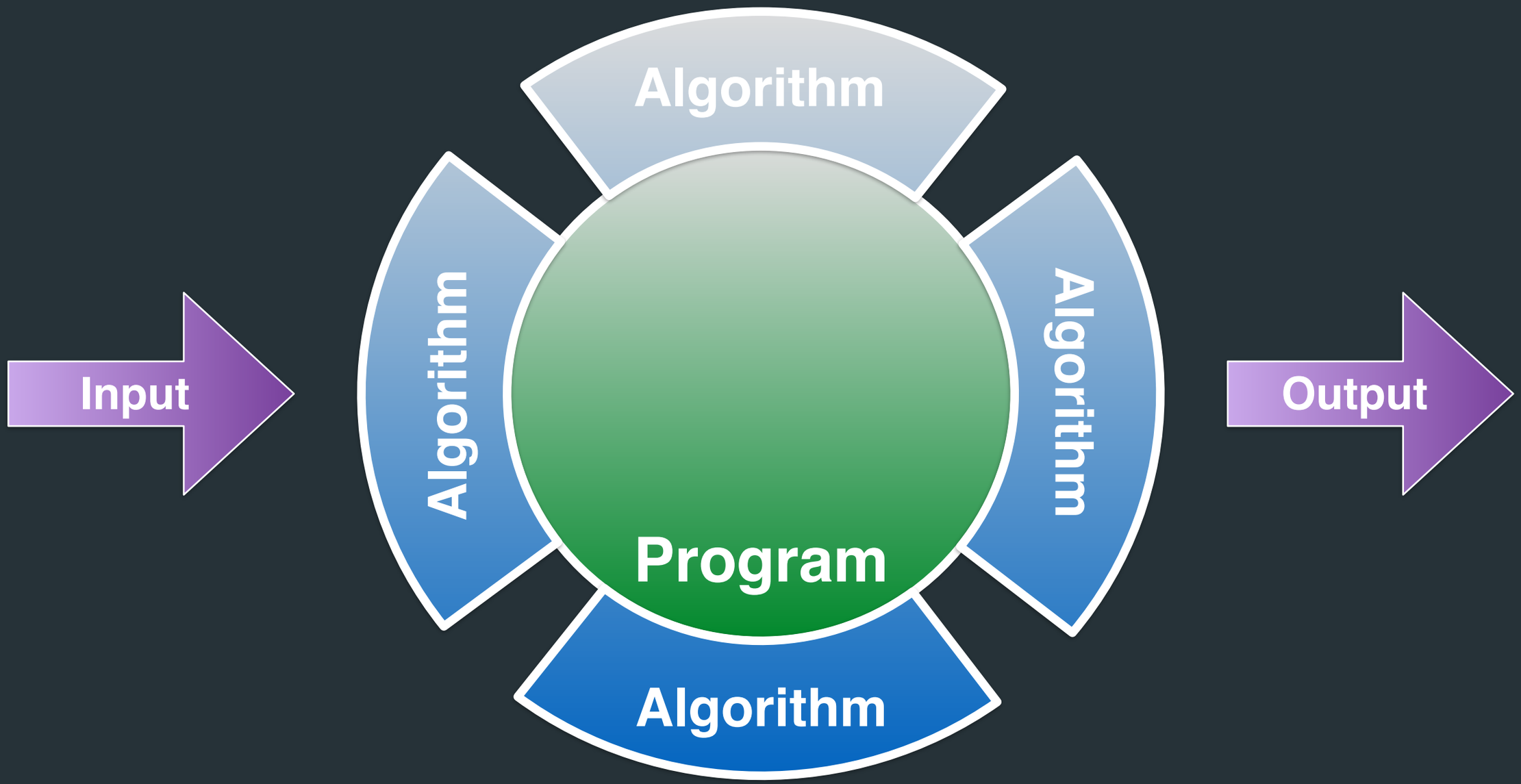
Our "hardware" is a cloud.

