Modelado de Problemas con Grafos

Certamen Nacional de OIA

Facundo Gutiérrez (Guty)

Octubre 2018

Grafos

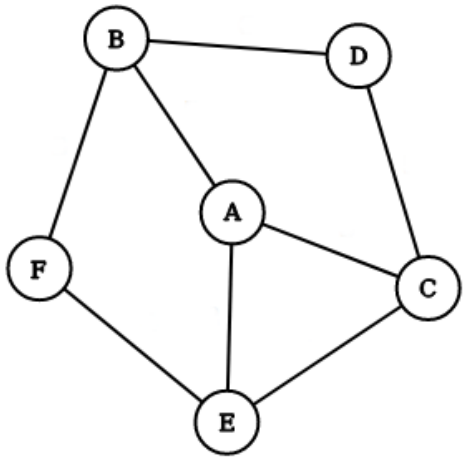
Definición:

Un grafo es un conjunto de objetos, llamados **vértices** o **nodos**, unidos mediantes líneas llamadas **aristas** que relacionan un par de vértices. En algunos casos, esas aristas pueden tener un **peso** asociado, o una **dirección** particular.

Grafos

Definición:

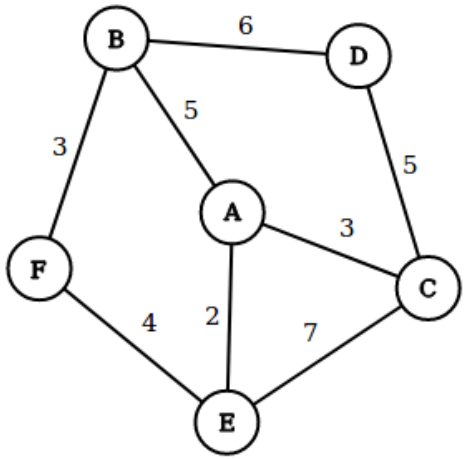
Un grafo es un conjunto de objetos, llamados **vértices** o **nodos**, unidos mediantes líneas llamadas **aristas** que relacionan un par de vértices. En algunos casos, esas aristas pueden tener un **peso** asociado, o una **dirección** particular.



Grafos

Definición:

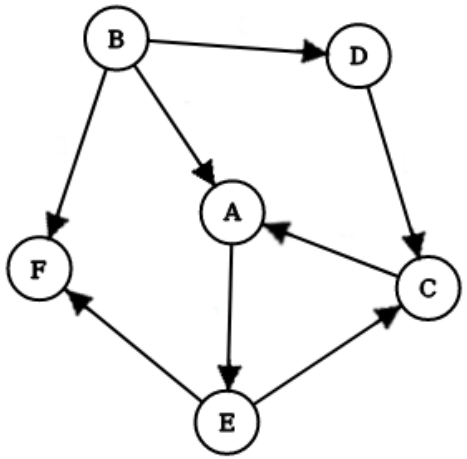
Un grafo es un conjunto de objetos, llamados **vértices** o **nodos**, unidos mediantes líneas llamadas **aristas** que relacionan un par de vértices. En algunos casos, esas aristas pueden tener un **peso** asociado, o una **dirección** particular.



Grafos

Definición:

Un grafo es un conjunto de objetos, llamados **vértices** o **nodos**, unidos mediantes líneas llamadas **aristas** que relacionan un par de vértices. En algunos casos, esas aristas pueden tener un **peso** asociado, o una **dirección** particular.



Algunas definiciones

Definiciones: Vecino y Grado

Dada una arista que conecta dos vértices u y v, decimos que u es **vecino** de v (y que v es vecino de u). Además, a la cantidad de vecinos de u se le llama el **grado** de u.

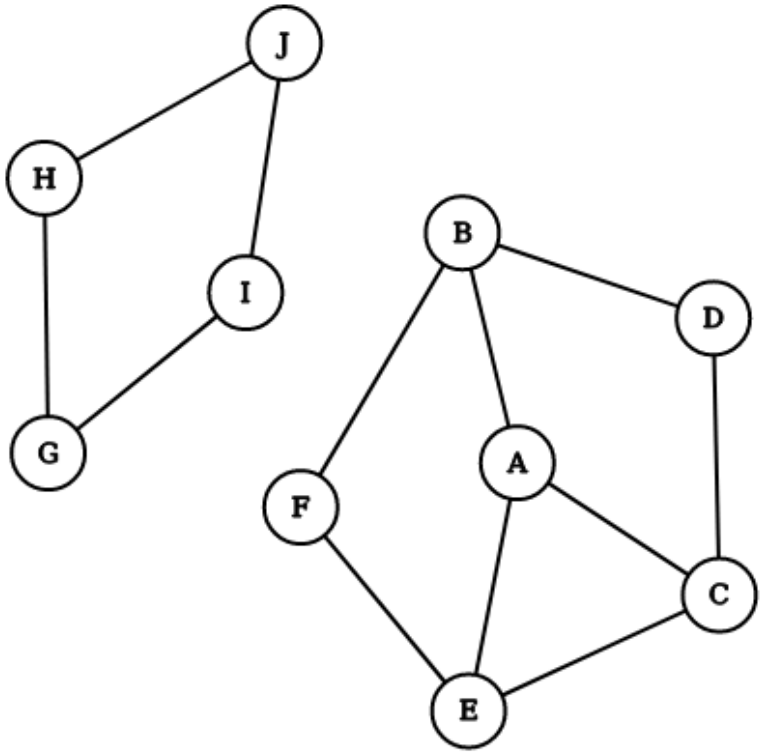
Algunas definiciones

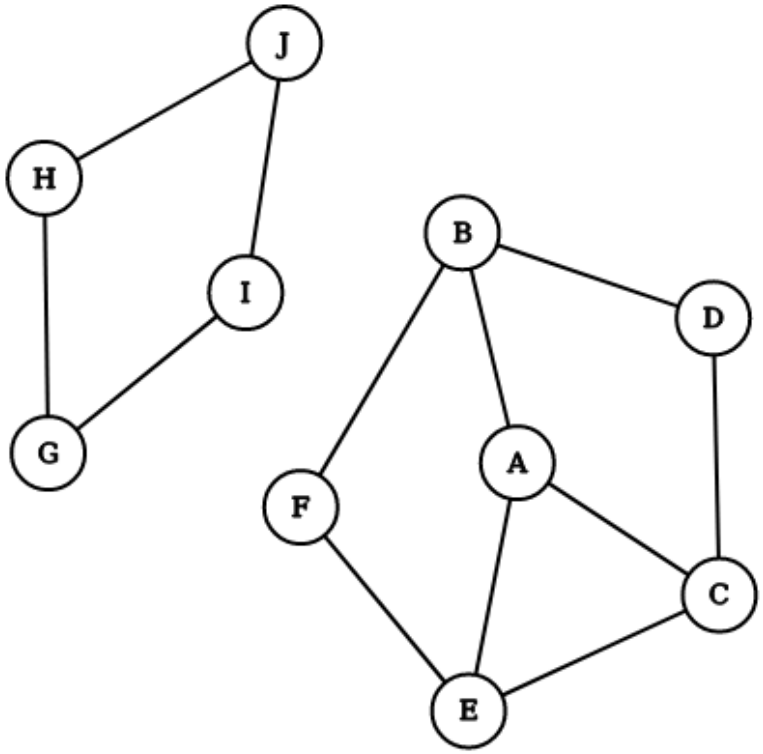
Definiciones: Vecino y Grado

Dada una arista que conecta dos vértices u y v, decimos que u es **vecino** de v (y que v es vecino de u). Además, a la cantidad de vecinos de u se le llama el **grado** de u.

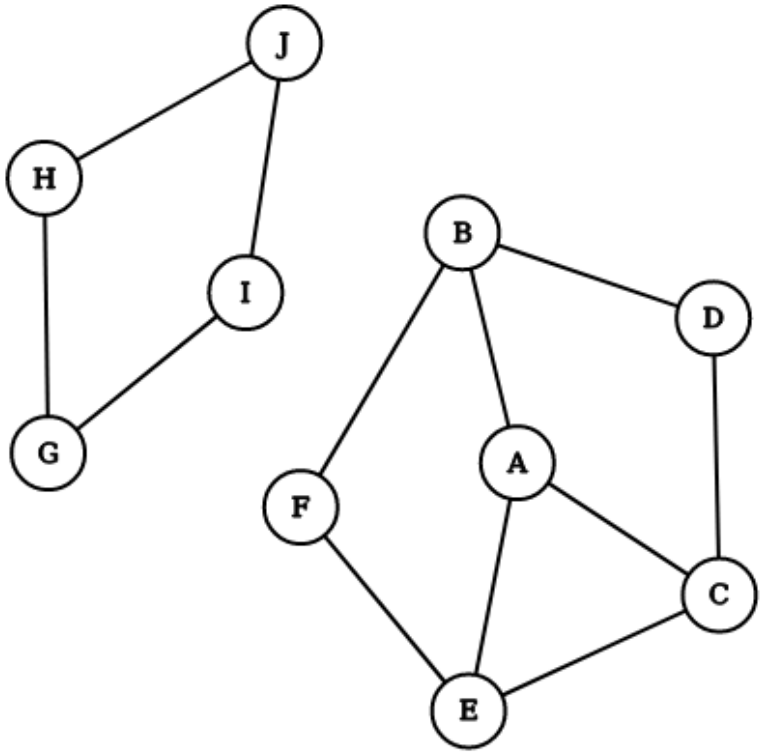
Definición: Conexión y Distancia

Decimos que dos vértices u y v están **conectados** si hay un camino (secuencia de aristas con vértice en común) que los une. A la menor cantidad de aristas de un camino que los une, le llamamos la **distancia** de u a v.

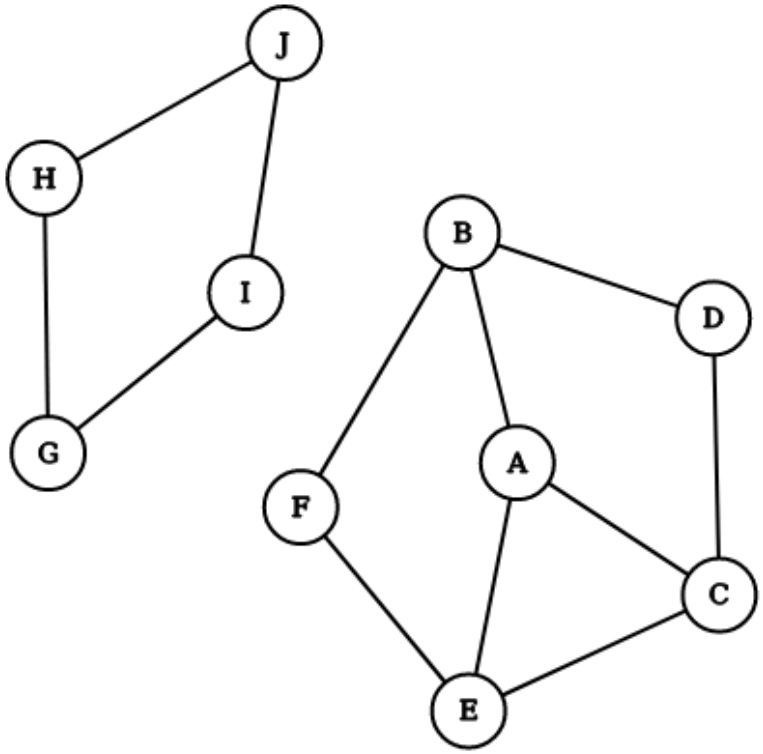

J
H
I
G
B
D
A
F
C
E



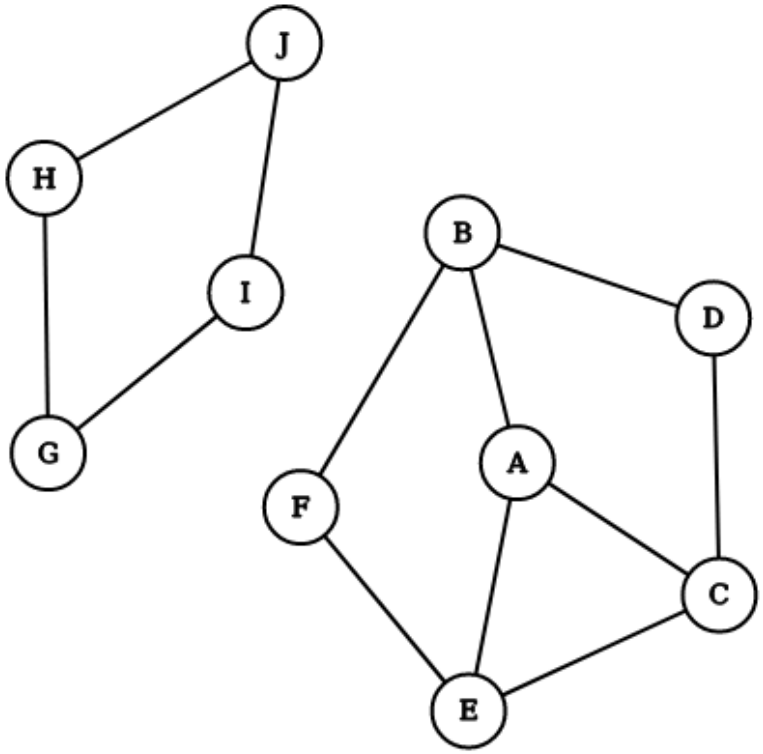
¿Cuál es el grado de B?



¿Cuál es el grado de B? ¿Cuál es la distancia de F a C?



¿Cuál es el grado de B? ¿Cuál es la distancia de F a C?
Los nodos B y E, ¿están conectados?



¿Cuál es el grado de B? ¿Cuál es la distancia de F a C?
Los nodos B y E, ¿están conectados? ¿y los nodos B y G?

Algunas definiciones

Definición: Bipartito

Un grafo se dice **bipartito** si le podemos pintar a los vértices de dos colores de forma que toda arista conecta a dos vértices de distinto color.

Algunas definiciones

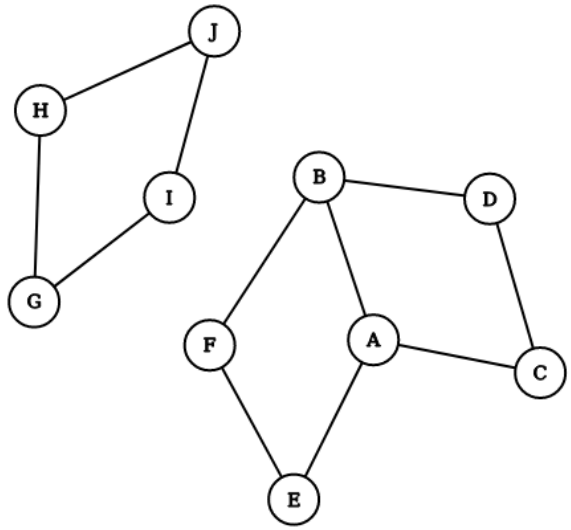
Definición: Bipartito

Un grafo se dice **bipartito** si le podemos pintar a los vértices de dos colores de forma que toda arista conecta a dos vértices de distinto color. O sea, si lo podemos pintar con el patrón de un tablero de ajedrez.

Algunas definiciones

Definición: Bipartito

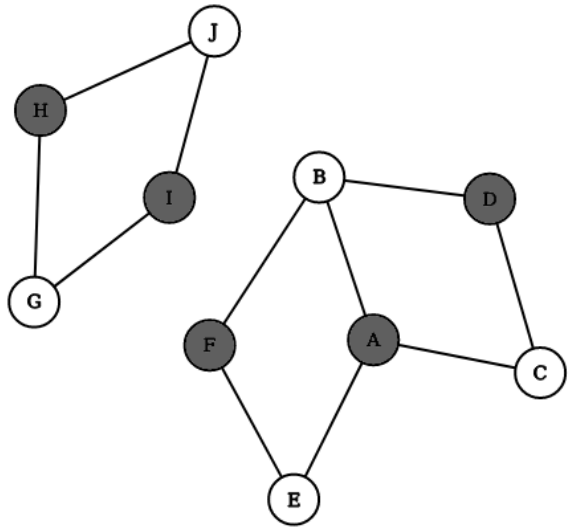
Un grafo se dice **bipartito** si le podemos pintar a los vértices de dos colores de forma que toda arista conecta a dos vértices de distinto color. O sea, si lo podemos pintar con el patrón de un tablero de ajedrez.



Algunas definiciones

Definición: Bipartito

Un grafo se dice **bipartito** si le podemos pintar a los vértices de dos colores de forma que toda arista conecta a dos vértices de distinto color. O sea, si lo podemos pintar con el patrón de un tablero de ajedrez.



