Concurso Programacion 3

Tecnicatura Universitaria en Inteligencia Artificial

Dámian Ariel Marotte

Section 1

Algoritmos de Búsqueda

Definicion

Un algoritmo de búsqueda es un algoritmo que toma como entrada un problema y devuelve una solución de la forma secuencia de acciones.

Para que un problema quede bien definido, necesitamos especificar los siguientes componentes:

Definicion

Un algoritmo de búsqueda es un algoritmo que toma como entrada un problema y devuelve una solución de la forma secuencia de acciones.

Para que un problema quede bien definido, necesitamos especificar los siguientes componentes:

Estado Inicial

Necesitamos definir cual es el estado a partir del cual empezamos a resolver el problema.

Definicion

Un algoritmo de búsqueda es un algoritmo que toma como entrada un problema y devuelve una solución de la forma secuencia de acciones.

Para que un problema quede bien definido, necesitamos especificar los siguientes componentes:

Estado Inicial

Necesitamos definir cual es el estado a partir del cual empezamos a resolver el problema.

Funciones sucesores

Tambien necesitamos hacer una lista de todas las posibles transofrmaciones sobre los estados, esto es, cada una de las acciones que podrian acercarnos a la solucion.

Definicion

Un algoritmo de búsqueda es un algoritmo que toma como entrada un problema y devuelve una solución de la forma secuencia de acciones.

Para que un problema quede bien definido, necesitamos especificar los siguientes componentes:

Estado Inicial

Necesitamos definir cual es el estado a partir del cual empezamos a resolver el problema.

Funciones sucesores

Tambien necesitamos hacer una lista de todas las posibles transofrmaciones sobre los estados, esto es, cada una de las acciones que podrian acercarnos a la solucion.

Test objetivo

Finalmente necesitamos una funcion que dado un estado, nos indique si se trata de una solucion al problema, o no.

Ejemplo

Un terrorista ha colocado una boma. En el lugar se encuentran disponibles un bidon con capacidad para 3 litros y otro para 5 litros. Para desactivar la bomba es necesario apoyar un bidon con exactamente 4 litros sobre una balanza.

Ejemplo

Un terrorista ha colocado una boma. En el lugar se encuentran disponibles un bidon con capacidad para 3 litros y otro para 5 litros. Para desactivar la bomba es necesario apoyar un bidon con exactamente 4 litros sobre una balanza.

Estado Inicial

Partimos de un estado donde ambos bidones se encuentran vacios.

Ejemplo

Un terrorista ha colocado una boma. En el lugar se encuentran disponibles un bidon con capacidad para 3 litros y otro para 5 litros. Para desactivar la bomba es necesario apoyar un bidon con exactamente 4 litros sobre una balanza.

Estado Inicial

Partimos de un estado donde ambos bidones se encuentran vacios.

Funciones sucesores

Las acciones que podemos realizar son:

- Vaciar el bidon de 3 litros.
- Vaciar el bidon de 5 litros.
- Llenar el bidon de 3 litros.
- Llenar el bidon de 5 litros.
- Volcar el bidon de 5 litros en el de 3.
- Volcar el bidon de 3 litros en el de 5.

Ejemplo

Un terrorista ha colocado una boma. En el lugar se encuentran disponibles un bidon con capacidad para 3 litros y otro para 5 litros. Para desactivar la bomba es necesario apoyar un bidon con exactamente 4 litros sobre una balanza.

Estado Inicial

Partimos de un estado donde ambos bidones se encuentran vacios.

Funciones sucesores

Las acciones que podemos realizar son:

- Vaciar el bidon de 3 litros.
- Vaciar el bidon de 5 litros.
- Llenar el bidon de 3 litros.
- Llenar el bidon de 5 litros.
- Volcar el bidon de 5 litros en el de 3.
- Volcar el bidon de 3 litros en el de 5.

Test objetivo

Podemos determinar si hemos llegado a la solucion observando si hay 4 litros en el bidon mas grande.

¿Como buscar?