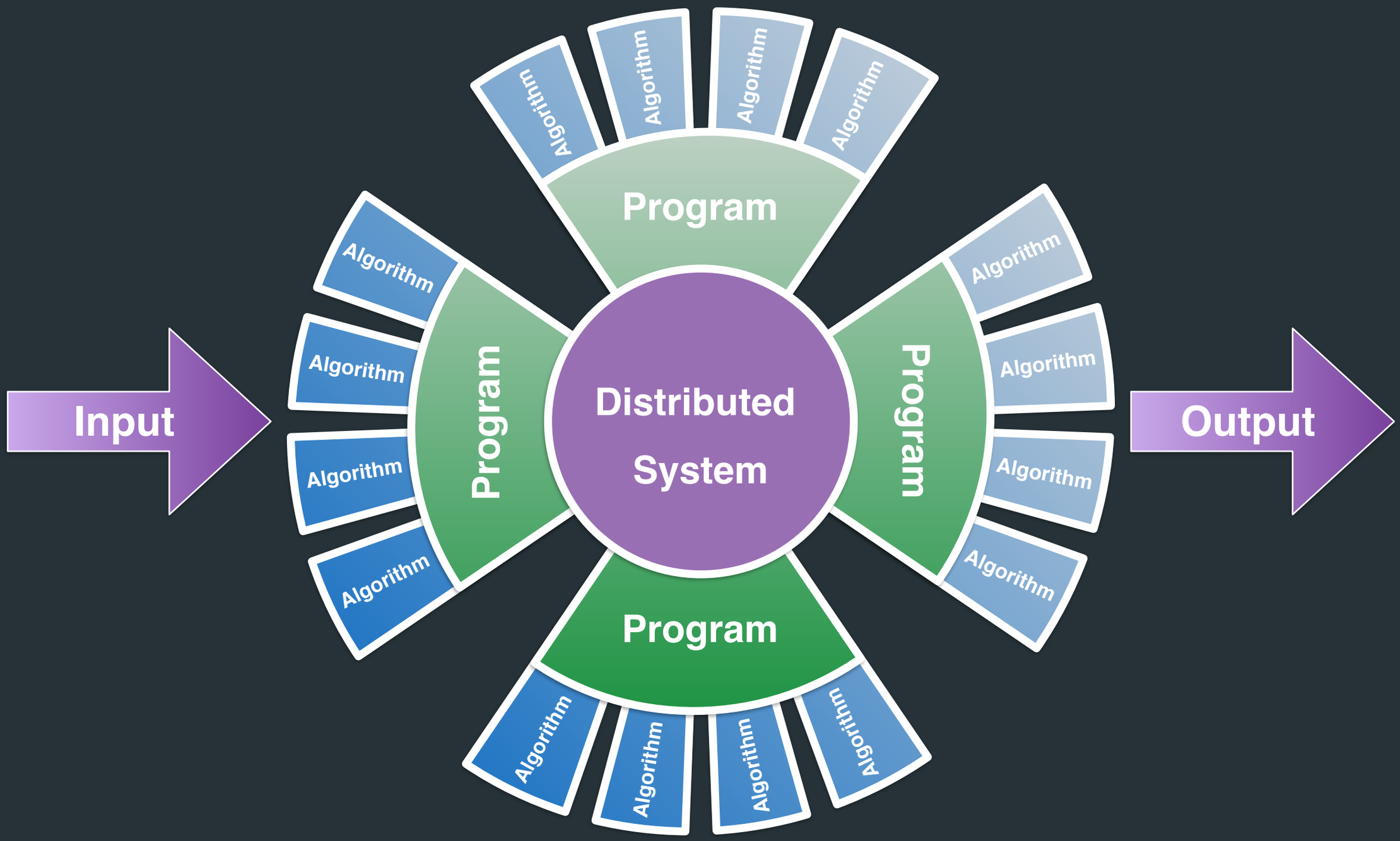
Input



Algorithm

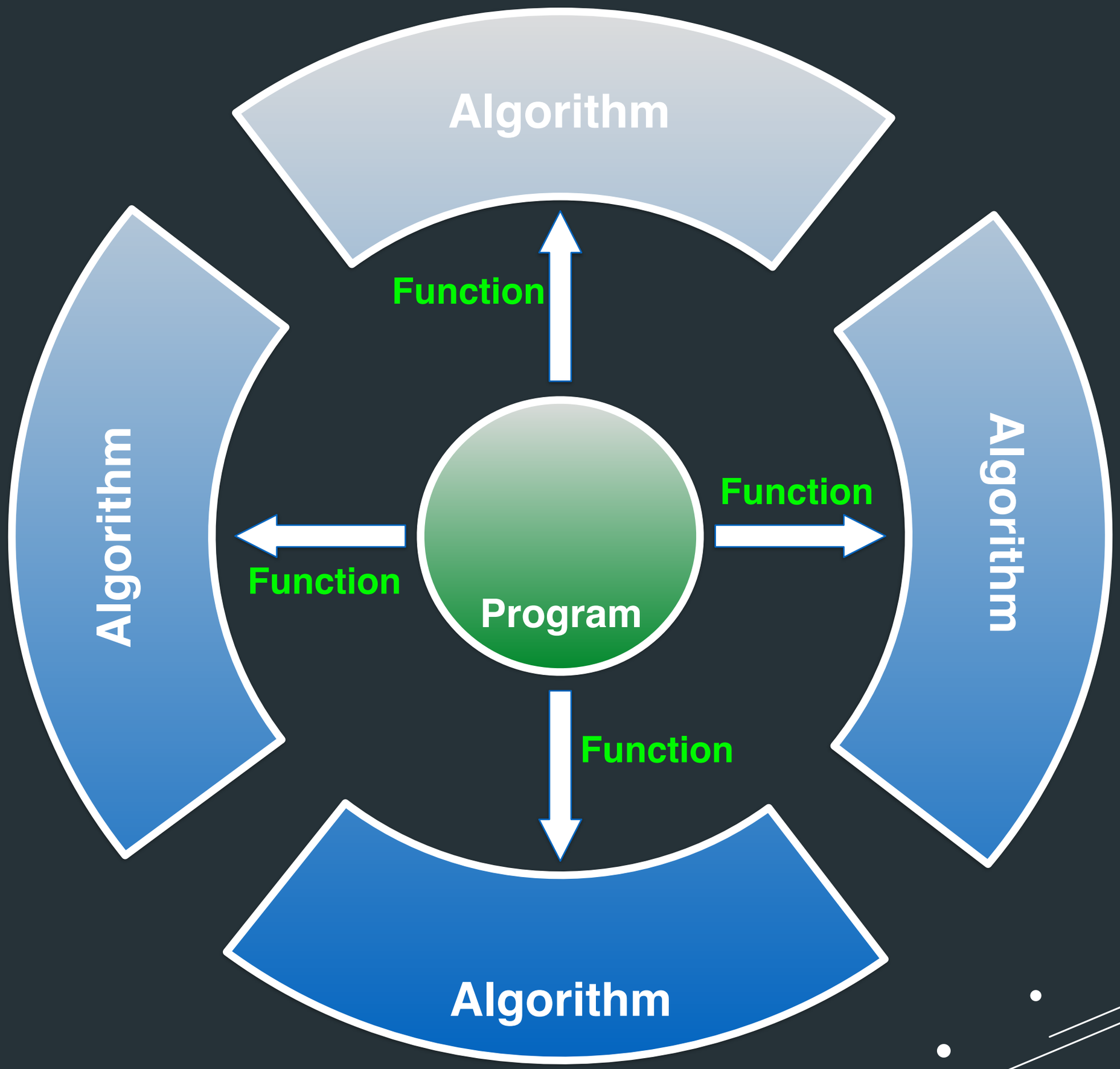
Output

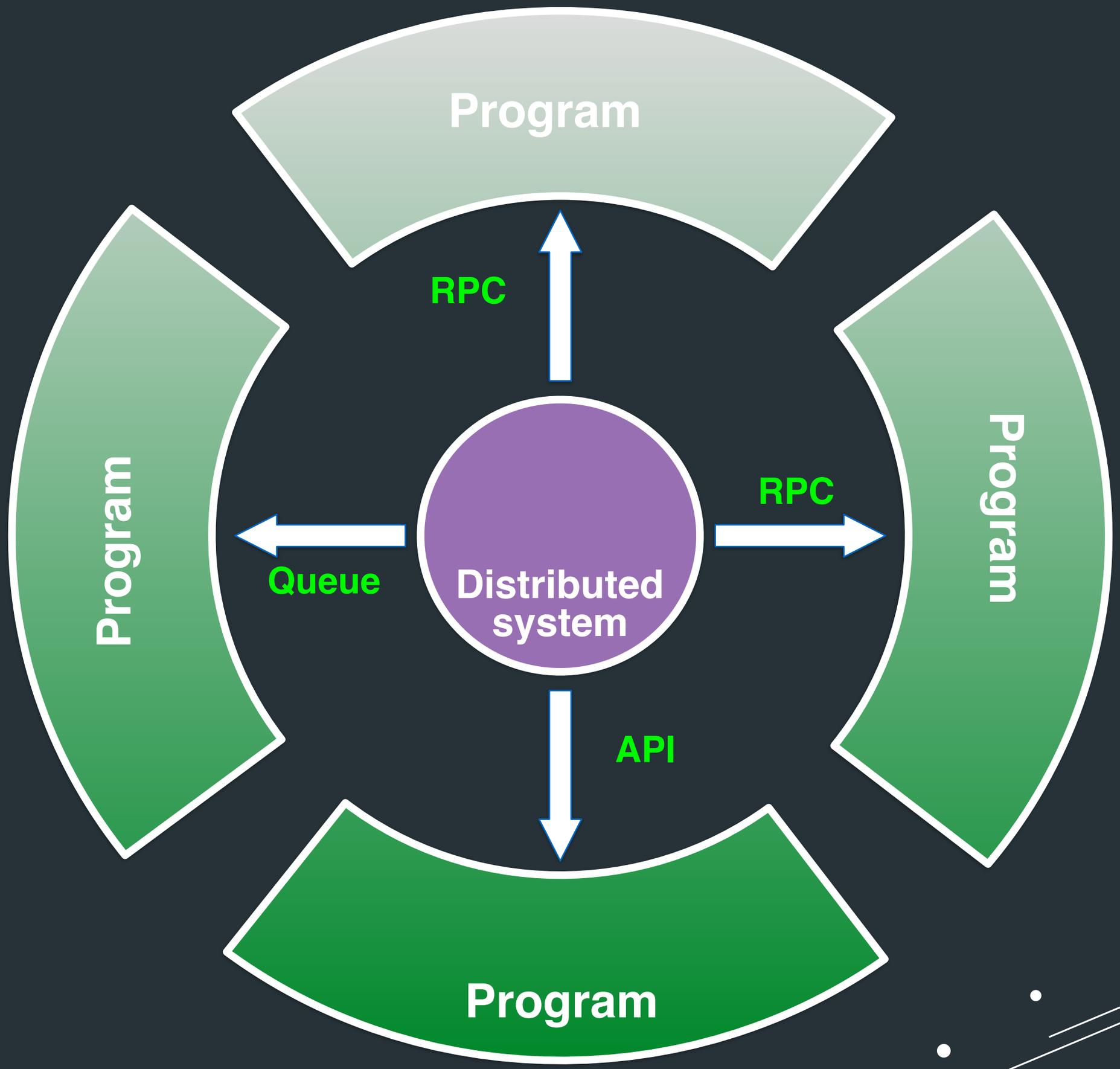

Algorithm
Algorithm
Algorithm
Input
Program
Output


Input
Output
Distributed System
Program
Program
Program
Program
Algorithm
Algorithm
Algorithm
Algorithm
Algorithm
Algorithm
Algorithm
Algorithm
Algorithm
Algorithm
Algorithm
Algorithm
Algorithm
Algorithm
Algorithm
Algorithm

It's just different scale.

You use RPC calls instead of functions.


Algorithm
Function
Algorithm
Function
Function
Program
Algorithm
Function
Algorithm


Program
RPC
Program
RPC
Queue
Distributed system
Program
API
Program

- Inlining function vs external function call

- Local code vs external RPC call

Imagine network call is as cheap as a local function call - what's the difference then?

Algorithmia is actually already doing that

TEXT ANALYSIS
Make sense of unstructured text

MACHINE LEARNING
Teach your app to teach itself

COMPUTER VISION
Identify objects in images

DEEP LEARNING
Learn from your data

Utilities · Microservices · Web Tools · Time Series

BROWSE ALL ALGORITHMS:

★ Top Rated | Most Called | Recently Added

Colorful Image Colorization
deeplearning
Colorizes given black & white images.

Summarizer
nlp
Summarize english text

Sentiment Analysis
nlp
Determine positive or negative sentiment from text

AutoTag
nlp
Automatically extract tags from text

```go
package main

import (
  "fmt"
  algorithmia "github.com/algorithmiaio/algorithmia-go"
)

func main() {
  input := 1429593869

  var client = algorithmia.NewClient("YOUR_API_KEY", "")
  algo, _ := client.Algo("algo://ovi_mihai/TimestampToDate/0.1.0")
  resp, _ := algo.Pipe(input)
  response := resp.(*algorithmia.AlgoResponse)
  fmt.Println(response.Result)
}
```

