Induction
Recursion
Lecture 15
Nim
Nim

Alice and Bob take turns removing matchsticks from two piles
Nim

- Alice and Bob take turns removing matchsticks from two piles.
- Initially both piles have equal number of matchsticks.
Nim

- Alice and Bob take turns removing matchsticks from two piles.
- Initially both piles have equal number of matchsticks.
- At every turn, a player must choose one pile and remove one or more matchsticks from that pile.
Nim

- Alice and Bob take turns removing matchsticks from two piles.
- Initially both piles have an equal number of matchsticks.
- At every turn, a player must choose one pile and remove one or more matchsticks from that pile.
- Goal: be the person to remove the last matchstick.
Nim

Alice and Bob take turns removing matchsticks from two piles.
Initially both piles have equal number of matchsticks.
At every turn, a player must choose one pile and remove one or more matchsticks from that pile.
Goal: be the person to remove the last matchstick.

Claim: In Nim, the second player has a winning strategy.
Nim

Alice and Bob take turns removing matchsticks from two piles.
Initially both piles have equal number of matchsticks.
At every turn, a player must choose one pile and remove one or more matchsticks from that pile.
Goal: be the person to remove the last matchstick.

Claim: In Nim, the second player has a winning strategy.

(Aside: in every finitely-terminating two player game without draws, one of the players has a winning strategy.)
Alice and Bob take turns removing matchsticks from two piles. Initially both piles have equal number of matchsticks. At every turn, a player must choose one pile and remove one or more matchsticks from that pile. Goal: be the person to remove the last matchstick.

Claim: In Nim, the second player has a winning strategy.

(Aside: in every finitely-terminating two player game without draws, one of the players has a winning strategy)

Claim: The following is a winning strategy for the second player: keep the piles matched at the end of your turn.
Nim

Claim: The following is a winning strategy for the second player: keep the piles matched at the end of your turn.
Nim

Claim: The following is a winning strategy for the second player: keep the piles matched at the end of your turn.

Rephrased: with this strategy for Bob (2nd player), at the end of each turn, either he has already won, or will win from there.
Claim: The following is a winning strategy for the second player: keep the piles matched at the end of your turn.

Rephrased: with this strategy for Bob (2nd player), at the end of each turn, either he has already won, or will win from there.

Induction variable: \( n = \text{number of matchsticks on each pile at the beginning of the turn.} \)
Nim

Claim: The following is a winning strategy for the second player: keep the piles matched at the end of your turn

Rephrased: with this strategy for Bob (2nd player), at the end of each turn, either he has already won, or will win from there

Induction variable: $n = \text{number of matchsticks on each pile at the beginning of the turn.}$

Base case: $n=1$. Alice must remove one. Then Bob wins. ✔
Claim: The following is a winning strategy for the second player: keep the piles matched at the end of your turn.

Rephrased: with this strategy for Bob (2nd player), at the end of each turn, either he has already won, or will win from there.

Induction variable: \( n = \) number of matchsticks on each pile at the beginning of the turn.

Base case: \( n=1 \). Alice must remove one. Then Bob wins.

Induction step: for all integers \( k \geq 2 \)

Induction hypothesis: when starting with \( n \leq k-1 \), Bob always wins.

To prove: when starting with \( n = k \), Bob always wins.
Claim: The following is a winning strategy for the second player: keep the piles matched at the end of your turn.

Rephrased: with this strategy for Bob (2nd player), at the end of each turn, either he has already won, or will win from there.

Induction variable: \( n = \) number of matchsticks on each pile at the beginning of the turn.

Base case: \( n=1 \). Alice must remove one. Then Bob wins.

Induction step: for all integers \( k \geq 2 \)

Induction hypothesis: when starting with \( n \leq k-1 \), Bob always wins.

