VOTACIÓN HEXAGONAL - SOLUCIÓN GUÍA

Universidad de Bogotá Jorge Tadeo Lozano

Alvarado Becerra Ludwig - Franco Calderón Alejandro

Estructuras de datos - 2026-1S

Índice

1. Interpretación del problema, entrada y salida de datos

1.1. Objetivo

Lograr calcular la cantidad total de votos en total, los votos totales de cada uno de los tres candidatos, obtener la distancia del puesto de votación más lejano y el Índice Energético de Votación (IEV) de la mesa con más votos.

1.2. Entradad de datos esperada

Se nos dan 6 líneas con números enteros separados por espacios, el ejemplo dado en el enunciado es:

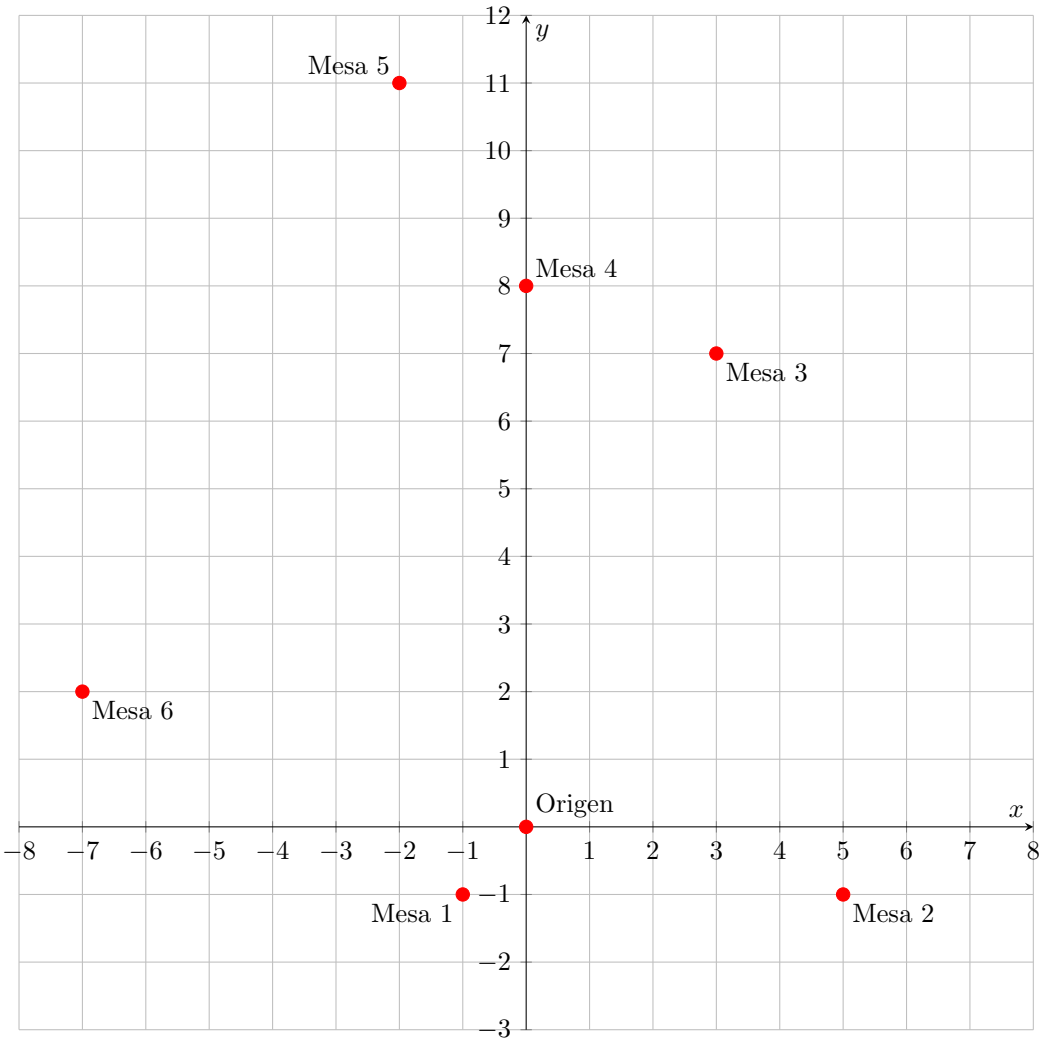
```
-1 -1 78 50 90
5 -1 100 90 115
3 7 20 15 10
0 8 40 55 30
-2 11 470 450 370
-7 2 90 95 80
```

