Combinei Programação Com Matemática e ganhei R$3.000 em menos de 24 horas

Status	Roterizado
Data de publicação	
Observações	

Você já parou pra pensar que algumas coisas são pra ser?

Era uma sexta feira e eu estava voltando do almoço para o trabalho, rezando pelo final de semana.

Abri meu e-mail e li a seguinte frase:

Eduardo, Ta valendo! Será que consegue faturar o grande prêmio?

Num primeiro pensamento, esse aquele típico clickbait que você acha que é só alguém querendo te oferecer um produto, mas pra minha sorte. Não era isso.

Era um desafio. Não só um desafio, um desafio de programação com matemática, algo que eu e a equipe que participo, Filhos do Python, temos um interesse por resolver

Como um dos integrantes dessa equipe trabalha comigo, O guilherme, chamei ele e lancei o desafio, que dizia o seguinte:

> **O octaedro regular é um poliedro formado por 12 arestas, 6 vértices e 8 faces.**
>
> **Dado um octaedro regular, considere o conjunto formado por seus vértices e os centros de suas faces.**
>
> **Quantos tetraedros distintos é possível construir com vértices nesse conjunto?**

A princípio, qualquer um que leia isso diz HÃN? IMPOSSÍVEL! Deixamos de lado e voltamos pro trabalho

No mesmo dia, perto da 11 da noite, aparece pra mim a mensagem, “

“Ou, tava pensando naquele código que você mostrou hoje cedo. Montei aqui e parece estar tudo certo, mas a resposta tá errada. Acho que isso é um trabalho para o Filhos do Python ”

“O Desafio estava lançado. Mas como resolver isso? Qual era o Problema por trás.

Para começo de tudo:

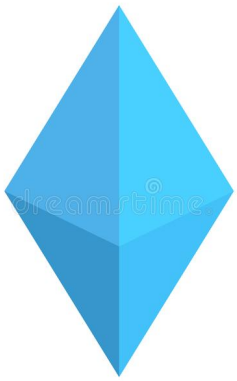
Analisar o racicínio.

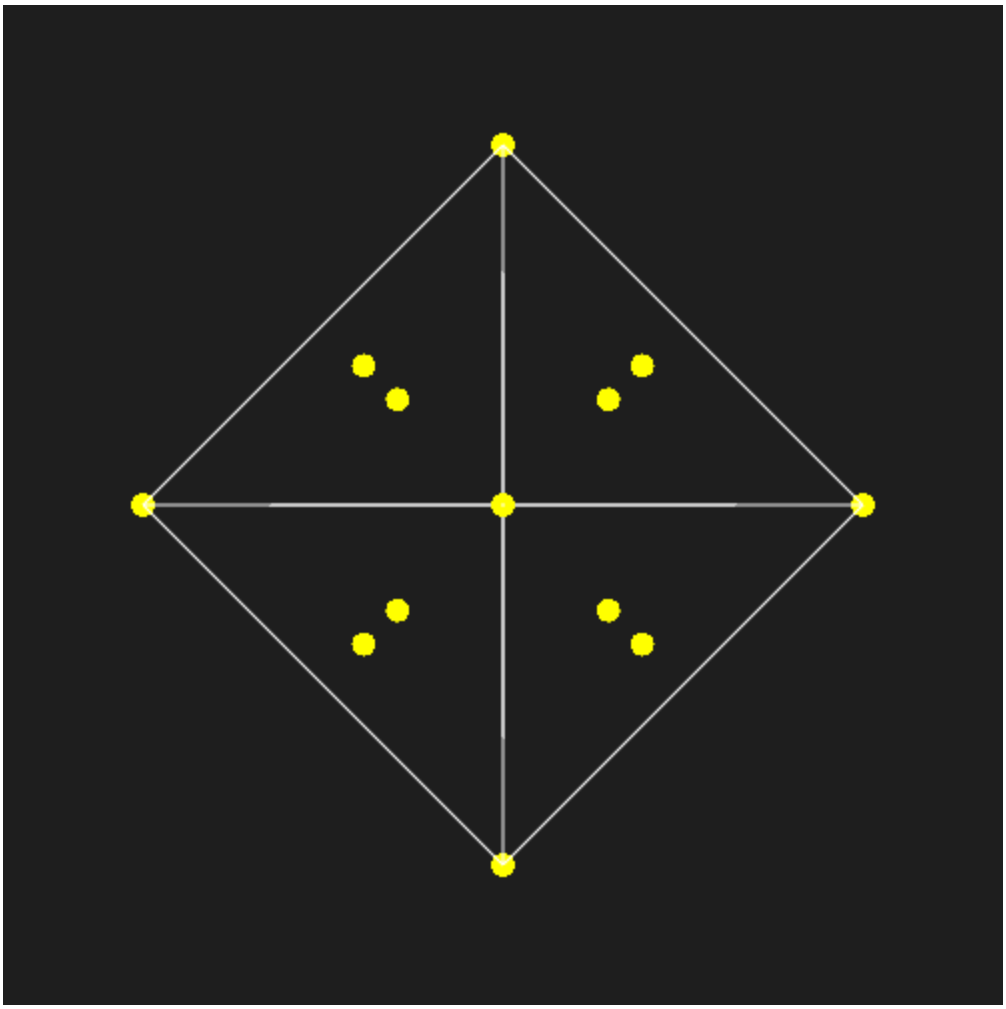
Para começo de conversa, o que é um poliedro?