To prove: when starting with \( n=k \), Bob always wins.
Nim

Claim: The following is a winning strategy for the second player: keep the piles matched at the end of your turn

Rephrased: with this strategy for Bob (2nd player), at the end of each turn, either he has already won, or will win from there

Induction variable: $n = \text{number of matchsticks on each pile at the beginning of the turn.}$

Base case: $n=1$. Alice must remove one. Then Bob wins.

Induction step: for all integers $k \geq 2$

Induction hypothesis: when starting with $n \leq k-1$, Bob always wins

To prove: when starting with $n=k$, Bob always wins

Case 1: Alice removes all $k$ from one pile. Then Bob wins.
Nim

Claim: The following is a winning strategy for the second player: keep the piles matched at the end of your turn

Rephrased: with this strategy for Bob (2nd player), at the end of each turn, either he has already won, or will win from there

Induction variable: $n =$ number of matchsticks on each pile at the beginning of the turn.

Base case: $n=1$. Alice must remove one. Then Bob wins.

Induction step: for all integers $k \geq 2$

Induction hypothesis: when starting with $n \leq k-1$, Bob always wins

To prove: when starting with $n=k$, Bob always wins

Case 1: Alice removes all $k$ from one pile. Then Bob wins.

Case 2: Alice removes $j$, $1 \leq j \leq k-1$ from one pile. After Bob’s move $k-j$ left in each pile. By induction hypothesis, Bob will always win from here.
Recursive Definitions
Programming the Definitions
Lecture 15
Recursive Definitions
Recursive Definitions

E.g., \( f(0) = 1 \)
\[ f(n) = n \cdot f(n-1) \quad \forall n \in \mathbb{Z} \text{ s.t. } n > 0 \]
Recursive Definitions

- **E.g.**, \( f(0) = 1 \)
  
  \[ f(n) = n \cdot f(n-1) \quad \forall n \in \mathbb{Z} \text{ s.t. } n > 0 \]

- \( f(n) = n \cdot (n-1) \cdot \ldots \cdot 1 \cdot 1 = n! \)
Recursive Definitions

- E.g., \( f(0) = 1 \)
  \[ f(n) = n \cdot f(n-1) \quad \forall n \in \mathbb{Z} \text{ s.t. } n > 0 \]

- \( f(n) = n \cdot (n-1) \cdot \ldots \cdot 1 \cdot 1 = n! \)

- This is the formal definition of \( n! \) (without using “...”)
Recursive Definitions

- E.g., \( f(0) = 1 \)
  \( f(n) = n \cdot f(n-1) \quad \forall n \in \mathbb{Z} \) s.t. \( n > 0 \)

- \( f(n) = n \cdot (n-1) \cdot \ldots \cdot 1 \cdot 1 = n! \)

- This is the formal definition of \( n! \) (without using "\( \ldots \)"

- A recursive program to compute factorial:

```c
factorial (n∈\mathbb{N}) { 
    if (n==0) return 1;
    else return n*factorial(n-1);
}
```
Question
Question

\[ f(0) = 3; \quad f(n) = 2 \cdot f(n-1) \text{ for } n \in \mathbb{Z}^+. \text{ Then for } n \in \mathbb{N} \]

A. \( f(n) = 3^{n+1} \)
B. \( f(n) = (3!)^n \)
C. \( f(n) = 6 \cdot 2^n \)
D. \( f(n) = 3 \cdot 2^n \)
E. None of the above
Fibonacci Sequence
Fibonacci Sequence

\[ F(0) = 0 \]
\[ F(1) = 1 \]
\[ F(n) = F(n-1) + F(n-2) \quad \forall n \geq 2 \]
Fibonacci Sequence

- $F(0) = 0$
- $F(1) = 1$
- $F(n) = F(n-1) + F(n-2)$ $\forall n \geq 2$

$F(n)$ called the $n^{th}$ Fibonacci number (starting with $0^{th}$)
Fibonacci Sequence

\[ F(0) = 0 \]
\[ F(1) = 1 \]
\[ F(n) = F(n-1) + F(n-2) \quad \forall n \geq 2 \]

\[ F(n) \] called the \( n^{th} \) Fibonacci number (starting with \( 0^{th} \))
Question
Question

\[
\text{fib} \ (n \in \mathbb{Z}) \ \{ \\
\text{if} \ (n == 0) \ \text{return} \ 0; \\
\text{else return fib}(n-1) + \text{fib}(n-2); \\
\text{print} \ "F(n) \ not \ defined \ for \ input"; \\
\}
\]

F(n) be the n\textsuperscript{th} Fibonacci number as just defined.

