

Algorithm Engineering Lecture 5: Workshop on Customized Data Structures

CSE 431

Spring 2024

Kevin Liu, Ph.D.

Michigan State University

Optional reading:
Skiena 2012
chapter 3

Includes material adapted from
Stony Brook 373,
U.Toronto BCH441H, and
previous editions of CSE431



A Problem-Solving Toolbox

	Search	Insert	Delete	Index	Min	Max	Pred	Succ
Unsorted Array	$O(n)$	$O(1)$	$O(1)$	$O(1)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Sorted Array	$O(\log n)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
Unsorted List	$O(n)$	$O(1)$	$O(1)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Sorted List	$O(n)$	$O(n)$	$O(1)$	$O(n)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
Binary Search Tree	$O(n)$	$O(n)$	$O(n)$	-	$O(n)$	$O(n)$	$O(n)$	$O(n)$
AVL Tree	$O(\log n)$	$O(\log n)$	$O(\log n)$	-	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(\log n)$
Hash Table (good hash!)	$O(1)$	$O(1)$	$O(1)$	-	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Heap	$O(n)$	$O(\log n)$	$O(\log n)$	-	$O(n)$	$O(1)$	$O(n)$	$O(n)$

Priority Queue With Removal

Design a data structure that supports `find_maximum`, `insert`, and `remove last inserted element` as fast as possible.

Min/Max Priority Queue With Removal

Design a data structure that supports `extract_maximum`, `extract_minimum`, and `insert` as fast as possible.

Min/Max Priority Queue With Removal

Design a data structure that supports `extract_maximum`, `extract_minimum`, and `insert` as fast as possible.

How would the data structure be different if one of the operations was much less frequent than the other two?

