Demuestra la admisibilidad o no admisibilidad de las dos heurísticas mencionadas anteriormente y comenta cual debería ser teóricamente mejor entre ellas (solo si ambas son admisibles) y porqué. NOTA: Para demostrar la admisibilidad, se espera una demostración maten ática.

Para demostrar que una heurística es admisible, debemos demostrar que nunca sobreestima el costo real para llegar desde un estado dado hasta el estado objetivo. Veamos la admisibilidad de las heurísticas wagdy_heuristic y repeated_color_heuristic:

wagdy_heuristic: Esta heurística asigna un costo estimado de 2 por cada par consecutivo de bolas de diferente color en cada tubo. Supongamos que tenemos un estado en el que hay un par consecutivo de bolas de diferente color en un tubo. En este caso, la heurística asigna un costo de 2, que es igual o mayor que el costo real necesario para corregir ese par. Siempre se necesita al menos un movimiento para intercambiar las bolas y colocarlas en el orden correcto. Por lo tanto, wagdy_heuristic nunca sobreestima el costo real y es admisible.

repeated_color_heuristic: Esta heurística asigna un costo estimado de 1 por cada bola que no tenga el mismo color que el color más repetido en cada tubo. Supongamos que tenemos un estado en el que hay una bola que no tiene el mismo color que el color más repetido en un tubo. En este caso, la heurística asigna un costo de 1, que es igual o mayor que el costo real necesario para colocar esa bola en el color correcto. Siempre se requiere al menos un movimiento para intercambiar esa bola con otra del color correcto. Por lo tanto, repeated_color_heuristic nunca sobreestima el costo real y es admisible.

Ambas heurísticas, wagdy_heuristic y repeated_color_heuristic, son admisibles, ya que nunca sobreestiman el costo real necesario para alcanzar el estado objetivo desde un estado dado. En cuanto a cuál de ellas es teóricamente mejor, depende del dominio específico del problema y de las características de los estados y acciones involucradas, pero para este problema quizás sea mejor wagdy_heuristic, ya que puede llegar a ser más eficiente.

Demuestra que cuando A* es usado con una heurística consistente entonces la secuencia de valores f de los estados que son extraídos de open es monótonamente creciente. Dicho de otra forma, si justo después de extraer un nodo n de Open, hicieras un print de f(n) notarias que la secuencia de valores impresos es no decreciente. Pista: demuestra primero que, si t es hijo de s, entonces siempre ocurriría que f(s) ≤ f(t). Usando eso, argumenta que el teorema es verdadero

Si t es hijo de s, entonces siempre ocurre que f(s) ≤ f(t):

La función de evaluación f(n) para un nodo n en A* se define como f(n) = g(n) + h(n), donde g(n) es el costo acumulado del camino desde el nodo inicial hasta n, y h(n) es la heurística estimada desde n hasta el objetivo.

Sea s un nodo padre y t un nodo hijo. El costo acumulado del camino desde el nodo inicial hasta t es mayor o igual que el costo acumulado del camino desde el nodo inicial hasta s, ya que t es un hijo de s y debe haber un costo asociado a la transición de s a t.

Formalmente, $g(t) \geq g(s)$.

Como la heurística es consistente, $h(t) \leq h(s)$, lo que implica que la estimación de la heurística desde t hasta el objetivo es menor o igual que la estimación de la heurística desde s hasta el objetivo.

Por lo tanto, $f(t) = g(t) + h(t) \geq g(s) + h(s) = f(s)$, lo que demuestra que $f(s) \leq f(t)$.

Argumento de la secuencia de valores f no decreciente después de extraer un nodo de open:

Supongamos que después de extraer un nodo n de open, la secuencia de valores f impresos no es no decreciente. Esto implicaría que existe al menos un nodo t que fue extraído después de n, pero $f(t) < f(n)$. Sin embargo, esto contradice la propiedad demostrada anteriormente de que si t es hijo de s, entonces $f(s) \leq f(t)$. En particular, si t fue extraído después de n, entonces t debe ser un hijo de n, lo que implica $f(n) \leq f(t)$.

Por lo tanto, la suposición de que la secuencia de valores f no es no decreciente después de extraer un nodo de open es incorrecta. Concluimos que la secuencia de valores f de los estados extraídos de open es monótonamente creciente.

En resumen, si la heurística utilizada en A* es consistente, entonces la secuencia de valores f de los estados extraídos de open es monótonamente creciente. Esto se debe a que la función de evaluación $f(s) \leq f(t)$ siempre se cumple cuando t es un hijo de s.

Demuestra que Lazy A*, tal como A*, también encuentra soluciones óptimas

A* es un algoritmo de búsqueda que utiliza una función de evaluación f(n) = g(n) + h(n), donde g(n) es el costo acumulado del camino desde el nodo inicial hasta n, y h(n) es una heurística admisible que estima el costo desde n hasta el objetivo.

La heurística admisible garantiza que h(n) nunca sobreestima el costo real desde n hasta el objetivo. A* explora los nodos en orden creciente de f(n), priorizando los nodos con menor costo estimado. Gracias a esto, A* siempre encuentra una solución óptima, es decir, una solución con el menor costo posible.

Lazy A* es una variante de A* que utiliza una heurística admisible y evita calcular la heurística para cada nodo generado hasta que es necesario. En Lazy A*, cuando se extrae un nodo de la lista open, se verifica si es un nodo objetivo antes de calcular su heurística.

Si el nodo extraído es un nodo objetivo, se devuelve como solución sin calcular su heurística.

Si el nodo extraído no es un nodo objetivo, se calcula su heurística y se generan sus sucesores.

Debido a que Lazy A* utiliza una heurística admisible y sigue el orden de exploración basado en f(n), la propiedad de optimalidad de A* se mantiene en Lazy A*. Lazy A* solo evita el cálculo anticipado de la heurística para los nodos que no son objetivo, pero cuando se necesita calcular la heurística para un nodo objetivo, se realiza y se verifica si es una solución óptima. Por lo tanto, Lazy A* también encuentra soluciones óptimas, ya que sigue los principios de A* y solo retrasa el cálculo de la heurística hasta que es necesario.

En resumen, Lazy A* también encuentra soluciones óptimas debido a que es una variante de A* que utiliza una heurística admisible y mantiene la propiedad de optimalidad al seguir el orden de exploración basado en f(n).