introduction to

WEB CRAWLING

& extraction

by Nate Murray

WHO AM I ?

▸ Nate Murray

AT&T Interactive (Yellowpages.com)

TB-scale data since 2009

Various crawlers since 2005

what is

WEB CRAWLING?

definition:

▸ web crawler

a program that browses the web.

definition:

▸ web extraction

transforming unstructured web data into structured data

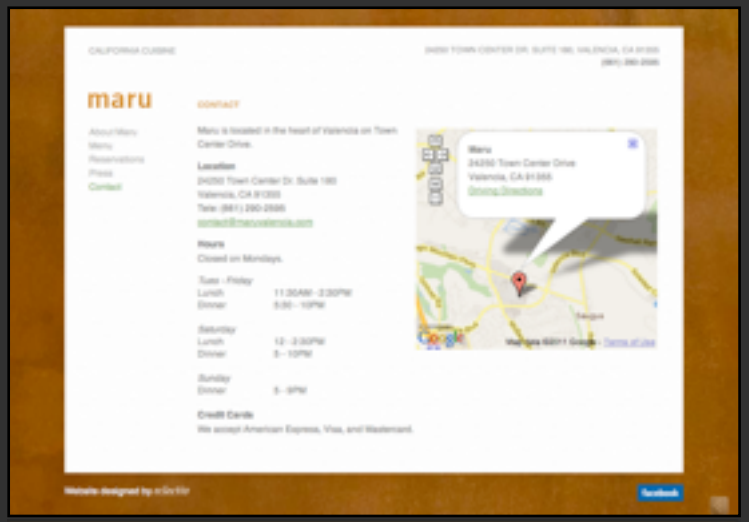
definition:

▸ web extraction

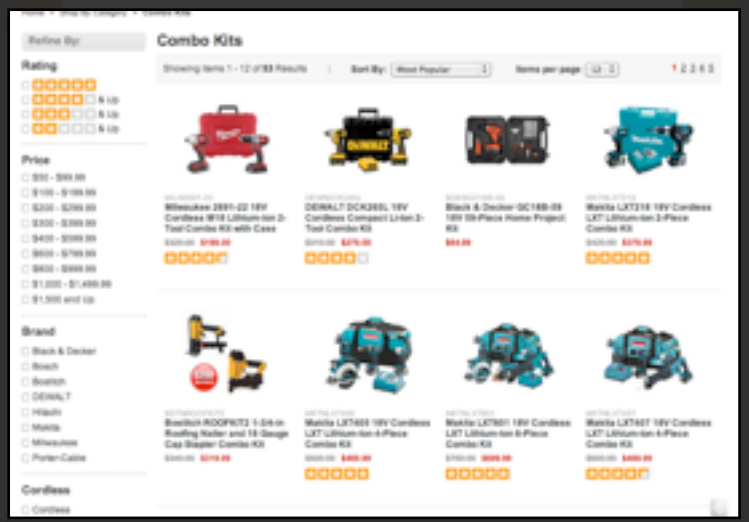
transforming semistructured web data into structured data

motivation


delicious
Scrapy | An open source web scraping framework for Python


maru


34,688


Combo Kits

motivation: *bookmark buddies*

Hi, jashmenn | Inbox | Settings | Help | Sign Out

delicious

Home | Bookmarks | People | Tags

Search Delicious | Search

Everyone's Bookmarks for:

Scrapy | An open source web scraping framework for Python

scrapy.org/

Save this bookmark
Share this bookmark
Look up another URL

History | Notes

Saved 1019 times, first saved by Alex O'Connor on 30 Dec 08. View Chart

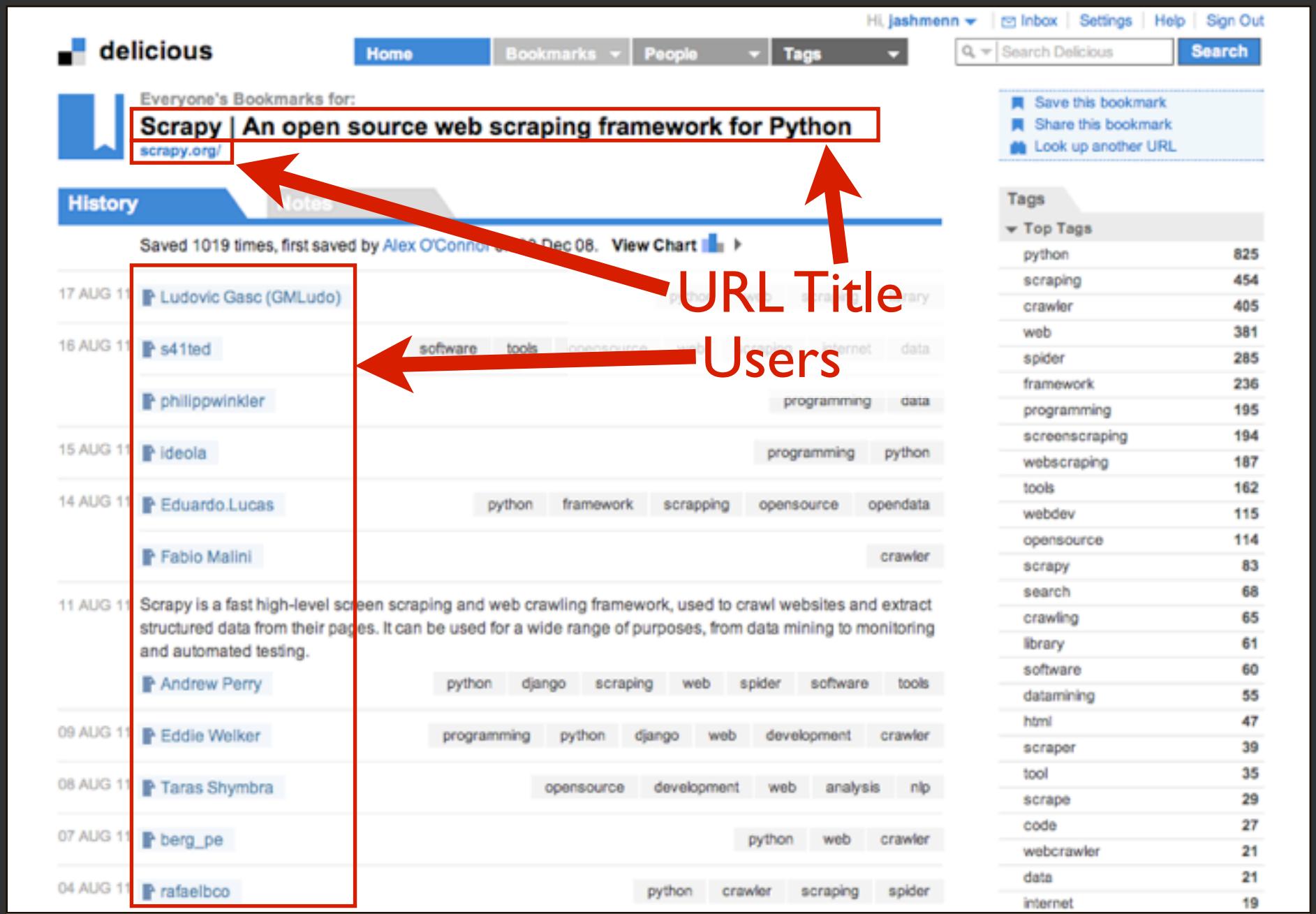
17 AUG 11	Ludovic Gasc (GMLudo)	python web scraping library
16 AUG 11	s41ted	software tools opensource web scraping internet data
	philippwinkler	programming data
15 AUG 11	ideola	programming python
14 AUG 11	Eduardo.Lucas	python framework scrapping opensource opendata
	Fabio Malini	crawler
11 AUG 11	Scrapy is a fast high-level screen scraping and web crawling framework, used to crawl websites and extract structured data from their pages. It can be used for a wide range of purposes, from data mining to monitoring and automated testing.	
	Andrew Perry	python django scraping web spider software tools
09 AUG 11	Eddie Welker	programming python django web development crawler
08 AUG 11	Taras Shymbra	opensource development web analysis nlp
07 AUG 11	berg_pe	python web crawler
04 AUG 11	rafaelbco	python crawler scraping spider

Tags

Top Tags

python	825
scraping	454
crawler	405
web	381
spider	285
framework	236
programming	195
screenscraping	194
webscraping	187
tools	162
webdev	115
opensource	114
scrapy	83
search	68
crawling	65
library	61
software	60
datamining	55
html	47
scraper	39
tool	35
scrape	29
code	27
webcrawler	21
data	21
internet	19

motivation: *bookmark buddies*

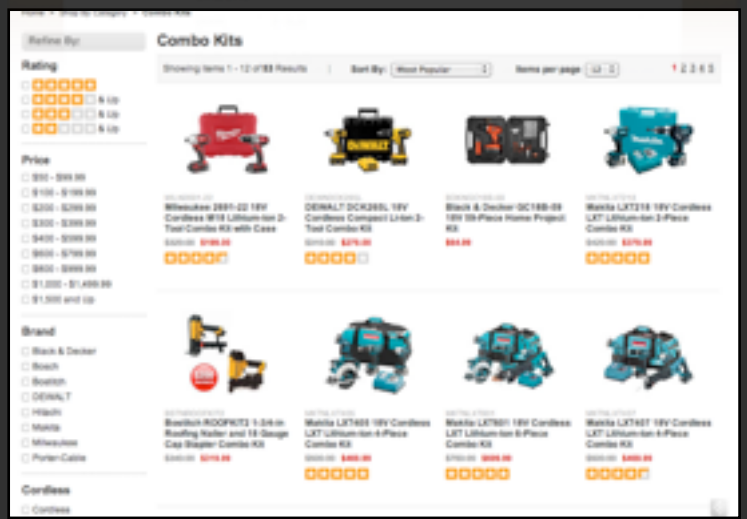


motivation:


delicious
Scrapy | An open source web scraping framework for Python


maru


34,688


Combo Kits

motivation: *business hours*

CALIFORNIA CUISINE

24250 TOWN CENTER DR. SUITE 180, VALENCIA, CA 91355
(661) 290-2595

maru

- About Maru
- Menu
- Reservations
- Press
- Contact

CONTACT

Maru is located in the heart of Valencia on Town Center Drive.

Location
24250 Town Center Dr. Suite 180
Valencia, CA 91355
Tele: (661) 290-2595
contact@maruvalencia.com

Hours
Closed on Mondays.

Tues - Friday

Lunch	11:30AM - 2:30PM
Dinner	5:30 - 10PM

Saturday

Lunch	12 - 2:30PM
Dinner	5 - 10PM

Sunday

Dinner	5 - 9PM

Credit Cards
We accept American Express, Visa, and Mastercard.

Website designed by *eclectiv*

facebook

motivation: *business hours*

CALIFORNIA CUISINE

24250 TOWN CENTER DR. SUITE 180, VALENCIA, CA 91355
(661) 290-2595

maru

About Maru
Menu
Reservations
Press
Contact

CONTACT

Maru is located in the heart of Valencia on Town Center Drive.

Location
24250 Town Center Dr. Suite 180
Valencia, CA 91355
Tele: (661) 290-2595
contact@maruvalencia.com

Hours
Closed on Mondays.

Tues - Friday
Lunch 11:30AM - 2:30PM
Dinner 5:30 - 10PM

Saturday
Lunch 12 - 2:30PM
Dinner 5 - 10PM

Sunday
Dinner 5 - 9PM

Credit Cards
We accept American Express, Visa, and Mastercard

Day	Openness	
Mon	Closed	
Tue	11:30–14:30	17:30–22:00
Wed	11:30–14:30	17:30–22:00
Thur	11:30–14:30	17:30–22:00
Fri	11:30–14:30	17:30–22:00
Sat	12:00–14:30	17:00–22:00
Sun	–	17:00–21:00

Website designed by *eclectiv*

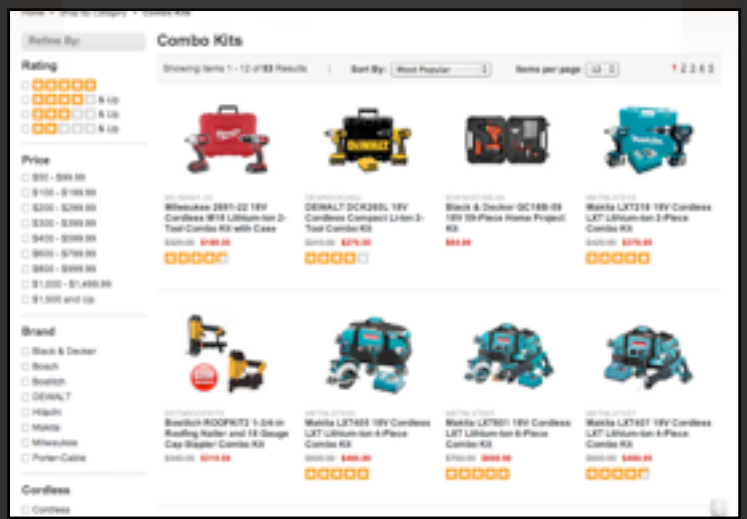
facebook

motivation:

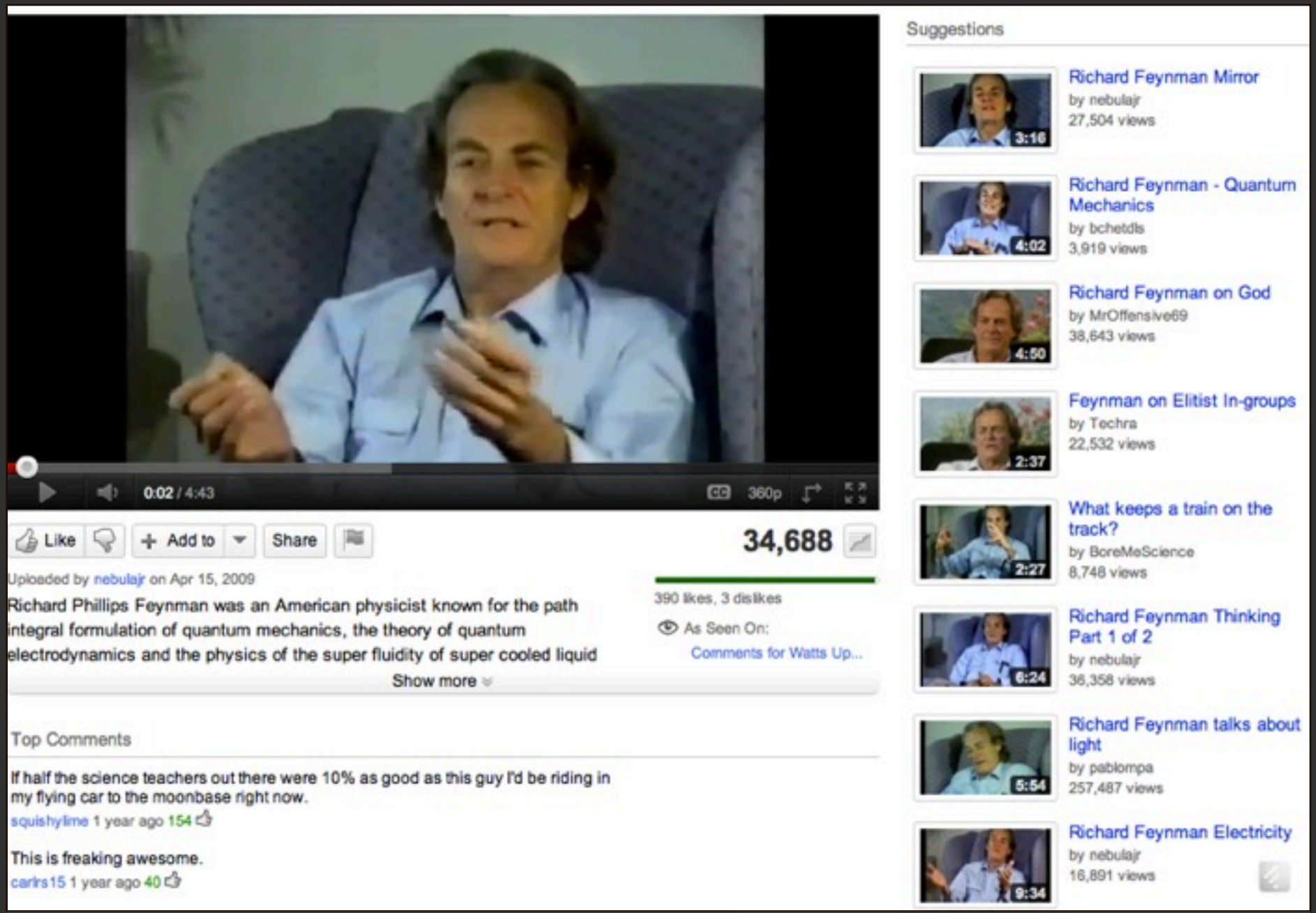

delicious
Scrapy | An open source web scraping framework for Python


maru

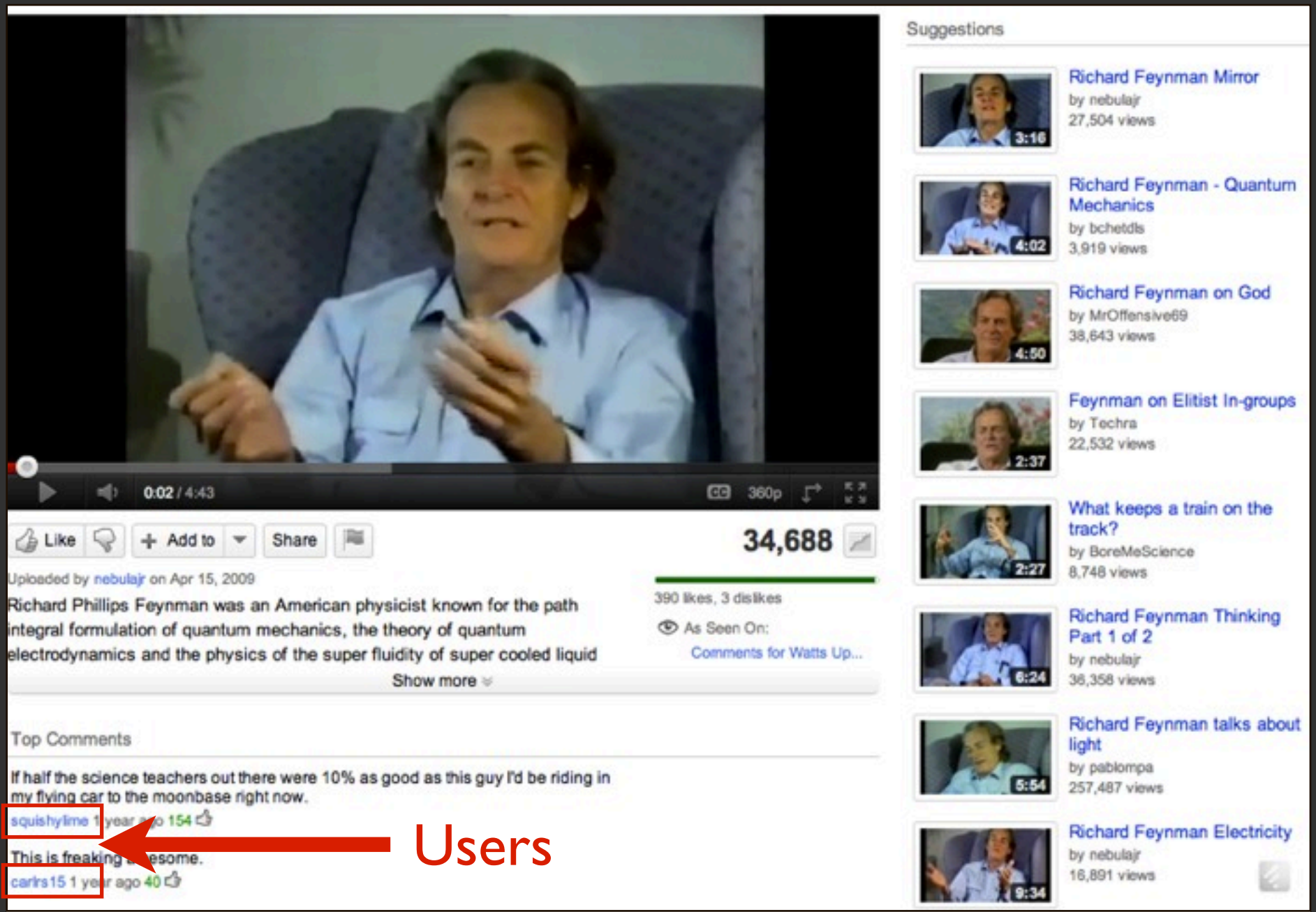

34,688


Combo Kits
Rating
Price
Brand
Cordless

motivation: *recommend videos*



motivation: *recommend videos*

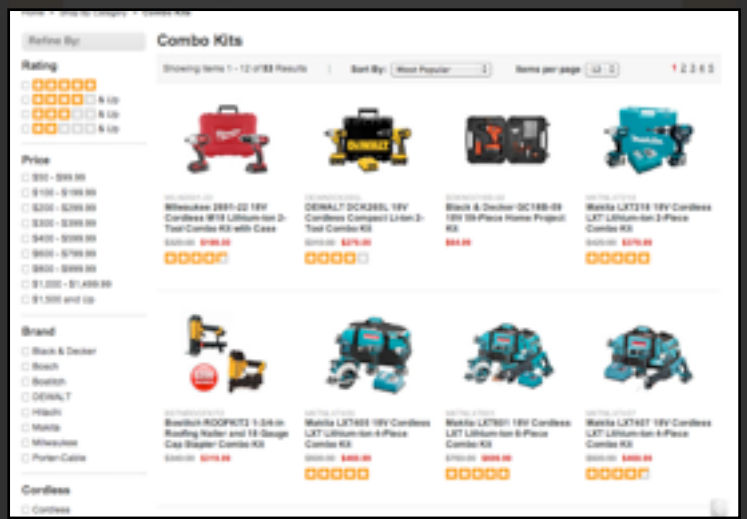


motivation:


delicious
Scrapy | An open source web scraping framework for Python


maru


34,688


Combo Kits

motivation: *vertical search*

Home > Shop By Category > Combo Kits

Refine By:

Rating

- ☐ ★★★★★
- ☐ ★★★★☆ & Up
- ☐ ★★★☆☆ & Up
- ☐ ★★☆☆☆ & Up

Price

- ☐ $50 - $99.99
- ☐ $100 - $199.99
- ☐ $200 - $299.99
- ☐ $300 - $399.99
- ☐ $400 - $599.99
- ☐ $600 - $799.99
- ☐ $800 - $999.99
- ☐ $1,000 - $1,499.99
- ☐ $1,500 and Up

Brand

- ☐ Black & Decker
- ☐ Bosch
- ☐ Bostitch
- ☐ DEWALT
- ☐ Hitachi
- ☐ Makita
- ☐ Milwaukee
- ☐ Porter-Cable

Cordless

- ☐ Cordless

Combo Kits

Showing items 1 - 12 of **53** Results | Sort By: Most Popular | Items per page 12 | 1 2 3 4 5

MILN2691-22
Milwaukee 2691-22 18V Cordless M18 Lithium-Ion 2-Tool Combo Kit with Case
~~$329.00~~ **$199.00**

DEWNDCK265L
DEWALT DCK265L 18V Cordless Compact Li-Ion 2-Tool Combo Kit
~~$319.00~~ **$279.00**

BDKNGC18B-59
Black & Decker GC18B-59 18V 59-Piece Home Project Kit
$64.99

MKTNLXT218
Makita LXT218 18V Cordless LXT Lithium-Ion 2-Piece Combo Kit
~~$429.99~~ **$379.99**

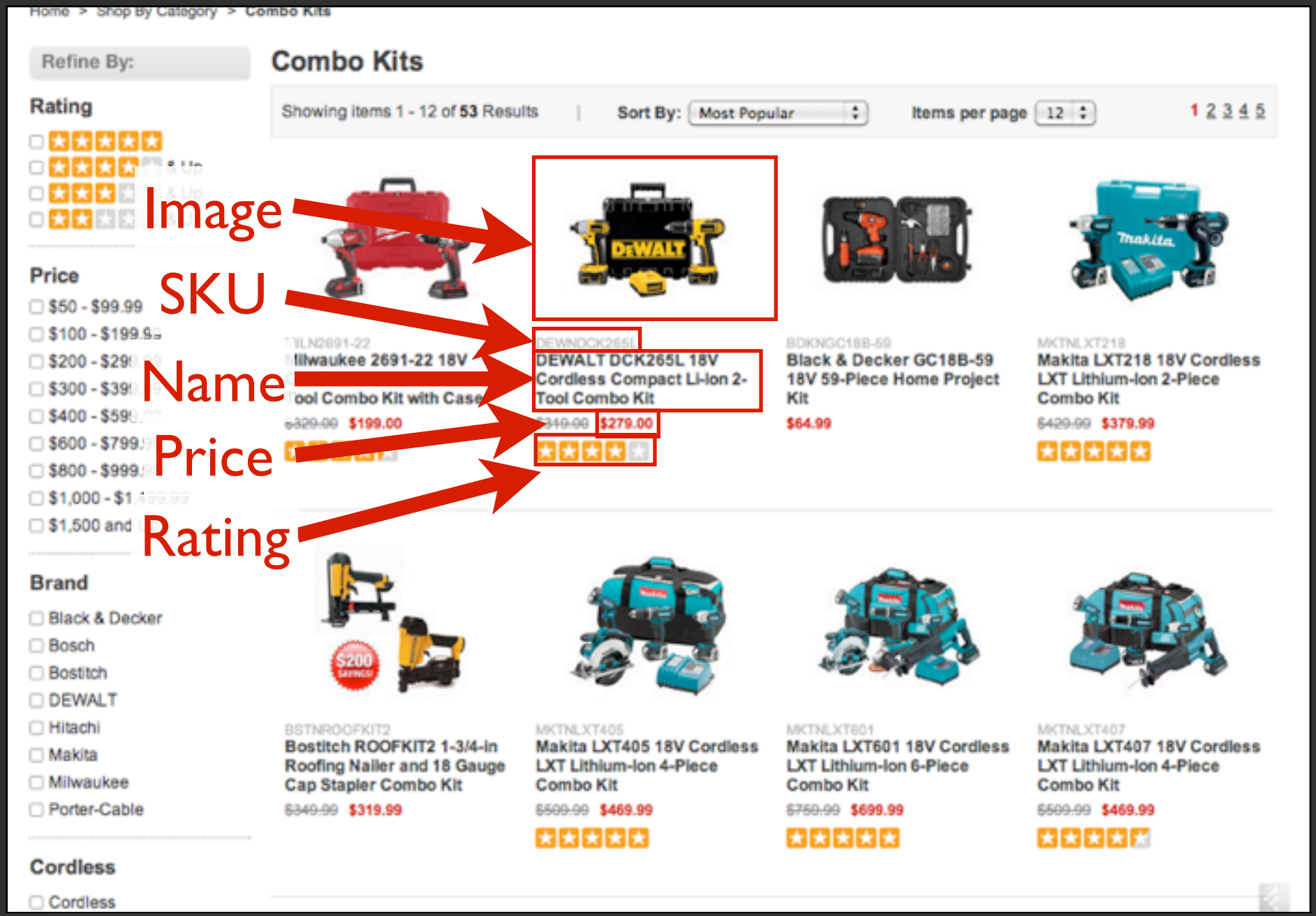
BSTNROOFKIT2
Bostitch ROOFKIT2 1-3/4-in Roofing Nailer and 18 Gauge Cap Stapler Combo Kit
~~$349.99~~ **$319.99**

MKTNLXT405
Makita LXT405 18V Cordless LXT Lithium-Ion 4-Piece Combo Kit
~~$599.99~~ **$469.99**

MKTNLXT601
Makita LXT601 18V Cordless LXT Lithium-Ion 6-Piece Combo Kit
~~$759.99~~ **$699.99**

MKTNLXT407
Makita LXT407 18V Cordless LXT Lithium-Ion 4-Piece Combo Kit
~~$599.99~~ **$469.99**

motivation: *vertical search*

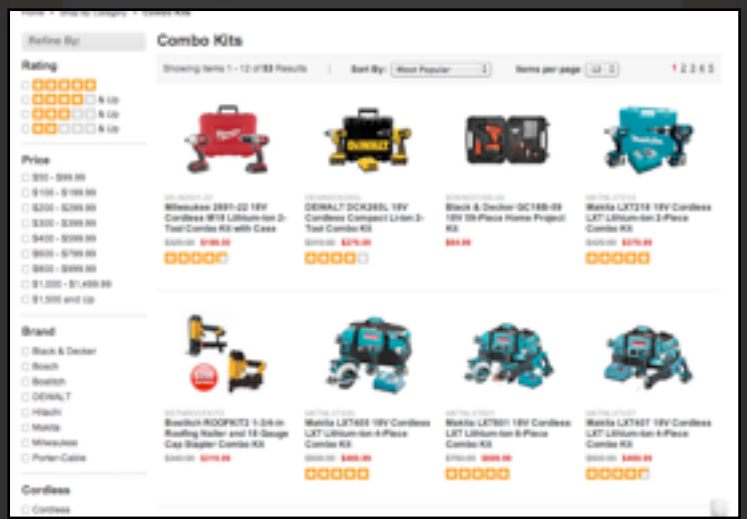


motivation:


delicious
Scrapy | An open source web scraping framework for Python


maru


34,688


Combo Kits

DESIRED PROPERTIES

DESIRED PROPERTIES

SPEED

CONSTRAINTS

CONSTRAINTS

- Politeness

CONSTRAINTS

- Politeness
- Distributed

CONSTRAINTS

- Politeness
- Distributed
  - Linear Scalability

CONSTRAINTS

- Politeness
- Distributed
  - Linear Scalability
  - Even partitioning

CONSTRAINTS

- Politeness
- Distributed
  - Linear Scalability
  - Even partitioning
  - Minimum overlap

CONSTRAINTS

- Politeness
- Distributed
  - Linear Scalability
  - Even partitioning
  - Minimum overlap

it's easy to burden small servers

CONSTRAINTS

- Politeness
- Distributed
  - Linear Scalability
  - Even partitioning
  - Minimum overlap

(for any significant crawl)

CONSTRAINTS

- Politeness
- Distributed
  - Linear Scalability
  - Even partitioning
  - Minimum overlap

n machines =
*n*m pages-per-second*

CONSTRAINTS

- Politeness
- Distributed
  - Linear Scalability
  - Even partitioning
  - Minimum overlap

every machine should perform equal work

CONSTRAINTS

- Politeness
- Distributed
  - Linear Scalability
  - Even partitioning
  - Minimum overlap

crawl each page exactly once

CONSTRAINTS

- Politeness
- Distributed
  - Linear Scalability
  - Even partitioning
  - Minimum overlap

BASIC ALGORITHM

```
Initialize:
    UrlsDone = null
    UrlFrontier = {'google.com/index.html', ..}
Repeat
    url = UrlFrontier.getNext()
    ip = DNSlookup(url.getHostname())
    html = DownloadPage(ip, url.getPath())
    UrlsDone.insert(url)
    newUrls = parseForLinks(html)
    For each newUrl
      If not UrlsDone.contains(newUrl)
      then UrlsTodo.insert(newUrl)
```

```
Initialize:
    UrlsDone = null
    UrlFrontier = {'google.com/index.html', ..}
Repeat
    url = UrlFrontier.getNext()
    ip = DNSlookup(url.getHostname())
    html = DownloadPage(ip, url.getPath())
    UrlsDone.insert(url)
    newUrls = parseForLinks(html)
    For each newUrl
      If not UrlsDone.contains(newUrl)
      then UrlsTodo.insert(newUrl)
```

```
Initialize:
    UrlsDone = null
    UrlFrontier = {'google.com/index.html', ..}
Repeat
    url = UrlFrontier.getNext()
    ip = DNSlookup(url.getHostname())
    html = DownloadPage(ip, url.getPath())
    UrlsDone.insert(url)
    newUrls = parseForLinks(html)
    For each newUrl
      If not UrlsDone.contains(newUrl)
      then UrlsTodo.insert(newUrl)
```

```
Initialize:
    UrlsDone = null
    UrlFrontier = {'google.com/index.html', ..}
Repeat
    url = UrlFrontier.getNext()
    ip = DNSlookup(url.getHostname())
    html = DownloadPage(ip, url.getPath())
    UrlsDone.insert(url)
    newUrls = parseForLinks(html)
    For each newUrl
      If not UrlsDone.contains(newUrl)
      then UrlsTodo.insert(newUrl)
```

```
Initialize:
    UrlsDone = null
    UrlFrontier = {'google.com/index.html', ..}
Repeat
    url = UrlFrontier.getNext()
    ip = DNSlookup(url.getHostname())
    html = DownloadPage(ip, url.getPath())
    UrlsDone.insert(url)
    newUrls = parseForLinks(html)
    For each newUrl
      If not UrlsDone.contains(newUrl)
      then UrlsTodo.insert(newUrl)
```

```
Initialize:
    UrlsDone = null
    UrlFrontier = {'google.com/index.html', ..}
Repeat
    url = UrlFrontier.getNext()
    ip = DNSlookup(url.getHostname())
    html = DownloadPage(ip, url.getPath())
    UrlsDone.insert(url)
    newUrls = parseForLinks(html)
    For each newUrl
      If not UrlsDone.contains(newUrl)
      then UrlsTodo.insert(newUrl)
```

```
Initialize:
    UrlsDone = null
    UrlFrontier = {'google.com/index.html', ..}
Repeat
    url = UrlFrontier.getNext()
    ip = DNSlookup(url.getHostname())
    html = DownloadPage(ip, url.getPath())
    UrlsDone.insert(url)
    newUrls = parseForLinks(html)
    For each newUrl
      If not UrlsDone.contains(newUrl)
      then UrlsTodo.insert(newUrl)
```

```
Initialize:
    UrlsDone = null
    UrlFrontier = {'google.com/index.html', ..}
Repeat
    url = UrlFrontier.getNext()
    ip = DNSlookup(url.getHostname())
    html = DownloadPage(ip, url.getPath())
    UrlsDone.insert(url)
    newUrls = parseForLinks(html)
    For each newUrl
      If not UrlsDone.contains(newUrl)
      then UrlsTodo.insert(newUrl)
```

```
Initialize:
    UrlsDone = null
    UrlFrontier = {'google.com/index.html', ..}
Repeat
    url = UrlFrontier.getNext()
    ip = DNSlookup(url.getHostname())
    html = DownloadPage(ip, url.getPath())
    UrlsDone.insert(url)
    newUrls = parseForLinks(html)
    For each newUrl
      If not UrlsDone.contains(newUrl)
      then UrlsTodo.insert(newUrl)
```