Cada línea corresponde a una mesa de votación, teniendo en cuenta el enunciado del problema; el primer valor es la coordenada en x de la mesa, el segundo la coordenada y, el tercero es la cantidad de votos para Andrea, el cuarto es la cantidad de votos para Bobby, el último dato es la cantidad de votos para Claudia. Por lo tanto, la mesa 1 se encuentra en la coordenada $(-1, -1)$, 78 estudiantes votaron por Andrea en esta mesa, 50 por Bobby y 90 por Claudia. Aplicando esto mismo a los demás datos podemos ver:

Número mesa	Coordenada x	Coordenada y	Votos Andrea	Votos Bobby	Votos Claudia
1	-1	-1	78	50	90
2	5	-1	100	90	115
3	3	7	20	15	10
4	0	8	40	55	30
5	-2	11	470	450	370
6	-7	2	90	95	80

Para este primer caso[1] podemos representarlo visualmente en un plano cartesiano tomando las coordenadas x y y de cada mesa:

[1] Tomar en cuenta que pueden haber muchos más casos



1.3. Salida de datos esperada

Una vez entendido cómo están organizados los datos de entrada, se prosigue a entender el formato de salida requerido, los únicos detalles a tener en cuenta es el redondeo de los porcentajes a dos dígitos y aplicar lo mismo para la distancia de la mesa más lejana al origen, para más información puede visitar la sección 3.8.

2. Solución conceptual

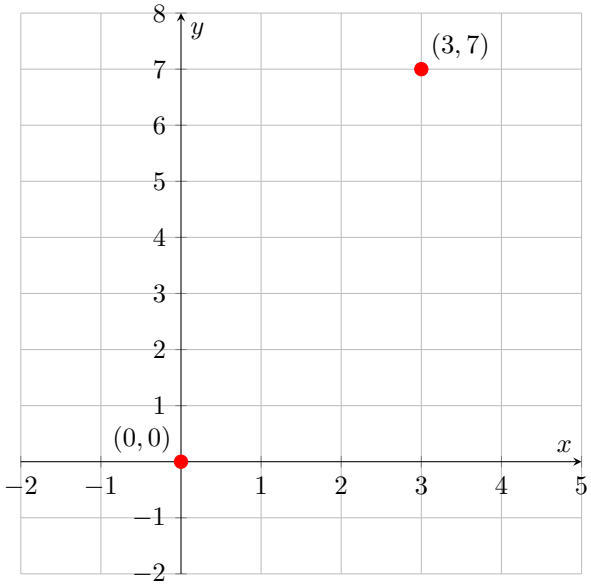
2.1. Conteo de votos

Se necesitan 4 variables para el conteo de votos; una para el conteo global y las otras tres para cada candidato. Cada vez que se leen los votos para un candidato en una mesa se deben agrega este valor a la variable de dicho candidato, repetir 6 veces dicha acción. Terminado el conteo individual de votos se puede asignar la suma de los votos de cada candidato a la variable global. Para obtener el porcentaje total de los votos de candidato se puede utilizar:

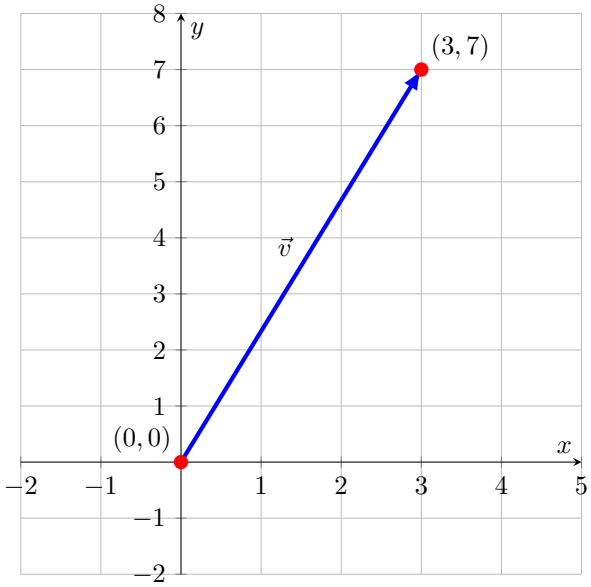
$$\%\text{Votos Candidato} = \frac{\text{Total Votos Candidato}}{\text{Total Votos Global}} \times 100$$

2.2. Distancia mesa con más votos

Inicialmente se inicializa en 0 una variable de `distancia_max` y cada vez que se reciben las coordenadas de una mesa se debe computar la distancia desde el origen hasta dicho punto. Tome en cuenta el siguiente plano cartesiano visualizando el origen y la mesa 3 (datos del ejemplo del ejercicio):

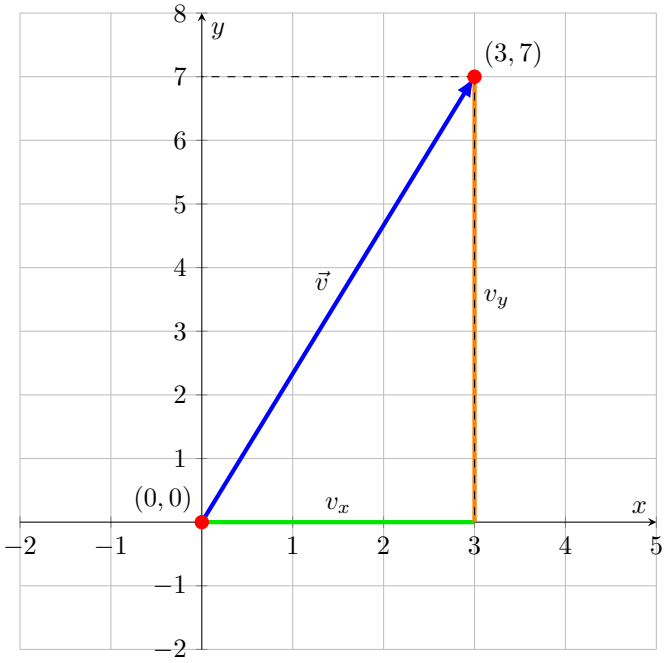


Se puede trazar un vector $\vec{v}$ desde el origen hasta el punto $(3, 7)$:



La magnitud $|\vec{v}|$ de este vector $\vec{v}$ se interpreta dentro del problema como "la distancia desde

el origen hasta la mesa 3", recordando clases de Física I, se sabe que para sacar la magnitud de este vector se deben sacar las componentes rectángulares v_x y v_y, gráficamente:

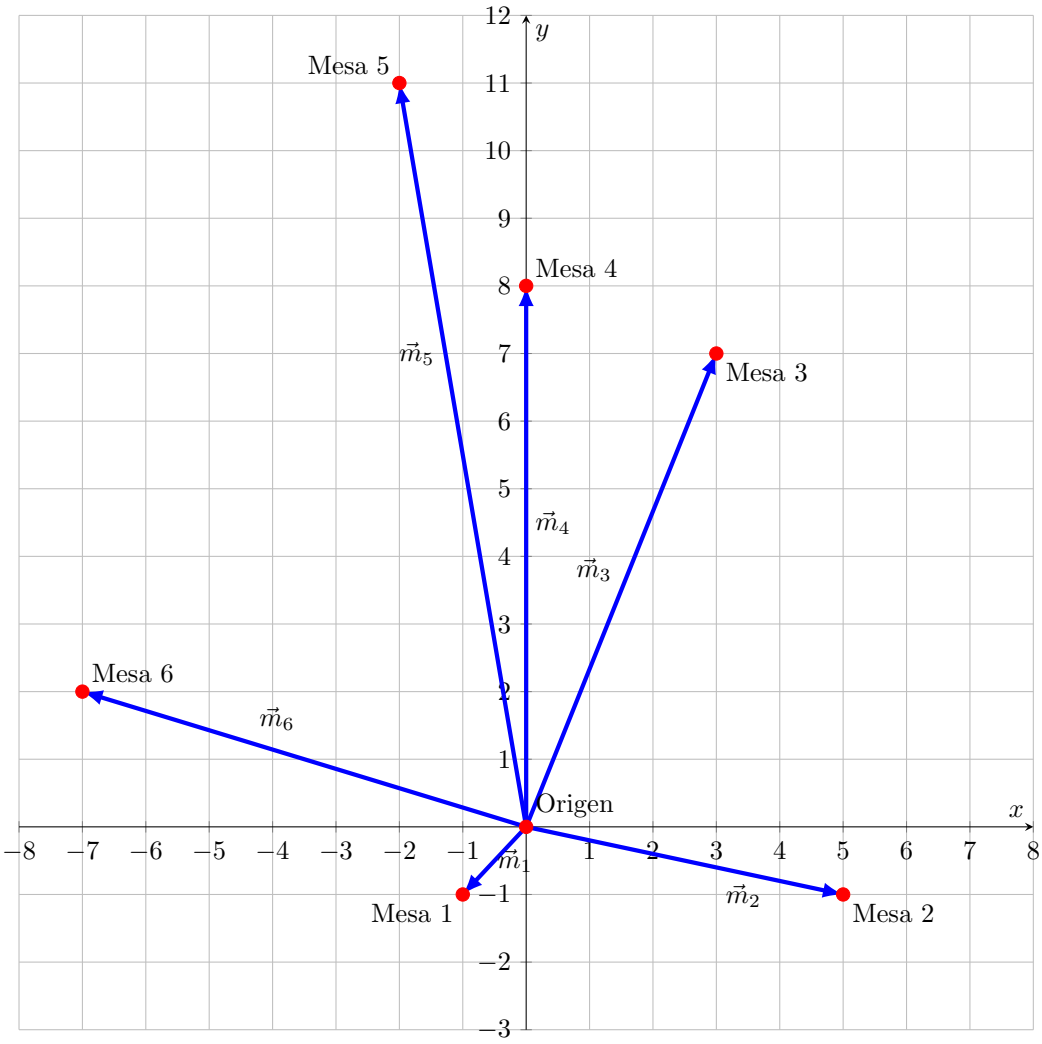


Para sacar la magnitud de $\vec{v}$ se puede utilizar la siguiente fórmula[2]:

$$|\vec{v}| = \sqrt{(v_x)^2 + (v_y)^2}$$

Una vez sacado el procedimiento para obtener la distancia, se sigue una estrategia de fuerza bruta para obtener la distancia máxima comparando entre todas las distancias (mire el gráfico de la siguiente pagina) cuál valor es mayor, si `distancia_max` o la distancia recién obtenida del cálculo.

[2]Para saber más información de esta fórmula puede visitar el siguiente enlace `https://en.wikipedia.org/wiki/Euclidean_space#Euclidean_norm`



En conclusión, sacar la magnitud para todos los vectores $\vec{m}_i$ $(1 \leq i \leq 6)$ y guardar en `distancia_max` la magnitud máxima.