Representaciones de grafos

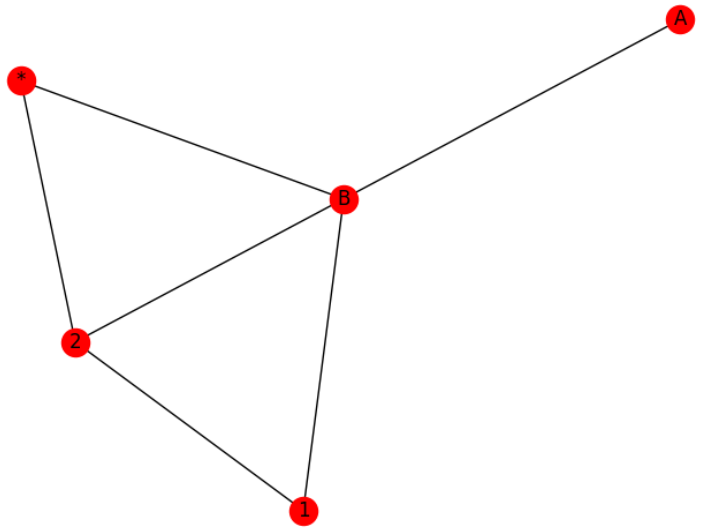
Hay muchas formas de representar un grafo. En particular:

Representaciones de grafos

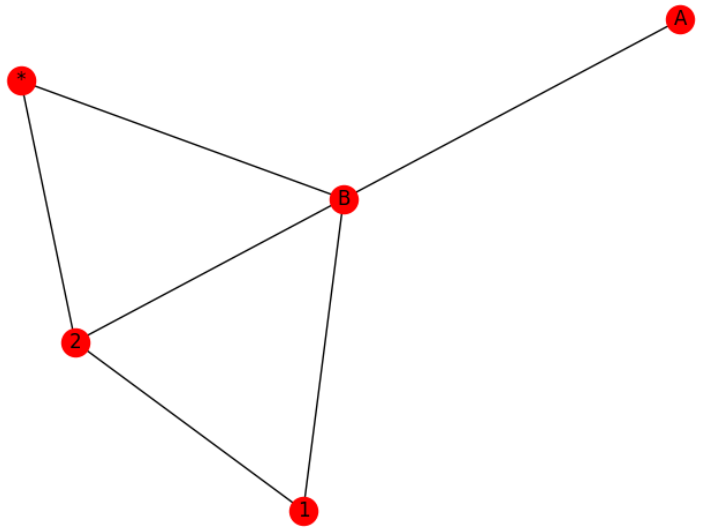
Hay muchas formas de representar un grafo. En particular:

- Lista de aristas (y cantidad de vértices)
- Matriz de adyacencia
- Listas de adyacencia

Representación de Grafos - Matriz de Adyacencia

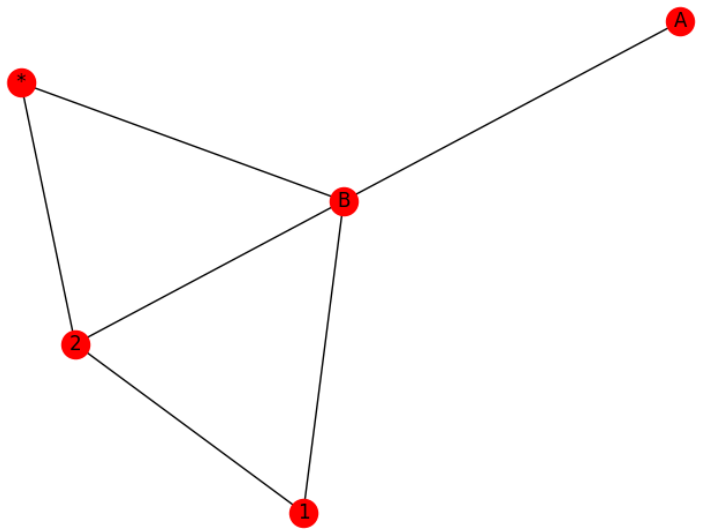


Representación de Grafos - Matriz de Adyacencia

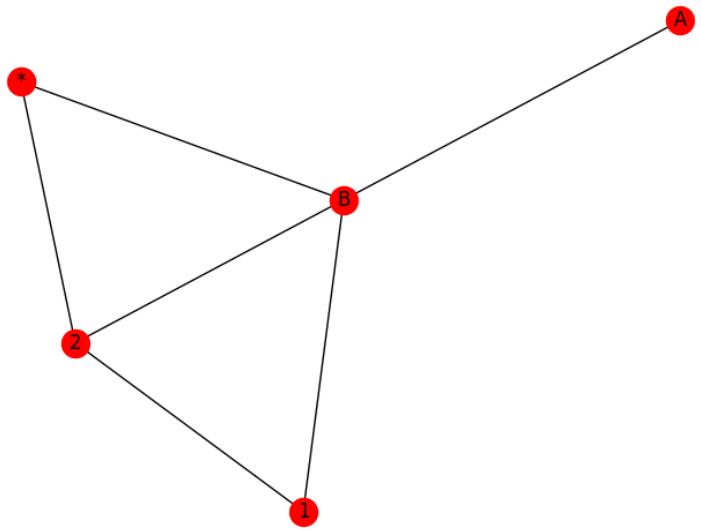


	1	2	A	B	$\star$
1	0	1	0	1	0
2	1	0	0	1	1
A	0	0	0	1	0
B	1	1	1	0	1
$\star$	0	1	0	1	0

Representaciones de grafos - Lista de adyacencia



Representaciones de grafos - Lista de adyacencia



1 :	{2,B}
2 :	{1,B,⋆}
A :	{B}
B :	{1,2,A,⋆}
⋆ :	{2,B}

Recorrido de Grafos - BFS

Queremos recorrer los vértices de un grafo. Según cómo decidamos recorrer el grafo vamos a poder distinguir distintas propiedades.

Recorrido de Grafos - BFS

Queremos recorrer los vértices de un grafo. Según cómo decidamos recorrer el grafo vamos a poder distinguir distintas propiedades.
En un recorrido **BFS** (*Breadth-First Search*) vamos a distinguir a un nodo en particular (que en un principio puede ser cualquiera) donde comenzaremos el recorrido del grafo. Llamémosle s a dicho nodo.
Comenzamos visitando el nodo s, luego visitamos a todos los vecinos de s (que son los nodos que están a distancia 1 de s), después visitamos a los vecinos de estos vecinos (que son los nodos que están a distancia 2 de s), y así ....

Recorrido de Grafos - BFS

Queremos recorrer los vértices de un grafo. Según cómo decidamos recorrer el grafo vamos a poder distinguir distintas propiedades.
En un recorrido **BFS** (*Breadth-First Search*) vamos a distinguir a un nodo en particular (que en un principio puede ser cualquiera) donde comenzaremos el recorrido del grafo. Llamémosle s a dicho nodo.
Comenzamos visitando el nodo s, luego visitamos a todos los vecinos de s (que son los nodos que están a distancia 1 de s), después visitamos a los vecinos de estos vecinos (que son los nodos que están a distancia 2 de s), y así ....
De alguna forma, podemos pensar que se va expandiendo ordenadamente en todas las direcciones a la vez desde el nodo de origen s.

Algoritmo BFS

- $\mathbb{S}$: indica la capa actual de vértices que estamos visitando.
- $\mathbb{W}$: indica la capa siguiente de vértices por visitar.
- `visto`: un arreglo que indica qué vértices ya fueron procesados.
- s: vértice inicial en el que comenzamos el recorrido.

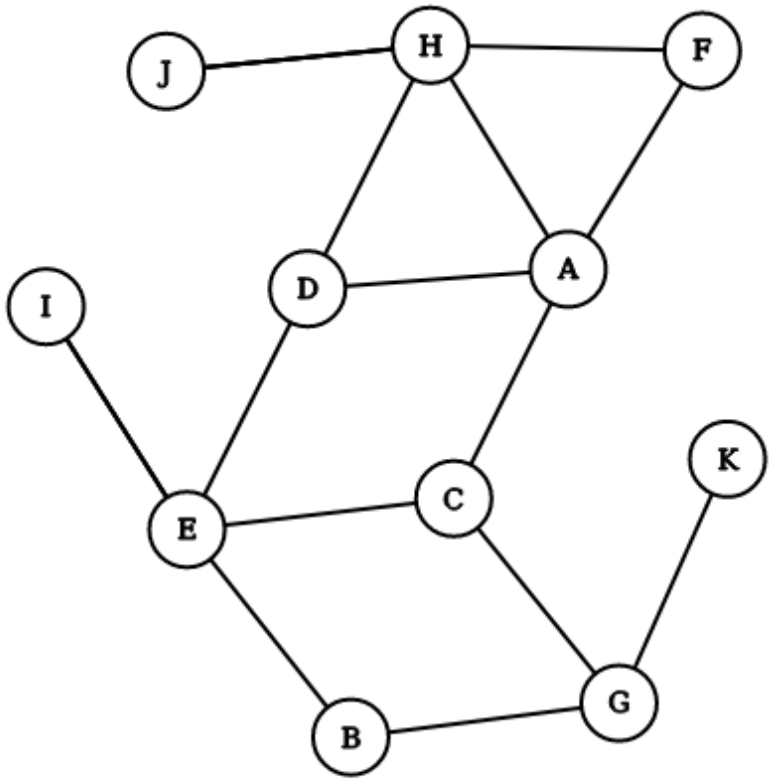
Algoritmo BFS

- $\mathbb{S}$: indica la capa actual de vértices que estamos visitando.
- $\mathbb{W}$: indica la capa siguiente de vértices por visitar.
- `visto`: un arreglo que indica qué vértices ya fueron procesados.
- s: vértice inicial en el que comenzamos el recorrido.

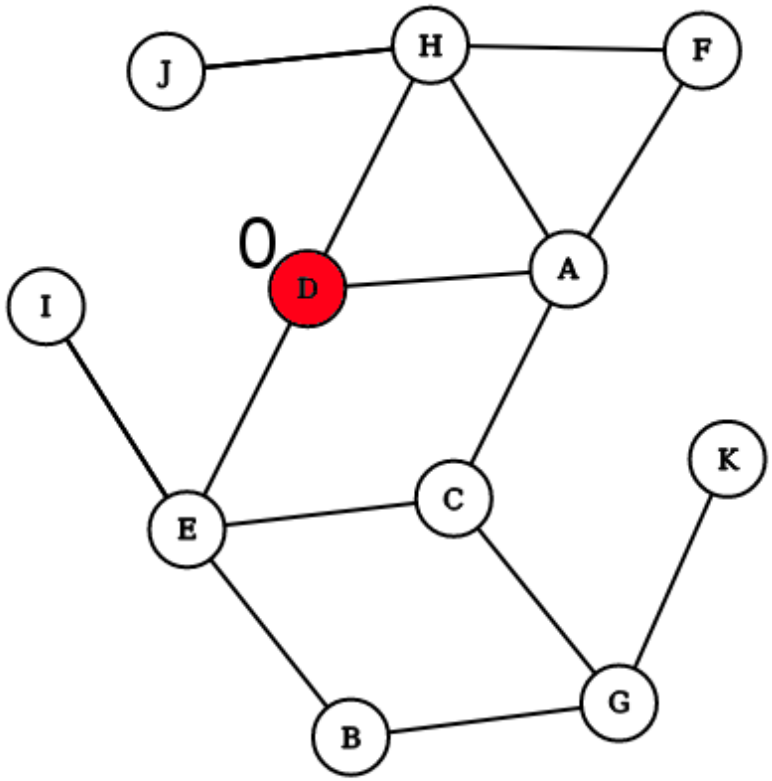
BFS($\mathcal{G}, s$): ($\mathcal{G}$ es nuestro grafo en alguna representación)

1 para todo vértice v : `visto`$[v] \leftarrow$ *Falso*
2 `visto`$[s] \leftarrow$ *Verdadero*, $\mathbb{S} \leftarrow \{s\}, \mathbb{W} \leftarrow \{\}$
3 Mientras $\mathbb{S}$ no esté vacío:
  3.1 para todo vértice v en $\mathbb{S}$:
    3.1.1 para todo w vecino de v con `visto`$[w] ==$ *Falso*:
      Agregar w a $\mathbb{W}$
      `visto`$[w] \leftarrow$ *Verdadero*
  3.2 $\mathbb{S} \leftarrow \mathbb{W}$
  3.3 $\mathbb{W} \leftarrow \{\}$