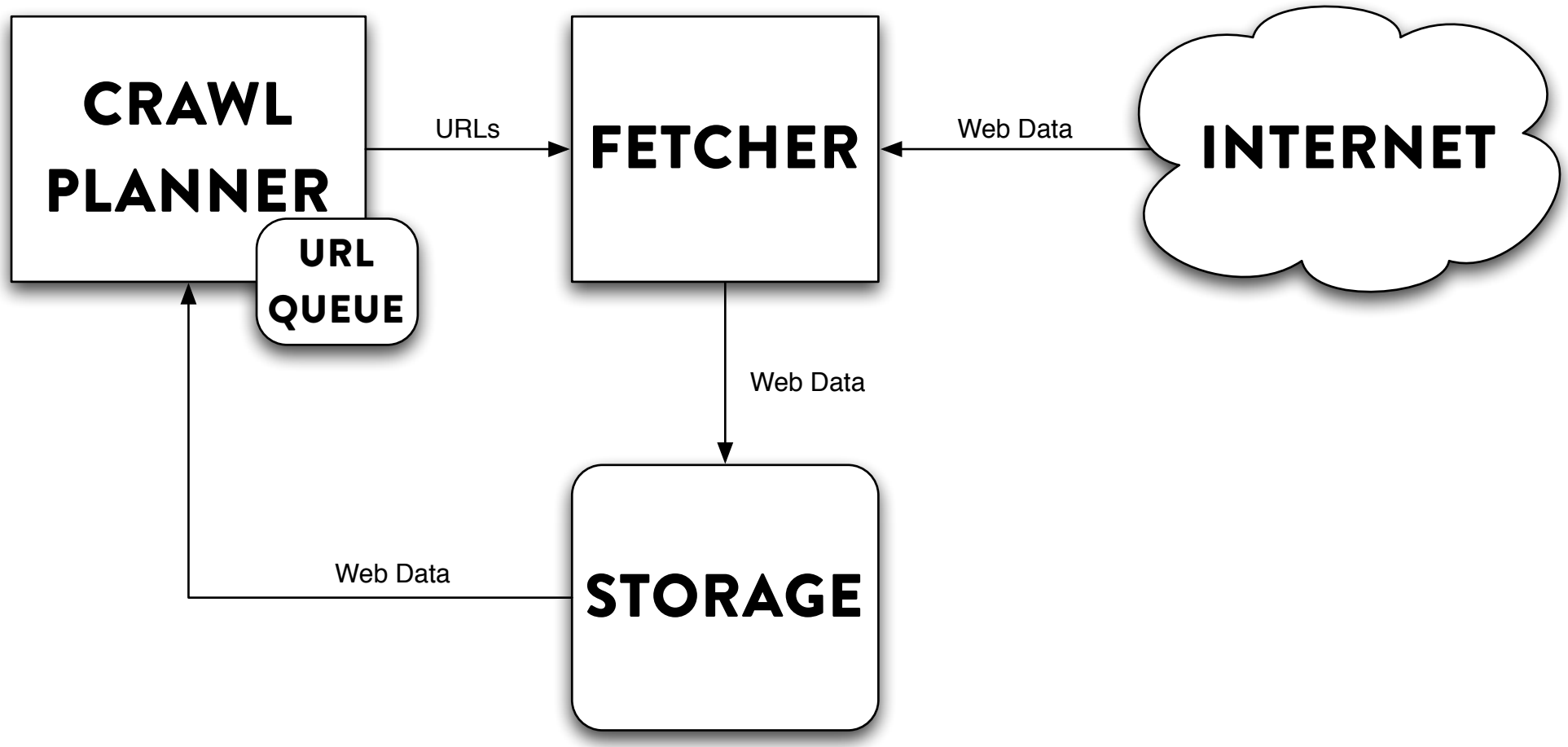
```
Initialize:
    UrlsDone = null
    UrlFrontier = {'google.com/index.html', ..}
Repeat
    url = UrlFrontier.getNext()
    ip = DNSlookup(url.getHostname())
    html = DownloadPage(ip, url.getPath())
    UrlsDone.insert(url)
    newUrls = parseForLinks(html)
    For each newUrl
      If not UrlsDone.contains(newUrl)
      then UrlsTodo.insert(newUrl)
```

```
Initialize:
    UrlsDone = null
    UrlFrontier = {'google.com/index.html', ..}
Repeat
    url = UrlFrontier.getNext()
    ip = DNSlookup(url.getHostname())
    html = DownloadPage(ip, url.getPath())
    UrlsDone.insert(url)
    newUrls = parseForLinks(html)
    For each newUrl
      If not UrlsDone.contains(newUrl)
      then UrlsTodo.insert(newUrl)
```

```
Initialize:
    UrlsDone = null
    UrlFrontier = {'google.com/index.html', ..}
Repeat
    url = UrlFrontier.getNext()
    ip = DNSlookup(url.getHostname())
    html = DownloadPage(ip, url.getPath())
    UrlsDone.insert(url)
    newUrls = parseForLinks(html)
    For each newUrl
        If not UrlsDone.contains(newUrl)
        then UrlsTodo.insert(newUrl)
```

```
Initialize:
    UrlsDone = null
    UrlFrontier = {'google.com/index.html', ..}
Repeat
    url = UrlFrontier.getNext()
    ip = DNSlookup(url.getHostname())
    html = DownloadPage(ip, url.getPath())
    UrlsDone.insert(url)
    newUrls = parseForLinks(html)
    For each newUrl
      If not UrlsDone.contains(newUrl)
      then UrlsTodo.insert(newUrl)
```

architecture overview



CHALLENGES

challenges:

depends on your ambitions

Google's Index Size:

1998 - 26 million

2005 - 8 billion

2008 - 1 trillion

http://www.nytimes.com/2005/08/15/technology/15search.html
http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html

challenges:

small crawls are easy

challenges:

< 10MM

small crawls are easy

challenges:

large crawls are interesting

challenges:

DNS Lookup

challenges:

DNS Lookup

URLs Crawled

challenges:

DNS Lookup

URLs Crawled

Politeness

challenges:

DNS Lookup

URLs Crawled

Politeness

URL Frontier

challenges:

DNS Lookup

URLs Crawled

Politeness

URL Frontier

Queueing URLs

challenges:

DNS Lookup

URLs Crawled

Politeness

URL Frontier

Queueing URLs

Extracting URLs

challenges:

DNS LOOKUP

```
Initialize:
    UrlsDone = null
    UrlFrontier = {'google.com/index.html', ..}
Repeat
    url = UrlFrontier.getNext()
    ip = DNSlookup(url.getHostname())
    html = DownloadPage(ip, url.getPath())
    UrlsDone.insert(url)
    newUrls = parseForLinks(html)
    For each newUrl
      If not UrlsDone.contains(newUrl)
      then UrlsTodo.insert(newUrl)
```

challenges:

▸ DNS LOOKUP

can easily be a bottleneck

challenges:

▸ DNS LOOKUP

- consider running your own DNS servers
  - djbdns
  - PowerDNS
  - etc.

challenges:

▸ DNS LOOKUP

- **be aware of software limitations**
  - `gethostbyaddr` **is synchronized**
  - **same with many "default" DNS clients**

challenges:

▸ DNS LOOKUP

You'll know when you need it

challenges:

URLs CRAWLED

```
Initialize:
    UrlsDone = null
    UrlFrontier = {'google.com/index.html', ..}
Repeat
    url = UrlFrontier.getNext()
    ip = DNSlookup(url.getHostname())
    html = DownloadPage(ip, url.getPath())
    UrlsDone.insert(url)
    newUrls = parseForLinks(html)
    For each newUrl
      If not UrlsDone.contains(newUrl)
      then UrlsTodo.insert(newUrl)
```

challenges:

▸ URLs CRAWLED

1 machine, store in memory

challenges:

▸ URLs CRAWLED

1 machine, store in memory

NAPKIN CALCULATION

challenges:

▸ URLs CRAWLED

1 machine, store in memory

NAPKIN CALCULATION

~50 bytes per URL

e.g. http://wiki.apache.org/cassandra/ArticlesAndPresentations

challenges:

▸ URLs CRAWLED

1 machine, store in memory

NAPKIN CALCULATION

~50 bytes per URL

e.g. http://wiki.apache.org/cassandra/ArticlesAndPresentations

+8 bytes for time-last-crawled

as long e.g. System.currentTimeMillis() -> 1314392455712

challenges:

▸ URLs CRAWLED

1 machine, store in memory

NAPKIN CALCULATION

~50 bytes per URL

e.g. http://wiki.apache.org/cassandra/ArticlesAndPresentations

+8 bytes for time-last-crawled

as long e.g. System.currentTimeMillis() -> 1314392455712

x 100 million

challenges:

▸ URLs CRAWLED

1 machine, store in memory

NAPKIN CALCULATION

~50 bytes per URL

e.g. http://wiki.apache.org/cassandra/ArticlesAndPresentations

+8 bytes for time-last-crawled

as long e.g. System.currentTimeMillis() -> 1314392455712

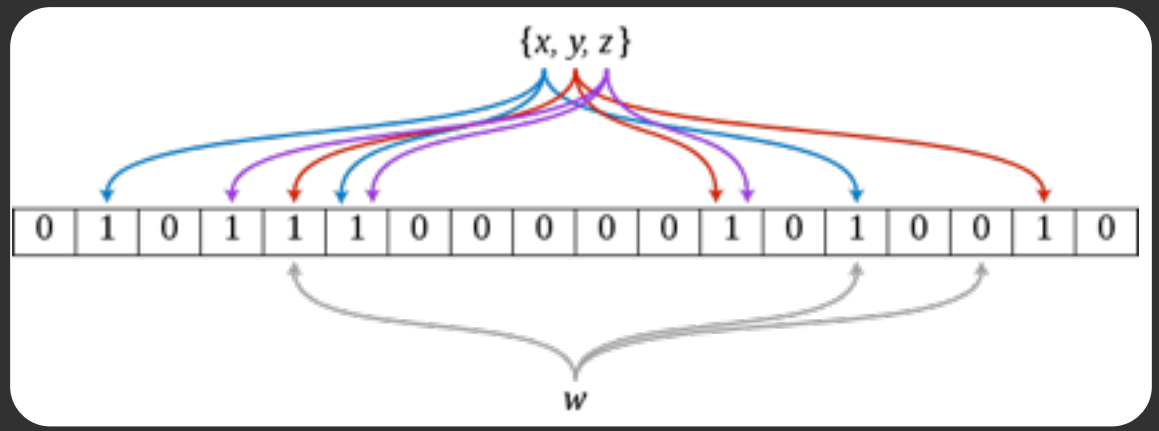
x 100 million

=~ **5.4 gigabytes**

can we do better?

BLOOM FILTERS

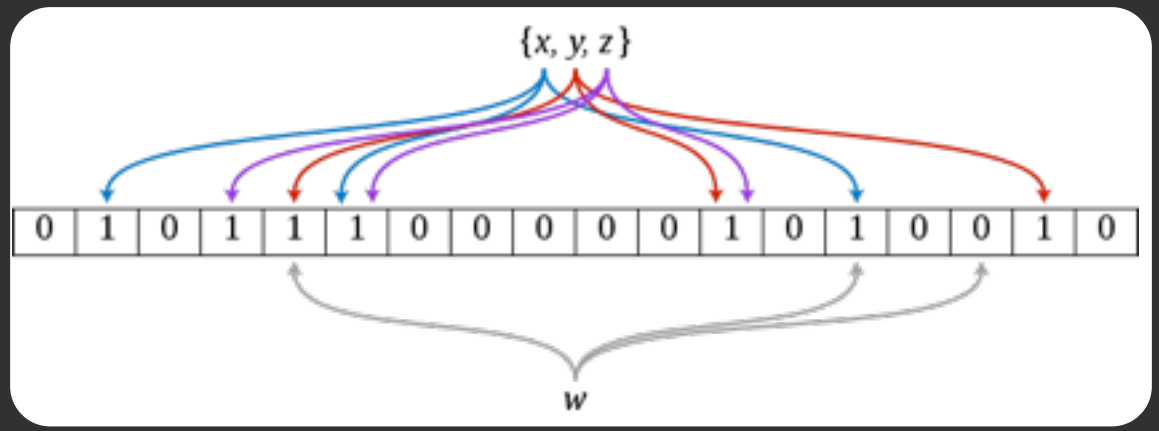
BLOOM FILTERS



answers the question:

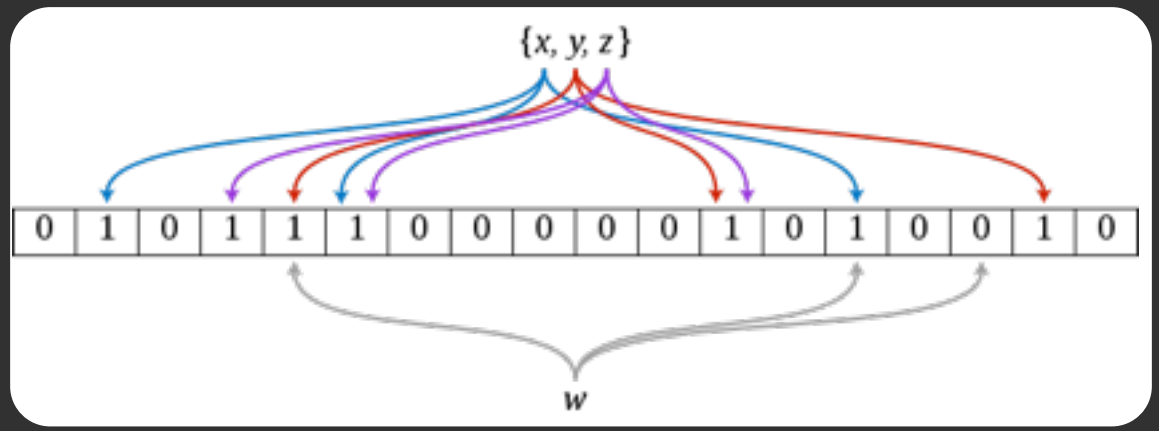
is this item in the set?

BLOOM FILTERS



answers either:

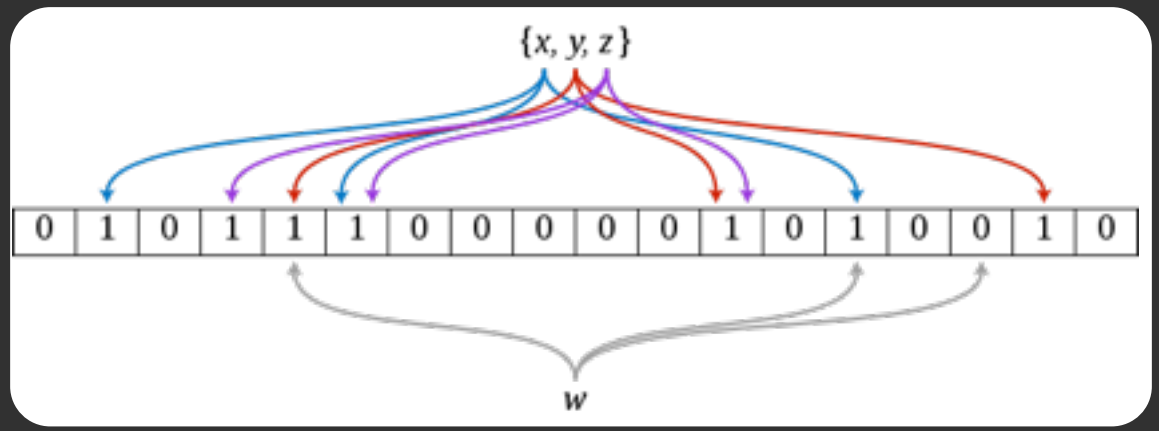
BLOOM FILTERS



answers either:

- yes, probably

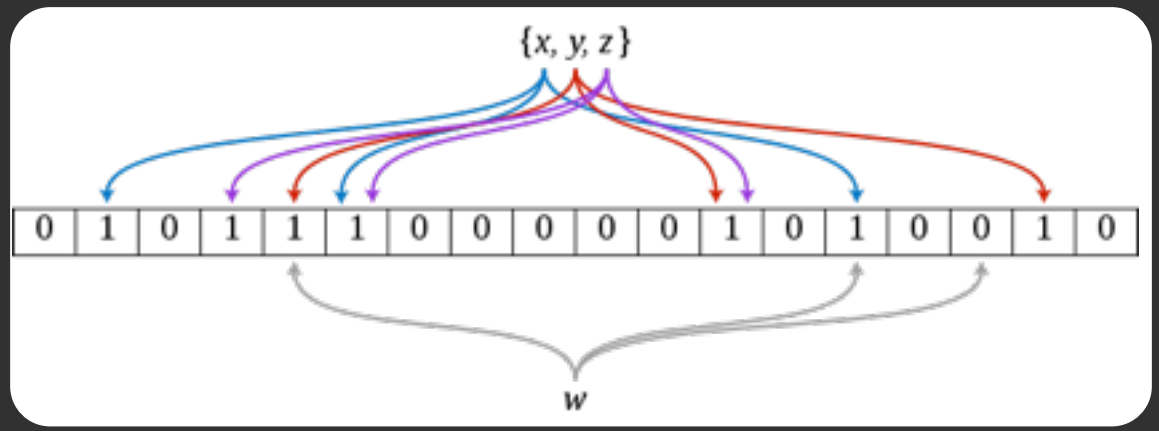
BLOOM FILTERS



answers either:

- yes, probably
- definitely not

BLOOM FILTERS

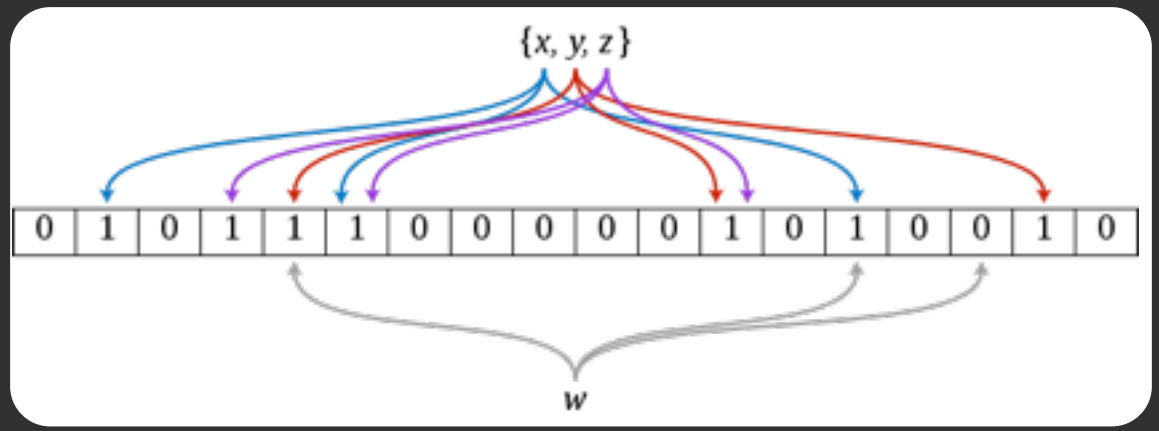


Have we crawled: `http://www.xcombinator.com`?

answers either:

- yes, probably
- definitely not

BLOOM FILTERS



Have we crawled: `http://www.xcombinator.com`?

answers either:

- yes, probably
- definitely not

challenges:

▸ URLs CRAWLED

1 machine, bloom filter

100 million URLs

1 in 100 million chance

of false positive

see: http://hur.st/bloomfilter?n=100000000&p=1.0E-8

challenges:

▸ URLs CRAWLED

1 machine, bloom filter

NAPKIN CALCULATION

100 million URLs

1 in 100 million chance

of false positive

see: http://hur.st/bloomfilter?n=100000000&p=1.0E-8

challenges:

▸ URLs CRAWLED

1 machine, bloom filter

NAPKIN CALCULATION

100 million URLs

1 in 100 million chance
of false positive

=~ **457 megabytes**

see: http://hur.st/bloomfilter?n=100000000&p=1.0E-8

▸ BLOOM FILTER

▸ BLOOM FILTER

drawbacks

▸ BLOOM FILTER

drawbacks

- probabilistic - occasional errors

▸ BLOOM FILTER

drawbacks

- probabilistic - occasional errors
- estimate # of items ahead of time

▸ BLOOM FILTER

drawbacks

- probabilistic - occasional errors
- estimate # of items ahead of time
- can’t delete