Una vez que tenemos definido nuestro problema, podemos buscar una solucion confeccionando un grafo de busqueda y recorriendo cada uno de sus nodos.

En un grafo de busqueda cada uno de los nodos corresponde con un posible estado del problema, y cada arista es una de las acciones disponibles en el.

¿Como buscar?

Una vez que tenemos definido nuestro problema, podemos buscar una solucion confeccionando un grafo de busqueda y recorriendo cada uno de sus nodos.

En un grafo de busqueda cada uno de los nodos corresponde con un posible estado del problema, y cada arista es una de las acciones disponibles en el.

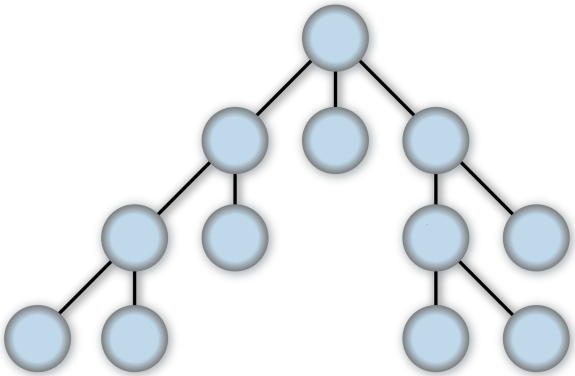

Figure 1: Grafo de busqueda

Recorrido en profundidad

La idea de un recorrido en profundidad es expandir siempre el nodo mas profundo del grafo hasta que ya no sea posible, en cuyo caso se vuelve hacia atras y se continua con el siguiente.

Recorrido en profundidad

La idea de un recorrido en profundidad es expandir siempre el nodo mas profundo del grafo hasta que ya no sea posible, en cuyo caso se vuelve hacia atras y se continua con el siguiente.

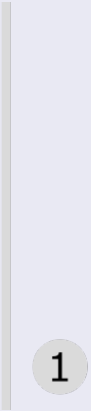

Figure 2: Empezamos con el nodo inicial

Recorrido en profundidad

La idea de un recorrido en profundidad es expandir siempre el nodo mas profundo del grafo hasta que ya no sea posible, en cuyo caso se vuelve hacia atras y se continua con el siguiente.

Figure 3: Realizamos el test objetivo

Recorrido en profundidad

La idea de un recorrido en profundidad es expandir siempre el nodo mas profundo del grafo hasta que ya no sea posible, en cuyo caso se vuelve hacia atras y se continua con el siguiente.

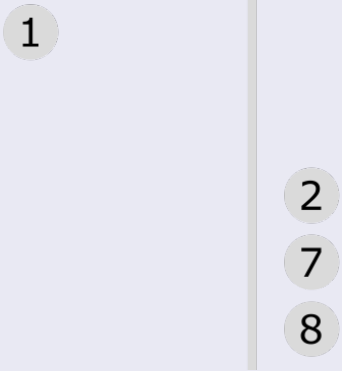

Figure 4: Generamos sus sucesores

Recorrido en profundidad

La idea de un recorrido en profundidad es expandir siempre el nodo mas profundo del grafo hasta que ya no sea posible, en cuyo caso se vuelve hacia atras y se continua con el siguiente.

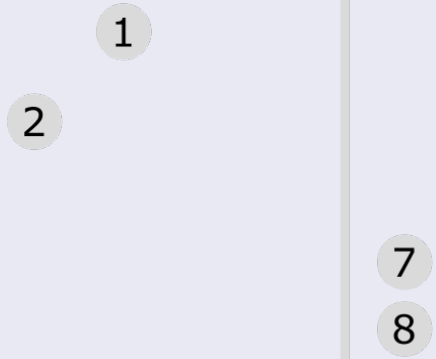

Figure 5: Consideramos el siguiente nodo

Recorrido en profundidad

La idea de un recorrido en profundidad es expandir siempre el nodo mas profundo del grafo hasta que ya no sea posible, en cuyo caso se vuelve hacia atras y se continua con el siguiente.