Um Poliedro é um sólido de três dimensões com faces e figuras planas, que contenha bordas e vértices.

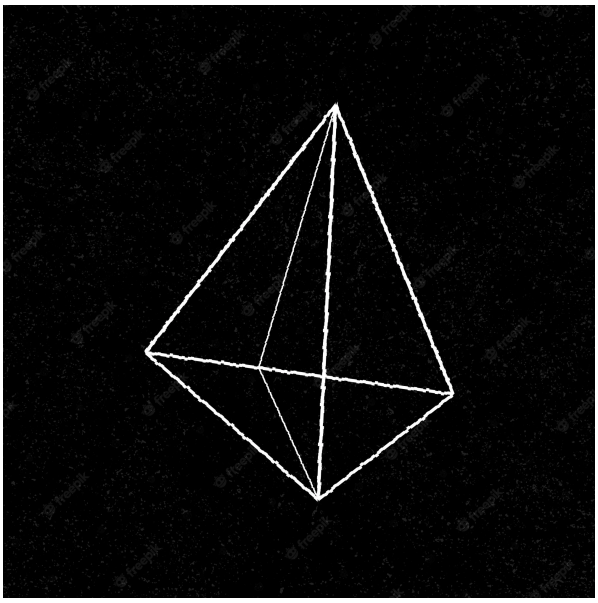
Portanto, um octaedro, é um poliedro com 8 faces,

algo parecido com isso aqui. Muito similiar àquele item que fica em cima da cabeça dos The Sims,





Então basicamente o que precisava ser feito era pegar os pontos em amarelo da imagem e tentar montar um tetraedro com isso. Ok! Mas vamos lembrar o que é um tetraedro



Um Tetraedro nada mais é do que uma pirâmide com 4 faces e base triangular! Algo assim ou assim em uma aplicação real.

Da pra ver que ela tem 4 vértices. né?

Então se um tetraedro tem quatro vértices. A Estratégia pra resolver isso aqui inicialmente vai ser montar todas as combinações possíveis de 4 pontos que a gente consegue formar usando os vértices e pontos centrais de um octaedro.

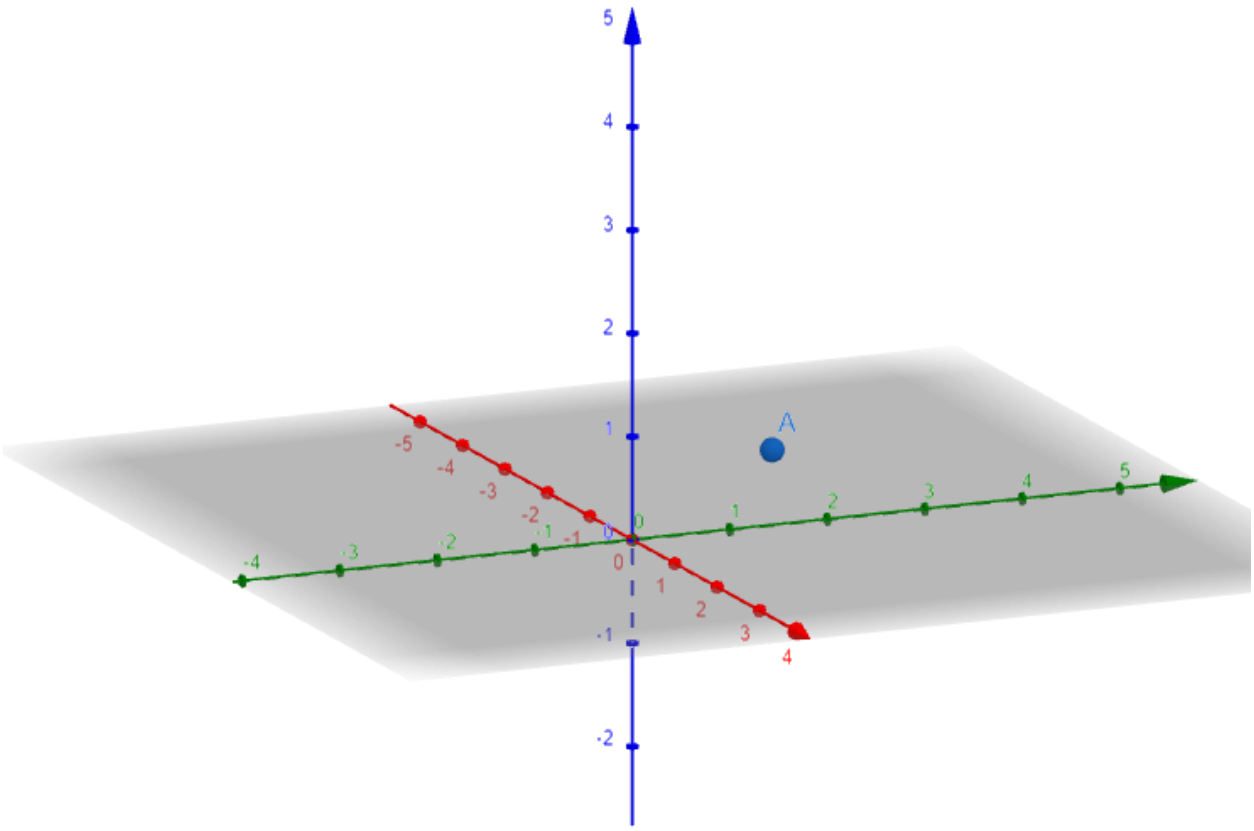
Essa não é a resposta final, mas é um começo. Ok?

