Compsoc x Mathsoc: Intro to Mathematical Programming

Make sure to check out both societies if you enjoy

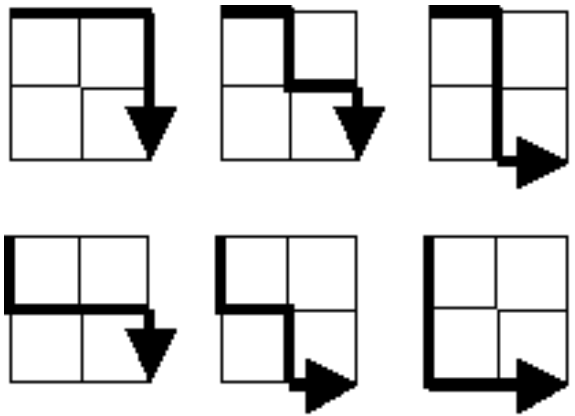
What can I with this information?

- Familiarise yourself with coding and mathematics
- Consider adding a project in animation or simulation to your portfolio
- Speed up your algorithms by employing mathematical simplification
- Absolutely nothing

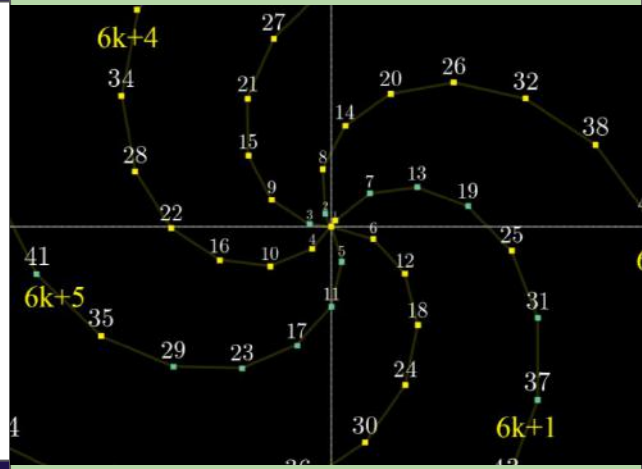
- Data Scientist
  - Citi Bank
  - Expedia Group
- Quant Trader
  - Jane Street
  - QRT
- Software Engineer
  - Motorola
  - Epic Games
- Robotics Engineer
  - Caterpillar
  - Phd with Engineering Department