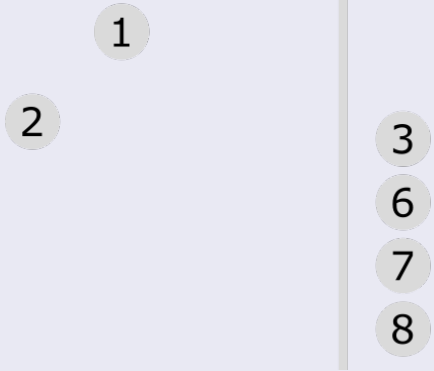

Figure 6: Generamos sus sucesores

Recorrido en profundidad

La idea de un recorrido en profundidad es expandir siempre el nodo mas profundo del grafo hasta que ya no sea posible, en cuyo caso se vuelve hacia atras y se continua con el siguiente.

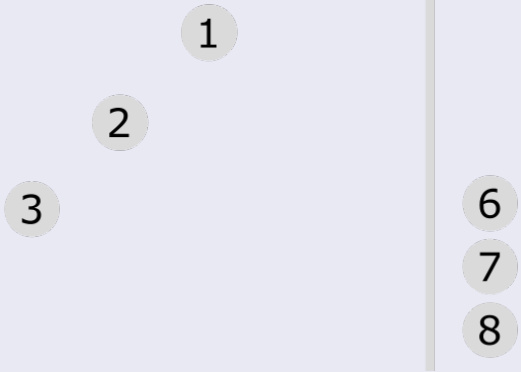

Figure 7: Consideramos el siguiente nodo

Recorrido en profundidad

La idea de un recorrido en profundidad es expandir siempre el nodo mas profundo del grafo hasta que ya no sea posible, en cuyo caso se vuelve hacia atras y se continua con el siguiente.

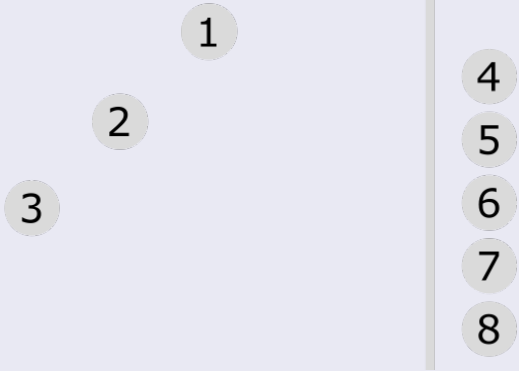

Figure 8: Generamos sus sucesores

Recorrido en profundidad

La idea de un recorrido en profundidad es expandir siempre el nodo mas profundo del grafo hasta que ya no sea posible, en cuyo caso se vuelve hacia atras y se continua con el siguiente.

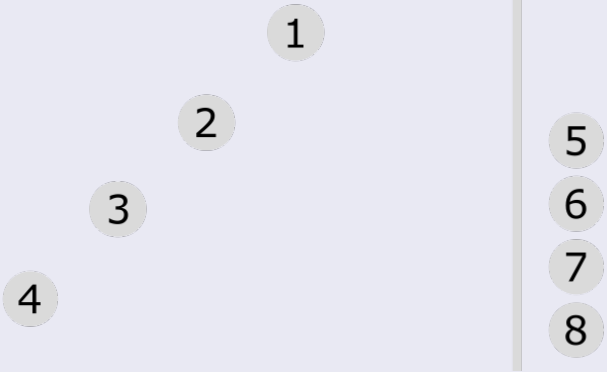

Figure 9: Consideramos el siguiente nodo

Recorrido en profundidad

La idea de un recorrido en profundidad es expandir siempre el nodo mas profundo del grafo hasta que ya no sea posible, en cuyo caso se vuelve hacia atras y se continua con el siguiente.

