1.- El concepto de Agente. Agentes Racionales vs. Agentes Inteligentes. Arquitecturas de Agentes.

Un Agente es un sistema (de ordenador) situado en algún entorno, que es capaz de realizar acciones de forma autónoma y que es flexible para lograr los objetivos planteados.

Cuando hablamos de la situación nos referimos a que el agente recibe entradas sensoriales de un entorno en donde está situado y realiza acciones que cambian dicho entorno. Y en cuanto a la autonomía nos referimos a que el agente es capaz de actuar sin la intervención directa de los humanos y tiene control sobre sus propias acciones y estado interno.

Además, los agentes deben de ser capaces de interactuar, cuando sea apropiado, con otros agentes artificiales o humanos para completar su propio proceso de resolución del problema y ayudar a otros con sus actividades, siendo capaces de exhibir comportamientos dirigidos a lograr objetivos, tomando la iniciativa cuando sea apropiado y respondiendo de forma temporal a los cambios que ocurren en dicho entorno.

- **Agente Racional:** El objetivo de este tipo de agentes es maximizar los valores esperados eligiendo siempre la acción con el resultado optimo de entre todas las acciones posibles.
- **Agente Inteligente:** Los agentes inteligentes son sistemas informáticos capaces de resolver problemas, que requieran de cierto grado de inteligencia y aprendizaje, de forma autónoma como si de un humano se tratara. Es necesario que el agente supere el test de Turing para que sea considerado agente inteligente.

La arquitectura de los agentes se puede dividir en 3 según su tipología:

- **Arquitectura de 3 capas verticales con 2 pasos:**
  - Capa de comportamiento: Se encarga de todo lo relacionado con la información del entorno.
  - Capa de planificación: Es la encargada de los los planes y acciones del agente.
  - Capa de cooperación: Se encarga de los planes y acciones con otros agentes del entorno.
- **Arquitectura de 3 capas horizontales:**
  - Capa modeladora: Se encarga de generar un modelo del mundo para anticipar conflictos y generar nuevos objetivos.
  - Capa planificadora: Se encarga del comportamiento proactivo basado en esquemas de planes.
  - Capa reactiva: Se encarga de dar respuestas inmediatas ante cambios en el entorno.
- **Arquitectura Hibrida:**
  - En esta arquitectura la claridad semántica y conceptual es inferior a la de las demas arquitecturas, siendo, además, la gestión de interacciones entre capas más compleja.
  - Se utiliza una descomposición natural de la funcionalidad del agente.

2.- Características de los Agentes reactivos y deliberativos. Similitudes y diferencias. Arquitecturas.

- **Agente Reactivo:**
  Una arquitectura reactiva es aquella que no incluye ninguna clase de modelo centralizado de representación simbólica del mundo, y no hace uso de razonamiento complejo.
  Suele utilizar una arquitectura horizontal.
- **Agente Deliberativo:**
  Este agente contiene un modelo simbólico del mundo explícitamente representado, y cuyas decisiones se realizan a través de un razonamiento lógico basado en emparejamientos de patrones y manipulaciones simbólicas.
  Suele utilizar una arquitectura vertical.

Ambos agentes realizan acciones en función de lo recibido por los sensores.

3.- Describir brevemente los métodos de búsqueda no informada.

- **Búsqueda en anchura:**
  Desde el estado inicial analizar todos los sucesores de cada nodo antes de pasar al nivel siguiente en el árbol de búsqueda.
- **Búsqueda con costo:**
  Se supone que el expandir el nodo Ei con la regla Rij para obtener el nodo Ej tiene un costo Cij. De este modo cualquier nodo Ek en el árbol de búsqueda tiene un costo Ck igual a la suma de los costos de los arcos que conducen a él desde la raíz del árbol.
  El procedimiento de búsqueda expande el nodo Ep con menor Cp de todos los de la frontera.
  Si el costo de expandir el nuevo fuera constante este actuaría como la búsqueda en anchura.
- **Búsqueda primero en profundidad:**
  Desde el estado inicial ir analizando un sucesor del nodo de mayor nivel generado hasta el momento.
  Si la profundidad del árbol de estados es indefinida hay que acotarla debiendo existir una solución dentro de ese límite.
- **Búsqueda bidireccional:**
  Se ejecutan dos búsquedas simultáneas. una hacia delante, desde el estado inicial y la otra hacia atrás, desde el objetivo.
  Se para cuando ambas búsquedas se encuentren, se comprueba que el nodo expandido de un árbol esté en la frontera del otro.
- **Backtracking:**
  Durante la búsqueda, si se encuentra una alternativa incorrecta, la búsqueda retrocede hasta el paso anterior y toma la siguiente alternativa. Cuando se han terminado las posibilidades, se vuelve a la elección anterior y se toma el siguiente hijo. Si no hay más alternativas la búsqueda falla. De esta manera, se crea un árbol implícito, en el que cada nodo es un estado de la solución.