Vamo levar isso pra código.

Para começar, nós precisamos fazer o python entender o que é um ponto no espaço tridimensional. E já vai deixando o Like pq esse raciocínio realmente é muito daroa! para isso vamos criar uma classe chamada ponto para paassar esse tipo de dado:

```
import numpy as np

class Point:
    def __init__(self,name:str, x_value:float, y_value:float,z_value:float):
        self._name = name
        self._x_value = x_value
        self._y_value = y_value
        self._z_value = z_value

    @property
    def name(self):
        return self._name
    @property
    def x(self):
        return self._x_value
    @property
    def y(self):
        return self._y_value
    @property
    def z(self):
        return self._z_value
```

Basicamente o que esse código faz é definir o que é um ponto por meio de de uma Classe. A função init diz os parâmetros que devem ser passados inicialmente para que seja possível, no caso um nome, e 3 coordenadas que definem o ponto no espaço tridimensional . As funções abaixo são apenas uma forma mais organizada de acessar o código



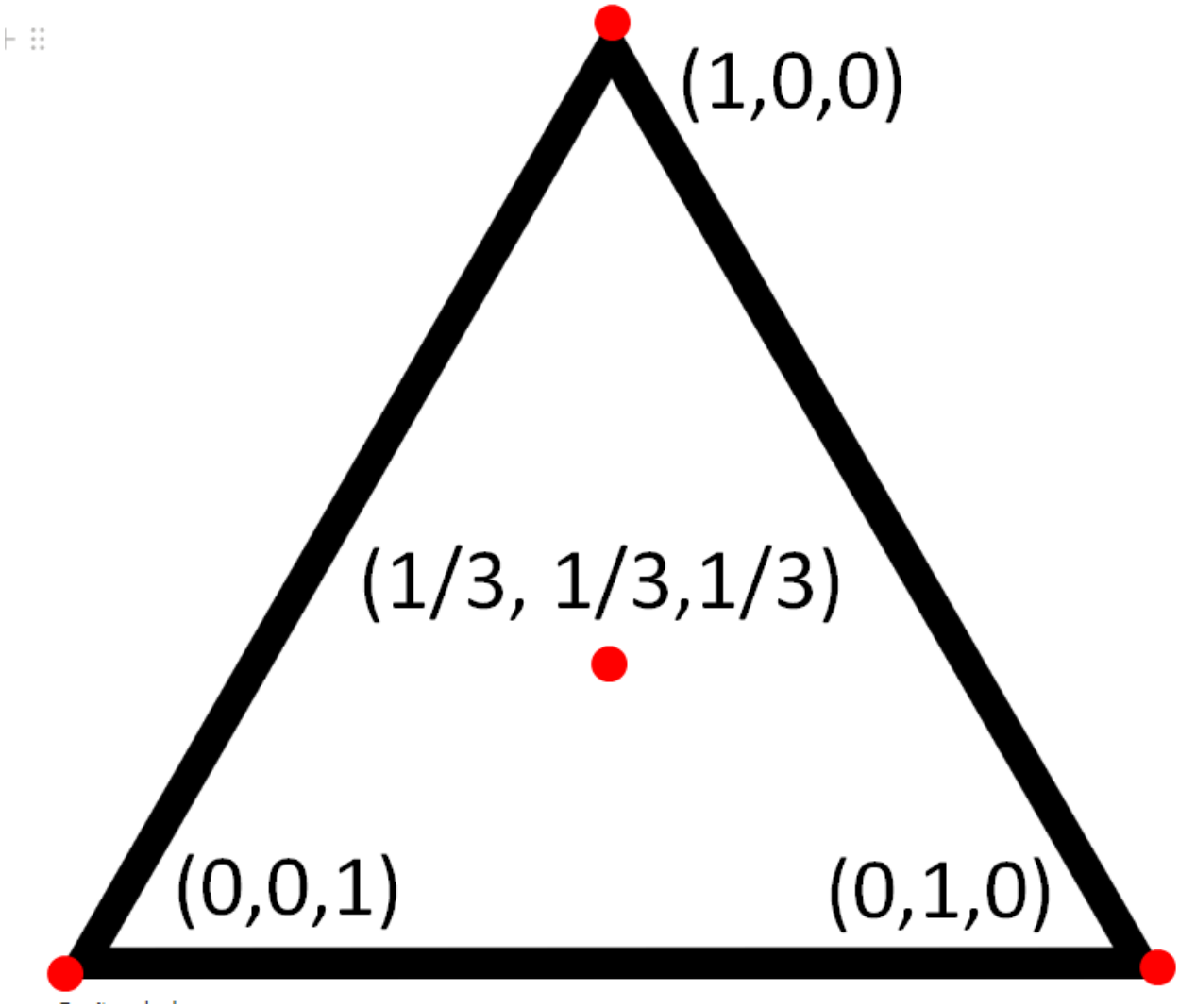
Com isso, é possível definir os 14 pontos do octaedro. Só vale fazer uma ressalva que o ponto do meio de cada face do octaedro pode ser calculado pela média dos três pontos que são vértices da face.

