FACULTAD DE CIENCIAS
INTELIGENCIA ARTIFICIAL

Exploradores de laberinto

Equipo: `Skynet Scribes`

Número de practica: `02`

Carlos Daniel Cortés Jiménez
420004846

Sarah Sophía Olivares García
318360638

Marco Silva Huerta
316205326

Juan Daniel Barrera Holan
417079372

Laura Itzel Tinoco Miguel
316020189

Profesora: Cecilia Reyes Peña
Ayudante teoría: Karem Ramos Calpulalpan
Ayudante laboratorio: Tania Michelle Rubí Rojas

Semestre 2024-2

Fecha de entrega:
28 de Febrero del 2024

1. Investigación

¿Cuál es la diferencia entre Backtracking y el algoritmo de búsqueda BFS?

¿Cuál es la diferencia entre Backtracking y el algoritmo de búsqueda DFS?

Primero expliquemos que es el algoritmos DFS (Depth First Search):

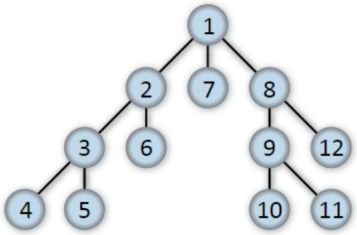
Es un algoritmo utilizado en la teoría de grafos para buscar y recorrer grafos o árboles, no utiliza información sobre el costo de los movimientos o la ubicación de la meta.

El recorrido comienza revisando el nodo actual (la raíz si es un árbol o seleccionando un nodo arbitrario si no), después se mueve a uno de sus sucesores para repetir el proceso y explora tanto como sea posible en esa rama, si el nodo actual no tiene sucesor a revisar, regresamos a su predecesor y el proceso continua, es decir se mueve a otro sucesor del nodo, si la meta o solución es encontrada la búsqueda termina.

- Pseudocódigo: El algoritmo DFS puede tener una implementación recursiva o iterativa (utilizando pilas), a continuación esta el pseudocodigo de la implementación recursiva:

```
1: procedure DFS(vertex v)
2:     visit(v)
3:     for all neighbor u of v do
4:         if u is undiscovered then
5:             call DFS(u)
6:         end if
7:     end for
8: end procedure
```

- Gráfica: La siguiente gráfica muestra como se van descubriendo los nodos con el algoritmo DFS.



- Aplicaciones: Los principales usos que le dan al algoritmo son:
  - Determinación de Conectividad: DFS se puede utilizar para determinar si un grafo es conexo, es decir, si hay un camino entre cada par de nodos.
  - Detección de Ciclos: DFS puede detectar ciclos en un grafo no dirigido. Si mientras se ejecuta el algoritmo encuentra un nodo ya visitado, entonces hay un ciclo.
  - Topological Sorting: DFS se puede utilizar para realizar una ordenación topológica en un grafo dirigido, que es una lista lineal de todos los nodos tal que para cada arista dirigida uv de un nodo u a un nodo v, u aparece antes que v en la lista.
  - Encontrar Componentes Conectados: DFS se puede utilizar para encontrar los componentes conectados de un grafo no dirigido.

- Resolver acertijos con una sola solución, como laberintos.

Tiene una complejidad de O(n) en el peor de los casos y funciona mejor en árboles poco profundos.

Diferencias entre DFS y Backtracking

Ahora teniendo todo esto en cuenta la principal diferencia que tiene el **algoritmo DFS** con **backtracking** es que DFS encuentra una solución a un problema (ya que al encontrar una meta se deja de recorrer la gráfica) y backtracking encuentra todas las posibles soluciones de ese problema, entonces podemos decir que tienen diferencia en el espacio de búsqueda:

- DFS: Se adentra tanto como sea posible en la estructura del grafo antes de retroceder para explorar otras ramas.
- Backtracking: Explora todas las posibles soluciones de manera incremental, retrocediendo cuando se alcanza una solución inválida o se agotan todas las opciones.

Es decir el backtracking evalúa exhaustivamente cada posibilidad y retrocede cuando encuentra una solución inválida, mientras que DFS se adentra en la exploración del espacio de búsqueda, pero no garantiza encontrar todas las soluciones.

2. Comparación de Resultados

DFS:

1. **Enfoque:** Backtracking, exploración profunda.
2. **Exploración:** Se adentra tanto como sea posible antes de retroceder.
3. **Optimización:** No garantiza encontrar la solución más corta.
4. **Encuentra:** Una solución y muestra la ruta.
5. **Distancia:** No considera la distancia, no hay garantía de que use la ruta más corta.
6. **Eficiencia:** Puede ser eficiente en laberintos con caminos ramificados y no es necesario encontrar la ruta más corta.

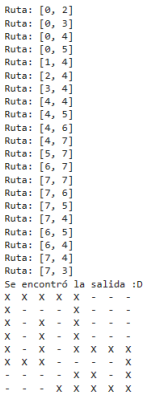


Figura 1: DFS

BFS:

1. **Enfoque:** Exploración nivel por nivel.
2. **Exploración:** Comienza explorando todos los nodos a una distancia d antes de pasar a los nodos a una distancia $d + 1$.
3. **Optimización:** Garantiza encontrar la solución más corta. Garantiza la optimización.
4. **Distancia:** Encuentra la ruta más corta desde el inicio hasta la salida.
5. **Eficiencia:**Evita Explorar caminos no óptimos.
6. **Evita:** Explorar caminos no óptimos.

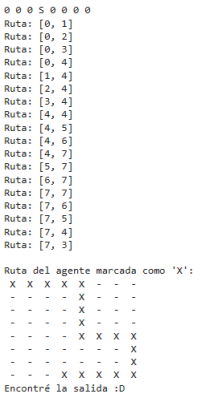


Figura 2: BFS

Ejemplo:

Si elegimos el Laberinto 02, ambos algoritmos van a encontrar la salida. La diferencia sería:

- **DFS:** Va a mostrar una ruta más larga pero válida.
- **BFS:** Va a garantizar la ruta más corta.

Entonces, la elección depende de nuestra prioridad, ya sea optimización o la exploración de diferentes caminos.

Referencias

[KT05] John Kleinberg y Eva Tardos. *Algorithm Algorithm Design*. Addison Addison-Wesley, 2005.

[Fig12] Jhosimar George Arias Figueroa. *Algorithms and More*. https://jariasf.wordpress.com/2012/03/02/algoritmo-de-busqueda-depth-first-search-parte-1/. 2012.

[RN16] Stuart Russell y Peter Norvig. *Inteligencia Artificial Un Enfoque moderno*. 2nd. Pearson Prentice Hall, 2016.

[Ref21] Programación Refactoriza. *Backtracking*. https://docs.jjpeleato.com/algoritmia/backtracking. 2021.

[Edu] Education-wiki. *DFS Algorithm*. https://es.education-wiki.com/7857184-dfs-algorithm.

[Tec] Techiedelight. *Búsqueda en amplitud (BFS): implementación iterativa y recursiva*. https://www.techiedelight.com/es/breadth-first-search/.