Introduction to Discrete Math

Felipe P. Vista IV

Course Outline

- Mathematical Thinking
  - Convincing Arguments, Find Example, Recursion, Logic, Invariants
- Probability & Combinatronics
  - Counting, Probability, Random Variables
- Graph Theory
  - Graphs (cycles, classes, parameters)
- Number Theory & Cryptography
  - Arithmetic in modular form
  - Intro to Cryptography

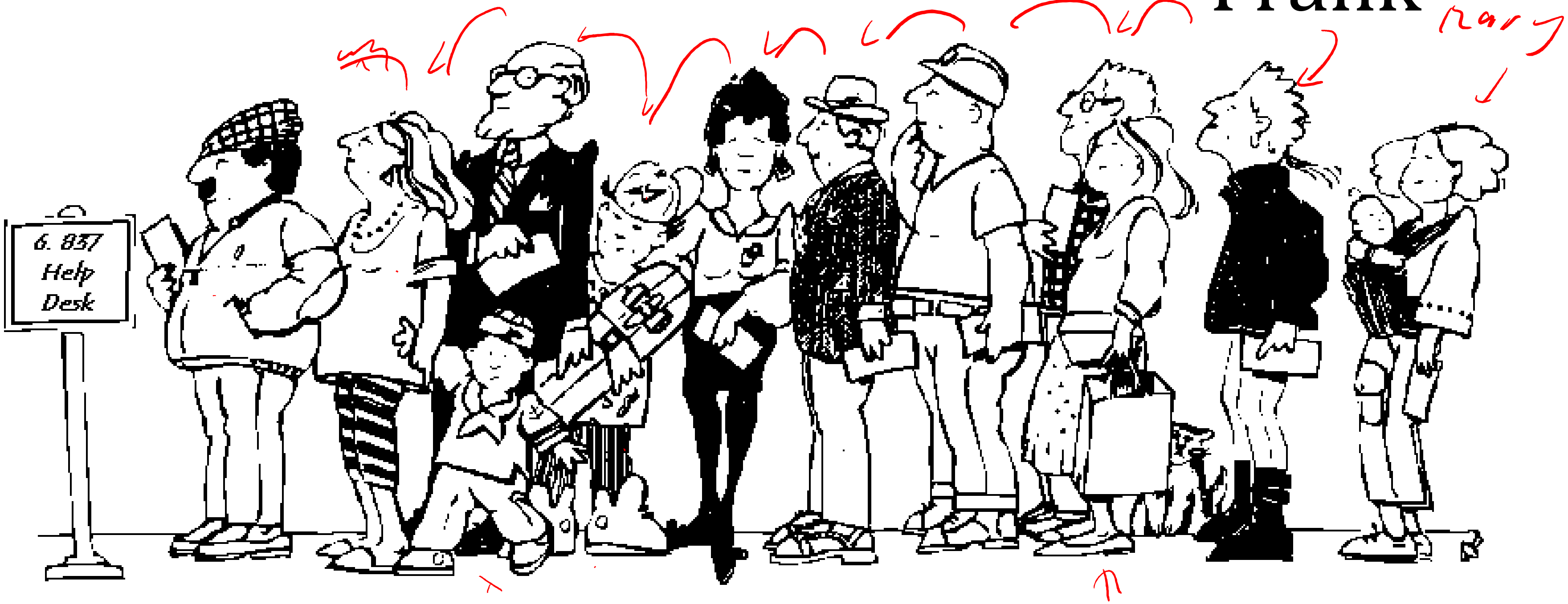
Mathematical Thinking – Recursion & Induction

RECURSION

- Recursion
- The Coin Problem
- Hanoi Towers

Line to process a paper

Frank



http://groups.csail.mit.edu/graphics/classes/6.837/F98/Lecture5/longline.gif

Compute Queue Length

1) Mary gets in line
2) She wonders how many people before her?
3) Mary asks Frank (in front of her)
    - *Could you please tell how many people ahead of you?*
4) Now, Frank has the same problem
    - But Frank was able to find out there are 8
5) Now Mary knows there are 9 people in front

Recursive Program

Algorithm:

numberOfPeopleInFront(F):

if there is no one before A:

return 0

F ← number of people before A

return *numberOfPeopleInFront*(F) + 1

Factorial Function

Definition:

- For a positive integer *n*, its factorial is the product of integers from 1 to n.
  - $5! = \underbrace{\overbrace{1 \text{ x } 2}^{2!} \text{ x } 3 \text{ x } 4}_{4!} \text{ x } 5 = 120 = 4! \text{ X } 5$

Recursive definition:

$$n! = \begin{cases} 1 & \text{if } n = 1 \\ n \times (n-1)! & \text{if } n > 1 \end{cases}$$

$n! = 4! = 4\,(3!)$

Iterative Program

"iterate"

```
def factorial(n):
    assert(n > 0)
    result = 1
    for i in range(1, n+1)
        result *= i
    return result
```

→ result = result * i

res = res + 1 => res += 1

~s 1 * 2 * ...

res = res * 2

res = res * 3

res = res * n

Recursive Program

```
def factorial(n):
    assert(n > 0)
    if n == 1
        return 1
    else:
        return n * factorial(n - 1)
```

Termination

- Must make sure that recursive program (or definition) terminates after finite number of steps
- Achieved by decreasing some parameter until it reaches the base case
  - line length: line of the length decreases by 1 with each recursive call, until it becomes 1
  - Factorial: n decreases by 1

Example of Infinite Recursion

n = 2

```
def infinite(n):
    if n == 1
        return 0
    return n * infinite(n+1)
```

if n == 50, stop

- In theory:
  - will never stop, parameters increase to infinity
- In practice:
  - will cause error message
    - “Stack overload” or “Recursion depth exceeded”

More Examples of Infinite Recursion

- No base case:

```
def factorial(n):
    return n * factorial(n - 1)
```

- Parameter do not change:

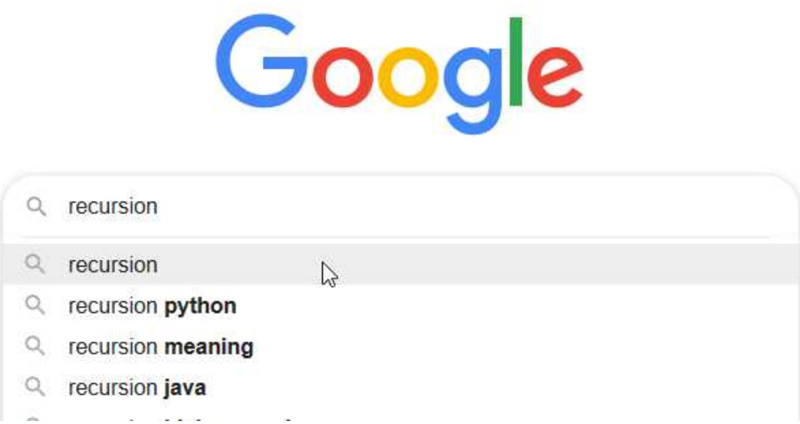
n = 1 , n = 2

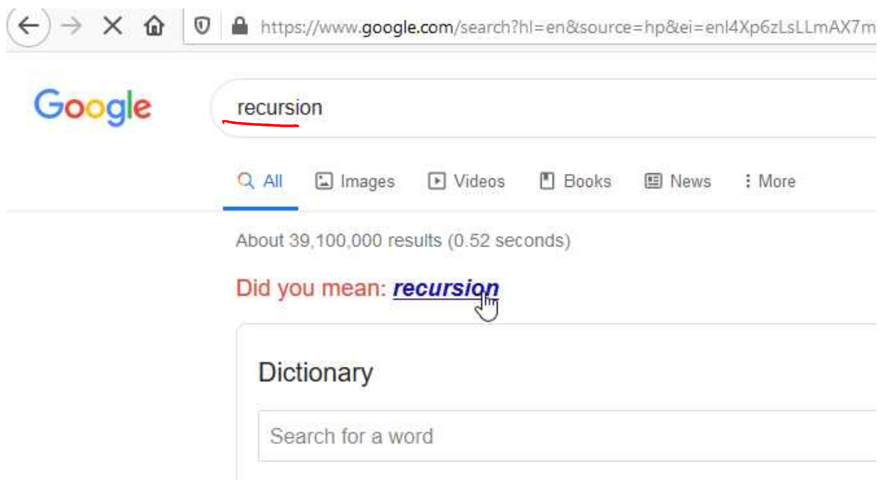
```
def infinite(n):
    if n == 1
        return 0
    return 1 + infinite(n)
```

More Examples (Not so serious :D)

- To understand recursion, one must first understand recursion

- Google is even in with the fun





- Recursion
- The Coin Problem
- Hanoi Towers

The Coin Problem

Problem

Prove that any amount starting from 8 **ewans** can be paid for using only 3 **ewans** and 5 **ewans.**

- $8 = 3+5$, $9=3+3+3$, $10=5+5$, $11=3+3+5$
  - looks promising, seems possible
- How to be sure it will always be possible?

Speculation

- *Speculation* – forming a theory without firm, solid concrete proof or evidence

- 8 can be definitely ~~be~~ done
- 11 is also possible by adding one 3 ewan coin
  - 11 = 8 + 3
  - same principle for 14, 17, 20, etc…
- Same for 9, keep on adding a 3 ewan coin
  - will give 12, 15, 18, 21, etc
- Checking 10 ewans, still keep adding 3 ewan coins
  - we'll get 13, 16, 19, 22, etc

Recursive Program

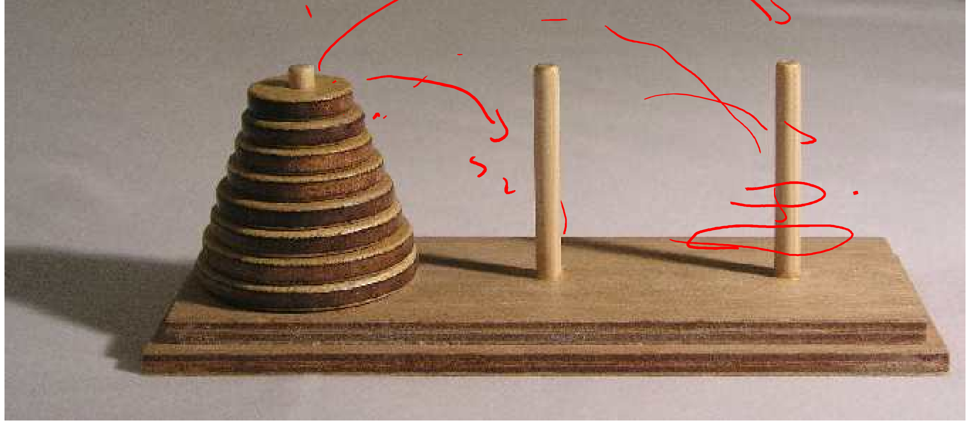
```
def change(amount):
    assert(amount >= 8)
    if amount == 8:
        return [3, 5]
    if amount == 9:
        return [3, 3, 3]
    if amount == 10:
        return [5, 5]

    coins = change(amount - 3)
    coins.append(3)
    return coins
```

- Recursion
- The Coin Problem
- Hanoi Towers

Hanoi Towers

Problem

There are 3 sticks with *n* discs sorted by size on one of the sticks. The goal is to move all *n* disks to another stick subject to 2 constraints: (1) move one disk at a time, and (2) don't put a larger disk over a smaller one.