  La diferencia con la búsqueda en profundidad es que se suelen diseñar funciones de cota, de forma que no se generen algunos estados si no van a conducir a ninguna solución, o a una solución peor de la que ya se tiene. De esta forma se ahorra espacio en memoria y tiempo de ejecución.

- **Descenso Iterativo:**
  El Descenso Iterativo es un algoritmo en el que se realizan sucesivas búsquedas en profundidad limitada incrementando el límite de profundidad en cada iteración hasta alcanzar, la profundidad del estado objetivo de menor profundidad. Es equivalente a la búsqueda en anchura, pero usa mucha menos memoria; en cada iteración, visita los nodos del árbol de búsqueda en el mismo orden que una búsqueda en profundidad, pero el orden en el que los nodos son visitados finalmente se corresponde con la búsqueda en anchura.

4.- El concepto de heurística. Como se construyen las heurísticas. Uso de las heurísticas en IA.

Las Heurísticas son criterios, métodos, o principios para decidir cual, entre una serie de cauces alternativos de acción, promete ser más efectivo a la hora de lograr alguna meta. Esto representa un compromiso entre dos exigencias: la necesidad de que tales criterios sean simples y, al mismo tiempo, puedan discriminar correctamente entre buenas y malas opciones.

Generalmente las heurísticas se crean a partir de modelos simplificados del dominio del problema, aunque, en ocasiones no se puede conocer como de cerca esta la solución dada por la heurística de la solución óptima del problema.

En las IA, el método heurístico es usado en determinadas circunstancias, cuando no existe una solución óptima bajo las restricciones dadas. En general la manera de actuar de los programas heurísticos consiste en encontrar algoritmos con buenos tiempos de ejecución y buenas soluciones.

Muchos algoritmos en la inteligencia artificial son heurísticos por naturaleza, o usan reglas heurísticas. Un ejemplo claro son los programas que detectan si un correo electrónico es o no spam. Cualquiera de las reglas usadas de forma independiente puede llevar a errores de clasificación, pero cuando se unen múltiples reglas heurísticas, la solución es más robusta y creíble.

5.- Los métodos de escalada. Caracterización general. Variantes.

Si dibujamos las soluciones como puntos en el espacio, una búsqueda local consiste en seleccionar la solución mejor en el vecindario de una solución inicial, e ir viajando por las soluciones del espacio hasta encontrar un óptimo (local o global).

Los algoritmos de escalada son:

- **Algoritmo de escalada simple:**
  Considera que la vecindad de un nodo es un conjunto de hijos obtenidos secuencialmente hasta que aparece uno mejor que el padre (según la función de evaluación).
  Se parte de un nodo inicial, para cada nodo en curso se obtiene la vecindad según el criterio anterior. cuando se obtiene un hijo mejor que el padre este pasa a ser el nodo en curso y el proceso continuo hasta encontrar la solución o no se encuentre un hijo mejor que el padre.
- **Algoritmo de escalada por la máxima pendiente:**
  Considera que la vecindad de un nodo es el conjunto de todos sus hijos obtenidos secuencialmente. Se parte de un nodo inicial, para cada nodo en curso se obtiene la vecindad según el criterio anterior, se selecciona el mejor hijo el cual pasa a ser el nodo en curso y el proceso continuo hasta que se encuentre la solución o no se encuentre un hijo mejor que el padre.

Sus características generales son:

- **Completitud:** No tiene por qué encontrar la solución.
- **Admisibilidad:** No siendo completo, aun menos será admisible.
- **Eficiencia:** Rápido y útil si la función es monótona (de)creciente.