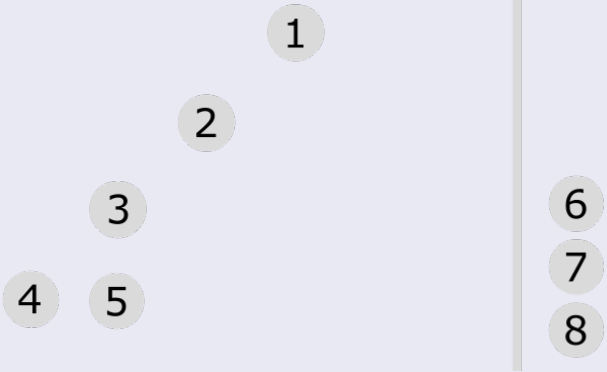

Figure 10: Generamos sus sucesores

Recorrido en profundidad

La idea de un recorrido en profundidad es expandir siempre el nodo mas profundo del grafo hasta que ya no sea posible, en cuyo caso se vuelve hacia atras y se continua con el siguiente.

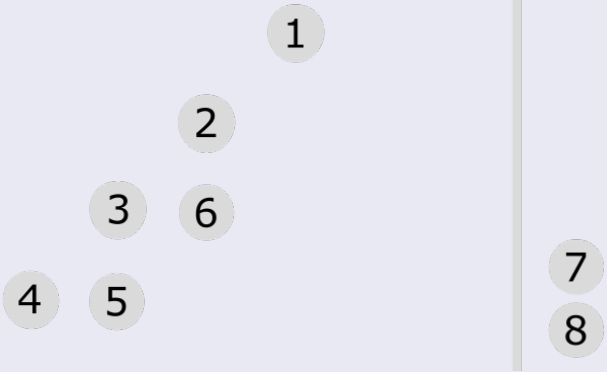

Figure 11: Consideramos el siguiente nodo

Recorrido en profundidad

La idea de un recorrido en profundidad es expandir siempre el nodo mas profundo del grafo hasta que ya no sea posible, en cuyo caso se vuelve hacia atras y se continua con el siguiente.

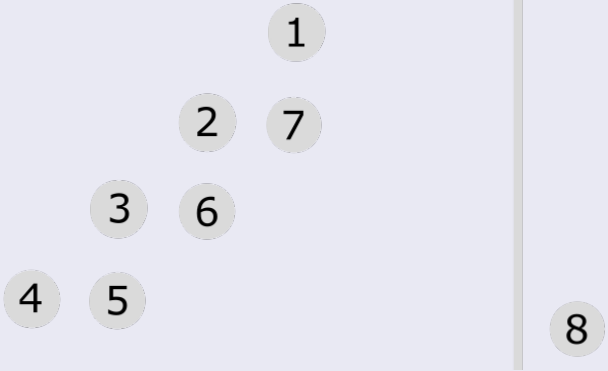

Figure 12: Generamos sus sucesores

Recorrido en profundidad

La idea de un recorrido en profundidad es expandir siempre el nodo mas profundo del grafo hasta que ya no sea posible, en cuyo caso se vuelve hacia atras y se continua con el siguiente.

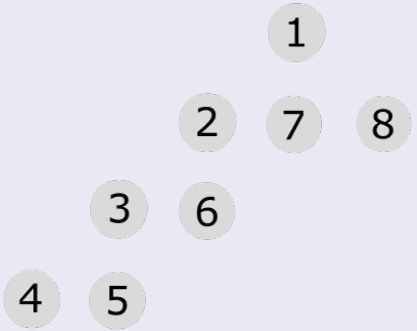

Figure 13: Consideramos el siguiente nodo

Recorrido en profundidad

La idea de un recorrido en profundidad es expandir siempre el nodo mas profundo del grafo hasta que ya no sea posible, en cuyo caso se vuelve hacia atras y se continua con el siguiente.

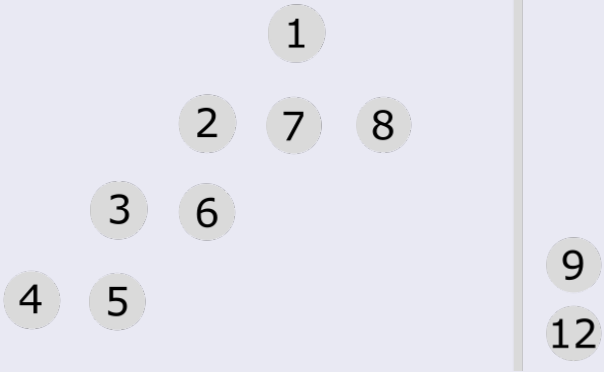

Figure 14: Generamos sus sucesores

Recorrido en profundidad

La idea de un recorrido en profundidad es expandir siempre el nodo mas profundo del grafo hasta que ya no sea posible, en cuyo caso se vuelve hacia atras y se continua con el siguiente.

