En `Python` podemos representar una matriz mediante una lista de listas. Es decir, cada fila la guardamos en una lista, y guardamos entonces la lista de las filas, obteniendo una lista de listas. Por ejemplo, la matriz `M=[[1,0,0],[0,1,1]]` representa la matriz:

$$M = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix}.$$

Implemente en `Python` tres funciones correspondientes a cada uno de los siguientes problemas, en los que se tienen en cuenta solamente matrices de 0's y 1's. Ver la imagen de la siguiente página como complemento.

Problema 1: Dada una matriz M devolver la submatriz rectangular de mayor área que no contiene ningún 1. La salida de la función será una tupla (i, j, a, b) de cuatro componentes que representa la submatriz. La esquina superior izquierda de la submatriz es la celda de fila i y columna j de la matriz M, y a y b son el ancho y la altura de la submatriz de salida, respectivamente. Se debe implementar la función

```
1 def maximo_rectangulo(M):
2     pass
```

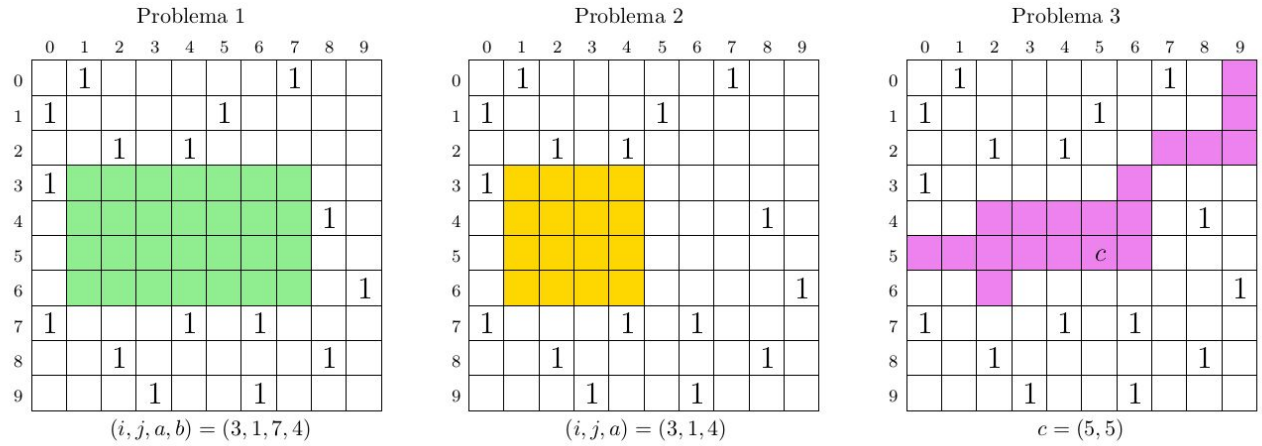
tal que `M` es la matriz de ceros y unos de cualquier dimensión. Si `M` es la matriz de la figura de la izquierda, entonces el resultado a retornar será la tupla `(3,1,7,4)`, correspondiente al rectángulo sombreado. Este rectángulo tiene esquina superior izquierda la celda de fila 3 y columna 1, así como ancho 7 y altura 4. En caso de existir más de un rectángulo vacío de 1's pero de áreas iguales se retornará cualquiera de ellos.

Problema 2: Dada una matriz M devolver la submatriz cuadrada de mayor área que no contiene ningún 1. La salida de la función será una tupla (i, j, a) de tres componentes que representa la submatriz. La esquina superior izquierda de la submatriz es la celda de fila i y columna j de la matriz M, y a es el el ancho y alto de la submatriz de salida. Se debe implementar la función

```
def maximo_cuadrado(M):
    pass
```

tal que `M` es la matriz de ceros y unos de cualquier dimensión. Si `M` es la matriz de la figura del centro, entonces el resultado a retornar puede ser la tupla `(3,1,4)`, correspondiente al cuadrado sombreado, como también cualquier tupla correspondiente a otro cuadrado vacío de 1's de lado 4.

Problema 3: Una celda de la una matriz se dice *libre* si en su vecindad todas las celdas contienen un 0. La *vecindad* de una celda, correspondiente a una submatriz de dimensiones 3×3, es la celda



misma y las (a lo más) ocho celdas adyacentes a esta. La función a implementar recibe la matriz M, la fila i y columna j de una celda de partida c, y devuelve un *iterador* de todas las celdas libres de la matriz que se conectan con la celda c a través de un camino formado únicamente por celdas libres. Se entiende que un camino es un movimiento de celda en celda, siempre yendo de una celda hacia una adyacente. Se debe implementar la función:

```
def iterador_celdas_libres(M, i, j):
    pass
```

tal que `M` es la matriz de ceros y unos de cualquier dimensión, e `i` y `j` son la fila y columna de la celda de partida. Para este problema se recomienda la lectura del la sección "Objetos iterables" de la guía de `Python`. Si `M` es la matriz de la figura de la derecha, el llamado a `iterador_celdas_libres(M,5,5)` debe devolver un *iterador* de las celdas sombreadas. Cada elemento entregado por el iterador será una tupla de dos componentes, equivalente a una celda de la matriz. Observemos en la misma figura que la celda $(0, 3)$ es libre, pero no será entregada por el iterador ya que no está conectada con la celda $c = (5, 5)$ a través de algún camino que solo contemple celdas libres.

En los problemas 1 y 2, si la matriz M es de n filas y m columnas, el tiempo de ejecución debe ser $O(n \cdot m \cdot \min\{n, m\})$. Para ello estudiar la versión en dos dimensiones del algoritmo de Kadane para calcular la submatriz o subarreglo de suma máxima. También mirar la sección "Subsecuencia de suma máxima" de la guía de `Python`. Para el problema 3 el tiempo de ejecución de toda la iteración debe ser $O(N)$, donde N es el total de celdas que nos entregará el iterador.

Como ejemplos podemos ver la siguiente figura:

1. Puede realizar los algoritmos como desee (pandas, numpy, machine lerning, etc.)
2. Guarde el resultado en sqlite.
3. Genere un Json de la de los datos creadas y guárdelo como data.json
4. La prueba debe ser entregada en un repositorio git
5. Usar Test Unitarios (plus)
6. Presenta un diseño de su solución.
7. Entregar Dockerfile y docker compose con la solucion

8. Guarde la info en SQLite dentro del docker
9. En el código tanto las estructuras de datos como los algoritmos tienen que ser eficientes, y esto suma puntos a la nota del trabajo
10. El código debe contener las tres funciones que se piden. Cada función debe recibir los parámetros de entrada y devolver el resultado. No se admitirán funciones que hagan algo distinto, como imprimir un mensaje en la pantalla.
11. PLUS. Resolver el siguiente sistema de ecuaciones

$$-F_r + \omega_x = ma$$
$$N - \omega_y = 0$$

$$F_r = N\mu \qquad \omega_x = wsen\theta \qquad \omega_y = wcos\theta$$

$$\mu = 0,2; \omega = mg; m = 10kg; a = .012m/s^2$$

y g es la gravedad tomar como 10 m/s^2