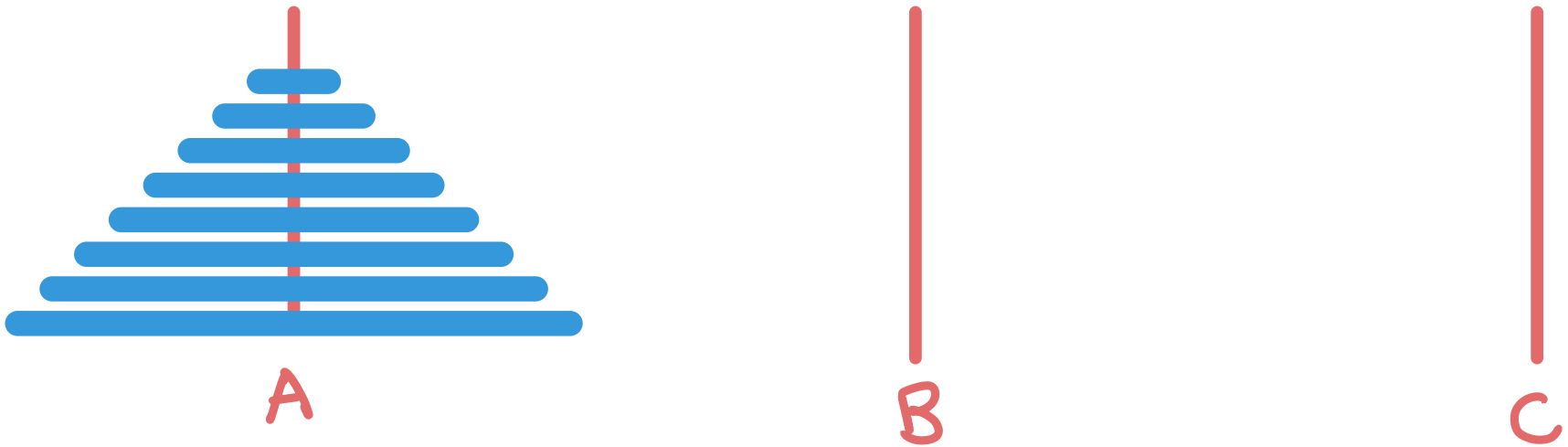
Towers of Hanoi

3 posts A, B, C. n rings of n incr. sizes (top-to-bottom) on A

can move ring from top of one post to top of another

rule: a ring cannot be placed on a smaller ring



Goal: move all n rings from A to B

Induction & Recursion

Lecture 1

Concepts & techniques

$\log(x)$ vs 2^x

recursion

divide & conquer

recursion trees

big-O & big-Ω

Algorithms & applications

Binary search

Merge sort

Sorting lower bounds

Designing a recursive algorithm

Divide-and-conquer

1. Sort first half recursively
2. Sort second half recursively
3. merge sorted halves together

① Recursive specification

`merge-sort`$(A[1..n])$: Given an array $A[1..n]$ of comparable elements, returns an array consisting of the elements of A sorted in increasing order.

② Implement spec

- declares input/output of algo
- "contract" algo must fulfill
- induction hypothesis for correctness

`merge-sort`$(A[1..n])$

```
/* In the base case, the list is so short there is nothing to do. */
1. If n ≤ 1 then return A.                                   ← Base case
/* In the general case, we divide the input in half and recursively sort each half. */
2. Let m = ⌊n/2⌋.
3. B1[1..m] ← merge-sort(A[1..m]).                           ← satisfies recursive spec by induction on n
4. B2[1..n − m] ← merge-sort(A[m + 1..n]).                   ← satisfies recursive spec by induction on n
/* Now we merge the two sorted halves in linear time. */
5. return merge(B1, B2).
```

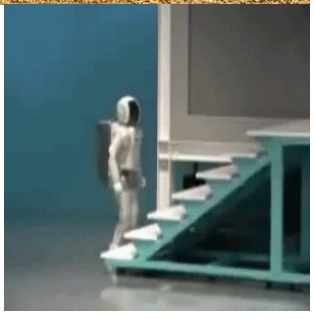
③ Prove correctness

argue algo satisfies recursive spec for all n by induction on n

Induction

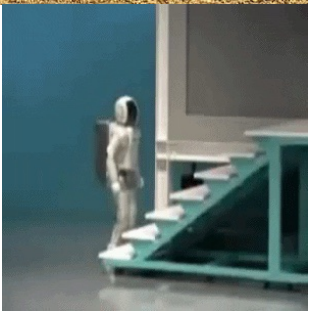
Climbing stairs

how would you explain to someone how to climb a long staircase?



Climbing stairs

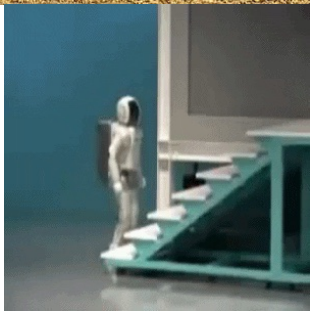
how would you explain to someone how to climb a long staircase?



Let's take it one step at a time. Say you're at a particular step. Now put one foot on the next step, and use that leg to push your body up and put the other foot on that step, so now your entire mass is on the next step. You can use either foot for the first step. Now if you keep repeating this for each step you will eventually get to the top.

Climbing stairs

how would you explain to someone how to climb a long staircase?

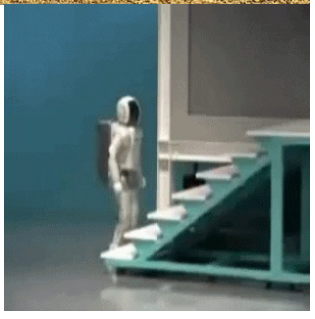


recognize that each step is identical

Let's take it one step at a time. Say you're at a particular step. Now put one foot on the next step, and use that leg to push your body up and put the other foot on that step, so now your entire mass is on the next step. You can use either foot for the first step. Now if you keep repeating this for each step you will eventually get to the top.

Climbing stairs

how would you explain to someone how to climb a long staircase?



recognize that each step is identical

Let's take it one step at a time. Say you're at a particular step. Now put one foot on the next step, and use that leg to push your body up and put the other foot on that step, so now your entire mass is on the next step. You can use either foot for the first step. Now if you keep repeating this for each step you will eventually get to the top.

how can we prove this?

"Proof" by induction

Let's take it one step at a time. Say you're at a particular step. Now put one foot on the next step, and use that leg to push your body up and put the other foot on that step, so now your entire mass is on the next step. You can use either foot for the first step. Now if you keep repeating this for each step you will eventually get to the top.

(assume instructions work for 1 step)

Base case: $n = 1$ stair.

General case: $n > 1$ stairs

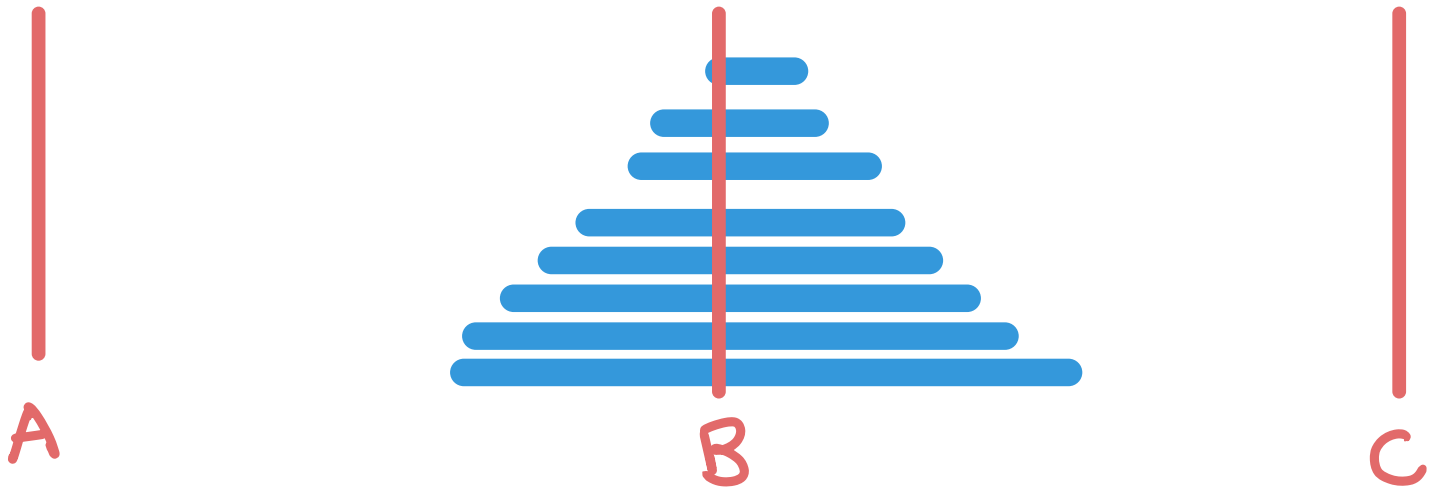
- instructions work for 1 step
- $n-1$ steps remain
- By induction on n, instructions work for staircase of $n-1$ remaining steps