On input \( n \in \mathbb{Z} \), this program will:

A. return F(n)
B. return F(n) if it is defined, else print error
C. return F(n) if it is defined, else go on forever
D. go on forever, unless n==0
E. None of the above
Closed Form
Closed Form

Sometimes possible to get a “closed form” expression for a quantity defined recursively
Sometimes possible to get a “closed form” expression for a quantity defined recursively

e.g., $f(0) = 0$ & $f(n) = f(n-1) + n$, $\forall n > 0$
Closed Form

Sometimes possible to get a “closed form” expression for a quantity defined recursively

- e.g., \( f(0)=0 \) & \( f(n) = f(n-1) + n, \forall n>0 \)
- \( f(n) = \frac{n(n+1)}{2} \)
Closed Form

Sometimes possible to get a “closed form” expression for a quantity defined recursively

\[ f(0) = 0 \quad \& \quad f(n) = f(n-1) + n, \ \forall n > 0 \]

\[ f(n) = \frac{n(n+1)}{2} \]

Sometimes, just give it a name
Closed Form

Sometimes possible to get a “closed form” expression for a quantity defined recursively

- e.g., $f(0)=0$ & $f(n) = f(n-1) + n$, $\forall n>0$

- $f(n) = n(n+1)/2$

Sometimes, just give it a name

- e.g., $n!$, Fibonacci(n)
Closed Form

Sometimes possible to get a “closed form” expression for a quantity defined recursively

- e.g., \( f(0) = 0 \) & \( f(n) = f(n-1) + n, \forall n > 0 \)

- \( f(n) = n(n+1)/2 \)

Sometimes, just give it a name

- e.g., \( n! \), Fibonacci\( (n) \)

In fact, formal definitions of integers, addition, multiplication etc. are recursive
Closed Form

Sometimes possible to get a “closed form” expression for a quantity defined recursively

- e.g., $f(0)=0$ & $f(n) = f(n-1) + n$, $\forall n>0$
- $f(n) = n(n+1)/2$

Sometimes, just give it a name

- e.g., $n!$, Fibonacci($n$)

In fact, formal definitions of integers, addition, multiplication etc. are recursive

- e.g., $0 \cdot a = 0$ & $n \cdot a = (n-1) \cdot a + a$, $\forall n>0$
Closed Form

Sometimes possible to get a “closed form” expression for a quantity defined recursively

- e.g., \( f(0) = 0 \) & \( f(n) = f(n-1) + n, \forall n>0 \)
- \( f(n) = n(n+1)/2 \)

Sometimes, just give it a name

- e.g., \( n! \), Fibonacci\( (n) \)

In fact, formal definitions of integers, addition, multiplication etc. are recursive

- e.g., \( 0 \cdot a = 0 \) & \( n \cdot a = (n-1) \cdot a + a, \forall n>0 \)
- e.g., \( 2^0 = 1 \) & \( 2^n = 2 \cdot 2^{n-1} \)
“Closed” form
“Closed” form

Suppose $g(1) = 1$ & $g(n) = 2 \cdot g(n-1) + n \quad \forall n > 1$. 
“Closed” form

Suppose \( g(1) = 1 \) & \( g(n) = 2 \ g(n-1) + n \ \forall n \geq 1. \)

\( g(n) \) is growing “exponentially” by (more than) doubling for each increment in \( n \)
"Closed" form

Suppose \( g(1) = 1 \) & \( g(n) = 2 \cdot g(n-1) + n \) \( \forall n>1 \).