Triangle sums

Lattice Paths

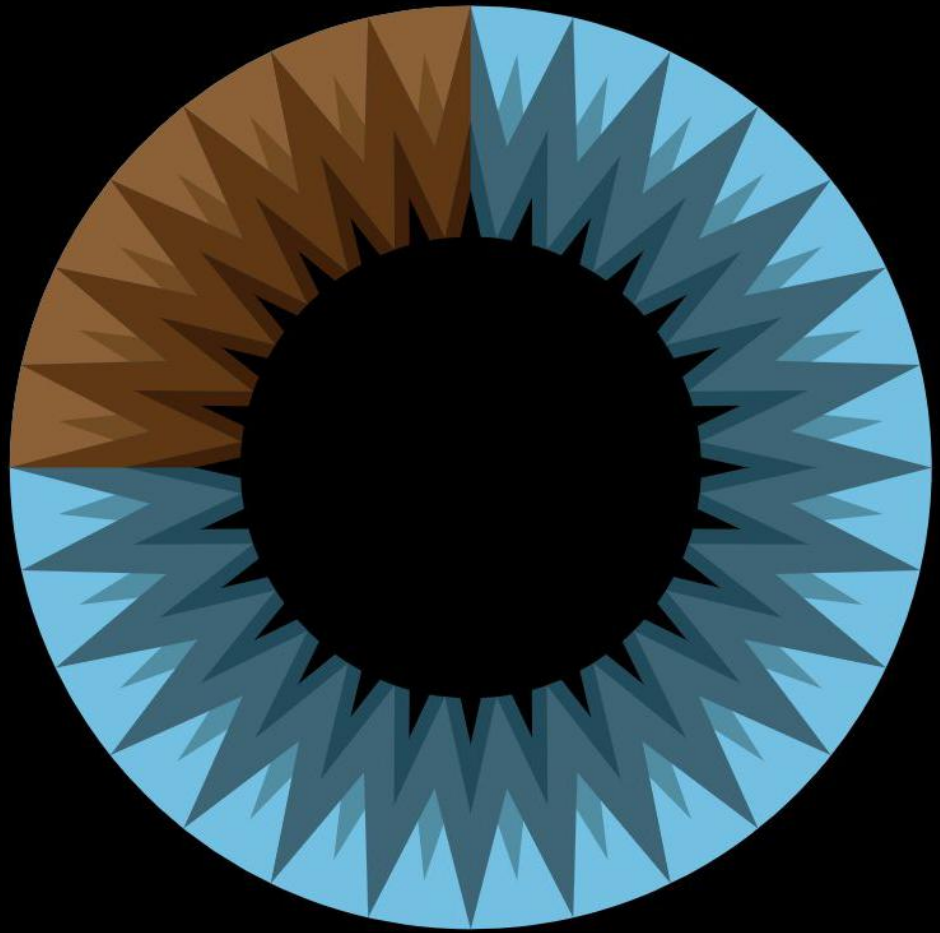


Many Factors



Coding

Maths




Manim

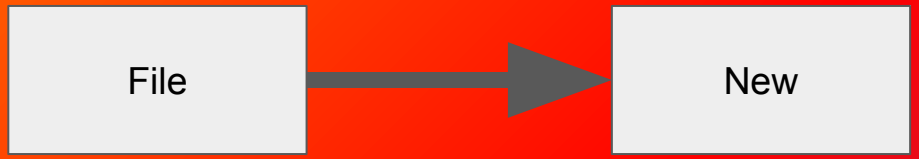
https://www.manim.community

https://www.manim.community

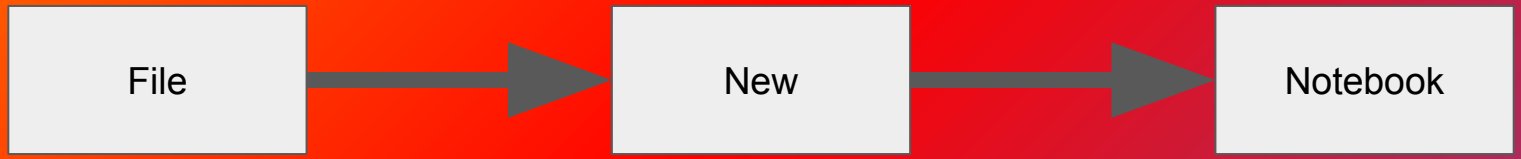
https://try.manim.community

Once you’re trying Manim

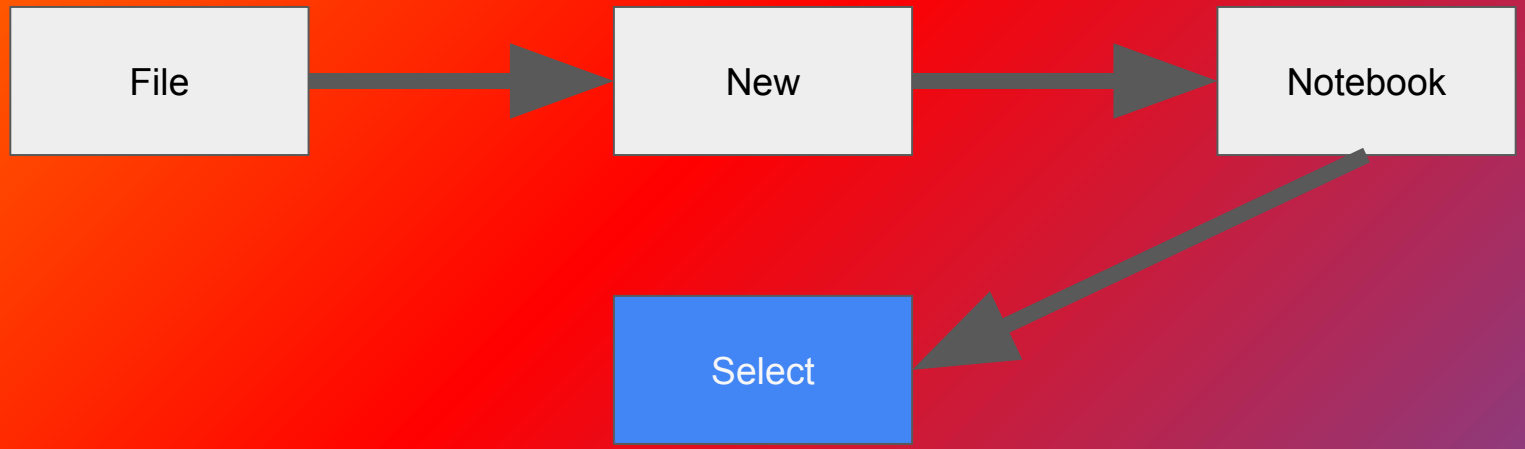
Once you’re trying Manim



Once you’re trying Manim



Once you’re trying Manim



Your first program

```
from manim import *

config.media_width = "75%"
config.verbosity = "WARNING"
```

Your first program

```
from manim import *