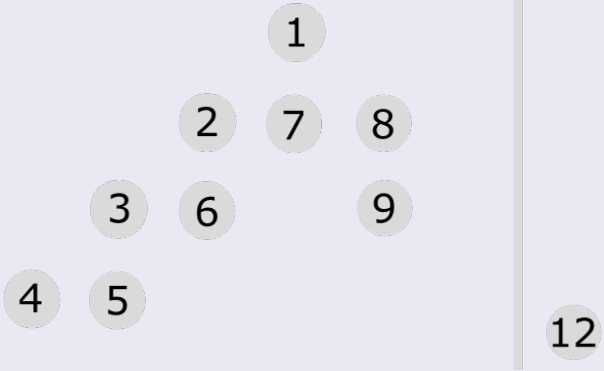

Figure 15: Consideramos el siguiente nodo

Recorrido en profundidad

La idea de un recorrido en profundidad es expandir siempre el nodo mas profundo del grafo hasta que ya no sea posible, en cuyo caso se vuelve hacia atras y se continua con el siguiente.

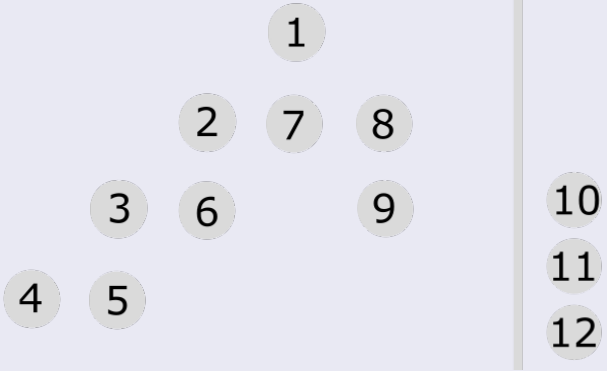

Figure 16: Generamos sus sucesores

Recorrido en profundidad

La idea de un recorrido en profundidad es expandir siempre el nodo mas profundo del grafo hasta que ya no sea posible, en cuyo caso se vuelve hacia atras y se continua con el siguiente.

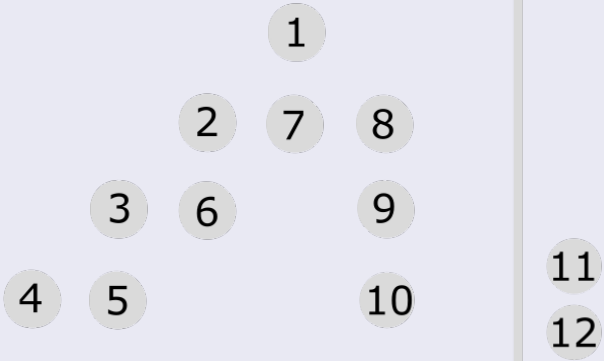

Figure 17: Consideramos el siguiente nodo

Recorrido en profundidad

La idea de un recorrido en profundidad es expandir siempre el nodo mas profundo del grafo hasta que ya no sea posible, en cuyo caso se vuelve hacia atras y se continua con el siguiente.

