# Sistemas de búsqueda y razonamiento

29/01/2024

1. Unidad 1: Introducción
   1. Problemas de búsqueda
   2. Espacios estado
   3. Teorías de búsqueda
2. Unidad 2: Técnicas no informadas
   1. Las listas de estados
   2. Técnica primero es amplitud
   3. Técnica primero es profundidad
3. Unidad 3: Técnicas informadas
   1. El valor de utilidad
   2. Búsqueda optima
   3. Técnica primero por la mejor
   4. Técnica avara
   5. Técnica A estrella (A\*)
4. Técnicas adversarias
   1. La competencia y la búsqueda
   2. Técnicas MinMax
   3. Poda α – β
5. Técnicas locales
   1. La vecindad de un estado
   2. El óptimo local
   3. Técnica por asenso de colina
   4. Técnica beam
   5. Técnica talxi
   6. Técnica por templado simulado
6. Búsqueda y resolución de problemas con restricciones
   1. Problemas con restricciones
   2. Variables y dominios
   3. Restricciones
   4. Búsqueda para resolver restricciones

Bibliografía:

Inteligencia artificial: un enfoque moderno (2da ed)  
Russell & Norvig  
Pearson

31/01/2024

**Unidad 1 = Antecedentes**

Tenemos un problema que debe ser resuelto, y ese problema tiene una representación muy especial:

1. En cualquier momento se encuentra en uno de n estados.  
   Al conjunto de n estados lo llamamos Q, y es como a continuación se indica:  
   Q = {q1, q2, … , qn}

Q = estados

A = Acciones

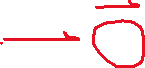
1. Estando en el estado qi, es posible cambiarse a otro estado qj usando una acción aj, para j = 1,2, … , m  
   Al conjunto con las m acciones le llamamos A, y es como a continuación se indica:  
   A = {a1, a2, … , am}
2. Y luego existe una función de transición s: Q x A -> Q (Es decir que es una función que toma un par formado por un estado qi ϵ Q y una acción aj ϵ A y regresa un estado qk ϵ Q) que indica a que estado qk cambia el problema estando actual en el estado qi y usando una acción aj
3. De todos los estados en Q uno es el estado de inicio, el estado en donde el problema inicia al “arrancar” el proceso de búsqueda. Al estado de inicio normalmente se le denota como q0 o qini

El problema siempre va a empezar en q0 . Solo existe un q0 pero puede existir más de un qmeta

1. Y de todos los estados en Q, uno o mas son estados meta, es decir estados cuyas características nos hacen considerar al problema como resuelto.  
   (qmeta c Q)  
   Así que un problema de búsqueda es una tupla como la siguiente:  
   P = {Q, A, q0, qmeta, ϭ}  
   Aquí P es el problema que se debe problema gráficamente, y por eso lo mas común es representar al problema como un grafo dirigido (un digrafo) en donde:
   1. El estado qi ϵ Q se representa como un “nodo”: qi



* 1. El estado q0 se representa como : q0



* 1. Si qi ϵ qmeta se representa como: qi



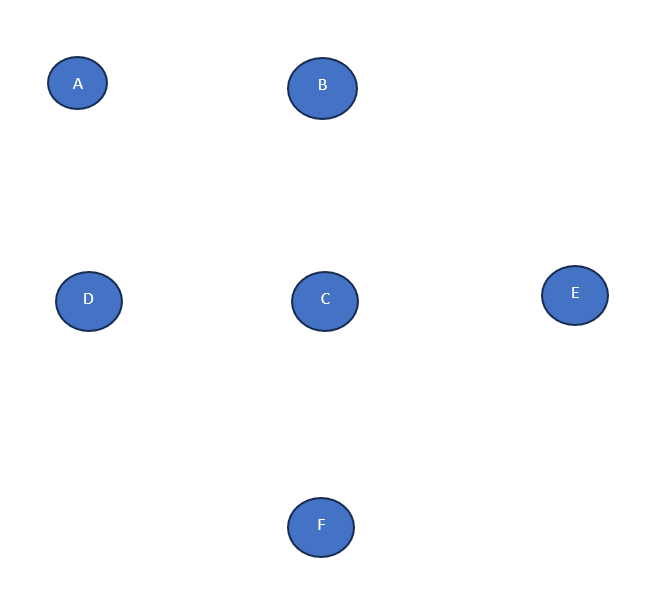
* 1. Y la transición ϭ(qi, aj) = qk se representa como una arista que sale de qi y llega a qk y se etiqueta como ak qi  ak qkEso de etiquetar la acción con su nombre  
     No siempre es necesario