design programs around the data

algorithms

design programs around the data

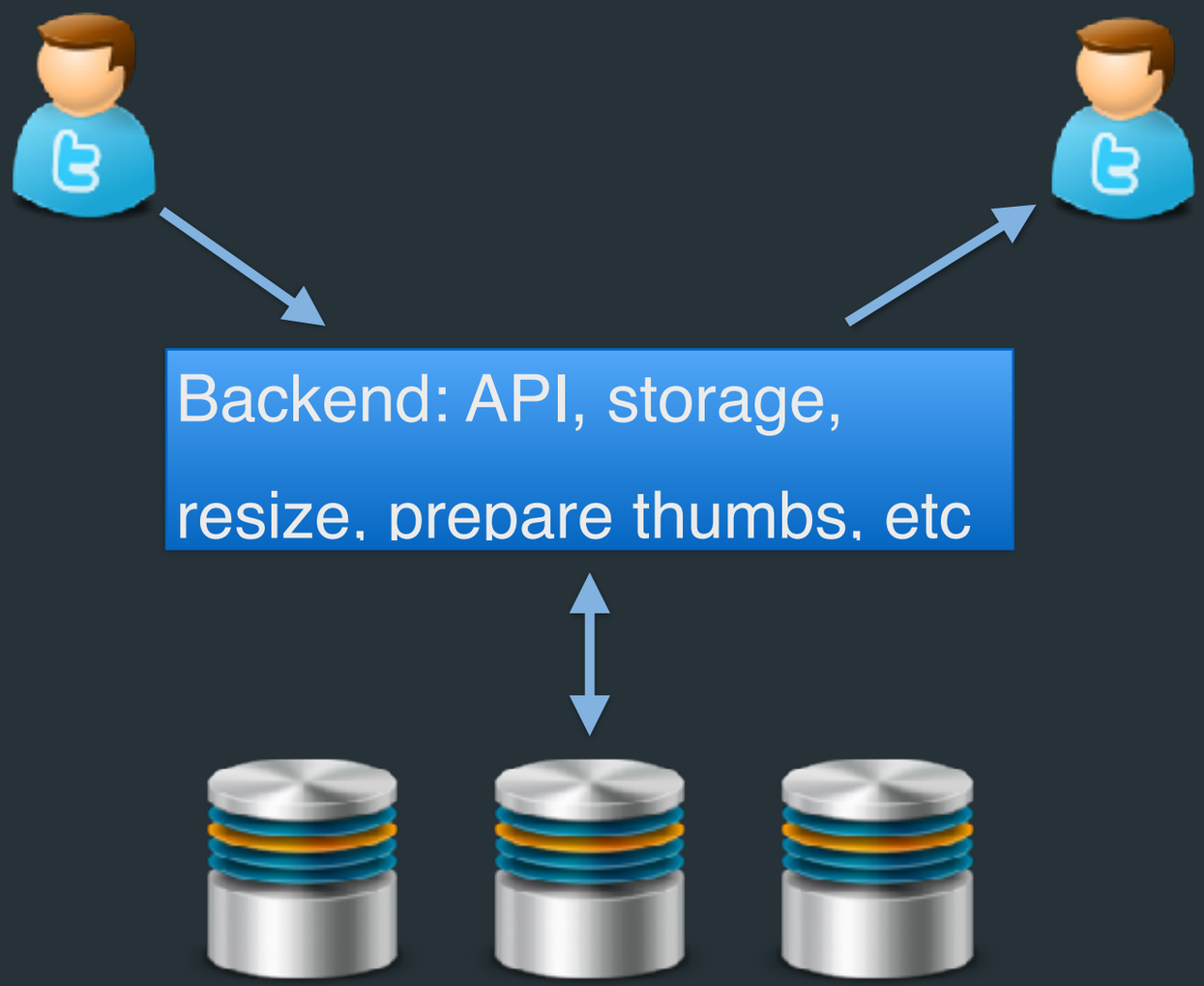
systems

Examples

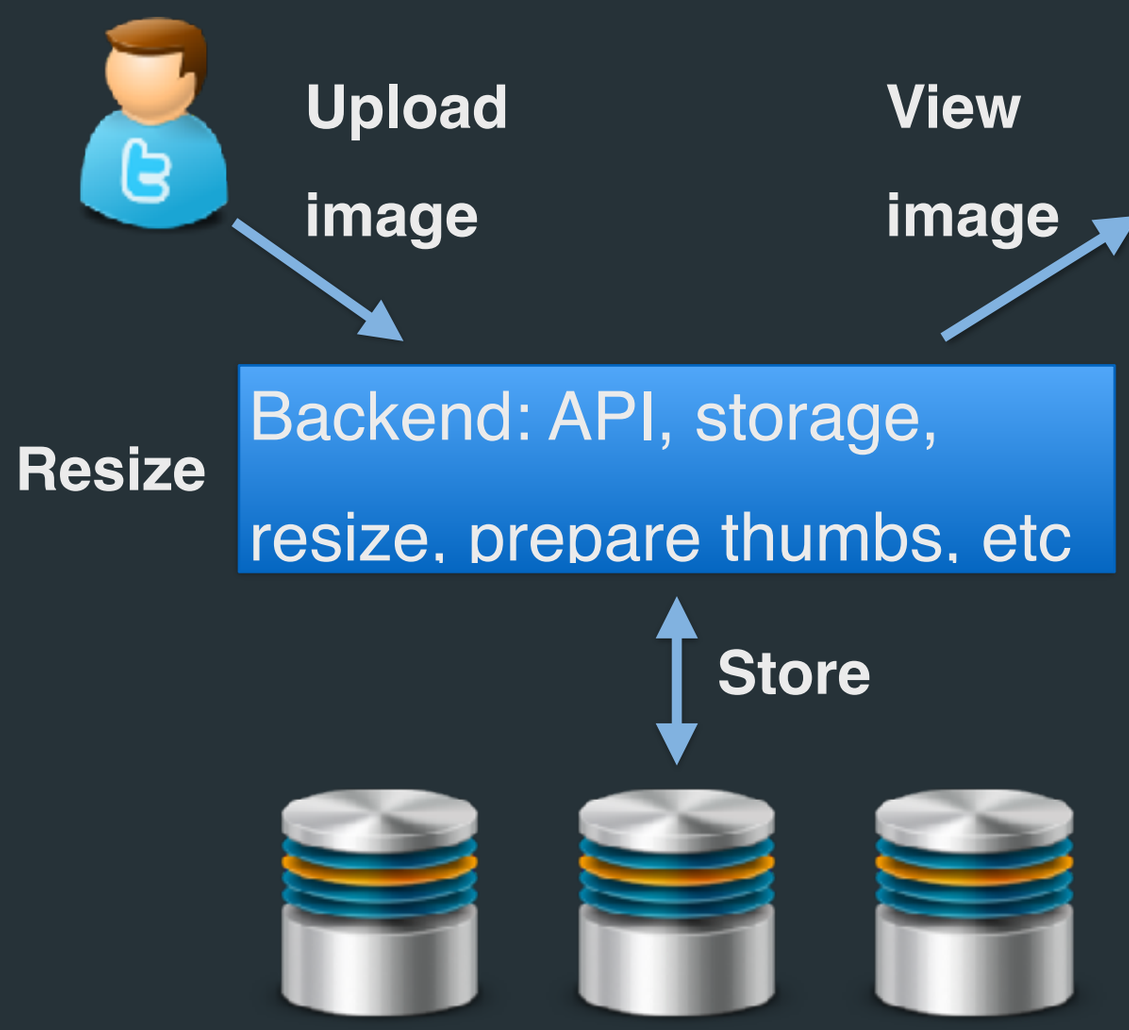
Twitter story

How Twitter **refactored** it's media storage system

Twitter in 2012

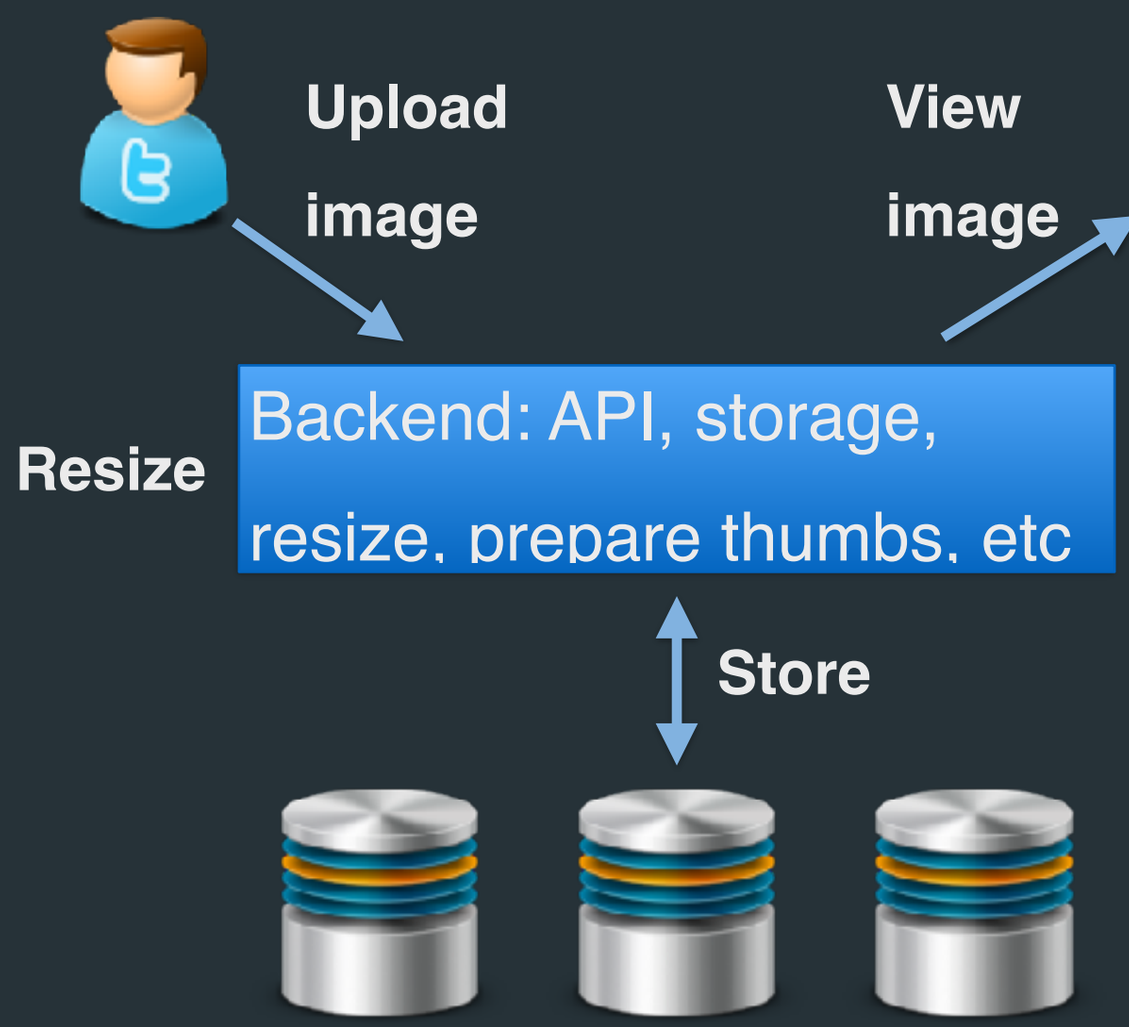


Twitter in 2012



- Handle user upload
- Create thumbnails and different size versions
- Store images
- Return on user request (view)

Twitter in 2012



- Handle user upload
- Create thumbnails and different size versions
- Store images
- Return on user request (view)

The Problem:

- a lot of storage space
- + 6TB per day

Twitter

Twitter did a research on the data and found interesting access patterns

Twitter

- 50% of requested images are at most 15 days old
- After 20 days, probability of image being accessed is negligibly low

Twitter in 2016

- Created server that can resize on the fly
- Slow, but it's a good space-time tradeoff.

Twitter in 2016

- Image variants kept only 20 days.
- Images older than 20 days resized on the fly.

Twitter in 2016

- Storage usage **dropped by 4TB per day**
- Twice as less of computing power
- Saved **$6 million** in 2015

One of my former employers story

How to ignore data

The Problem

- Similar to twitter, they had users writing and reading stuff
- Stuff had to be filtered and searched

The Problem

- It became slow as the data grew
- DB outages were for 2-3 days(!)

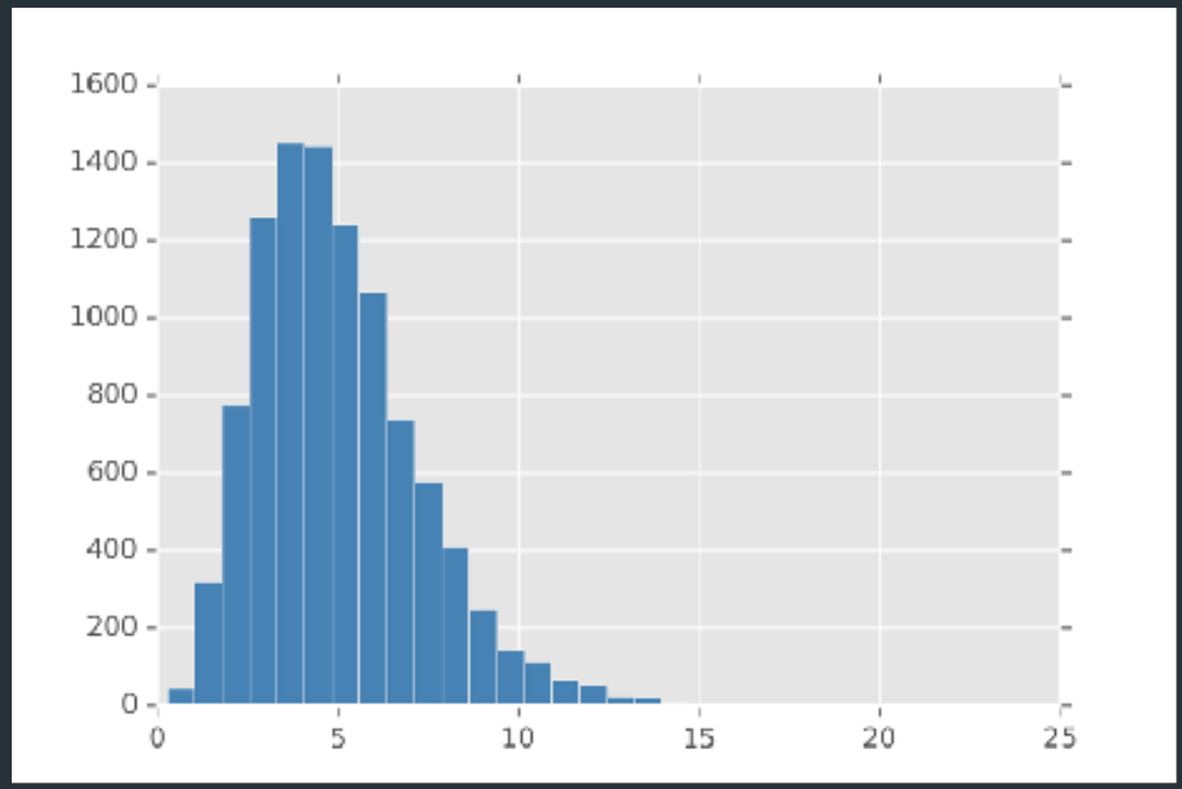
We made a huge data research...