$$(1,0,0) + (0,1,0) + (0,0,1) = (1,1,1)$$

$$|(1,1,1)|^2 = 1^2 + 1^2 + 1^2 = 3$$

$$\frac{(1,1,1)}{|(1,1,1)|^2} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$$

face.



Então com isso criamos os seguintes códigos para os pontos

```
a = Point(name='a',x_value=1,y_value=0,z_value=0) # (1,0,0)
b = Point(name='b',x_value=-1,y_value=0,z_value=0)# (-11,0,0)
c = Point(name='c',x_value=0,y_value=1,z_value=0)# (0,1,0)
d = Point(name='d',x_value=0,y_value=-1,z_value=0)# (0,-1,0)

e = Point(name='e',x_value=0,y_value=0,z_value=1)# (0,0,1)
f = Point(name='f',x_value=0,y_value=0,z_value=-1)# (0,0,-1)

g = Point(name='g',x_value=1/3,y_value=1/3,z_value=1/3)# (1/3,1/3,1/3)
h = Point(name='h',x_value=-1/3,y_value=1/3,z_value=1/3)# (-1/3,1/3,1/3)
i = Point(name='i',x_value=1/3,y_value=-1/3,z_value=1/3)# (1/3,-1/3,1/3)
j = Point(name='j',x_value=1/3,y_value=1/3,z_value=-1/3)# (1/3,1/3,-1/3)

k = Point(name='k',x_value=-1/3,y_value=-1/3,z_value=1/3)# (-1/3,-1/3,1/3)
```

```
l = Point(name='l',x_value=1/3,y_value=-1/3,z_value=-1/3)# (-1/3,1/3,-1/3)
m = Point(name='m',x_value=-1/3,y_value=1/3,z_value=-1/3)# (1/3,-1/3,-1/3)
n = Point(name='n',x_value=-1/3,y_value=-1/3,z_value=-1/3)# (-1/3,-1/3,-1/3)
```

Com os pontos criados, precisamos fazer a combinação dos 14 pontos em grupos de 4. Precisamos cobrir todas as possibilidades, sem repetir os grupos. Por exemplo, um grupo formado por [a,b,c,d] é a mesma coisa que um grupo formado por [d,a,b,c].

Para resolver essa questão. Precisamos usar um propriedade chamada combinação. A combinação é definida pelo seguinte símbolo:

$$C_{14,4} = \frac{14!}{10!4!} = 1001$$

Só a resalva que o ! ai na fórmula indica uma notação de fatorial. que é a basicamente todos os números abaixo de n multiplicadso juntos. No caso, fatorial de 3 $3! = 3 \cdot 2 \cdot 1$

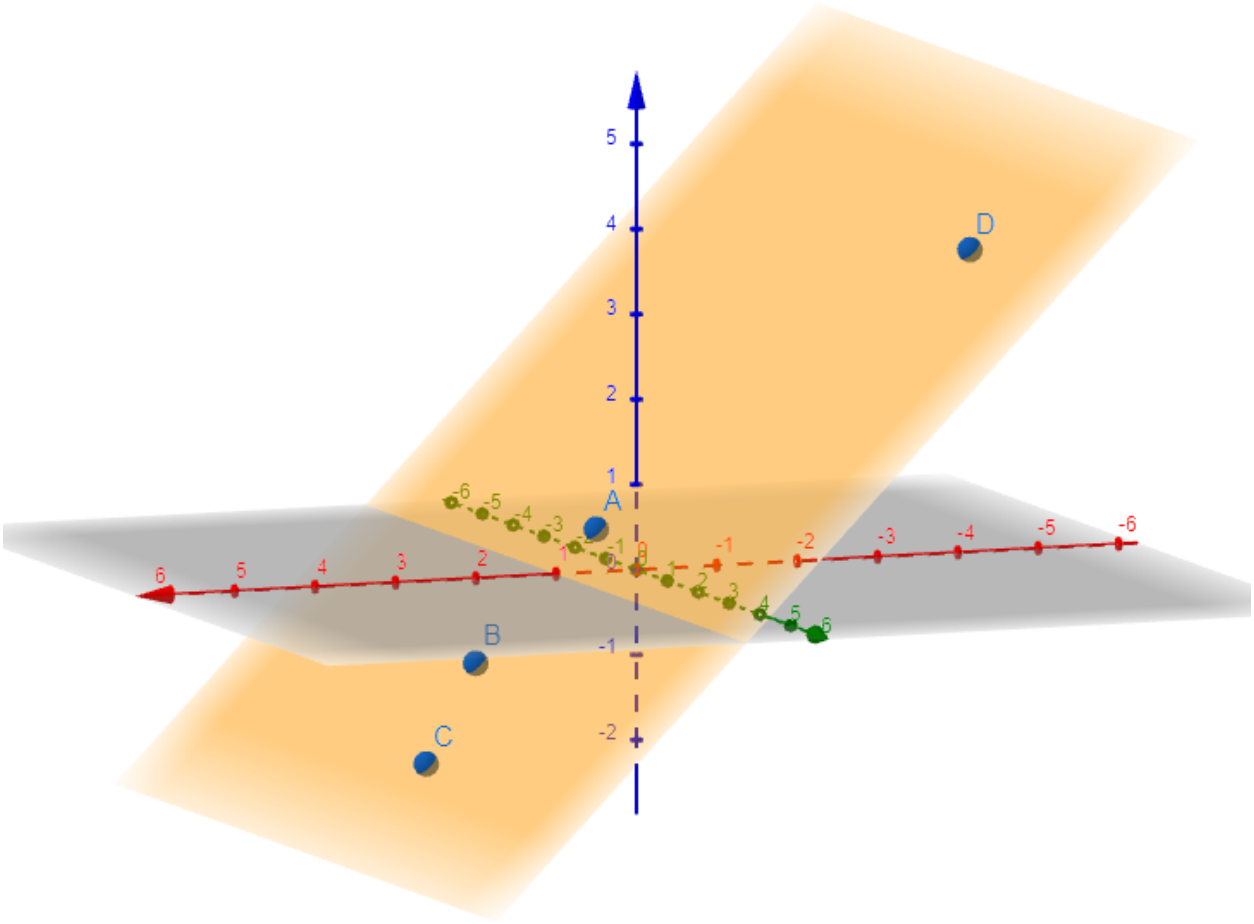
Ou seja, existem 1001 grupos de 4 pontos que podem ser formados com os 14 pontos do octaedro