\( g(n) \) is growing “exponentially” by (more than) doubling for each increment in \( n \)

\( g(n) = \sum_{i=1}^{n} i \cdot 2^{n-i} \)
“Closed” form

Suppose $g(1) = 1$ & $g(n) = 2 \cdot g(n-1) + n$ $\forall n>1$.

g(n) is growing “exponentially” by (more than) doubling for each increment in n

g(n) = \sum_{i=1}^{n} i \cdot 2^{n-i}

A “guess” (make sure the base case matches)
“Closed” form

Suppose \( g(1) = 1 \) & \( g(n) = 2 \ g(n-1) + n \ \forall n>1. \)

- \( g(n) \) is growing “exponentially” by (more than) doubling for each increment in \( n \)
- \( g(n) = \sum_{i=1}^{n} i \cdot 2^{n-i} \)
- A “guess” (make sure the base case matches)
- Then prove by induction
"Closed" form

Suppose \( g(1) = 1 \) & \( g(n) = 2 \cdot g(n-1) + n \) \( \forall n \geq 1 \).

\( g(n) \) is growing "exponentially" by (more than) doubling for each increment in \( n \).

\( g(n) = \sum_{i=1}^{n} i \cdot 2^{n-i} \)

A "guess" (make sure the base case matches)

Then prove by induction

How do we guess? (Not always easy/possible.)
"Closed" form

Suppose \( g(1) = 1 \) & \( g(n) = 2 \ g(n-1) + n \quad \forall n \geq 1 \).

- \( g(n) \) is growing "exponentially" by (more than) doubling for each increment in \( n \)

\[ g(n) = \sum_{i=1}^{n} i \cdot 2^{n-i} \]

- A "guess" (make sure the base case matches)

- Then prove by induction

How do we guess? (Not always easy/possible.)

\[ g(n) = n + 2 \cdot g(n-1) \]

\[ = n + 2 \cdot ( (n-1) + 2 \cdot g(n-2) ) \]

\[ = n + 2 \cdot ( (n-1) + 2 \cdot ( (n-2) + 2 \cdot g(n-3) ) ) \]

\[ = n + 2 \cdot (n-1) + 2^2 \cdot (n-2) + 2^3 \cdot g(n-3) \]
Unrolling a recursion
Unrolling a recursion

Often helpful to try “unrolling” the recursion to see what is happening
Unrolling a recursion

Often helpful to try “unrolling” the recursion to see what is happening

e.g., expand into a chain:
Unrolling a recursion

Often helpful to try “unrolling” the recursion to see what is happening

e.g., expand into a chain:

\[ T(0) = 0 \quad \& \quad T(n) = T(n-1) + n^2 \quad \forall n \geq 1 \]
Unrolling a recursion

Often helpful to try “unrolling” the recursion to see what is happening

e.g., expand into a chain:

- $T(0) = 0 \land T(n) = T(n-1) + n^2 \quad \forall n \geq 1$
- $T(n-1) = T(n-2) + (n-1)^2, \; T(n-2) = T(n-3) + (n-2)^2, \; ...$
Unrolling a recursion

Often helpful to try “unrolling” the recursion to see what is happening

e.g., expand into a chain:

\[ T(0) = 0 \quad \& \quad T(n) = T(n-1) + n^2 \quad \forall n \geq 1 \]

\[ T(n-1) = T(n-2) + (n-1)^2, \quad T(n-2) = T(n-3) + (n-2)^2, \quad \ldots \]

\[ T(n) = n^2 + (n-1)^2 + (n-2)^2 + T(n-3) \quad \forall n \geq 3 \]
Unrolling a recursion

Often helpful to try "unrolling" the recursion to see what is happening

e.g., expand into a chain:

\[ T(0) = 0 \quad & \quad T(n) = T(n-1) + n^2 \quad \forall n \geq 1 \]
\[ T(n-1) = T(n-2) + (n-1)^2, \quad T(n-2) = T(n-3) + (n-2)^2, \ldots \]
\[ T(n) = n^2 + (n-1)^2 + (n-2)^2 + T(n-3) \quad \forall n \geq 3 \]
\[ T(n) = \sum_{i=0}^{n} i^2 \quad \forall n \geq 0 \]
Unrolling a recursion