#dfua

...and found two things:

- access pattern was similar to Twitter's one — most of the data was a "dead weight" after two weeks

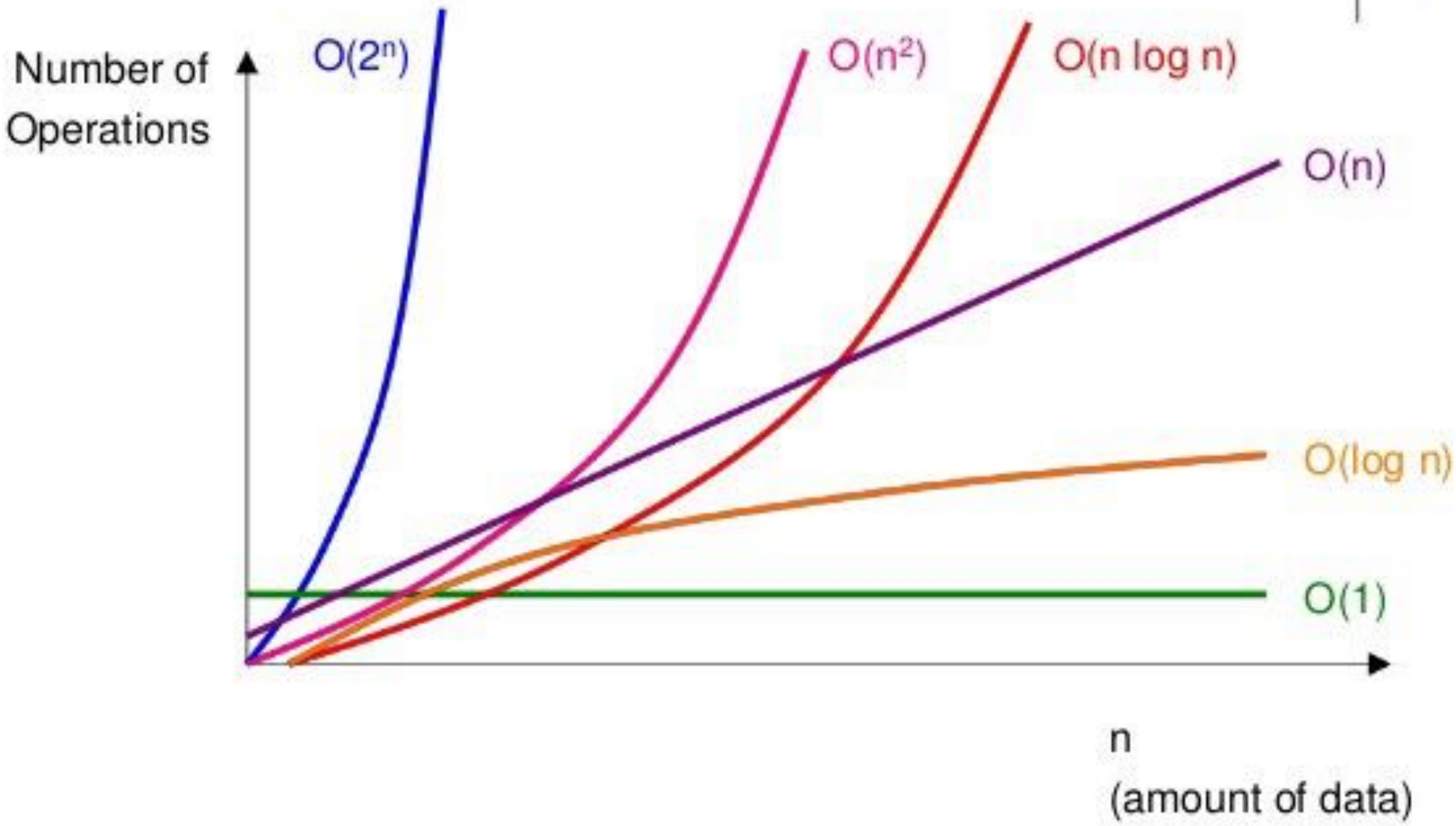
- data was isolated and most (90%) of the data was really small — 10x10 table of strings



Property of small numbers

- Everything is fast for the small "N"
- Linear search can outperform binary search if N is small

Comparing Big O Functions



Company decided to solve problem...

...by switching to another DB

MySQL → ElasticSearch

"We have a problem with DB on search...

...let's switch to another DB with 'search' word in its name."

When you ignore the data — it's not a software engineering anymore

Ravelin story

How Ravelin designed fraud detection system

Ravelin

- Ravelin does fraud detection for financial sector
- Clients make an API call to check if they allow order to proceed
- So the latency is critical here.

Ravelin

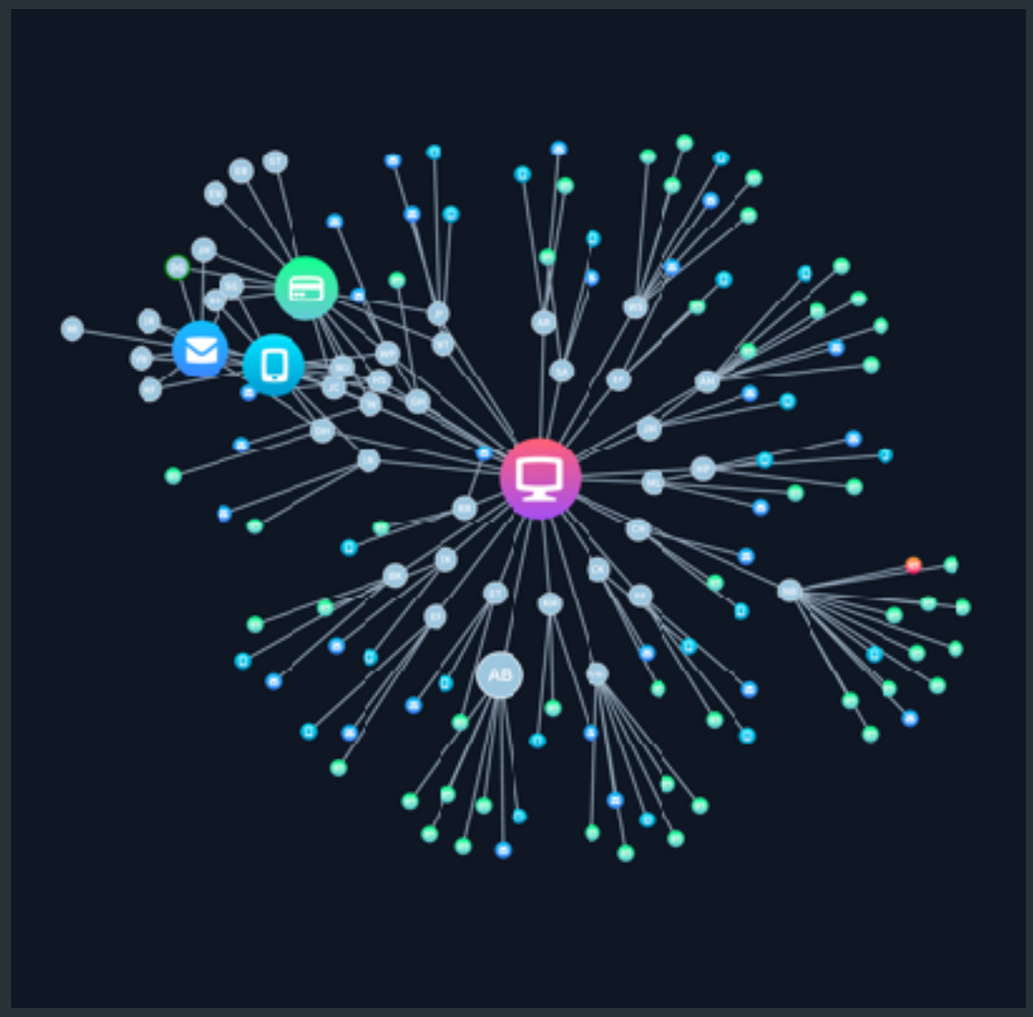
- They use machine learning for that
- For the machine learning they need data
- Data is a different features