Towers of Hanoi

3 posts A, B, C. n rings of n incr. sizes (top-to-bottom) on A

can move ring from top of one post to top of another

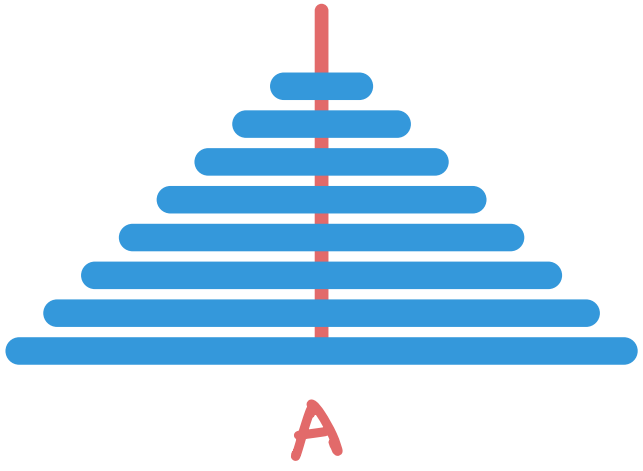
rule: a ring cannot be placed on a smaller ring



Goal: move all n rings from A to B

① Recursive specification

ToH(n, A, B, C): Given n rings on pole A, move all the rings to pole B (while following rules)



Designing a recursive algorithm

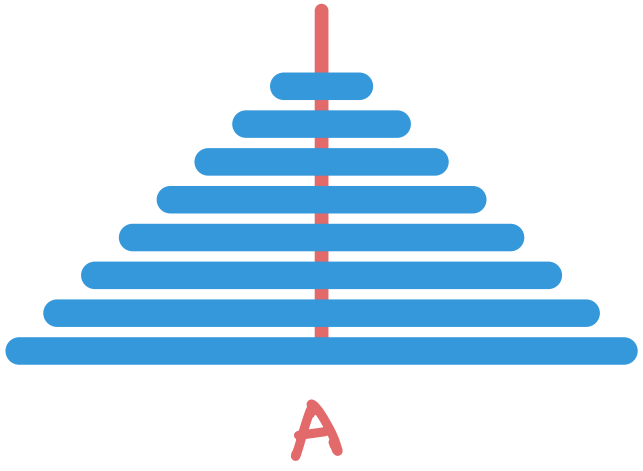
① Recursive specification
- declares input/output of algo
- "contract" algo must fulfill
- induction hypothesis for correctness

② Implement spec

③ Prove correctness
argue algo satisfies recursive spec for all n by induction on n

① Recursive specification

ToH(n, A, B, C): assuming the n smallest rings are on A, moves top n rings from post A to post B



Designing a recursive algorithm

① Recursive specification
- declares input/output of algo
- "contract" algo must fulfill
- induction hypothesis for correctness

② Implement spec

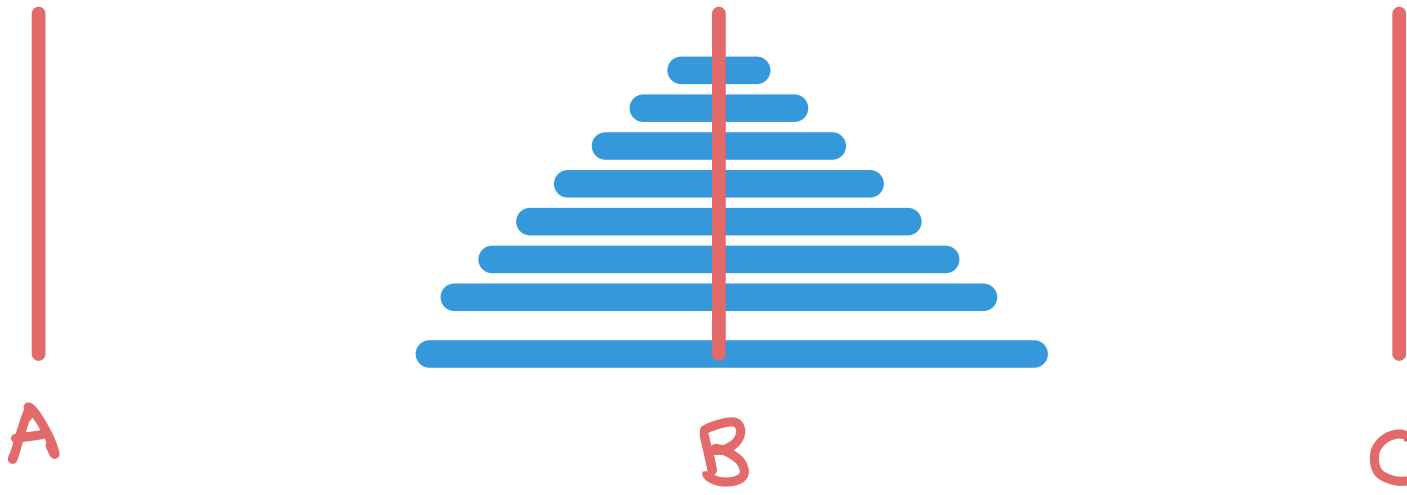
③ Prove correctness
argue algo satisfies recursive spec for all n by induction on n

ToH(n, A, B, C): assuming the n smallest rings are on A, moves top n rings from post A to post B

② Implement spec

TOH(n, A, B, C):

1. If n=0: return
2. else n>1:
   a. ToH(n-1, A, C, B)
   b. move 1 from A to B
   c. ToH(n-1, C, B, A)



Designing a recursive algorithm

① Recursive specification
- declares input/output of algo
- "contract" algo must fulfill
- induction hypothesis for correctness

② Implement spec

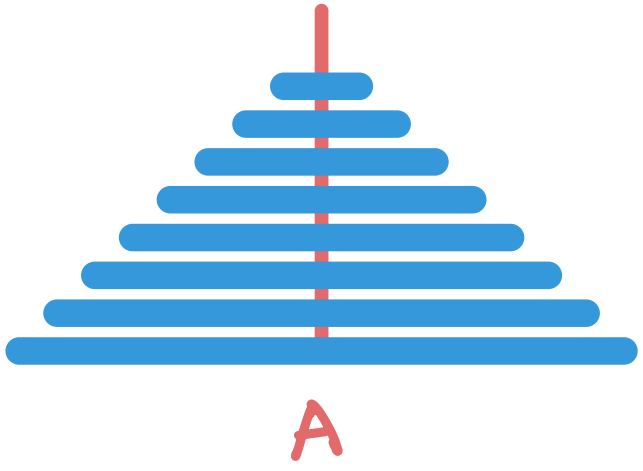
③ Prove correctness
argue algo satisfies recursive spec for all n by induction on n

ToH(n, A, B, C): assuming the n smallest rings are on A, moves top n rings from post A to post B

② Implement spec

TOH(n, A, B, C):

1. if n > 0:
   - A. ToH(n-1, A, C, B)
   - B. move ring from A to B
   - C. ToH(n-1, C, B, A)



Designing a recursive algorithm

① Recursive specification
- declares input/output of algo
- "contract" algo must fulfill
- induction hypothesis for correctness

② Implement spec

③ Prove correctness
argue algo satisfies recursive spec for all n by induction on n

ToH(n, A, B, C): assuming the n smallest rings are on A, moves top n rings from post A to post B

Designing a recursive algorithm

① Recursive specification
- declares input/output of algo
- "contract" algo must fulfill
- induction hypothesis for correctness

② Implement spec

③ Prove correctness
argue algo satisfies recursive spec for all n by induction on n

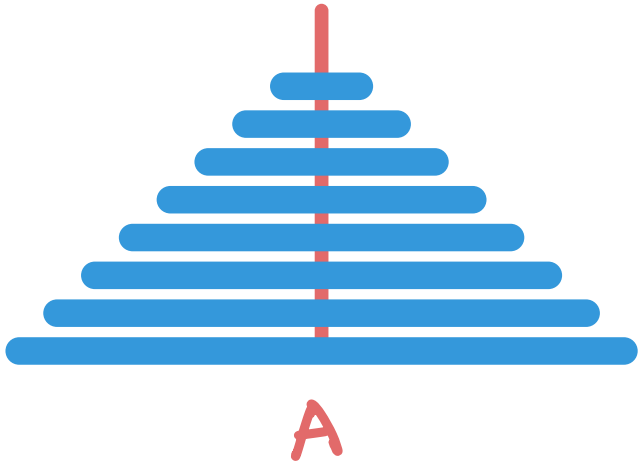
③ Prove correctness

Base case n=0.