Las variantes estocásticas son necesarias para evitar caer en óptimos locales y estancamientos gracias al componente aleatorio que estas nos ofrecen. Las variantes son:

- **Algoritmo de escalada estocástico:**
  Escoge aleatoriamente entre los sucesores con mejor valoración que el estado actual
- **Algoritmo de escalada de primera opción:**
  Se generan aleatoriamente sucesores, escogiendo el primero con mejor valoración que el estado actual.
- **Algoritmo de escalada de reinicio aleatorio:**
  Se repite varias veces la búsqueda. Partiendo cada vez de un estado inicial distinto, generado aleatoriamente.
  Si la probabilidad de éxito de una búsqueda individual es p, entonces el número esperado de reinicios es 1/p.

6.- Características esenciales de los métodos "primero el mejor".

- Son variantes del algoritmo de Dijkstra en el que el algoritmo avanza a través del mejor nodo encontrado hasta el momento.
- Las decisiones que toman están basadas en la información disponible en cada momento.
- Una vez se toma una decisión esta no vuelve a replantearse.
- No garantiza una respuesta optima en el 100% de los casos.
- Usan dos listas de nodos:
  - **Nodos Abiertos:**
    Nodos que se han generado y a los que se les ha aplicado la función heurística, pero que aún no han sido explorados (es decir, no se han generado sus sucesores), ordenada según el valor de la heurística.
  - **Nodos Cerrados:**
    Lista de nodos que ya se han generado sus hijos utilizada para comprobar si el nodo ha sido generado anteriormente.
- Procedimiento Primero el Mejor:
  - SI N es Estado-final ENTONCES EXITO=Verdadero
  - Si NO expandir N, generando el conjunto S de sucesores N, que no son antecesores de N en el grafo.
  - Generar un nodo en G por cada s de S
  - Establecer un puntero a N desde aquellos s de S que no estuvieran ya en G
  - Añadirlos a ABIERTOS
  - Para cada s de S que estuviera ya en ABIERTOS o CERRADOS decidir si redirigir sus punteros de los nodos en sus subárboles.
  - Reordenar ABIERTOS según f(n)
  - Si ÉXITO entonces Solución=camino desde I a N a través de los punteros de G
  - Si no Solución=Fracaso

7.- Elementos esenciales del algoritmo A*.

Se trata de un algoritmo heurístico que mejora el algoritmo de Dijsktra, el cual se encarga de encontrar rutas más cortas dentro de un grafo. En esta modificación se toma como punto central la observación búsquedas informadas dentro del grafo que nos permitan tomar decisiones óptimas sobre los caminos que deben tomarse para recorrer de forma eficiente el grafo.

Este algoritmo deriva del procedimiento de primero el mejor, por lo que comparte las premisas de estos.

Se parte de que en cada paso se selecciona el nodo más prometedor que se haya generado hasta el momento, calculando se el valor de cada nodo de la siguiente manera.

El costo de la ruta se divide en dos partes:

- **g(n):** representa el costo de la ruta desde su origen hasta el nodo n dentro del grafo.
- **h(n):** representa el costo estimado de la ruta desde el nodo n al nodo de destino, calculado por una suposición inteligente.

Siendo la función de evaluación del nodo **f (n) = g (n) + h (n)**.

A continuación, se expande el nodo elegido generando todos sus sucesores, si alguno de ellos es el destino a alcanzar el proceso acaba, si no el algoritmo continua.

El Procedimiento de A* es bastante similar al visto en Primero el Mejor:

1. Crear un grafo de búsqueda G. Inicializar G con I.
2. Crear ABIERTOS con I, crear CERRADOS vacío.
3. Si ABIERTOS está vacía terminar con FALO.
4. Tomar primero nodo, N, de ABIERTOS e insertarlo en CERRADOS.
5. Si N=O entonces ÉXITO. Devolver N y el camino de I a N (Usando el árbol de búsqueda dado por los punteros que se construyen en el paso 7)
6. Expandir N y crear el conjunto M con todos sus sucesores. Eliminar de M aquellos nodos que sean ancestros de N en G. Añadir M a los sucesores de N en G.
7. Establecer un puntero a N desde aquellos nodos de M que no estén ni en ABIERTOS ni en CERRADOS (no han sido visitados en G). Insertar estos elementos en ABIERTOS.
   Para cada nodo de M que esté en ABIERTOS o en CERRADOS modificar el puntero para que este apunte a N siempre que el mejor camino encontrados hasta el momento hacia dicho nodo pase por N.
   Para cada nodo de M que ya esté en CERRADOS, modificar los punteros de cada uno de sus descendientes de modo que apunten hacia atrás a los mejores caminos encontrados hasta el momento a esos descendientes.
8. Reordenar ABIERTOS según f(n) de menor a mayor. En caso de empate emplear el criterio de profundidad en el árbol de búsqueda.
9. Volver al paso 3.

8.- Elementos esenciales de un algoritmo genético.

Son llamados así porque se inspiran en la evolución biológica y su base genético-molecular.
Estos algoritmos hacen evolucionar una población de individuos sometiéndola a acciones aleatorias semejantes a las que actúan en la evolución biológica (mutaciones y recombinaciones genéticas), así como también a una selección de acuerdo con algún criterio, en función del cual se decide cuáles son los individuos más adaptados, que sobreviven, y cuáles los menos aptos, que son descartados.

Los algoritmos genéticos funcionan entre el conjunto de soluciones de un problema llamado fenotipo, y el conjunto de individuos de una población natural, codificando la información de cada solución en una cadena, generalmente binaria, llamada cromosoma. Los símbolos que forman la cadena son llamados genes. Cuando la representación de los cromosomas se hace con cadenas de dígitos binarios se le conoce como genotipo. Los cromosomas evolucionan a través de iteraciones, llamadas generaciones. En cada generación, los cromosomas son evaluados usando alguna medida de aptitud.

Un algoritmo genético puede presentar diversas variaciones, dependiendo de cómo se aplican los operadores genéticos (cruzamiento, mutación), de cómo se realiza la selección y de cómo se decide el reemplazo de los individuos para formar la nueva población. En general, el pseudocódigo consiste de los siguientes pasos:

- **Inicialización:**
  Se genera aleatoriamente la población inicial, que está constituida por un conjunto de cromosomas los cuales representan las posibles soluciones del problema. En caso de no hacerlo aleatoriamente, es importante garantizar que, dentro de la población inicial, se tenga la diversidad estructural de estas soluciones para tener una representación de la mayor parte de la población posible
- **Evaluación:**
  A cada uno de los cromosomas de esta población se aplicará la función de aptitud para saber la calidad de la solución que se está codificando.
- **Condición de Termino:**
  El algoritmo genético se deberá detener cuando se alcance la solución óptima, pero esta generalmente se desconoce, por lo que se deben utilizar otros criterios de detención. Normalmente se usan dos criterios: correr el algoritmo un número máximo de iteraciones (generaciones) o detenerlo cuando no haya cambios en la población. Mientras no se cumpla la condición de término se hace lo siguiente:
  - **Selección:**
    Después de saber la aptitud de cada cromosoma se procede a elegir los cromosomas que serán cruzados en la siguiente generación. Los cromosomas con mejor aptitud tienen mayor probabilidad de ser seleccionados.
  - **Recombinación:**
    La recombinación es el principal operador genético, representa la reproducción sexual, opera sobre dos cromosomas a la vez para generar dos descendientes donde se combinan las características de ambos cromosomas padres.

- **Mutación:**
  Modifica al azar parte del cromosoma de los individuos, y permite alcanzar zonas del espacio de búsqueda que no estaban cubiertas por los individuos de la población actual.
- **Reemplazo:**
  Una vez aplicados los operadores genéticos, se seleccionan los mejores individuos para conformar la población de la generación siguiente.

Los algoritmos genéticos tienen varias limitaciones como, por ejemplo:

- Para problemas de alta complejidad la función de evaluación puede tornarse demasiado costosa en términos de tiempo y recursos.
- Puede haber casos en los cuales dependiendo los parámetros que se utilicen. para la evaluación el algoritmo podría no llegar a converger en una solución óptima.
- La "mejor" solución lo es solo en comparación a otras soluciones por lo que no se tiene demasiado claro un criterio de cuándo detenerse ya que no se cuenta con una solución específica.