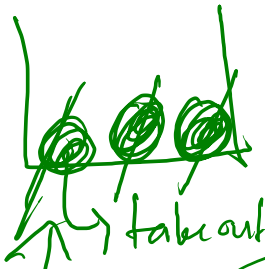
Rock Pile

```
count = 0
for (i = 0; i < s.length; i++)
{ if (s[i] == "-")
  { if (count == 0)
      continue
    else {
      count --
    }
  } else if (s[i] == '+')
  { count ++
  }
}
print count
```

s = "1-1-2-2-3-3-"

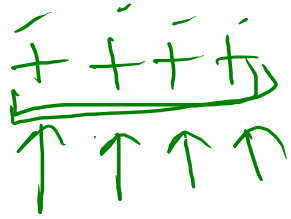
c = 0



# of min. stones to make open possible.

Rock Pile

```
count = 0
for(i=0; i< s.length; i++)
{ if(s[i] == "-")
  { if(count == 0)
      continue
    else
      count --
  }
  else if s[i] == "+")
  { count ++
  }
}
print count
```



c = x+1

c = x+1+1

c = x+1+1+1

c = x+1+1+1+1

c = x+y

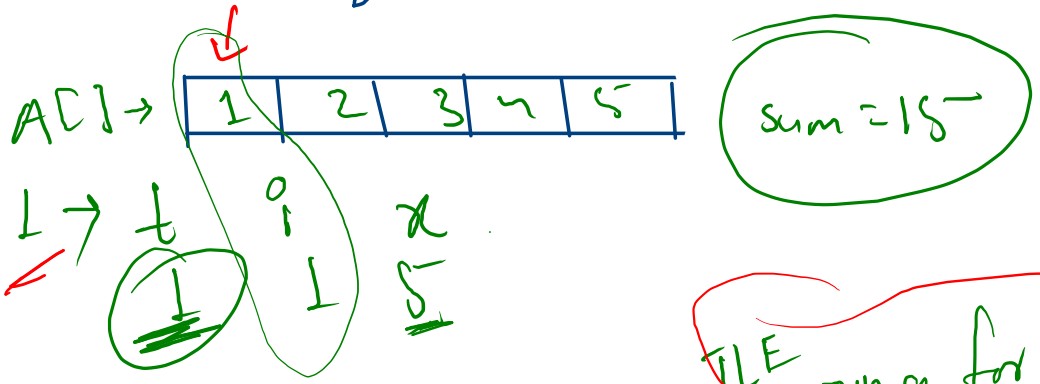
No. of stones in pile at the end of open "-"

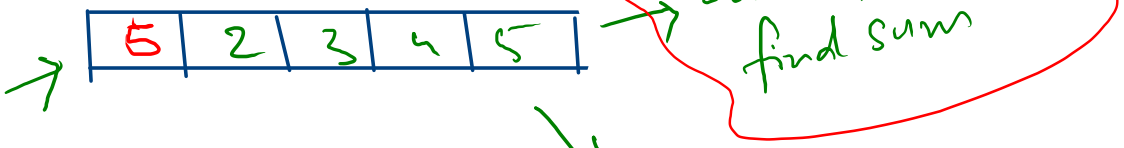
if x=0, c=y

c = x+y

Mike's Revenge :-

$n = 5$, $q = 5$



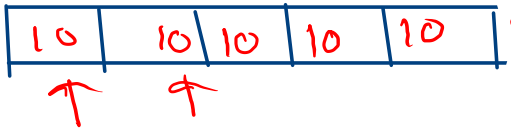


sum = sum − A[i−1] + x = 15 − 1 + 5 = 19

A[i−1] = x

2 → t = 2

x = 10



for loop to fill & get sum simultaneously.

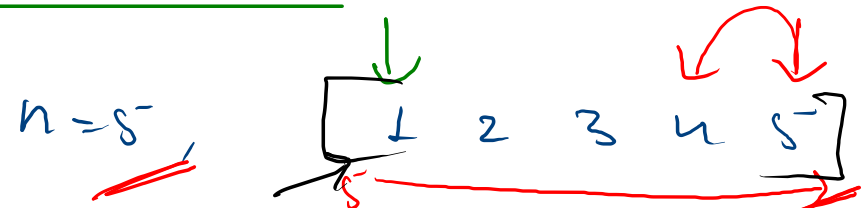
sum = 50

(3) Maximum Money

back at 10:43 am.