Case n>0:

"By induction on n, ToH(n-1, A, C, B) moves n-1 from ~~~

```
TOH(n, A, B, C):
1. if n>0:
   A. ToH(n-1, A, C, B)
   B. move ring from A to B
   C. ToH(n-1, C, B, A)
```



ToH(n, A, B, C): assuming the n smallest rings are on A, moves top n rings from post A to post B

Designing a recursive algorithm

① Recursive specification
- declares input/output of algo
- "contract" algo must fulfill
- induction hypothesis for correctness

② Implement spec

③ Prove correctness
argue algo satisfies recursive spec for all n by induction on n

③ Prove correctness

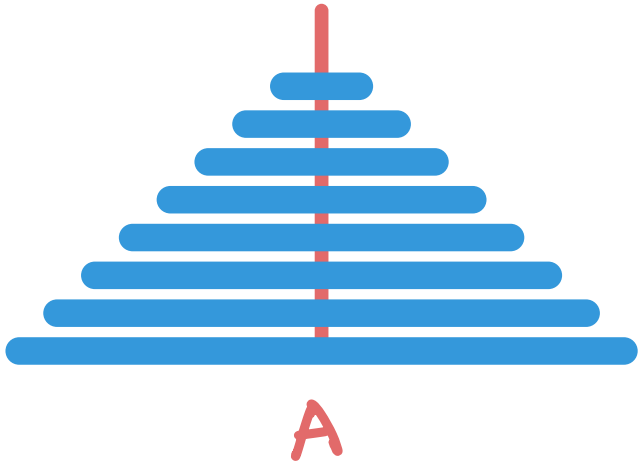
Base case n=0. ✓

Case n>0:

recursive calls correct by induction

⇒ correct for n

```
TOH(n, A, B, C):
1. if n>0:
   A. TOH(n-1, A, C, B)
   B. move ring from A to B
   C. TOH(n-1, C, B, A)
```

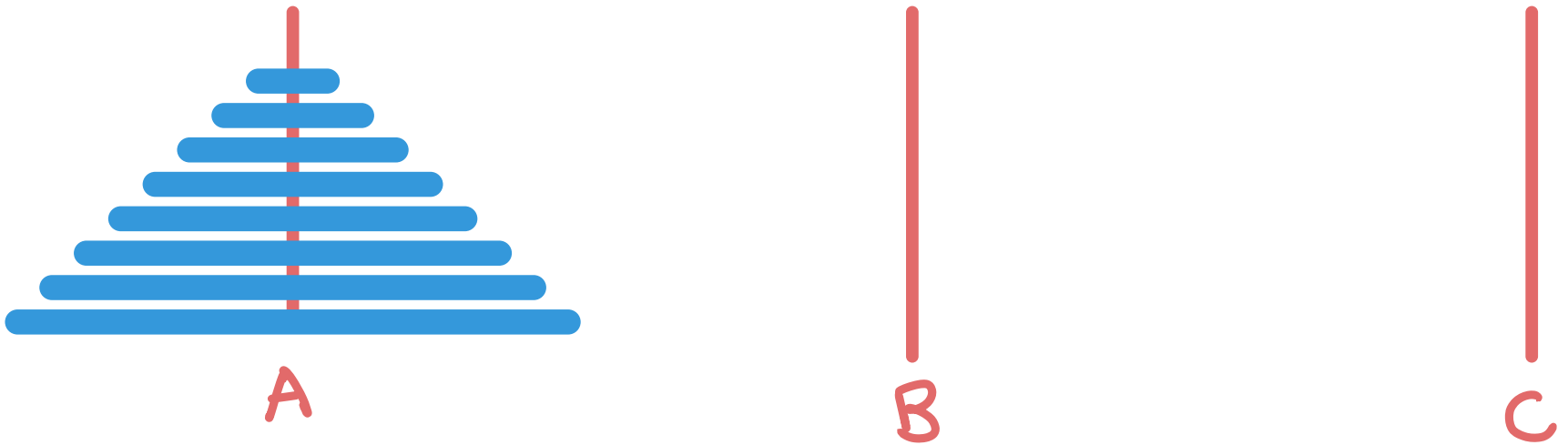


Towers of Hanoi

3 posts A, B, C. n rings of n incr. sizes (top-to-bottom) on A

can move ring from top of one post to top of another

rule: a ring cannot be placed on a smaller ring

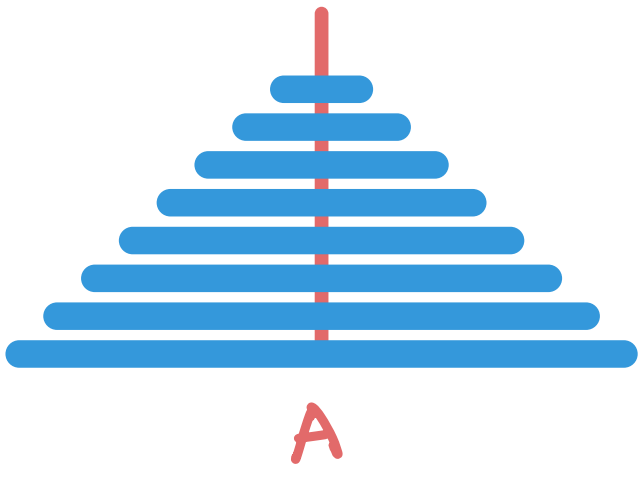


Goal: move all n rings from A to B

w/ minimum # of moves

① Recursive specification

ToH(n, A, B, C):



Designing a recursive algorithm

① Recursive specification
- declares input/output of algo
- "contract" algo must fulfill
- induction hypothesis for correctness

② Implement spec

③ Prove correctness
argue algo satisfies recursive spec for all n by induction on n

① Recursive specification

ToH(n, A, B, C): assuming the n smallest rings are on A, moves top n rings from post A to post B

w/ the minimum # of moves

everything else stays the same!

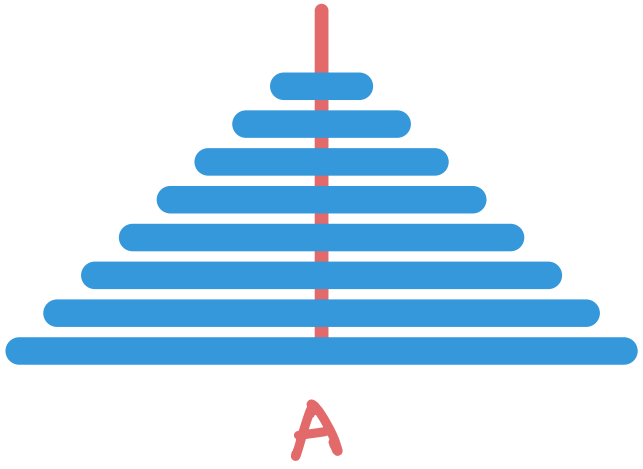
Designing a recursive algorithm

① Recursive specification
- declares input/output of algo
- "contract" algo must fulfill
- induction hypothesis for correctness

② Implement spec

③ Prove correctness
argue algo satisfies recursive spec for all n by induction on n

```
TOH(n, A, B, C):
1. if n>0:
   A. ToH(n-1, A, C, B)
   B. move ring from A to B
   C. ToH(n-1, C, B, A)
```



Greatest common divisor

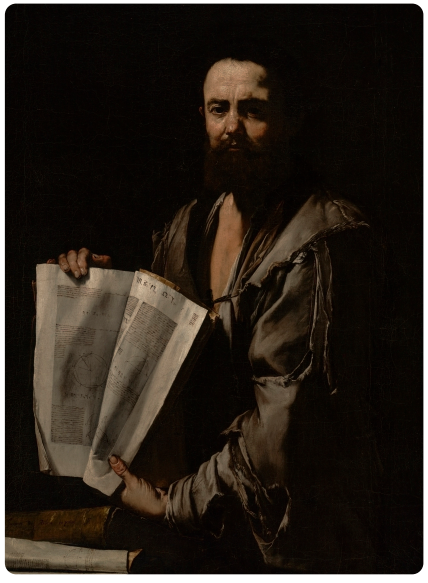
given $x, y \in \mathbb{N}$, compute largest integer q that divides x and y

e.g. $GCD(146740, 12180) = 580$

Greatest common divisor

given $x, y \in \mathbb{N}$, compute largest integer q that divides x and y

e.g. GCD(146740, 12180) = 580



Euclid

let $x \geq y$.

q divides x and y $\iff$ q divides y and $x-y$

① Recursive specification

Designing a recursive algorithm

① Recursive specification
- declares input/output of algo
- "contract" algo must fulfill
- induction hypothesis for correctness

② Implement spec

③ Prove correctness
argue algo satisfies recursive spec for all n by induction on n

let $x \geq y$.

q divides x and y $\iff$ q divides y and x-y

GCD(x, y): Assuming $x \geq y$, returns the GCD of x and y.

