2 Introduction to Algorithm Design

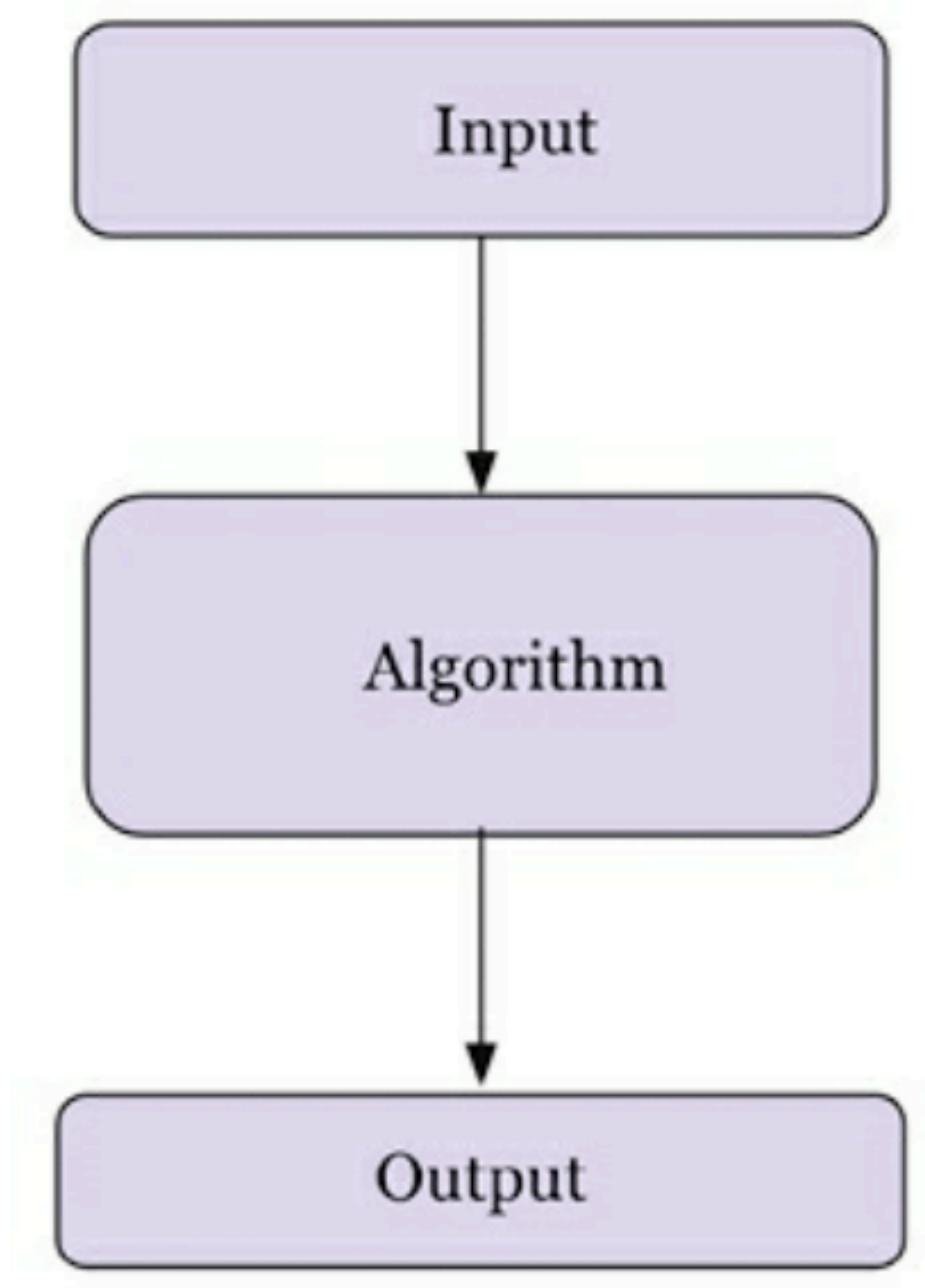
For COMSC 132

Sam Bowne

Aug 20, 2024

Algorithms

- Well-defined procedure
- Takes input data
- Process it
- Produces desired output