config.media_width = "75%"
config.verbosity = "WARNING"
```

alt + ENTER

Your first program

```
from manim import *

config.media_width = "75%"
config.verbosity = "WARNING"
```

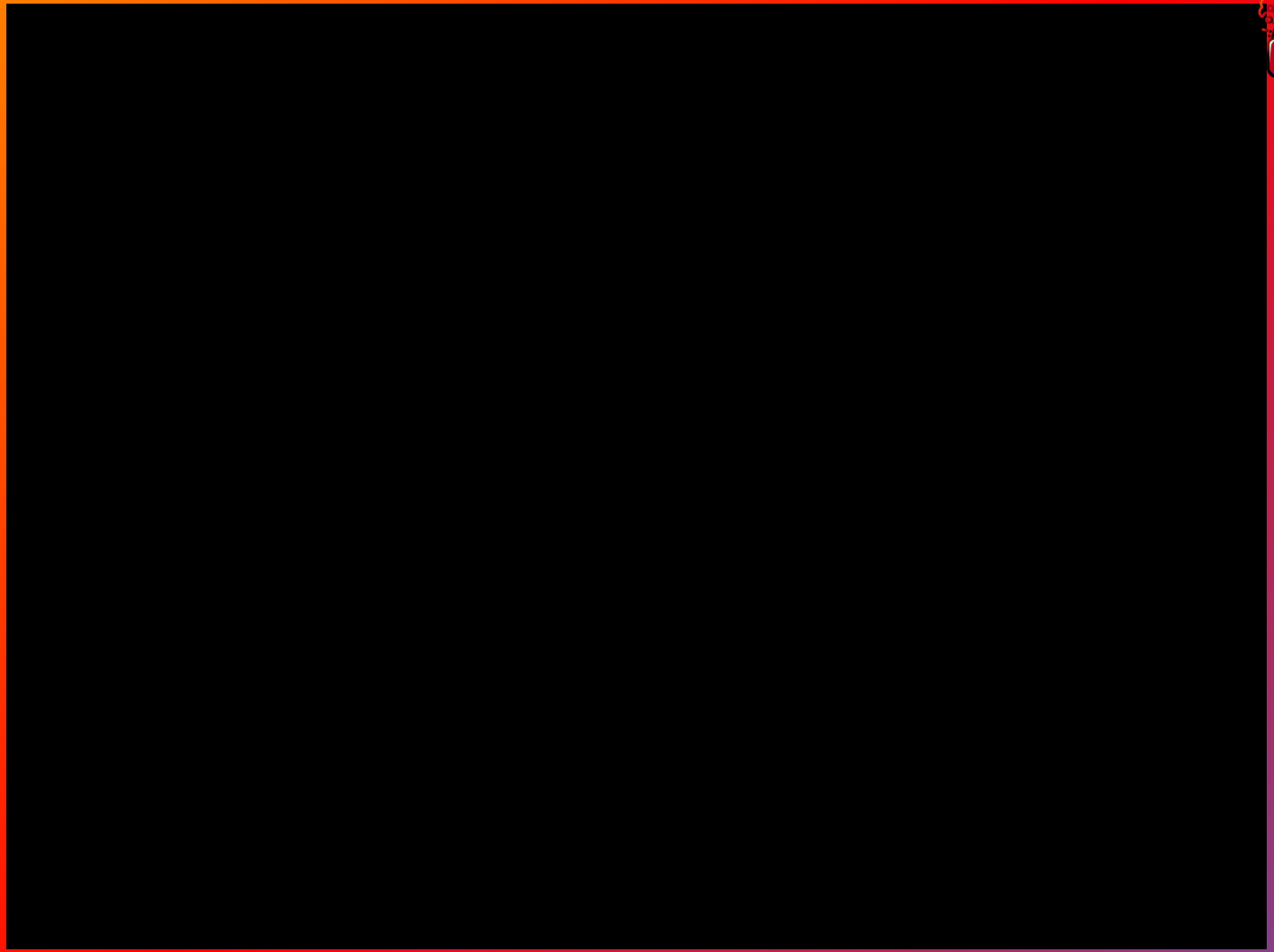
```
%%manim -qm CircleToSquare

class CircleToSquare(Scene):
    def construct(self):
        blue_circle = Circle(color=BLUE, fill_opacity=0.5)
        green_square = Square(color=GREEN, fill_opacity=0.8)
        self.play(Create(blue_circle))
        self.wait()

        self.play(Transform(blue_circle, green_square))
        self.wait()
```

