**Jueves 4 de julio del 2024**

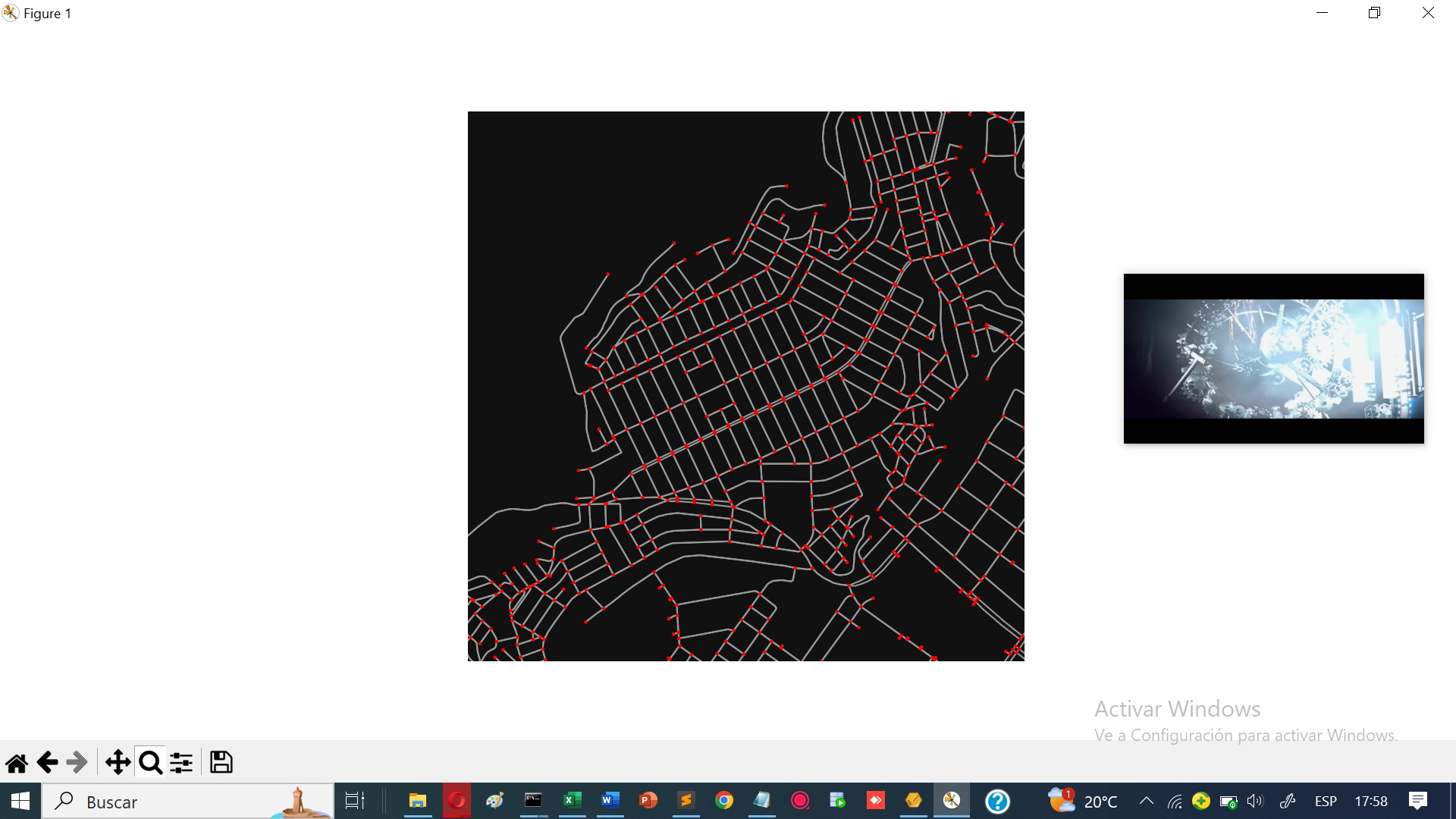
* *Desarrolle el siguiente examen de forma grupal*
* *Fecha de entrega martes 9 de julio a las 23:59 pm del 2024*
* *Elabore todos sus procedimientos en GoogleColab con Python*

*Los productos de este examen son los siguientes:*

1. *Documento donde se expone el procedimiento de solución al problema*
2. *Programas fuentes.*
3. *Procedimiento de ejecución del procedimiento / paso a paso.*
4. *Datos recolectados y sus respectivas transformaciones.*
5. *Todos estos elementos deben estar disponibles una dirección de Google drive.*

# **OBJETIVO**

Elaborar un procedimiento para implementar un **agente de búsqueda** en grafos planares (un grafo sin cruces de aristas).