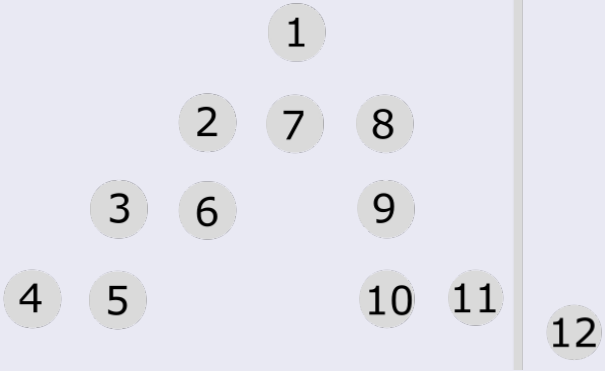

Figure 18: Generamos sus sucesores

Recorrido en profundidad

La idea de un recorrido en profundidad es expandir siempre el nodo mas profundo del grafo hasta que ya no sea posible, en cuyo caso se vuelve hacia atras y se continua con el siguiente.

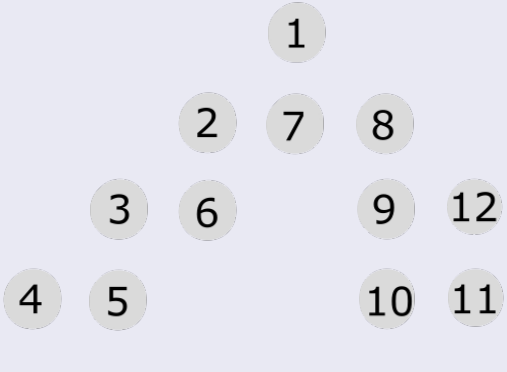

Figure 19: Consideramos el siguiente nodo

Recorrido en profundidad

La idea de un recorrido en profundidad es expandir siempre el nodo mas profundo del grafo hasta que ya no sea posible, en cuyo caso se vuelve hacia atras y se continua con el siguiente.

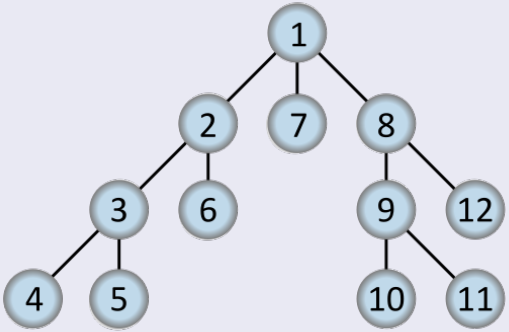

Figure 20: Recorrido en profundidad

Recorrido en anchura

La búsqueda primero en anchura es una estrategia sencilla en la que se expande primero el nodo raíz, a continuación se expanden todos los sucesores del nodo raíz, después sus sucesor, etc.

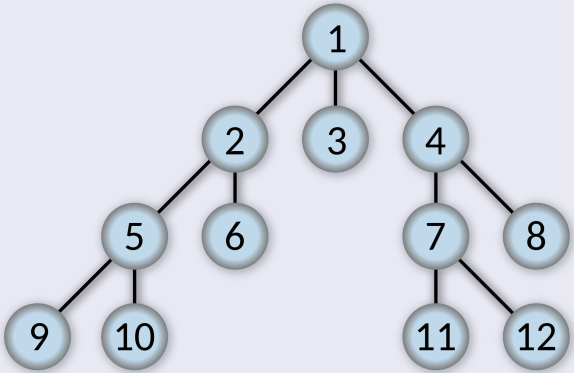

Figure 21: Recorrido en anchura

Características

- Si nuestro grafo no es un arbol, debemos tener un cuidado especial: tenemos que evitar recorrer los ciclos, pues quedaremos atrapados en un bucle infinito.

Características

- Si nuestro grafo no es un arbol, debemos tener un cuidado especial: tenemos que evitar recorrer los ciclos, pues quedaremos atrapados en un bucle infinito.
- La busqueda en profundidad ocupa menos memoria, pero podría no encontrar la solución: si una rama se puede expandir infinitamente el algoritmo nunca visitara las demas ramas. Además, no garantiza encontrar la solucion con menor cantidad de aristas.

Características

- Si nuestro grafo no es un arbol, debemos tener un cuidado especial: tenemos que evitar recorrer los ciclos, pues quedaremos atrapados en un bucle infinito.
- La busqueda en profundidad ocupa menos memoria, pero podría no encontrar la solución: si una rama se puede expandir infinitamente el algoritmo nunca visitara las demas ramas. Además, no garantiza encontrar la solucion con menor cantidad de aristas.
- La busqueda a lo ancho garantiza encontrar la solucion que recorre la menor cantidad de aristas, pero ocupa mucha mas memoria.