① Recursive specification

Designing a recursive algorithm
① Recursive specification
- declares input/output of algo
- "contract" algo must fulfill
- induction hypothesis for correctness

② Implement spec

③ Prove correctness
argue algo satisfies recursive spec for all n by induction on n

let $x \geq y$.
q divides x and y $\iff$ q divides y and x-y

GCD(x, y): Given two positive integers $x, y \in \mathbb{Z}_{\geq 0}$ with $x \geq y$ and $x > 0$, returns the greatest common divisor of x and y.

$y \geq 0$

GCD(x,y):
// assert $x \geq y \geq 0$
1. if $y = 0$ then return x
2. return GCD(a,b) w/ $a = \max\{x-y, y\}$
$b = \min\{x-y, y\}$

GCD(x, y): Given two positive integers $x, y \in \mathbb{Z}_{\geq 0}$ with $x \geq y$ and $x > 0$, returns the greatest common divisor of x and y.

② Implement spec

GCD(x, y)

/ Given two positive integers $x, y \in \mathbb{Z}_{\geq 0}$ with $x \geq y$ and $x > 0$, returns the greatest common divisor of x and y. */*

1. If $y = 0$ then return x.
2. Return GCD$(\max\{x - y, y\}, \min\{x - y, y\})$.

Induction on y:

y=0 ✓

y>0:

↓

Designing a recursive algorithm

① Recursive specification
- declares input/output of algo
- "contract" algo must fulfill
- induction hypothesis for correctness

② Implement spec

③ Prove correctness
argue algo satisfies recursive spec for all n by induction on n

let $x \geq y$.

q divides x and y <=> q divides y and x-y

GCD(x,y): Given two positive integers $x, y \in \mathbb{Z}_{\geq 0}$ with $x \geq y$ and $x > 0$, returns the greatest common divisor of x and y.

GCD(x,y)

/ Given two positive integers $x, y \in \mathbb{Z}_{\geq 0}$ with $x \geq y$ and $x > 0$, returns the greatest common divisor of x and y. */*

1. If $y = 0$ then return x.
2. Return GCD$(\max\{x-y, y\}, \min\{x-y, y\})$.

Designing a recursive algorithm
1. Recursive specification
   - declares input/output of algo
   - "contract" algo must fulfill
   - induction hypothesis for correctness
2. Implement spec
3. Prove correctness
   argue algo satisfies recursive spec for all n by induction on n

let $x \geq y$.
q divides x and y <=> q divides y and x-y

3 Prove correctness

we ~~claim~~ prove that GCD(x,y) fulfills specification by induction on x+y

$x+y=1$: $x=2$, $y=0$, returns 1 ✓

$x+y>1$:
- if $y=0$, then ✓
- if $y>1$: $x-y+y = x < x+y$ b/c $y \geq 1$

Assume by induction that GCD(x,y) is correct for all $x+y<n$

GCD(x,y): Given two positive integers $x, y \in \mathbb{Z}_{\geq 0}$ with $x \geq y$ and $x > 0$, returns the greatest common divisor of x and y.

GCD(x,y)

/ Given two positive integers $x, y \in \mathbb{Z}_{\geq 0}$ with $x \geq y$ and $x > 0$, returns the greatest common divisor of x and y. */*

1. If $y = 0$ then return x.
2. Return GCD$(\max\{x-y, y\}, \min\{x-y, y\})$.

Designing a recursive algorithm

① Recursive specification
- declares input/output of algo
- "contract" algo must fulfill
- induction hypothesis for correctness

② Implement spec

③ Prove correctness
argue algo satisfies recursive spec for all n by induction on n

let $x \geq y$.

q divides x and y <=> q divides y and x-y

Running time

1000 1

$O(x)$

GCD(x,y): Given two positive integers $x, y \in \mathbb{Z}_{\geq 0}$ with $x \geq y$ and $x > 0$, returns the greatest common divisor of x and y.

GCD(x,y)

/ Given two positive integers $x, y \in \mathbb{Z}_{\geq 0}$ with $x \geq y$ and $x > 0$, returns the greatest common divisor of x and y. */*

1. If $y = 0$ then return x.
2. Return GCD$(\max\{x-y, y\}, \min\{x-y, y\})$.

Designing a recursive algorithm

① Recursive specification
- declares input/output of algo
- "contract" algo must fulfill
- induction hypothesis for correctness

② Implement spec

③ Prove correctness
argue algo satisfies recursive spec for all n by induction on n

let $x \geq y$.

q divides x and y <=> q divides y and x-y

Running time

each iteration decreases $x+y$ by 1

$O(x+y)$ time (assuming $O(1)$-time arithmetic)

Is this "linear time"?

x, y take $\log(x)$, $\log(y)$ bits to represent

$O(x+y)$ is *exponential* in the input size

Bad case

x	y
100	2
98	2
96	2
	⋮
2	2
2	0

`GCD`(x,y): Given two positive integers $x, y \in \mathbb{Z}_{\geq 0}$ with $x \geq y$ and $x > 0$, returns the greatest common divisor of x and y.

`GCD`(x,y)

/ Given two positive integers $x, y \in \mathbb{Z}_{\geq 0}$ with $x \geq y$ and $x > 0$, returns the greatest common divisor of x and y. */*

1. If $y = 0$ then return x.
2. Return `GCD`$(\max\{x-y, y\}, \min\{x-y, y\})$.

let $x \geq y$ and $r = x \bmod y$

$x = ky + r$ w/ $k \in \mathbb{N}$, $ky \leq x$

q divides x and y $\Leftrightarrow$ q divides y and r

GCD(x, y): Given two positive integers $x, y \in \mathbb{Z}_{\geq 0}$ with $x \geq y$ and $x > 0$, returns the greatest common divisor of x and y.

$x \geq y$ and $r = x \bmod y$
$x = ky + r$ w/ $k \in \mathbb{N}$, $ky \leq x$
q divides x and y $\Leftrightarrow$ q divides y and r

GCD(x, y)

/ Given two positive integers $x, y \in \mathbb{Z}_{\geq 0}$ with $x \geq y$ and $x > 0$, returns the greatest common divisor of x and y. */*

1. If $y = 0$ then return x.
2. Return GCD$(y, x \bmod y)$.

Proof by induction on y

Base case $y=0$ ✓

General case $y>0$:

GCD(x, y): Given two positive integers $x, y \in \mathbb{Z}_{\geq 0}$ with $x \geq y$ and $x > 0$, returns the greatest common divisor of x and y.

$x \geq y$ and $r = x \bmod y$
$x = ky + r$ w/ $k \in \mathbb{N}$, $ky \leq x$
q divides x and y $\Leftrightarrow$ q divides y and r

GCD(x, y)

/ Given two positive integers $x, y \in \mathbb{Z}_{\geq 0}$ with $x \geq y$ and $x > 0$, returns the greatest common divisor of x and y. */*

