

Contenido

[1. Explicación de la clase variable 3](#_Toc170286901)

[2. Explicación restricciones 4](#_Toc170286902)

[3. Explicación casillas fijas 6](#_Toc170286903)

[4. Explicación detallada de algoritmos 7](#_Toc170286904)

[5. Explicación formal de un problema pequeño 13](#_Toc170286905)

[Especificación Formal 14](#_Toc170286906)

[6. Grafo de restricciones 15](#_Toc170286907)

[7. Traza problema pequeño FC 16](#_Toc170286908)

[8. Traza problema AC3 19](#_Toc170286909)

[9. Sección experimentación 21](#_Toc170286910)

[10. Estudio tiempos 24](#_Toc170286911)

[Código para Generar la Gráfica 25](#_Toc170286912)

[Pasos para Ejecutar el Código 26](#_Toc170286913)

# Explicación de la clase variable

La clase Variable se utiliza para representar una variable del crucigrama, que corresponde a una palabra que debe ser colocada en el tablero. A continuación, se detallan los principales componentes de la clase Variable:

#### Atributos

* coorInicio y coorFin: Estas variables almacenan las coordenadas de inicio y fin de la variable en el tablero, definidas como tuplas (fila, columna).
* nombre: Un identificador único para la variable, que en este caso es de tipo Int.
* tam: La longitud de la variable, calculada a partir de las coordenadas de inicio y fin.
* dominio: Una lista que contiene las palabras posibles (del dominio) que pueden ser asignadas a esta variable.
* palabra: La palabra asignada actualmente a esta variable.
* restriccion: Una lista de restricciones que aplican a esta variable.
* borradas y borradasFC: Listas utilizadas para almacenar temporalmente las palabras que se eliminan del dominio durante el proceso de asignación y verificación (forward checking).

#### Métodos

* \_\_init\_\_(self, coorInicio, coorFin, nombre): Constructor de la clase, inicializa los atributos con los valores proporcionados y calcula la longitud de la variable.
* horizontal(self): Determina si la variable está orientada horizontalmente.
* longitud(self): Calcula y retorna la longitud de la variable.
* getDominio(self), setDominio(self, newDominio): Métodos para acceder y modificar el dominio de la variable.
* getNombre(self), setNombre(self, newNombre): Métodos para acceder y modificar el nombre de la variable.
* getRestriccion(self), setRestriccion(self, newRestriccion): Métodos para acceder y modificar las restricciones de la variable.
* getPalabra(self), setPalabra(self, palabra): Métodos para acceder y modificar la palabra asignada a la variable.
* getCoorIni(self), getCoorFin(self): Métodos para acceder a las coordenadas de inicio y fin.
* getBorradas(self), setBorradas(self, borradas): Métodos para acceder y modificar la lista de palabras borradas del dominio.
* getBorradasFC(self): Método para acceder a la lista de palabras borradas durante el forward checking.

# Explicación restricciones

las restricciones se implementan mediante la clase Restriccion. Las restricciones son esenciales para asegurar que las palabras del crucigrama se coloquen de manera coherente, respetando las reglas y evitando conflictos. A continuación, se detalla cómo se gestionan estas restricciones:

#### Clase Restriccion

La clase Restriccion está diseñada para representar las restricciones entre variables del crucigrama. Esta clase incluye atributos y métodos necesarios para definir y manejar estas restricciones.

##### Atributos

* varY: Representa la variable con la cual se tiene la restricción.
* coor: Almacena las coordenadas relativas que definen la relación entre la variable actual y varY.

##### Métodos

* \_\_init\_\_(self, varY, coor): Constructor de la clase que inicializa los atributos con los valores proporcionados.
* getY(self): Devuelve la variable varY.
* setY(self, varY): Establece una nueva variable varY.

#### Integración de Restricciones en la Clase Variable

* **Atributo restriccion**: Lista de objetos Restriccion que representan las restricciones con otras variables.
* **Métodos**:
  + getRestriccion(): Devuelve la lista de restricciones.
  + setRestriccion(newRestriccion): Establece una nueva lista de restricciones.

#### Creación de Restricciones en Forward Checking (FC)

1. **Inicialización de Variables**:
   * Durante el proceso de inicialización, se identifican todas las variables horizontales y verticales en el tablero.
   * Para cada variable horizontal, se recorren las coordenadas que ocupa en el tablero.
2. **Identificación de Intersecciones**:
   * Para cada posición ocupada por una variable horizontal, se verifica si hay una variable vertical que también ocupe esa posición.
   * Si se encuentra una intersección, se crea una restricción entre las dos variables.
3. **Creación de la Restricción**:
   * La restricción se representa mediante un objeto Restriccion que almacena la variable intersectada y las coordenadas relativas de la intersección.
   * Esta restricción se añade a la lista de restricciones de la variable.