Your first program

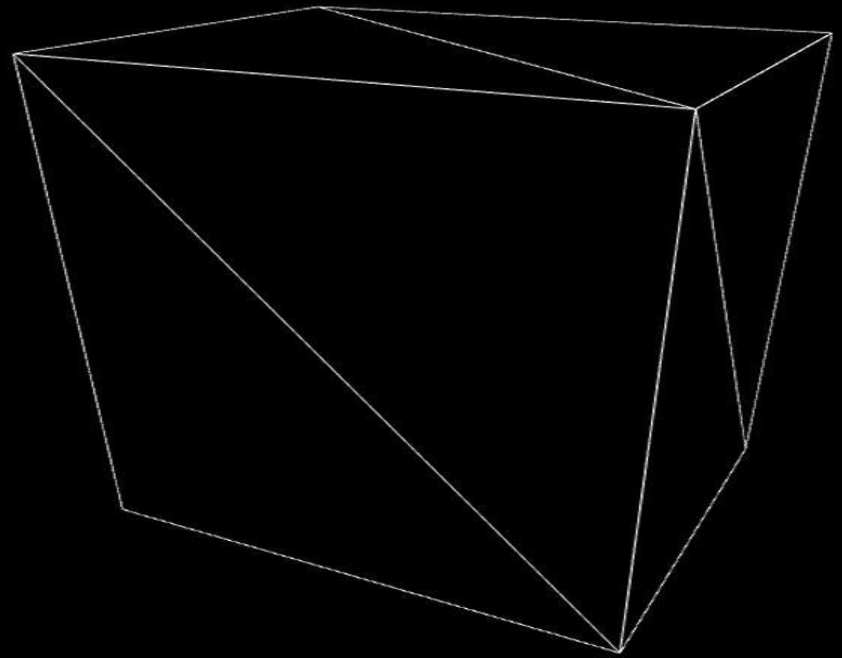
```
from manim import *

config.media_width = "75%"
config.verbosity = "WARNING"
```

```
%%manim -qm CircleToSquare

class CircleToSquare(Scene):
    def construct(self):
        blue_circle = Circle(color=BLUE, fill_opacity=0.5)
        green_square = Square(color=GREEN, fill_opacity=0.8)
        self.play(Create(blue_circle))
        self.wait()

        self.play(Transform(blue_circle, green_square))
        self.wait()
```