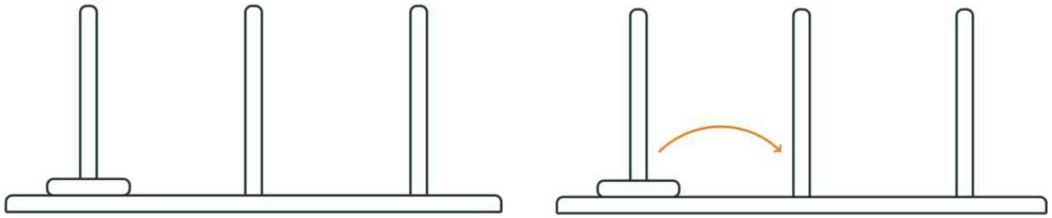


https://en.wikipedia.org/wiki/Tower_of_Hanoi#/media/File:Tower_of_Hanoi.jpeg

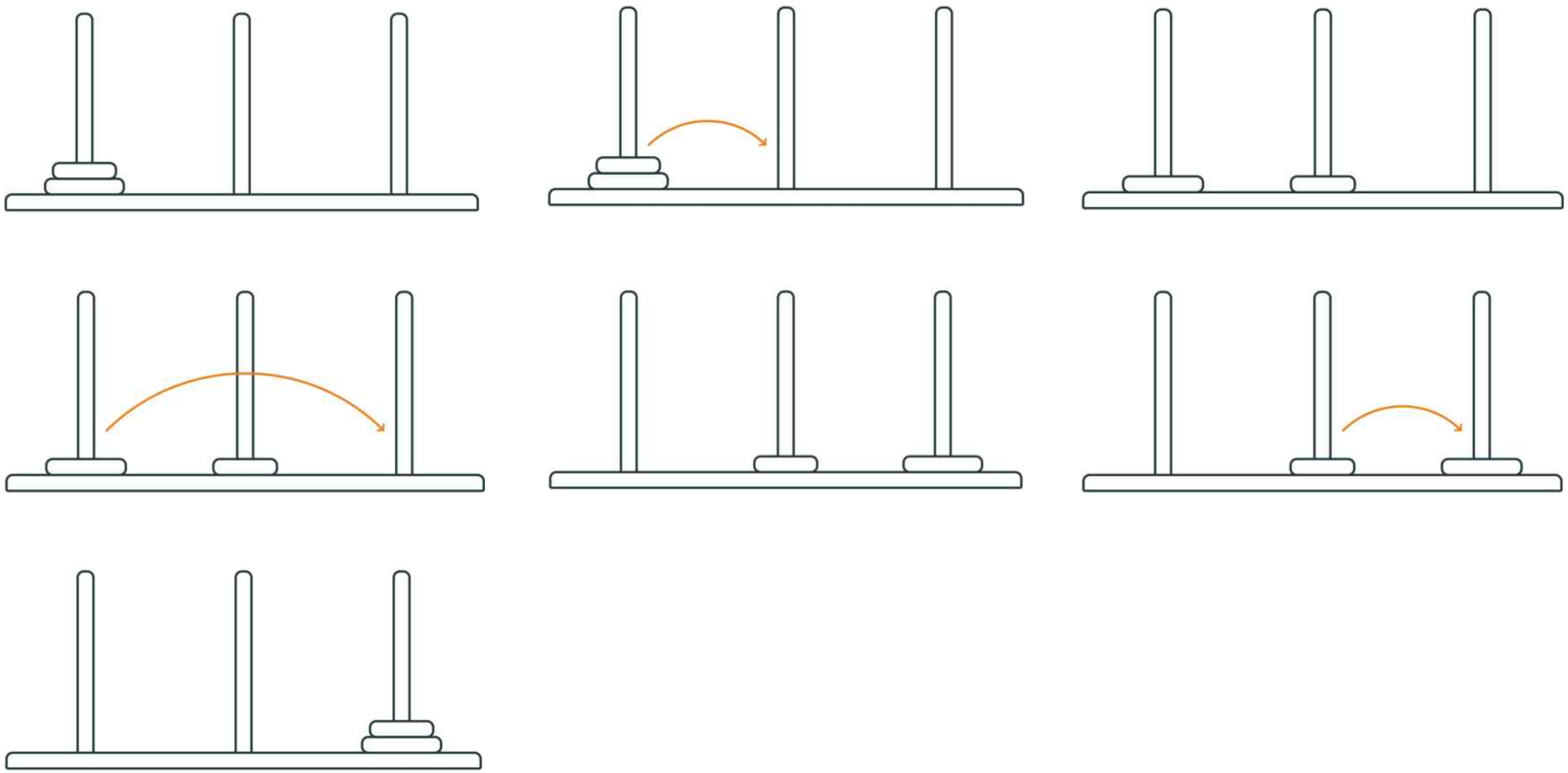
Can it be done?

- For what value of n is this possible?
  - For all!
- How can we be sure?
  - Design a recursive program that will solve the puzzle for every value of n

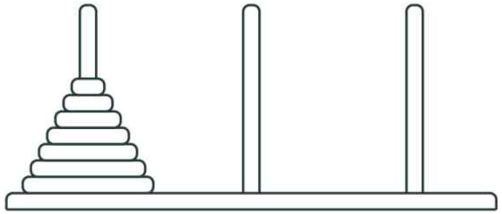
Simplest scenario, n=1 disk

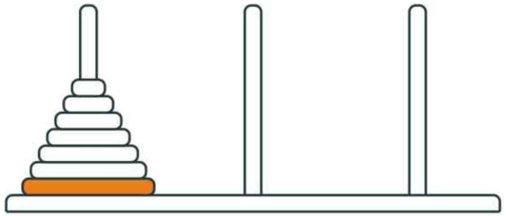


n=2 Disks Scenario



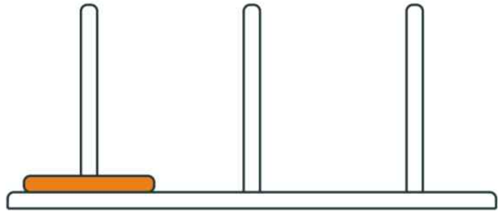
How about for *n* disks? Let's speculate





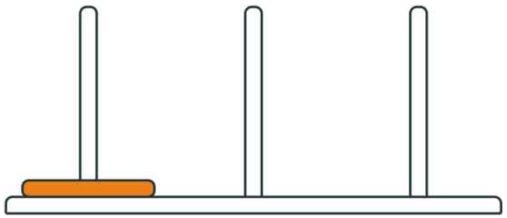
1

Consider largest disk



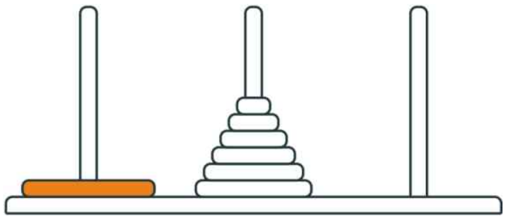
When can we move it?

C_1 → When there are no other disks on the main stick



Where can we move it?

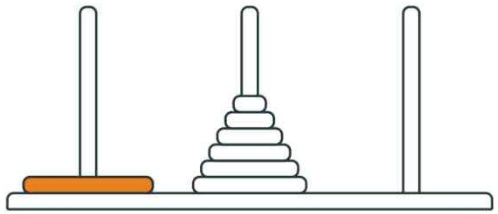
C_2 → Only to an empty stick



2

All other disks are on another disk

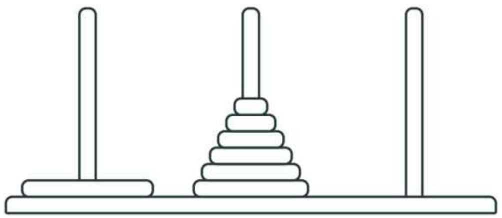
C_1 && C_2



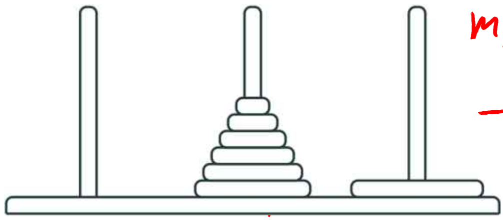
n-1

This is same problem, just n-1 disks!

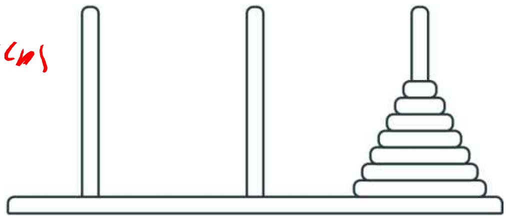
n-1 disks? Let's do it recursively



Move n-1 disks recursively



Move largest disk to free stick



Move n-1 disks recursively until done

Summary

- A solution has been proposed for all values of ***n***:
  - Base scenario: possible for *disks = 1*
    - Therefore, it is possible for *disks = 2*
    - Therefore, it is possible for *disks = 3*, … until *disks = n*
  - Or putting it other way, It is possible to solve *n*, solve first for *n-1*
    - Therefore, it is possible for *n – 1*
    - Therefore, it is possible for *n – 2*; … until n = 1
- recursion & induction
  - *recursion*: method of defining/implementing something
  - *induction*: mathematical method of proving something

Reminder

- Everybody, make sure that your name in ZOOM is in the following format:
  - Ex: 202054321 Juan Dela Cruz

Not changing your name to this format

* you **will** be marked Absent * → absent?