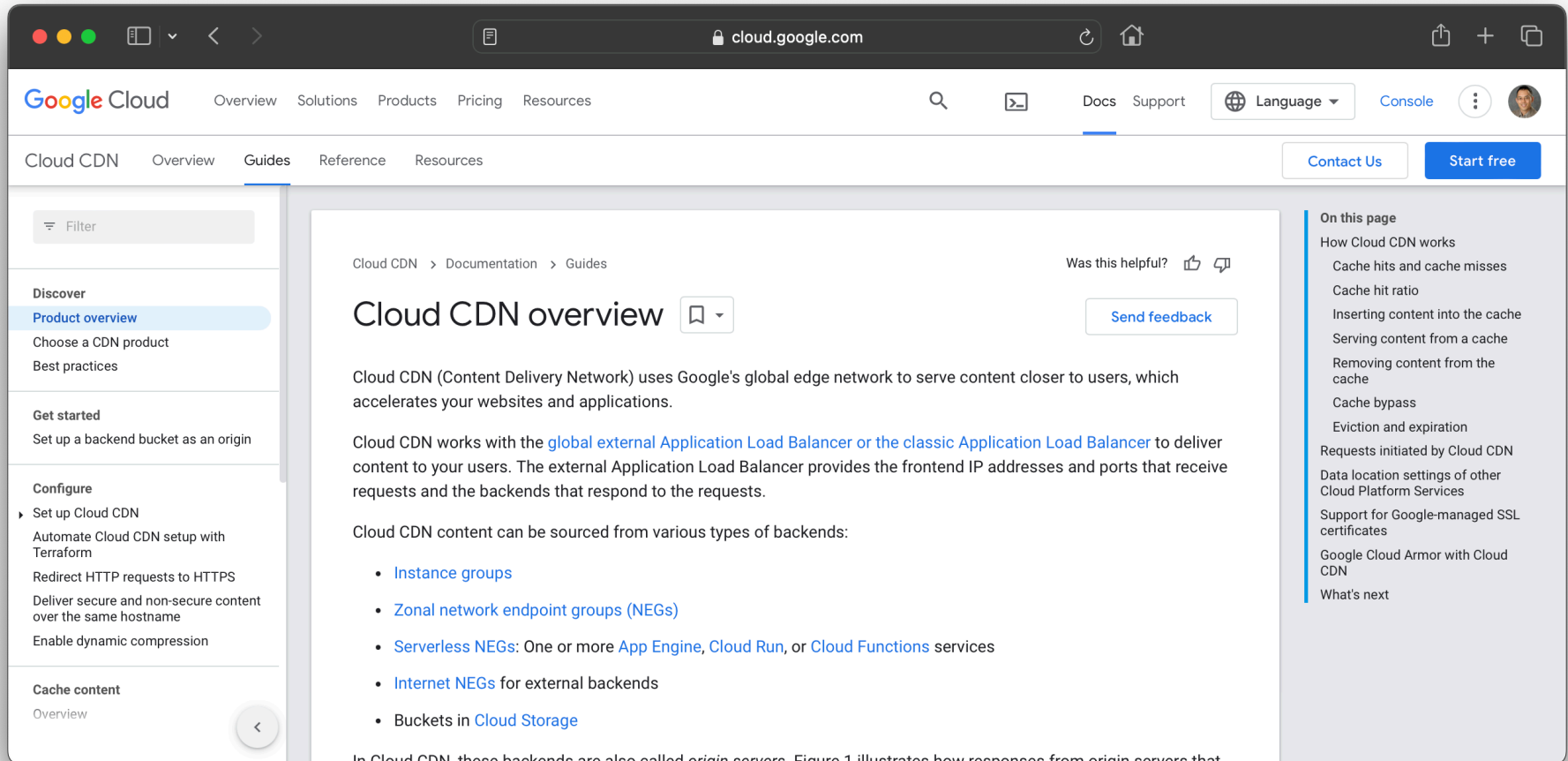
Weighted Random Numbers

Given a `random()` function that returns values uniformly distributed between 0 and 1, come up with an efficient algorithm that can randomly choose between outcomes that each have a specified weight.

Example: $A = 30\%$; $B = 10\%$; $C = 5\%$; ...
Assume that there are many outcomes.

REAL-WORLD APPLICATION I: WEB CACHES

Content Delivery Networks (CDNs) and Web Caching



The screenshot shows the Google Cloud website with the 'Cloud CDN' section selected. The main content area is titled 'Cloud CDN overview' and provides a detailed explanation of how Cloud CDN works, including its use of Google's global edge network and its integration with various load balancers and backends. A left sidebar contains navigation links for 'Discover', 'Get started', 'Configure', and 'Cache content'. A right sidebar lists 'On this page' topics, such as 'How Cloud CDN works', 'Cache hits and cache misses', and 'What's next'.

Google Cloud Overview Solutions Products Pricing Resources

Cloud CDN Overview **Guides** Reference Resources

Filter

Discover

- Product overview**
- Choose a CDN product
- Best practices

Get started

- Set up a backend bucket as an origin

Configure



- Set up Cloud CDN
 - Automate Cloud CDN setup with Terraform
 - Redirect HTTP requests to HTTPS
 - Deliver secure and non-secure content over the same hostname
 - Enable dynamic compression

Cache content

- Overview

Cloud CDN > Documentation > Guides

Cloud CDN overview

Was this helpful?  

[Send feedback](#)

Cloud CDN (Content Delivery Network) uses Google's global edge network to serve content closer to users, which accelerates your websites and applications.

Cloud CDN works with the [global external Application Load Balancer](#) or the [classic Application Load Balancer](#) to deliver content to your users. The external Application Load Balancer provides the frontend IP addresses and ports that receive requests and the backends that respond to the requests.

Cloud CDN content can be sourced from various types of backends:

- [Instance groups](#)
- [Zonal network endpoint groups \(NEGs\)](#)
- [Serverless NEGs](#): One or more [App Engine](#), [Cloud Run](#), or [Cloud Functions](#) services
- [Internet NEGs](#) for external backends
- Buckets in [Cloud Storage](#)

In Cloud CDN, these backends are also called *origin servers*. Figure 1 illustrates how responses from origin servers that

On this page

- How Cloud CDN works
 - Cache hits and cache misses
 - Cache hit ratio
 - Inserting content into the cache
 - Serving content from a cache
 - Removing content from the cache
 - Cache bypass
 - Eviction and expiration
- Requests initiated by Cloud CDN
- Data location settings of other Cloud Platform Services
- Support for Google-managed SSL certificates
- Google Cloud Armor with Cloud CDN
- What's next