ctrl + ENTER



Worked Problem: Lattice Paths

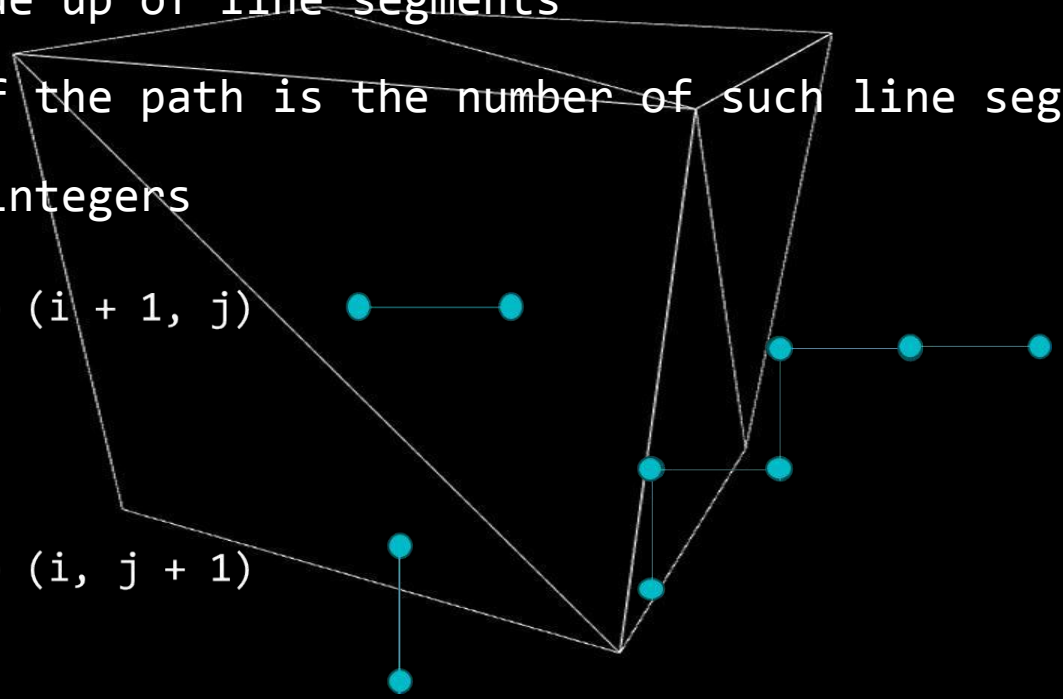


Lattice Paths

- a “curve” made up of line segments
- the length of the path is the number of such line segments
- i and j are integers

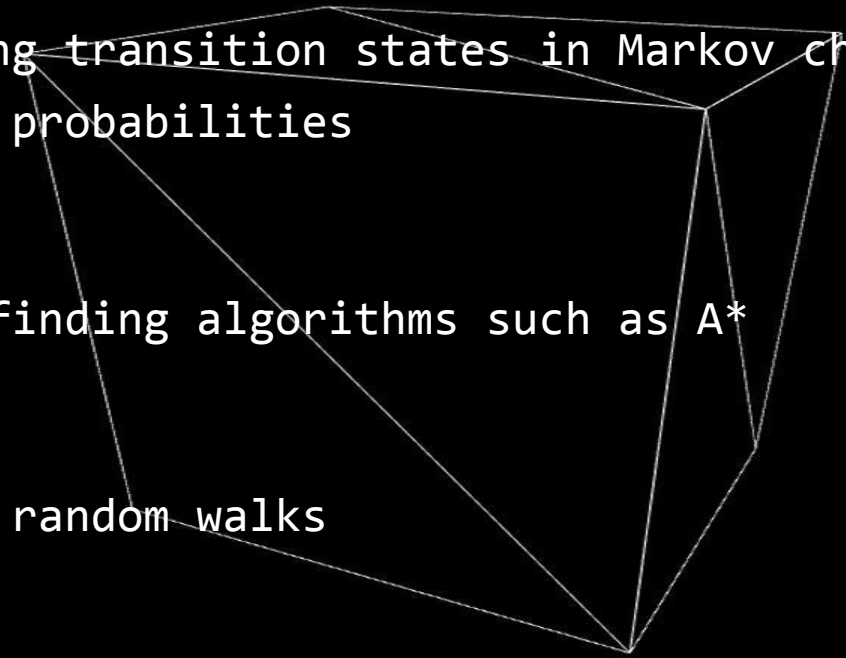
(i, j) to (i + 1, j)