1. If $y = 0$ then return x.
2. Return GCD$(y, x \mod y)$.

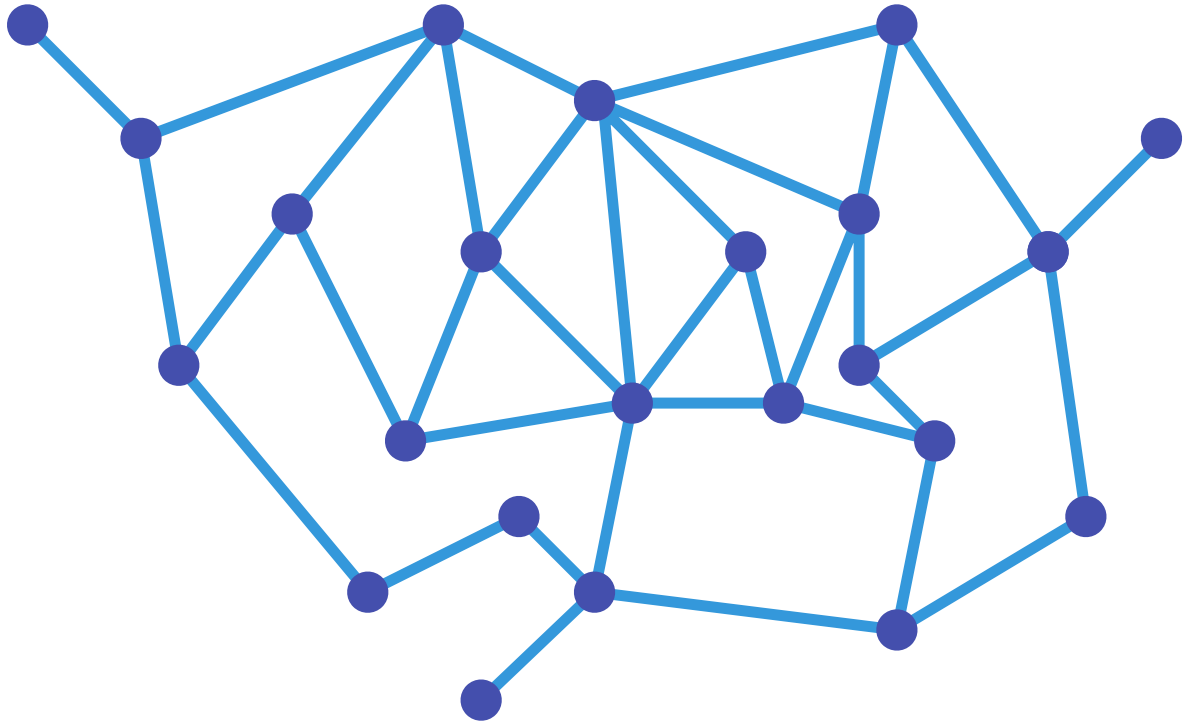
Running time

$x \bmod y$

$y \geq \frac{x}{2}$ $\quad r \leq \frac{x}{2}$

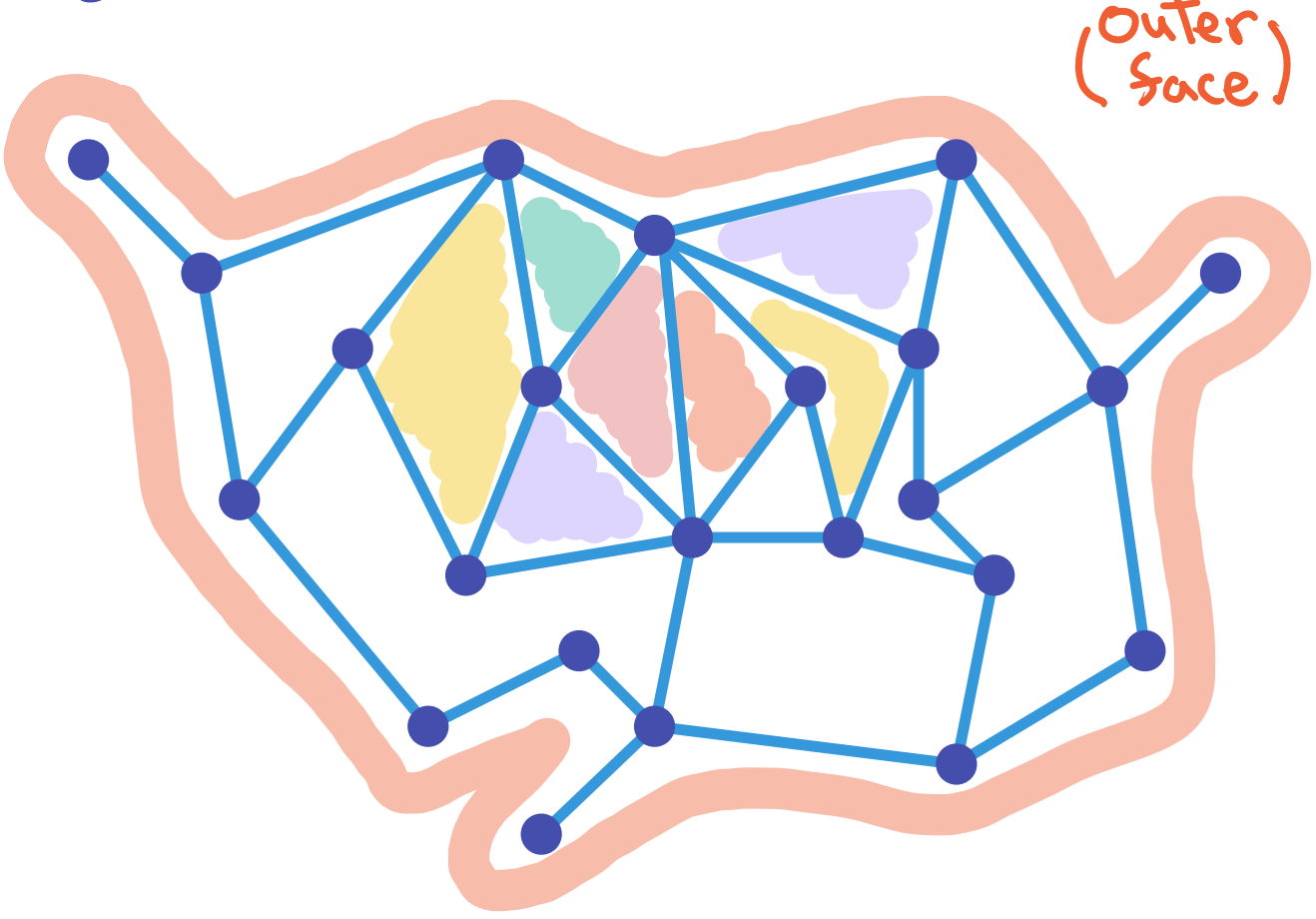
$y \leq \frac{x}{2}$: $\quad r \leq \frac{x}{2}$

Planar graphs



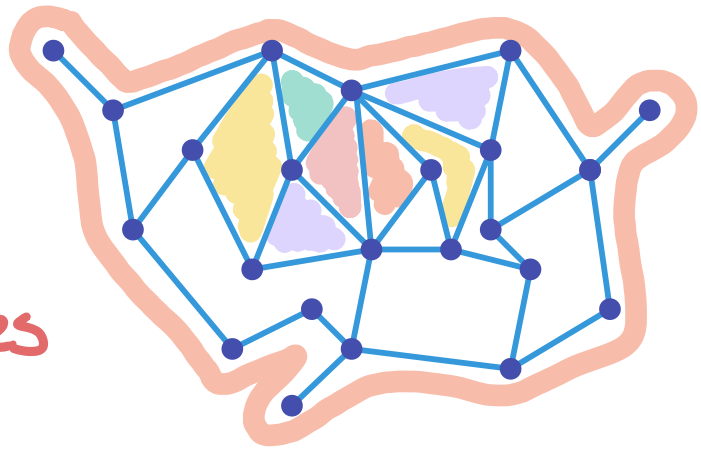
graph can be drawn so that edges only interect at their endpoints (aka embedding in the plane)

Planar graphs



for fixed embedding, graph divides plane into regions called **faces**.

Euler's Formula



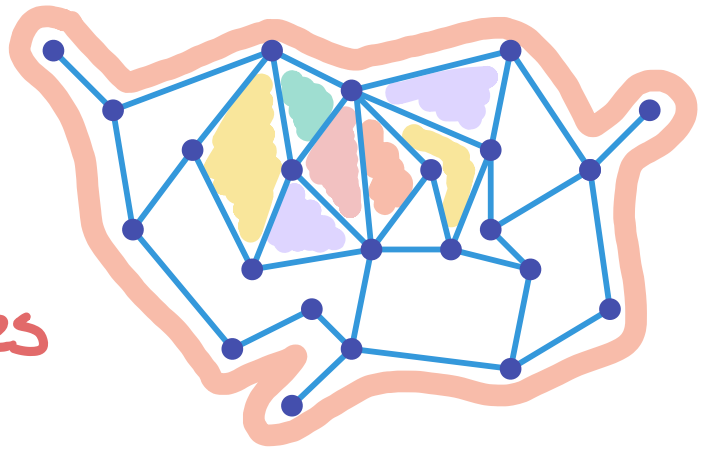
$G=(V,E)$ connected planar graph.

$m=\#$ edges, $n=\#$ vertices, $p=\#$ faces

then $n-m+p=2$

Proof

Euler's Formula



$G=(V,E)$ connected planar graph.

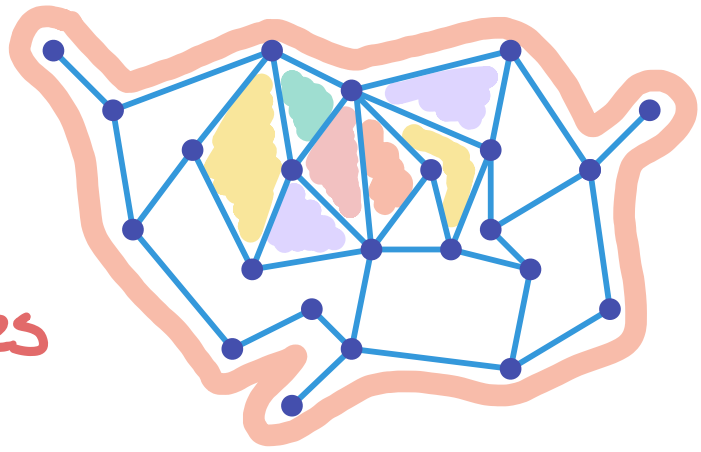
$m=\#$ edges, $n=\#$ vertices, $p=\#$ faces

then $n-m+p=2$

Proof by induction on n

Base case $n=1$

Euler's Formula



$G=(V,E)$ connected planar graph.