Content Delivery Networks (CDNs) and Web Caching

The image shows a screenshot of the Google Cloud website, specifically the Cloud CDN documentation page. The browser's address bar shows 'cloud.google.com'. The page has a navigation bar with links for Overview, Solutions, Products, Pricing, and Resources. Below this, there's a sub-navigation bar for Cloud CDN with links for Overview, Guides, Reference, and Resources. A 'Filter' button is visible on the left. The main content area is titled 'Cloud CDN > Documentation > Guides'. On the right, there's a 'Was this helpful?' section with thumbs up and down icons. A sidebar on the right lists 'On this page' with links for 'How Cloud CDN works' and 'Cache hits and cache misses'. A large white text box is overlaid on the page, containing the text: 'Global CDN market was \$20 billion USD in 2023 (https://www.mordorintelligence.com/industry-reports/content-delivery-market)'. Below the overlay, the 'Configure' section is partially visible, showing options like 'Set up Cloud CDN', 'Automate Cloud CDN setup with Terraform', 'Redirect HTTP requests to HTTPS', 'Deliver secure and non-secure content over the same hostname', and 'Enable dynamic compression'. The 'Cache content' section is also partially visible, showing an 'Overview' link. The 'requests and the backends that respond to the requests.' section is also partially visible, showing a list of backends: 'Instance groups', 'Zonal network endpoint groups (NEGs)', 'Serverless NEGs: One or more App Engine, Cloud Run, or Cloud Functions services', 'Internet NEGs for external backends', and 'Buckets in Cloud Storage'.

Global CDN market was \$20 billion USD in 2023
(<https://www.mordorintelligence.com/industry-reports/content-delivery-market>)

Configure

- Set up Cloud CDN
 - Automate Cloud CDN setup with Terraform
 - Redirect HTTP requests to HTTPS
 - Deliver secure and non-secure content over the same hostname
 - Enable dynamic compression
- Cache content
 - Overview

requests and the backends that respond to the requests.

Cloud CDN content can be sourced from various types of backends:

- Instance groups
- Zonal network endpoint groups (NEGs)
- Serverless NEGs: One or more App Engine, Cloud Run, or Cloud Functions services
- Internet NEGs for external backends
- Buckets in Cloud Storage

On this page

- How Cloud CDN works
- Cache hits and cache misses

Cloud Platform Services

- Support for Google-managed SSL certificates
- Google Cloud Armor with Cloud CDN
- What's next

In Cloud CDN, these backends are also called *origin servers*. Figure 1 illustrates how responses from origin servers that

Web Cache Filtering

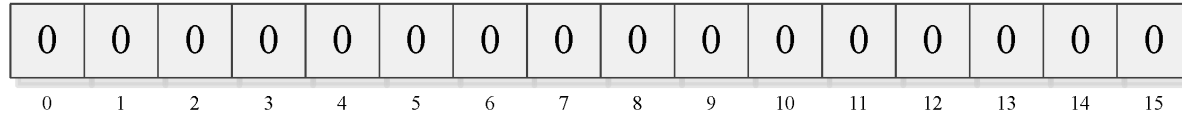
- URL requests in high-throughput and online manner
- Well over half of URLs are requested once only and never again requested
- We want to cache URLs (and URL content) that are accessed more than once
- The set of URLs is many orders of magnitude larger than we can store on our computing infrastructure, and growing all the time

Web Cache Filtering

- Propose an algorithmic solution that keeps track of all URL requests that have been accessed at least once before
- Memory usage is capped to a fixed size b
- We really don't want false negatives: check if a URL request is brand-new with 100% certainty
 - Note: this is by FAR the most common case
 - Recall discussion about abstracting away less important cases and focusing on the one/few cases that really matter
- But false positives are ok: occasionally caching some URLs that have been accessed once (but not two or more times) only adds a bit of overhead