2.3. Índice Energético de Votación

Se nos pide sacar el IEV de la mesa que posee más votos recogidos. Para obtener la mesa con mayor votos se crean dos variables globales `mesa_iev` y `votos_iev` inicializadas en 0, cada vez que se lea cada línea y se extraigan los votos para cada candidato se compara si ese resultado es mayor al global `votos_iev` entonces se asigna `votos_iev` a la suma de los votos de la mesa actual y guarde el número de la mesa en `mesa_iev`, si no, continue con el programa.
Una vez obtenida qué mesa tiene la mayor cantidad de votos se puede aplicar la fórmula dada en el problema:

$$IEV = \lceil \sqrt[3]{V} \rceil$$

Debido a que se está permitido usar las funciones `cbrt()` y `ceil()` del módulo `math`, realizar estas dos operaciones es trivial. Sin embargo, no se puede utilizar la función `fact()`, por lo

tanto, toca realizar este cálculo utilizando la definición del factorial para cualquier número entero positivo n:

$$n! = \prod_{i=1}^{n} i$$

Modificando para el caso del IEV, se cómputa entonces:

$$IEV = \prod_{i=1}^{\lceil \sqrt[3]{V} \rceil} i$$

Donde V son los votos de la mesa con mayor votos recaudados.

3. Código modelo

Se va a explicar parte a parte una posible solución al problema, recuerde que puede probar con diferentes implementaciones o incluso formulando más soluciones al problema, se invita al estudiante que haga ese ejercicio.
En primer lugar, se importa el modulo de `math` para utilizar las funciones de `ceil()`, `sqrt()` y `cbrt()`.

```
1 import math
```

3.1. Declaración de variables

Ahora, se declaran 9 variables para ayudar al almacenamiento de los datos:

- `total_votos`: Conteo global de votos.
- `andrea`: Votos totales de Andrea.
- `bobby`: Votos totales de Bobby.
- `claudia`: Votos totales de Claudia.
- `mesa_lejana`: Número de la mesa más lejana al origen.
- `distancia`: Distancia desde el origen hasta la mesa más lejana.
- `IEV`: Índice Energético de Votación de la mesa con mayor número de votos.
- `mesa_iev`: Número de la mesa con más votos recogidos.
- `votos_iev`: Número de votos de la mesa con más votos recogidos.

Teniendo el contexto del significado de cada variable se pueden inicializar cada una de ellas de la siguiente manera:

```
1 total_votos = 0
2 andrea = 0
3 bobby = 0
4 claudia = 0
```



```
mesa_lejana = 0
distancia = -1
IEV = 1
mesa_iev = 0
votos_iev = 0
```

3.2. Lectura de datos

Ahora, se deben leer los datos de entrada, se sabe que solo van a haber 6 líneas se puede realizar un ciclo `for` con rango de 1 a 6 (tenga en cuenta que i va a ser el número de la mesa), incluyendo ambos extremos.

```
for i in range(1, 7):
```

Dentro del ciclo `for` se guardan los datos de entrada utilizando la función `map()`, esta función toma dos parámetros; una función y un *objeto iterable*. A cada elemento del *objeto iterable* (e.j., una lista) se le aplica la función colocada en el primer parámetro[3]. Utilizando el desempaquetado de tuplas de Python se le pueden asignar a 6 variables la línea de datos de entrada[4]:

```
x, y, a, b, c = map(int, input().split())
```

Por ejemplo, cuando el usuario digite:

```
-1 -1 78 50 90
```

Las variables `x`, `y`, `a`, `b` y `c` van a tener los valores de; -1, -1, 78, 50, y 90, respectivamente. **Recordatorio:** La función `input()` retorna una cadena y `.split()` convierte una cadena a lista usando un separador, por defecto es el carácter de espacio, es decir, cuando el usuario escriba:

```
"-1 -1 78 50 90"
```

Se aplica el `split()` quedando como:

```
["-1", "-1", "78", "50", "90"]
```

Permitiendo que se pueda aplicar la función `map()` explicada anteriormente.