#### Creación de Restricciones en AC3

1. **Inicialización de Variables**:
   * Similar al proceso en FC, se identifican todas las variables horizontales y verticales en el tablero.
2. **Identificación de Intersecciones**:
   * Para cada variable, se recorren las posiciones que ocupa en el tablero y se verifica si hay una variable intersectante en esas posiciones.
   * Cuando se encuentra una intersección, se crea una restricción entre las dos variables involucradas.
3. **Creación de la Restricción**:
   * Se crea un objeto Restriccion que contiene la variable intersectada y las coordenadas relativas de la intersección.
   * Esta restricción se añade a la lista de restricciones de la variable.
4. **Inicialización de la Cola de Restricciones**:
   * Todas las restricciones creadas se añaden a una cola que será utilizada por el algoritmo AC3 para asegurar la consistencia arc-consistente (AC).
5. **Proceso de Revisión de AC3**:
   * El algoritmo AC3 utiliza la cola de restricciones para revisar y ajustar los dominios de las variables.
   * Se eliminan palabras de los dominios que no satisfacen las restricciones, asegurando que solo se consideren asignaciones consistentes.

# Explicación casillas fijas

1. **Lectura de Letras Fijas**:

* Durante la inicialización, las letras fijas se leen y se colocan en las celdas correspondientes del tablero.
* Estas letras fijas se mantienen inmutables durante todo el proceso de asignación de palabras.

1. **Creación de Variables**:

* Al crear las variables que representan las palabras a colocar en el crucigrama, se tienen en cuenta las casillas con letras fijas.

1. **Filtrado de Dominios**:

* Las letras fijas se utilizan para filtrar el dominio de posibles palabras para cada variable.
* Durante este filtrado, se eliminan del dominio todas las palabras que no coinciden con las letras fijas en las posiciones correspondientes.

**Implementacion en cada algoritmo:**

**Forward Checking (FC)**:

* Durante la ejecución del algoritmo FC, cada vez que se asigna una palabra a una variable, se verifica que esta palabra respete todas las letras fijas en las celdas correspondientes.
* Si una palabra no respeta las letras fijas, se elimina del dominio de la variable antes de proceder con la asignación.

**AC3**:

* Similarmente, el algoritmo AC3 utiliza las letras fijas para asegurar que las restricciones se mantengan consistentes.
* Las palabras que no cumplen con las letras fijas se eliminan del dominio de las variables durante el proceso de revisión y ajuste de dominios.

# Explicación detallada de algoritmos

**Forward Checking:**

Forward Checking se basa en la idea de que, al asignar un valor (palabra) a una variable (posición en el crucigrama), se deben eliminar del dominio de las variables futuras todos aquellos valores que no sean consistentes con esta asignación. Esto permite detectar inconsistencias tempranas y reducir el espacio de búsqueda.

1. **Función start(almacen, tablero, varHor, varVer)**

Esta función inicializa el proceso de Forward Checking. Se encarga de configurar los dominios y restricciones iniciales y luego invoca la función FC para comenzar el proceso de asignación de palabras.

* **Parámetros**:
  + almacen: El almacén de palabras disponibles.
  + tablero: El tablero del crucigrama.
  + varHor: Lista de variables horizontales.
  + varVer: Lista de variables verticales.
* **Proceso**:
  + Se inicializan los dominios de las variables horizontales y verticales con las palabras correspondientes del almacén.
  + Se actualizan los dominios según las letras preinsertadas en el tablero.
  + Se crean las restricciones entre variables basadas en las intersecciones.
  + Se llama a la función FC para comenzar el proceso de asignación de palabras.
  + Si se encuentra una solución, se actualiza el tablero con las palabras asignadas.

1. **Función FC(variables, aux, tablero)**

Esta es la función principal del algoritmo Forward Checking, que realiza la asignación de palabras a las variables del crucigrama.

**Parámetros**:

* + variables: Lista de variables pendientes de asignación.
  + aux: Índice auxiliar para el backtracking.
  + tablero: El tablero del crucigrama.

**Proceso**:

* + Si no quedan variables por asignar, se retorna True indicando que se ha encontrado una solución.
  + Se selecciona la primera variable de la lista y se intenta asignar una palabra de su dominio.
  + Para cada palabra del dominio:
    - Se asigna la palabra a la variable.
    - Se realiza el forward checking.
    - Si no hay inconsistencias, se llama recursivamente a FC para las variables restantes.
    - Si se encuentra una inconsistencia, se restaura el dominio y se prueba con otra palabra.
  + Si ninguna palabra es consistente, se realiza el backtracking y se retorna False.