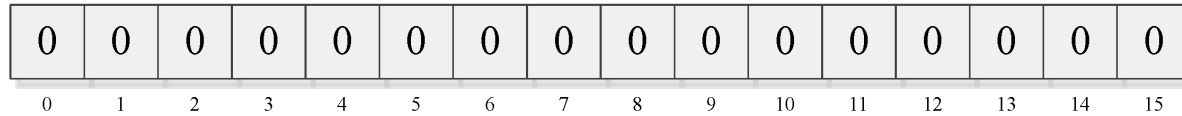
Web Cache Filtering

Bloom Filters

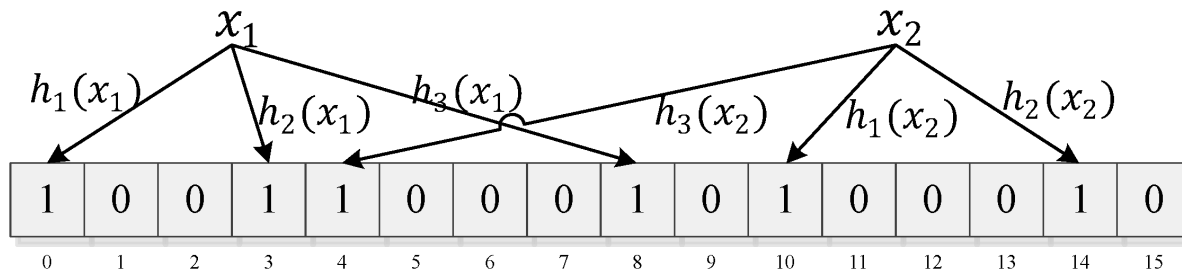


(a) A Bloom filter initialized with 0s

Bloom Filters

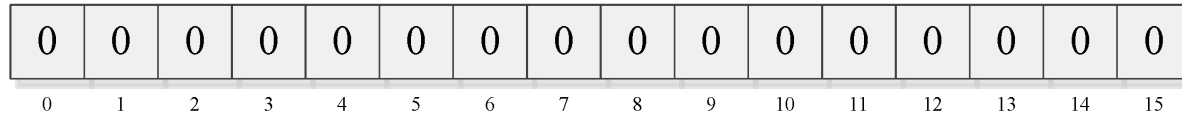


(a) A Bloom filter initialized with 0s

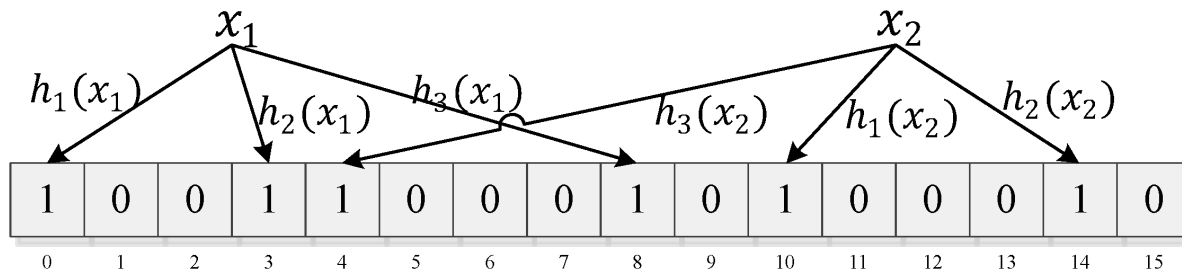