5 min break

Recursion :-



use recursion only : this question depends on base case.

what is min valid value of n? (= 1)

```
funct_point(n)
{
    if(n == 1)
    {
        print(n)
        return
    }
    funct_point(n-1)
    print(n)
}
```

base condition

recursive part

OK, I can print 5 or n but before that I need 4 or n-1 to be printed first :

f(5) → I can print 5, but print 4 first, so he called

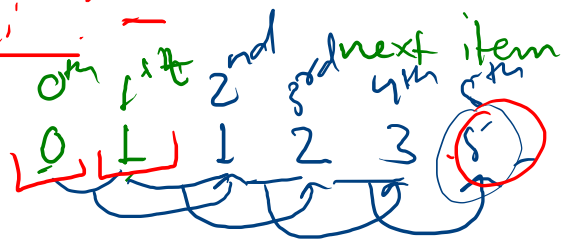
f(4) → I can print 4, but print 3 first, so he called

f(3)

f(2)

f(1) → I can print 1, but print 0 first ✗

Fibonacci :- next item is the sum of previous 2 items.

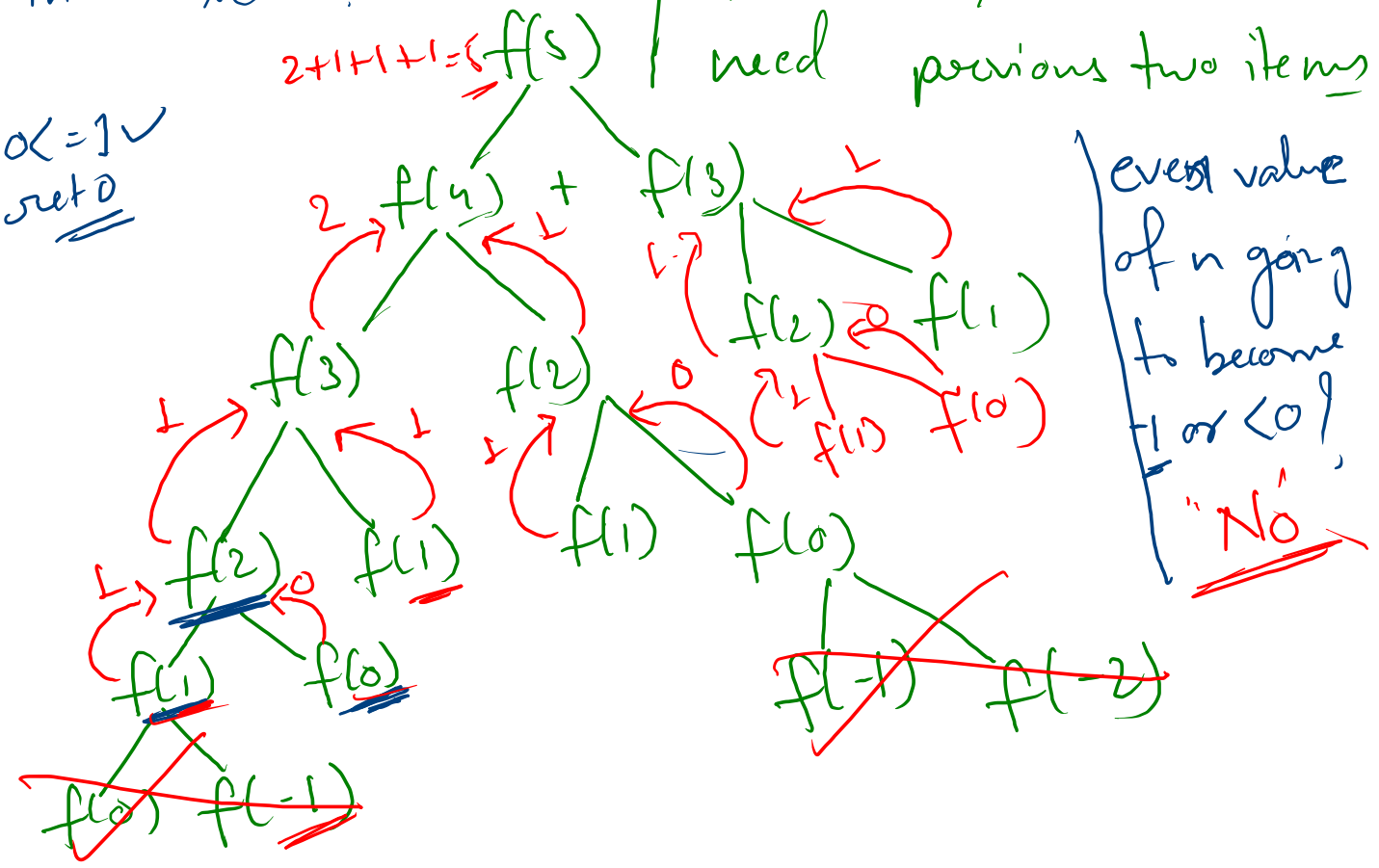


find out 5^{th} item in this series

I can find out 5^{th} item in series but need previous two items.

```
fib-series(n)
{
  if(n<=1)
  {
    return n
  }
  ret (fib(n-1) + fib(n-2))
}
```

n<=1 ret 0



every value of n going to become 1 or <0 ? "No"

```
fib (n)
{  if (n = 1)
       return 1
   if (n = 0)
       return 0

   return (fib(n-1) +
           fib(n-2))
}
```

⇒ Power Function (Recursive)

find a to the power b ?

a = 2 , b = 4

```
power(a, b)
{
  if (b == 0)
    ret 1
  ret a * power(a, b-1)
}
```

$a^n \rightarrow O(n)$

$a^b = 2^4 = 16$