Para ver todas essas combinações, decidimos escrever o seguinte algoritmo:

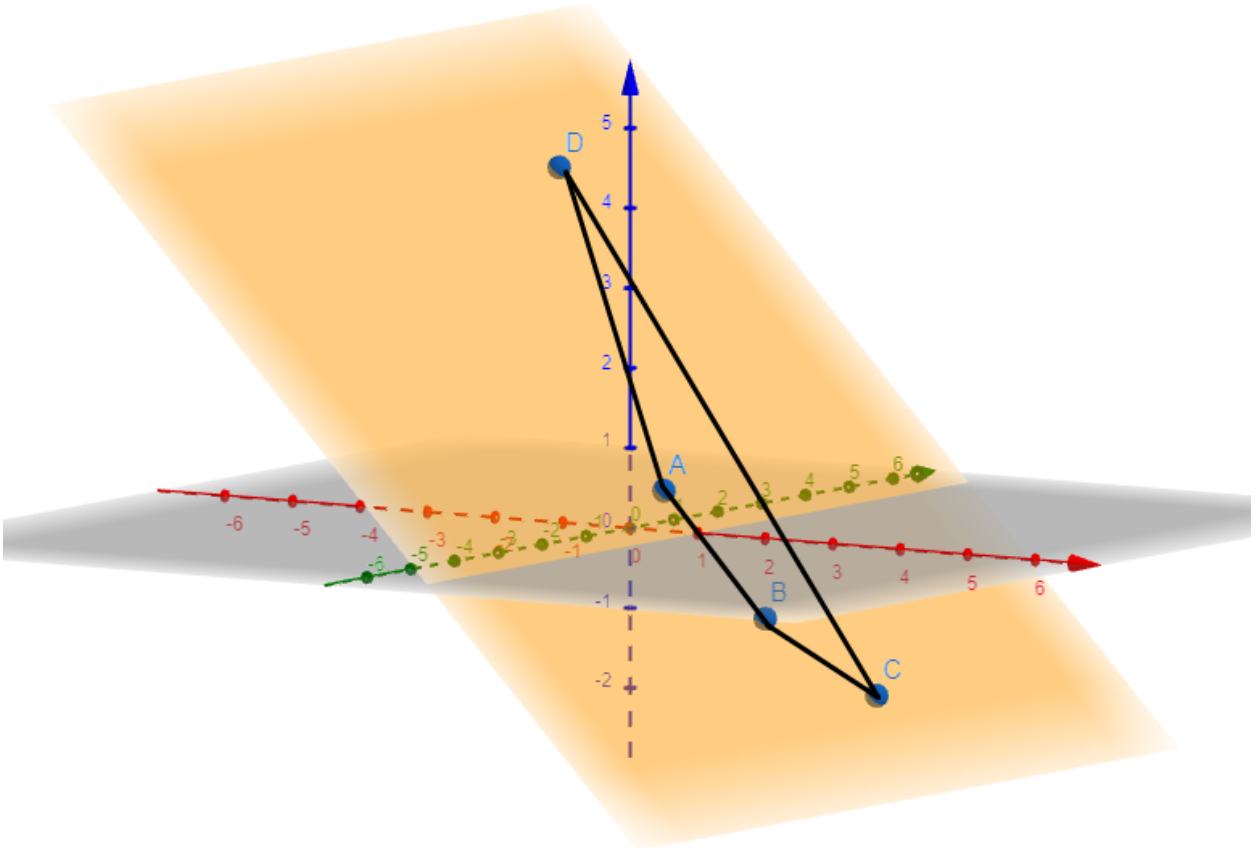
```
from itertools import combinations

points = [a,b,c,d,e,f,g,h,i,j,k,l,m,n]

def get_combinations(lst):
    return list(combinations(lst, 4))

combinations = get_combinations(points)
print('combinations',len(combinations))
```