(i, j) to (i, j + 1)



Applications

- visualising transition states in Markov chains and computing probabilities



- some pathfinding algorithms such as A*

- modelling random walks

LIVE DEMONSTRATION

Motivating Example:

A school play requires a ten-dollar donation per person

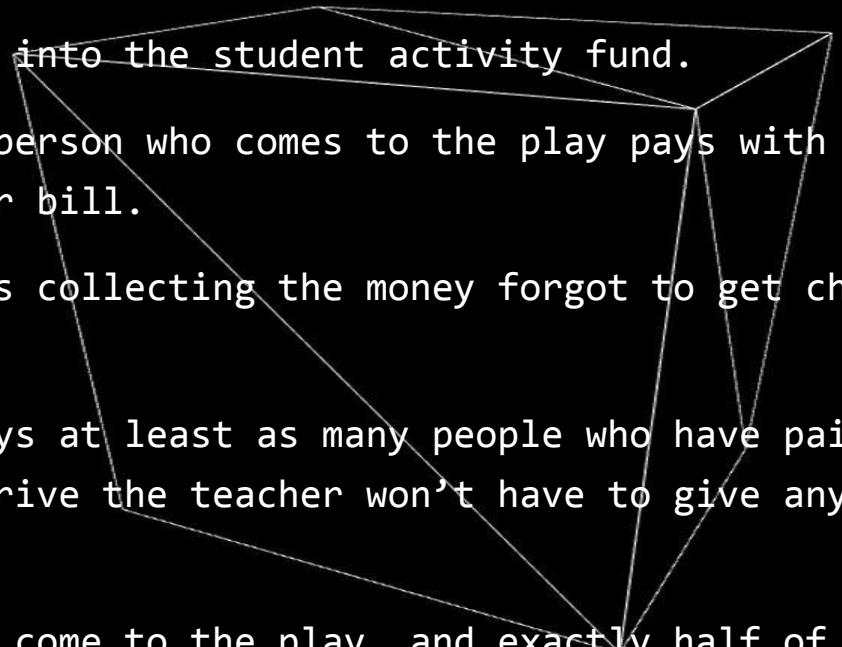
The donation goes into the student activity fund.

Assume that each person who comes to the play pays with a ten-dollar bill or a twenty-dollar bill.

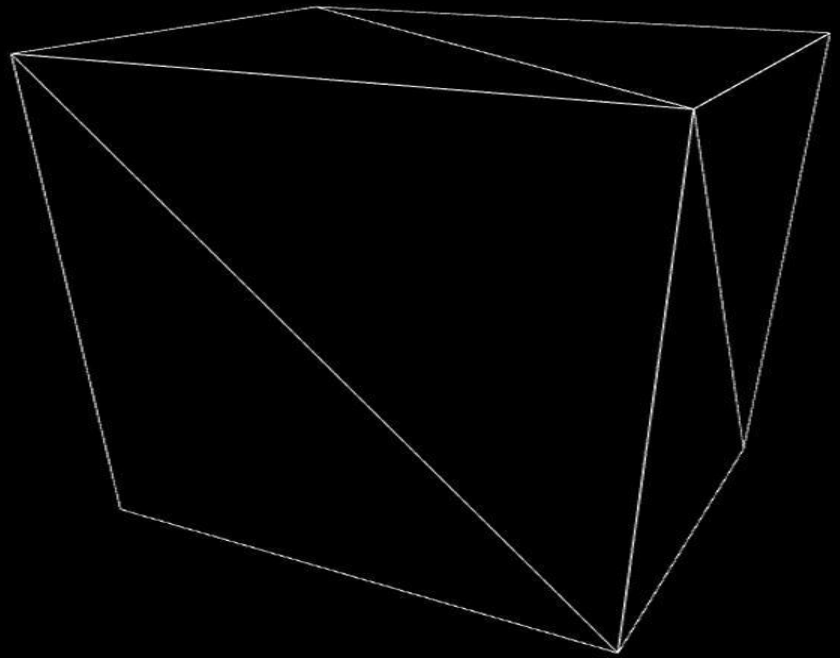
The teacher who is collecting the money forgot to get change before the event.