Performance Analysis

- Time complexity
  - Time required to perform task
- Space complexity
  - Memory required to perform task

Linear in n

- As size of problem increases
- Time required grows linearly

```
import time

for power in range(3, 9):
  n = 10 ** power
  t0 = time.time()
  total = 0
  for i in range(n):
    total += 1
  elapsed = round(time.time() - t0, 4)
  print("{0:>5}".format(elapsed), n)
```

```
0.0002 1000
0.0018 10000
0.0141 100000
0.1177 1000000
1.0664 10000000
11.5965 100000000
```


```
import time

for power in range(3, 9):
  n = 10 ** power
  t0 = time.time()
  total = 0
  for i in range(n):
    total += 1
  elapsed = round(time.time() - t0, 4)
  print("{0:>5}".format(elapsed), n)
```

Asymptotic notation

- ϴ (Theta)
  - Worst-case running time with a tight bound
- O (Big Oh)
  - Worst-case with an upper bound
- Ω (Omega)
  - Lower bound of running time

- Quadratic $O(n^2)$
- Cubic $O(n^3)$
- Exponential $O(2^n)$

Time Complexity	Name
`O(1)`	Constant
`O(logn)`	Logarithmic
`O(n)`	Linear
`O(nlogn)`	Linear-logarithmic
`O(n2)`	Quadratic
`O(n3)`	Cubic
`O(2n)`	Exponential

Table 2.1: Runtime complexity of different functions

Take Largest Term

- We only care about case for large ***n***
- So if
  - Time = 4***n*****3 + 100***n*** + 1000
  - Complexity is O(***n*****3)

O(n**2) Complexity

[11]

```python
import time

for power in range(2, 5):
  n = 10 ** power
  t0 = time.time()
  total = 0
  for i in range(n):
    for j in range(n):
      total += 1
  elapsed = round(time.time() - t0, 4)
  print(f'{elapsed:9.4f}', f'{n:>7,}')
```

12s

```
   0.0015     100
   0.1825   1,000
  12.2375  10,000
```


```python
import time

for power in range(2, 5):
  n = 10 ** power
  t0 = time.time()
  total = 0
  for i in range(n):
    for j in range(n):
      total += 1
  elapsed = round(time.time() - t0, 4)
  print(f'{elapsed:9.4f}', f'{n:>7,}')
```

O(2**n) Complexity

```python
import time

for n in range(15, 25):
  t0 = time.time()
  total = 0
  for i in range(2**n):
      total += 1
  elapsed = round(time.time() - t0, 4)
  print(f'{elapsed:9.4f}', f'{n:>7,}')
```

```
   0.0048      15
   0.0067      16
   0.0187      17
   0.0277      18
   0.0557      19
   0.1086      20
   0.2344      21
   0.4506      22
   0.8724      23
   1.7665      24
```


```python
import time

for n in range(15, 25):
  t0 = time.time()
  total = 0
  for i in range(2**n):
      total += 1
  elapsed = round(time.time() - t0, 4)
  print(f'{elapsed:9.4f}', f'{n:>7,}')
```