A partir dai a gente tem uma ideia de como todas as combinações. Mas essa resposta ainda está errada pelo seguinte fato:

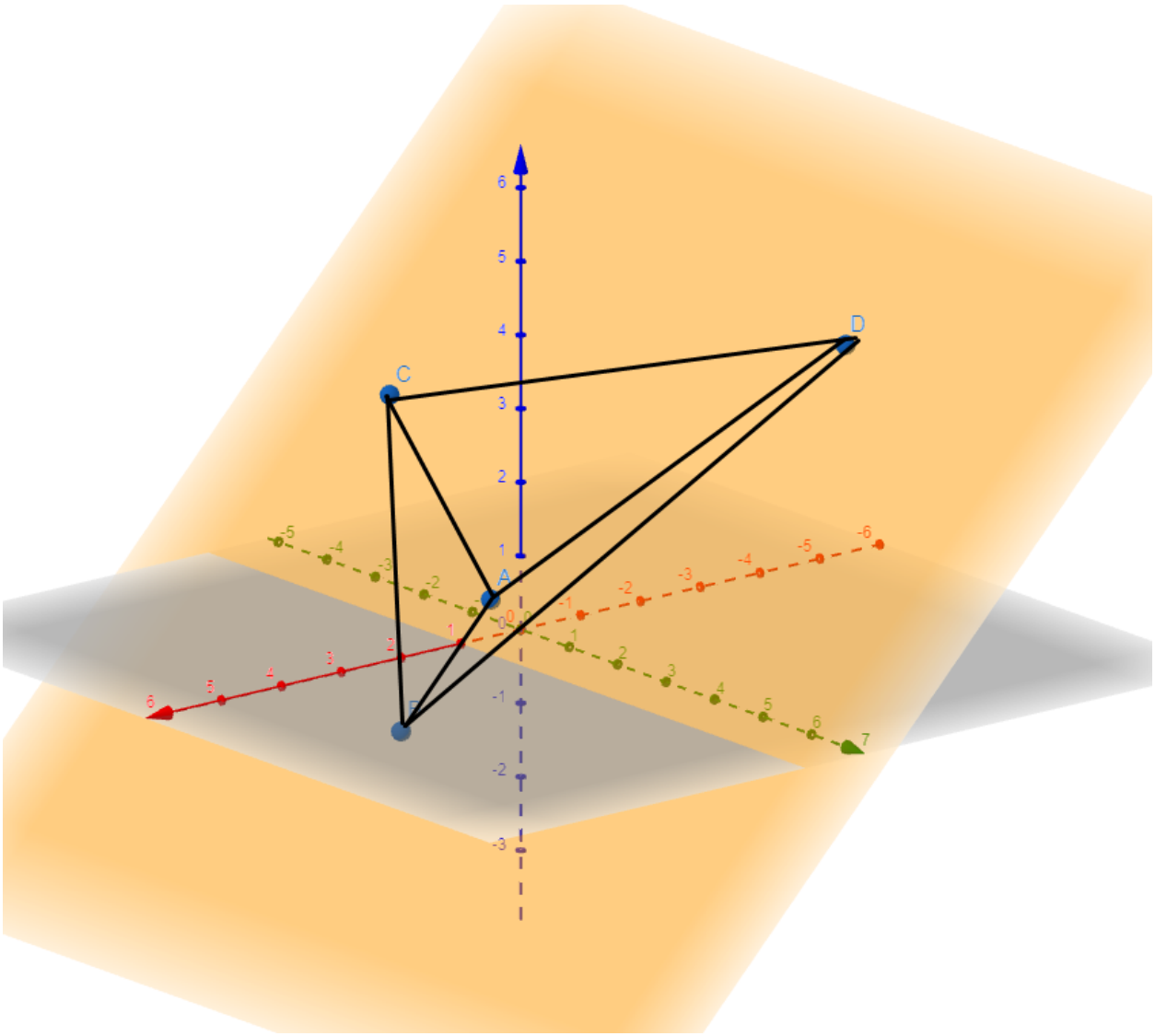
Imagine que todos os pontos estejam sobre o mesmo plano e por plano eu quero dizer. Imagine que fosse possível colocar bolinhas em cima sobre uma folha de papel.