▸ BLOOM FILTER

drawbacks

solutions

- probabilistic - occasional errors
- estimate # of items ahead of time
- can’t delete

▸ BLOOM FILTER

drawbacks

solutions

- probabilistic - occasional errors
  - acceptable
- estimate # of items ahead of time
- can’t delete

▸ BLOOM FILTER

drawbacks

solutions

- probabilistic - occasional errors
  - acceptable
- estimate # of items ahead of time
  - not hard, see Dynamic BFs
- can’t delete

▸ BLOOM FILTER

drawbacks | **solutions**

- probabilistic - occasional errors
  - acceptable
- estimate # of items ahead of time
  - not hard, see Dynamic BFs
- can't delete
  - pick granularity (days)

▸ BLOOM FILTER

drawbacks

solutions

- probabilistic - occasional errors
  - acceptable
- estimate # of items ahead of time
  - not hard, see Dynamic BFs
- can't delete
  - pick granularity (days)
  - cascade them

BLOOM FILTERS

references:

http://en.wikipedia.org/wiki/Bloom_filter
http://spyced.blogspot.com/2009/01/all-you-ever-wanted-to-know-about.html
http://www.igvita.com/2010/01/06/flow-analysis-time-based-bloom-filters/

challenges:

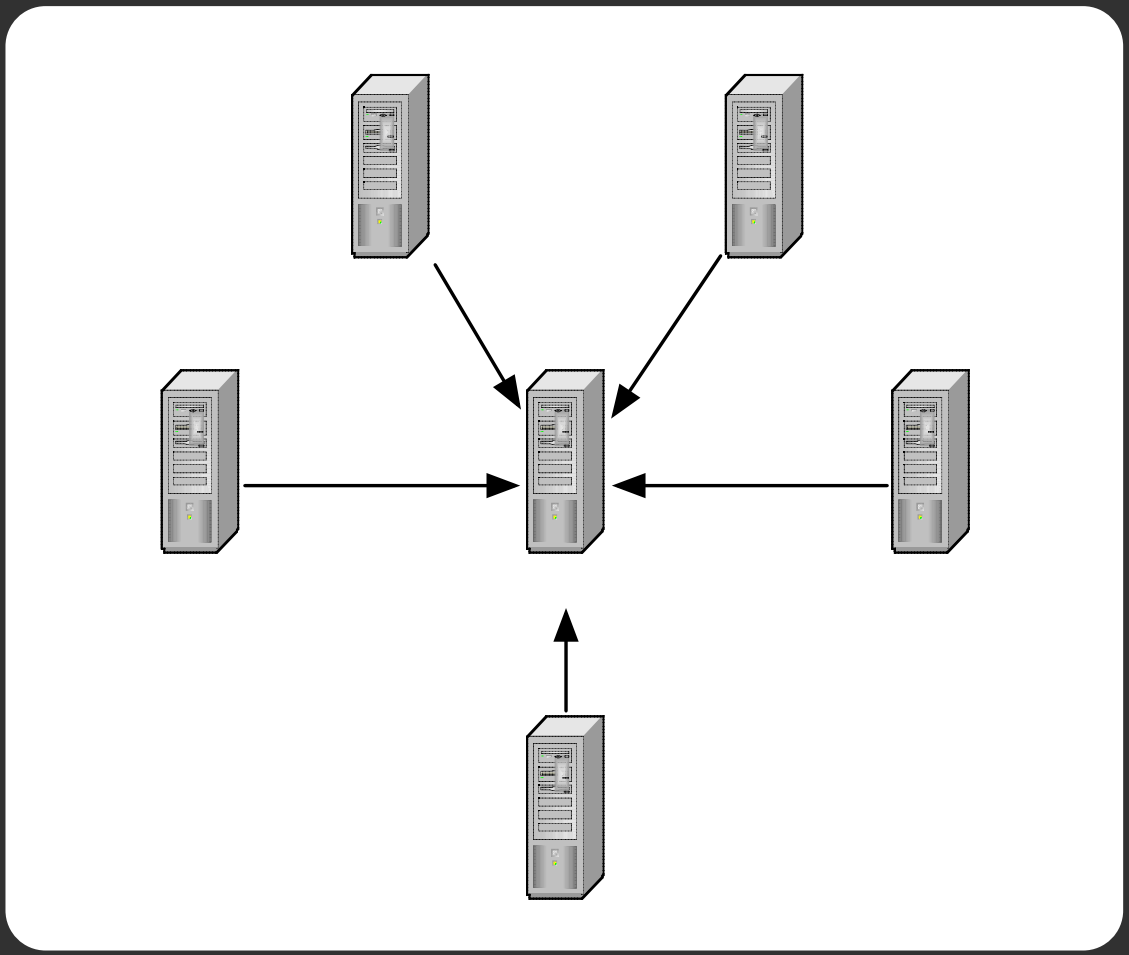
POLITENESS

obey `robots.txt`

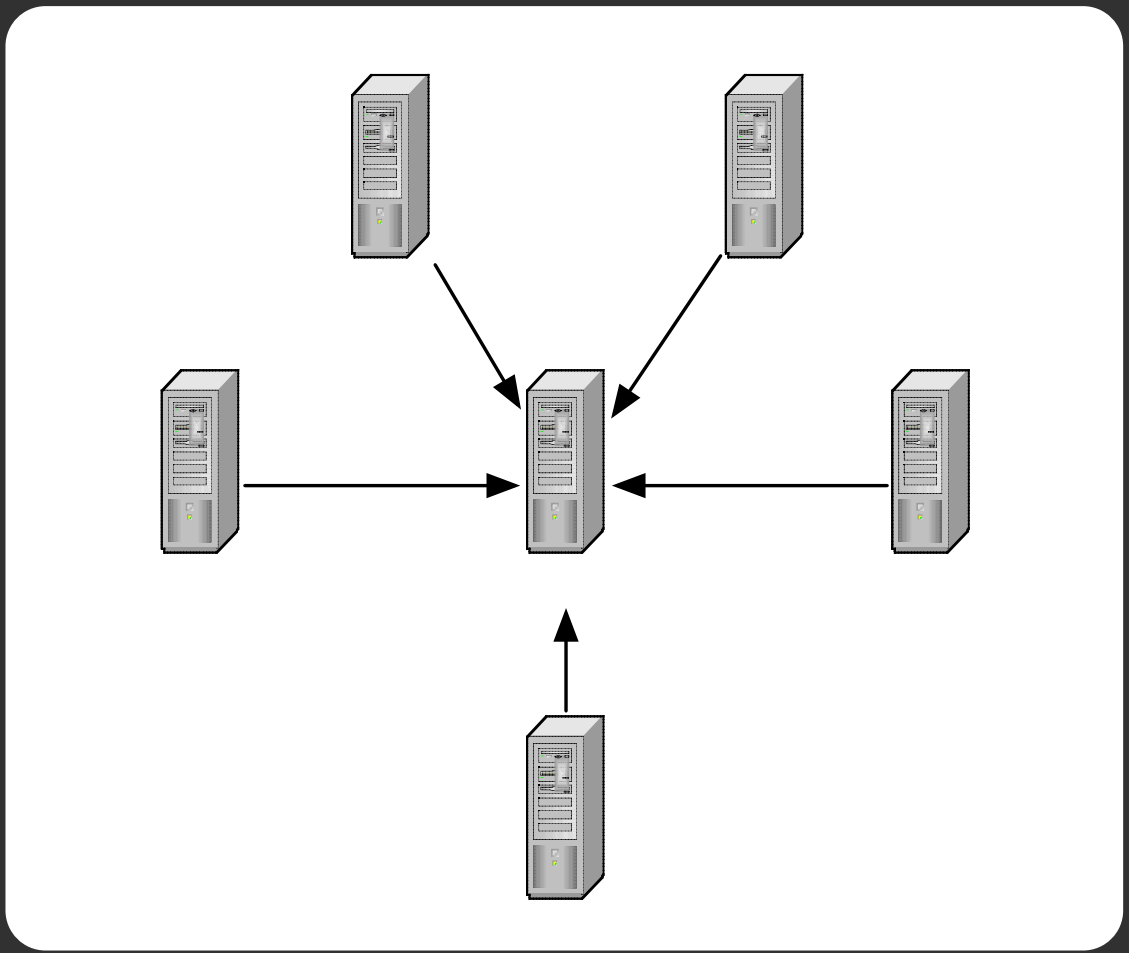
rule of thumb:

wait 2 seconds (*w.r.t. ip*)

centralized politeness

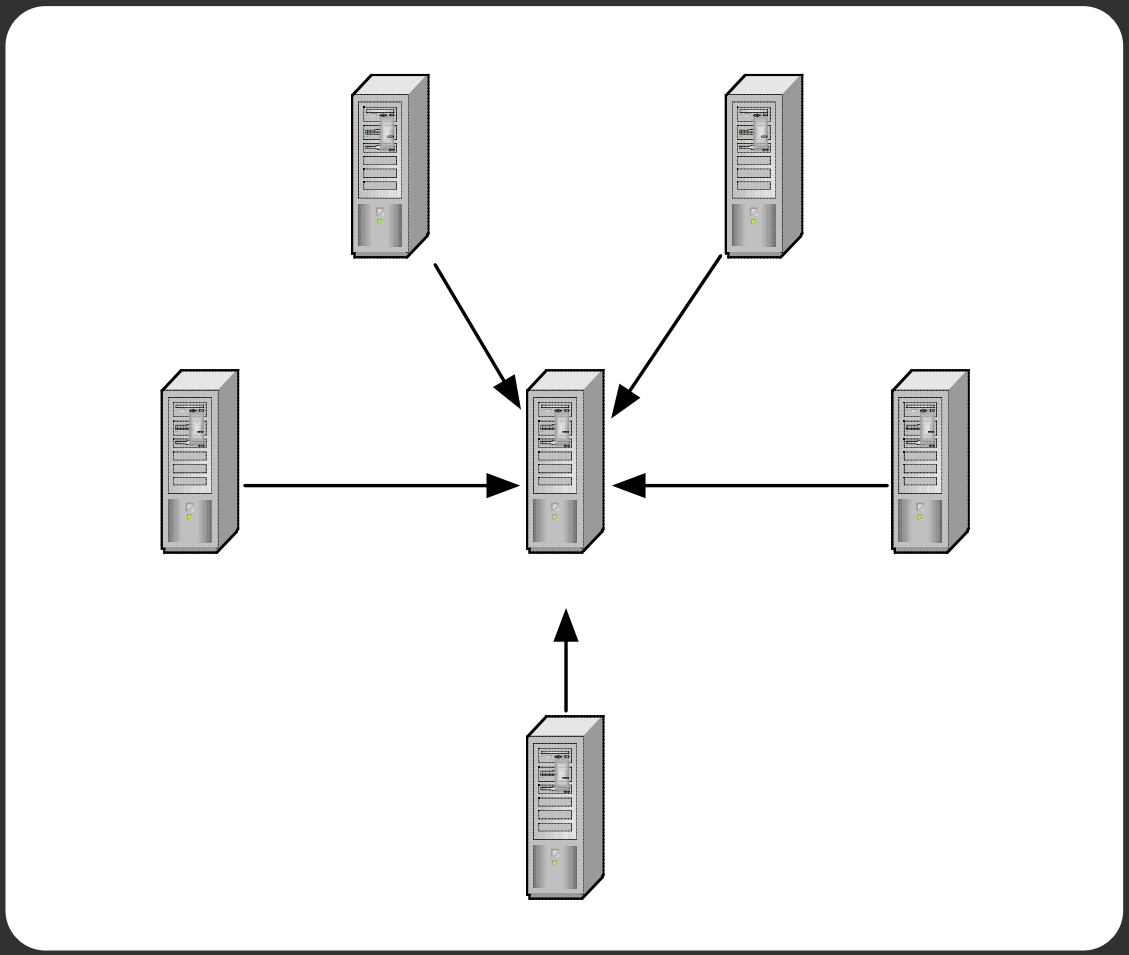


centralized politeness



SPOF

centralized politeness



SPOF

contention

challenges:

▸ POLITENESS

challenges:

▸ POLITENESS

- Options:

challenges:

▸ POLITENESS

- Options:
  - central database

challenges:

▸ POLITENESS

- Options:
  - central database
  - distributed locks (paxos/sigma/zookeeper)

challenges:

▸ POLITENESS

- Options:
  - central database
  - distributed locks (paxos/sigma/zookeeper)
  - controlled URL distribution

challenges:

▸ POLITENESS

- Options:
  - central database
  - distributed locks (paxos/sigma/zookeeper)
  - controlled URL distribution

http://en.wikipedia.org/wiki/Paxos_(computer_science)

challenges:

▸ POLITENESS

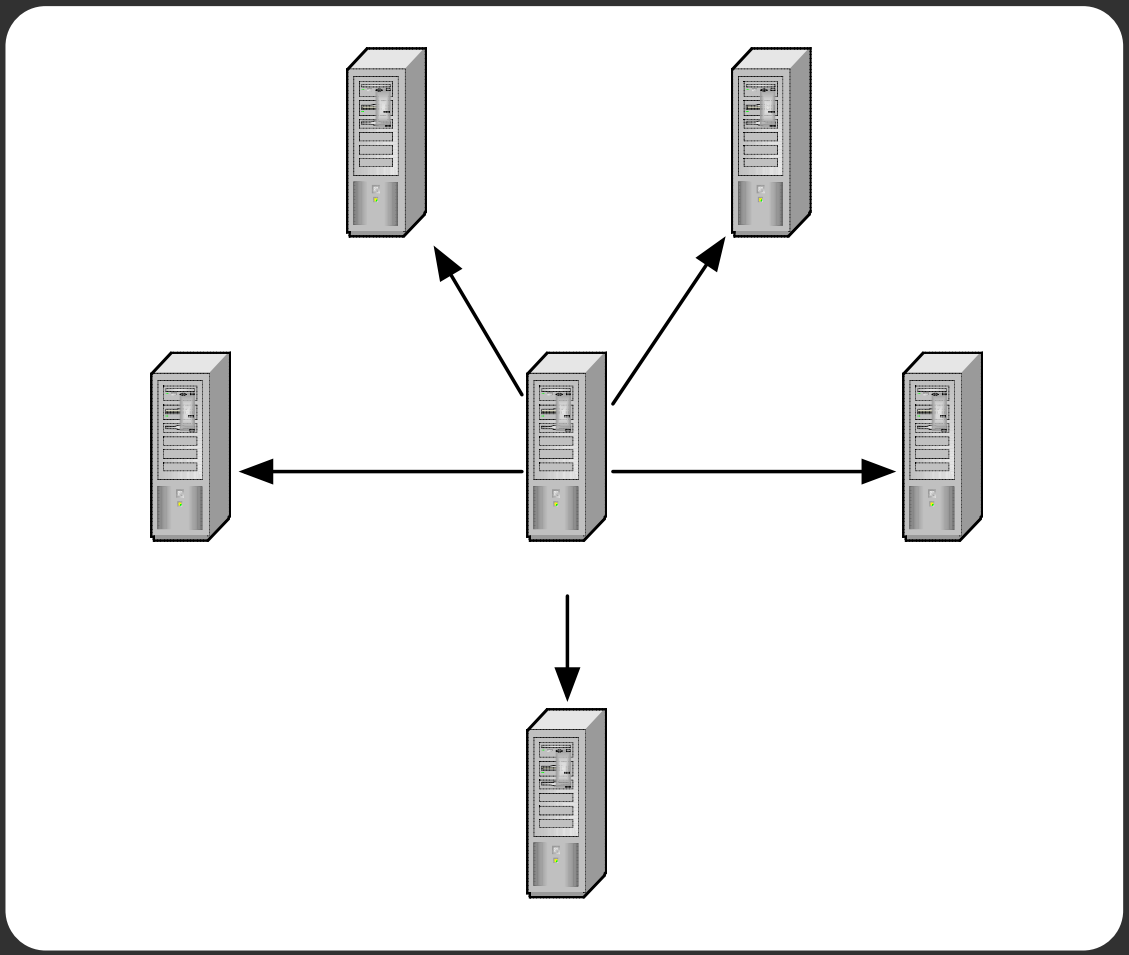
- Options:
  - central database
  - distributed locks (paxos/sigma/zookeeper)
  - controlled URL distribution

http://en.wikipedia.org/wiki/Paxos_(computer_science)
http://zookeeper.apache.org/

challenges:

URL FRONTIER

url frontier



idea:

consistently distribute URLs based on IP

modulo

IP	SHA-1	bucket (mod 5)
174.132.225.106	4dd14b0b...	2
74.125.224.115	cf4b7594...	1
157.166.255.19	0ac4d141...	4
69.22.138.129	6c1584fa...	4
98.139.50.166	327252c5...	3

benefits:

same IP *always* goes to same machine

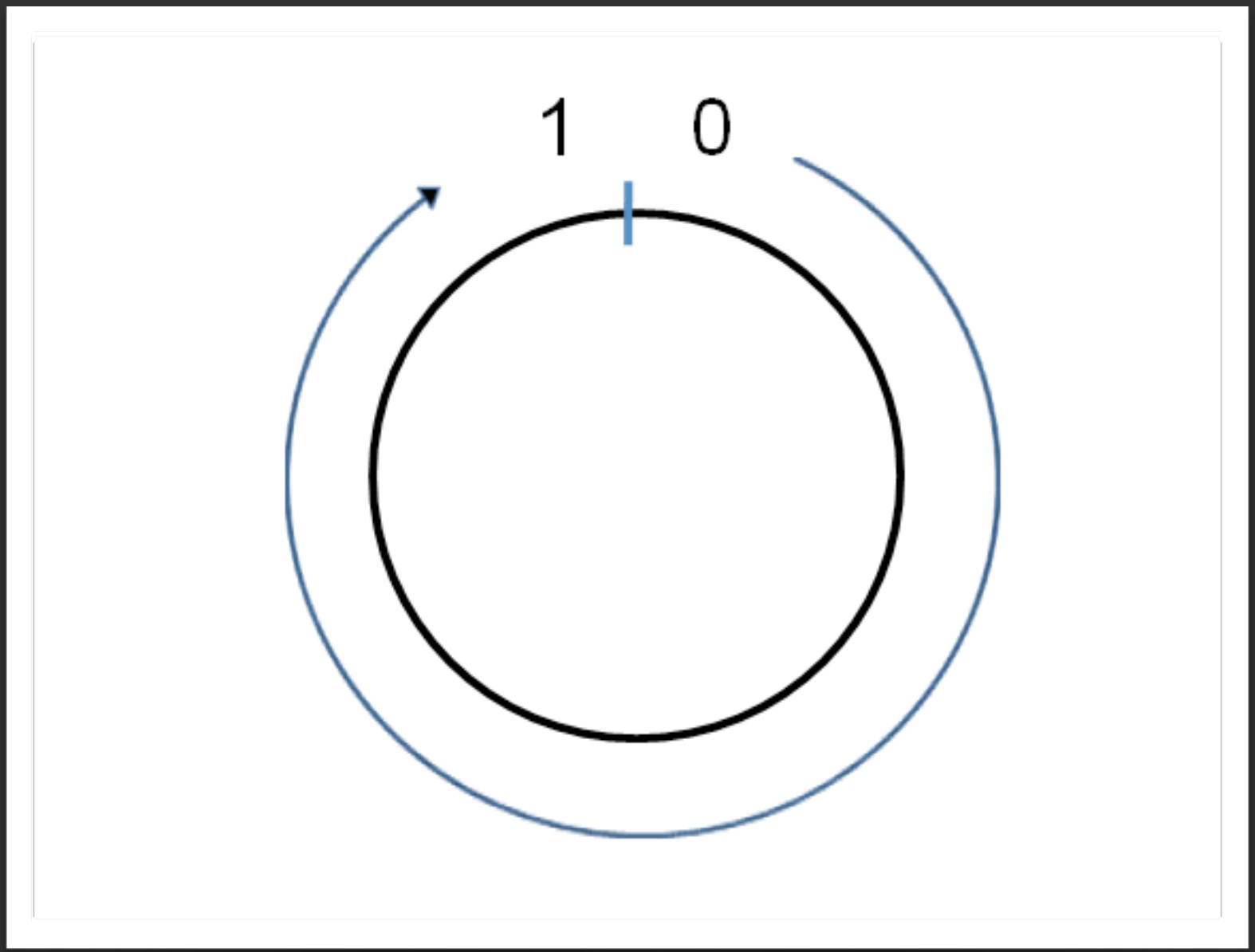
simple

drawbacks:

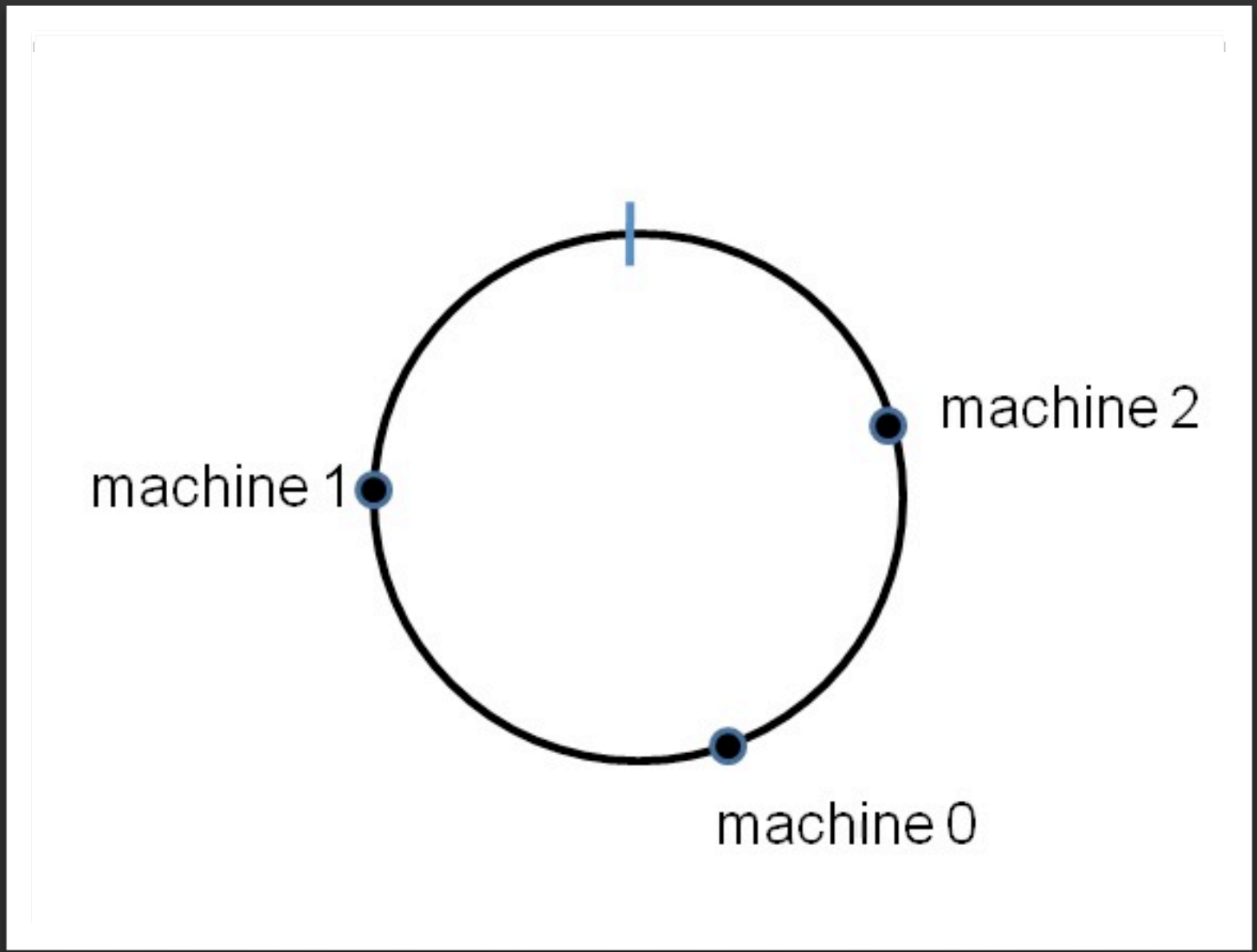
susceptible to skew

can't add / remove nodes without pain

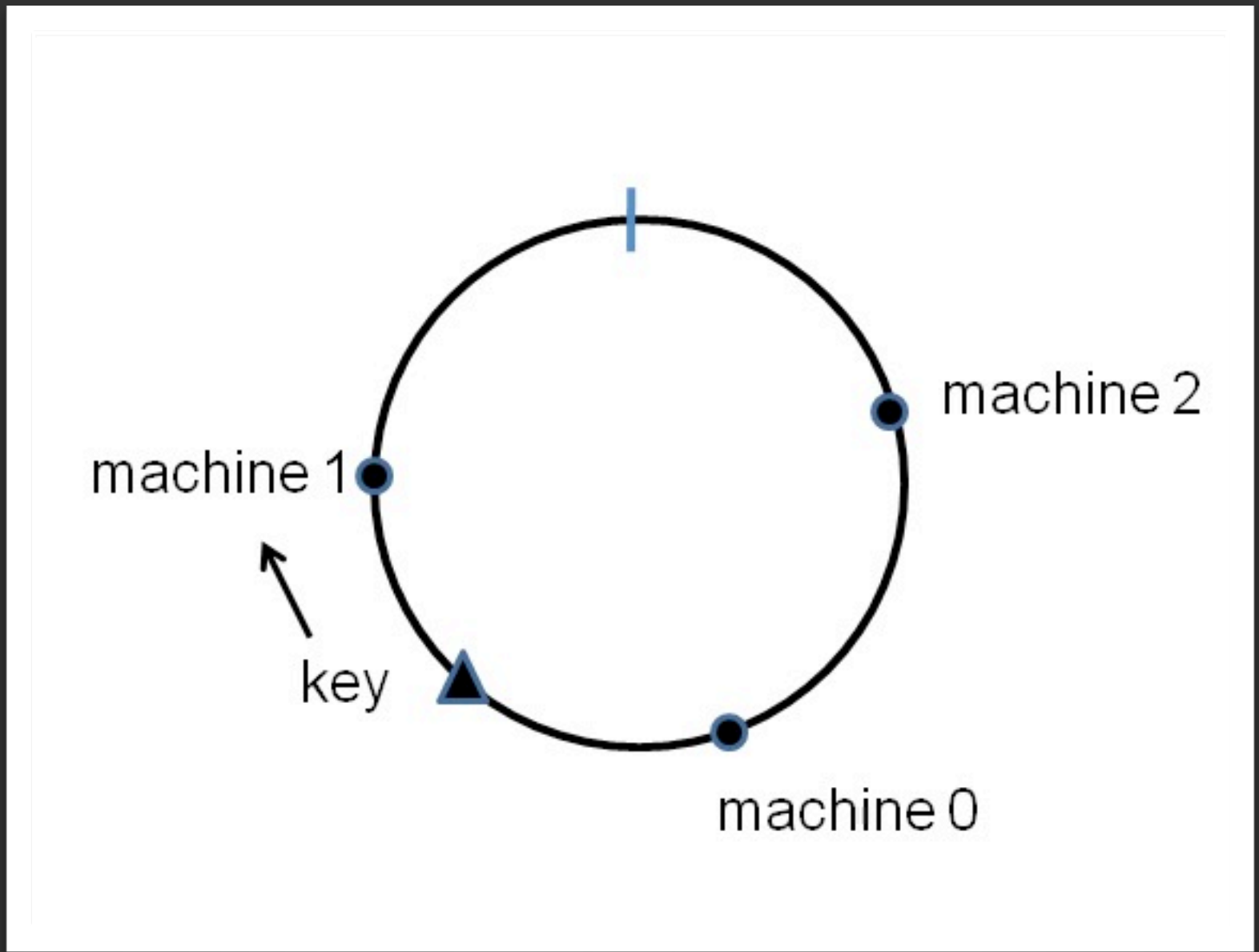
consistent hashing



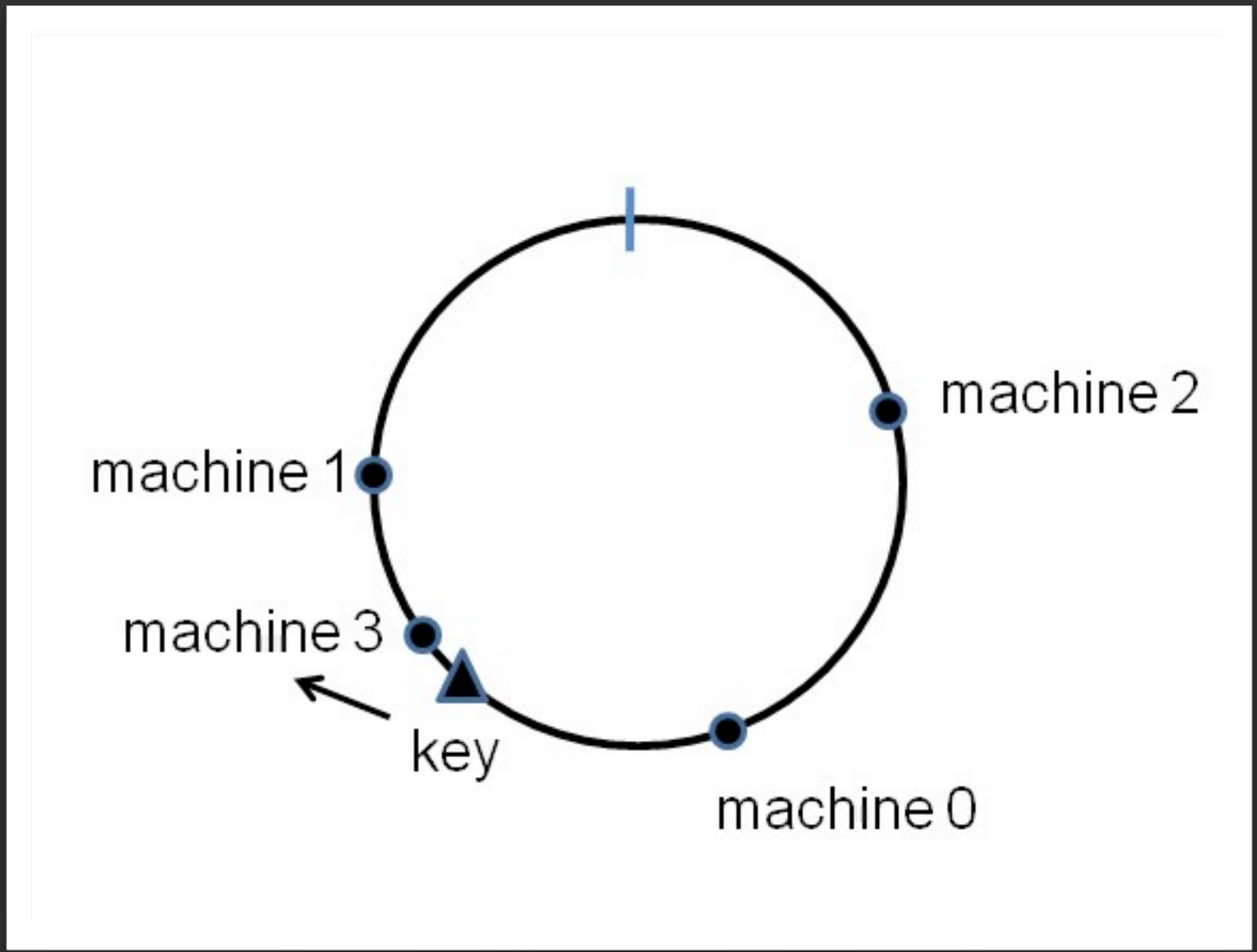
source: http://michaelnielsen.org/blog/consistent-hashing/



source: http://michaelnielsen.org/blog/consistent-hashing/



source: http://michaelnielsen.org/blog/consistent-hashing/



source: http://michaelnielsen.org/blog/consistent-hashing/

benefits:

~ 1/(n+1) URLs move on add/remove

virtual nodes help skew

robust (no SOP)

drawbacks:

naive solution won't work for large sites

further reading:

Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications (2001) Stoica et al.

Dynamo: Amazon's Highly Available Key-value Store, SOSP 2007

Tapestry: A Resilient Global-Scale Overlay for Service Deployment (2004) Zhao et al.

challenges:

QUEUEING URLS

situation:

situation:

URL

situation:

URL

not recently crawled

situation:

URL

not recently crawled

allowed by robots.txt

situation:

URL

not recently crawled

allowed by robots.txt

polite

how to you order them?

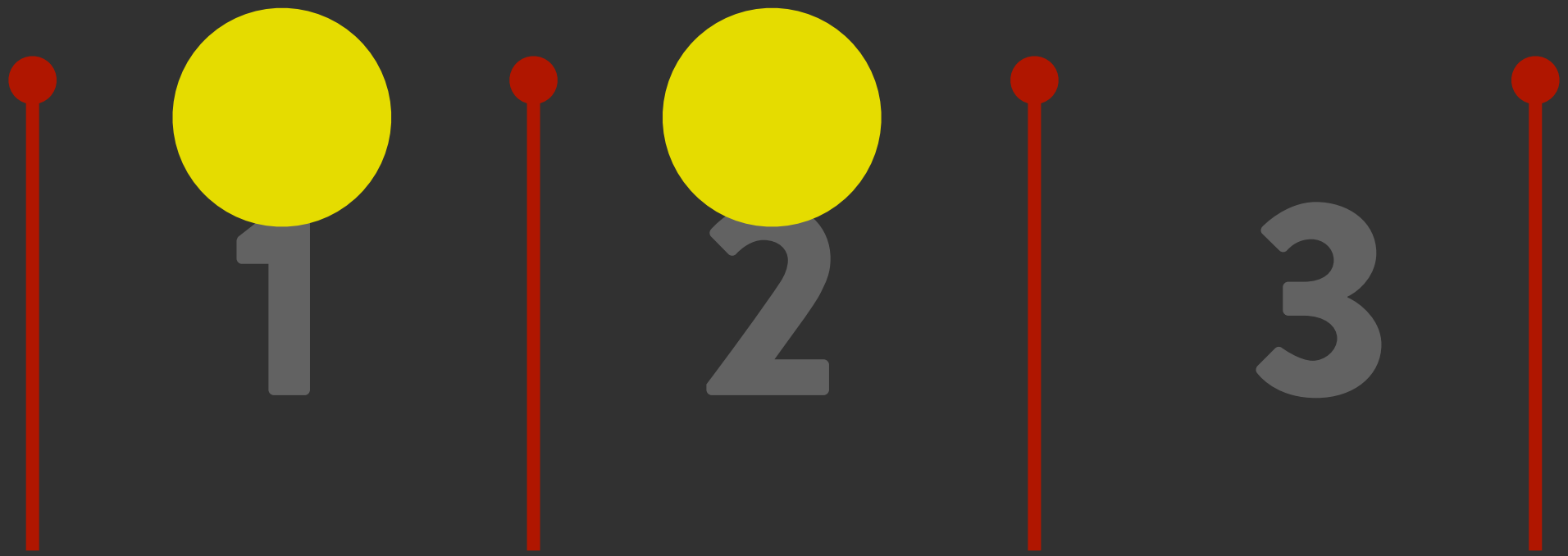
(within a single machine)