If there are always at least as many people who have paid with a ten as a twenty as they arrive the teacher won’t have to give anyone an IOU for change.

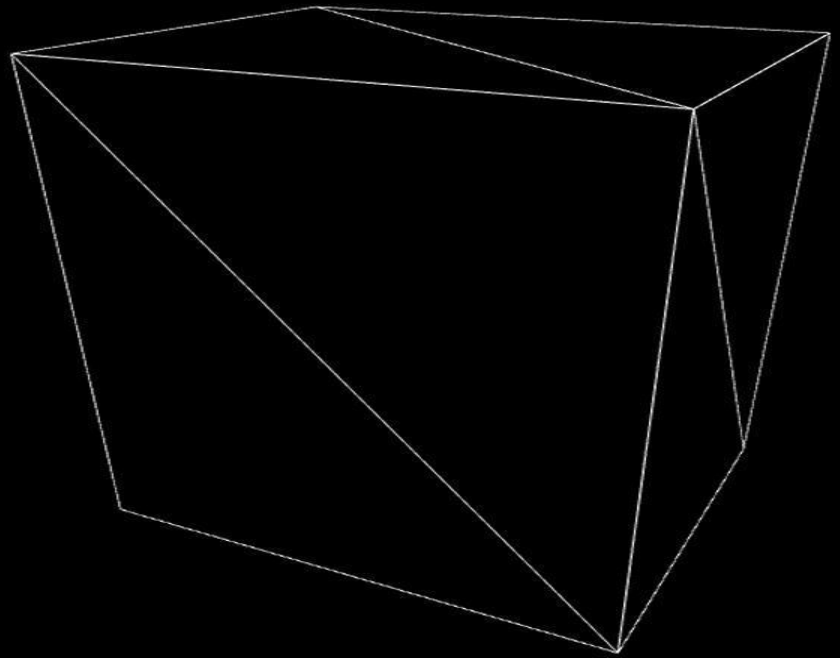
Suppose 2n people come to the play, and exactly half of them pay with ten-dollar bills.



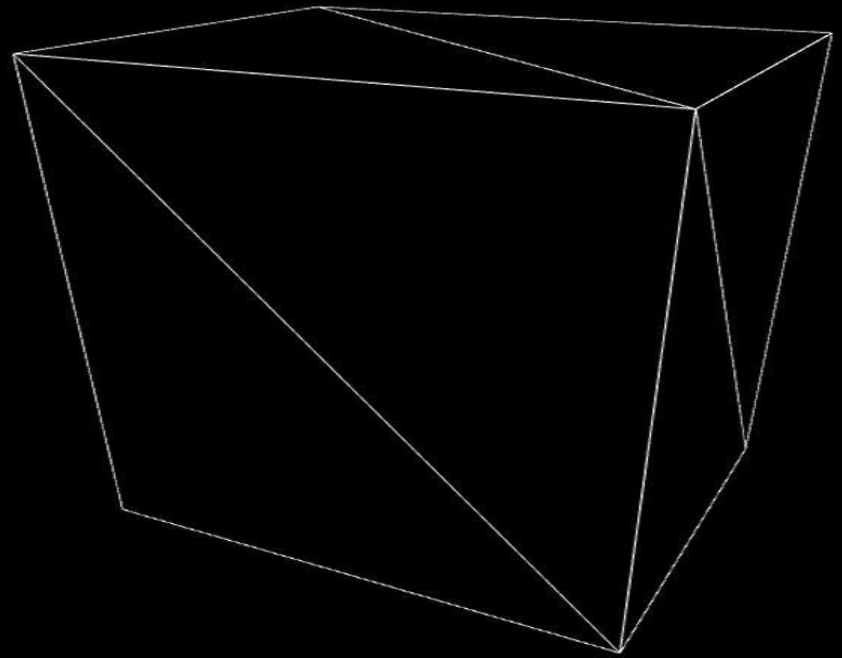
Describe a bijection between the set of sequences of tens and twenties people give the teacher and the set of lattice paths from (0, 0) to (n, n).