Se todas as bolinhas estiverem sobre o mesmo plano, é nós conectarmos os pontos, nós vamos ter um quadrilatero, não um tetraedro



porém se pelo menos um dos pontos não pertencer ao plano, é possível desenhar um tetraedro a partir disso



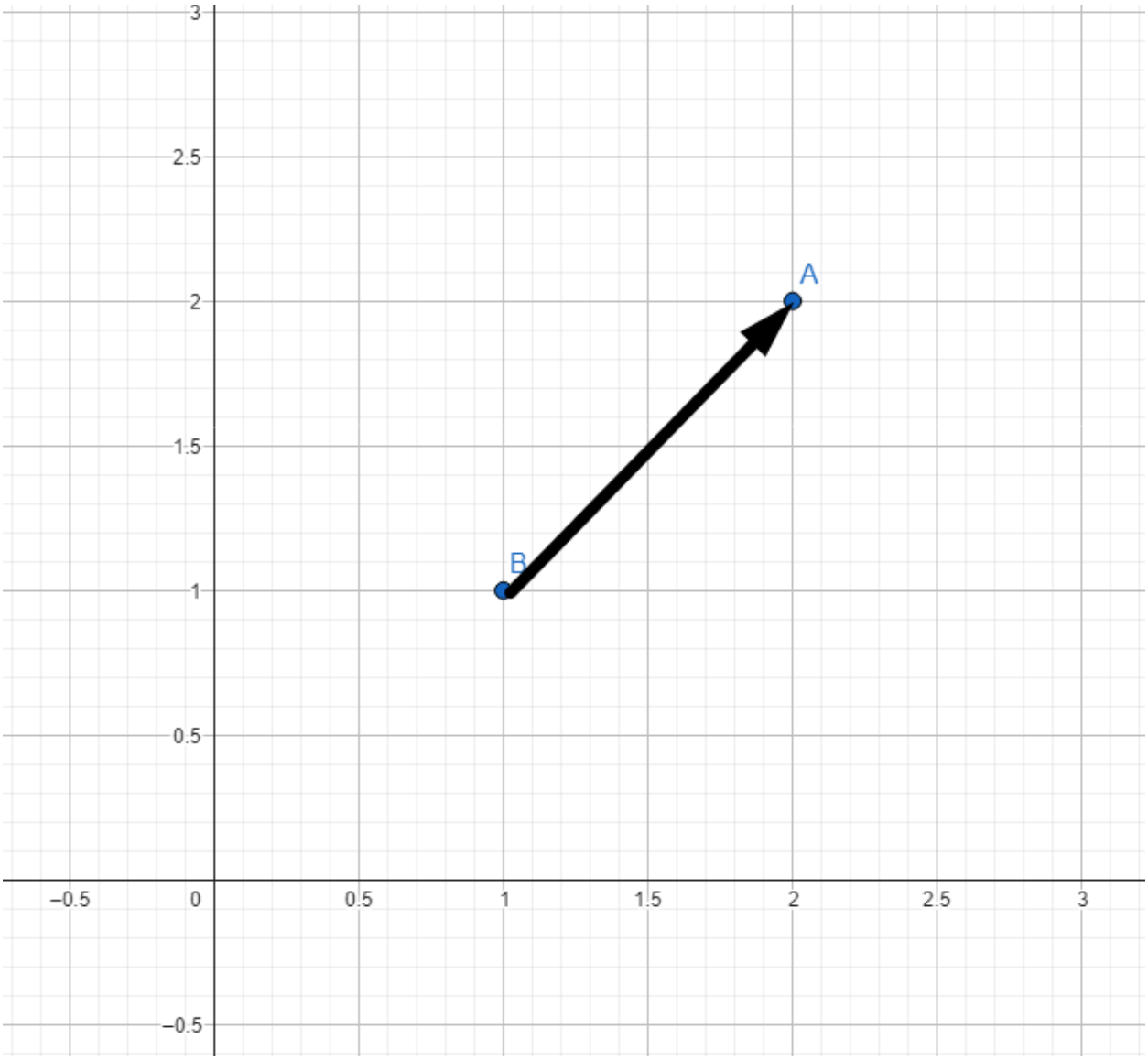
Então nós temos que de alguma forma, descosinderar os pontos que estejam no mesmo plano. E é ai que provavelmente vem a grande sacada desse desafio.

A matemática ensina, mais especificamente a algebra linear, que pontos pertecem ao mesmo plano quando o determinante de seus vetores correspondentes é igual a zero.

Calma! eu sei que ficou complexo, mas vamos desmistificar o que isso significa.

Para começo de conversa, o que é um vetor? um vetor é nada mais que uma flecha que aponta para um determinado lugar do espaço tridimensional.

Em outras palavras, é como se fosse um pauzinho que vai de um ponto a outro e indica um determinado sentido. Um graveto que aponta para a esquerda indica é um vetor que aponta para a esquerda.



Se partirmos do ponto B para ir ao ponto A, como fazemos para calcular seu vetor?

$$B = (1, 1)$$
$$A = (2, 2)$$
$$A - B = (2, 2) - (1, 1) = (2 - 1, 2 - 1) = \vec{V} = (1, 1)$$

Dado que é assim que calcula um vetor, para testar se estão no mesmo plano, ou, sua coplanariedade, precisamos verificar se os vetores, formados pelos pontos estão no mesmo plano. Para verificar isso, Precisamos pegar os vetores, coloca-los numa matriz e calcular seu determinante. Caso ele seja zero, os pontos estão no mesmo plano, caso contrário formam um tetraedro.