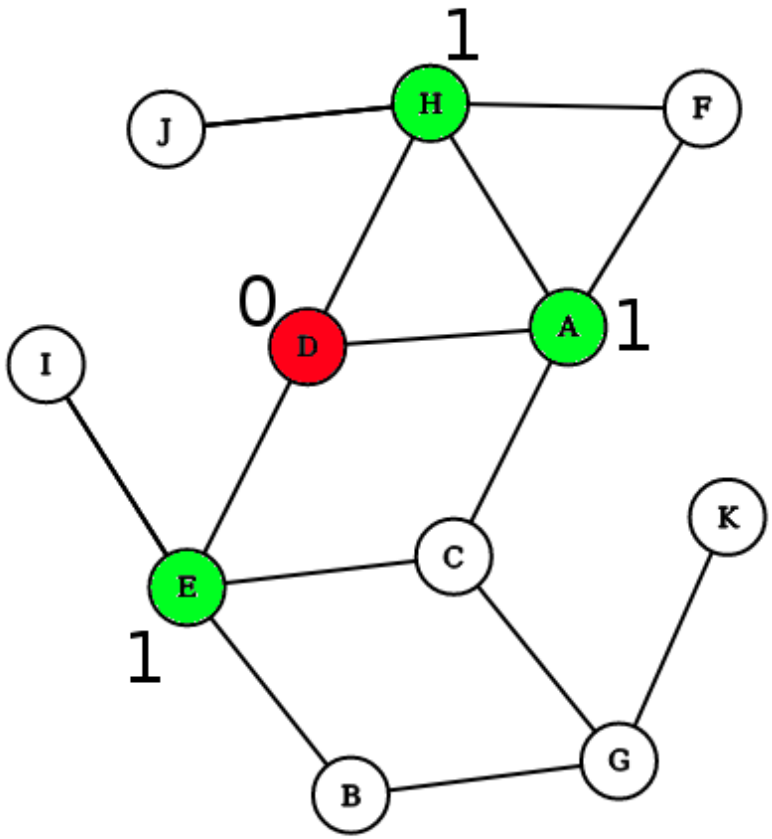
Recorrido de Grafos - BFS



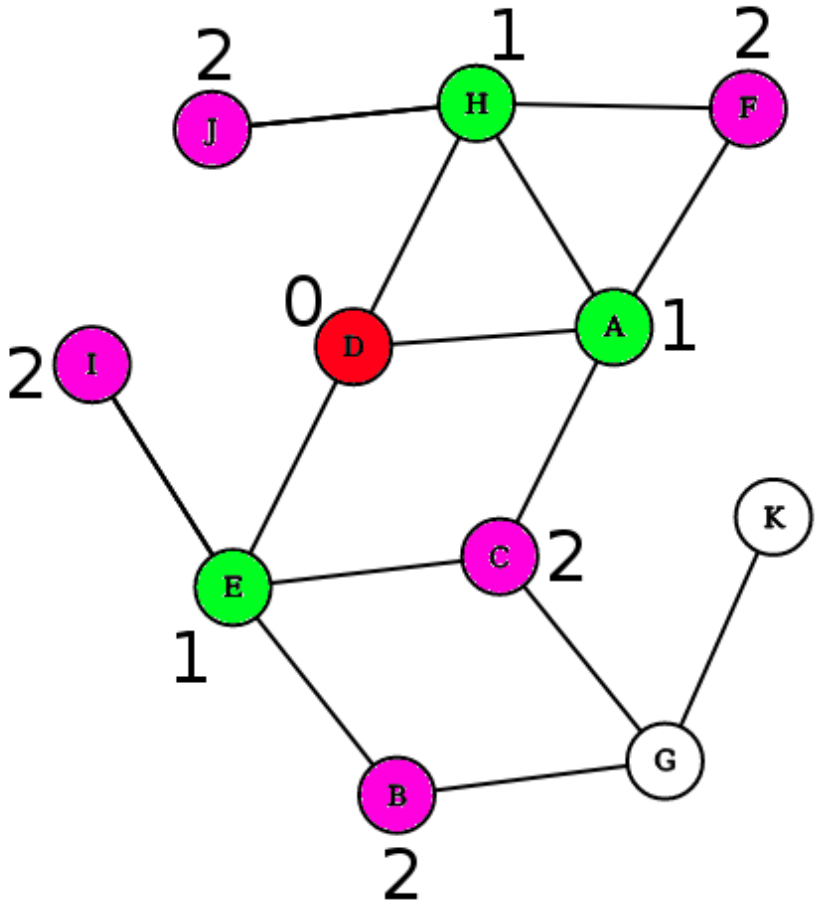
Recorrido de Grafos - BFS



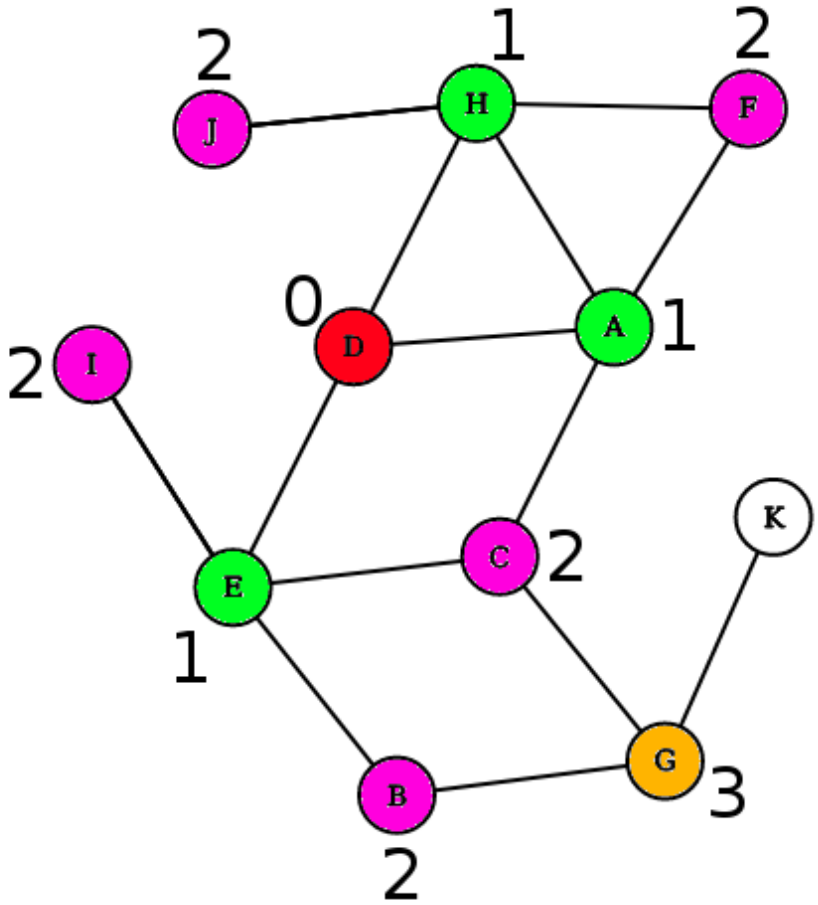
Recorrido de Grafos - BFS



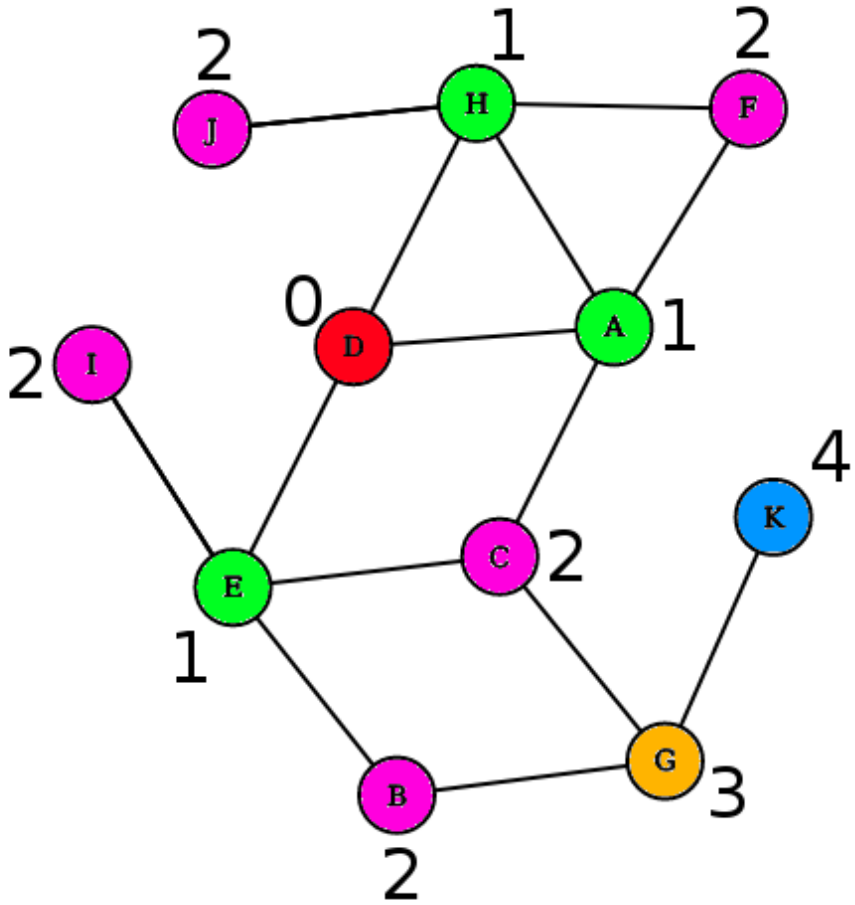
Recorrido de Grafos - BFS



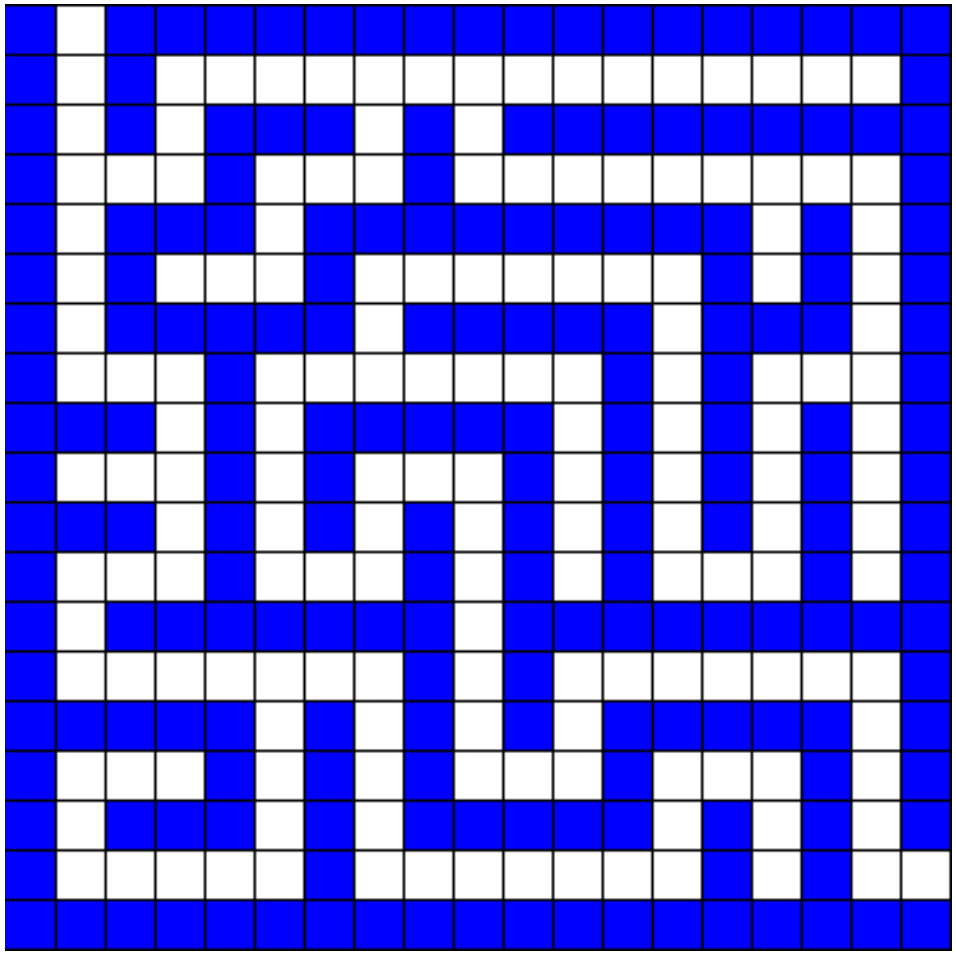
Recorrido de Grafos - BFS



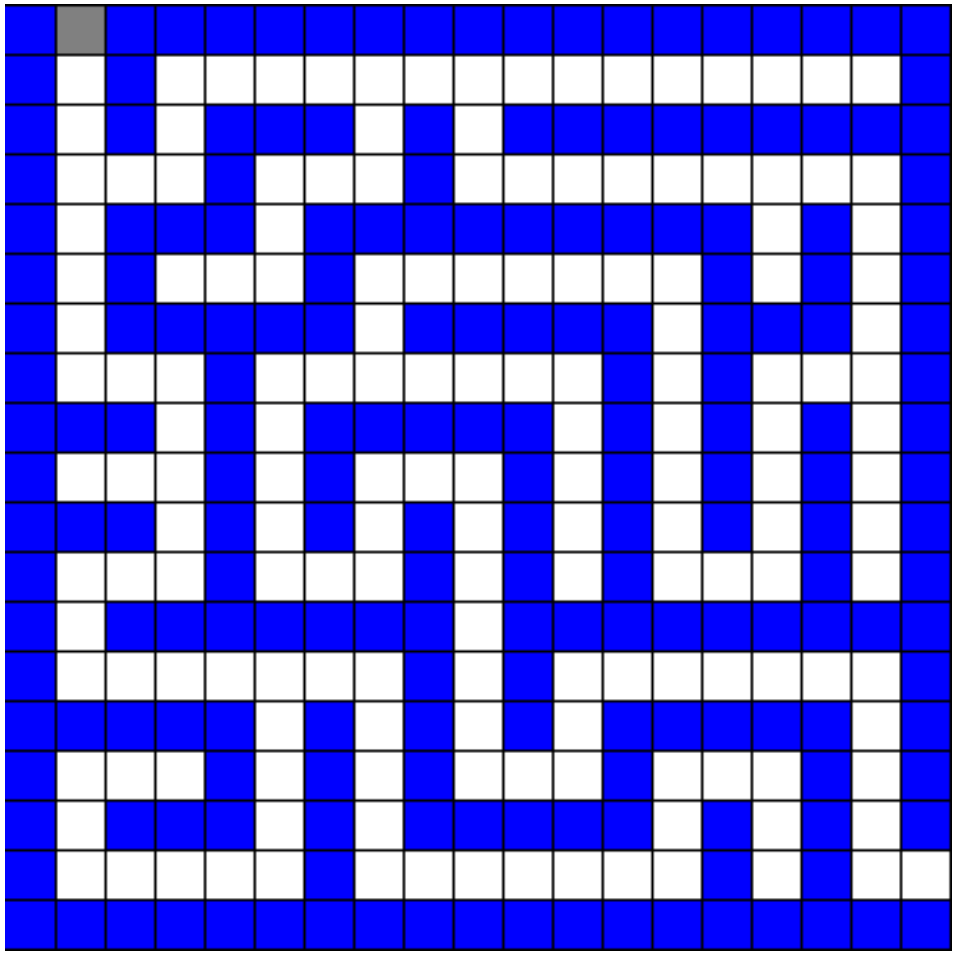
Recorrido de Grafos - BFS



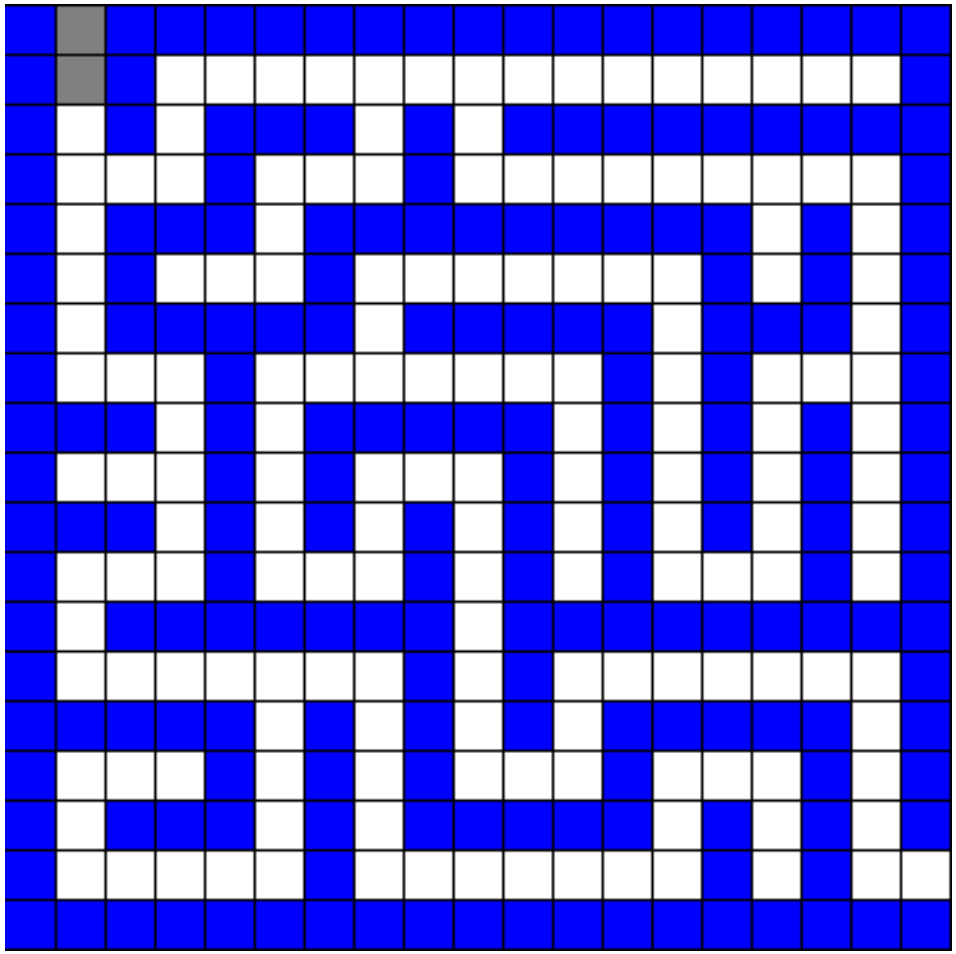
Recorrido de Grafos - BFS



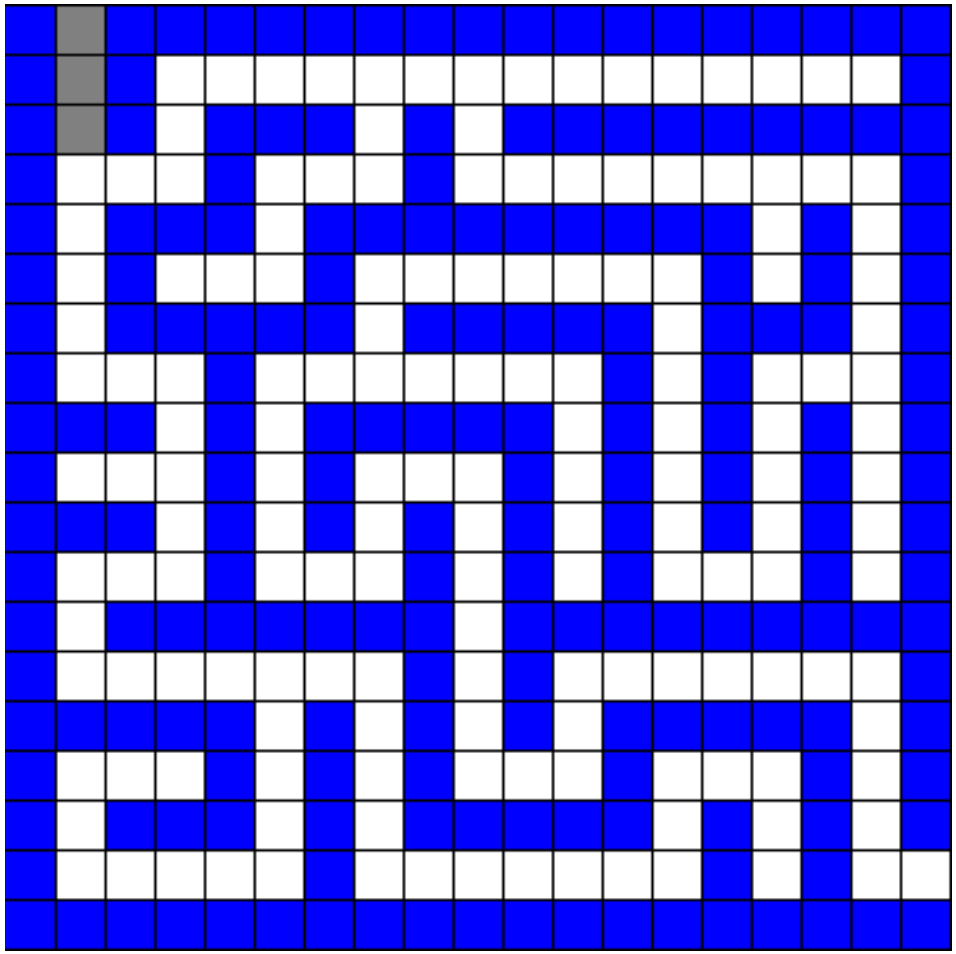
Recorrido de Grafos - BFS



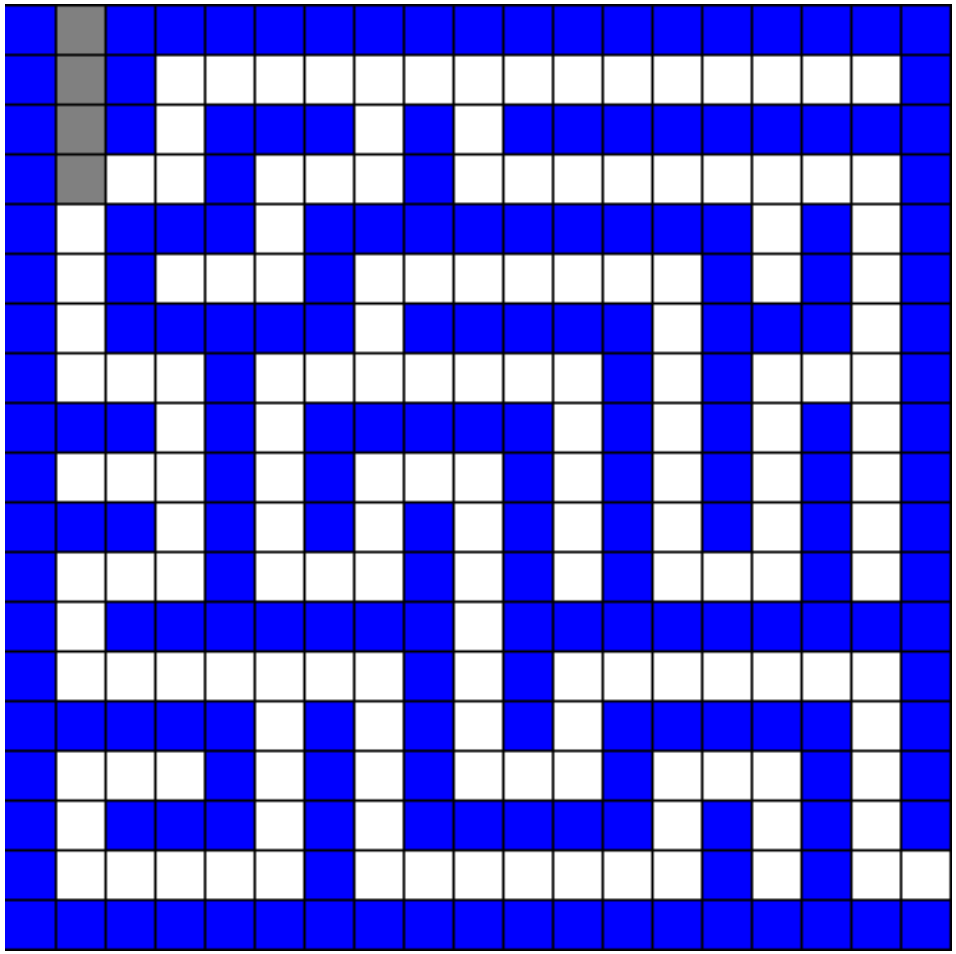
Recorrido de Grafos - BFS



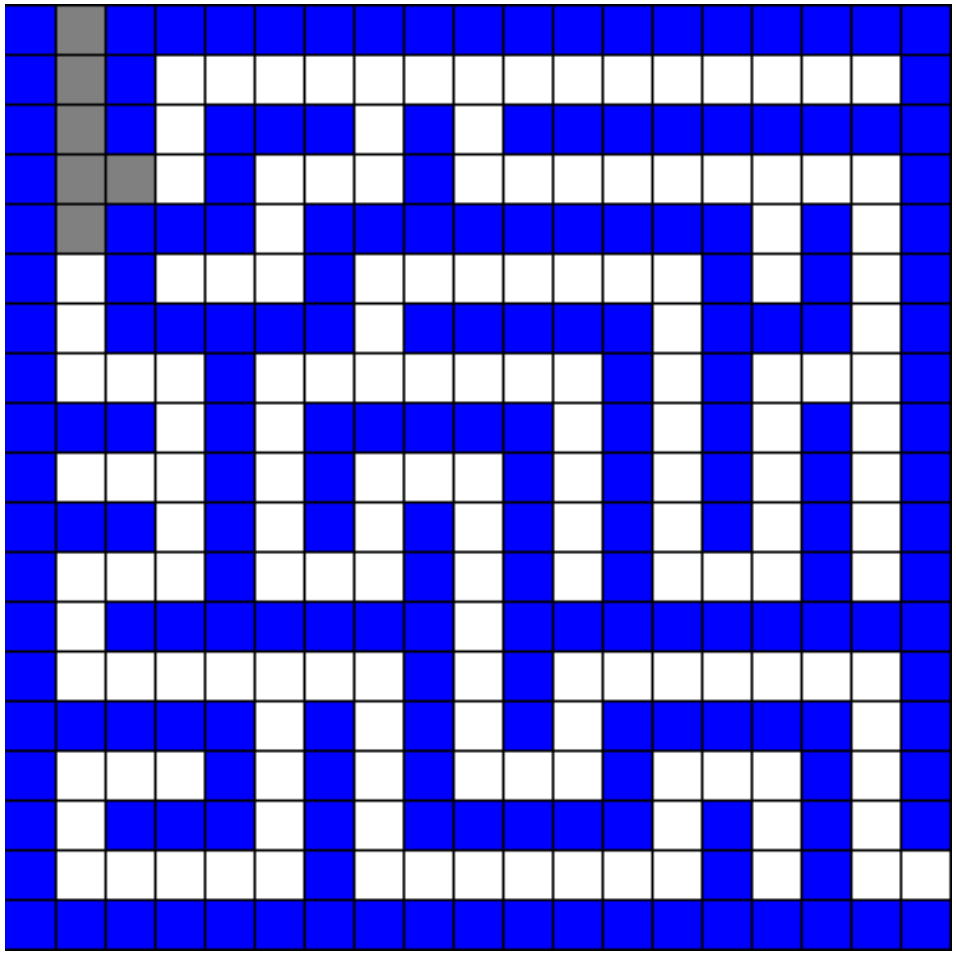
Recorrido de Grafos - BFS