Often helpful to try “unrolling” the recursion to see what is happening

e.g., expand into a chain:

- $T(0) = 0$ & $T(n) = T(n-1) + n^2 \quad \forall n \geq 1$
- $T(n-1) = T(n-2) + (n-1)^2$, $T(n-2) = T(n-3) + (n-2)^2$, ...
- $T(n) = n^2 + (n-1)^2 + (n-2)^2 + T(n-3) \quad \forall n \geq 3$
- $T(n) = \sum_{i=0}^{n} i^2 \quad \forall n \geq 0$

e.g., expand a recurrence relation into a “tree”
Unrolling a recursion

Often helpful to try “unrolling” the recursion to see what is happening

e.g., expand into a chain:

\[ T(0) = 0 \quad \& \quad T(n) = T(n-1) + n^2 \quad \forall n \geq 1 \]

\[ T(n-1) = T(n-2) + (n-1)^2, \quad T(n-2) = T(n-3) + (n-2)^2, \ldots \]

\[ T(n) = n^2 + (n-1)^2 + (n-2)^2 + T(n-3) \quad \forall n \geq 3 \]

\[ T(n) = \sum_{i=0}^{n} i^2 \quad \forall n \geq 0 \]

e.g., expand a recurrence relation into a “tree”

\[ T(0) = 1 \quad \& \quad T(n) = 2T(n-1) + 1 \quad \forall n \geq 1 \]
Recursion & “Trees”
Recursion & “Trees”

- $T(0) = 1$
- $T(n) = 2T(n-1) + 1$
Recursion & “Trees”

- $T(0) = 1$
- $T(n) = 2T(n-1) + 1$
Recursion & “Trees”

\[ T(0) = 1 \]
\[ T(n) = 2T(n-1) + 1 \]
Recursion & “Trees”

$T(0) = 1$
$T(n) = 2T(n-1) + 1$
Recursion & “Trees”

- $T(0) = 1$
- $T(n) = 2T(n-1) + 1$
Recursion & “Trees”

- \( T(0) = 1 \)
- \( T(n) = 2T(n-1) + 1 \)
Recursion & “Trees”

- $T(0) = 1$
- $T(n) = 2T(n-1) + 1$
Recursion & “Trees”

- $T(0) = 1$
- $T(n) = 2T(n-1) + 1$
Recursion & “Trees”

- $T(0) = 1$
- $T(n) = 2T(n-1) + 1$
Recursion & “Trees”

- $T(0) = 1$
- $T(n) = 2T(n-1) + 1$

Doing it bottom-up. Could also think top-down.
Recursion & “Trees”

- $T(0) = 1$
- $T(n) = 2T(n-1) + 1$
Recursion & “Trees”

- $T(0) = 1$
- $T(n) = 2T(n-1) + 1$
- Exponential growth
Recursion & “Trees”

- $T(0) = 1$
- $T(n) = 2T(n-1) + 1$
- Exponential growth
- $T(1) = 3$, $T(2) = 7$, ...

[Diagram of a tree structure with nodes labeled 1, 1, 1, 1, 1, 1, 1, 1]
Recursion & “Trees”

- $T(0) = 1$
- $T(n) = 2T(n-1) + 1$
- Exponential growth
- $T(1) = 3$, $T(2) = 7$, ...
- $T(n) = 2^{n+1} - 1$ (guess)
Recursion & “Trees”

- $T(0) = 1$
- $T(n) = 2T(n-1) + 1$

- Exponential growth

- $T(1) = 3$, $T(2) = 7$, ...
- $T(n) = 2^{n+1} - 1$ (guess)

- Works for base case: $n=0$. 
Recursion & "Trees"