Mathematical Thinking – Recursion & Induction

MATHEMATICAL INDUCTION

Mathematical Induction

- Very powerful method of proving
- Take the case of falling dominos
  - Push 1st one, it'll fall & push 2nd one
  - 2nd one will fall & push 3rd one, etc...
  - So on until all dominos fall
- How to prove for *n* dominos
  - push $k^{th} \rightarrow k^{th}$ falls $\rightarrow$ *push* $k+1 \rightarrow k+1$ *falls* $\rightarrow \ldots \rightarrow n^{th}$ falls
- Lots of Comp Sci algo proved using induction

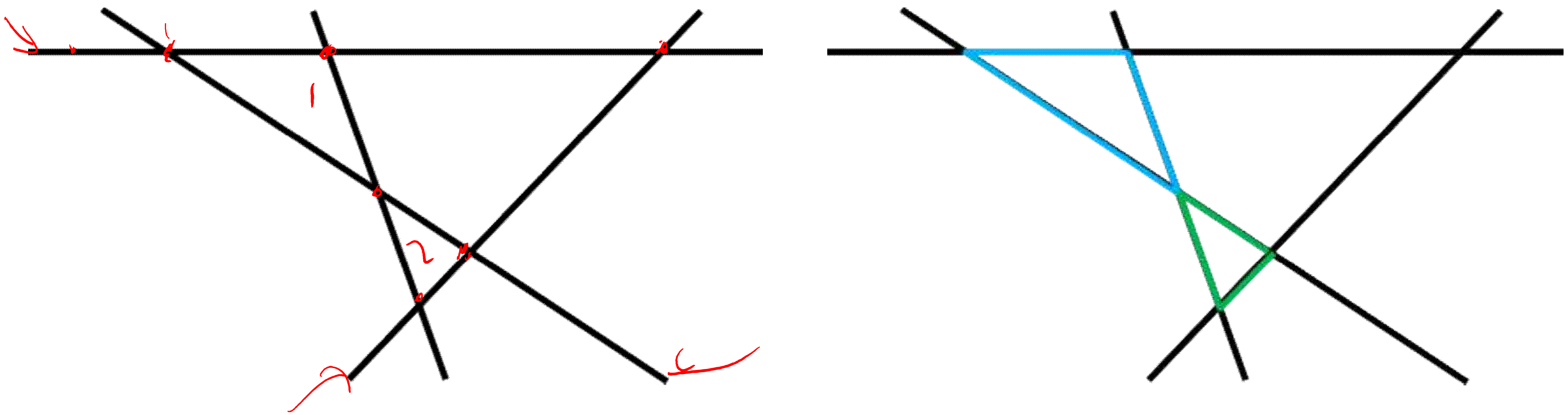
https://spiritualguidancefortheso ul.com/wp-content/uploads/2015/01/dominoes-04.jpg

- Lines and Triangles
- Connecting Points
- Sums of Numbers
- Bernoulli's Inequality
- Coins
- Cutting a Triangle
- Flawed Induction Proofs
- Alternating Sum

Lines & Triangles

Problem

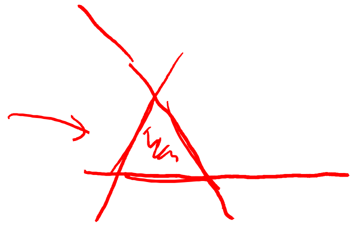
Several straight lines (at least three) cut a plane into pieces. Each line intersects with every other line, and all intersection points are different. **Prove that there is at least one triangular piece.**



Lines & Triangles

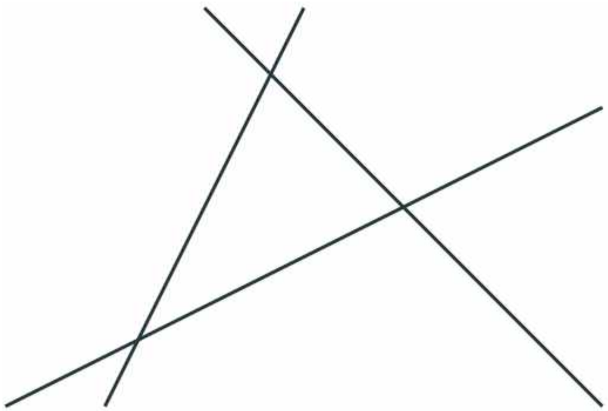
Proof Idea

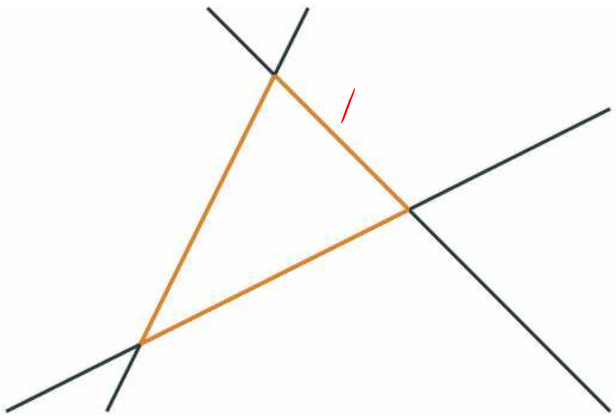
- A triangle appears as soon as there are 3 lines
- Each time we add more lines one at a time,
  - Either the same triangle remains
  - Or a new one is formed