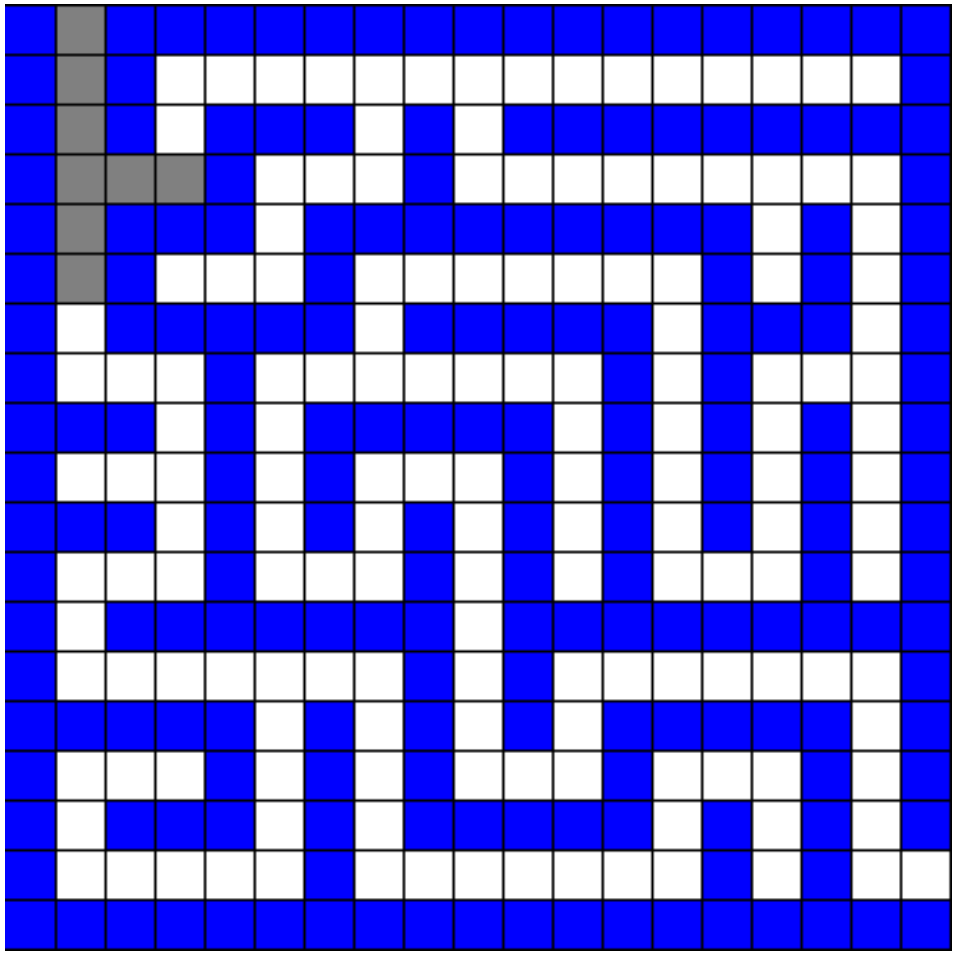
Recorrido de Grafos - BFS



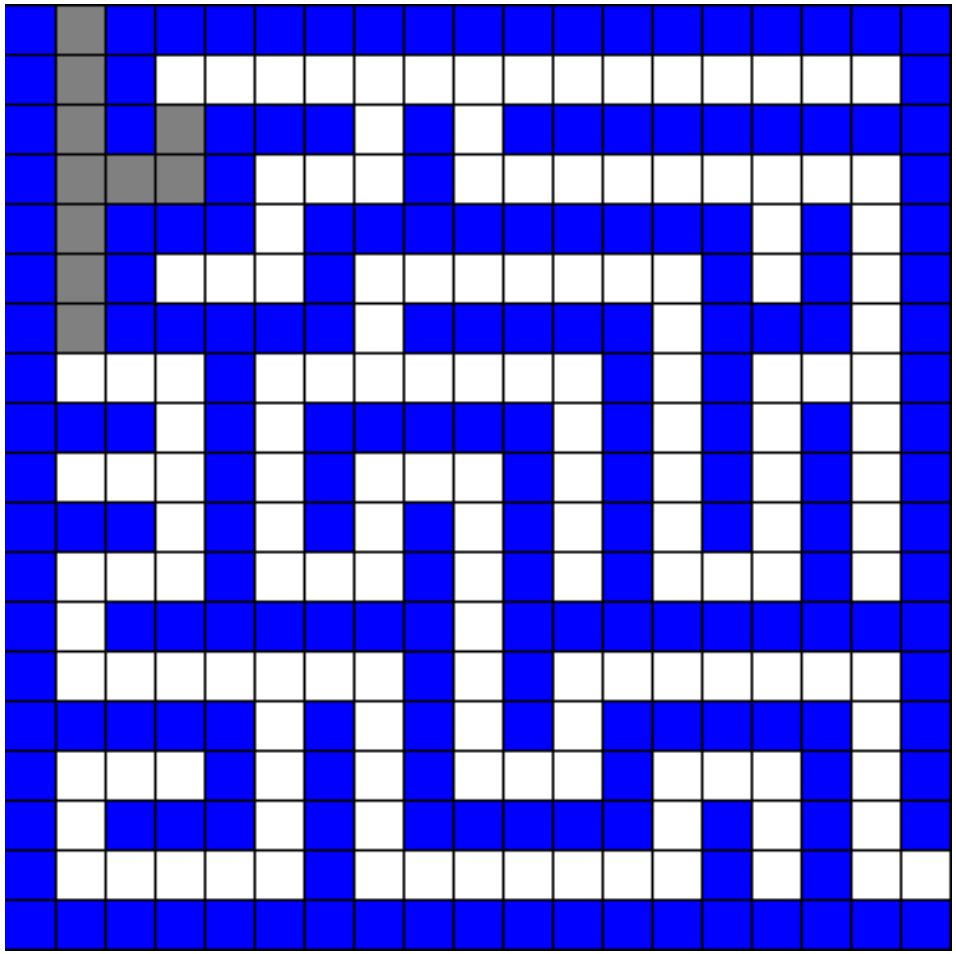
Recorrido de Grafos - BFS



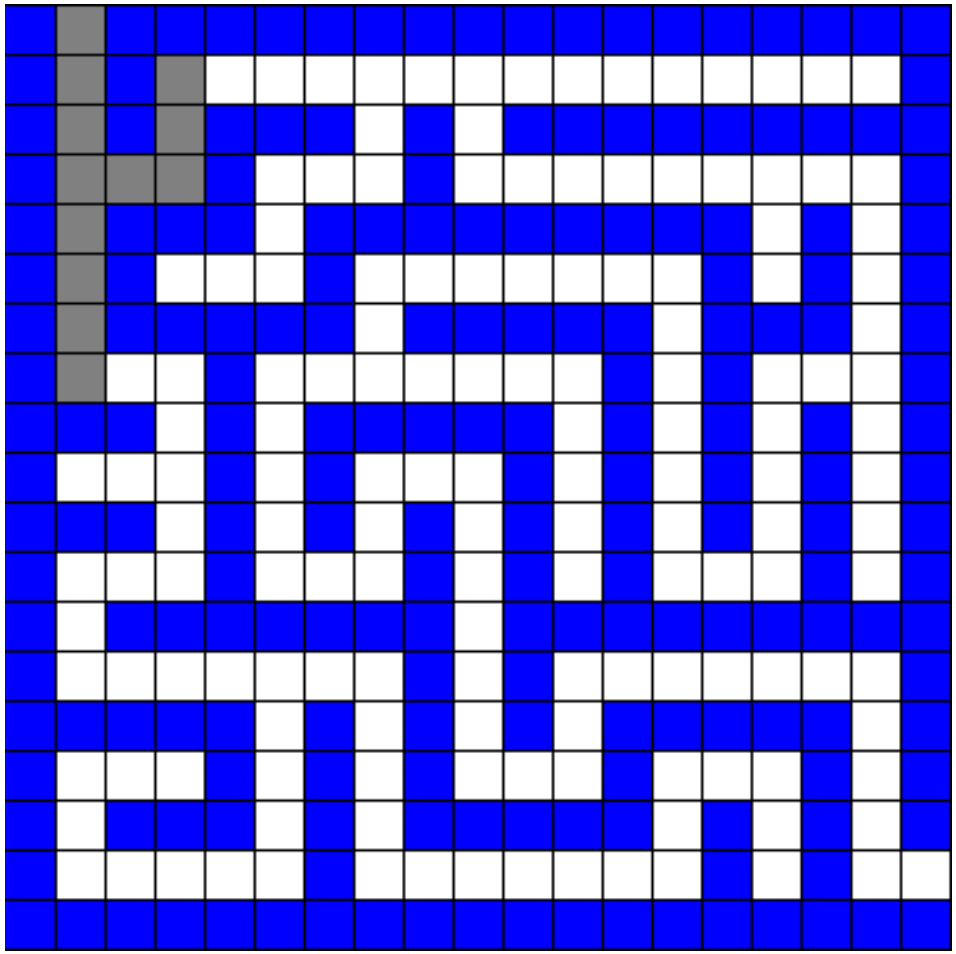
Recorrido de Grafos - BFS



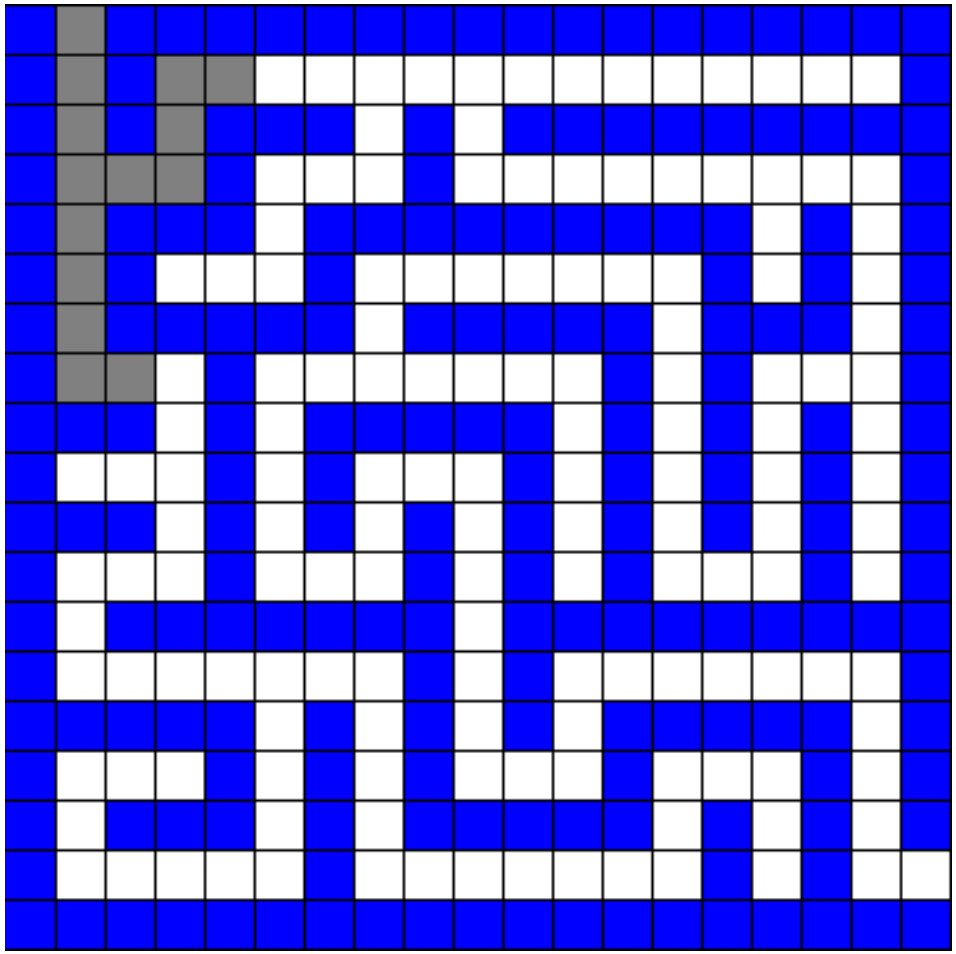
Recorrido de Grafos - BFS



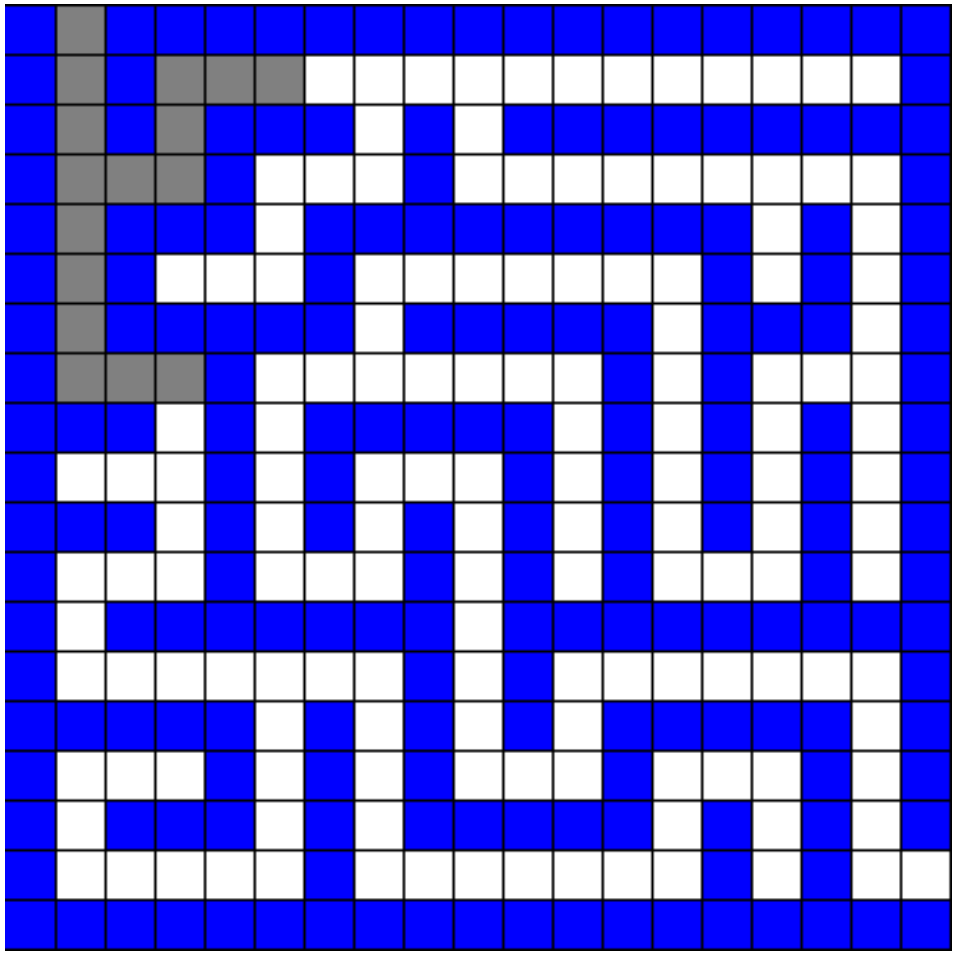
Recorrido de Grafos - BFS



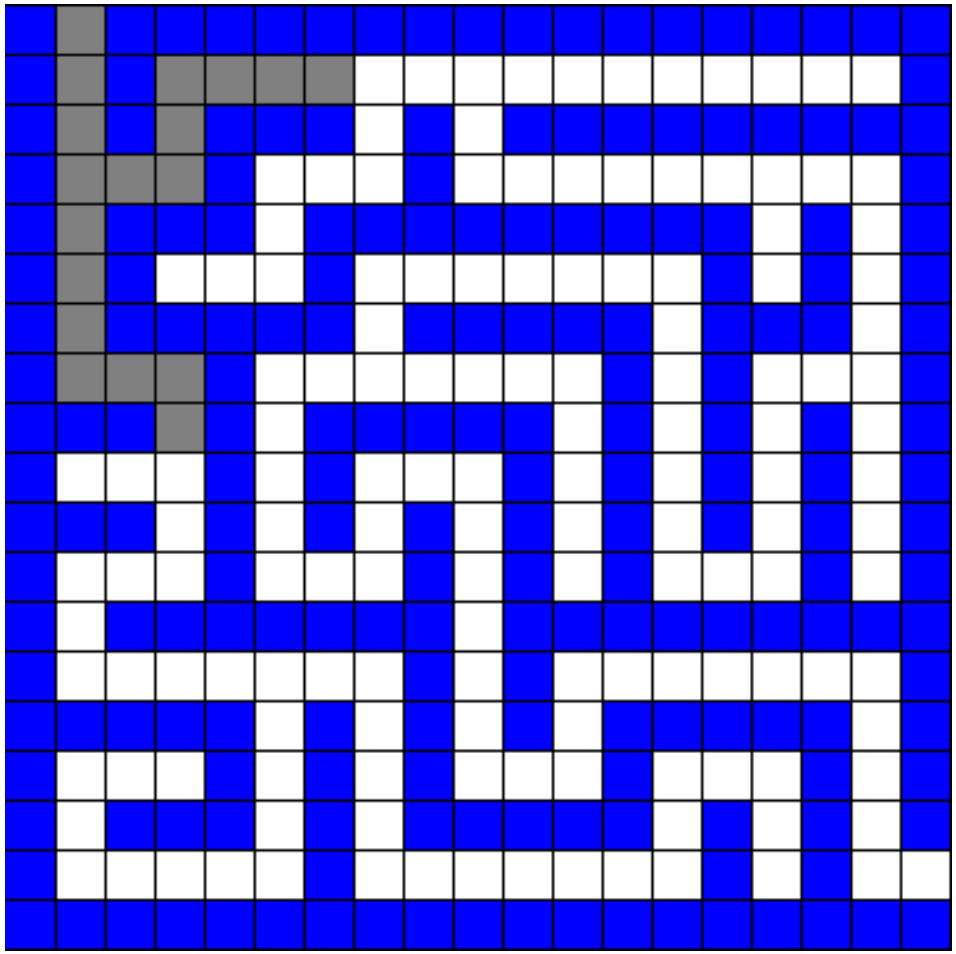
Recorrido de Grafos - BFS



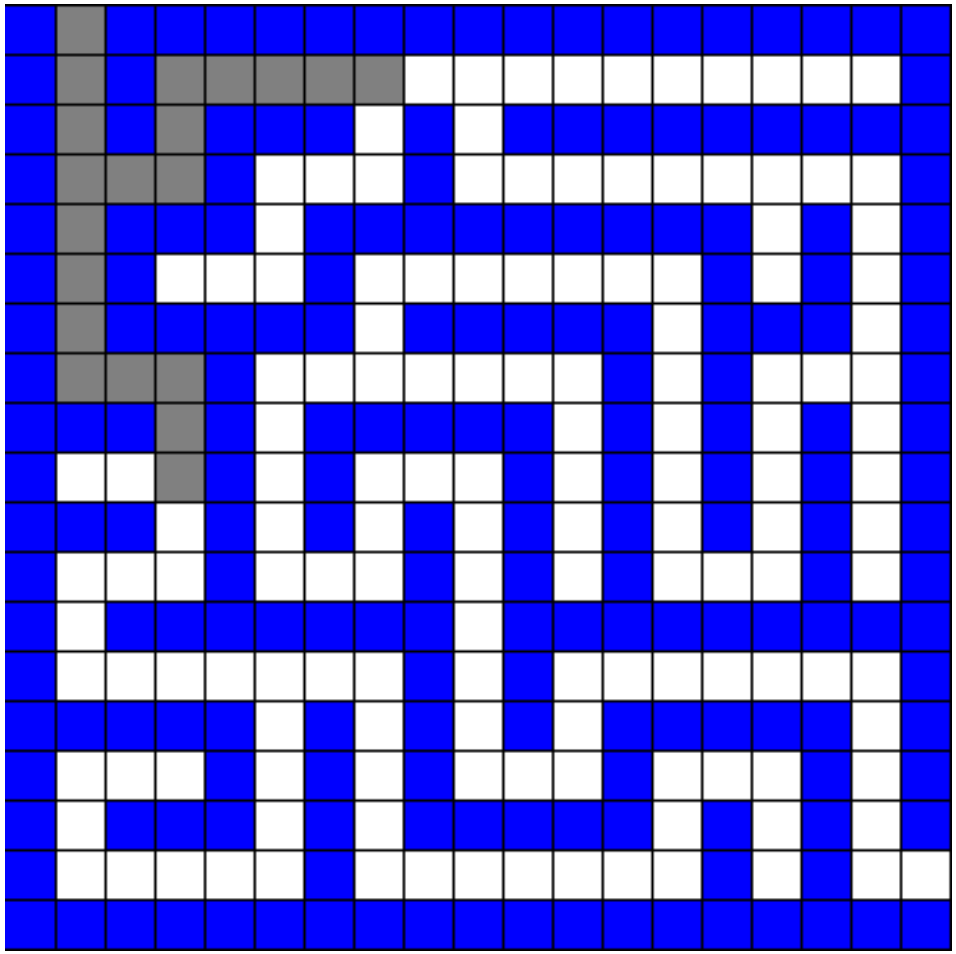
Recorrido de Grafos - BFS



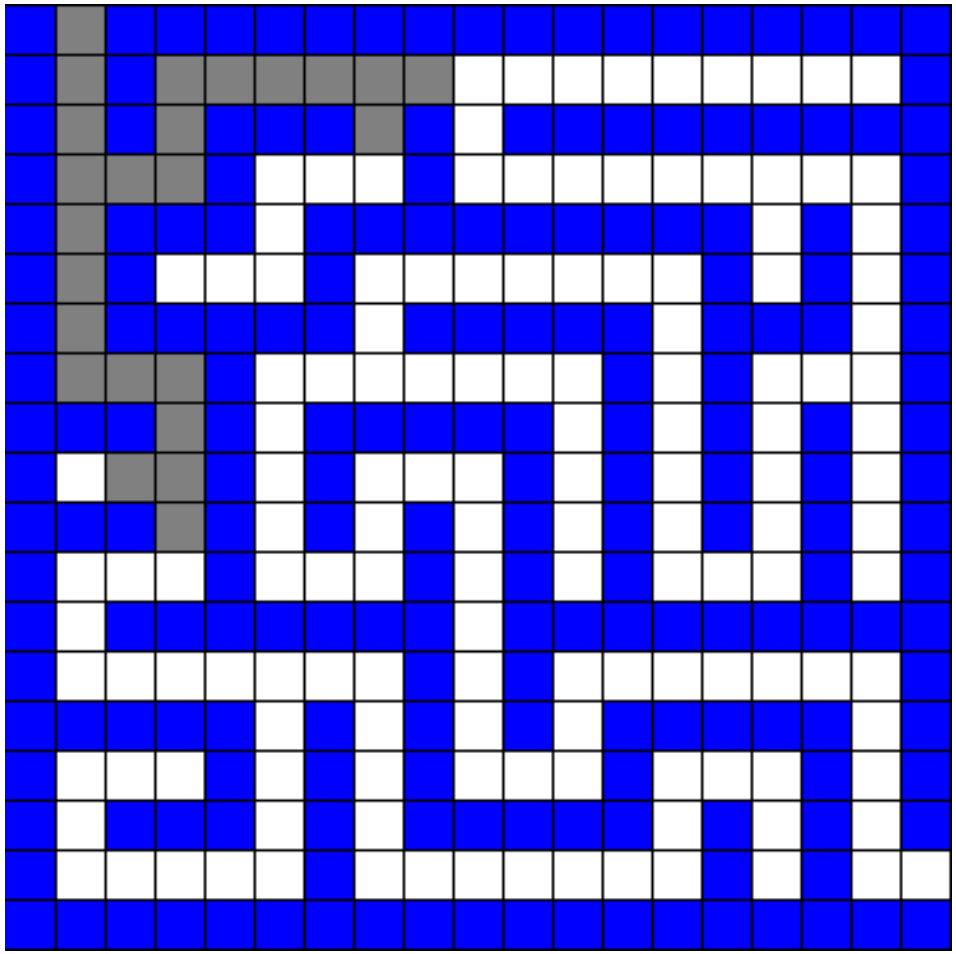
Recorrido de Grafos - BFS



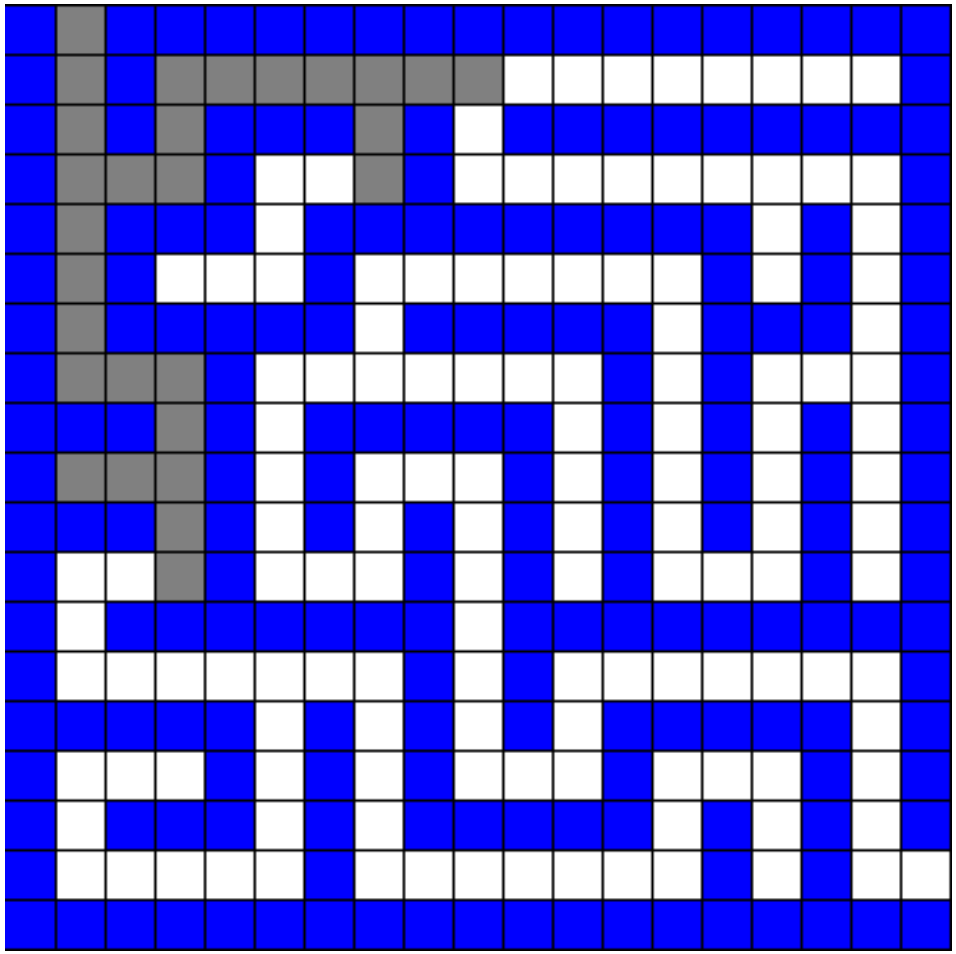
Recorrido de Grafos - BFS



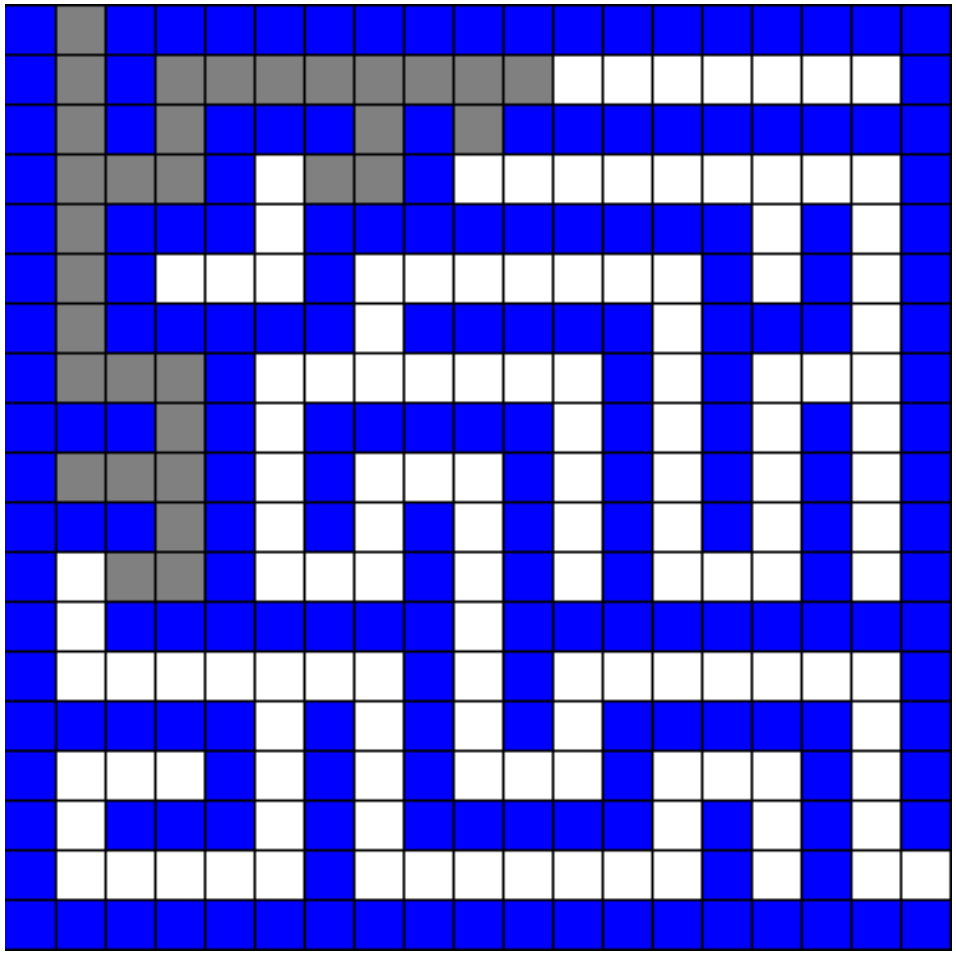
Recorrido de Grafos - BFS



