**Proyecto 1**

**Karen Fuentes – 202122467**

**Alison Aristizabal – 202125179**

**Explicación del algoritmo**

El algoritmo min\_movimientos, se encarga de examinar las alturas de las torres para determinar el mínimo número de movimientos necesarios con el fin de organizar descendentemente todas las alturas dejando la posibilidad de que entre ellas se tenga la misma altura. Para lograrlo, emplea un enfoque de programación dinámica donde en un diccionario guarda los resultados que pueden ser reutilizados más adelante, disminuyendo la cantidad de cálculos hechos. Dentro de este podemos encontrar dos casos base, cuando solo se tiene una torre o cuando el numero de torres es dos ya que el algoritmo auxiliar no puede utilizarse dentro de estos casos.

Se tienen dos funciones auxiliares min\_move\_0 y min\_move\_1, en el min\_move\_0 se inicia el algoritmo de fin a principio, este toma 3 torres y compara la del centro para ver si esta es menor a la anterior en el caso de que si lo sea se mira si se puede mover alguna ficha desde la torre izquierda o si es mejor mover una ficha de la torre de la derecha, cuando la condición de que la torre sea mayor o igual que la de su derecha se cumple se llama de nuevo la función pero la posición se mueve hacia la izquierda, esto se repite hasta llegar al final de las torres. min\_move\_1 invierte el arreglo e inicia de fin a principio nuevamente, pero buscando que este se ascendente y va comparando con el mínimo obtenido con la primera función, teniendo en cuenta que esta función puede tomar mas movimientos para organizar las torres que el anterior.

Diagrama, Esquemático

Descripción generada automáticamente

Diagrama, Texto

Descripción generada automáticamente

**Análisis de complejidad espacial y temporal**

La complejidad espacial del problema es exponencial esto se debe a que memo crece con la cantidad de problemas que resuelve y estos en su mayoría son únicos.

La complejidad temporal de este problema también es exponencial específicamente O() en el peor de los casos ya que en caso de que halle otro mínimo de casos este va a volver a realizar toda las comparaciones, pero en el caso de que se llegue a superar el mínimo antes calculado la función se detiene.

ESCENARIO 1: En un movimiento se puede mover más de una ficha.

Nuevos retos:

* Este nuevo escenario aumenta la complejidad del problema, ya que ahora se deben considerar múltiples opciones de movimiento en cada iteración.
* Se requiere una estrategia distinta para determinar la cantidad mínima de fichas que se deben mover en cada movimiento, considerando que mover más de una ficha puede generar diferentes configuraciones y posibles soluciones.

Cambios en la solución:

* La función min\_move deberá modificarse para permitir mover más de una ficha en cada iteración. Esto requerirá una revisión completa del enfoque utilizado para explorar las opciones de movimiento y calcular el mínimo número de movimientos necesarios.
* Las condiciones de parada y las condiciones de borde también deberán modificarse para tener en cuenta el movimiento de múltiples fichas en una sola iteración.

ESCENARIO 2: Se pueden mover fichas entre torres no necesariamente contiguas, pero sí dentro de un radio de r torres.

Nuevos retos:

* Este escenario amplía la flexibilidad del movimiento de fichas, lo que puede generar un espacio de búsqueda de movimiento mucho más grande.
* El algoritmo ahora debe considerar las torres dentro de un radio de r torres alrededor de la torre actual, lo que aumenta la complejidad de la búsqueda de soluciones como se tiene planteado en nuestro algoritmo.

Cambios en la solución:

* La función min\_move deberá modificarse para considerar las torres dentro de un radio r de la torre actual al determinar las opciones de movimiento.
* Se necesitarán cambios en la implementación de la función de cálculo del mínimo número de movimientos para asegurar que solo se consideren las torres dentro del radio especificado.