3.3. Votos totales para cada candidato

Una vez obtenido los votos de cada candidato en la mesa i se añade a su respectiva variable global los votos de esa mesa:

```
andrea += a
bobby += b
claudia += c
```

[3]Para más información del funcionamiento de la función `map()` puede visitar la documentación oficial `https://docs.python.org/3/library/functions.html#map`

[4]Se recomienda realizar la lectura de esta sección de la documentación de Python `https://docs.python.org/3/tutorial/datastructures.html#tuples-and-sequences`

3.4. Distancia de la i-ésima mesa de votación

Como también se tienen las coordenadas de la mesa i se obtiene la distancia desde el origen hasta le mesa i utilizando lo explicado en la sección 2.2 y comparar si esta es mayor a la variable declarada anteriormente de `distancia`:

```
dist_actual = math.sqrt(x**2 + y**2)

if (dis_actual > distancia):
    distancia = dist_actual
    mesa_lejana = i
```

3.5. Mesa con más votos recogidos

Ahora, se debe obtener cuál fue la mesa con más votos y cuántos tuvo utilizando la suma de las variables `a`, `b` y `c`, comparando con la variable anteriormente declarada `votos_iev`:

```
if (a + b + c > votos_iev):
    votos_iev = a + b + c
    mesa_iev = i
```

3.6. Votos totales en las elecciones

Esas fueron las acciones que debe repetir ese ciclo `for`, una vez finalizado se puede obtener la cantidad de votos en las elecciones utilizando las variables declaradas inicialmente:

```
total_votos = andrea + bobby + claudia
```

3.7. Índice Energético de Votación

El último dato que se necesita sacar es el IEV, teniendo en cuenta la teoría de la sección 2.3 se itera desde un intervalo de $[1, \lceil \sqrt[3]{\texttt{total\_votos}} \rceil]$ y se múltiplica por i el valor actual de `IEV`, reasignando este mismo a dicha operación:

```
for i in range(1, math.ceil(math.cbrt(votos_iev))+1):
    IEV *= i
```

3.8. Salida de resultados

Ya obtenidos todos los datos necesarios se utiliza la ayuda de la función `round()` para redondear los datos necesarios, además de computar el porcentaje de votos totales obtenidos por cada candidato. La salida que va a ver el usuario entonces se codifica[5] como:

```
print(f"Total Votos: {total_votos}")
print(f"Votos Andrea: {andrea} {round(andrea/total_votos*100,
    2)}%")
```

[5] El penúltimo dato se debe escribir como “Mesa más lejana”, se escribe sin tilde debido a errores con la compilación de LaTeX


```
print(f"Votos Bobby: {bobby} {round(bobby/total_votos*100, 2)}%")
print(f"Votos Claudia: {claudia} {round(claudia/total_votos*100, 2)}%")
print(f"Mesa mas lejana: {mesa_lejana} {round(distancia, 2)}")
print(f"IEV: {mesa_iev} {IEV}")
```

Finalmente, se puede ensamblar cada pieza de código en uno solo como se muestra a continuación:

```
import math

total_votos = 0
andrea = 0
bobby = 0
claudia = 0
mesa_lejana = 0
distancia = -1
IEV = 1
mesa_iev = 0
votos_iev = 0

for i in range(1, 7):
    x, y, a, b, c = map(int, input().split())
    dist_actual = math.sqrt(x**2 + y**2)

    if (dist_actual > distancia):
        distancia = dist_actual
        mesa_lejana = i

    if (a + b + c > votos_iev):
        votos_iev = a + b + c
        mesa_iev = i

    andrea += a
    bobby += b
    claudia += c
total_votos = andrea + bobby + claudia

for i in range(1, math.ceil(math.cbrt(votos_iev))+1):
    IEV *= i

print(f"Total Votos: {total_votos}")
print(f"Votos Andrea: {andrea} {round(andrea/total_votos*100, 2)}%")
print(f"Votos Bobby: {bobby} {round(bobby/total_votos*100,
```



```
    2)}%")
36 print(f"Votos Claudia: {claudia}
    {round(claudia/total_votos*100, 2)}%")
37 print(f"Mesa mas lejana: {mesa_lejana} {round(distancia, 2)}")
38 print(f"IEV: {mesa_iev} {IEV}")
```


UTADEO

UNIVERSIDAD DE BOGOTÁ JORGE TADEO LOZANO