Lines & Triangles

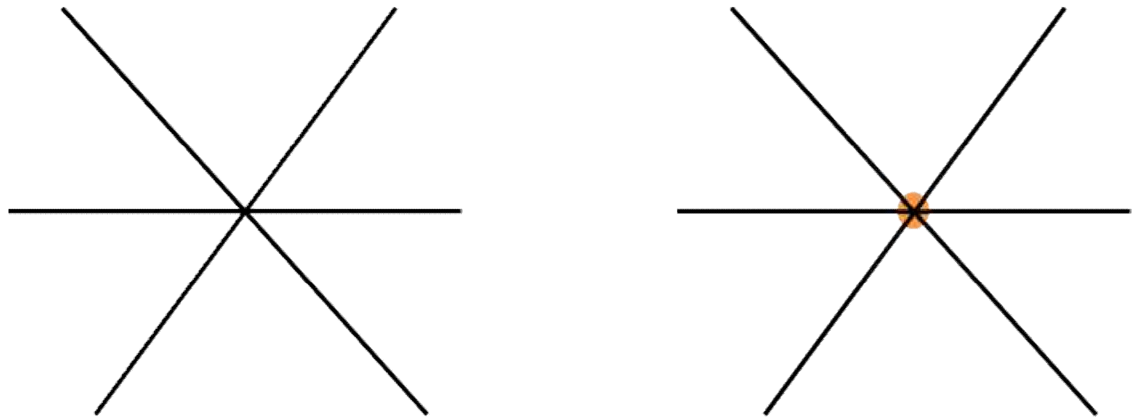
Three lines





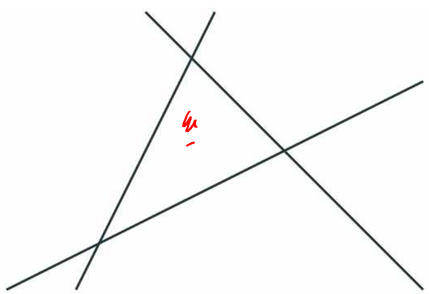
Lines & Triangles

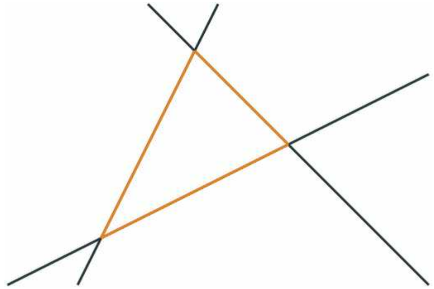
Three lines – Bad Case

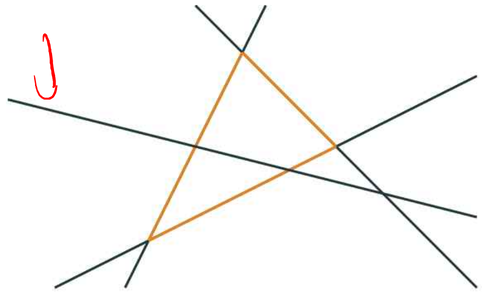


Lines & Triangles

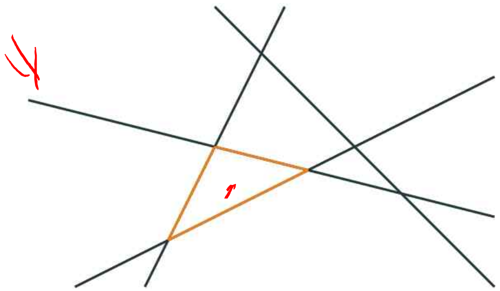
Three lines – Adding one more line



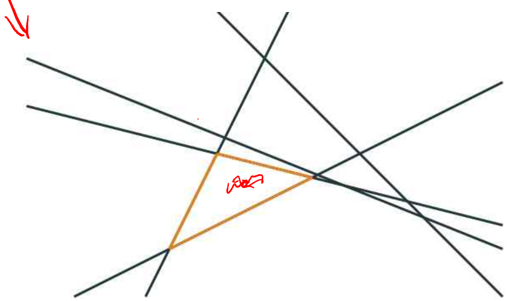




When new line intersects triangle…



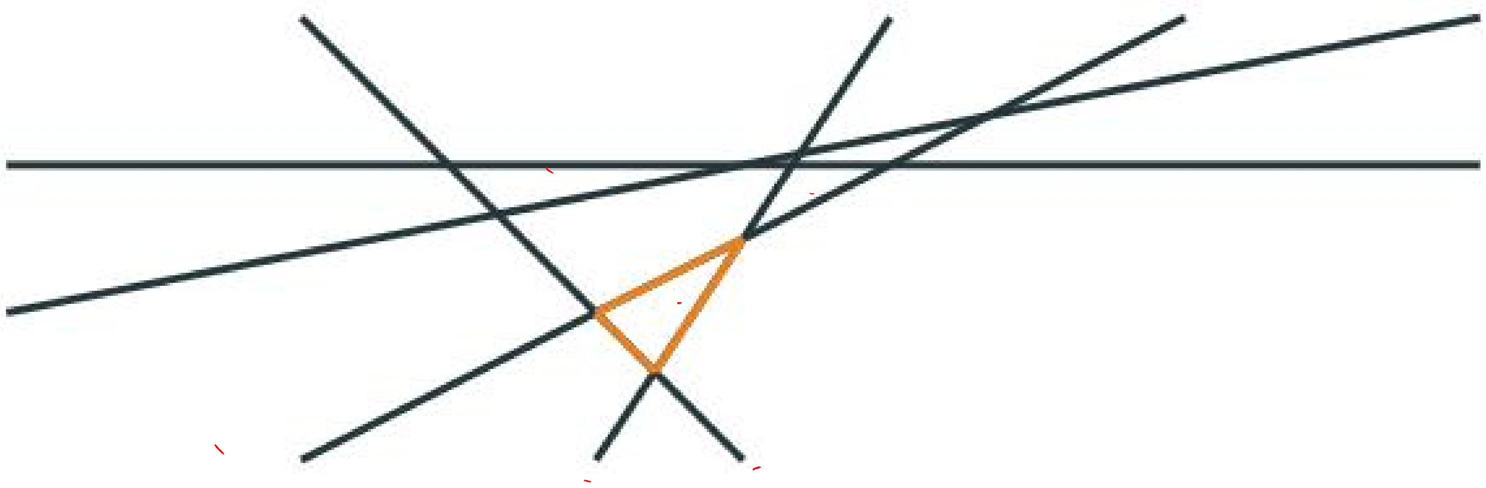
When new line intersects triangle, a new triangle appears



When new line doesn't touch existing triangle, the triangle remains intact

Lines & Triangles

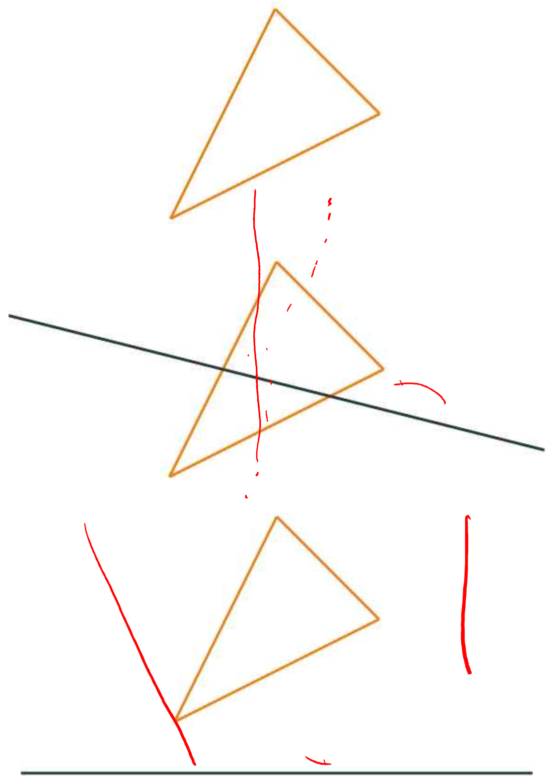
General Case



With a lot of lines, we might spot the triangle.
But with thousands of lines, we might not be able to see it.

Lines & Triangles

General Case – Adding a line



We assume lot of lines have already been added.
A triangle has already been formed

When new line intersects 2 sides of the triangle,
a new triangle appears

When new line doesn't touch existing triangle,
the triangle remains intact

Lines & Triangles

Theorem

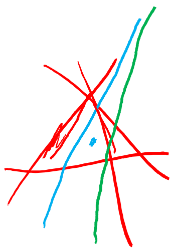
- For any $n \geq 3$ and n number of straight lines on a plane, if every two lines intersect, and all the intersection points are different, there is a triangular piece among the pieces into which these lines cut the plane.