El agente debe de ayudar a un conjunto de **feligreses** quienes, luego de haber cometido **muchos pecados**, tienen la siguiente **penitencia**:

1. Recorrer todas las iglesias del distrito (sin distinción de la religión, ni de su estado de conservación), en el menor tiempo posible.
2. Si el tiempo que toma un computador para calcular la ruta más corta es mayor que el tiempo que demora un feligrés para recorrer todas las iglesias usando la estrategia de búsqueda avara, entonces todos los feligreses pasaran de inmediato al **purgatorio**, esperando hasta que el computador termine de hacer el cálculo.

El grupo debe de preparar un procedimiento para apoyar a estos **preocupados feligreses**, a no tener una **penitencia** tan larga.

Con esta finalidad deben de desarrollar las siguientes acciones.

# **DISEÑAR UN AGENTE DE BÚSQUEDA.**

El agente de búsqueda tiene las siguientes características:

## **TIPO DE AGENTE**

Agente basado en metas, búsqueda en espacio de estados.

## **PERCEPCIONES**

Las percepciones del agente de búsqueda son:

1. Un conjunto de **ubicaciones** de una región entregadas en un plano 2D.
2. Un conjunto de **líneas de dirección** que permite pasar de una **ubicación** a la siguiente **ubicación** a la que se puede dirigir.
3. Un conjunto de **lugares de interés** identificados sobre la misma **región**.
4. El número de **agentes viajeros**.
5. La **ubicación inicial** del que parten los agentes viajeros es solo un punto, que corresponde a una ubicación.
6. Considere que todas las líneas de dirección se encuentran **habilitadas**, es decir no existen líneas inhabilitadas.

## **ACCIONES**

Las acciones (resultado) del agente de búsqueda es una secuencia de **ubicaciones** y **lugares de interés** por las que deben caminar los feligreses, desde la primera iglesia para recorrer todas las iglesias.

Los preocupados **feligreses** solo pueden caminar sobre las líneas de dirección, no pueden usar atajos.

## **MEDIDA DE RACIONALIDAD**

La medida de racionalidad de cada ejecución del agente es:

Donde i es una instancia del problema, es decir de una percepción del agente.

## **TAMAÑO DEL ESPACIO DE ESTADO.**

El grupo calcula el tamaño del espacio de estados para el problema, según los parámetros estadísticos calculados en la sección 3.5.

Asumiendo cierta velocidad de cálculo de su computador, el grupo estima el **tiempo máximo** que se requiere para recorrer todo el espacio de estado. Este sería el tiempo máximo de **penitencia**.

# **DISEÑAR UN AGENTE DE BÚSQUEDA.**

## **DESCARGAR LAS UBICACIONES Y LÍNEAS DE DIRECCIÓN.**

Usar la librería [**osmnx**](https://osmnx.readthedocs.io/en/stable/) para descargar ubicaciones y líneas de dirección desde [**OpenStreetMap**](http://www.openstreetmap.org), de todo un distrito.

|  |
| --- |
| import osmnx as ox  import matplotlib.pyplot as plt  # Definir el lugar y descargar el gráfico de red  place\_name = "Chorrillos, Lima, Peru"  G = ox.graph\_from\_place(place\_name, network\_type='drive')  # Convertir los nodos a un DataFrame  nodes, edges = ox.graph\_to\_gdfs(G)  # Visualizar el gráfico de red  fig, ax = ox.plot\_graph(G)  plt.show() |

En este caso se ha colocado el distrito de Chorrillos, el grupo selecciona alguno de su interés, no usar el distrito de Chorrillos.

## **DESCARGA DE LUGARES DE INTERÉS**

Usar la librería **osmnx** para descargar un conjunto de **lugares de interés**, en este caso las iglesias (points).

|  |
| --- |
| place\_name = "Chorrillos, Lima, Peru"  tags = {'amenity': 'place\_of\_worship', 'religion': 'christian'} # Tags para buscar iglesias  graph = ox.graph\_from\_place(place\_name, network\_type='drive')  points= ox.features\_from\_place(place\_name, tags) |

Exponga con claridad los criterios que ha usado para seleccionar los lugares de interés, alternativamente puede usar otros tipos de ubicaciones como: centros comerciales, farmacias, colegios, etc.