1. **Función forward(var, tablero)**

Esta función realiza el forward checking para una variable específica. Verifica y actualiza los dominios de las variables futuras basándose en la palabra asignada a la variable actual.

* **Parámetros**:
  + var: La variable a la que se ha asignado una palabra.
  + tablero: El tablero del crucigrama.
* **Proceso**:
  + Para cada restricción de la variable actual, se copia el dominio de la variable futura asociada.
  + Se eliminan las palabras del dominio futuro que no son consistentes con la palabra asignada.
  + Si algún dominio futuro queda vacío, se retorna False indicando una inconsistencia

1. **Función restaura(var)**

Esta función restaura el dominio de las variables afectadas por una asignación fallida, devolviendo los valores eliminados previamente.

* **Parámetros**:
  + var: La variable cuyo dominio debe ser restaurado.
* **Proceso**:
  + Para cada restricción de la variable actual, se restauran las palabras eliminadas del dominio de la variable futura asociada.
  + La palabra asignada a la variable actual se desasigna.

**Algoritmo AC3:**

AC3 se utiliza para mantener la consistencia de los dominios de las variables a medida que se hacen asignaciones. El objetivo de AC3 es iterativamente reducir los dominios de las variables eliminando valores que no pueden ser parte de una solución, asegurando así que todas las variables sean arc-consistentes

1. **Función start\_ac3(almacen, tablero, varHor, varVer)**

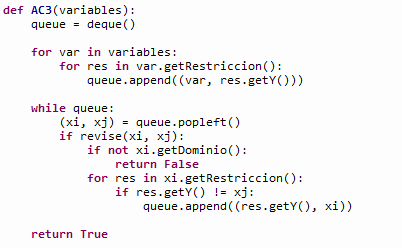
Texto

Descripción generada automáticamente Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Esta función inicializa el proceso de AC3. Configura los dominios y restricciones iniciales de las variables y luego invoca la función AC3 para comenzar el proceso de aseguramiento de la consistencia.

* + **Proceso**:
    - Inicializa los dominios de las variables horizontales y verticales con las palabras correspondientes del almacén.
    - Actualiza los dominios según las letras preinsertadas en el tablero.
    - Crea las restricciones entre variables basadas en las intersecciones.
    - Llama a la función AC3 para reducir los dominios y asegurar la consistencia de los arcos.

1. **Función AC3(variables)**

Esta es la función principal del algoritmo AC3, que revisa y ajusta los dominios de las variables para asegurar la consistencia de los arcos.

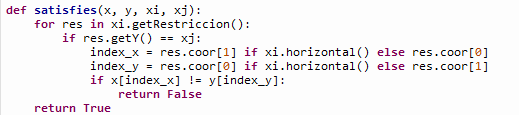
* + **Proceso**:
    - Inicializa una cola con todos los arcos (pares de variables con restricciones entre ellas).
    - Mientras la cola no esté vacía, toma un arco (xi, xj) de la cola.
    - Llama a la función revise para revisar el arco y ajustar el dominio de xi.
    - Si se elimina algún valor del dominio de xi, se añaden los arcos afectados de nuevo a la cola para ser revisados.

1. Imagen que contiene Texto

   Descripción generada automáticamente**Función revise(xi, xj)**

Esta función revisa un arco específico entre dos variables xi y xj. Ajusta el dominio de xi eliminando los valores que no tienen soporte en xj.

* + **Proceso**:
    - Para cada valor en el dominio de xi, verifica si existe un valor en el dominio de xj que satisfaga la restricción entre xi y xj.
    - Si un valor en el dominio de xi no tiene soporte en xj, se elimina del dominio de xi.
    - Retorna True si se eliminó algún valor del dominio de xi, indicando que el dominio fue revisado y ajustado.

1. **Función satisfies(x, y, xi, xj)**

Esta función verifica si dos valores x y y satisfacen la restricción entre dos variables xi y xj.

* + **Proceso**:
    - Calcula los índices relativos de las coordenadas donde las variables xi y xj se intersectan.
    - Verifica si los valores en estas posiciones específicas son iguales.
    - Retorna True si los valores satisfacen la restricción, y False en caso contrario.

1. **Función `print\_domains(variables, message="")**

Esta función imprime los dominios de las variables con un mensaje opcional.