Bitcoin

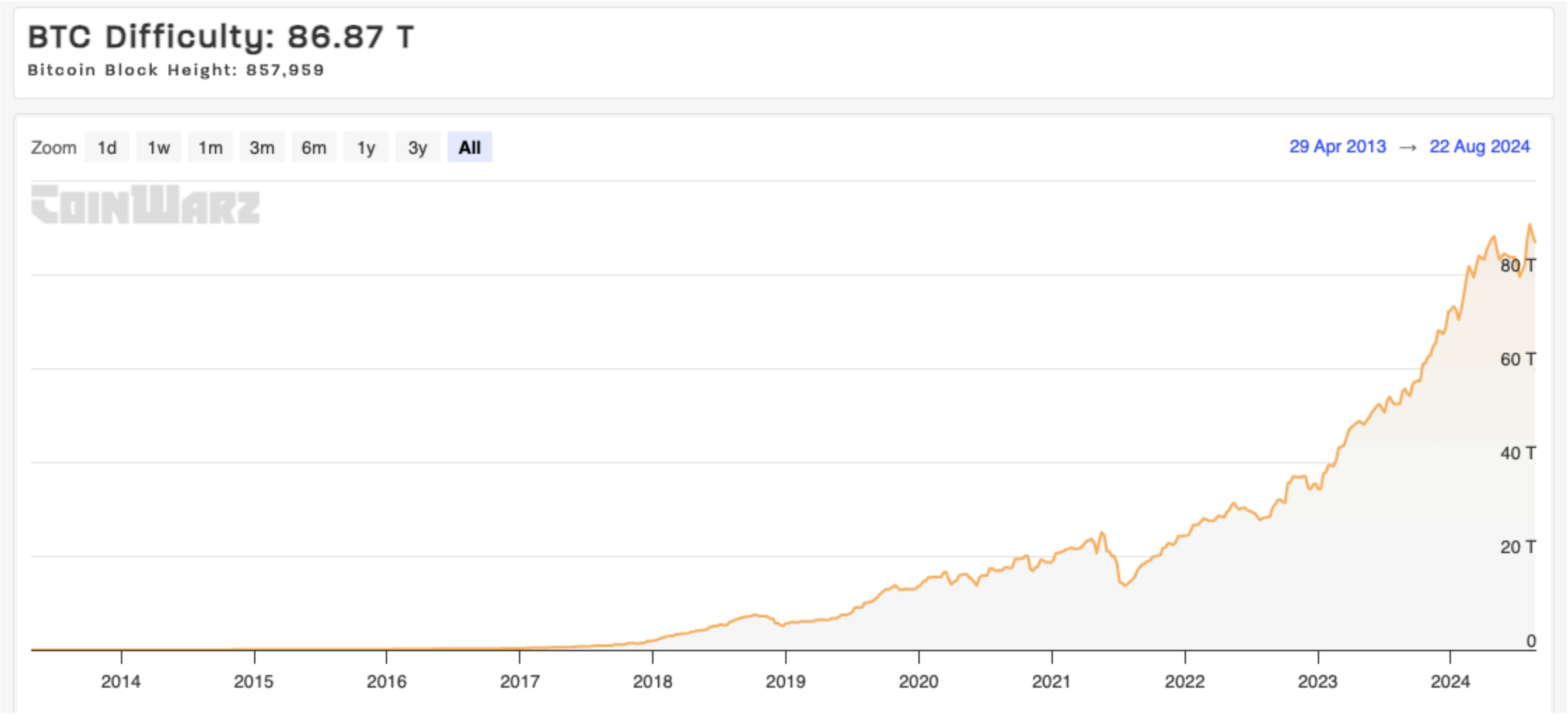
- Miners take a block of transactions
  - Add a random "nonce"
  - Calculate SHA256(SHA256(block))
  - If (hash < **target**):
    - Win! You get a reward.
  - else:
    - Go pick a new nonce

Bitcoin Target Values

- Adjusted to keep average block mining time near 10 minutes
- From https://learnmeabitcoin.com/technical/mining/target/

0	00000000ffff0000000000000000000000000000000000000000000000000000		03 Jan 2009, 18:15:05
185472	00000000000009b78a0000000000000000000000000000000000000000000000	x 0.91695152962047	20 Jun 2012, 16:16:04
856800	000000000000000000033d760000000000000000000000000000000000000000	x 1.0436855505421	14 Aug 2024, 23:59:21

