INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

UNIDAD DE APRENDIZAJE: INTELIGENCIA ARTIFICIAL

CUADRO MÁGICO PRIMERO EL MEJOR

NOMBRE: FARRERA MENDEZ EMMANUEL SINAI

PROFESOR: ROMERO HERRERA RODOLFO

GRUPO: 6CM2

FECHA DE ENTREGA: 18 DE MARZO DEL 2024

Resolveremos el cuadro mágico con el algoritmo “Primero el mejor”

Y siguiendo los siguientes pasos:

1. **Definir el problema**: Comprende el problema del cuadro mágico, que consiste en llenar una cuadrícula 3x3 con números del 1 al 9, de modo que la suma de cada fila, columna y diagonal sea la misma.
2. **Estado inicial y objetivo**: Define un estado inicial, que es el cuadro vacío, y un estado objetivo, que es el cuadro mágico completo con las sumas correctas.
3. **Función de evaluación (heurística)**: Define una función de evaluación que evalúe qué tan cerca está un estado dado del estado objetivo. En el caso del cuadro mágico, la función de evaluación podría ser la diferencia absoluta entre la suma de cada fila, columna y diagonal del estado actual y la suma objetivo. Cuanto menor sea esta diferencia, mejor será el estado.
4. **Generar sucesores**: Define cómo generar los sucesores de un estado dado. En este caso, los sucesores serían los estados obtenidos al intercambiar dos números adyacentes en el cuadro.
5. **Ordenar los sucesores**: Ordena los sucesores según la función de evaluación (heurística), de modo que se explore primero el sucesor que esté más cerca del estado objetivo.
6. **Búsqueda**: Aplica el algoritmo de "primero el mejor" para explorar los estados sucesores hasta encontrar el estado objetivo.
7. **Finalización**: Una vez que se encuentra el estado objetivo, el algoritmo termina y se devuelve la secuencia de acciones (intercambios de números) que lleva del estado inicial al estado objetivo.
8. **Complejidad**: Considera la complejidad del algoritmo, que puede variar dependiendo de la eficacia de la función heurística y del número de estados generados y explorados.

Código:

```
import heapq
import itertools

def magic_sum(n):
    return n*(n**2 + 1)//2

def heuristic(perm, magicSum):
    square = [perm[n:n+3] for n in range(0, len(perm), 3)]
    sums = [sum(row) for row in square] + [sum(col) for col in zip(*square)] +
[sum(square[i][i] for i in range(3))] + [sum(square[i][2-i] for i in range(3))]
    return sum(abs(s - magicSum) for s in sums)

def best_first_search():
    magicSum = magic_sum(3)
    perms = list(itertools.permutations(range(1, 10)))
```

```
    heapq.heapify(perms)
    while perms:
        perm = heapq.heappop(perms)
        if heuristic(perm, magicSum) == 0:
            return [perm[n:n+3] for n in range(0, len(perm), 3)]

magicSquare = best_first_search()
for row in magicSquare:
    print(row)'
```

Y el código funciona de la siguiente manera:

magic_sum(n): Esta función calcula la suma mágica para un cuadro mágico de tamaño n. La suma mágica es la suma de los números en cualquier fila, columna o diagonal en un cuadro mágico.

heuristic(perm, magicSum): Esta función calcula la heurística para una permutación dada perm. La heurística es la suma de las diferencias absolutas entre la suma de cada fila, columna y diagonal y la suma mágica. Esta heurística se utiliza para determinar qué permutaciones son más prometedoras y deben ser exploradas primero.

best_first_search(): Esta es la función principal que implementa el método de búsqueda "Primero el Mejor". Primero, calcula la suma mágica para un cuadro mágico de 3x3. Luego, genera todas las permutaciones posibles de los números del 1 al 9 y las coloca en una cola de prioridad. La cola de prioridad está ordenada por la heurística de cada permutación, por lo que las permutaciones más prometedoras se exploran primero. Finalmente, extrae permutaciones de la cola de prioridad una por una hasta que encuentra una permutación que forma un cuadro mágico (es decir, la heurística de la permutación es 0).

¡Es importante tener en cuenta que este código puede tardar un tiempo considerable en ejecutarse, ya que hay 9! (362,880) permutaciones posibles de los números del 1 al 9. Además, este código no garantiza que se encuentre una solución si no existe un cuadro mágico con la suma objetivo dada.