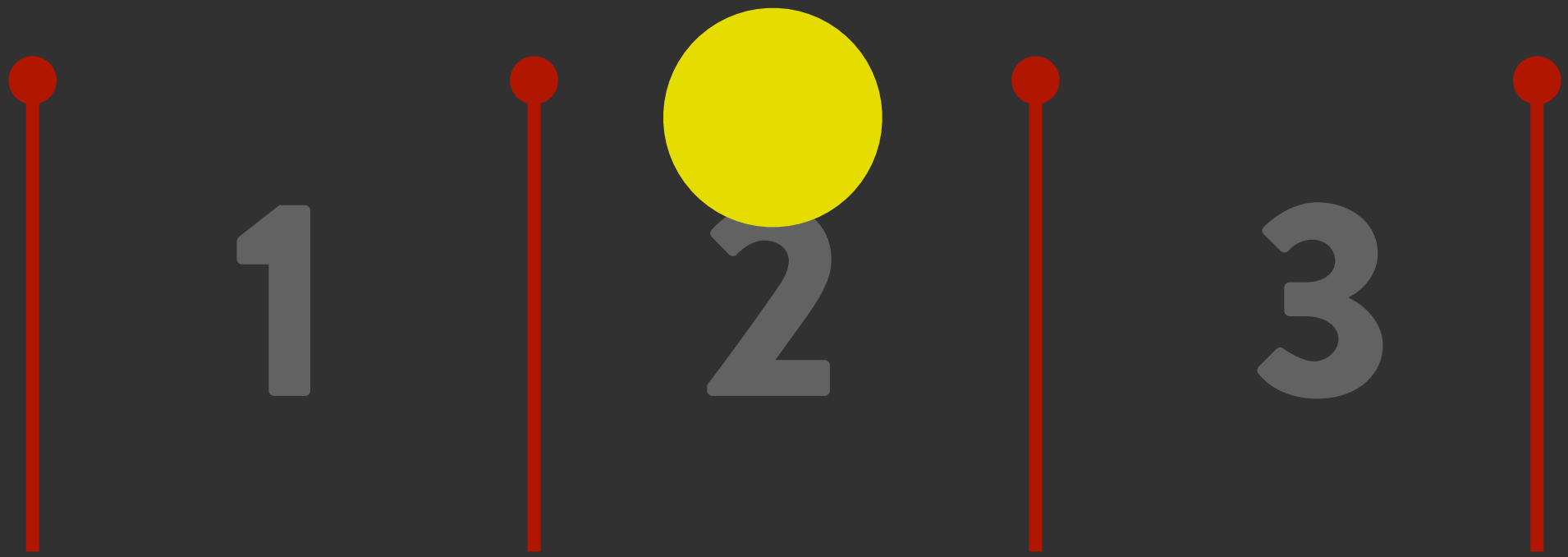
hash each lane:

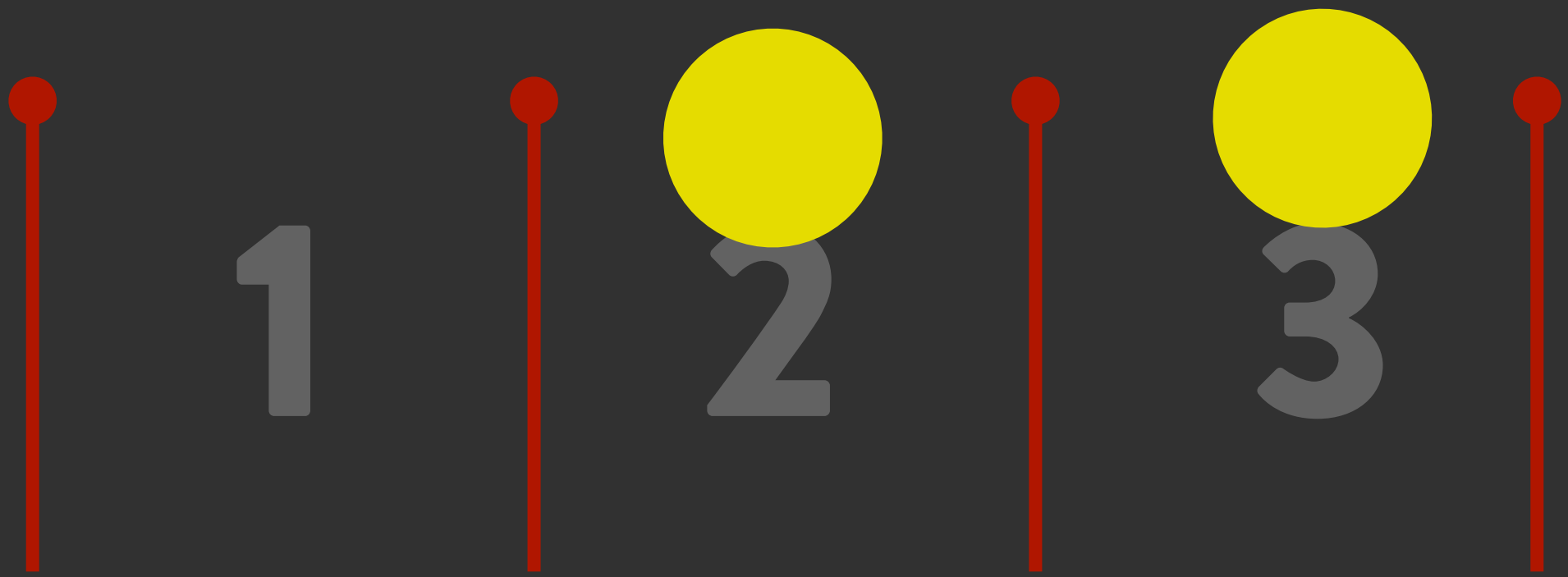
http://yachtmaintenanceco.com/
http://www.amsterdamports.nl/
http://www.4s-dawn.com/
http://www.embassysuiteslittlerock.com/
http://members.tripod.com/airfields_freeman/NM/Airfields_NM_NW.htm
http://mdgroover.iweb.bsu.edu
http://music.imbc.com/
http://www.robertjbradshaw.com
http://www.kerkattenhoven.be
http://www.escolania.org/