Vou mostrar um exemplo passo a passo aqui.

$$A = (1, 0, 0)$$
$$B = (0, 1, 0)$$
$$C = (0, 0, 1)$$
$$D = (0, 0, 0)$$

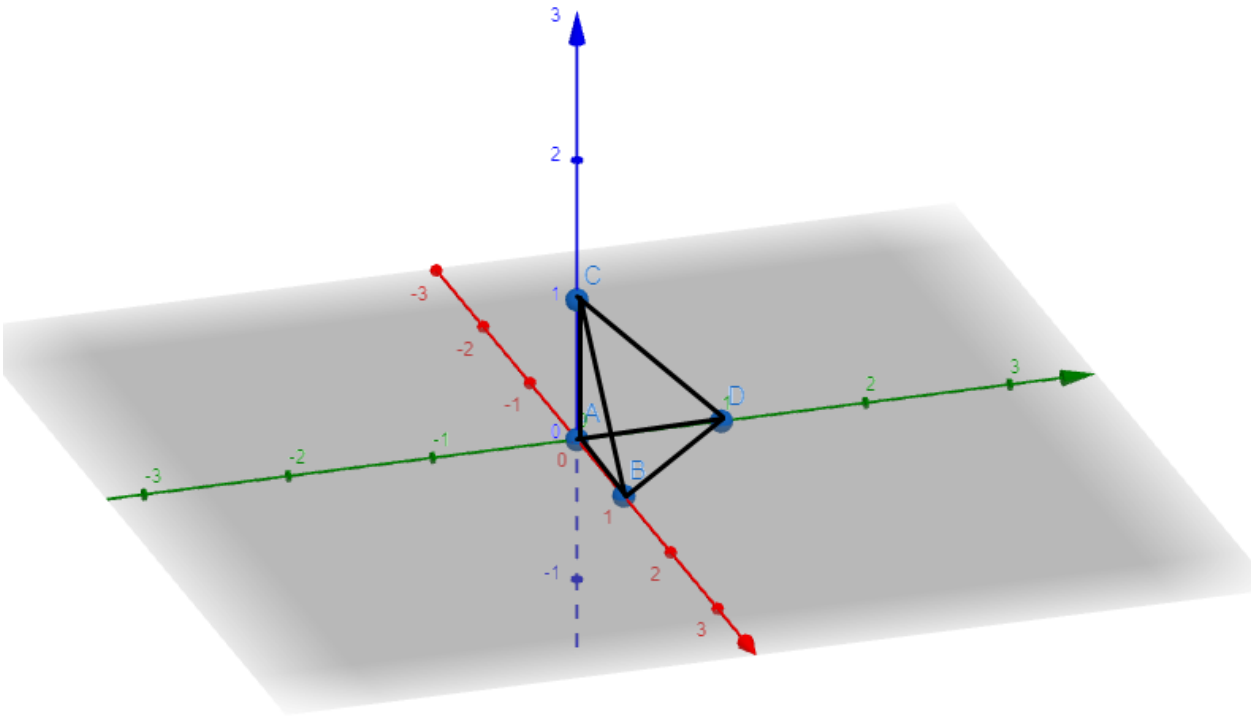
Se D for adotado como o “ponto de partida” dos vetores, é possível desenhar 3 vetores a partir dos 4 pontos.

$$\vec{x} = A - D$$
$$\vec{y} = B - D$$
$$\vec{z} = C - D$$

Com isso, colocando esses valores numa matriz

$$\begin{bmatrix} \vec{x} \\ \vec{y} \\ \vec{z} \end{bmatrix} = \begin{bmatrix} 1\ 0\ 0 \\ 0\ 1\ 0 \\ 0\ 0\ 1 \end{bmatrix}$$

e calculando o determinante, que é uma forma de calcular a relação entre essas coordenadas, chegamos à conclusão de que ele é 1, e portanto, por ser diferente de zero, forma um tetraedro.



Bom, sabendo de tudo isso, agora precisamos traduzir para código

```
import math

def same_plane(p0:Point, p1:Point, p2:Point, p3:Point):
    if p0 ==p1 or p0 == p2 or p1 == p3 or p1 == p2 or p1 == p3  or p2==p3:
        raise Exception('Same points passed!')

    v1 =[p1.x-p0.x, p1.y-p0.y, p1.z-p0.z]
    v2 = [p2.x-p0.x, p2.y-p0.y, p2.z-p0.z]
    v3 = [p3.x-p0.x, p3.y-p0.y, p3.z-p0.z]

    mtx =[v1,v2,v3]

    det = np.linalg.det(mtx)
    #print('det',det)
    if round(det,1)==0:
        return True
    return False
```

```
for comb in combinations:
        p0 = comb[0]
        p1 = comb[1]
        p2 = comb[2]
        p3 = comb[3]
        if not same_plane(p0=p0,p1=p1,p2=p2,p3=p3):
            tets.append(comb)

print(len(tests))
```

Com todo o raciocínio feito, nós entramos no site. Clicamos em ‘submeter’ resposta. E BOOM!

A resposta estava certa. 870 tetraedros podem ser formados dentro de um octaedro. E foi assim que nós conseguimos ganhar essa competição!

Se você gostou, não esqueça de deixar se inscrever para acomppanhar os próximos

Um grande abraço e fui!