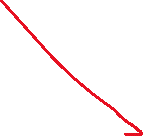
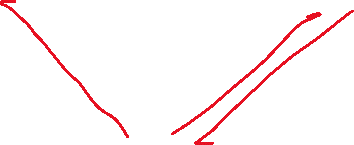
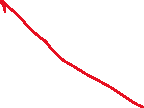
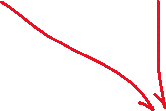
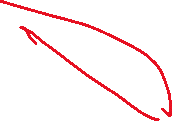


* 1. Imaginemos que tenemos un problema como el siguiente:  
     Q = {A, B, C, D, E, F}  
     A = {ir a A, ir a B, ir a C, ir a D, ir a E, ir a F}  
     ϭ ={ ϭ(A ir a B) = B ϭ(B ir a C) = C  
      ϭ(A ir a C) = C ϭ(B ir a E) = E  
      ϭ(A ir a D) = D ϭ(C ir a A) = A  
      ϭ(B ir a A) = A ϭ(C ir a B) = B  
      ϭ(C ir a D) = D ϭ(E ir a B) = B  
      ϭ(C ir a E) = E ϭ(E ir a C) = C  
      ϭ(D ir a A) = A ϭ(E ir a F) = F  
      ϭ(D ir a C) = C ϭ(F ir a D) = D  
      ϭ(D ir a F) = F ϭ(F ir a E) = E}

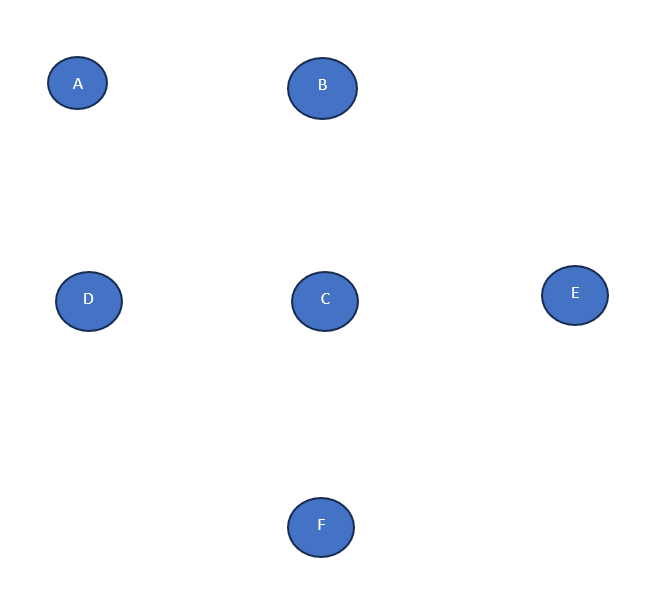
q0 = a  
qmeta = F

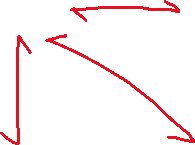
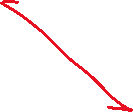
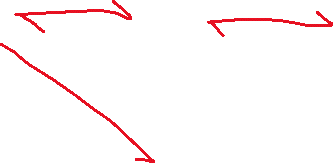
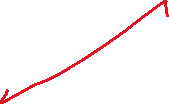
Y este es un problema muy simple y completo, aquí cada estado podría representar una ciudad, y el problema en su mismo es como un mapa. Su representación gráfica es la siguiente, tomando en cuenta como si el nombre de la acción fuera prescindible (No importa o es demasiado obvio):