Ravelin

- But there are complex features

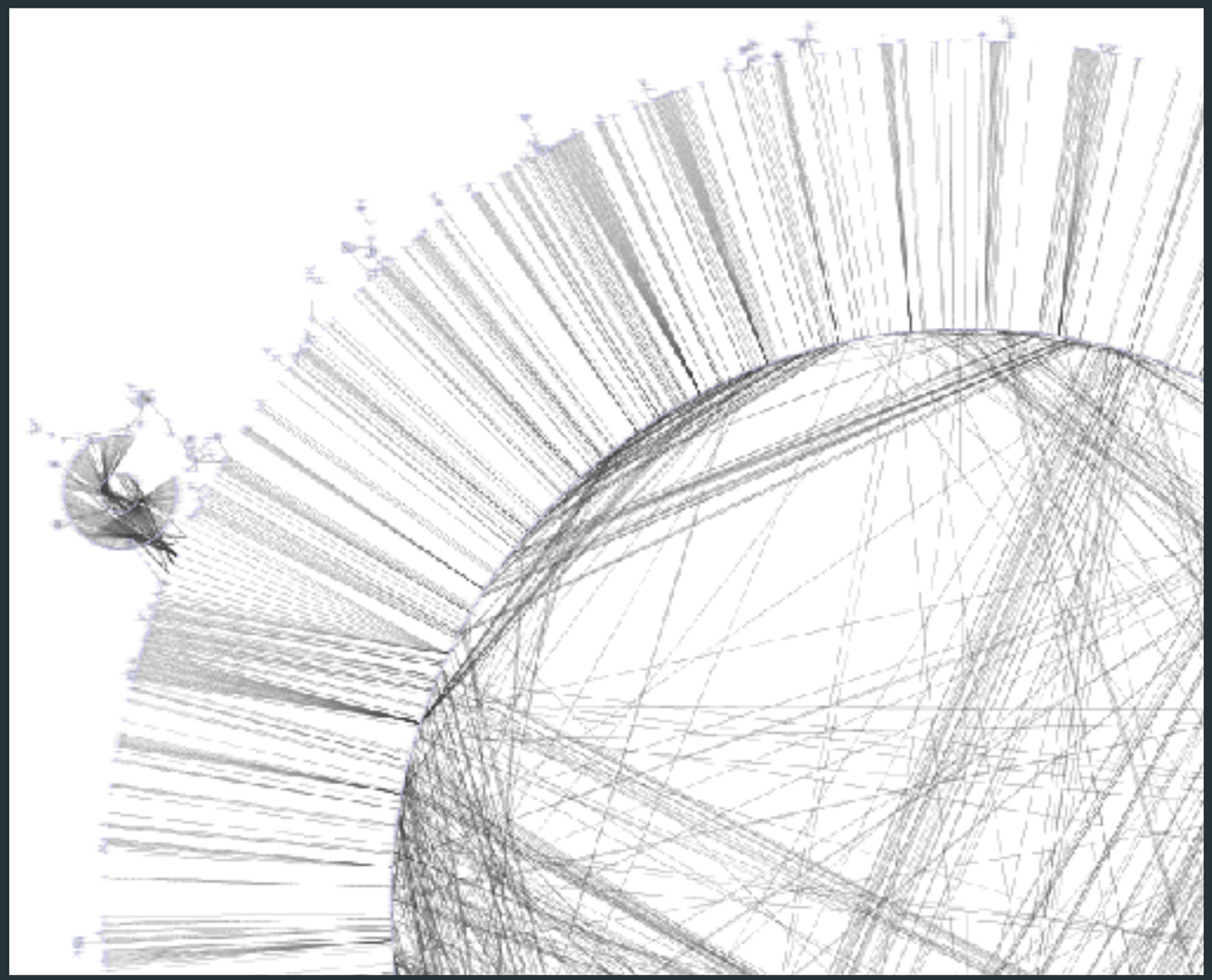
- They need to connect things like phone numbers, credit cards, emails, devices and vouchers

- So new people could be easily connected to known "fraudsters" with very little data.



Ravelin

- They studied the problem offline
- It was clear they need a graph database



Ravelin

So, they looked at major players in Graph databases world...

Cayley

Titan

Neo4j

...but were not happy with any of them.

Ravelin

They returned to the drawing board and asked the question:

"What data do we actually need?"

Ravelin

- "What we care about is the **number of people that are connected** to you."
- "And if any of those people are known fraudsters."

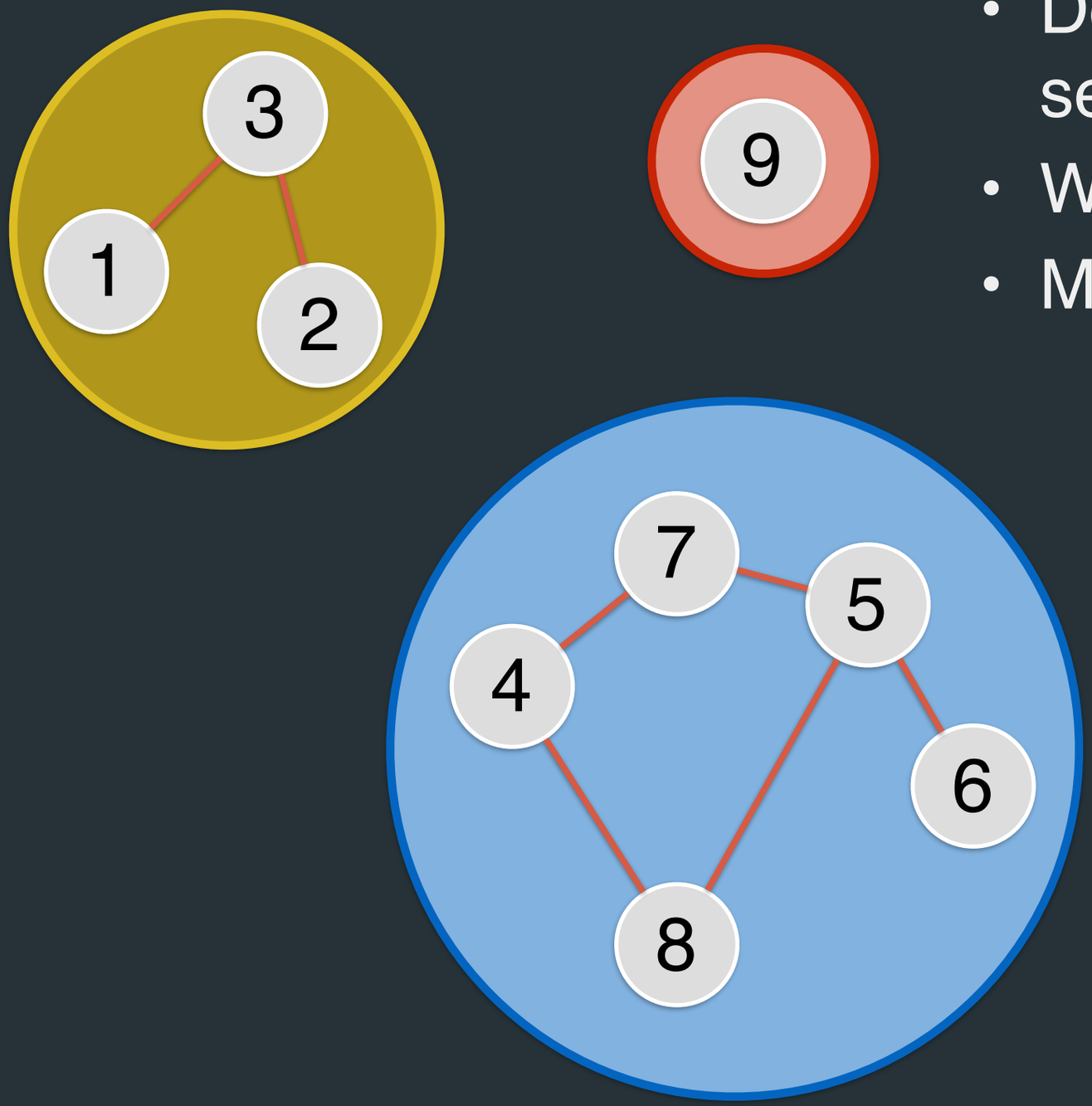
Ravelin

So they come up with the solution by using Union Find (disjoint-set) data structure

It allows you to very quickly:

- find items (and sets they are in)
- join sets

Union Find



Can very quickly do things as:

- Does 2 belong to the same set as 7?
- What set 3 is in?
- Merge subsets with 9 and 7

Union Find

- Really low memory footprint
- Crazy fast:
    - CreateSet - O(1)
    - FindSet - O(α(n))* (worst case)
    - MergeSet - O(α(n))* (worst case)
- Visualization: https://visualgo.net/en/ufds

** α(n) - is an inverse Ackerman function, grows slower than log(n)*

Union Find

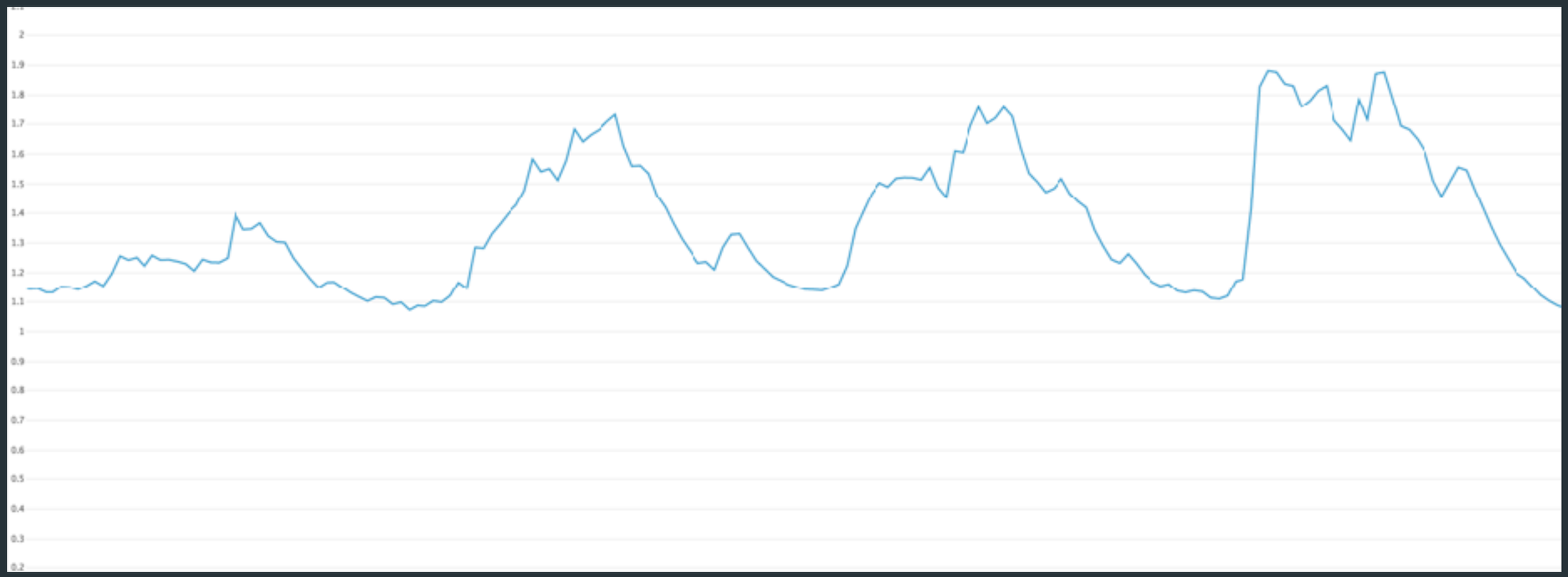
```
type Node struct {
    Count  int32
    Parent string
}

type UnionFind struct {
    Nodes map[string]*Node
}
```

```
func (uf *UnionFind) Add(a, b string) int32 {
   first, second := uf.parentNodeOrNew(a), uf.parentNodeOrNew(b)

    var parent *Node
    if first.Parent == second.Parent {
       return first.Count
    } else if first.Count > second.Count {
       parent = uf.setParent(first, second)
    } else {
       parent = uf.setParent(second, first)
    }
   return parent.Count
}
```