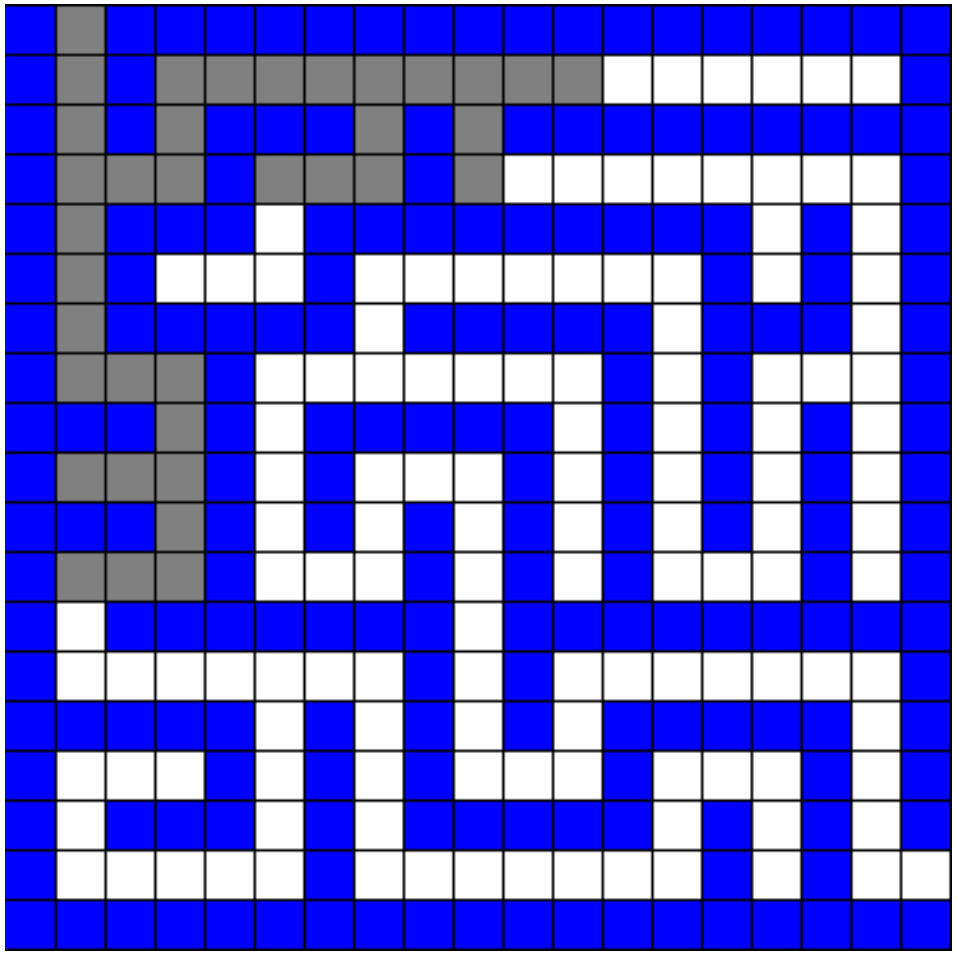
Recorrido de Grafos - BFS



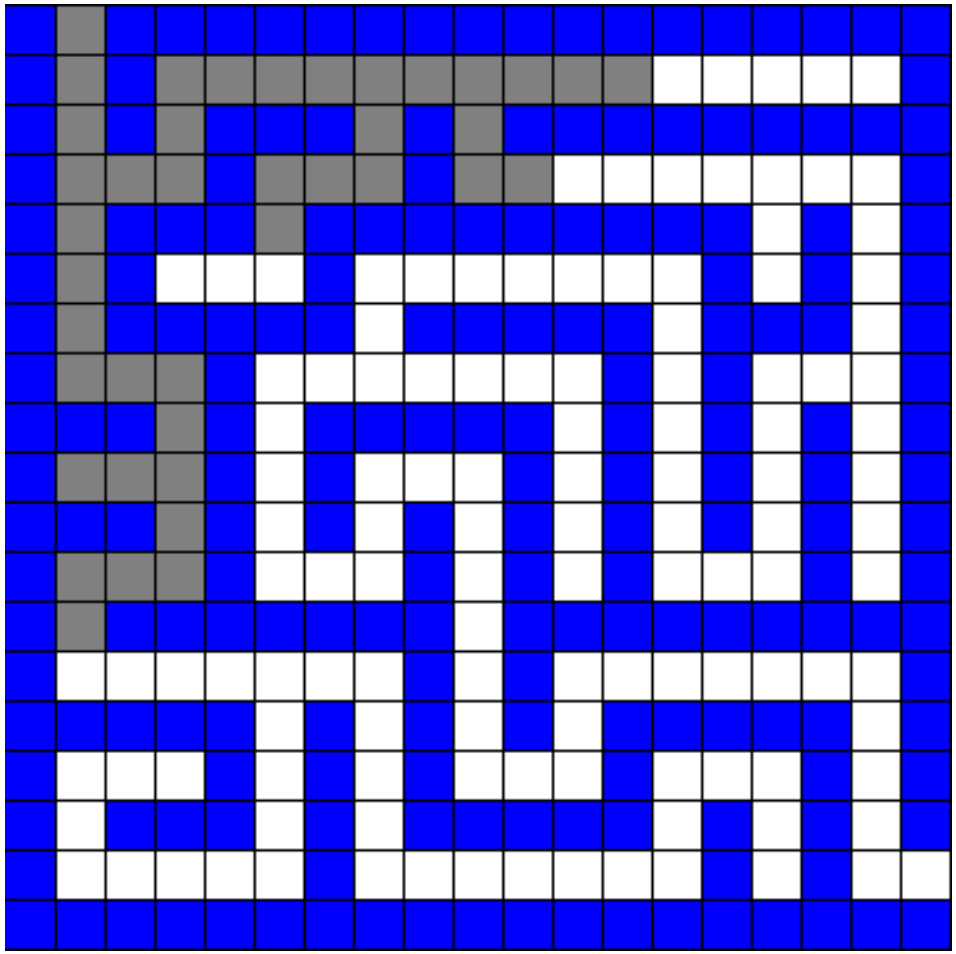
Recorrido de Grafos - BFS



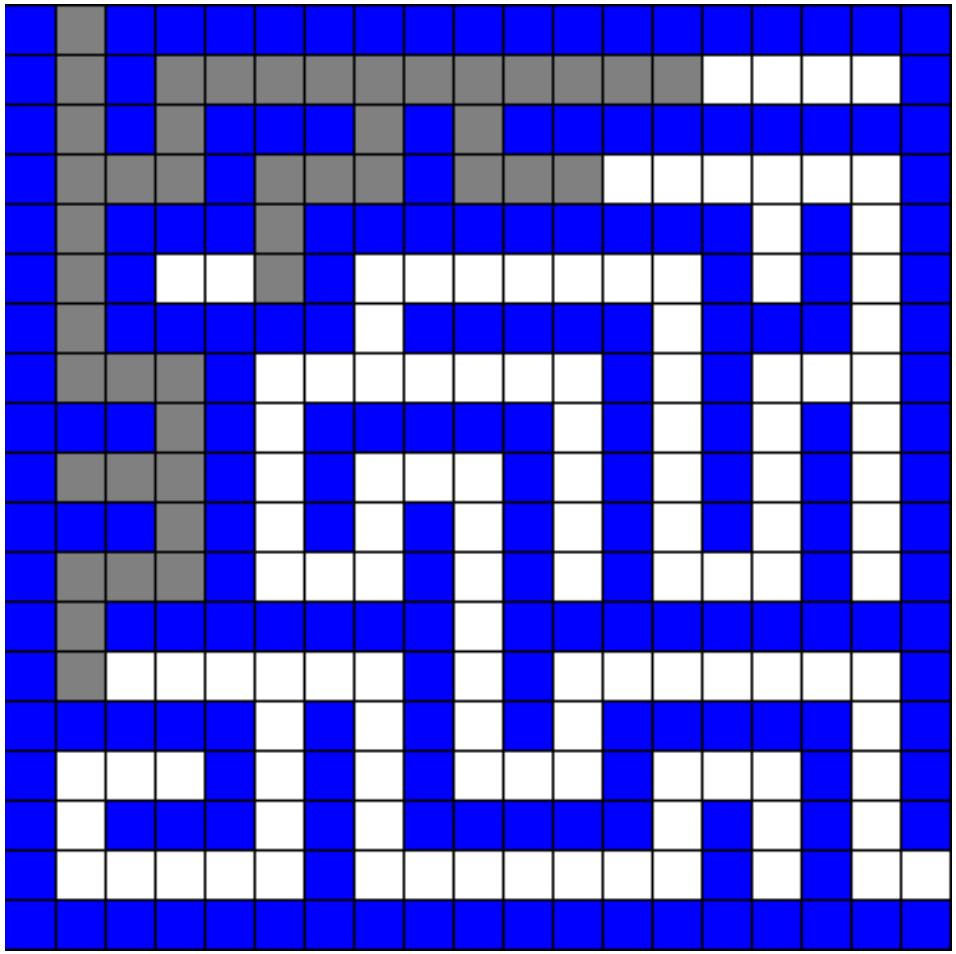
Recorrido de Grafos - BFS



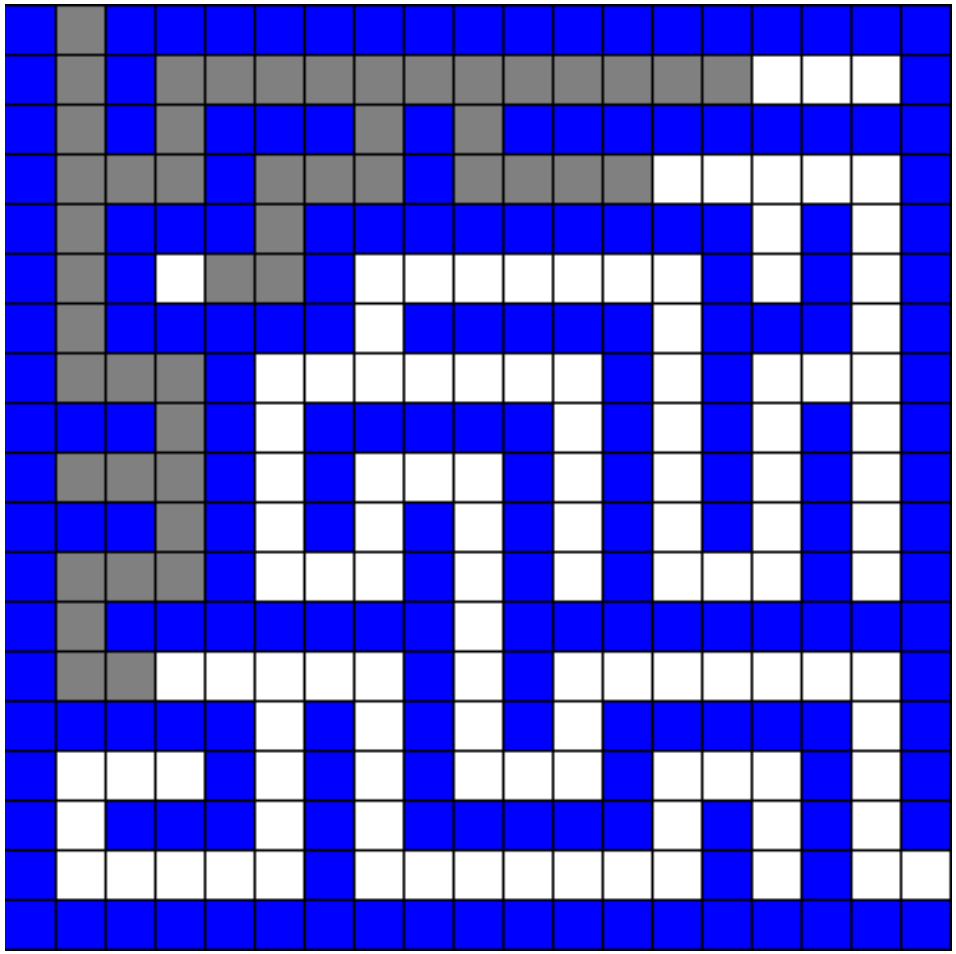
Recorrido de Grafos - BFS



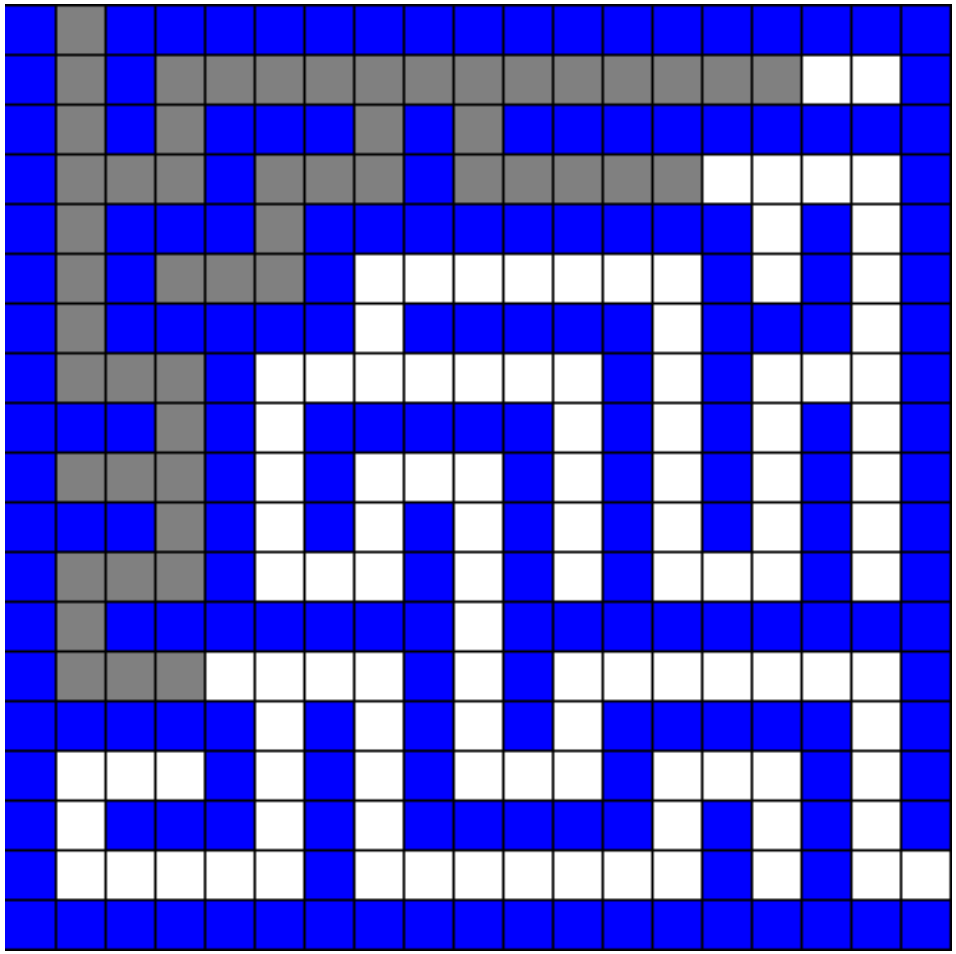
Recorrido de Grafos - BFS



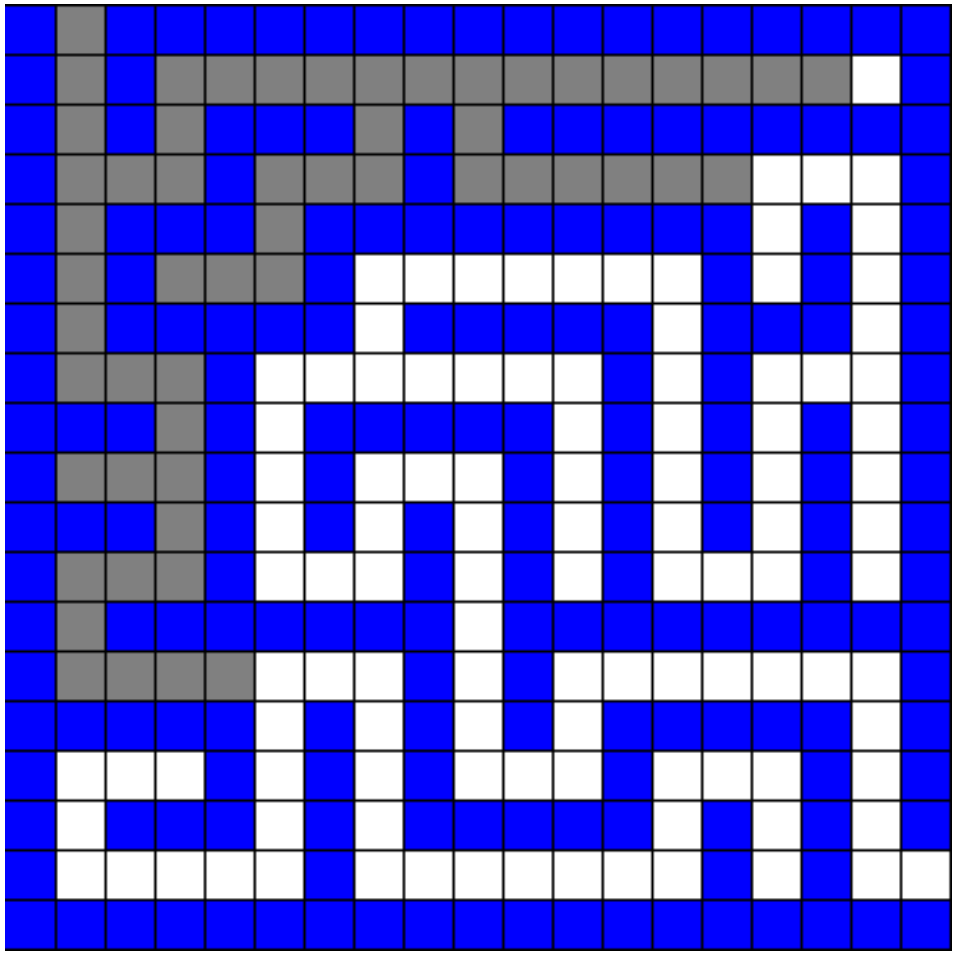
Recorrido de Grafos - BFS



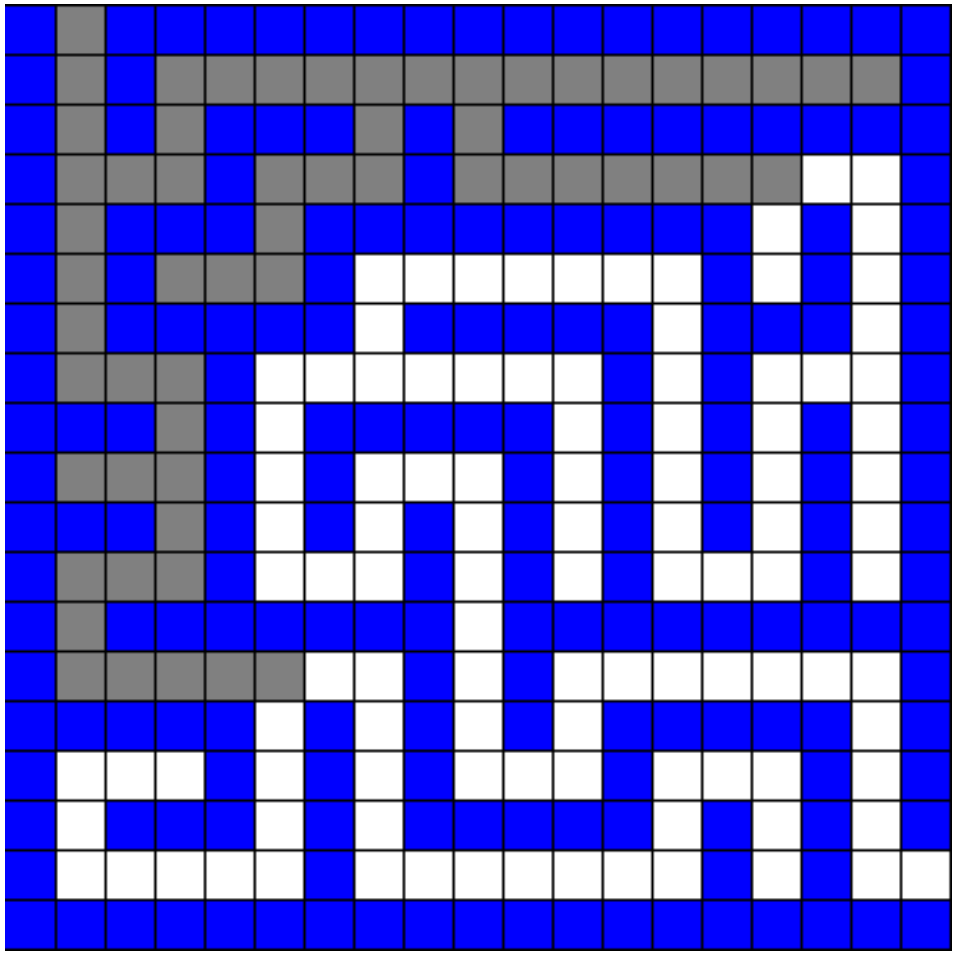
Recorrido de Grafos - BFS



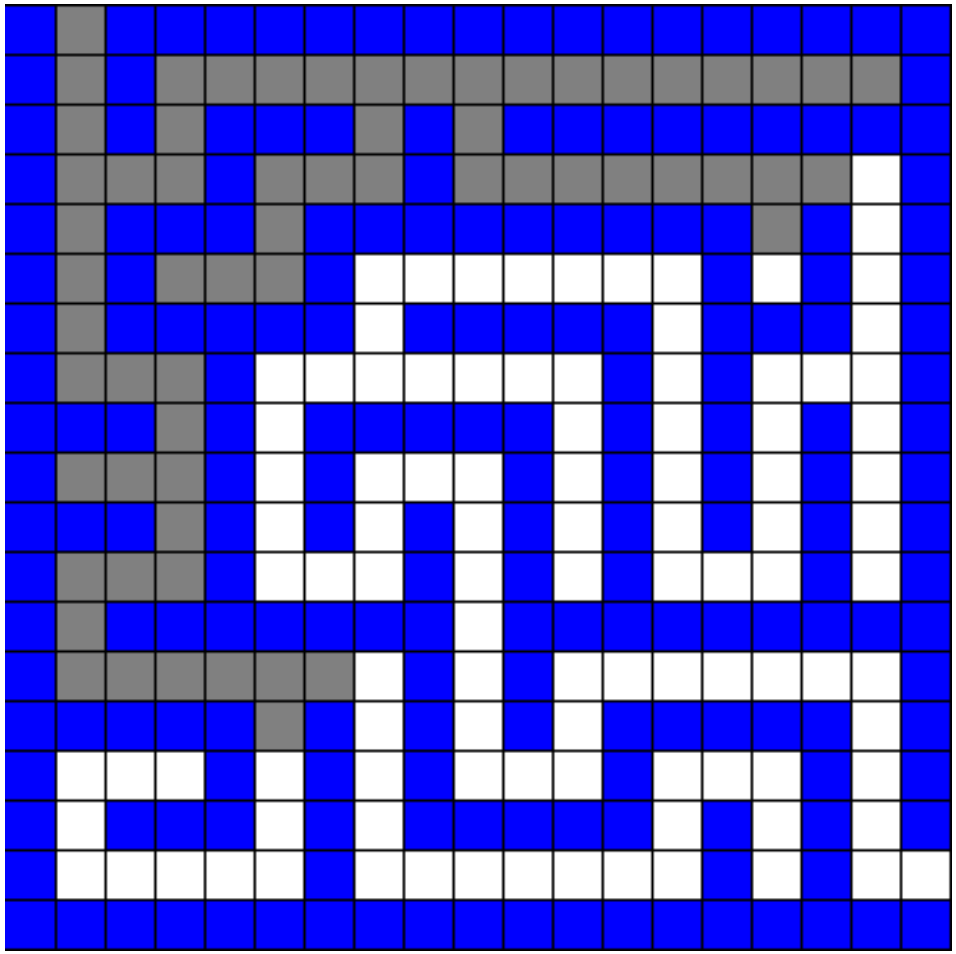
Recorrido de Grafos - BFS



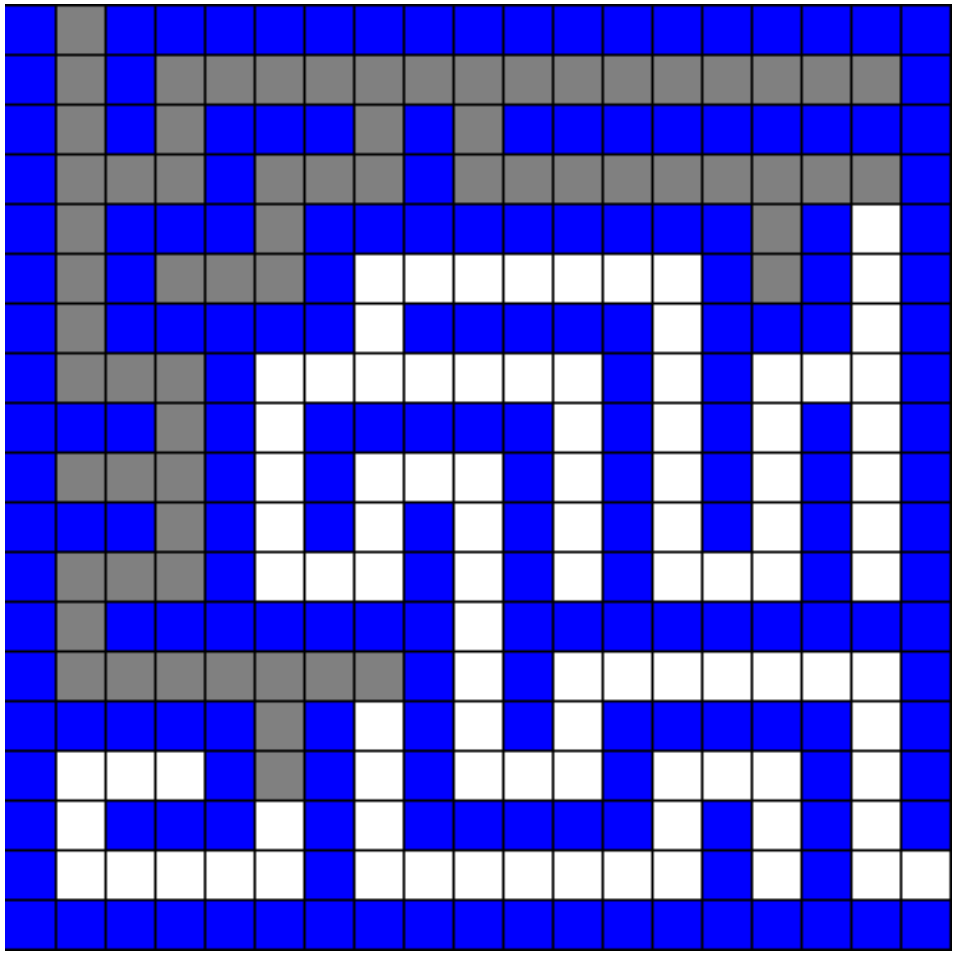
Recorrido de Grafos - BFS



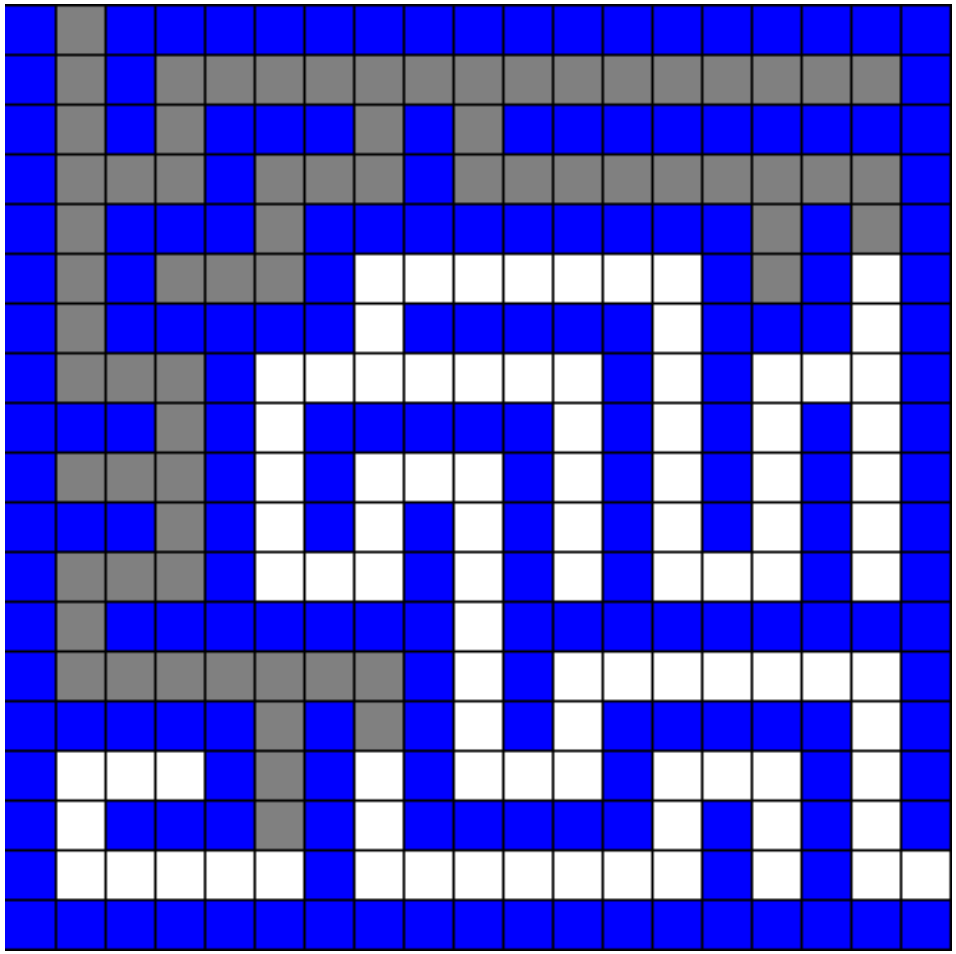
