Tarea 1 - Inteligencia Artificial

Entrega: Miércoles 26 de Abril

Nombre:

1.- (10 pts) De un ejemplo de un espacio de búsqueda (considerando factores como factor de ramificación, profundidad, ubicación del objetivo en profundidad) en el cual el método de búsqueda de profundidad iterativa tenga un peor desempeño que el de búsqueda en profundidad (básico). Puede usar un ejemplo artificial, es decir, el espacio de estados puede no corresponder a ningún problema real.

2.- Considere un problema en el cual tiene una mesa y cuatro bloques, del mismo tamaño. Cada bloque es identificado por una letra, A, B, C o D. Cada bloque puede estar en la mesa, o sobre otro bloque. Al comienzo, los bloques A, B y D están sobre la mesa, y el bloque C está sobre el bloque D. Un bloque puede moverse a la vez, sólo si no tiene otro bloque encima. Cada bloque puede moverse ya sea a la mesa o sobre otro bloque (si es que no hay nada sobre ese bloque). El objetivo es crear una torre con los bloques A, B y C, donde A es el bloque de más arriba y C el de más abajo (sobre la mesa).

2.1.- (10 pts) Describa la formulación del problema (cuáles son los estados, estado inicial, test objetivo y acciones)

2.2.- (5 pts) Proponga una heurística no trivial (no h(n)=constante). ¿Es la heurística propuesta admisible?

3.- (5 pts) ¿Es el algoritmo de Hill Climbing apropiado para el problema de misioneros y caníbales? Justifique

4.- (10 pts) Suponga que tiene dos heurísticas admisibles, h1 y h2. Ud. decide crear nuevas heurísticas de la siguiente manera:
h3(n) = max (h1(n), h2(n))
h4(n) = max (h1(n), 1.1*h2(n))
h5(n) = min(h1(n), 3*h2(n))
h6(n) = (h1(n) + h2(n))/2

Para cada una de estas nuevas heurísticas, especifique si son, o no, admisibles. Justifique su respuesta. ¿Usaría ud. alguna de estas nuevas heurísticas en lugar de h1 o h2?

5.- Programación. Debe adjuntar un enlace a repositorio github con su código fuente documentado e instrucciones para su ejecución (se descontará puntaje de no contar con una documentación adecuada y/o instrucciones). Su código debe ser creado en C, C++ o Python, debe ser original (se comparará con el de sus compañeros y de otros repositorios), y no se

pueden utilizar bibliotecas más allá de las que facilitan el uso de listas o arrays (por ej., numpy en Python). Cualquier evidencia de copia implicará una nota 1.0 en la evaluación.

5.1.- (25 pts) Implemente un programa (en C, C++ o Python) que, utilizando una estructura de búsqueda tipo árbol, encuentre una solución para el problema de encontrar una ruta entre A y H, diagramado en la Figura 1. Debe utilizar los siguientes métodos:

- Búsqueda en profundidad (escogiendo un sucesor al azar)
- Búsqueda por costo uniforme
- Búsqueda greedy
- A*

Para cada tipo de búsqueda, su código debe retornar:
- El camino encontrado y su costo (y si es la solución óptima, que ud. puede calcular a mano)
- Cantidad de nodos expandidos (veces por nodo y en total)

Su código debe leer el problema (grafo) desde un archivo .txt con el siguiente formato:
Init: <nodo_inicial>
Goal: <nodo_objetivo>
<Nodo1> <valor_heuristica1>
<Nodo2> <valor_heurística2>
…..
<Nodo1>, <nodo2>, <costo> // en el caso en que exista una arista entre nodo1 y nodo2
…..

Su código podría ser evaluado, además, cambiando los nodos de inicio y/o objetivo. El formato de salida debe ser:
Nodo_inicial → nodo1 → nodo2 …->nodo_objetivo
Costo: <costo>
<nodo_inicial>: número de veces que se expandió
<nodo1>: número de veces que se expandió
….

5.2 .- (10 pts) Responda:
- ¿Qué puede decir de la comparación entre los métodos implementados? En concreto, los que encontraron la solución óptima, a qué se debió? Y los que no, ¿por qué no la encontraron?
- De los métodos vistos en clase, ¿hay alguno que NO retorne una solución (es decir, que se mantenga realizando búsqueda ad infinitum) en este problema? Si es así, cuál? Y si no, por qué no?

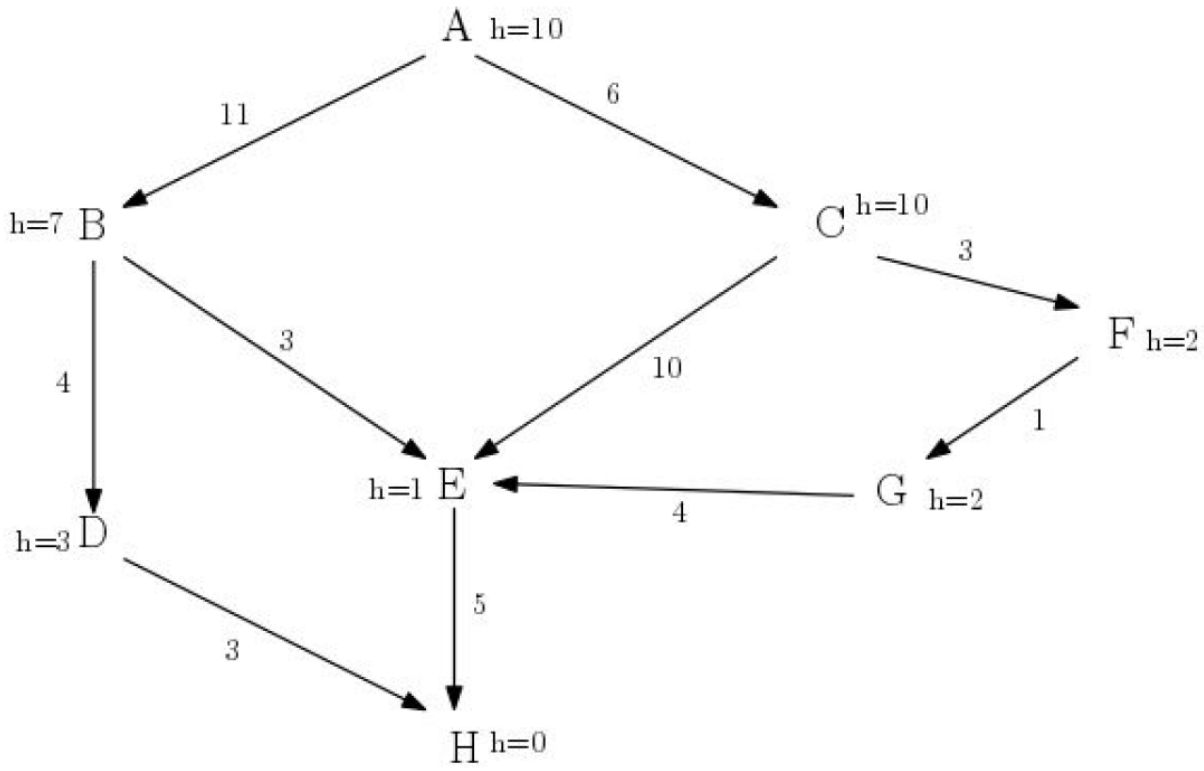

Figura 1