Describe a bijection between the set of sequences of tens and twenties that people give the teacher and the set of diagonal lattice paths between (0, 0) and (2n, 0).



In each of the previous parts, what is the geometric interpretation of a sequence that does not require the teacher to give any IOUs?



- HackNotts & DurHack
- Intro to BigTech & Autonomous Driving (ft. Nvidia)

- Logic Lounge
- Integration Bee

Triangle sums

By starting at the top of the triangle below and moving to adjacent numbers on the row below, the maximum total from top to bottom is 23.

3
7 4
2 **4** 6
8 5 **9** 3

That is, 3 + 7 + 4 + 9 = 23.

Find the maximum total from top to bottom in **triangle.txt** (right click and 'Save Link/Target As...'), a 15K text file containing a triangle with one-hundred rows.

Project Euler Problem 67 : https://projecteuler.net/problem=67

Many factors

The number of divisors of 120 is 16.
In fact 120 is the smallest number having 16 divisors.

Find the smallest number with 2^{500500} divisors.
Give your answer modulo 500500507.

Project Euler Problem 500 : https://projecteuler.net/problem=500

Clues:

1. Write out the factors of 120, what do you notice?

2.How can you count the factors of a number, knowing its prime factors?

3. The numbers here are clearly too big, how can you only deal with small numbers?

Triangle code

```
#reading file into cool formats
f = open("0067_triangle.txt", "r")
triangl = f.read().split("\n")
triangle = [x.split(" ") for x in triangl]

#changing all entries to integers
for i in range(len(triangle)-1):
    for j in range(len(triangle[i])):
        triangle[i][j] = int(triangle[i][j])


#print(triangl[3])
#print(triangle[3])
#print(triangle[3][1])
#print(type(triangle[0][0]))
```

```
#starting at the penultimate row of the triangle, and iterating
backwards until index 0
for row in range(len(triangle)-3, -1, -1):
    for index in range(len(triangle[row])):

      #adding the maximum of the entries below, to see the max
sum passing through this point in the triangle
        triangle[row][index] = triangle[row][index] +
max(triangle[row+1][index], triangle[row+1][index+1])

#the maximum sum overall
print(triangle[0][0])
```

```python
import math
primes = []
primes2 =[]

#checking ig a number is prime (not very interesting)
def primecheck(num):
    for x in range(2, int(math.sqrt(num)) + 1):
        if num % x == 0:
            return False
    return True

#generating a list of prime numbers
a = 0
for k in range(1, 7376508):
    if primecheck(k) == True:
        primes.append(k)
        a += 1
        print(k, "making primes")


# actually awful list usage but this is old code im so sorry
primes.pop(0)
```

```python
#expanding list to include numbers of the form p^(2^n) (will also multiply number of factors by 2)
for t in primes:
    if t**2 < 7376508:
        primes.append(t** 2)
        print(t**2, "making powers" )

#preparing to select the 500500 smallest elements of the list
primes.sort()


#multiplying them all mod 500500
product = 1
for s in range(0, 500500):
    product *= primes[s]
    product = product %  500500507
    print(product, s)    #i like watching numbers tick up
```

Factors code (takes a bit longer to run)