Recorrido de Grafos - BFS



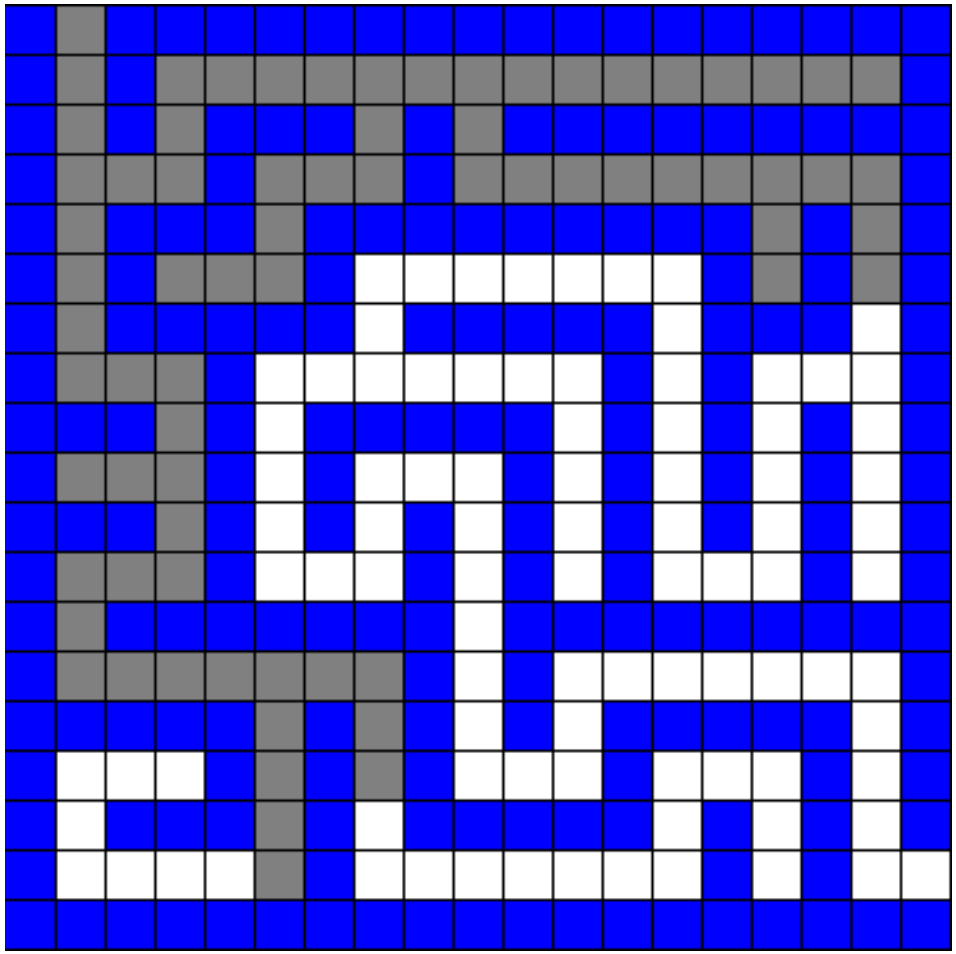
Recorrido de Grafos - BFS



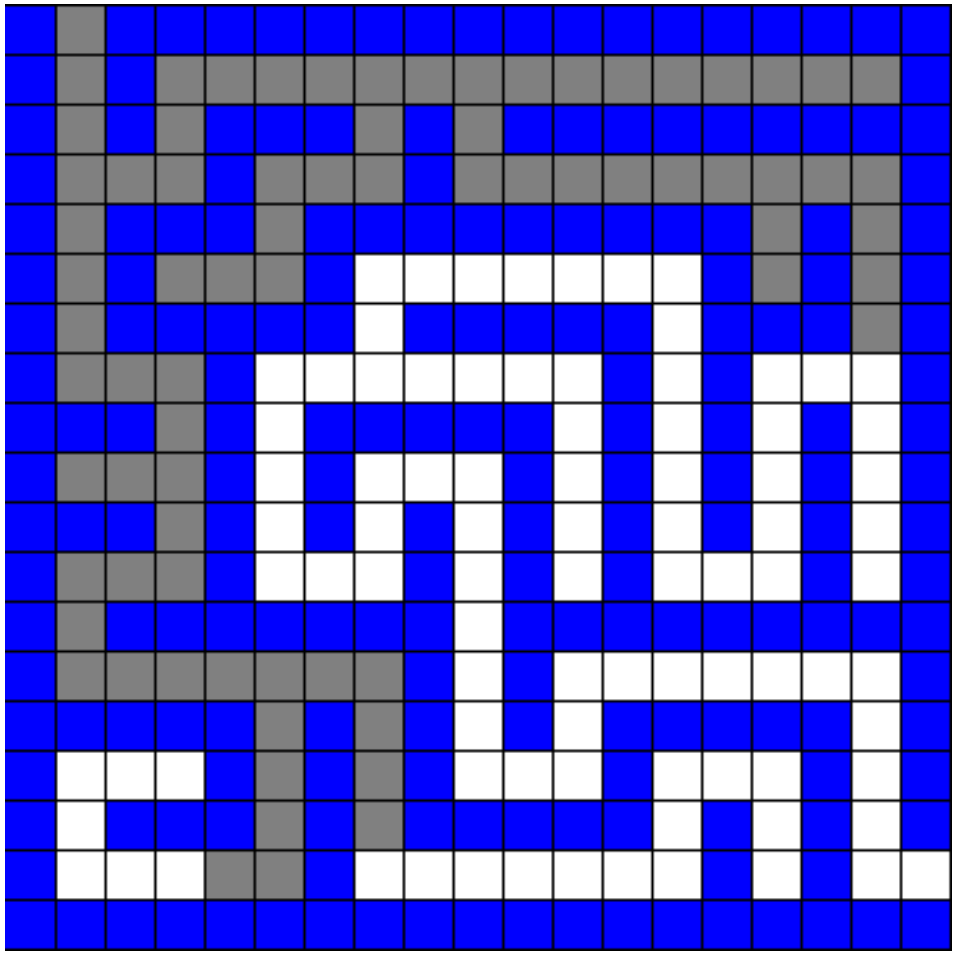
Recorrido de Grafos - BFS



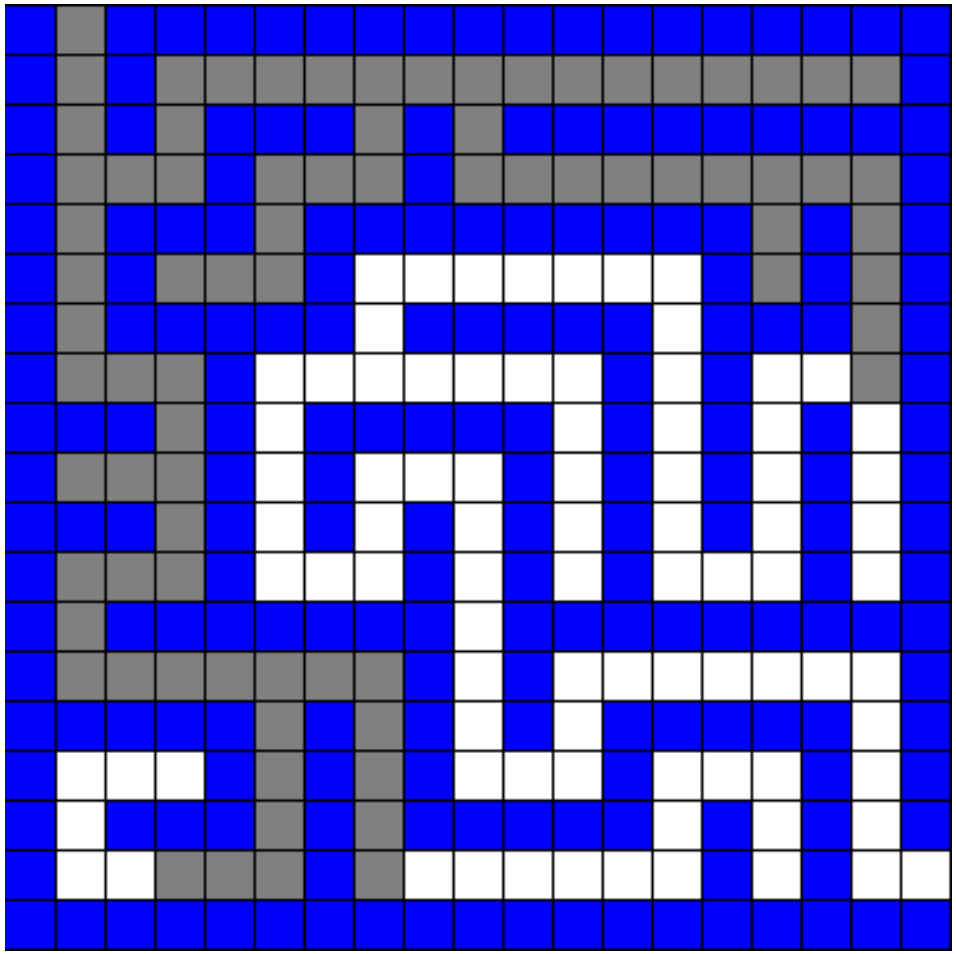
Recorrido de Grafos - BFS



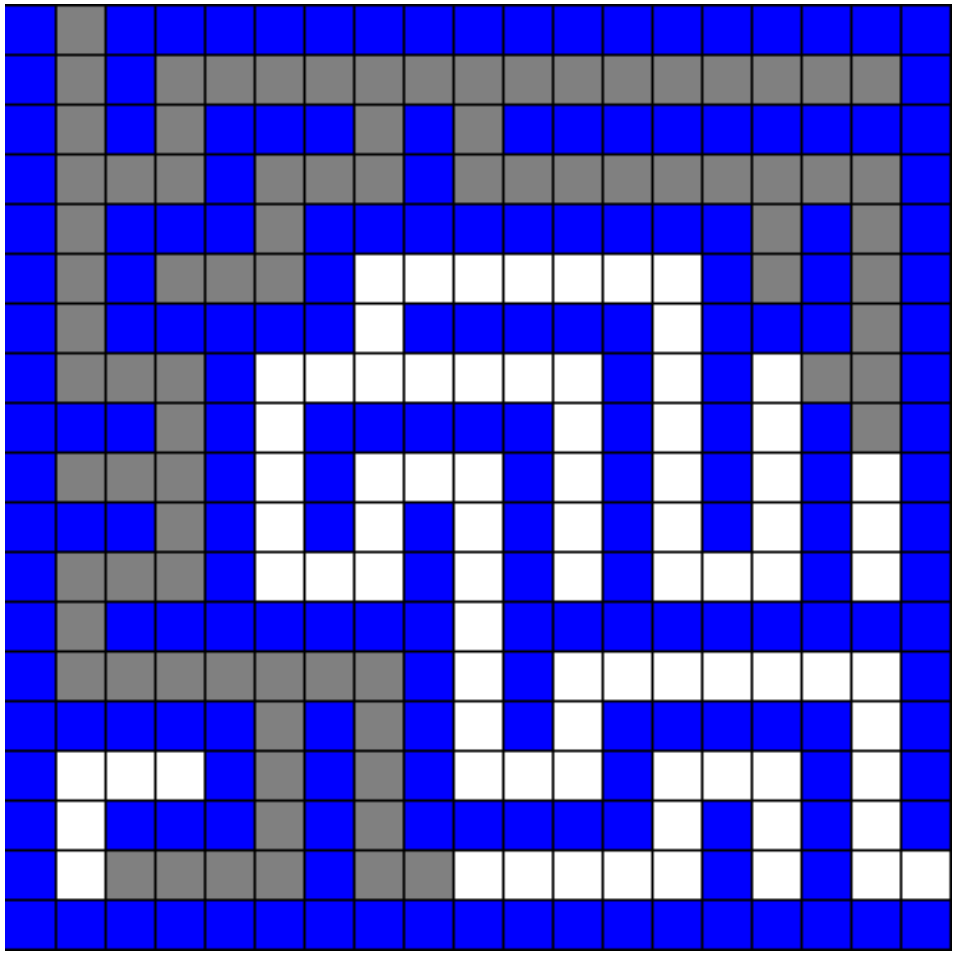
Recorrido de Grafos - BFS



Recorrido de Grafos - BFS



Recorrido de Grafos - BFS



Recorrido de Grafos - BFS

Además de simplemente recorrer un grafo en general, los usos más comunes de BFS (o alguna modificación del mismo) involucran:

Recorrido de Grafos - BFS

Además de simplemente recorrer un grafo en general, los usos más comunes de BFS (o alguna modificación del mismo) involucran:

- Calcular distancias desde una fuente

Recorrido de Grafos - BFS

Además de simplemente recorrer un grafo en general, los usos más comunes de BFS (o alguna modificación del mismo) involucran:

- Calcular distancias desde una fuente
- Chequear si un cierto grafo es bipartito

Recorrido de Grafos - BFS

Además de simplemente recorrer un grafo en general, los usos más comunes de BFS (o alguna modificación del mismo) involucran:

- Calcular distancias desde una fuente
- Chequear si un cierto grafo es bipartito
- Calcular componentes conexas

Recorrido de Grafos - DFS

Veamos otra forma de recorrer a un grafo. Vamos a distinguir a un nodo en particular que indica el nodo actual del recorrido. Llamémosle v a dicho nodo.