* + **Proceso**:
    - Recorre todas las variables y sus dominios, imprimiéndolos junto con el mensaje proporcionado.

#### Funcionamiento del Algoritmo AC3

1. **Inicialización con start\_ac3**:
   * Se inicializan los dominios de las variables horizontales y verticales con las palabras del almacén.
   * Se actualizan los dominios según las letras preinsertadas en el tablero.
   * Se crean las restricciones basadas en las intersecciones entre variables.
   * Se invoca la función AC3 para asegurar la consistencia de los arcos.
2. **Consistencia de Arcos con AC3**:
   * Se inicializa una cola con todos los arcos (pares de variables con restricciones entre ellas).
   * Se revisa iterativamente cada arco, ajustando los dominios de las variables.
   * Si se elimina algún valor de un dominio, los arcos afectados se revisan nuevamente.
3. **Revisión de Arcos con revise**:
   * Se verifica cada valor en el dominio de una variable para asegurarse de que tiene soporte en la variable relacionada.
   * Los valores sin soporte se eliminan, ajustando el dominio de la variable.
4. **Verificación de Restricciones con satisfies**:
   * Se comprueba si los valores en las posiciones específicas de intersección satisfacen la restricción entre dos variables.
5. **Impresión de Dominios con print\_domains**:
   * Se imprimen los dominios de las variables antes y después de aplicar AC3, mostrando cómo se han ajustado.

# Explicación formal de un problema pequeño

Para especificar formalmente un problema de crucigrama utilizando una tupla ⟨V,E,c,l,a⟩\langle V, E, c, l, a \rangle⟨V,E,c,l,a⟩, donde:

* VVV: Conjunto de variables.
* EEE: Conjunto de restricciones (arcos) entre variables.
* ccc: Función de consistencia que valida una asignación parcial.
* lll: Función que define el dominio de cada variable.
* aaa: Algoritmo a aplicar.

#### Diccionario de Palabras

Vamos a utilizar el siguiente diccionario de palabras pequeñas:

* "SO"
* "OS"
* "LO"
* "OL"

#### Tablero y Variables

Vamos a definir un tablero pequeño de 2x2 celdas y un conjunto de 4 variables. El tablero y las variables se definen de la siguiente manera:

1. **Variable H1 (Horizontal)**
   * Coordenadas: (0,0) a (0,1)
   * Longitud: 2
   * Nombre: H1
   * Posibles palabras: "SO", "LO"
2. **Variable H2 (Horizontal)**
   * Coordenadas: (1,0) a (1,1)
   * Longitud: 2
   * Nombre: H2
   * Posibles palabras: "SO", "LO"
3. **Variable V1 (Vertical)**
   * Coordenadas: (0,0) a (1,0)
   * Longitud: 2
   * Nombre: V1
   * Posibles palabras: "SO", "OS"
4. **Variable V2 (Vertical)**
   * Coordenadas: (0,1) a (1,1)
   * Longitud: 2
   * Nombre: V2
   * Posibles palabras: "SO", "OS"

### Especificación Formal

#### Conjunto de Variables VVV

V={H1,H2,V1,V2}V = \{ \text{H1}, \text{H2}, \text{V1}, \text{V2} \}V={H1,H2,V1,V2}

#### Conjunto de Restricciones EEE

Las restricciones EEE se definen como pares de variables que comparten al menos una celda en el tablero. Las restricciones son las siguientes:

* (H1,V1)(\text{H1}, \text{V1})(H1,V1): H1 y V1 se intersectan en la celda (0,0).
* (H1,V2)(\text{H1}, \text{V2})(H1,V2): H1 y V2 se intersectan en la celda (0,1).
* (H2,V1)(\text{H2}, \text{V1})(H2,V1): H2 y V1 se intersectan en la celda (1,0).
* (H2,V2)(\text{H2}, \text{V2})(H2,V2): H2 y V2 se intersectan en la celda (1,1).

#### Función de Consistencia (diria de borrarlo)

La función de consistencia ccc valida que, para cada par de variables (X,Y)∈E(X, Y) \in E(X,Y)∈E que comparten una celda, los valores asignados a XXX y YYY son consistentes en dicha celda. Es decir, si la variable XXX y YYY se intersectan en la posición (i,j)(i, j)(i,j), entonces el iii-ésimo carácter del valor asignado a XXX debe ser igual al jjj-ésimo carácter del valor asignado a YYY.

#### Función de Dominio lll

La función de dominio lll asigna a cada variable un conjunto de palabras posibles de acuerdo a su longitud.