Eso de que exista una acción que cambia al problema de qi y qj , y luego otra acción que hace lo contrario significa que con acciones “complementarias”, y si el nombre de la acción no importa podemos simplificar todavía más la representación gráfica usando un único arista bidireccional entre qi y qj :





A menos que se indique lo contrario, en todos los problemas que habremos de resolver en clase, esta será la representación que usaremos (Grafos dirigidos).

Por otro lado, existen algunas preguntas que seria oportuno responder, por ejemplo:

* ¿De que esta hecho un estado? Depende del problema, pero digamos que contienen una o mas “variables de estado”, que son las variables con las que se controla el proceso de búsqueda. En el ejemplo anterior la única variable de control es el nombre del estado actual.
* En el problema anterior, ¿Por qué no hay una acción “ir a F” desde A? Pues por que en este problema no existe forma de cambiar de A a F (no hay carretera directa entre A y F). Y es que no es necesario que cada estado aplique todas las acciones disponibles
* ¿Puede una acción etiquetarse con algo que no sea su nombre especifico? Si, en ocasiones una acción se etiqueta con atributos como el costo de la acción (Costo monetario), la distancia de la acción (la distancia entre los estados que conecta), etc.

Clase 07/02/2024

Unidad 2 Técnicas de búsqueda no informada

Una técnica de búsqueda, en general, parte del siguiente algoritmo:

* Sea actual = inicio
* Sea listo = False
* While listo == False:
  + If actual == meta
    - Listo = True
  + Else:
    - Actual = siguiente ()

En este algoritmo la idea es cambiar del estado etiquetado como actual (iniciando con el estado de inicio) a un nuevo estado. Para cambiar el estado etiquetado como actual a un nuevo estado usamos la función siguiente () que de alguna marera se las arregla para indicarnos a que nuevo estado del problema nos vamos a cambiar. Y esto continua hasta que en algún momento el estado actual sea el estado meta, porque allí nos detenemos.

En nuestro curso todos nuestros problemas solo tienen un estado meta.

Así que gran parte del peso para hacer que esta función se lo lleva la función siguiente (). ¿Como funciona la función siguiente? ¿Como decide que hacer para cambiar de estado? Pues todo depende de la técnica de búsqueda que se utilice. Aquí hay de dos:

* Técnicas no informadas (Técnicas ciegas)
* Técnicas informadas

En una técnica ciega a siguiente () no le queda otra que explorar todas las opciones en algún orden

Una técnica informada le proporciona a la función siguiente “información” que le permite a siguiente () preferir alguna opción para cambiar de estado antes que otras. Pero en las técnicas no informadas (También conocidas como “técnicas ciegas) a la función siguiente () no se le ofrece nada en absoluto, no tiene manera de preferir una opción de cambio antes que otras, así que:

Nótese el “… en algún orden…” en la afirmación anterior. Y es que las técnicas ciegas son exhaustivas en su proceder, es decir, si para encontrar el estado meta deben recorrer todo el espacio estado del problema (el grafo), pues entonces eso harán. Con esto en mente, técnicas de búsqueda ciega hay dos:

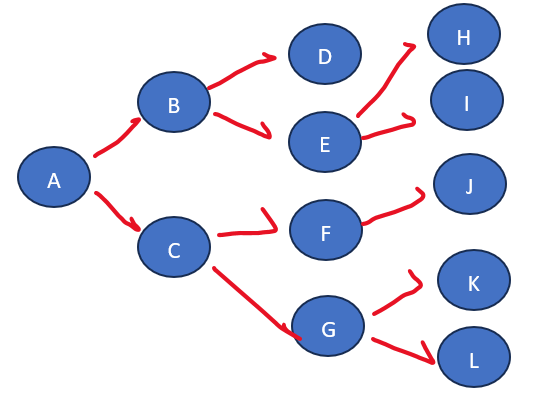
* Búsquedas ciegas primero en amplitud
* Búsquedas ciegas primero en profundidad