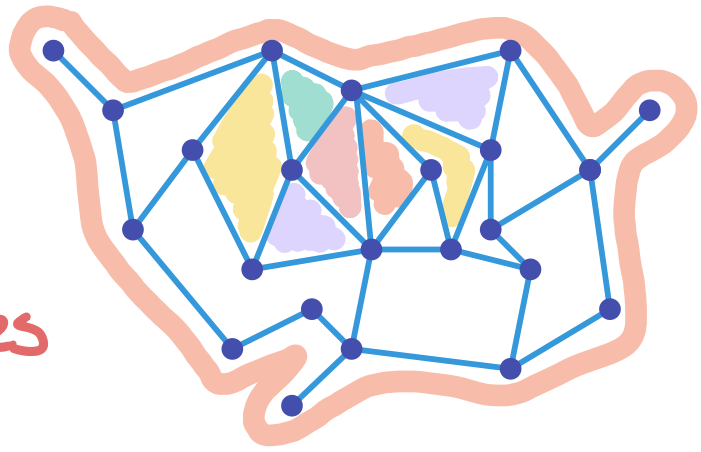
$m=\#$ edges, $n=\#$ vertices, $p=\#$ faces

then $n-m+p=2$

Proof by induction on n

General case $n>1$.

Euler's Formula



$G=(V,E)$ connected planar graph.

$m=\#$ edges, $n=\#$ vertices, $p=\#$ faces

then $n-m+p=2$

(no parallel edges or self-loops)

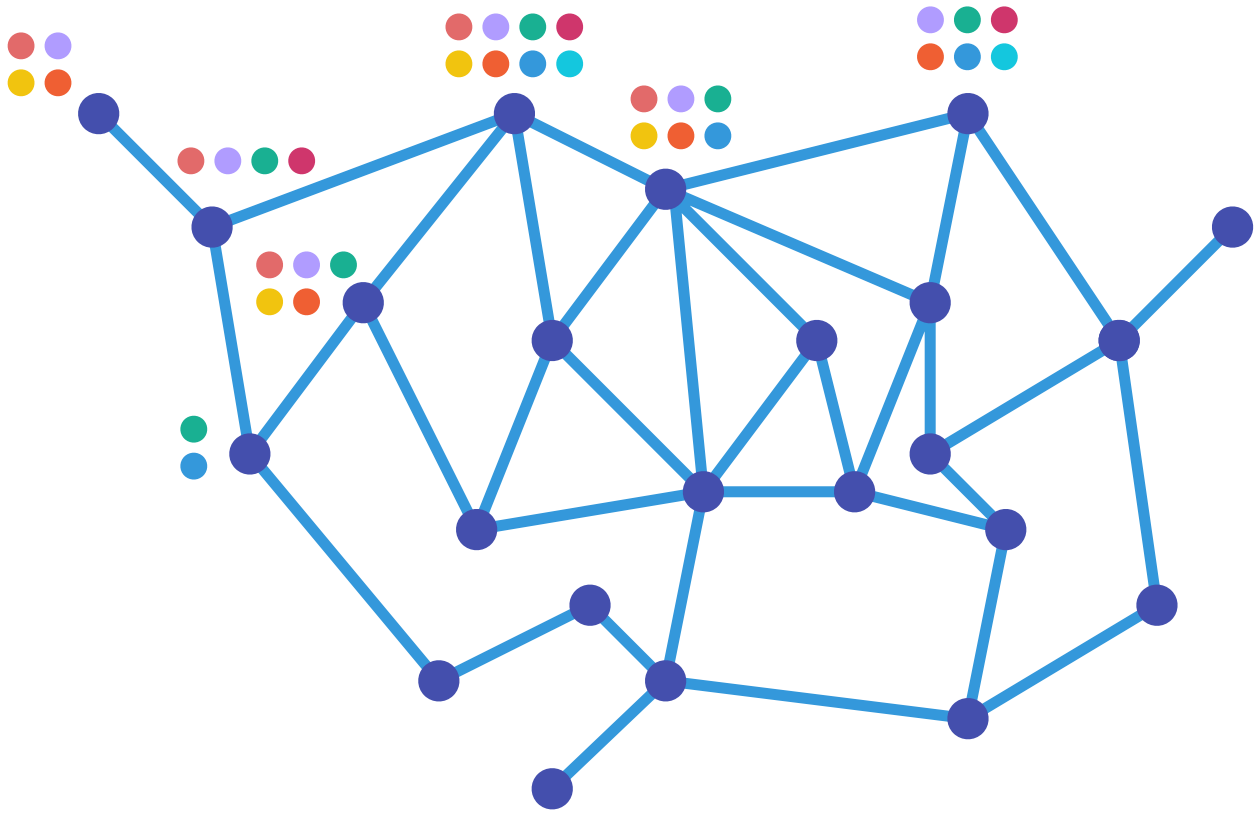
Corollary for simple planar G, and $n \geq 3$,

$$m \leq 3n-6$$

proof

List coloring

each vertex v has set of colors $C(v)$



Goal: choose color from $C(v)$ for all v so that each edge has 2 diff. colors

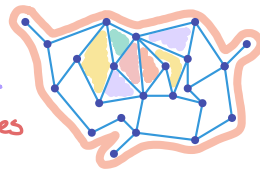
Theorem: all planar graphs are 6-list colorable

Euler's Formula

$G=(V,E)$ connected planar graph.

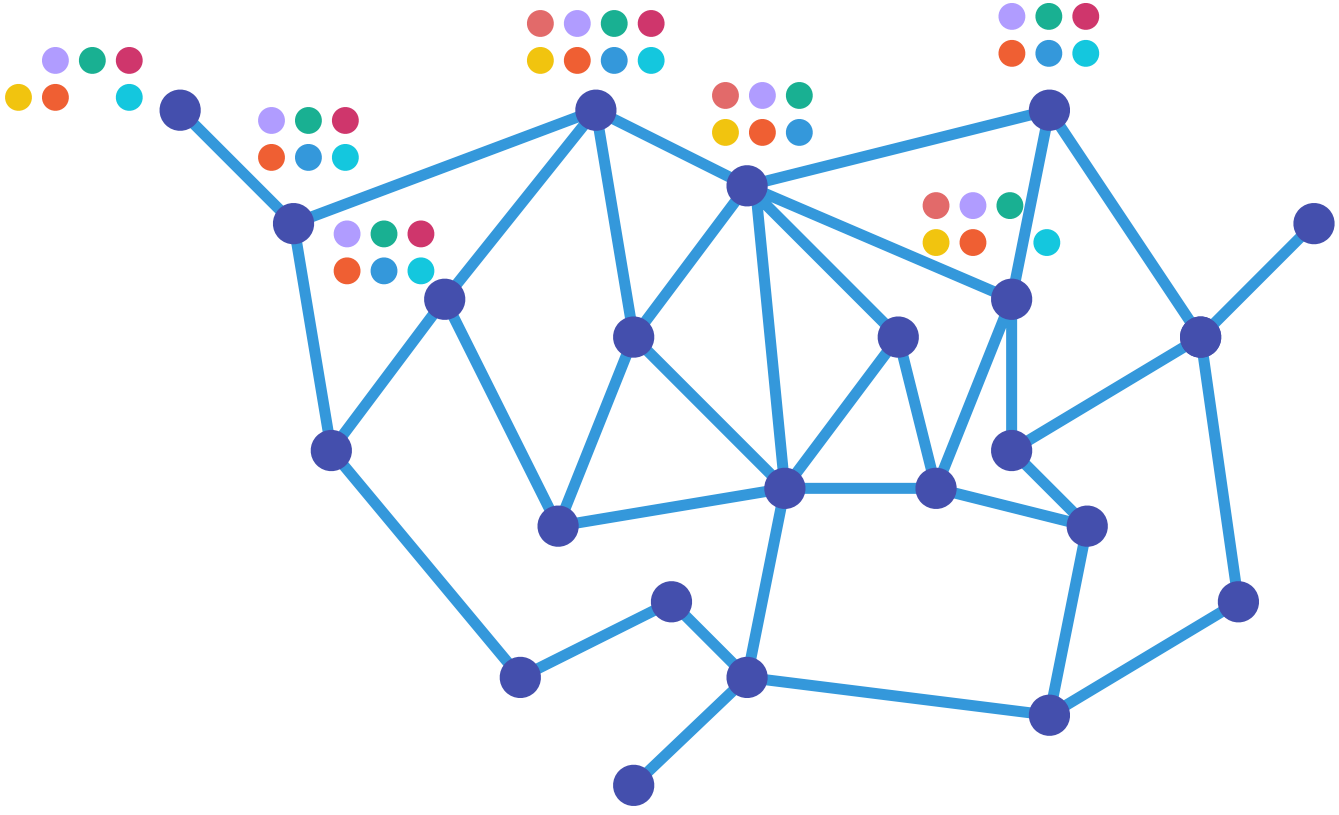
m = # edges, n = # vertices, p = # faces

then $n-m+p=2$



Corollary for (no parallel edges or self-loops) simple planar G, and $n \geq 3$,

$m \leq 3n-6$



Theorem: all planar graphs are 6-list colorable

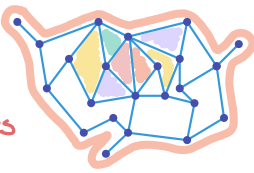
proof

Euler's Formula

$G=(V,E)$ connected planar graph.