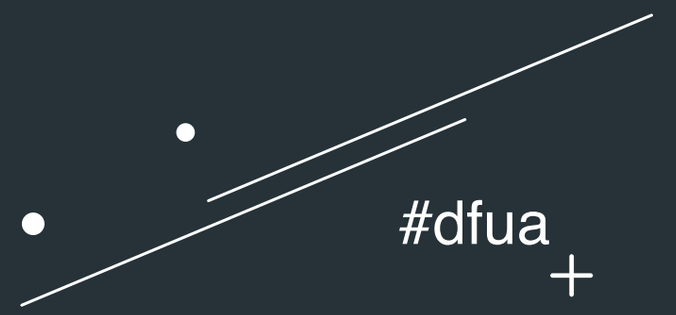
95th is under 2ms, average is closer to 1ms



- Just by analyzing the data they really needed, they simplified system drastically.
- Neo4J is more than 1 million Lines of Code
- Less code by two or three orders of magnitude

- Less code
- Less bugs
- Less maintenance
- Smaller attack surface
- Code fully owned by team
- Easier to refactor and grow

Systems designed around the data are much simpler, than you think.



Is it enough just to look at data?

- Our brains are really bad with data analysis
- We have a lot of cognitive biases
- We can't even intuitively grasp probabilities

What's typical requests distribution?

Requests distribution

Constant interval

Requests distribution

Constant interval

Uniform distribution

Requests distribution

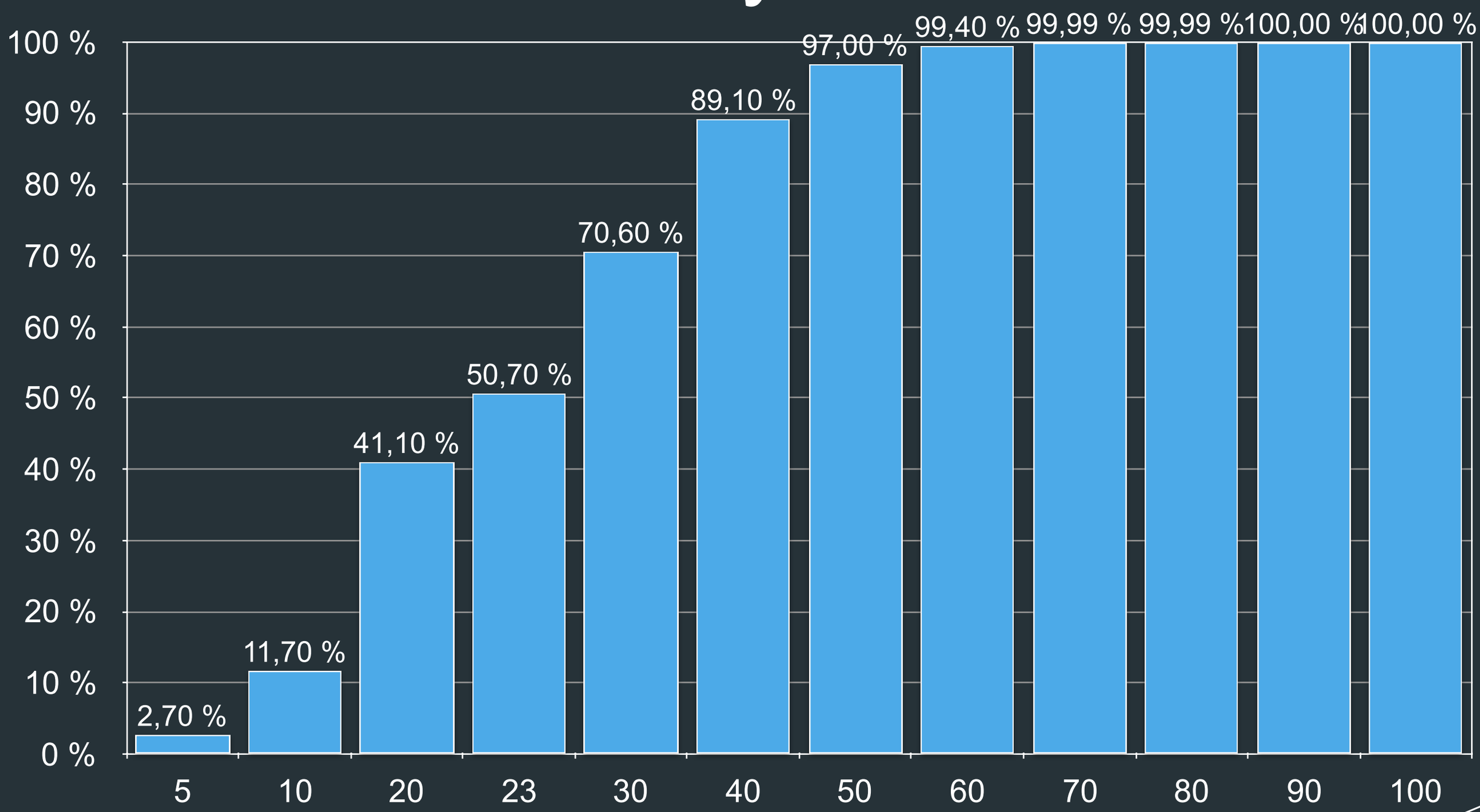
Constant interval

Uniform distribution

Poisson distribution

What's the probability of two people in this room to have birthday at the same day?

Birthday Paradox



#dfua

US AirForce and averages

- In late 1940s, the USAF had a serious problem: the pilots could not keep control of their planes
- Planes were crashing even in the non-war period
- Sometimes up to 17 crashes per day

- Blaming pilots and training program didn't help
- Investigations confirmed that planes were ok
- But people were keep dying

- They turned the attention to the design of the cabin
- It was designed in late 20s for the average pilot
- Data was taken from the massive study of soldiers during the Civil War

- USAF conducted new study of 4000+ pilots
- Measured 140 different body parameters
- Checked how many pilots fit to average

What % of pilots fit into average by 10 parameters, relevant to the cabin design?

0

- Only by 3 metrics, just 3.5% of the pilots were "average sized"
- There was no such thing as an "average pilot"

- So, USAF ordered to make cabins adjustable, to fit wide range of different pilots.
- Unexplained plane mishaps had reduced drastically

But why?

Average in high-dimension spaces

- Our intuition is built mostly on 1 dimension
- We tend to think that average is "where the most of values are"

Average in high-dimension spaces

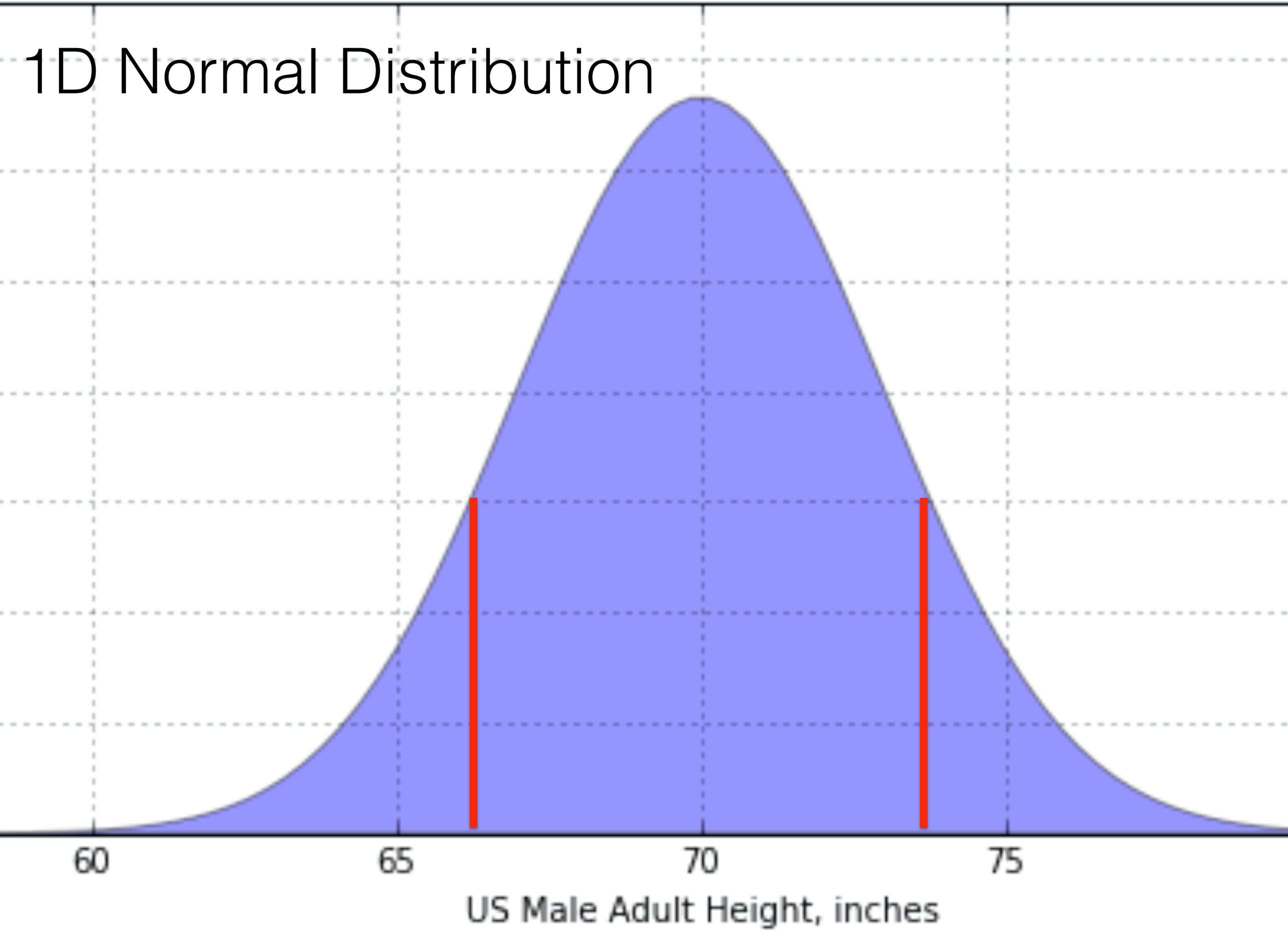
But it's only the particular case of:

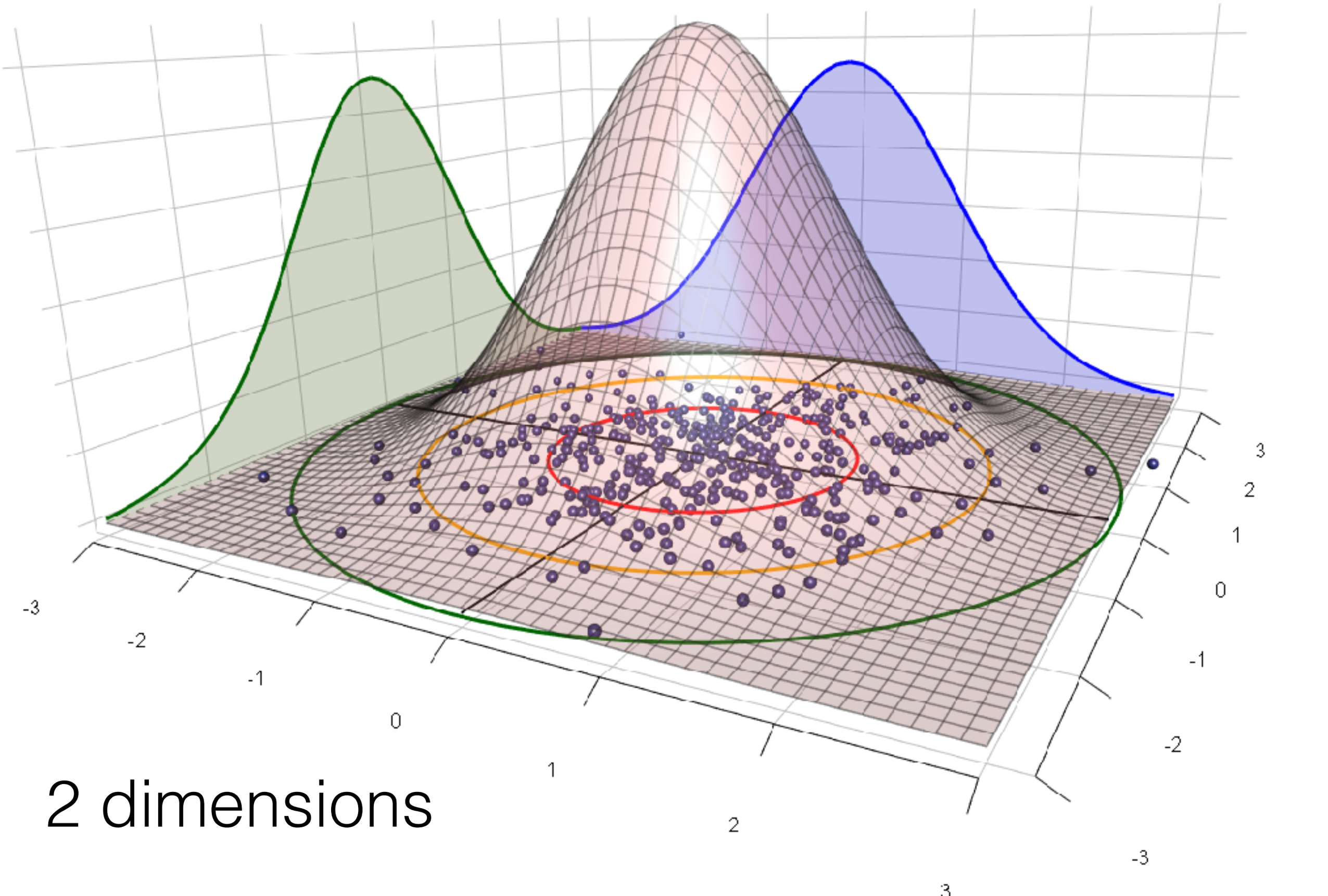
- 1 dimensional data
- Normal or similar distribution

Average in high-dimension spaces

- Average is actually more like "center of the mass"
- Average value of the donut is inside the hole
- But for high dimensions everything is really messed up

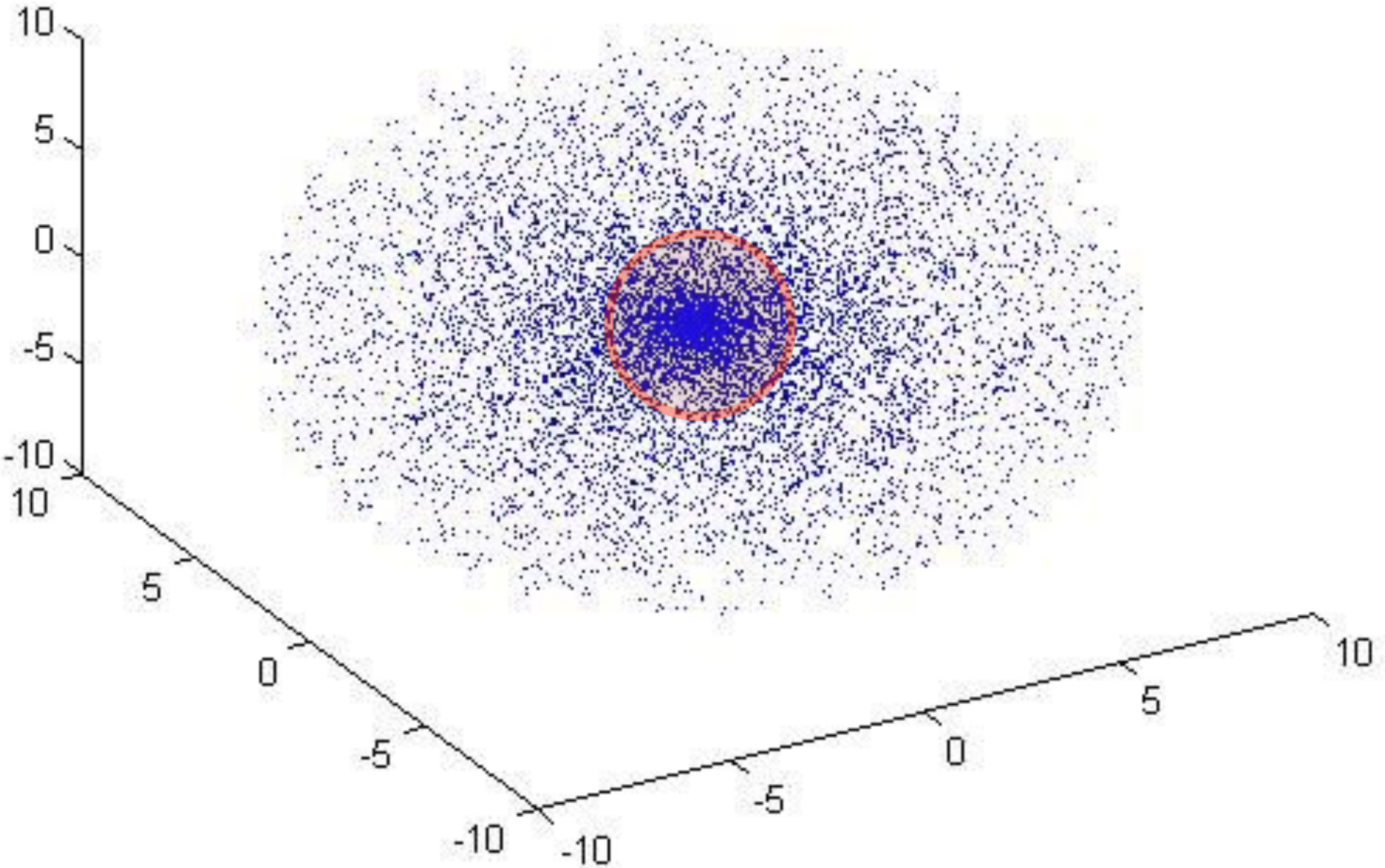
1D Normal Distribution





2 dimensions

3 dimensions



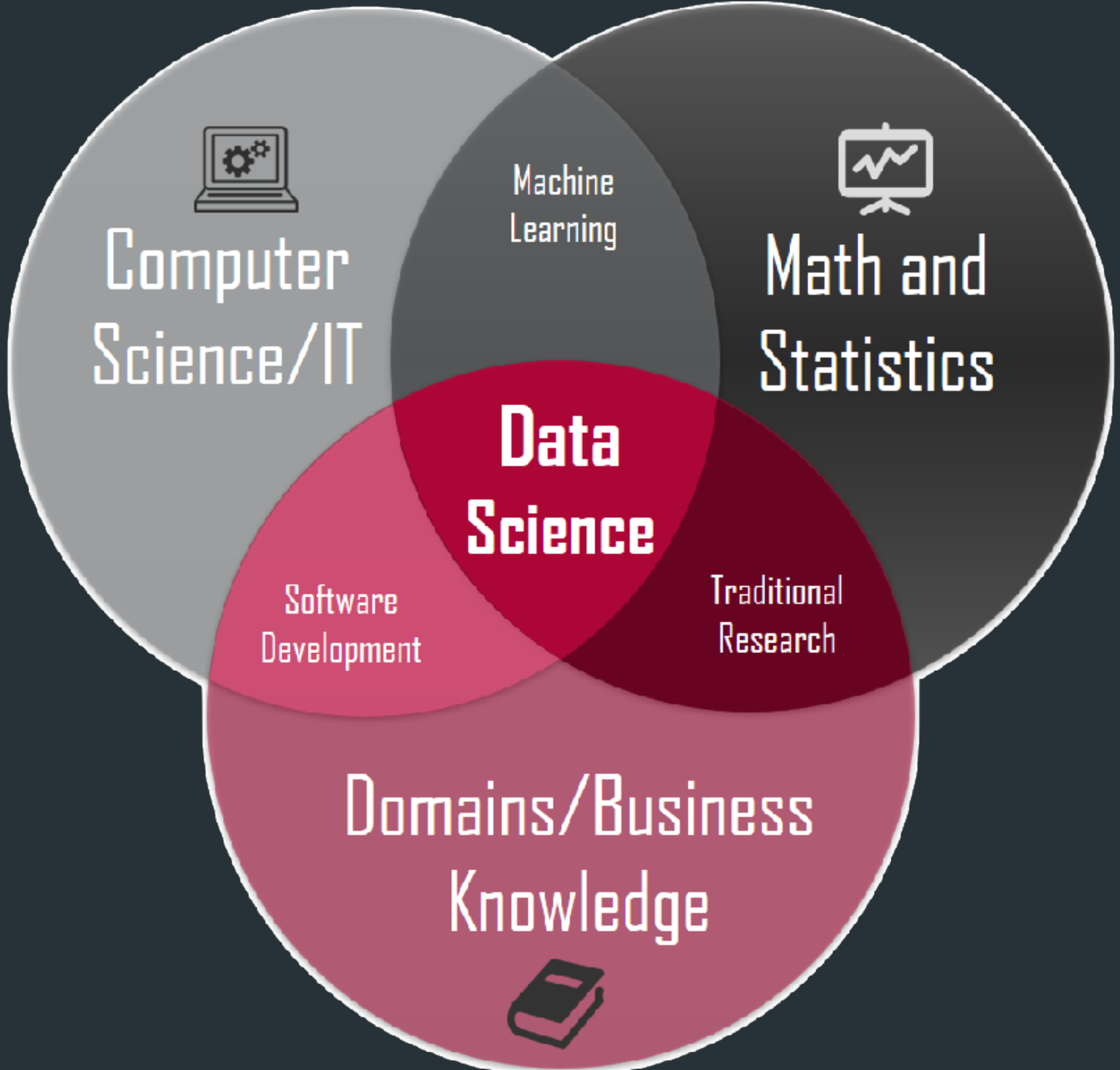
- As number of dimensions grows, mass moves from center to the perifery
- In 10 dimensions, all values are on the edges - "curse of dimensionality"
- As some professors say **"The N-dimensonal orange is all skin"**