* l(H1)={"SO","LO"}l(\text{H1}) = \{ \text{"SO"}, \text{"LO"} \}l(H1)={"SO","LO"}
* l(H2)={"SO","LO"}l(\text{H2}) = \{ \text{"SO"}, \text{"LO"} \}l(H2)={"SO","LO"}
* l(V1)={"SO","OS"}l(\text{V1}) = \{ \text{"SO"}, \text{"OS"} \}l(V1)={"SO","OS"}
* l(V2)={"SO","OS"}l(\text{V2}) = \{ \text{"SO"}, \text{"OS"} \}l(V2)={"SO","OS"}

# Grafo de restricciones

Imagen que contiene Diagrama

Descripción generada automáticamente

# Traza problema pequeño FC

Para el problema pequeño he utilizado este diccionario:  
la es as zu mi le

Donde este es el resultado correcto:

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Dejo la traza:

FC

Inicializando Forward Checking

Se añadió una restricción entre 0 y 2 en (0, 0)

Se añadió una restricción entre 0 y 3 en (0, 1)

Se añadió una restricción entre 1 y 2 en (1, 0)

Se añadió una restricción entre 1 y 3 en (1, 1)

Se añadió una restricción entre 2 y 0 en (0, 0)

Se añadió una restricción entre 2 y 1 en (1, 0)

Se añadió una restricción entre 3 y 0 en (0, 1)

Se añadió una restricción entre 3 y 1 en (1, 1)

Iniciando Forward Checking

Variable 0 dominio inicial: ['LA', 'ES', 'AS', 'ZU', 'MI', 'LE']

Asignando la palabra LA a la variable 0

Forward checking para la variable: 0 ((0, 0) -> (0, 1))

Dominio inicial para 2 ((0, 0) -> (1, 0)): ['LA', 'ES', 'AS', 'ZU', 'MI', 'LE']

Dominio actualizado para 2 después de eliminar ES: ['LA', 'AS', 'ZU', 'MI', 'LE']

Dominio actualizado para 2 después de eliminar AS: ['LA', 'ZU', 'MI', 'LE']

Dominio actualizado para 2 después de eliminar ZU: ['LA', 'MI', 'LE']

Dominio actualizado para 2 después de eliminar MI: ['LA', 'LE']

Dominio inicial para 3 ((0, 1) -> (1, 1)): ['LA', 'ES', 'AS', 'ZU', 'MI', 'LE']

Dominio actualizado para 3 después de eliminar LA: ['ES', 'AS', 'ZU', 'MI', 'LE']

Dominio actualizado para 3 después de eliminar ES: ['AS', 'ZU', 'MI', 'LE']

Dominio actualizado para 3 después de eliminar ZU: ['AS', 'MI', 'LE']

Dominio actualizado para 3 después de eliminar MI: ['AS', 'LE']

Dominio actualizado para 3 después de eliminar LE: ['AS']

Iniciando Forward Checking

Variable 1 dominio inicial: ['LA', 'ES', 'AS', 'ZU', 'MI', 'LE']

Asignando la palabra LA a la variable 1

Forward checking para la variable: 1 ((1, 0) -> (1, 1))

Dominio inicial para 2 ((0, 0) -> (1, 0)): ['LA', 'LE']

Dominio actualizado para 2 después de eliminar LA: ['LE']

Dominio actualizado para 2 después de eliminar LE: []

El dominio de la variable 2 está vacío después del forward checking.

Restaurando variable: 1

Dominio restaurado para 2: ['LA']

Dominio restaurado para 2: ['LA', 'LE']

Dominio actualizado después de la eliminación: ['ES', 'AS', 'ZU', 'MI', 'LE']

Asignando la palabra ES a la variable 1

Forward checking para la variable: 1 ((1, 0) -> (1, 1))

Dominio inicial para 2 ((0, 0) -> (1, 0)): ['LA', 'LE']

Dominio actualizado para 2 después de eliminar LA: ['LE']

Dominio inicial para 3 ((0, 1) -> (1, 1)): ['AS']

Iniciando Forward Checking

Variable 2 dominio inicial: ['LE']

Asignando la palabra LE a la variable 2

Forward checking para la variable: 2 ((0, 0) -> (1, 0))

Dominio inicial para 0 ((0, 0) -> (0, 1)): ['LA', 'ES', 'AS', 'ZU', 'MI', 'LE']

Dominio actualizado para 0 después de eliminar ES: ['LA', 'AS', 'ZU', 'MI', 'LE']

Dominio actualizado para 0 después de eliminar AS: ['LA', 'ZU', 'MI', 'LE']