$2^4 = 2^1 \times 2^1 \times 2^1 \times 2^1 \times 2^0$ ($2^0 = 1$)

p(2,4) ↓ (b == 0 ✗)

ret 2 * p(2,3) 2*8 = 16

2 * p(2,2) 2*4 = 8

2 * p(2,1) = 4 ↓↑ 2

T.C ⇒ $2^n \times O(1) \times$ ret

$O(n)$ ← can we do it in less time

$b = 16$, $a = 2$

$2^{16} = 2^8 * 2^8$

$2^8 = 2^4 * 2^4$

$2^4 = 2^2 * 2^2$

$2^2 = 2^1 * 2^1$

$2^{16} = 2^1 * 2^7 * 2^7$

$2^7 = 2^1 * 2^3 * 2^3$

$2^3 = 2^1 * 2^1 * 2^1$

$5 < 2^5$

$5 < 32$

$O(n) < 2^n$

(1) $O(n) <$ exponential

or (2) exponential $< O(n)$ X

b = 16 , a = 2

even

$2^{16} = 2^8 * 2^8$

$2^8 = 2^4 * 2^4$

$2^4 = 2^2 * 2^2$

$2^2 = 2^1 * 2^1$

odd

$15-1 = 14/2 = 7$

$2^{15} = 2^1 * 2^7 * 2^7$

$2^7 = 2^1 * 2^3 * 2^3$

$2^3 = 2^1 * 2^1 * 2^1$

```
power(a, b)
{ if(b==1) return a
  if(b%2==0)
  { res = power(a, b/2)
    return res * res;
  }else
  { res = power(a, (b-1)/2)
    return a * res * res.
  }
}
```

b/2 , n/2 → reducing rate.

T.C → O(log n)

The Power fn. is getting called for log n times only.

new soln

$(\log n)$ 2^{36} $a=2, b=32$.

p(2, 32)

↓

p(2, 16)

↓

p(2, 8)

↓

p(2, 4)

↓

p(2, 2)

↓

p(2, 1)

5+1 = 6 calls.

$O(\log n)$

p(a, b-1)

p(2, 32)

p(2, 31)

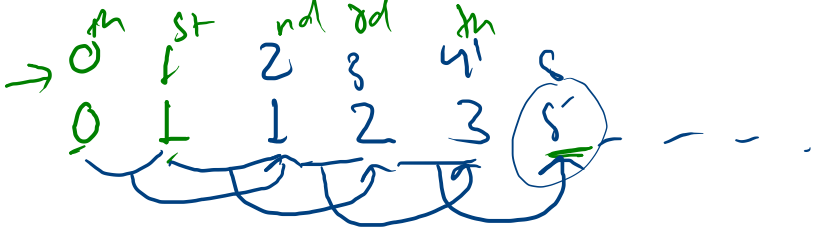
p(2, 30)

p(2, 29)

32 calls

$O(n)$

p(2, 1).



find out s^{th} item in this series

$f(s) = f$