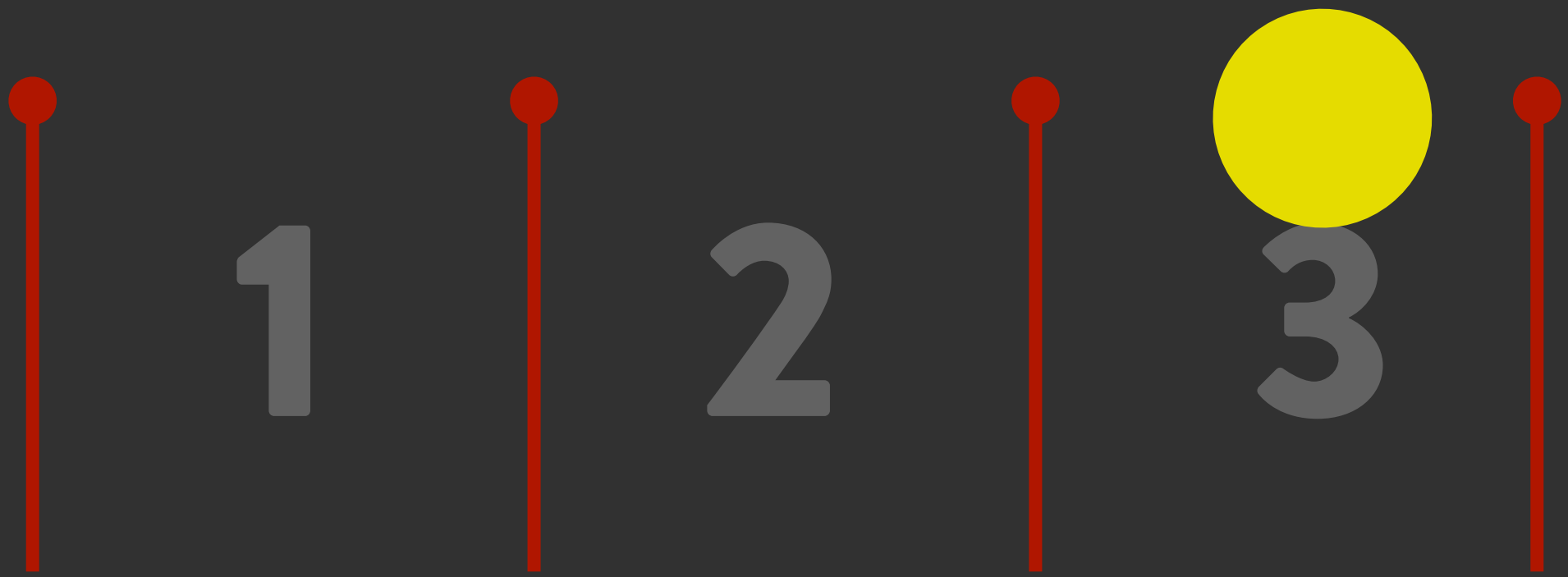

1
2
3


1
2
3


1
2
3


1
2
3

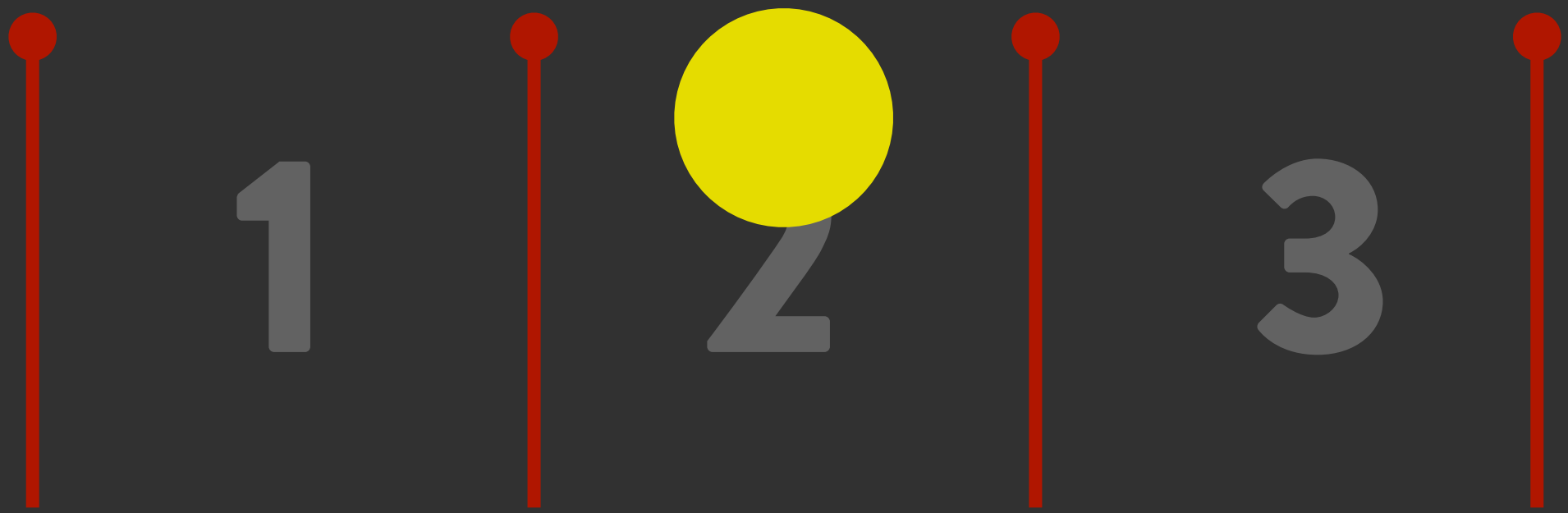

1
2
3

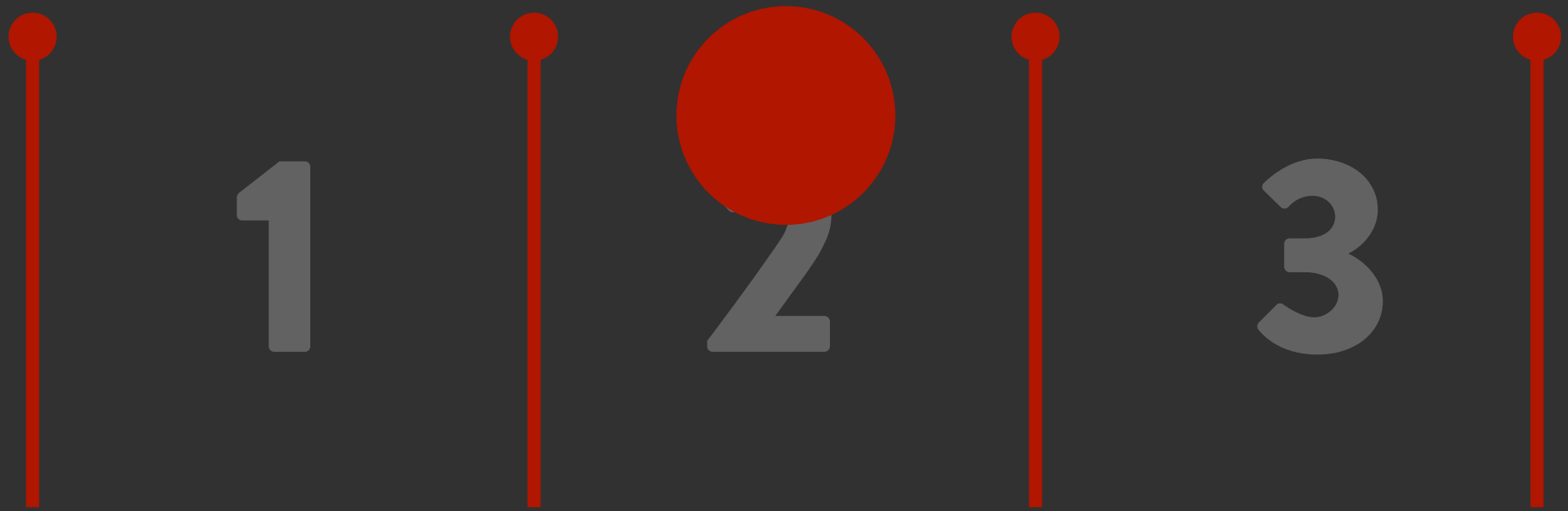

1
2
3

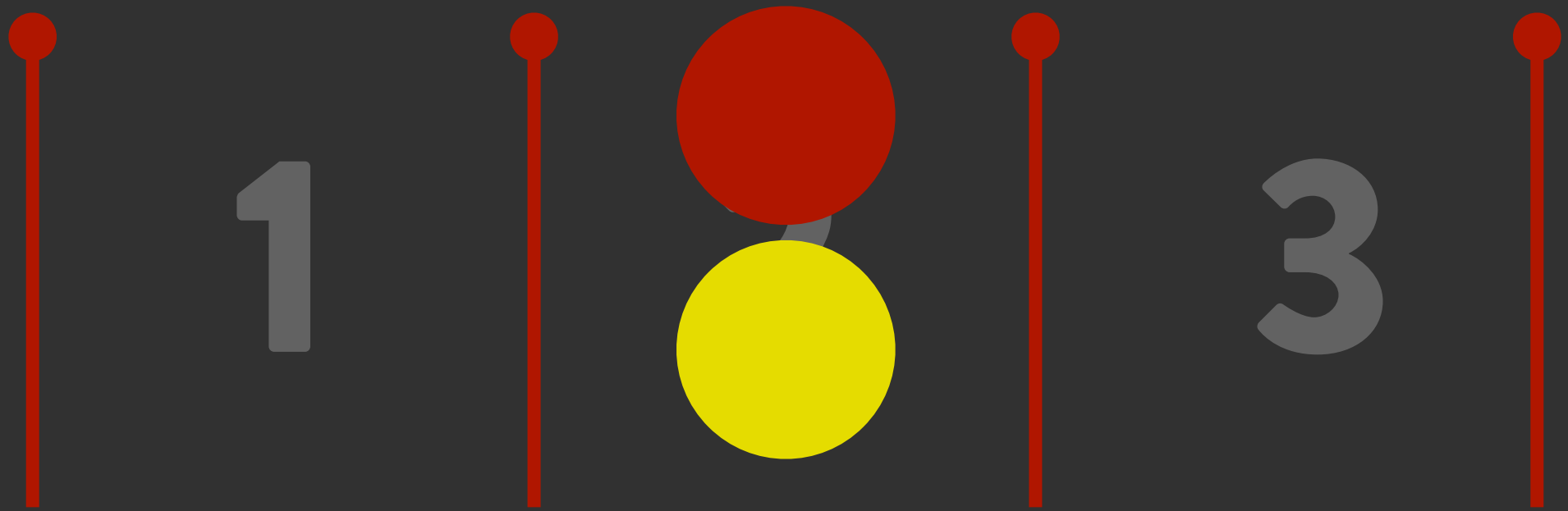

1
2
3

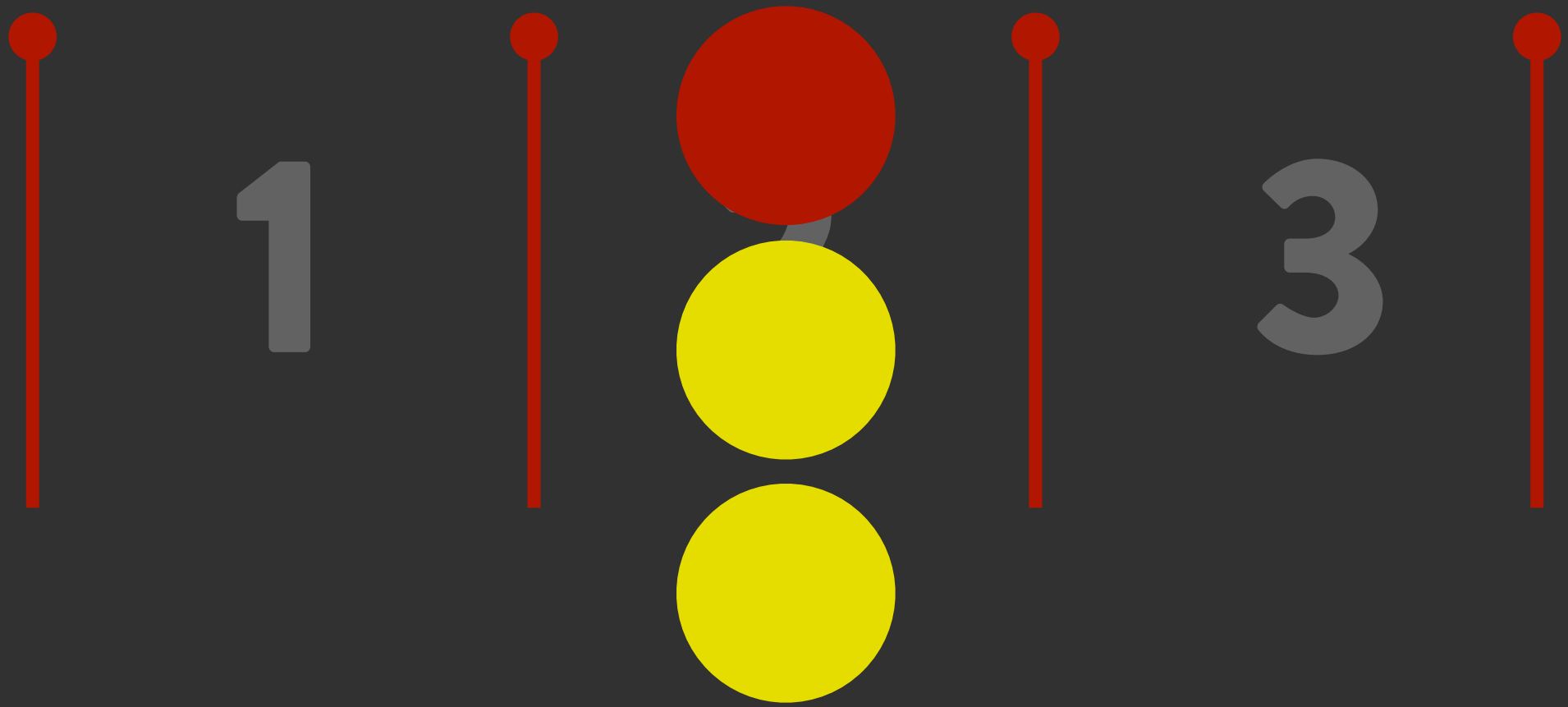

1
2
3

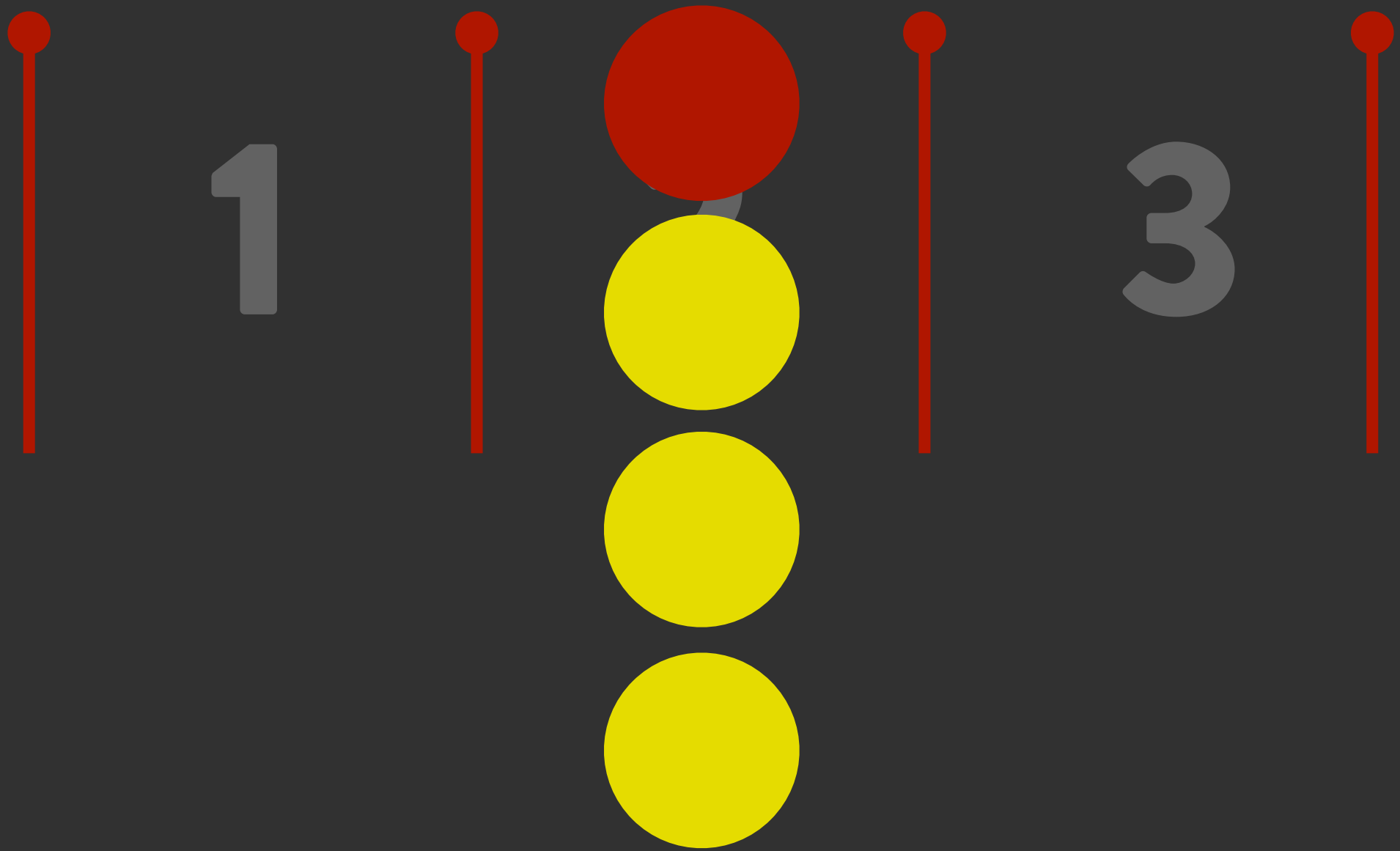

1
2
3

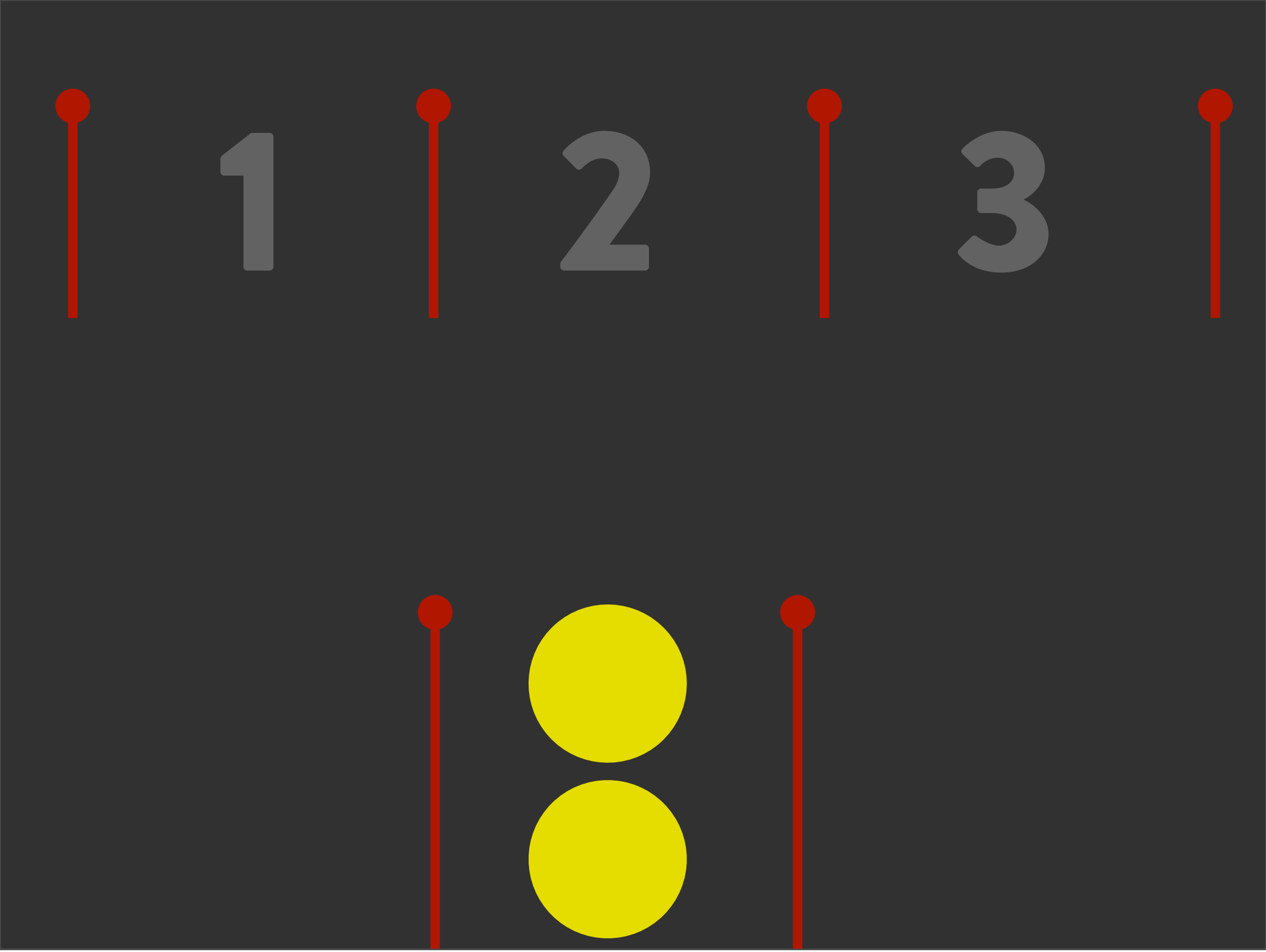

1
2
3

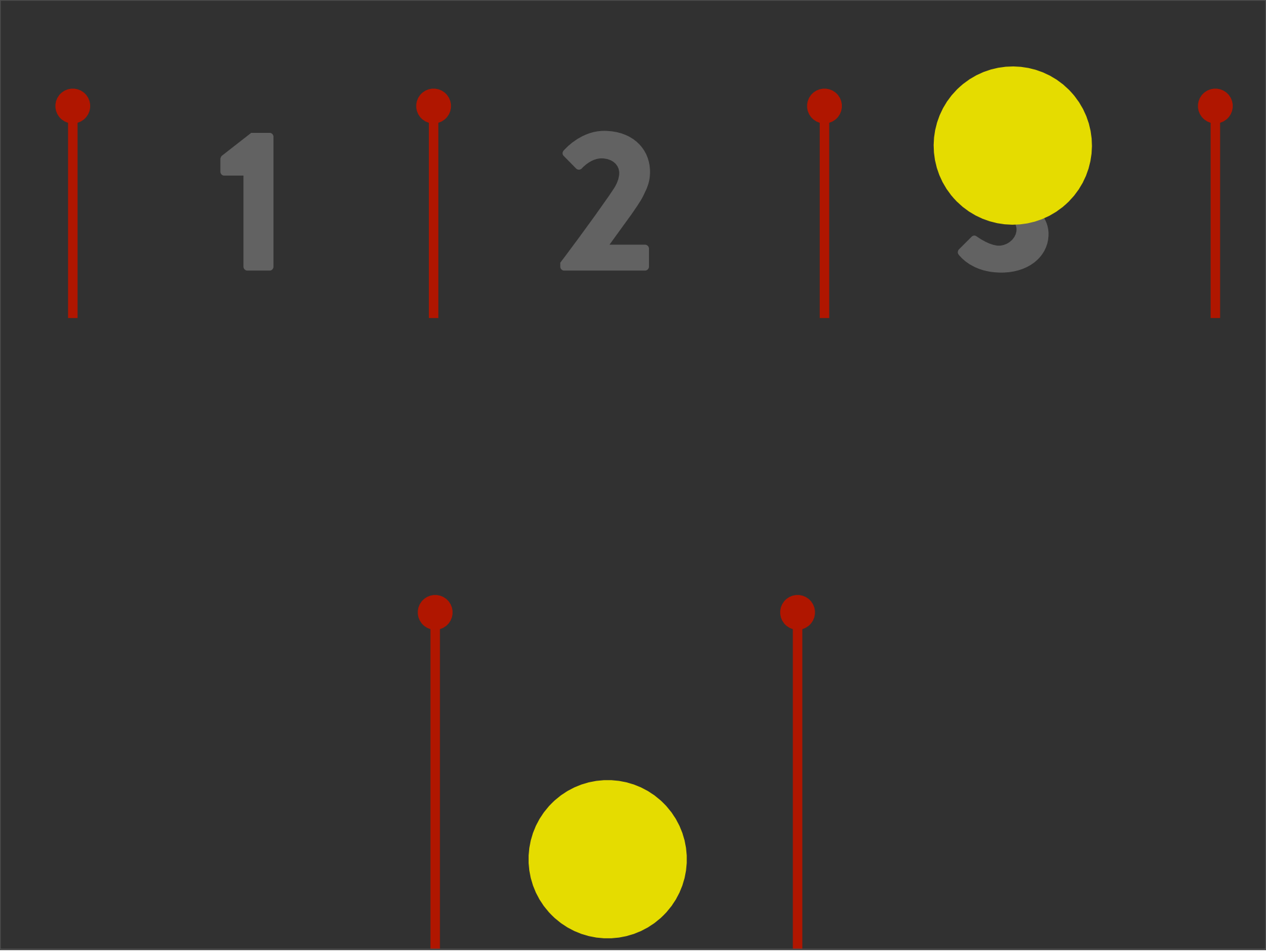

1
2
3

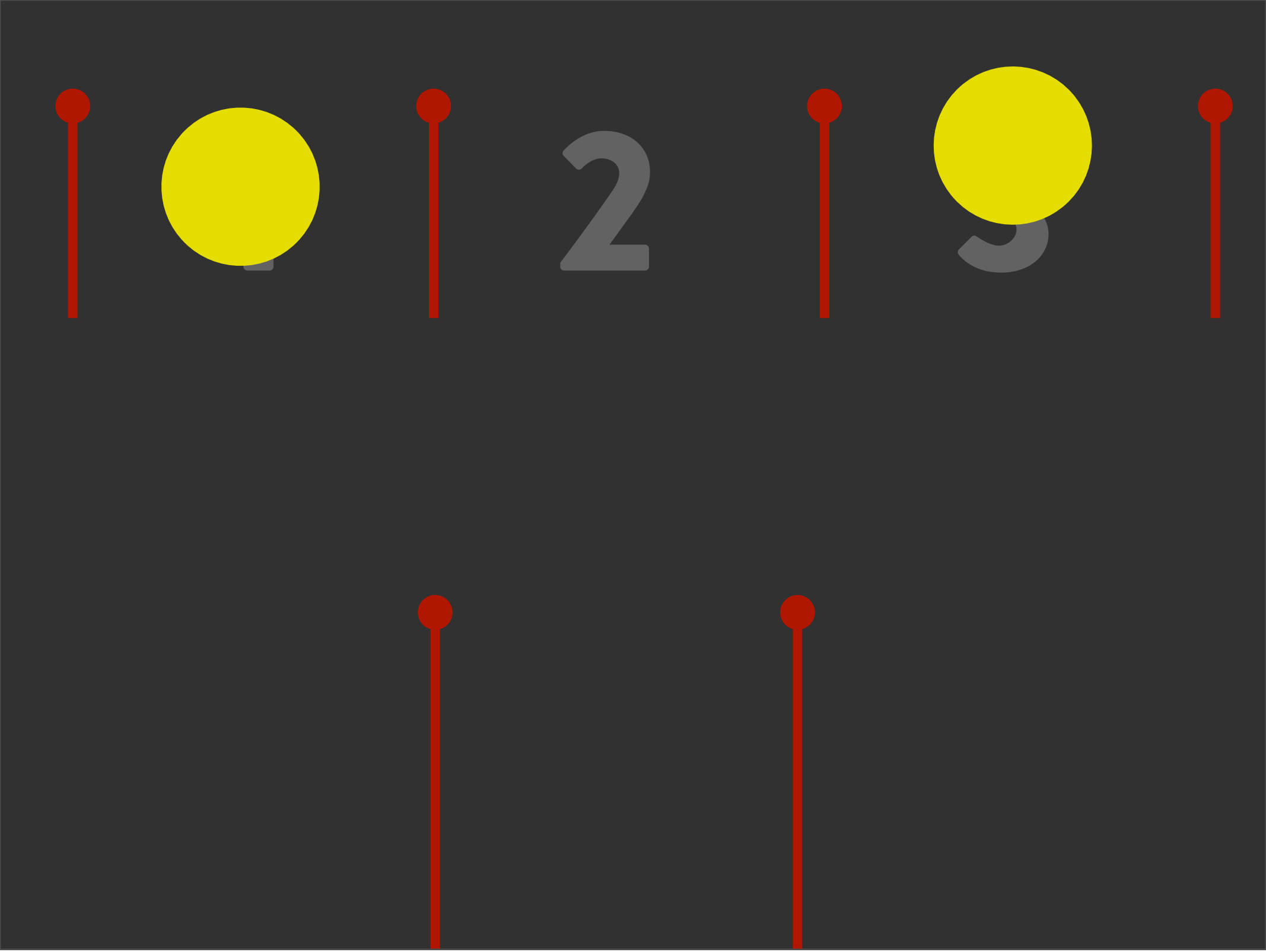

1
8
3

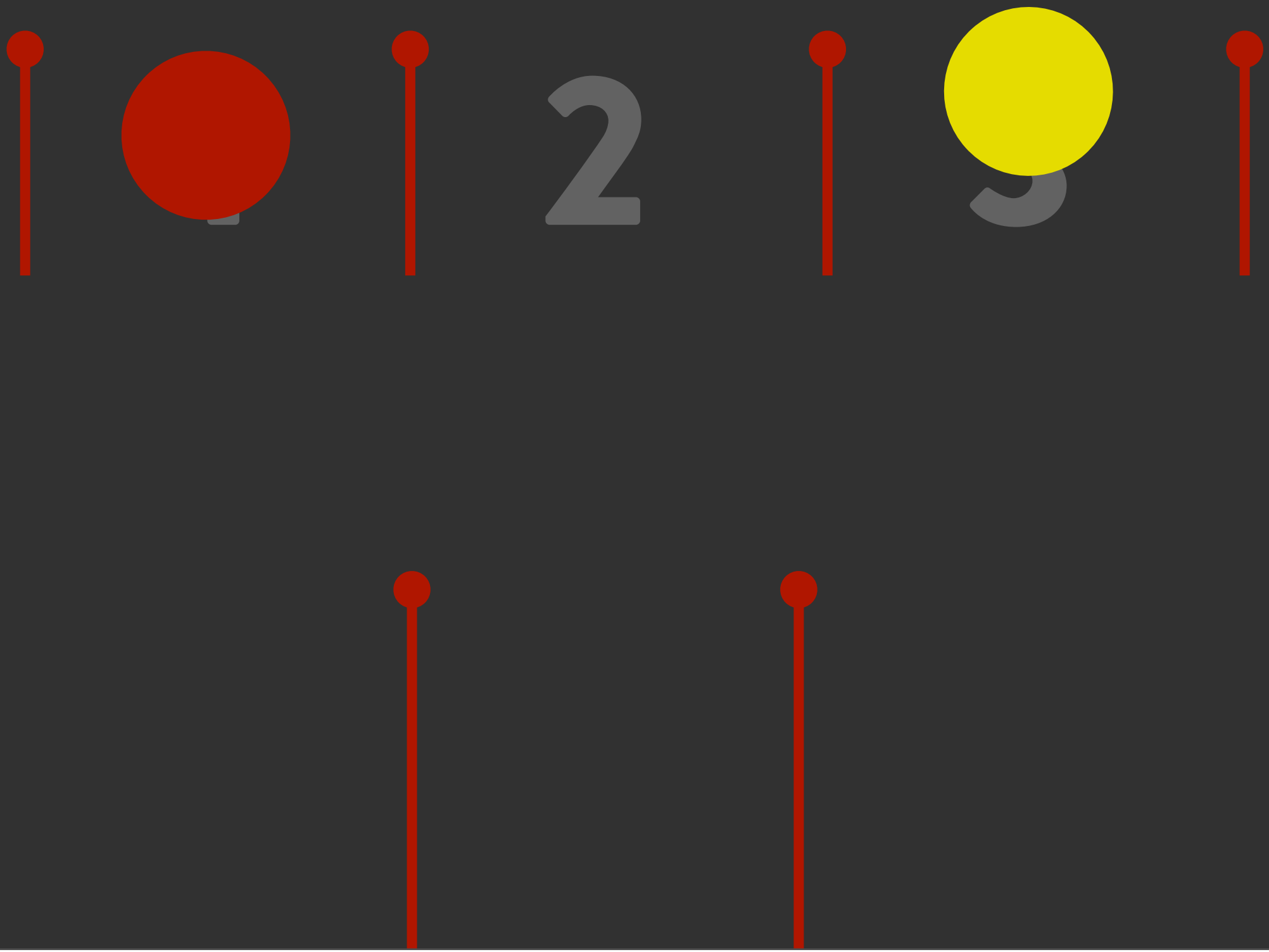

1
2
3

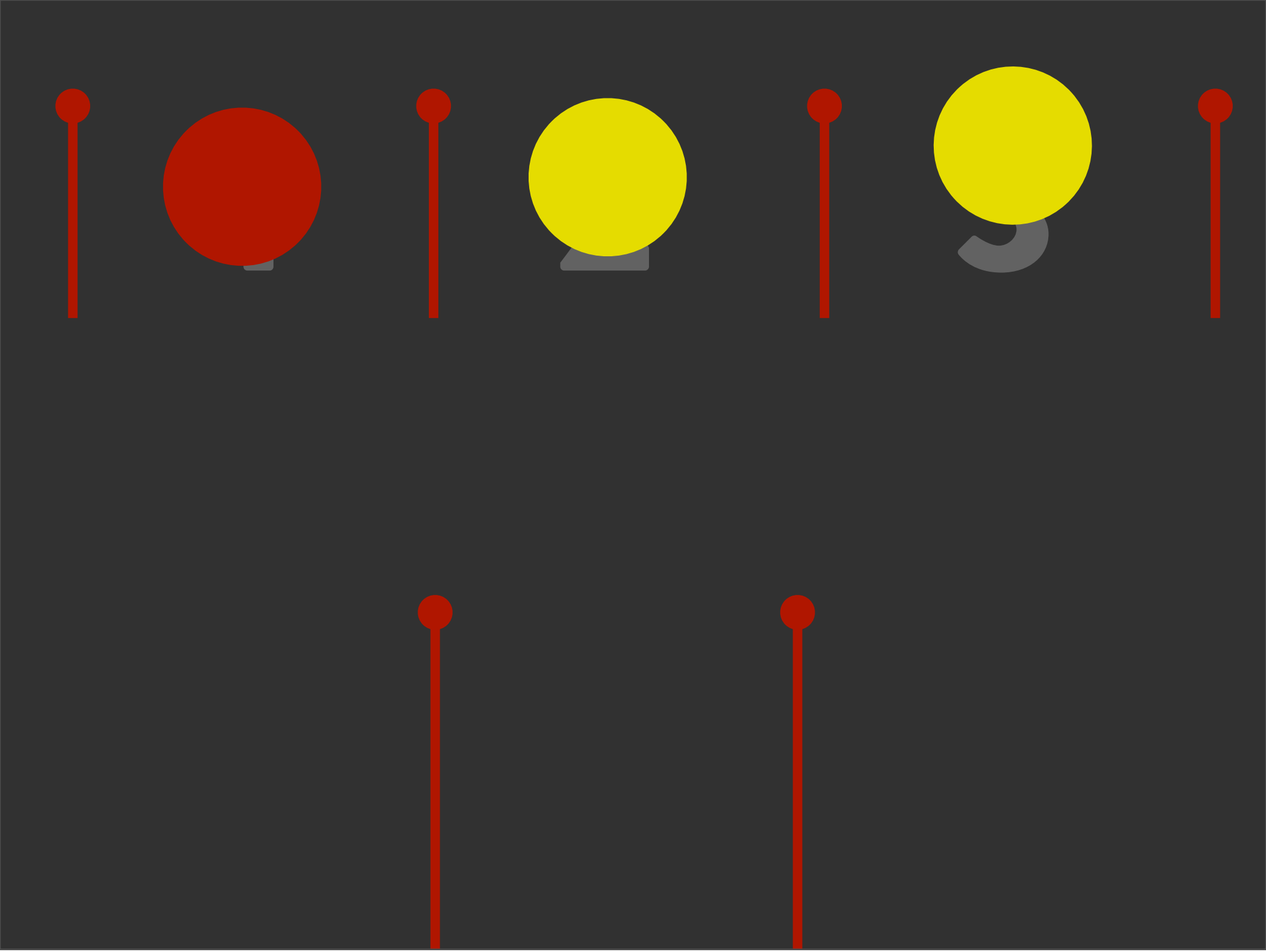

1
2
3

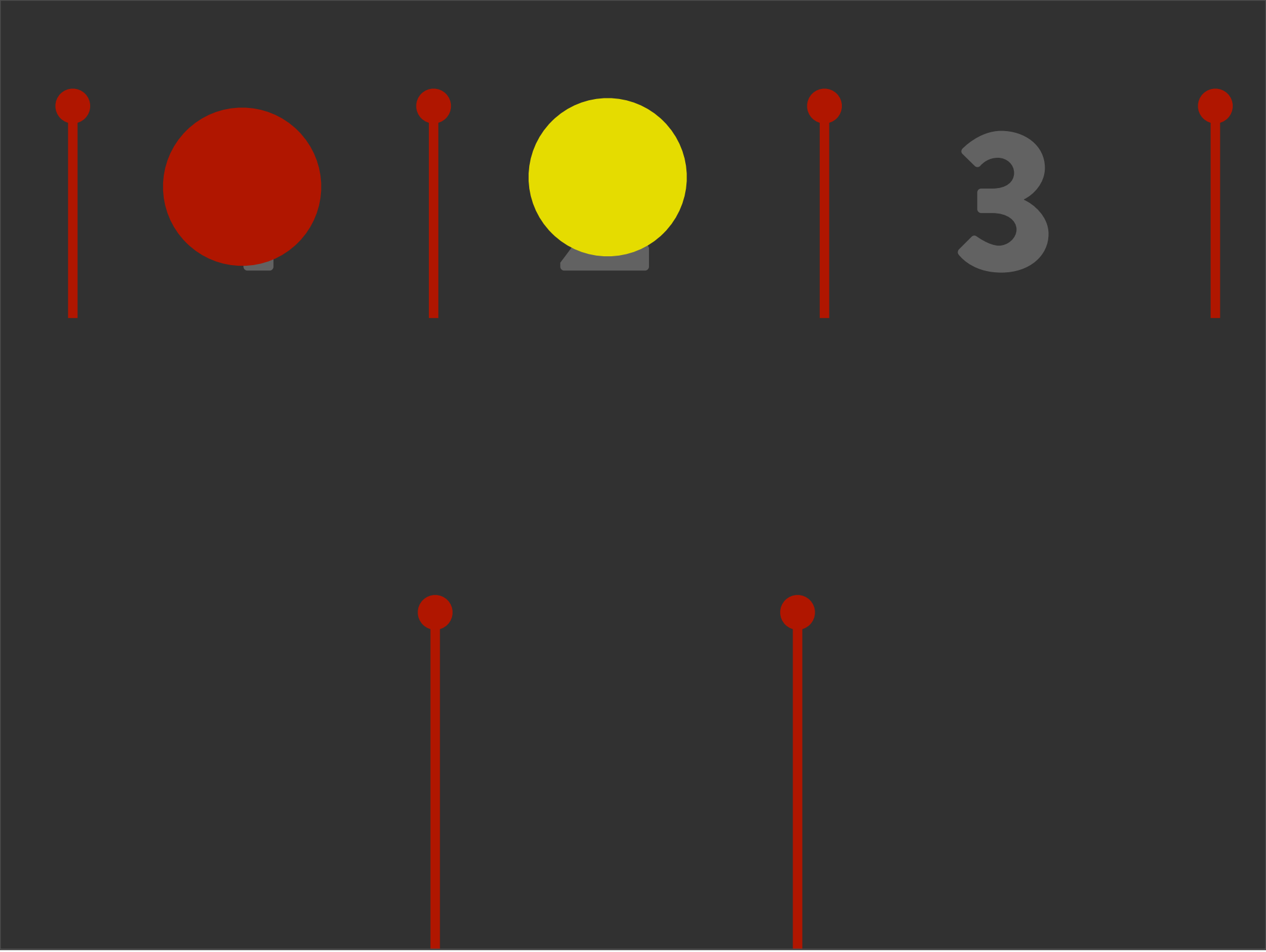

1
2
3


1
2
9


2


2




3


2
3

ERLANG

lookup: erlang B / C / engset

as many threads as possible

don’t sort input URLs

http://abcnews.go.com/
http://abcnews.go.com/2020/ABCNEWSSpecial/
http://abcnews.go.com/2020/story?id=207269&page=1
http://abcnews.go.com/2020/story?id=207269&page=1
http://abcnews.go.com/GMA/JoelSiegel/story?id=1734395
http://abcnews.go.com/International/News/story?id=203089&page=1
http://abcnews.go.com/International/Pope/
http://abcnews.go.com/International/story?id=81417&page=1

```
http://abcnews.go.com/
http://abcnews.go.com/2020/ABCNEWSSpecial/
http://abcnews.go.com/2020/story?id=207269&amp;page=1
http://abcnews.go.com/2020/story?id=207269&amp;page=1
http://abcnews.go.com/GMA/JoelSiegel/story?id=1734395
http://abcnews.go.com/International/News/story?
id=203089&amp;page=1
http://abcnews.go.com/International/Pope/
http://abcnews.go.com/International/story?id=81417&amp;page=1
```

fetch

```
http://abcnews.go.com/
http://abcnews.go.com/2020/ABCNEWSSpecial/
http://abcnews.go.com/2020/story?id=207269&amp;page=1
http://abcnews.go.com/2020/story?id=207269&amp;page=1
http://abcnews.go.com/GMA/JoelSiegel/story?id=1734395
http://abcnews.go.com/International/News/story?
id=203089&amp;page=1
http://abcnews.go.com/International/Pope/
http://abcnews.go.com/International/story?id=81417&amp;page=1
```

wait

```
http://abcnews.go.com/
http://abcnews.go.com/2020/ABCNEWSSpecial/
http://abcnews.go.com/2020/story?id=207269&amp;page=1
http://abcnews.go.com/2020/story?id=207269&amp;page=1
http://abcnews.go.com/GMA/JoelSiegel/story?id=1734395
http://abcnews.go.com/International/News/story?
id=203089&amp;page=1
http://abcnews.go.com/International/Pope/
http://abcnews.go.com/International/story?id=81417&amp;page=1
```

fetch

```
http://abcnews.go.com/
http://abcnews.go.com/2020/ABCNEWSSpecial/
http://abcnews.go.com/2020/story?id=207269&amp;page=1
http://abcnews.go.com/2020/story?id=207269&amp;page=1
http://abcnews.go.com/GMA/JoelSiegel/story?id=1734395
http://abcnews.go.com/International/News/story?
id=203089&amp;page=1
http://abcnews.go.com/International/Pope/
http://abcnews.go.com/International/story?id=81417&amp;page=1
```

wait

```
http://abcnews.go.com/
http://abcnews.go.com/2020/ABCNEWSSpecial/
http://abcnews.go.com/2020/story?id=207269&amp;page=1
http://abcnews.go.com/2020/story?id=207269&amp;page=1
http://abcnews.go.com/GMA/JoelSiegel/story?id=1734395
http://abcnews.go.com/International/News/story?
id=203089&amp;page=1
http://abcnews.go.com/International/Pope/
http://abcnews.go.com/International/story?id=81417&amp;page=1
```

fetch

```
http://abcnews.go.com/
http://abcnews.go.com/2020/ABCNEWSSpecial/
http://abcnews.go.com/2020/story?id=207269&amp;page=1
http://abcnews.go.com/2020/story?id=207269&amp;page=1
http://abcnews.go.com/GMA/JoelSiegel/story?id=1734395
http://abcnews.go.com/International/News/story?
id=203089&amp;page=1
http://abcnews.go.com/International/Pope/
http://abcnews.go.com/International/story?id=81417&amp;page=1
```

wait

http://abcnews.go.com/
http://abcnews.go.com/2020/ABCNEWSSpecial/
http://abcnews.go.com/2020/story?id=207269&page=1
http://abcnews.go.com/2020/story?id=207269&page=1
http://abcnews.go.com/GMA/JoelSiegel/story?id=1734395
http://abcnews.go.com/International/News/story?id=203089&page=1
http://abcnews.go.com/International/Pope/
http://abcnews.go.com/International/story?id=81417&page=1