(b) Insert operation of Bloom filter

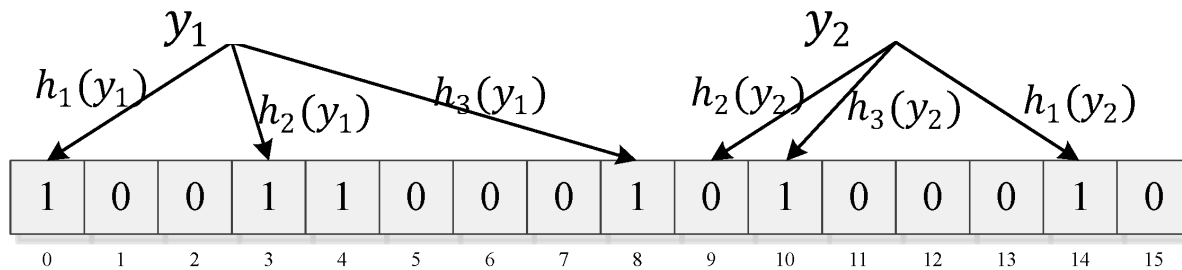
Bloom Filters



(a) A Bloom filter initialized with 0s



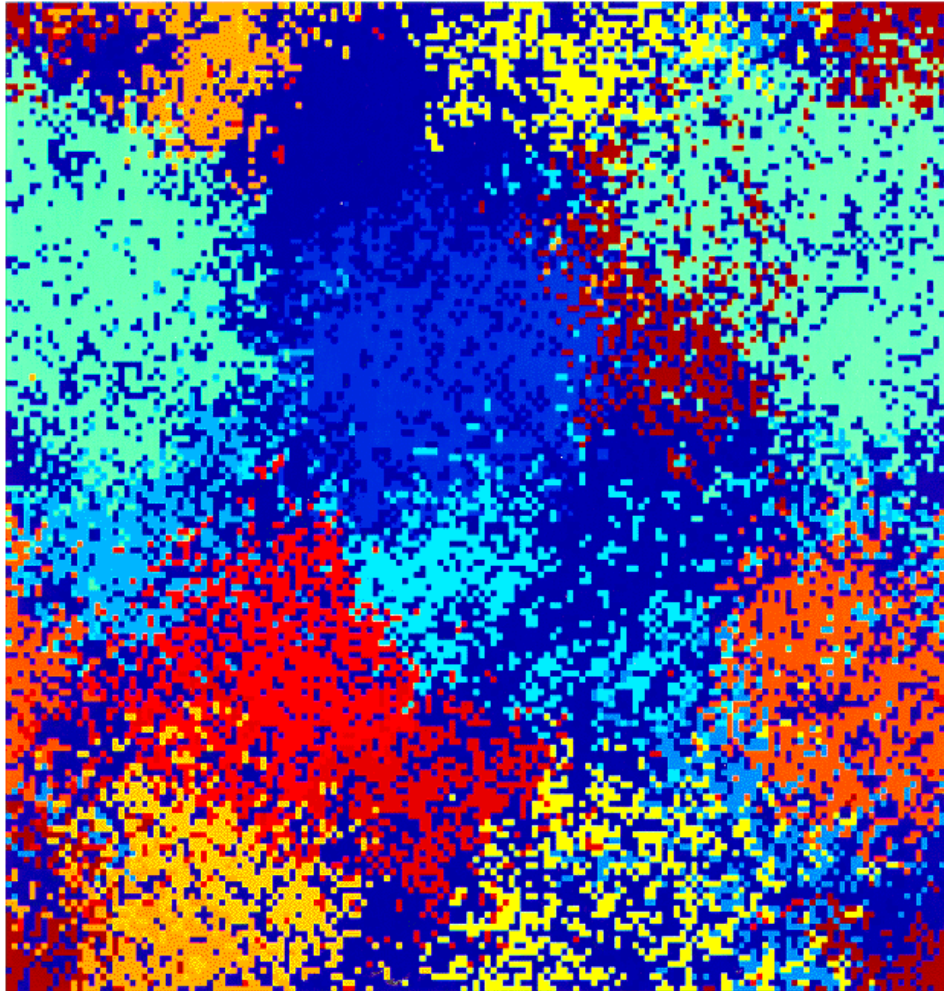
(b) Insert operation of Bloom filter



(c) Query operation of Bloom filter

REAL-WORLD APPLICATION 2: SCIENTIFIC SIMULATION SOFTWARE