Características

- Si nuestro grafo no es un arbol, debemos tener un cuidado especial: tenemos que evitar recorrer los ciclos, pues quedaremos atrapados en un bucle infinito.
- La busqueda en profundidad ocupa menos memoria, pero podría no encontrar la solución: si una rama se puede expandir infinitamente el algoritmo nunca visitara las demas ramas. Además, no garantiza encontrar la solucion con menor cantidad de aristas.
- La busqueda a lo ancho garantiza encontrar la solucion que recorre la menor cantidad de aristas, pero ocupa mucha mas memoria.
- Ambos son algoritmos de fuerza bruta no informados, esto es, que no tienen ninguna prioridad particular a la hora de elegir un nodo.

Características

- Si nuestro grafo no es un arbol, debemos tener un cuidado especial: tenemos que evitar recorrer los ciclos, pues quedaremos atrapados en un bucle infinito.
- La busqueda en profundidad ocupa menos memoria, pero podría no encontrar la solución: si una rama se puede expandir infinitamente el algoritmo nunca visitara las demas ramas. Además, no garantiza encontrar la solucion con menor cantidad de aristas.
- La busqueda a lo ancho garantiza encontrar la solucion que recorre la menor cantidad de aristas, pero ocupa mucha mas memoria.
- Ambos son algoritmos de fuerza bruta no informados, esto es, que no tienen ninguna prioridad particular a la hora de elegir un nodo.
- Si nuestros operadores tienen un costo asociado, ninguno de los algoritmos garantiza encontrar la solucion de menor costo.

Algoritmo general

```
1  def busqueda(inicial):
2      visitados = []
3      nodos = [inicial]
4
5      while nodos:
6          nodo = nodos.pop()
7
8          if nodo.test_objetivo():
9              return nodo.acciones
10         elif nodo not in visitados:
11             visitados.append(nodo)
12
13             for s in nodo.sucesores():
14                 nodos.append(s)
```

Solucion al problema

```
1  class Nodo:
2      def __init__(self, bidon3 = 0, bidon5 = 0, acciones = ""):
3          self.bidon3, self.bidon5, self.acciones = bidon3, bidon5, acciones
4      def __eq__(self, other):
5          return self.bidon3 == other.bidon3 and self.bidon5 == other.bidon5
6      def test_objetivo(self):
7          return self.bidon5 == 4
8      def llenar3(self): # Analogo para el bidon de 5 litros.
9          self.bidon3 = 3
10         self.acciones += "Llenar 3. "
11     def vaciar3(self): # Analogo para el bidon de 5 litros.
12         self.bidon3 = 0
13         self.acciones += "Vaciar 3. "
14     def volcar3(self): # Analogo para el bidon de 5 litros.
15         volumen = min(5 - self.bidon5, self.bidon3)
16         self.bidon3, self.bidon5 = self.bidon3 - volumen, self.bidon5 + volumen
17         self.acciones += "Volcar 3. "
18     def sucesores(self):
19         s1 = Estado(self.bidon3, self.bidon5, self.acciones).llenar3()
20         s2 = Estado(self.bidon3, self.bidon5, self.acciones).vaciar3()
21         s3 = Estado(self.bidon3, self.bidon5, self.acciones).mover3()
22         # Analogo para el bidon de 5 litros...
23         return [ s1, s2, s3, s4, s5, s6 ]
```

```
>>> print(busqueda(Nodo()))
Llenar 5. Volcar 5. Vaciar 5. Volcar 3. Llenar 3. Volcar 3. Vaciar 5. Volcar 3.
Llenar 3. Volcar 3.
```

Recursos

- Stuart Russell & Peter Norvig - Inteligencia Artificial
- https://www.mathsisfun.com/games/jugs-puzzle.html
- https://www.youtube.com/watch?v=giM1Xdu5SIE