## **CALCULAR LA DISTANCIA ENTRE UBICACIONES**

Con esta finalidad, debe de recuperar la latitud y longitud de cada ubicación

|  |
| --- |
| nodes, edges = ox.graph\_to\_gdfs(G)  print(nodes) |

Luego proceder a calcular la distancia entre pares de ellos, observar que la distancia debe ser entre ubicaciones adyacentes y a través de las líneas, no es la distancia geográfica.

Presente evidencias de este cálculo, en particular en vías que tiene posibilidad de hacer vueltas en U.

## **PREPROCESAMIENTO DEL GRAFO**

Ejecutar algunas tareas de preprocesamiento del grafo, de tal manera que facilite la ejecución de agente de búsqueda (opcional), por ejemplo:

1. Eliminar las **rutas ciegas** (aquellas que no tiene opción de retorno)
2. Eliminar las **ubicaciones aisladas**, aquellos que no tienen vecinos.

|  |
| --- |
| import networkx as nx  G = nx.gnm\_random\_graph(20, 15)  isolated\_nodes = list(nx.isolates(G))  print("Nodos aislados:", isolated\_nodes) |

1. Eliminar **componentes conectados** (ubicaciones conectadas entre ellas, pero que no se conectan con otras)

|  |
| --- |
| C = = list(nx.connected\_components(G))  C.sort(key=len, reverse=True)  M = largest\_components = C[:n] |

1. Tratar las **rutas cerradas**, aquellas que se originan en una ubicación y terminan en la misma ubicación.
2. Otros, que sean necesarios.

Con esta finalidad puede hacer uso de la librería networkX.

## **PRESENTAR ESTADÍSTICAS.**

Presentar estadísticas que permitan tener alguna idea abstracta del distrito, tales como:

* Número de ubicaciones.
* Número de bordes
* Histograma del grado de cada ubicación (el grado es el número de vecinos de un nodo dado)
* Estimación inicial de la distancia más corta.
* Otros

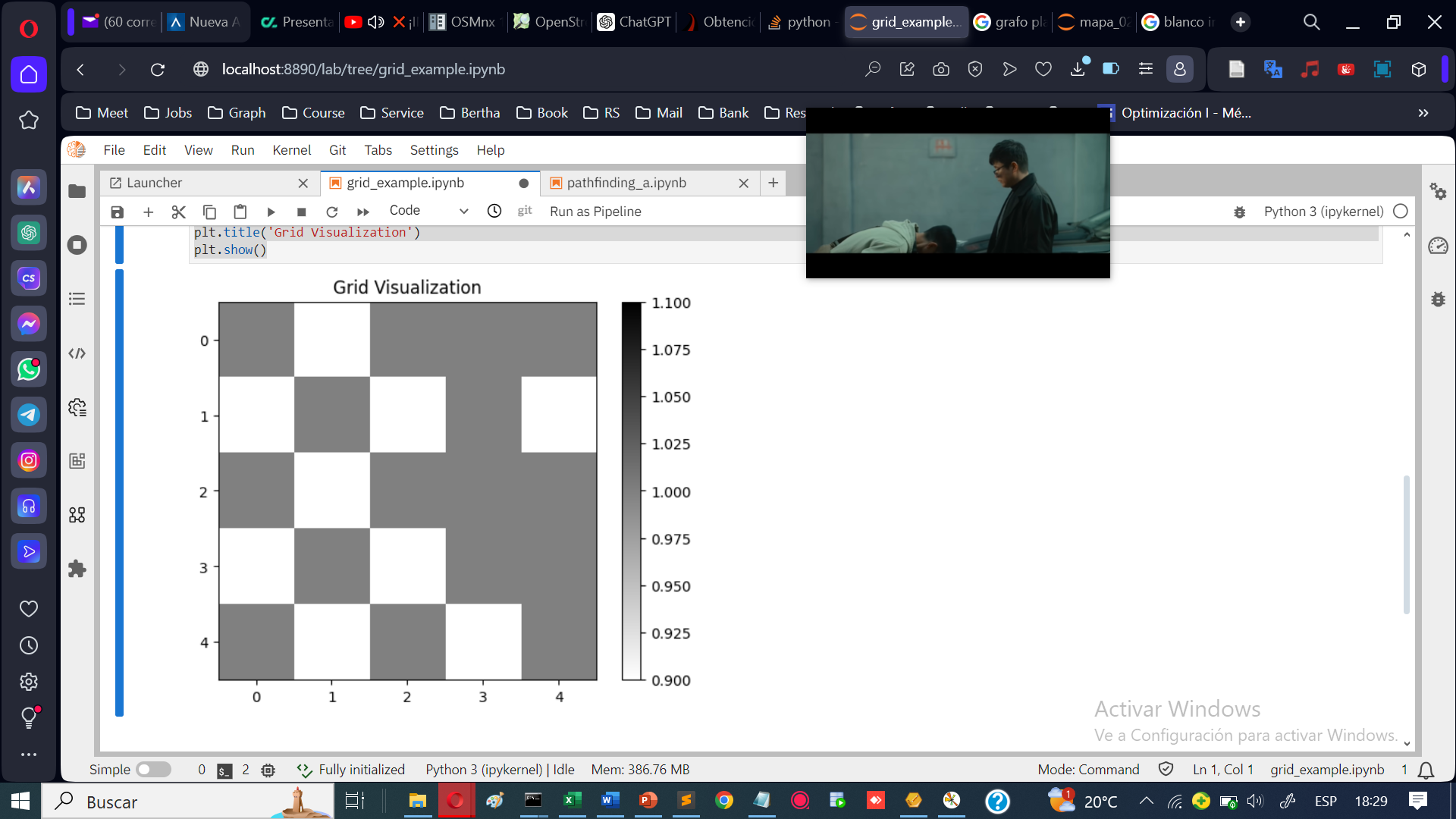
Opcionalmente:

* Radio del grafo (la menor distancia que existe entre las distancias más cortas entre dos ubicaciones)
* Diámetro del grafo (la mayor distancia que existe entre las distancias más cortas entre dos ubicaciones)

## **REPRESENTACIÓN DE LA ZONA DE BÚSQUEDA**

El grupo **puede** transformar las ubicaciones dadas en un esquema en **grilla** (no es indispensable).

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  from pathfinding.core.grid import Grid  # Crear una matriz de ejemplo  matrix = [[3, 0, 1, 0, 1],[0, 2, 0, 4, 0],[1, 0, 3, 0, 1],[0, 3, 0, 1, 0],[1, 0, 3, 0, 5]]  # Crear el objeto Grid  grid = Grid(matrix=matrix)  # Convertir la matriz de nodos a un array numpy  grid\_array = np.array([[node.walkable for node in row] for row in grid.nodes], dtype=np.float64)  grid\_array[grid\_array == 0] = np.nan  # Visualizar la cuadrícula con matplotlib  plt.imshow(grid\_array, cmap='binary', interpolation='nearest')  plt.colorbar()  plt.title('Grid Visualization')  plt.show() |



## **DEFINIR AL MENOS CUATRO ALGORITMOS DE BÚSQUEDA EN ESPACIO DE ESTADOS.**

El grupo hace uso de diversos algoritmos de búsqueda en espacio de estados para resolver el problema.

**Búsquedas no informadas (ciegas)**

* Búsqueda en amplitud (Breadth-First Search, BFS)
* Búsqueda en profundidad (Depth-First Search, DFS)
* Búsqueda con profundidad limitada (Depth-Limited Search)
* Búsqueda en profundidad iterativa (Iterative Deepening Depth-First Search, IDDFS)
* Búsqueda bidireccional (Bidirectional Search)

**Búsquedas informadas (heurísticas)**

* Búsqueda Voraz (Greedy Best-First Search)
* Algoritmo A\* (A-Star)
* Búsqueda de costo uniforme (Uniform Cost Search)

Con esta finalidad, puede hacer uso de librería orientadas a la búsqueda en espacio de estados.

Opcionalmente el grupo puede hacer uso de la librería **pathfinding**, para implementar estos algoritmos.