Dominio actualizado para 0 después de eliminar ZU: ['LA', 'MI', 'LE']

Dominio actualizado para 0 después de eliminar MI: ['LA', 'LE']

Dominio inicial para 1 ((1, 0) -> (1, 1)): ['ES', 'AS', 'ZU', 'MI', 'LE']

Dominio actualizado para 1 después de eliminar AS: ['ES', 'ZU', 'MI', 'LE']

Dominio actualizado para 1 después de eliminar ZU: ['ES', 'MI', 'LE']

Dominio actualizado para 1 después de eliminar MI: ['ES', 'LE']

Dominio actualizado para 1 después de eliminar LE: ['ES']

Iniciando Forward Checking

Variable 3 dominio inicial: ['AS']

Asignando la palabra AS a la variable 3

Forward checking para la variable: 3 ((0, 1) -> (1, 1))

Dominio inicial para 0 ((0, 0) -> (0, 1)): ['LA', 'LE']

Dominio actualizado para 0 después de eliminar LE: ['LA']

Dominio inicial para 1 ((1, 0) -> (1, 1)): ['ES']

Iniciando Forward Checking

FORWARD CHECKING True

# Traza problema AC3

Para el mismo caso que antes dejo la traza al pulsar AC3:

>>> %Run main.py

pygame 2.5.2 (SDL 2.28.3, Python 3.10.11)

Hello from the pygame community. https://www.pygame.org/contribute.html

AC3

Inicializando AC3

Se añadió una restricción entre 0 y 2 en (0, 0)

Se añadió una restricción entre 0 y 3 en (0, 1)

Se añadió una restricción entre 1 y 2 en (1, 0)

Se añadió una restricción entre 1 y 3 en (1, 1)

Se añadió una restricción entre 2 y 0 en (0, 0)

Se añadió una restricción entre 2 y 1 en (1, 0)

Se añadió una restricción entre 3 y 0 en (0, 1)

Se añadió una restricción entre 3 y 1 en (1, 1)

Dominios antes de AC3:

Variable 0 ((0, 0) -> (0, 1)): ['LA', 'ES', 'AS', 'ZU', 'MI', 'LE']

Variable 1 ((1, 0) -> (1, 1)): ['LA', 'ES', 'AS', 'ZU', 'MI', 'LE']

Variable 2 ((0, 0) -> (1, 0)): ['LA', 'ES', 'AS', 'ZU', 'MI', 'LE']

Variable 3 ((0, 1) -> (1, 1)): ['LA', 'ES', 'AS', 'ZU', 'MI', 'LE']

Eliminado ES del dominio de 0 porque no hay valor consistente en 3.

Eliminado AS del dominio de 0 porque no hay valor consistente en 3.

Eliminado ZU del dominio de 0 porque no hay valor consistente en 3.

Eliminado MI del dominio de 0 porque no hay valor consistente en 3.

Eliminado LA del dominio de 1 porque no hay valor consistente en 2.

Eliminado ZU del dominio de 1 porque no hay valor consistente en 2.

Eliminado MI del dominio de 1 porque no hay valor consistente en 2.

Eliminado LE del dominio de 1 porque no hay valor consistente en 2.

Eliminado ES del dominio de 2 porque no hay valor consistente en 0.

Eliminado AS del dominio de 2 porque no hay valor consistente en 0.

Eliminado ZU del dominio de 2 porque no hay valor consistente en 0.

Eliminado MI del dominio de 2 porque no hay valor consistente en 0.

Eliminado LA del dominio de 3 porque no hay valor consistente en 0.

Eliminado ZU del dominio de 3 porque no hay valor consistente en 0.

Eliminado MI del dominio de 3 porque no hay valor consistente en 0.

Eliminado LE del dominio de 3 porque no hay valor consistente en 0.

Dominios después de AC3:

Variable 0 ((0, 0) -> (0, 1)): ['LA', 'LE']

Variable 1 ((1, 0) -> (1, 1)): ['ES', 'AS']

Variable 2 ((0, 0) -> (1, 0)): ['LA', 'LE']

Variable 3 ((0, 1) -> (1, 1)): ['ES', 'AS']

AC3 completado con éxito.