- But our intuition is built upon 1-2-3 dimensions
- For many types of data, intuition is not enough, we need math
- Knowing these properties at that time, many human deaths in USAF could have been avoided

Data Science

Data Science

- It's an interdisciplinary field
- Math, statistics, computer science, visualization, machine learning, etc


Computer Science/IT
Machine Learning
Math and Statistics
Data Science
Software Development
Traditional Research
Domains/Business Knowledge

If you want to be a good software engineer, you should be passionate about data science.

- Learning data science improves your understanding of complex real-world problems, after all — including politics, economy, wars and poverty.

- It inevitably boosts your intellectual curiosity.

Where to start?

Where to start?

Whatever works best for you:

- video courses
- boring textbooks
- meetups and classes
- marrying a data scientist :)

Where to start?

Must topics:

- basic statistics
- probabilities
- R Language / Python Pandas / Go Gonum
- basics of neural networks
- basics of linear algebra

Where to start?

Video courses:

- Coursera: search "data science"
- Udemy: search "data science"
- Khan Academy
- Educational Youtube channels (they're gems!)

3Blue1Brown

406,735 subscribers

SUBSCRIBED 406K

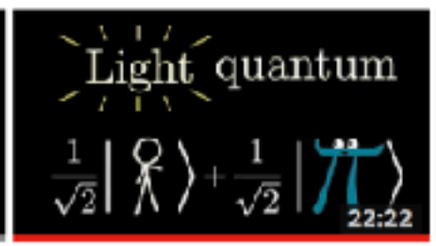
HOME VIDEOS PLAYLISTS CHANNELS DISCUSSION ABOUT

Uploads PLAY ALL

SORT BY

But what *is* a Neural Network? | Deep learning,

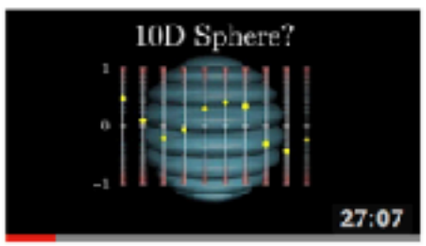
239K views • 1 week ago



Some light quantum mechanics (with

283K views • 1 month ago

CC



Thinking visually about higher dimensions

758K views • 2 months ago

CC

Hilbert's Curve: Is infinite math useful?

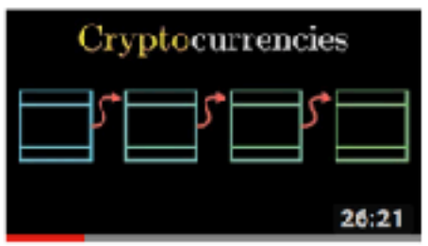
136K views • 2 months ago

CC

How secure is 256 bit security?

335K views • 3 months ago

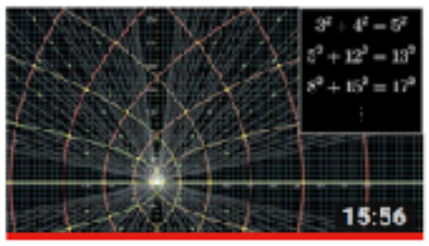
CC



Ever wonder how Bitcoin (and other cryptocurrencies)

617K views • 3 months ago

CC



All possible pythagorean triples, visualized

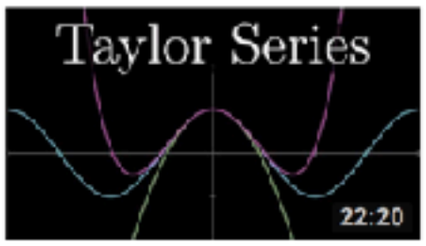
353K views • 4 months ago

CC

Pi hiding in prime regularities

307K views • 4 months ago

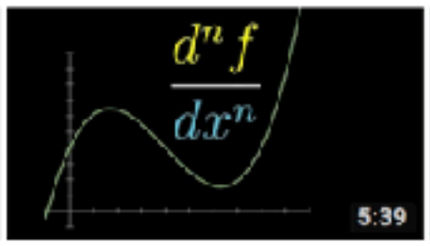
CC



Taylor series | Chapter 10, Essence of calculus

205K views • 5 months ago

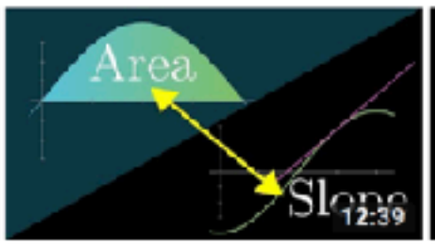
CC



Higher order derivatives | Footnote, Essence of

68K views • 5 months ago

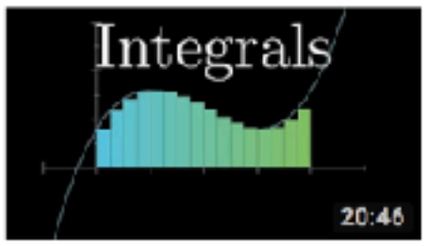
CC



What does area have to do with slope? | Chapter 9,

107K views • 5 months ago

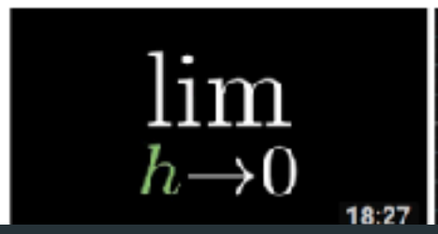
CC

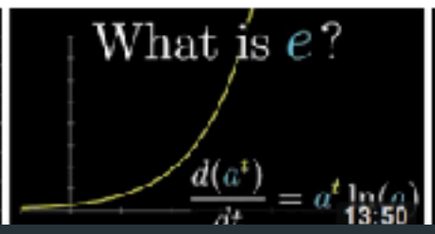


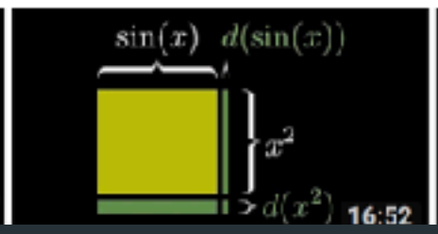
Integration and the fundamental theorem of

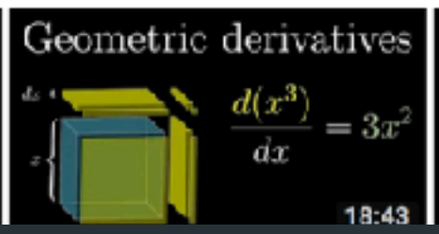
116K views • 5 months ago

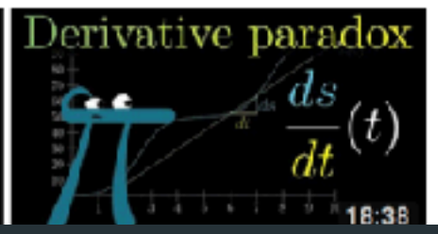
CC











Papers We Love Conf

Join us in St. Louis on September 28th, 2017

Tickets for Strangeloop and PWLConf are still available!

Papers We Love is a repository of academic computer science papers and a community who loves reading them.

San Francisco *New York* *Seattle* *Montreal* *St. Louis* *Vienna* *Bangalore* *Singapore* *London* *Toronto*
Columbus *Los Angeles* *Pune* *Chattanooga* *Winnipeg* *Bucharest* *Washington, DC* *Boston* *San Diego*
Portland *Madrid* *Hyderabad* *Amsterdam* *Gothenburg* *Chicago* *Munich* *Utrecht* *Belfast* *Raleigh-Durham*
Lebanon *Teresina* *Zürich* *Brasilia* *Denver* *Philadelphia* *Budapest* *Hamburg* *Reykjavik* *Berlin* *Kathmandu*
Seoul *Kyiv*

October Meetups

Meetups **Posted by:** Joshua | Oct 1, 2017

We have another great line-up of meet-ups scheduled for October across a number of our chapters:

Vienna 10/2: October: Bitcoin

Chattanooga 10/3: Jacob Kobernik on Blockchain: A Graph Primer

Resume

1. Think about the whole system as one program

2. Always ask questions about data you work with

3. To make sense of this data, learn data science

Links

- http://highscalability.com/blog/2016/4/20/how-twitter-handles-3000-images-per-second.html
- https://skillsmatter.com/skillscasts/8355-london-go-usergroup
- https://www.thestar.com/news/insight/2016/01/16/when-us-air-force-discovered-the-flaw-of-averages.html
- https://medium.com/@charlie.b.ohara/breaking-down-big-o-notation-40963a0f4e2a
- https://www.youtube.com/watch?v=gas2v1emubU
- https://algorithmia.com/algorithms/ovi_mihai/TimestampToDate
- https://en.wikipedia.org/wiki/Disjoint-set_data_structure

Thank you!

Questions?

Ivan Danyliuk
@idanyliuk

GDG DevFest Ukraine 2017