http://yachtmaintenanceco.com/
http://www.amsterdamports.nl/
http://www.4s-dawn.com/
http://www.embassysuiteslittlerock.com/
http://members.tripod.com/airfields_freeman/NM/Airfields_NM_NW.htm
http://mdgroover.iweb.bsu.edu
http://music.imbc.com/
http://www.robertjbradshaw.com
http://www.kerkattenhoven.be
http://www.escolania.org/
http://www.musiciansdfw.org/
http://www.ariana.org/

http://yachtmaintenanceco.com/
http://www.amsterdamports.nl/
http://www.4s-dawn.com/
http://www.embassysuiteslittlerock.com/
http://members.tripod.com/airfields_freeman/NM/Airfields_NM_NW.htm
http://mdgroover.iweb.bsu.edu
http://music.imbc.com/
http://www.robertjbradshaw.com
http://www.kerkattenhoven.be
http://www.escolania.org/
http://www.musiciansdfw.org/
http://www.ariana.org/

no waiting!

http://yachtmaintenanceco.com/
http://www.amsterdamports.nl/
http://www.4s-dawn.com/
http://www.embassysuiteslittlerock.com/
http://members.tripod.com/airfields_freeman/NM/Airfields_NM_NW.htm
http://mdgroover.iweb.bsu.edu
http://music.imbc.com/
http://www.robertjbradshaw.com
http://www.kerkattenhoven.be
http://www.escolania.org/
http://www.musiciansdfw.org/
http://www.ariana.org/

challenges:

EXTRACTING URLS

challenges:

▸ EXTRACTING URLS

the internet is full of garbage

challenges:

▸ EXTRACTING URLS

challenges:

▸ EXTRACTING URLS

enormous pages

challenges:

▸ EXTRACTING URLS

enormous pages

terrible markup

challenges:

▸ EXTRACTING URLS

enormous pages

terrible markup

ridiculous urls

challenges:

▸ EXTRACTING URLS

enormous pages

terrible markup

ridiculous urls

☃.net/

challenges:

▸ EXTRACTING URLS

enormous pages

terrible markup

ridiculous urls

☃.net/

“unicode snowman dot net”

challenges:

▸ EXTRACTING URLS

be prepared:

challenges:

▸ EXTRACTING URLS

be prepared:

use a streaming XML parser

challenges:

▸ EXTRACTING URLS

be prepared:

use a streaming XML parser

use a library that handle's bad markup

challenges:

▸ EXTRACTING URLS

be prepared:

use a streaming XML parser

use a library that handle's bad markup

be aware that URLs aren't ASCII

challenges:

▸ EXTRACTING URLS

be prepared:

use a streaming XML parser

use a library that handle's bad markup

be aware that URLs aren't ASCII

use a URL normalizer

SOFTWARE

software advice:

software advice:

- goals determine scale

software advice:

- goals determine scale
- someone else has already done it

2 second crawler:

```
function wgetspider() {
  wget --html-extension --convert-links --mirror \
    --page-requisites --progress=bar --level=5   \
    --no-parent --no-verbose \
    --no-check-certificate "$@";
}

$ wgetspider http://www.ischool.berkeley.edu/
```

java crawlers:

java crawlers:

- Heritrix (Internet Archive)

java crawlers:

- Heritrix (Internet Archive)
- Nutch (Lucene)

java crawlers:

- Heritrix (Internet Archive)
- Nutch (Lucene)
- Bixo (Hadoop / Cascading)

java crawlers:

- Heritrix (Internet Archive)
- Nutch (Lucene)
- Bixo (Hadoop / Cascading)

http://crawler.archive.org/
http://nutch.apache.org/
http://bixo.101tec.com/

extraction packages:

extraction packages:

- mechanize

extraction packages:

- mechanize
- BeautifulSoup & urllib2

extraction packages:

- mechanize
- BeautifulSoup & urllib2
- Scrapy

extraction packages:

- mechanize
- BeautifulSoup & urllib2
- Scrapy

http://wwwsearch.sourceforge.net/mechanize/
http://www.crummy.com/software/BeautifulSoup/
http://scrapy.org/

wrapper induction(ish)

wrapper induction(ish)

- Ariel

wrapper induction(ish)

- Ariel
- RoadRunner

wrapper induction(ish)

- Ariel
- RoadRunner
- TemplateMaker

wrapper induction(ish)

- Ariel
- RoadRunner
- TemplateMaker
- scrubyt

wrapper induction(ish)

- Ariel
- RoadRunner
- TemplateMaker
- scrubyt

http://ariel.rubyforge.org/index.html
http://www.dia.uniroma3.it/db/roadRunner/
http://code.google.com/p/templatemaker/
http://scrubyt.rubyforge.org/files/README.html

SCRAPING TUTORIAL (code)

QUESTIONS?

FEEDBACK:

nate@xcombinator.com

www.xcombinator.com

@xcombinator