# Sección experimentación

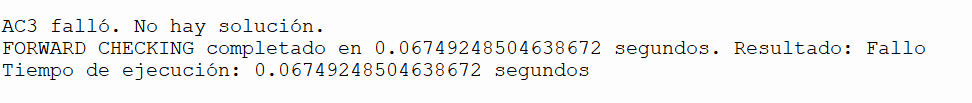
#### Prueba 1: Crucigrama Pequeño (3x3)

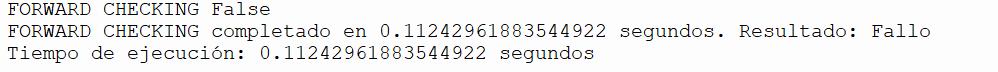
* **Objetivo:** Verificar la funcionalidad básica de los algoritmos en un crucigrama pequeño.
* **Descripción:** Un crucigrama de 3x3 con palabras de 2 y 3 letras. Se incluyen letras preinsertadas para comprobar la consistencia.
* **Resultados Esperados:**
  + **Caso 1:** Sin solución determinada por AC3 ni solucion con FC.
  + **Caso 2:** Tanto AC3 como FC encuentran una solución.

##### Calendario Descripción generada automáticamente con confianza mediaCaso 1: Sin solución determinada por AC3 ni solucion con FC

* La letra "Z" y "A" no permiten formar palabras válidas.

**Resultado:**





##### Caso 2: Crucigrama 3x3 con Solución Encontrada por Ambos Algoritmos

* Tanto AC3 como FC deben encontrar una solución válida.

**Resultado:**

Texto, Carta

Descripción generada automáticamente

Imagen que contiene naranja, hombre, cuarto, sostener

Descripción generada automáticamente

#### Prueba 2: Crucigrama Pequeño (4x4)

* **Objetivo:** Verificar la funcionalidad básica de los algoritmos en un crucigrama pequeño.
* **Descripción:** Un crucigrama de 3x3 con palabras de 2 y 3 letras. Se incluyen letras preinsertadas para comprobar la consistencia.
* **Resultados Esperados:**
  + **Caso 1:** Sin solución determinada por AC3 ni solucion con FC.
  + **Caso 2:** Tanto AC3 como FC encuentran una solución.

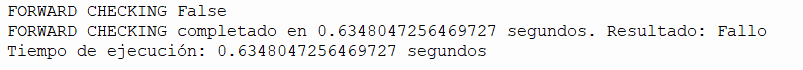
##### Calendario Descripción generada automáticamenteCaso 1: Sin solución determinada por AC3 ni solucion con FC

* La letra "Z" y "A" no permiten formar palabras válidas.

**Resultado:**

**Texto

Descripción generada automáticamente**

****

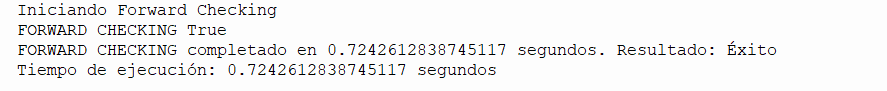
##### Caso 2: Crucigrama 3x3 con Solución Encontrada por Ambos Algoritmos

* Tanto AC3 como FC deben encontrar una solución válida.

**Resultado:**

Texto

Descripción generada automáticamente



# Estudio tiempos

Para realizar la comparativa de los tiempos de las distintas pruebas, primero he procedido a quitar todo tipo de impresiones por consola.

**Resultados:**

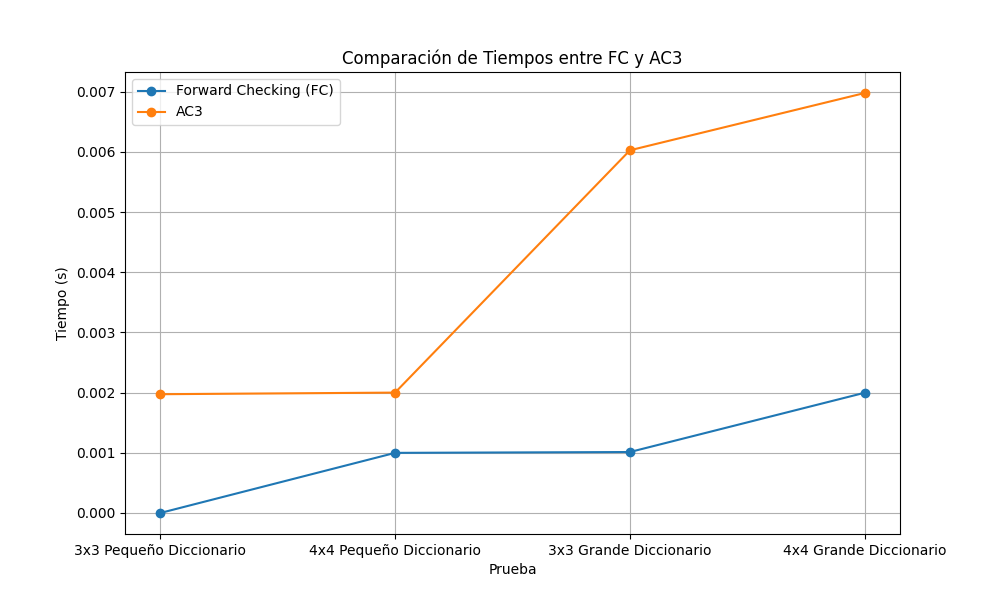
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba | Tamaño | Tiempo FC | Tiempo AC3 | Consistente |
| Prueba 1 | 3x3 | 0.0s | 0.00197362s | Si |
| Prueba 2 | 4x4 | 0.00099921s | 0.00200057029s | Si |