Lines & Triangles

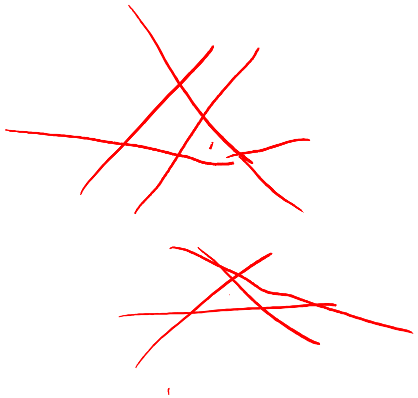
Proof Structure

Number of lines

$$3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow n$$



$$k \rightarrow K+1$$



Mathematical Induction

- Prove induction base: $n = 3$, three lines
- Prove that if theorem is true for $n = 3$, then it is true for $n = 4$
- Prove that if theorem is true for $n = 4$, then it is true for $n = 5$
- …
- Prove induction step: from n to $n + 1$, adding one more line in the general case
- …
- Profit! (Proved the whole statement)

But wait, we can shorten this, can you see how?

Mathematical Induction

Prove only induction base is $n = 3$ and prove induction step from n to $n+1$ in the general case

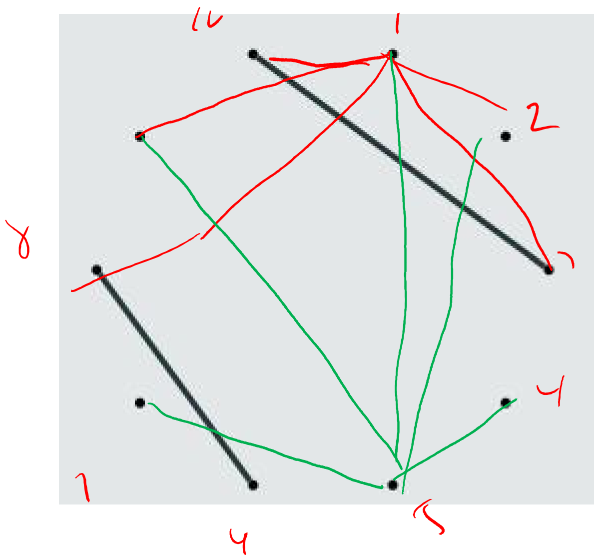
- Prove induction base: $n = 3$, three lines
- Prove induction step: from n to $n + 1$, adding one more line in the general case
- Profit!

- Lines and Triangles
- Connecting Points
- Sums of Numbers
- Bernoulli's Inequality
- Coins
- Cutting a Triangle
- Flawed Induction Proofs
- Alternating Sum

Connecting Points

Problem

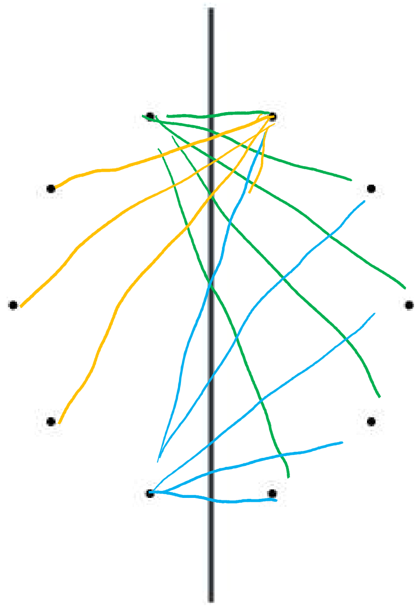
Connect some of these 10 points with segments, such that every point is connected with 5 other points.



Connecting Points

Solution

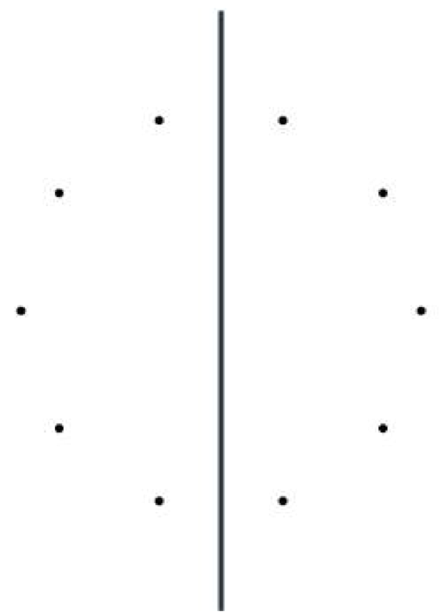
Separate the points into the left half and the right half. Each half has 5 points.



Connecting Points

Solution

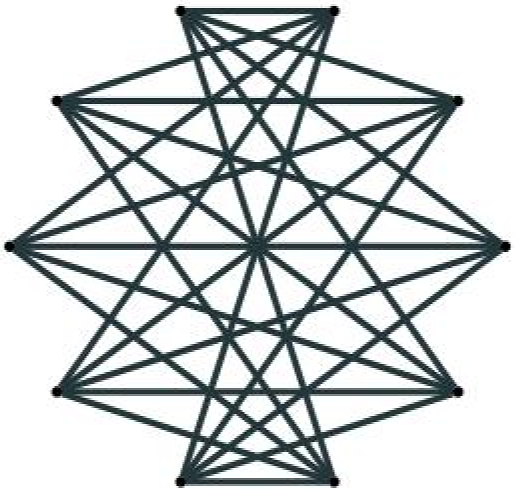
Connect each half from the left half to each point of the right half.



Connecting Points

Solution

Connect each half from the left half to each point of the right half.



Connecting Points

Problem

Now we have 9 points. Can we connect some of them with segments so that each point is connected with 5 other points?

10 pts → 5

9 pts → 5

Connecting Points

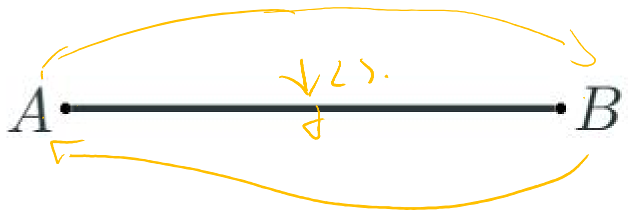
Even and Odd

- The numbers 0, 2, 4, 6, 8, … are called even
  - Even numbers are divisible by 2
- Numbers 1, 3, 5, 7, 9, … are called odd
  - Odd numbers are not divisible by 2

Connecting Points

Neighbors

- Let us call point B the neighbor of point A if points A & B are connected with a segment.
- If point B is a neighbor of A, then A is also a neighbor of B.



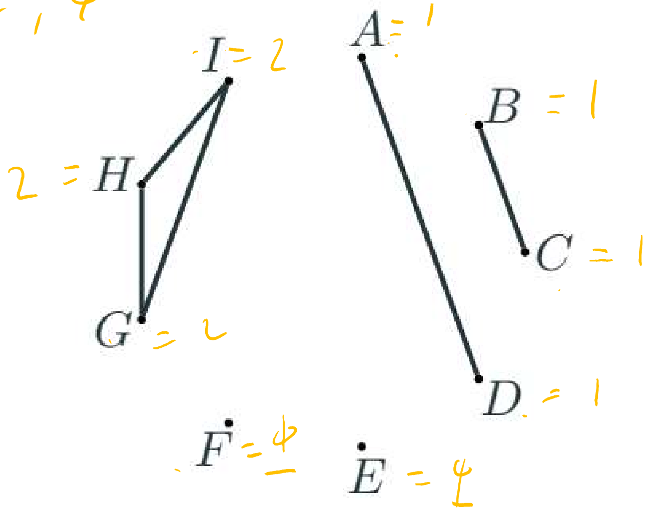
Connecting Points

Even and Odd

- Let us call a point **even** if it has even number of neighbors, otherwise we call it odd.
- For the given example, points A, B, C & D are odd while all the other points are even.

odd = 2

even = 2, 4



e = 5

o = 4

Connecting Points

Theorem

- The number of odd points is always even, regardless of how many points & segments are there and which pairs are connected by segments.