|  |
| --- |
| from pathfinding.core.grid import Grid  from pathfinding.finder.a\_star import AStarFinder  matrix = [  [1, 1, 0, 0, 0, 0, 0, 0, 0, 0],  [0, 1, 1, 1, 1, 1, 0, 0, 0, 0],  [0, 0, 0, 0, 0, 1, 0, 1, 1, 1],  [0, 1, 1, 1, 0, 1, 0, 1, 0, 0],  [0, 0, 0, 1, 0, 1, 0, 1, 0, 0],  [0, 1, 0, 1, 0, 1, 0, 1, 0, 0],  [0, 1, 0, 1, 1, 1, 0, 0, 0, 0],  [0, 0, 0, 1, 0, 0, 0, 1, 1, 0],  [0, 1, 0, 1, 1, 1, 1, 1, 1, 0],  [0, 0, 0, 0, 0, 0, 0, 0, 1, 1]  ]  grid = Grid(matrix=matrix)  start = grid.node(0, 0)  end = grid.node(9, 9)  finder = AStarFinder()  path, runs = finder.find\_path(start, end, grid)  # Mostrar el resultado  for (x, y) in path: print(x,y)  print("Pasos:", len(path))  print("Operaciones:", runs) |

Observar el formato que requiere este algoritmo para hacer los cálculos.

Explique el formato entregado por la librería **osmnx** y el requerido por la librería **pathfinding**, indique el procedimiento de conversión.

En todos los casos, el grupo debe de describir la forma como la librería representa cada estado del problema.

# **EXPERIMENTACIÓN**

Presente ejemplos de las corridas del procedimiento elaborado, de tal forma que se pueda verificar que su procedimiento entrega el resultado esperado.

Redacte su informe como si usted estuviese describiendo el procedimiento elaborado, paso a paso.

## **ANÁLISIS DE RESULTADOS**

Presente los resultados y las pruebas de las corridas que ha ejecutado para el problema planteado.

Calcule la medida de rendimiento del procedimiento elaborado.

Observe que la medida de rendimiento contiene la **respuesta ideal**, es decir el mejor tiempo, indique ¿cómo ha logrado calcular este valor?

Estime el tiempo que le tomaría a un **feligrés** recorrer todas las ciudades usando una estrategia ávara (diga la velocidad del caminante, el tiempo de descanso u otros parámetros que considere).

Estime el tiempo que le tomaría a su computador buscar en todo el espacio de estados, dado la complejidad del algoritmo (O()) y la velocidad de su computador (K).

**Tiempo = K \* O()**

Estime el tiempo que toma su procedimiento resolver el problema.

Finalmente, ¿los **feligrese** lograran absolver sus pecados?

¿El tiempo de cálculo es linealmente dependiente del número de ubicaciones del distrito?, dado esta respuesta, ¿es posible que los **feligreses** no logren absolver sus pecados?

# **EL DOCUMENTO**

El documento por entregar debe tener al menos la siguiente estructura.

1. Título Autores, email, universidad, curso
2. Fecha
3. Resumen
4. Palabras claves
5. Introducción al problema, antecedentes, procedimiento, resultados esperados.
6. Planteamiento del problema, exponer la sistematización de los conceptos, la ontología, red semántica.
7. Metodología de desarrollo del proyecto. Especifique de qué forma es qué resolverá el problema, indicar los pasos o componentes. En esta sección solo se presenta el método, no se desarrolla.
8. Construcción de los componentes de tal forma que le permita lograr cada uno de sus objetivos.

Apoye la explicación de cada componente, con el código fuente respectivo y el resultado de la corrida respectiva, explique que ha hecho, no solo presente el resultado.

Observe que todo procedimiento tiene una parte declarativa y una parte procedural.

1. Conclusiones, toda conclusión debe estar planteada como una métrica, no se trata de lanzar opiniones. Concluya con respecto a cada componente y respecto al objetivo general.

Concluya en qué medida el agente de búsqueda elaborado logra resolver el problema.

1. Recomendaciones, prepare al menos 3 recomendaciones, luego implemente esas recomendaciones y diga si su propuesta es viable.
2. Referencias, solo las leídas, toda referencia debe tener su respectivo DOI, ISBN o Patente, no se acepta documentos de otra naturaleza, colocar el documento solo si hace uso de la referencia (al menos 1 referencia).

* Todas las tablas, figuras y fórmulas deben estar numeradas.
* Documento a doble columna, letra de tamaño: 10 u 11.
* El código fuente debe ser leíble y colocado en su integridad como un anexo. Coloque en su documento del examen final subsecciones del código fuente para explicar el procedimiento, así como muestras del resultado obtenido.
* Las imágenes colocadas deben ser leíbles.
* Se recomienda la redacción de este documento en formato Latex.
* Coloque este documento como parte del examen, en la primera parte.
* Coloque todos los resultados de su examen en un directorio de Google drive y envíelo como resultado final. Ordene de forma legible cada elemento de su examen final.