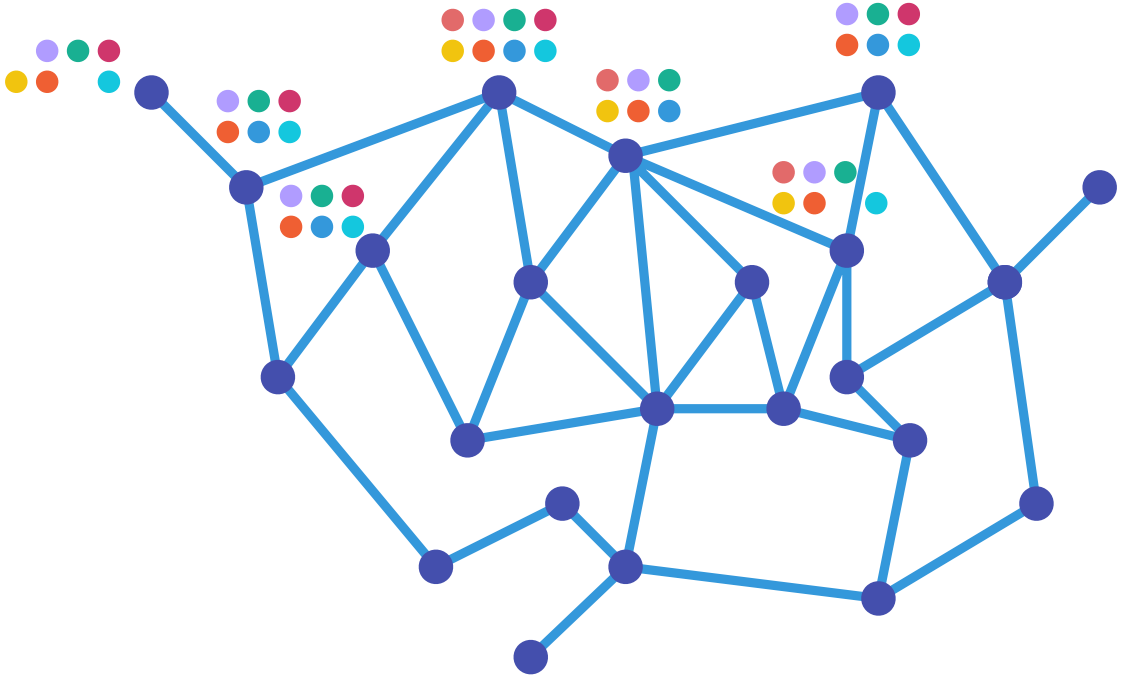
m = # edges, n = # vertices, p = # faces

then $n-m+p=2$

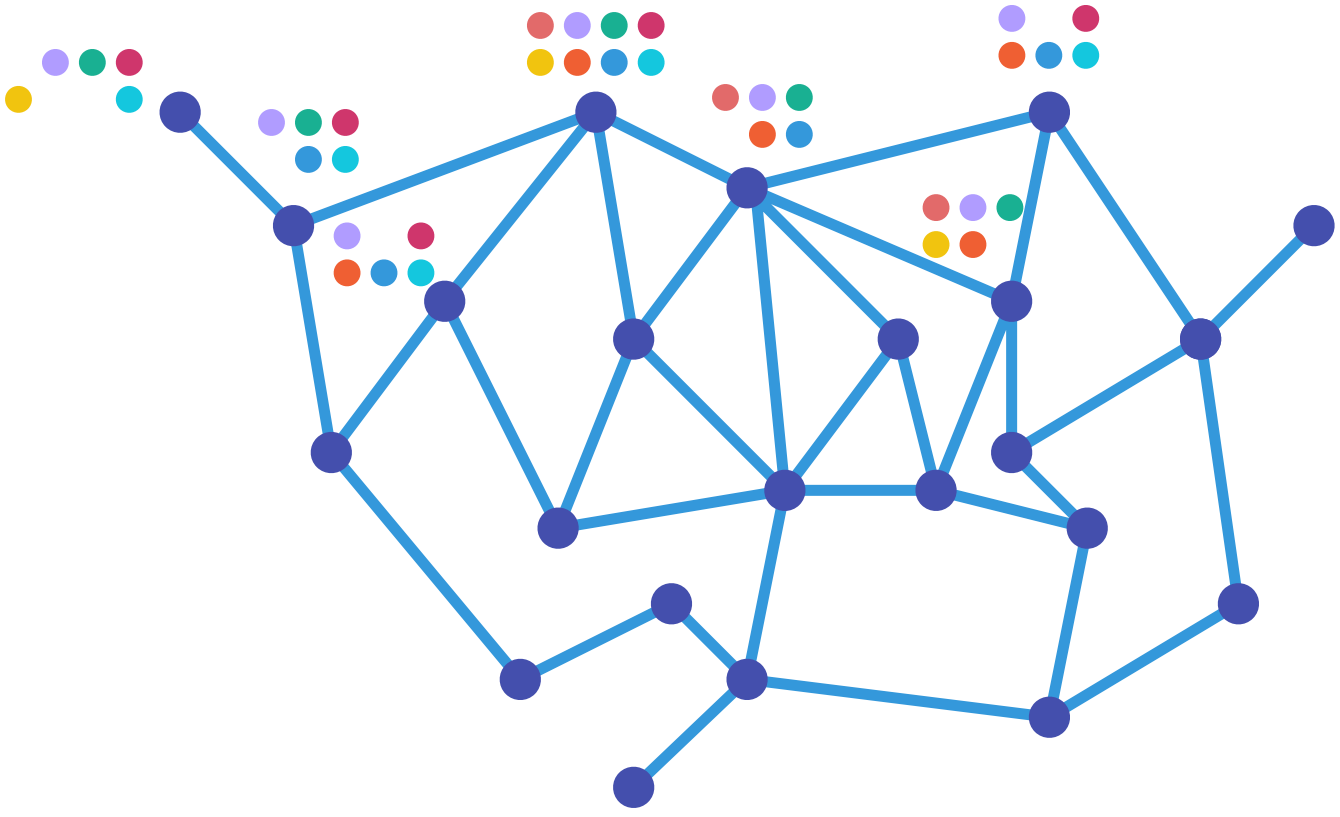


Corollary for (no parallel edges or self-loops) simple planar G, and $n \geq 3$,

$m \leq 3n-6$



Theorem: all planar graphs are 5-list colorable



Theorem: all planar graphs are 5-list colorable

Proof: use stronger induction hypothesis:

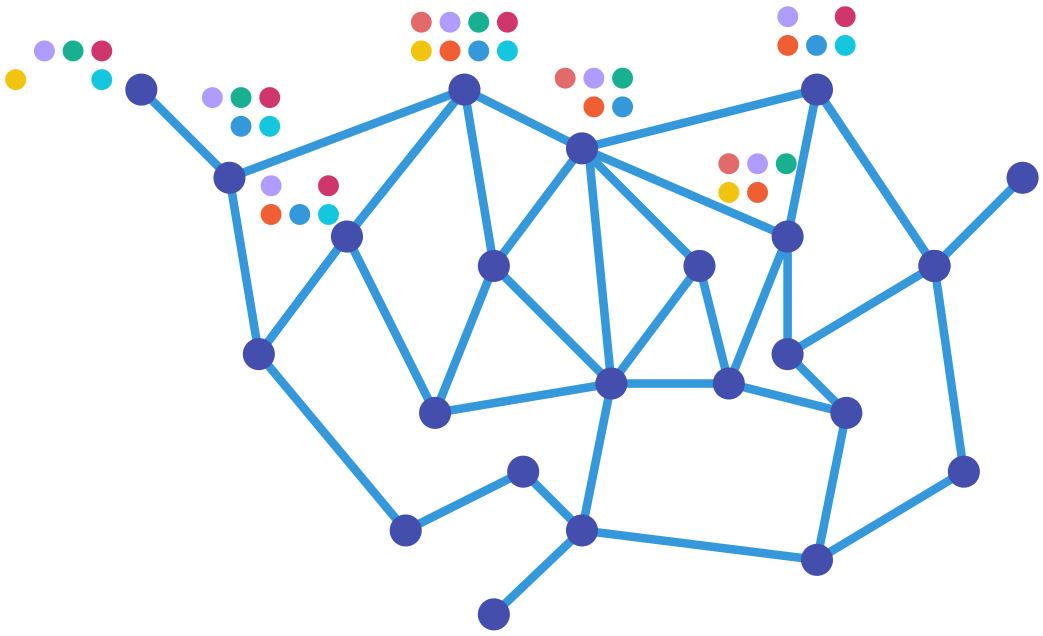
Lemma 2.7. *Let $G = (V, E)$ be a planar graph, and let $B = (V_B, E_B)$ be the boundary of G.*

(a) Two vertices $x, y \in V_B$ adjacent in B are already colored α and β, respectively.