Se puede observar que a medida que el tamaño del tablero crece como tambien se necesitan mas palabras en el diccionario el tiempo tambien aumenta. Aunque la diferencia es minima.

Aquí hay otra tabla con muchas mas palabras en el diccionario que en la anterior.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba | Tamaño | Tiempo FC | Tiempo AC3 | Consistente |
| Prueba 1 | 3x3 | 0.00101327s | 0.00602650s | Si |
| Prueba 2 | 4x4 | 0.001998424s | 0.0069785s | Si |

De esta tabla observamos que tambien el numero de palabras en el diccionario aumenta el tiempo considerablemente

Aquí presento una grafica comparativa de los tiempos:  


BIBLIOGRAFIA   
  
Chatgpt:  
Prompt :

Chat por favor fame la grafica de los tiempos.

Para realizar la comparativa de los tiempos de las distintas pruebas, primero he procedido a quitar todo tipo de impresiones por consola.

**Resultados:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba | Tamaño | Tiempo FC | Tiempo AC3 | Consistente |
| Prueba 1 | 3x3 | 0.0s | 0.00197362s | Si |
| Prueba 2 | 4x4 | 0.00099921s | 0.00200057029s | Si |

Se puede observar que a medida que el tamaño del tablero crece como tambien se necesitan mas palabras en el diccionario el tiempo tambien aumenta. Aunque la diferencia es minima.

Aquí hay otra tabla con muchas mas palabras en el diccionario que en la anterior.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba | Tamaño | Tiempo FC | Tiempo AC3 | Consistente |
| Prueba 1 | 3x3 | 0.00101327s | 0.00602650s | Si |
| Prueba 2 | 4x4 | 0.001998424s | 0.0069785s | Si |

De esta tabla observamos que tambien el numero de palabras en el diccionario aumenta el tiempo considerablemente

Respuesta:  
  
Parece que estamos enfrentando dificultades para generar un enlace de descarga. Sin embargo, te proporcionaré el código completo que puedes ejecutar en tu entorno local para generar la gráfica de comparación de tiempos.

### Código para Generar la Gráfica

Copia el siguiente código en un archivo Python (grafica\_tiempos.py) y ejecútalo en tu entorno local. Esto generará la gráfica y la guardará como un archivo PNG.

python

Copiar código

import matplotlib.pyplot as plt

# Datos de las pruebas

pruebas = ['3x3 Pequeño Diccionario', '4x4 Pequeño Diccionario', '3x3 Grande Diccionario', '4x4 Grande Diccionario']

tiempos\_fc = [0.0, 0.00099921, 0.00101327, 0.001998424]

tiempos\_ac3 = [0.00197362, 0.00200057029, 0.00602650, 0.0069785]

# Crear la gráfica

plt.figure(figsize=(10, 6))

plt.plot(pruebas, tiempos\_fc, label='Forward Checking (FC)', marker='o')

plt.plot(pruebas, tiempos\_ac3, label='AC3', marker='o')

# Añadir etiquetas y título

plt.xlabel('Prueba')

plt.ylabel('Tiempo (s)')

plt.title('Comparación de Tiempos entre FC y AC3')

plt.legend()

plt.grid(True)

# Guardar la gráfica

plt.savefig('comparacion\_tiempos.png')

plt.show()

### Pasos para Ejecutar el Código

1. Guarda el código anterior en un archivo llamado grafica\_tiempos.py.
2. Ejecuta el archivo en tu entorno local con el comando:

bash

Copiar código

python grafica\_tiempos.py

1. El archivo comparacion\_tiempos.png se generará en el mismo directorio donde ejecutaste el script.

Este código generará una gráfica que compara los tiempos de ejecución de los algoritmos FC y AC3 para diferentes tamaños de crucigramas y diferentes tamaños de diccionarios, y guardará la gráfica en un archivo PNG. ​