- $T(0) = 1$
- $T(n) = 2T(n-1) + 1$
- Exponential growth
- $T(1) = 3$, $T(2) = 7$, ...
- $T(n) = 2^{n+1} - 1$ (guess)
- Works for base case: $n=0$.
- Inductive step: $k \geq 1$. 
Recursion & “Trees”

- \( T(0) = 1 \)
- \( T(n) = 2T(n-1) + 1 \)

Exponential growth

- \( T(1) = 3, \ T(2) = 7, \ldots \)
- \( T(n) = 2^{n+1} - 1 \) (guess)

Works for base case: \( n=0 \).

Inductive step: \( k \geq 1 \).

- \( T(k) = 2T(k-1) + 1 = 2(2^k-1) + 1 = 2^{k+1}-1 \) ✔
Another example
Another example

\[ T(1) = 0 \]
\[ T(N) = T\left( \lfloor N/2 \rfloor \right) + 1 \quad \forall N \geq 2 \]
Another example

- $T(1) = 0$
  
- $T(N) = T\left(\left\lfloor N/2 \right\rfloor \right) + 1 \quad \forall N \geq 2$

- Let us consider $N$ of the form $2^n$ (so we can forget the ceiling)
Another example

\[ T(1) = 0 \]
\[ T(N) = T\left( \left\lfloor \frac{N}{2} \right\rfloor \right) + 1 \quad \forall N \geq 2 \]

Let us consider \( N \) of the form \( 2^n \) (so we can forget the ceiling)

\[ T(N) = 1 + T(N/2) \]
\[ = 1 + 1 + T(N/4) \]
\[ = \ldots \]
\[ = 1 + 1 + \ldots + T(1) \]
Another example

- $T(1) = 0$
- $T(N) = T(\lfloor N/2 \rfloor) + 1 \quad \forall N \geq 2$

Let us consider $N$ of the form $2^n$ (so we can forget the ceiling)

- $T(N) = 1 + T(N/2)$
  - $= 1 + 1 + T(N/4)$
  - $= \ldots$
  - $= 1 + 1 + \ldots + T(1)$

How many 1's there?
Another example

- \( T(1) = 0 \)
  \[ T(N) = T( \lfloor N/2 \rfloor ) + 1 \quad \forall N \geq 2 \]

- Let us consider \( N \) of the form \( 2^n \) (so we can forget the ceiling)
  \[ T(N) = 1 + T(N/2) \]
  \[ = 1 + 1 + T(N/4) \]
  \[ = ... \]
  \[ = 1 + 1 + ... + T(1) \]

- \( T(2^n) = n \)

How many 1's there?
Another example

- \( T(1) = 0 \)
- \( T(N) = T(\lfloor N/2 \rfloor) + 1 \quad \forall N \geq 2 \)

Let us consider \( N \) of the form \( 2^n \) (so we can forget the ceiling)

- \( T(N) = 1 + T(N/2) \)
  - \( = 1 + 1 + T(N/4) \)
  - \( = ... \)
  - \( = 1 + 1 + ... + T(1) \)
- \( T(2^n) = n \)
- \( T(N) = \log_2 N \) (or simply \( \log N \)) for \( N \) a power of 2

How many 1's there?
Another example

- $T(1) = 0$
- $T(N) = T\left(\left\lfloor \frac{N}{2} \right\rfloor \right) + 1 \quad \forall N \geq 2$

Let us consider $N$ of the form $2^n$ (so we can forget the ceiling)

- $T(N) = 1 + T(N/2)$
  - $= 1 + 1 + T(N/4)$
  - $= ...$
  - $= 1 + 1 + ... + T(1)$

- $T(2^n) = n$
- $T(N) = \log_2 N$ (or simply $\log N$) for $N$ a power of 2
Another example

- \( T(1) = 0 \)
  \( T(N) = T(\lfloor N/2 \rfloor) + 1 \quad \forall N \geq 2 \)

- Let us consider \( N \) of the form \( 2^n \) (so we can forget the ceiling)
  \( T(N) = 1 + T(N/2) \)
  \( = 1 + 1 + T(N/4) \)
  \( = \ldots \)
  \( = 1 + 1 + \ldots + T(1) \)

- \( T(2^n) = n \)

- \( T(N) = \log_2 N \) (or simply \( \log N \)) for \( N \) a power of 2

- \( T \) monotonically increasing (can prove by strong induction: later)

How many 1's there?