Cuando dijimos que las búsquedas ciegas explorar todo el espacio estado del problema siguiendo algún orden, nos referíamos a:

* Primero en amplitud: El orden es por amplitud.
* Primero en profundidad: el orden es por profundidad

¿Amplitud y profundidad? ¿Qué es eso? Eso se refiere a dos parámetros que todo grafo dirigido como los que usamos para representar nuestros problemas.

Imaginemos un problema cuyo espacio estado tiene forma de árbol, como el siguiente:



En la vida real casi ningún problema tiene forma de arbol

En todo grafo siempre nos vamos a encontrar que dos estados tienen la siguiente forma:

B es padre de E



A es padre de B



Y claro, si A es padre de B entonces B es hijo de A. Entonces digamos que dos estados son cogeneracionales si:

1. Tienen el mismo padre.
2. O el padre del primero es cogeneracional del padre del segundo.

Esto nos permite deducir que:

1. C y B son cogeneracionales (Tienen el mismo padre)
2. D y G son cogeneracionales (El padre de D es B, el padre de G es C, y B y C son cogeneracionales)
3. D y L no son cogeneracionales (El padre de D es B, el padre de L es G, pero B y G no son cogeneracionales por que el padre de B es A y el padre de G es C, y A y C no son cogeneracionales por que A es padre de C)

Todos los estados cogeneracionales entre si forman una generación o nivel en un grafo.

Esto de ser estados cogeneracionales es importante porque:

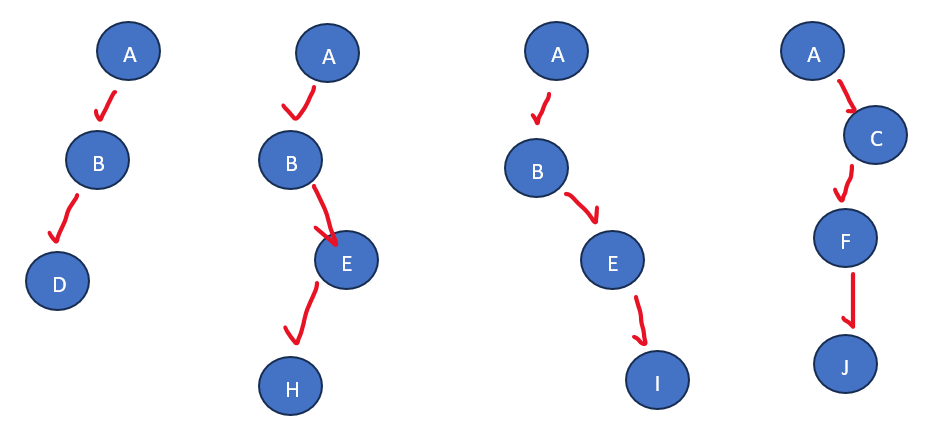
En el grafo anterior tenemos:

* Generación 0 {A}
* Generación 1 {B, C}
* Generación 2 {D, E, F, G}
* Generación 3 {H, I, J, K, L}

Pero no solo existen generaciones o niveles en un grafo, también existen “ramas”.

Una rama es el camino que se forma desde un estado y partiendo de el cambiar a uno de sus hijos, luego a uno de los hijos de este, y luego a uno de los hijos de este y así hasta que se alcanza un estado que ya no tiene hijos

Así, partiendo de A tendríamos las siguientes Ramas:



Camino: dícese de la secuencia de estados que están conectados directamente conectados uno con el siguiente.