\# $n_{odd\,pts}$ = even ; n_{pts} ; $P_{segments}$

Connecting Points

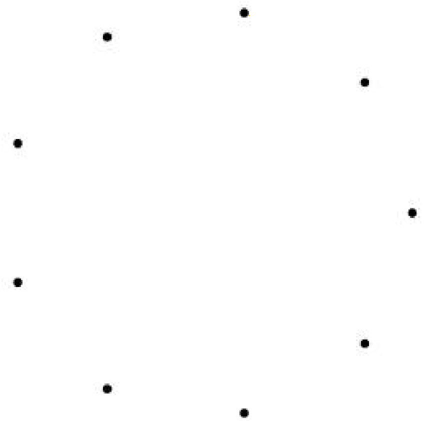
Proof Idea

- When there are no segments, there are no odd points (recall 0 is even), so the number of odd points is indeed even.
- When we add segments one by one, the number of odd points either doesn't change, increases by 2 or decreases by 2. hence the number of odd points stays even.

Connecting Points

Easy Case

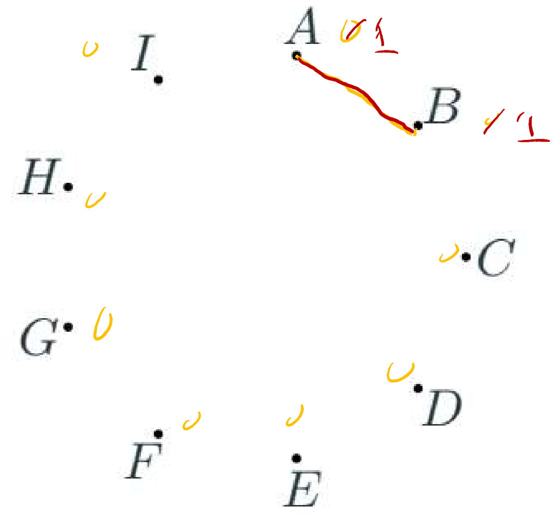
- When there are no segments, each points has 0 neighbors, hence there are no odd points. The number of odd points is 0, which is even, hence there is indeed an even number of points.



Connecting Points

Adding a Segment

- Segment AB adds two odd points A & B.

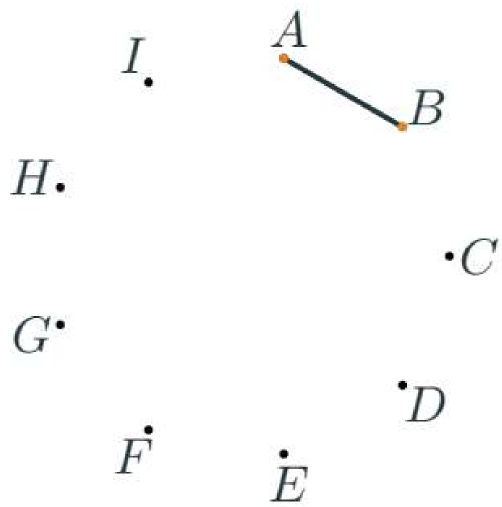


No of odd points: 0

Connecting Points

Adding a Segment

- Segment AB adds two odd points A & B.

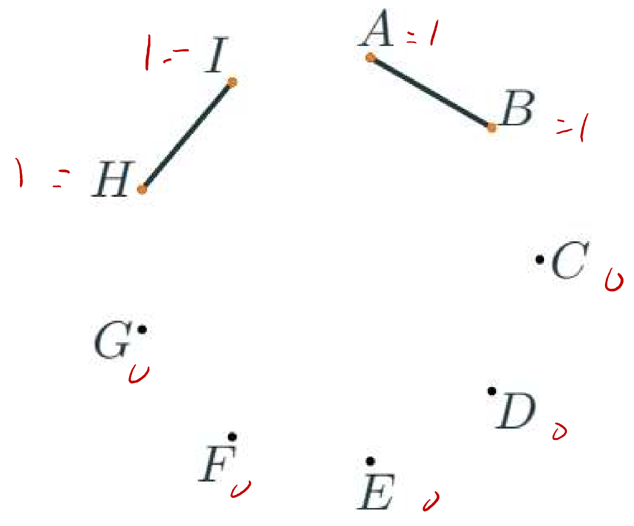


No of odd points: 2

Connecting Points

Adding a Segment

- Segment *HI* adds two odd points *H* & *I*.

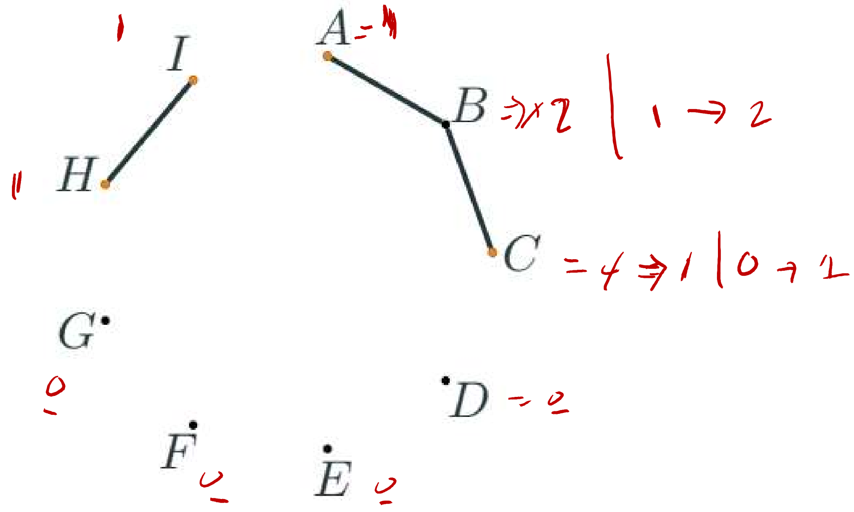


No of odd points: 4

Connecting Points

Adding a Segment

- Segment BC makes B even and C odd.

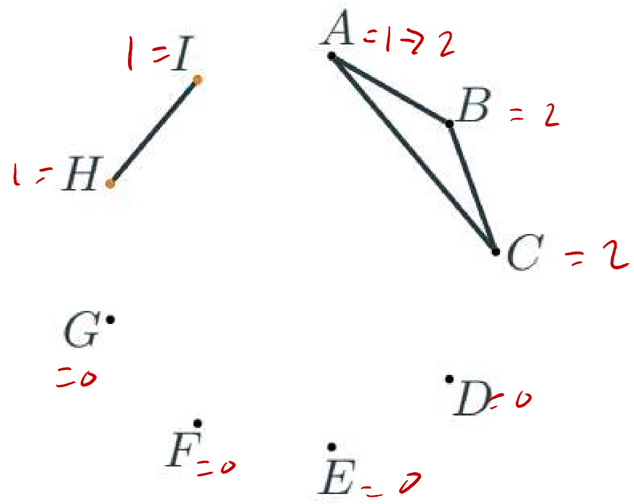


No of odd points: 4

Connecting Points

Adding a Segment

- Segment AC makes A and C even.

