Proyecto Final del espacio académico Análisis de algoritmos 1.
Presentado por: Diego Armando Calero Galvis.

Análisis del problema: Nos encontramos en una situación en la cual deseamos que una maquina resuelva un juego del puzzle 15. Los únicos datos que tenemos es una matriz con el puzzle desordenado de X manera. El programa debe solucionar cualquier puzzle en un tiempo razonable. Existen diversas formas para llegar de un estado inicial A hasta un estado Final B, por ejemplo: búsqueda primero en anchura, búsqueda primero en profundidad, búsqueda primero el mejor (Heurística), etc. Claramente para este problema una búsqueda primero el mejor debería ser más rápida para llegar a la solución. Al querer aplicar este tipo de búsqueda nos encontramos con otro problema, y es que debemos verificar cada estado con cada uno de los estados anteriormente verificados y es por esto que para el peor de los casos la búsqueda por heurística tendría el mismo costo computacional que un backtraking o una búsqueda primero en anchura, además de este problema la implementación de la búsqueda primero el mejor es más compleja de entender e implementar. Por eso se decidió aplicar un backtraking que se corta en la profundidad 46 para la búsqueda de la solución, este número fue escogido de manera arbitraria.

En este algoritmo lo que movemos es el espacio vacío en lugar de pensar en mover las fichas adyacentes a este. Así nos evitamos una búsqueda de adyacencias, lo que haría más complejo el algoritmo.

Diseño de la solución: El algoritmo utilizado es recursivo, a continuación se presenta un seudocódigo.

Solucionar (p: profundidad del estado actual; sol: lista con las fichas movidas; actual: matriz del estado actual; i: coordenada en i del espacio vacío; j: coordenada en j del espacio vacío)
Crear aux: matriz auxiliar
Si: actual es solución
    Solución = sol
    Solucionado = verdadero
Si: p = 46
    Retorno vacío
Si: es válida la posición i + 1, j y solucionado = falso
    Aux = mover espacio vacío a la posición i + 1, j
    Agregar aux[i] [j] a sol
    Llamado recursivo solucionar (p + 1, sol, aux, i + 1, j)
    Eliminar aux[i] [j] de sol
Si: es válida la posición i - 1, j y solucionado = falso
    Aux = mover espacio vacío a la posición i - 1, j
    Agregar aux[i] [j] a sol
    Llamado recursivo solucionar (p + 1, sol, aux, i - 1, j)
    Eliminar aux[i] [j] de sol
Si: es válida la posición i, j + 1 y solucionado = falso
    Aux = mover espacio vacío a la posición i, j + 1
    Agregar aux[i] [j] a sol
    Llamado recursivo solucionar (p + 1, sol, aux, i, j + 1)
    Eliminar aux[i] [j] de sol
Si: es válida la posición i, j - 1 y solucionado = falso
    Aux = mover espacio vacío a la posición i, j - 1
    Agregar aux[i] [j] a sol
    Llamado recursivo solucionar (p + 1, sol, aux, i, j - 1)
    Eliminar aux[i] [j] de sol

Proyecto Final del espacio académico Análisis de algoritmos 1.
Presentado por: Diego Armando Calero Galvis.

Análisis del algoritmo: El algoritmo aplicado repite algunos estados y no tiene aplicado algún algoritmo heurístico que elimine nodos.

A partir de algunos cálculos matemáticos se llegó a la conclusión de que este algoritmo es de orden factorial O(n!) para el peor de los casos y en otros casos menos graves puede ser de orden exponencial O(exp(n)).

Se concluye que no es un buen algoritmo ya que requiere de mucho tiempo de ejecución y de recursos computacionales. Claramente se puede mejorar con un poco más de tiempo e investigación de los algoritmos heurísticos. El A* es un buen algoritmo para la solución de este problema.