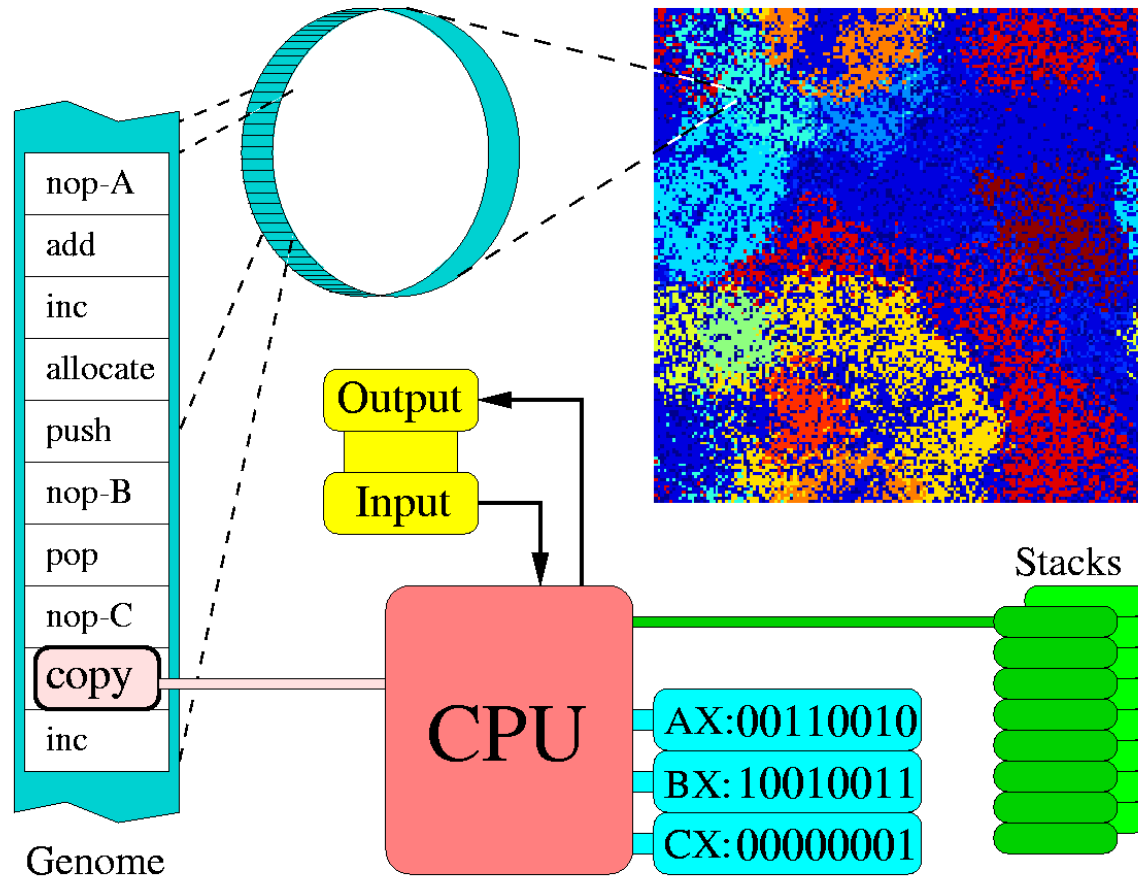
A Research Platform: Avida



Avida is a research platform for experimental studies of evolution and ecology with digital organisms.

Researchers can control most aspects of the system and record any measures of their choosing.