A slowly growing function
Another example

- \( T(1) = 0 \)
  \[ T(N) = T\left( \lfloor N/2 \rfloor \right) + 1 \quad \forall N \geq 2 \]

- Let us consider \( N \) of the form \( 2^n \) (so we can forget the ceiling)
  \[ T(N) = 1 + T(N/2) \]
  \[ = 1 + 1 + T(N/4) \]
  \[ = \ldots \]
  \[ = 1 + 1 + \ldots + T(1) \]
  \[ T(2^n) = n \]

- \( T(N) = \log_2 N \) (or simply \( \log N \)) for \( N \) a power of 2

- \( T \) monotonically increasing (can prove by strong induction: later)
  \[ T\left(2^{\lfloor \log N \rfloor}\right) \leq T(N) \leq T\left(2^{\lceil \log N \rceil}\right) : i.e., \quad \lfloor \log N \rfloor \leq T(N) \leq \lceil \log N \rceil \]
Another example

- $T(1) = 0$
- $T(N) = T(\lfloor N/2 \rfloor) + 1 \quad \forall N \geq 2$

Let us consider $N$ of the form $2^n$ (so we can forget the ceiling)

- $T(N) = 1 + T(N/2)$
- $= 1 + 1 + T(N/4)$
- $= ...$
- $= 1 + 1 + ... + T(1)$

- $T(2^n) = n$
- $T(N) = \log_2 N$ (or simply $\log N$) for $N$ a power of 2

$T$ monotonically increasing (can prove by strong induction: later)

- $T(2^{\lfloor \log N \rfloor}) \leq T(N) \leq T(2^{\lceil \log N \rceil})$ : i.e., $\lfloor \log N \rfloor \leq T(N) \leq \lceil \log N \rceil$
- In fact, $T(N) = T(2^{\lfloor \log N \rfloor}) = \lfloor \log N \rfloor$ (Exercise)

How many 1's there?

A slowly growing function
Recursion & Induction
Recursion & Induction

Claim: $F(3n)$ is even, where $F(n)$ is the $n^{th}$ Fibonacci number, $\forall n \geq 0$
Recursion & Induction

Claim: $F(3n)$ is even, where $F(n)$ is the $n^{th}$ Fibonacci number, $\forall n \geq 0$

Proof by induction:
Recursion & Induction

Claim: $F(3n)$ is even, where $F(n)$ is the $n^{\text{th}}$ Fibonacci number, $\forall n \geq 0$

Proof by induction:

Base case:
$n=0$: $F(3n) = F(0) = 0$ ✔
$n=1$: $F(3n) = F(3) = 2$ ✔
Recursion & Induction

Claim: \( F(3n) \) is even, where \( F(n) \) is the \( n^{th} \) Fibonacci number, \( \forall n \geq 0 \)

Proof by induction:

Base case:
\( n=0: \ F(3) = F(0) = 0 \ \checkmark \quad n=1: \ F(3) = F(3) = 2 \ \checkmark \)

Induction step: for all \( k \geq 2 \)
Induction hypothesis: suppose for \( 0 \leq n \leq k-1 \), \( F(3n) \) is even
To prove: \( F(3k) \) is even
Claim: \( F(3n) \) is even, where \( F(n) \) is the \( n^{th} \) Fibonacci number, \( \forall n \geq 0 \)

Proof by induction:

Base case:
\[ n=0: \quad F(3n) = F(0) = 0 \quad \checkmark \quad n=1: \quad F(3n) = F(3) = 2 \quad \checkmark \]

