Blockchain and Mining

1. News
    1. Hack-a-Thon has picked up an additional $20,000 challenge. (You can still sign up at the door today)
    2. Goldman Sacks - is not doing bitcoin trading, Bitcoin down 10 %, Eth under $200 a token.
    3. SEC's internal review of ICOs suggests that about $^1/_2$ of them are fraudulent.
    4. Next Week - assignment 1 is due.
    5. UW has finally gotten my account set up to use the "wyocourses" stuff - so PDF's and other things will be on the Learning Management System.
    6. 10th (Monday) we will be sending out links for the "late coupons".
2. Definitions / Economics

"arbitrage" ::= the simultaneous buying and selling of securities, currency, or commodities in different markets or in derivative forms in order to take advantage of differing prices for the same asset.

1. How Soros broke the Bank of England.

Price Stable Cryptocurrences (or pegged cryptocurrencies)

Reference

Soros Broke the Bank of England

1. What is Mining and How is it implemented.

What is proof-of-work? What is proof-of-stake? What is a public blockchian / private blockchain?

Diagram of blocks

Diagram of mining

What is the mining process

1. More on Go

Maps do not synchronize automatically. So... Synchronization Primitives:

```
package main
```

```
package main

import (
        "fmt"
        "sync"
        "time"
)

// SafeCounter is safe to use concurrently.
type SafeCounter struct {
        v   map[string]int
        mux sync.Mutex
}

// Inc increments the counter for the given key.
func (c *SafeCounter) Inc(key string) {
        c.mux.Lock()
        // Lock so only one goroutine at a time can access the map c.v.
        c.v[key]++
        c.mux.Unlock()
}

// Value returns the current value of the counter for the given key.
func (c *SafeCounter) Value(key string) int {
        c.mux.Lock()
        // Lock so only one goroutine at a time can access the map c.v.
        defer c.mux.Unlock()
        return c.v[key]
}

func main() {
        c := SafeCounter{v: make(map[string]int)}
        for i := 0; i < 1000; i++ {
                go c.Inc("somekey")
        }

        time.Sleep(time.Second)
        fmt.Println(c.Value("somekey"))
}
```

A Go Core/Panic

First the Code

```
        package main

        import "fmt"
```

```
var mm map[string]int

func main() {
        fmt.Println("vim-go")
        mm["bob"] = 3
}
```

Then the bad output.

```
panic: assignment to entry in nil map

goroutine 1 [running]:
panic(0x10a5540, 0x10d03a0)
        /usr/local/Cellar/go/1.10.3/libexec/src/runtime/panic.go:551 +0x3c1 fp=0xc42005
runtime.mapassign_faststr(0x10a4e80, 0x0, 0x10be68a, 0x3, 0x0)
        /usr/local/Cellar/go/1.10.3/libexec/src/runtime/hashmap_fast.go:696 +0x407 fp=0
main.main()
        /Users/corwin/go/src/github.com/Univ-Wyo-Education/Blockchain-4010-Fall-2018/Le
runtime.main()
        /usr/local/Cellar/go/1.10.3/libexec/src/runtime/proc.go:198 +0x212 fp=0xc42005f
runtime.goexit()
        /usr/local/Cellar/go/1.10.3/libexec/src/runtime/asm_amd64.s:2361 +0x1 fp=0xc420

goroutine 2 [force gc (idle)]:
runtime.gopark(0x10c5580, 0x11397a0, 0x10bfafe, 0xf, 0x10c5414, 0x1)
        /usr/local/Cellar/go/1.10.3/libexec/src/runtime/proc.go:291 +0x11a fp=0xc42004a
runtime.goparkunlock(0x11397a0, 0x10bfafe, 0xf, 0x14, 0x1)
        /usr/local/Cellar/go/1.10.3/libexec/src/runtime/proc.go:297 +0x5e fp=0xc42004a7
runtime.forcegchelper()
        /usr/local/Cellar/go/1.10.3/libexec/src/runtime/proc.go:248 +0xcc fp=0xc42004a7
runtime.goexit()
        /usr/local/Cellar/go/1.10.3/libexec/src/runtime/asm_amd64.s:2361 +0x1 fp=0xc420
created by runtime.init.4
        /usr/local/Cellar/go/1.10.3/libexec/src/runtime/proc.go:237 +0x35
```

“assignment to entry in nil map”

...“/Lect-04/samp.go:9”...