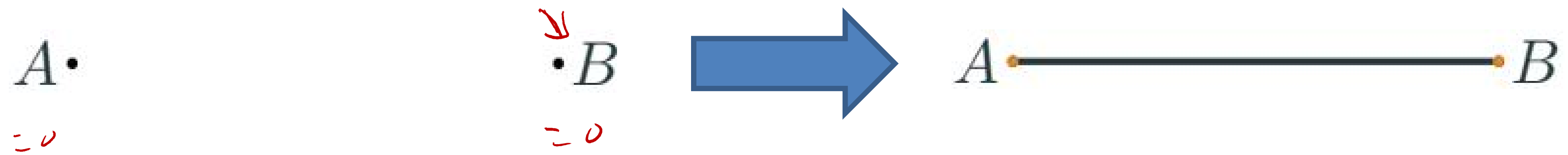


No of odd points: 2

Connecting Points

Adding a Segment in General

- If A & B are even, segment AB makes them both odd and adds 2 odd points.

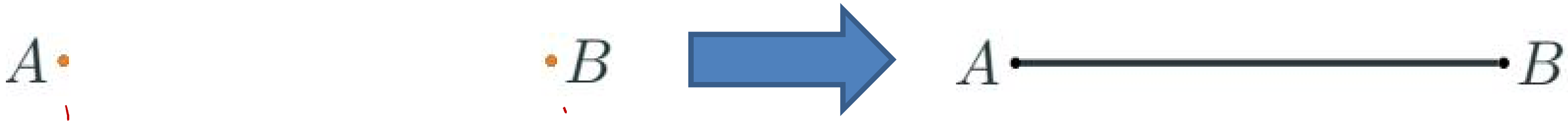


*Note:
- A●: represents even point
- A●: represents odd point

Connecting Points

Adding a Segment in General

- If A & B are odd, segment AB makes them both even and removes 2 odd points.

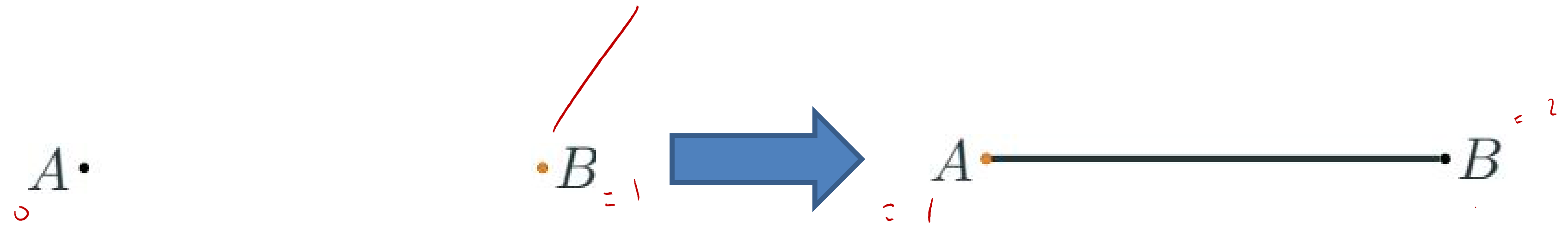


*Note:
- A●: represents even point
- A●: represents odd point

Connecting Points

Adding a Segment in General

- If A is even & B is odd, segment AB swaps their status, keeping number of odd points the same.



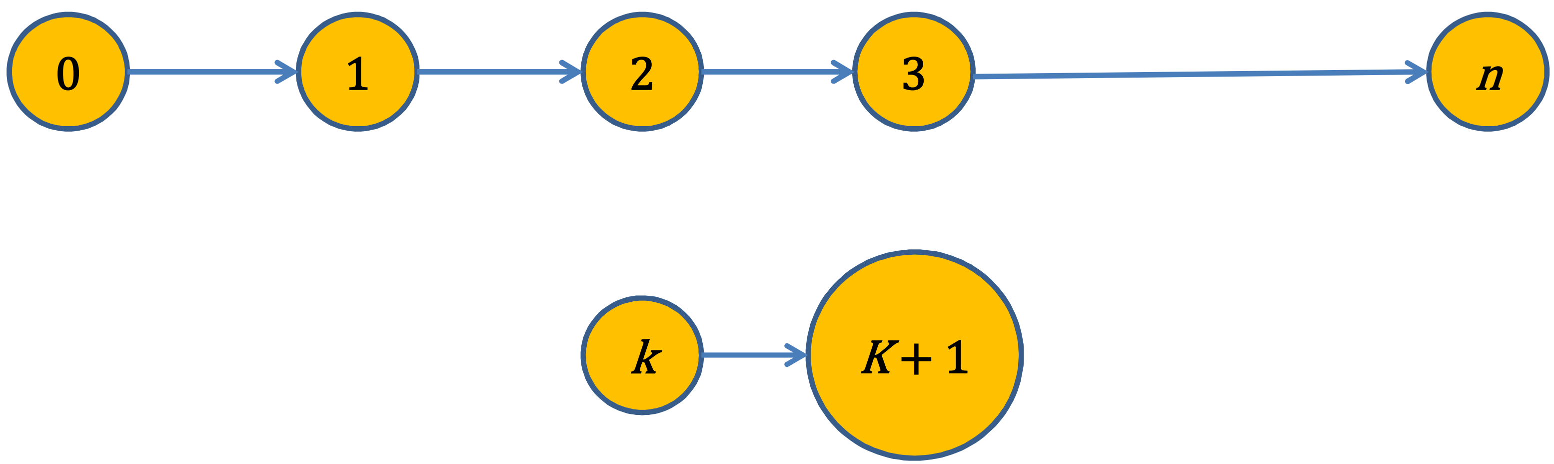
*Note:

- A●: represents even point
- A●: represents odd point

Connecting Points

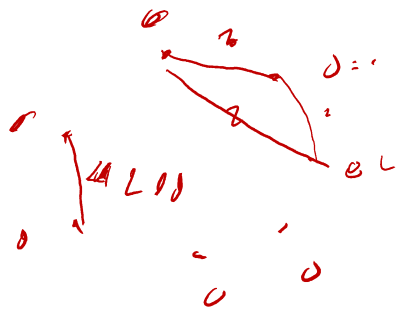
Proof Structure

Number of segments



Mathematical Induction

- Prove induction base: $n = 0$, no segments
- Prove that if theorem is true for $n = 0$, then it is true for $n = 1$
- Prove that if theorem is true for $n = 1$, then it is true for $n = 2$
- ...
- Prove induction step: from n to $n + 1$, adding one more line in the general case
- ...
- Profit!



Mathematical Induction

- Prove induction base: $n = 0$, no segments
- Prove induction step: from n to $n + 1$, adding one more line in the general case
- ...
- Profit!

Thank you.