Induction step: for all \( k \geq 2 \)
Induction hypothesis: suppose for \( 0 \leq n \leq k-1 \), \( F(3n) \) is even
To prove: \( F(3k) \) is even

\[ F(3k) = F(3k-1) + F(3k-2) = ? \]
Recursion & Induction

Claim: $F(3n)$ is even, where $F(n)$ is the $n^{th}$ Fibonacci number, $\forall n \geq 0$

Proof by induction:

Base case:
- $n=0$: $F(3n) = F(0) = 0$ ✔
- $n=1$: $F(3n) = F(3) = 2$ ✔

Induction step: for all $k \geq 2$

Induction hypothesis: suppose for $0 \leq n \leq k-1$, $F(3n)$ is even

To prove: $F(3k)$ is even

$F(3k) = F(3k-1) + F(3k-2) = ?$

Unroll further: $F(3k-1) = F(3k-2) + F(3k-3)$

$F(3k) = 2 \cdot F(3k-2) + F(3(k-1)) = \text{even, by induction hypothesis}$
Recursion & Induction
Recursion & Induction

Let $a, b$ be arbitrary (non-zero) numbers
Recursion & Induction

Let $a, b$ be arbitrary (non-zero) numbers.

$f(0) = 0$.  $f(1) = a - b$.  $f(n) = (a + b) \cdot f(n-1) - a \cdot b \cdot f(n-2)$  $\forall n \geq 2$. 

Recursion & Induction

Let $a, b$ be arbitrary (non-zero) numbers

\[ f(0) = 0. \quad f(1) = a - b. \quad f(n) = (a + b) \cdot f(n-1) - a \cdot b \cdot f(n-2) \quad \forall n \geq 2. \]

Claim: $f(n) = a^n - b^n$
Recursion & Induction

Let $a, b$ be arbitrary (non-zero) numbers

$f(0) = 0$.  $f(1) = a-b$.  $f(n) = (a+b) \cdot f(n-1) - a \cdot b \cdot f(n-2)$  $\forall n \geq 2$.

Claim:  $f(n) = a^n - b^n$

Base cases:  $n=0$  ($f(0) = a^0 - b^0$) and  $n=1$  ($f(1) = a^1 - b^1$)  ✔
Recursion & Induction

Let $a, b$ be arbitrary (non-zero) numbers

$f(0) = 0$.  $f(1) = a - b$.  $f(n) = (a+b) \cdot f(n-1) - a \cdot b \cdot f(n-2) \quad \forall n \geq 2.$

Claim:  $f(n) = a^n - b^n$

Base cases:  $n=0$  ($f(0) = a^0 - b^0$) and  $n=1$  ($f(1) = a^1 - b^1$) ✔

Inductive step: for all $k \geq 2$

Induction hypothesis:  $\forall n$ s.t. $1 \leq n \leq k-1$,  $f(n) = a^n - b^n$

To prove:  $f(k) = a^k - b^k$
Recursion & Induction

Let $a, b$ be arbitrary (non-zero) numbers

$f(0) = 0$.  $f(1) = a-b$.  $f(n) = (a+b) \cdot f(n-1) - a \cdot b \cdot f(n-2)$  $\forall n \geq 2$.

Claim:  $f(n) = a^n - b^n$

Base cases:  $n=0$ ($f(0) = a^0 - b^0$) and  $n=1$ ($f(1) = a^1 - b^1$) ✔

Inductive step: for all $k \geq 2$

Induction hypothesis: $\forall n$ s.t. $1 \leq n \leq k-1$,  $f(n) = a^n - b^n$

To prove:  $f(k) = a^k - b^k$

$f(k) = (a+b) \cdot f(k-1) - ab \cdot f(k-2)$

$= (a+b) (a^{k-1} - b^{k-1}) - ab \cdot (a^{k-2} - b^{k-2})$

$= a^{k-1}(a+b-b) - b^{k-1}(a+b-a) = a^k - b^k$