(b) $|C(v)| \geq 3$ for all $v \in V_B$.

(c) $|C(v)| \geq 5$ for all $v \in V \setminus V_B$.

Then the coloring of x and y can be extended to a proper list coloring of G.



Lemma 2.7. *Let $G = (V, E)$ be a planar graph, and let $B = (V_B, E_B)$ be the boundary of G.*

(a) Two vertices $x, y \in V_B$ adjacent in B are already colored α and β, respectively.

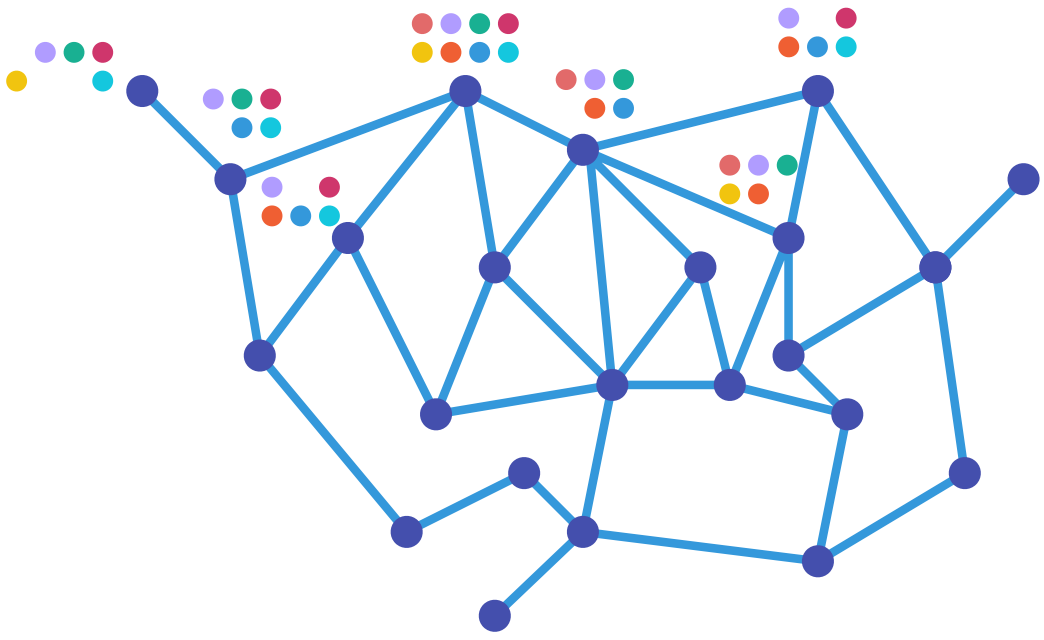
(b) $|C(v)| \geq 3$ for all $v \in V_B$.

(c) $|C(v)| \geq 5$ for all $v \in V \setminus V_B$.

Then the coloring of x and y can be extended to a proper list coloring of G.

proof by induction on n.

base case: n=3



Lemma 2.7. *Let $G = (V, E)$ be a planar graph, and let $B = (V_B, E_B)$ be the boundary of G.*

(a) Two vertices $x, y \in V_B$ adjacent in B are already colored α and β, respectively.

(b) $|C(v)| \geq 3$ for all $v \in V_B$.

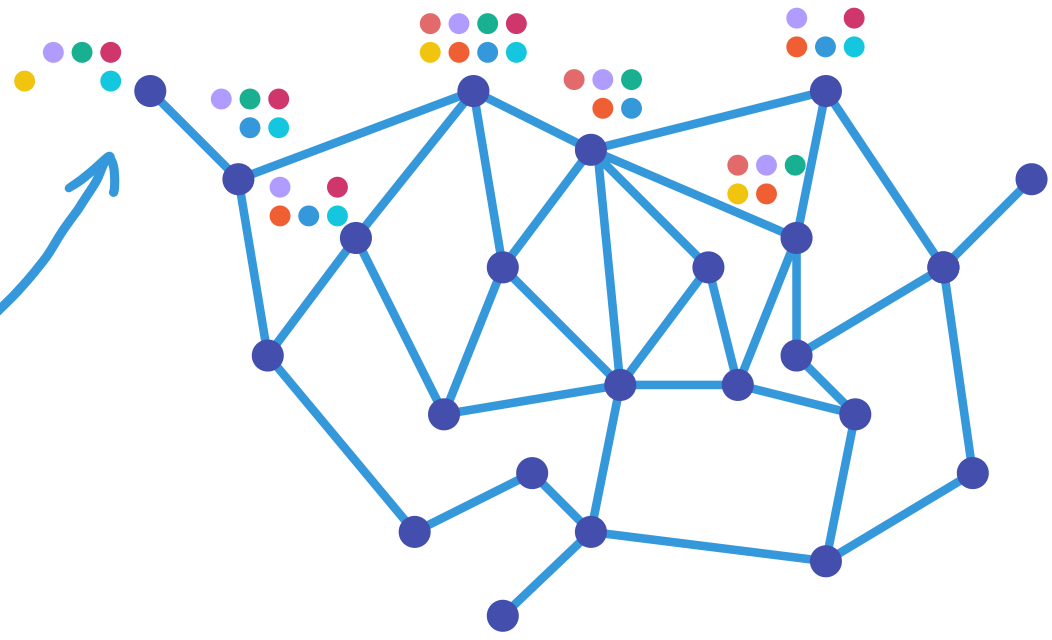
(c) $|C(v)| \geq 5$ for all $v \in V \setminus V_B$.

Then the coloring of x and y can be extended to a proper list coloring of G.

proof by induction on n.

let n>3.

Case 1: B has a leaf



let $n > 3$.

Case 2: B is a cycle

Case 2.a: B has a chord

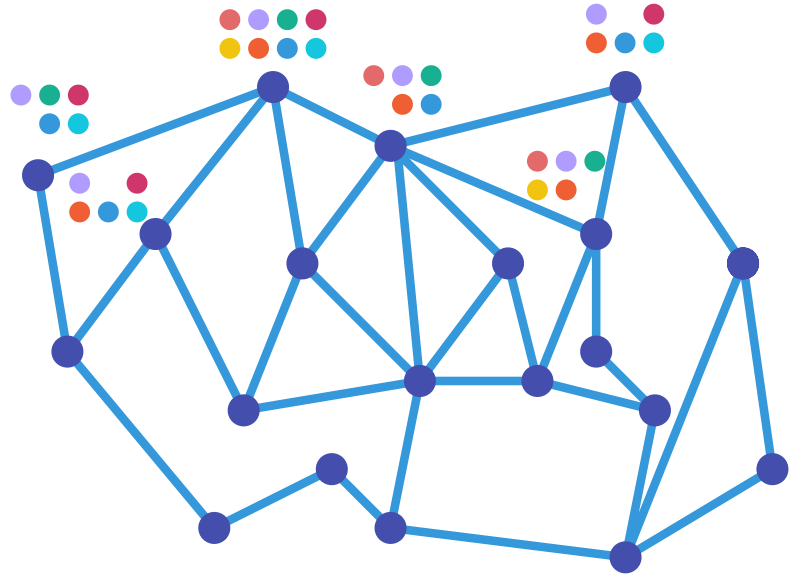
Lemma 2.7. *Let $G = (V, E)$ be a planar graph, and let $B = (V_B, E_B)$ be the boundary of G.*

(a) Two vertices $x, y \in V_B$ adjacent in B are already colored α and β, respectively.

(b) $|C(v)| \geq 3$ for all $v \in V_B$.

(c) $|C(v)| \geq 5$ for all $v \in V \setminus V_B$.

Then the coloring of x and y can be extended to a proper list coloring of G.



let $n > 3$.

Case 2: B is a cycle

Case 2.b: B has no chord

Lemma 2.7. *Let $G = (V, E)$ be a planar graph, and let $B = (V_B, E_B)$ be the boundary of G.*

(a) Two vertices $x, y \in V_B$ adjacent in B are already colored α and β, respectively.

(b) $|C(v)| \geq 3$ for all $v \in V_B$.

(c) $|C(v)| \geq 5$ for all $v \in V \setminus V_B$.

Then the coloring of x and y can be extended to a proper list coloring of G.