Physical World in Avida



Adaptation in Avida

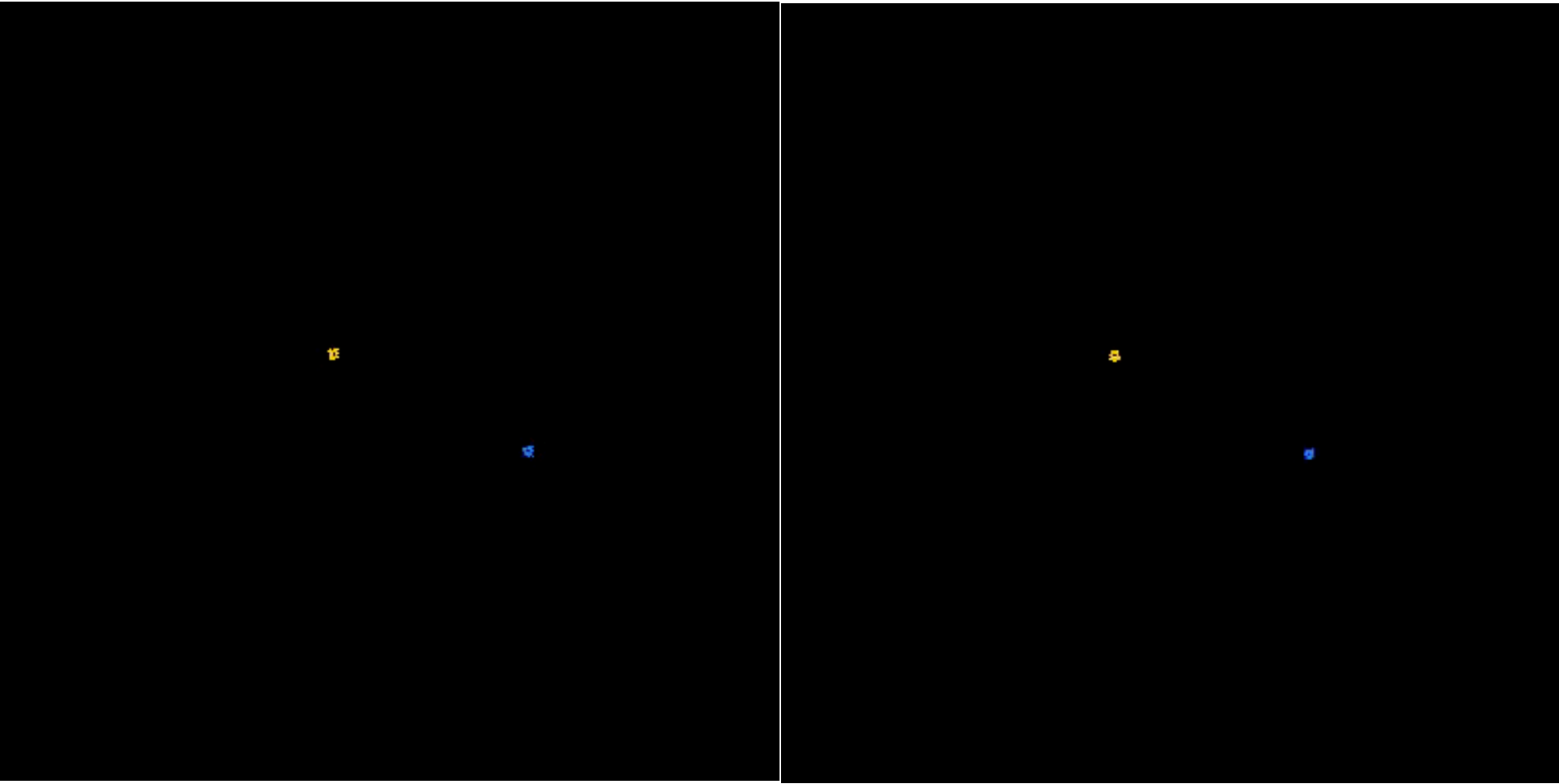
CPU cycles = Energy

Simple environments: all organisms receive the same amount of energy.

Complex environments: organisms compute math functions to metabolize a resource and gain energy.

Evolution *minimizes* time to produce offspring and adjusts priorities to *maximize* net energy absorbed.

Competition Experiments



The Situation:

We have n virtual CPUs each of which have a processing speed P_i . We want to each virtual CPU to be called with a probability proportional to its processing speed.

Example: $P_A = 2, P_B = 5, P_C = 3$

Sequence: B C A B B C B C A B B C A B B C B C A ...

The Problem:

Given n objects, each of which have a processing speed, devise an algorithm that will produce a sequence in which the objects should be executed.

Analyze for speed of

Select() – Pick a random CPU

AdjustSpeed(ID, speed) – Change speed of a CPU

Assume that both of the functions we are analyzing will be run $O(n)$ times. No other functions are needed.

Random returns a uniform random value in constant time.

Updated Problem:

What if we want to make sure that our output is as evenly distributed as possible? In other words:

Example: $P_A = 2$, $P_B = 4$

GOOD Sequence: B B A B B A B B A B B A B B A B B A

POOR Sequence: A A B B B B A A B B B B A A B B B B

BAD Sequence: B B B A B B B B B B B B A A A B A A

Next time:
Everything You Ever Wanted to Know About
Sorting (In 80 Minutes or Less!)