Bitcoin Difficulty

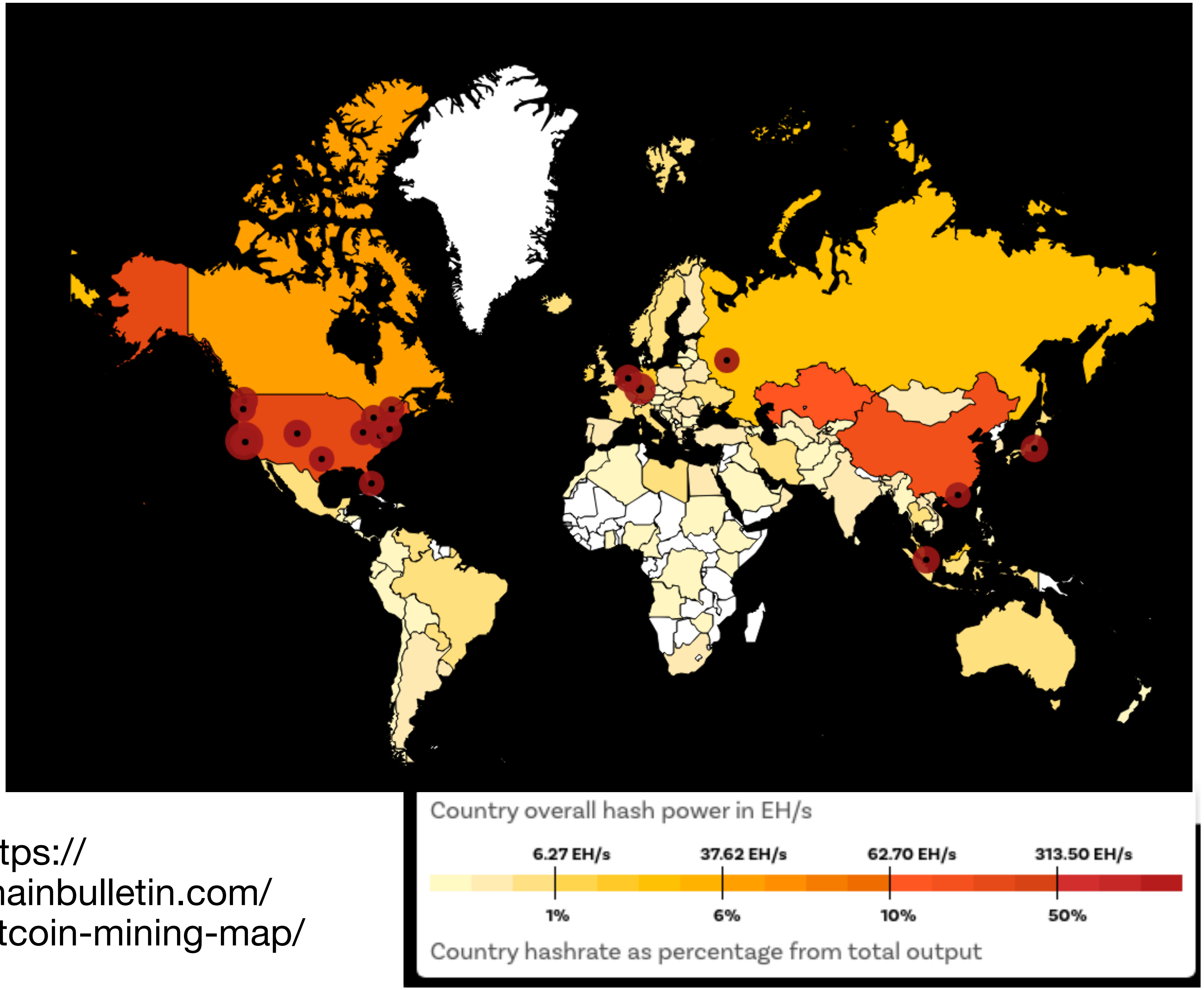


- From https://www.coinwarz.com/mining/bitcoin/difficulty-chart

Bitcoin is CPU-Hard

- Requires lots of CPU cycles to calculate SHA256
- Ultimate cost is power
- Miners move to locations with cheap electricity

What Countries Mine Bitcoin?



- https://chainbulletin.com/bitcoin-mining-map/

Ethereum is Memory-Hard

- Mining requires a data structure called a **Directed Acyclic Graph (DAG)**
- DAG must be generated in each epoch
- The DAG is currently 5 GB
- It grows with each epoch
    - https://deploi.ca/ethereum-algorithm

Ethereum Mainnet Statistics

Clients | **Countries** | Sync Status | OS | Network Types | History

COUNTRIES

Total	6387 (100%)
United States	2199 (34.43%)
Germany	881 (13.79%)
Finland	272 (4.26%)
France	263 (4.12%)
United Kingdom	261 (4.09%)
Canada	217 (3.40%)
Singapore	209 (3.27%)
Australia	165 (2.58%)
Japan	146 (2.29%)
Switzerland	127 (1.99%)

- https://www.ethernodes.org/countries

Kahoot!

Ch 2