Recorrido de Grafos - DFS

Veamos otra forma de recorrer a un grafo. Vamos a distinguir a un nodo en particular que indica el nodo actual del recorrido. Llamémosle v a dicho nodo.
En un recorrido **DFS** (*Depth-First Search*) comenzamos visitando el nodo v, luego visitamos a algún vecino y cuando vemos a un vecino sin visitar seguimos un recorrido DFS desde allí. Así hasta que todos los vecinos de v sean visitados.
Una vez que todos los vecinos de v están visitados, se da por finalizado el recorrido desde v.

Recorrido de Grafos - DFS

Veamos otra forma de recorrer a un grafo. Vamos a distinguir a un nodo en particular que indica el nodo actual del recorrido. Llamémosle v a dicho nodo.
En un recorrido **DFS** (*Depth-First Search*) comenzamos visitando el nodo v, luego visitamos a algún vecino y cuando vemos a un vecino sin visitar seguimos un recorrido DFS desde allí. Así hasta que todos los vecinos de v sean visitados.
Una vez que todos los vecinos de v están visitados, se da por finalizado el recorrido desde v.
De alguna forma, podemos pensar que se va expandiendo ordenadamente en una dirección todo lo que se puede.

Algoritmo DFS

Por la naturaleza del algoritmo, las implementaciones usuales del recorrido DFS de un grafo son recursivas. A lo largo del algoritmo utilizaremos el arreglo auxiliar `visto` de manera similar.

Algoritmo DFS

Por la naturaleza del algoritmo, las implementaciones usuales del recorrido DFS de un grafo son recursivas. A lo largo del algoritmo utilizaremos el arreglo auxiliar `visto` de manera similar.

DFS($\mathcal{G}$): ($\mathcal{G}$ es nuestro grafo en alguna representación)

1 para todo vértice v :
  1.1 `visto`$[v] \leftarrow$ *Falso*
2 para todo vértice v :
  2.1 Si `visto`$[v] ==$ *Falso* :
    2.1.1 DFS-VISITA($\mathcal{G}, v$)

Algoritmo DFS

Por la naturaleza del algoritmo, las implementaciones usuales del recorrido DFS de un grafo son recursivas. A lo largo del algoritmo utilizaremos el arreglo auxiliar `visto` de manera similar.

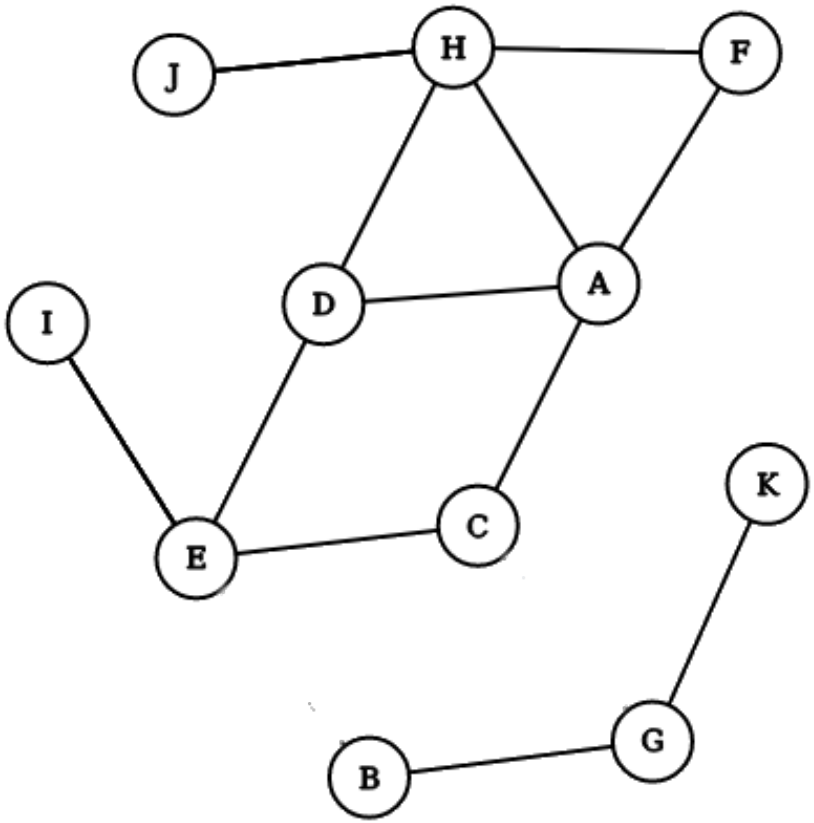
DFS($\mathcal{G}$): ($\mathcal{G}$ es nuestro grafo en alguna representación)

1 para todo vértice v :
  1.1 `visto`$[v] \leftarrow$ *Falso*
2 para todo vértice v :
  2.1 Si `visto`$[v] ==$ *Falso* :
    2.1.1 DFS-VISITA($\mathcal{G}, v$)

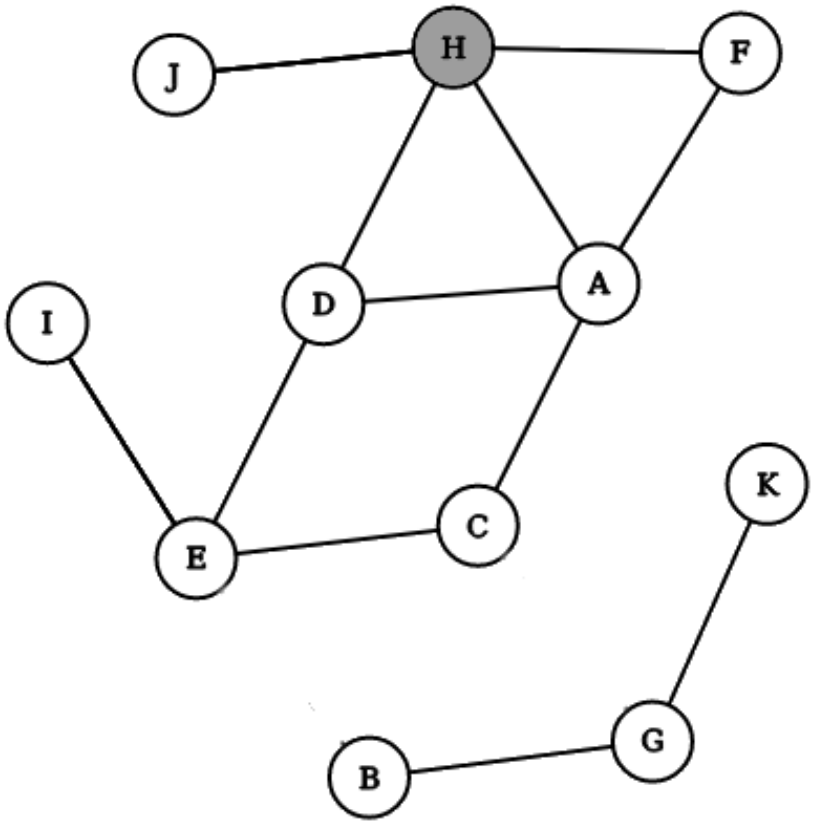
DFS-VISITA($\mathcal{G}, v$):

1 `visto`$[v] \leftarrow$ *Verdadero*
2 para todo vecino w de v con `visto`$[w] ==$ *Falso*:
  2.1 DFS-VISITA($\mathcal{G}, w$)

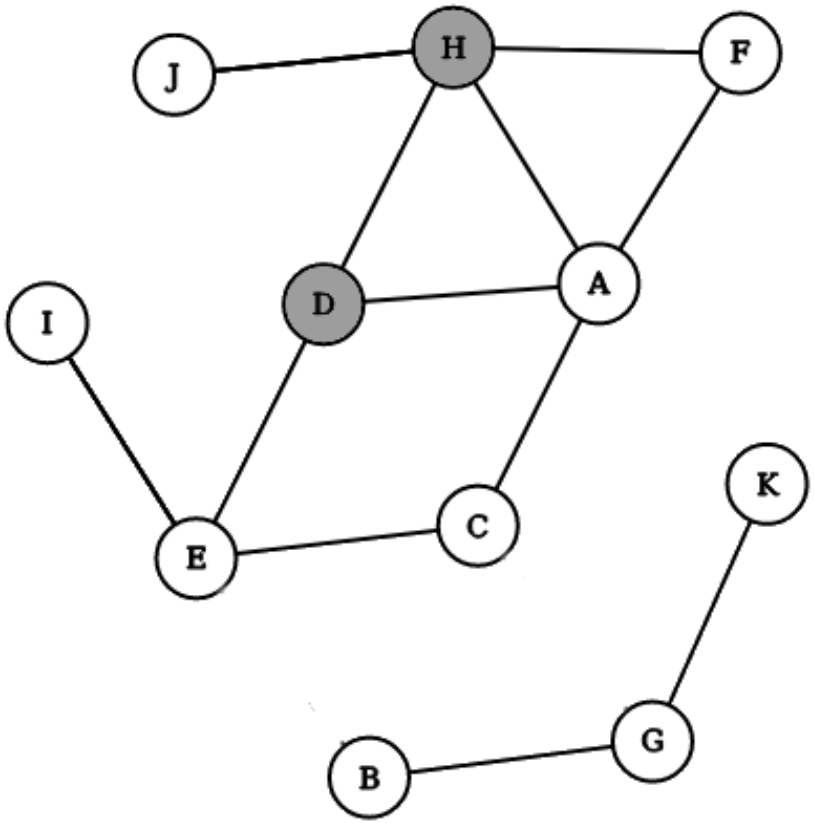
Recorrido de Grafos - DFS



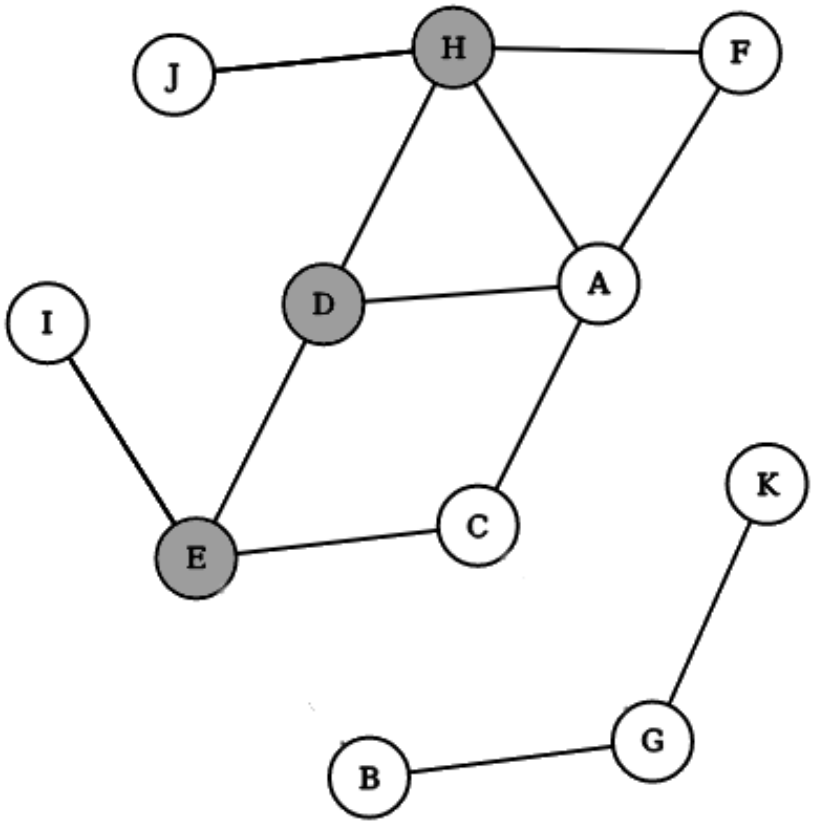
Recorrido de Grafos - DFS



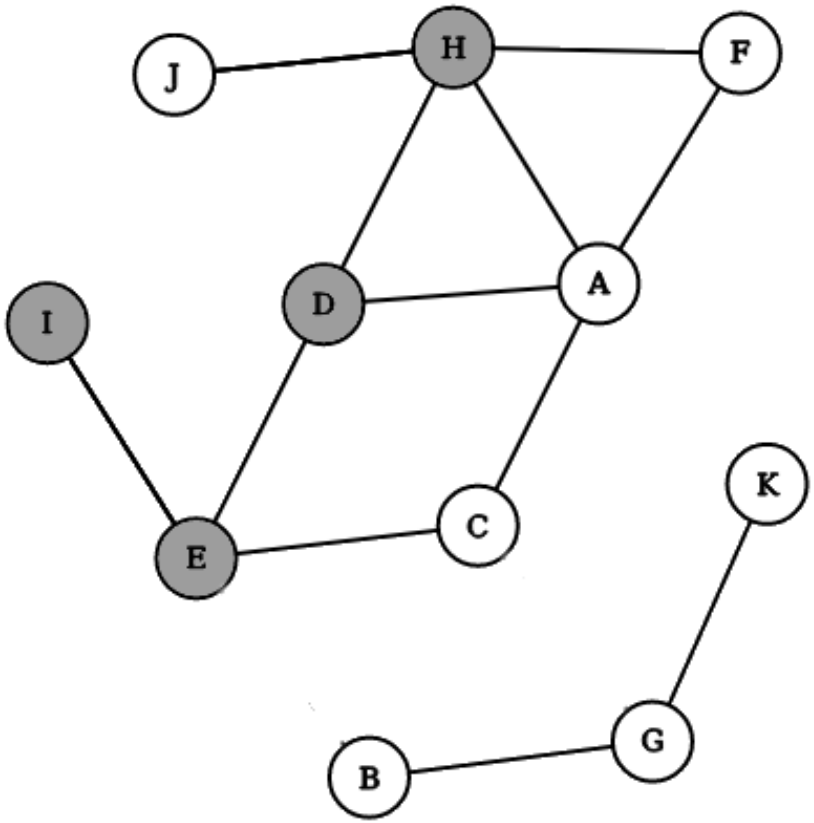
Recorrido de Grafos - DFS



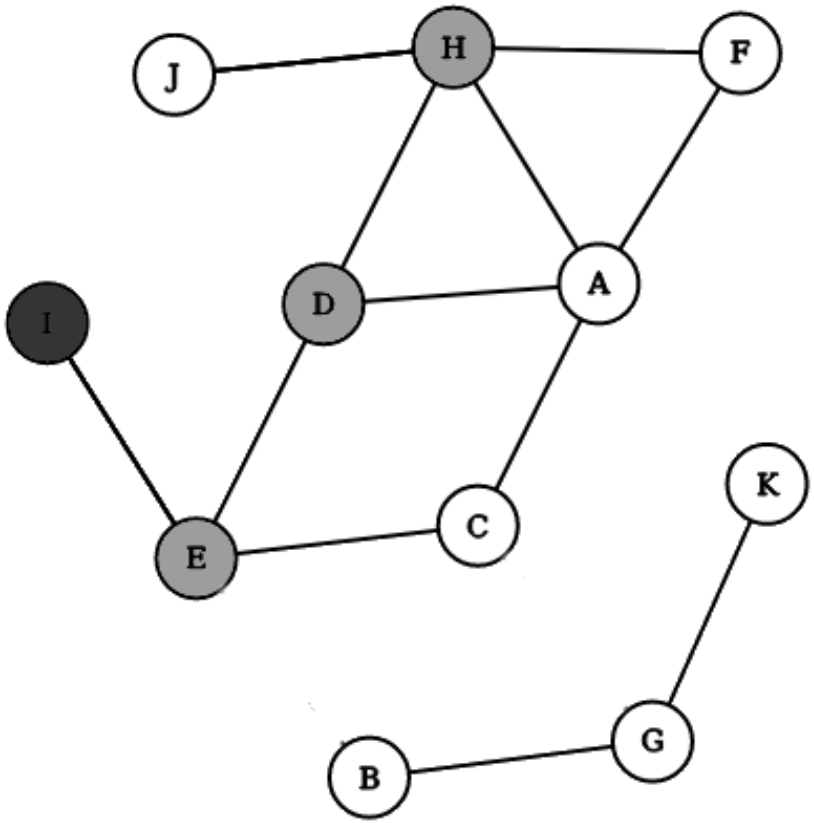
Recorrido de Grafos - DFS



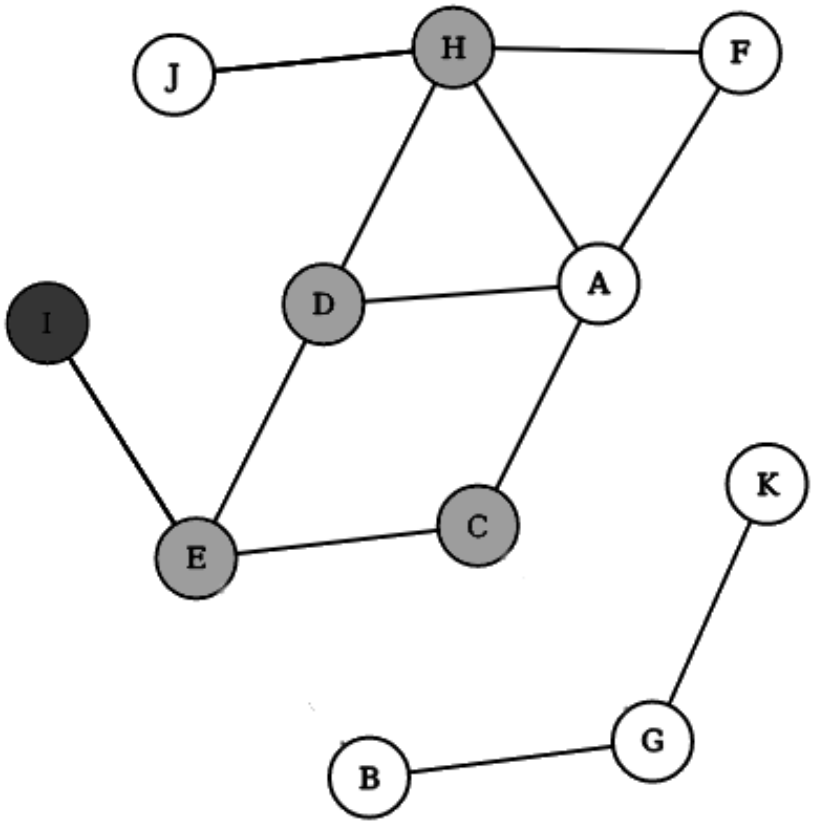
Recorrido de Grafos - DFS



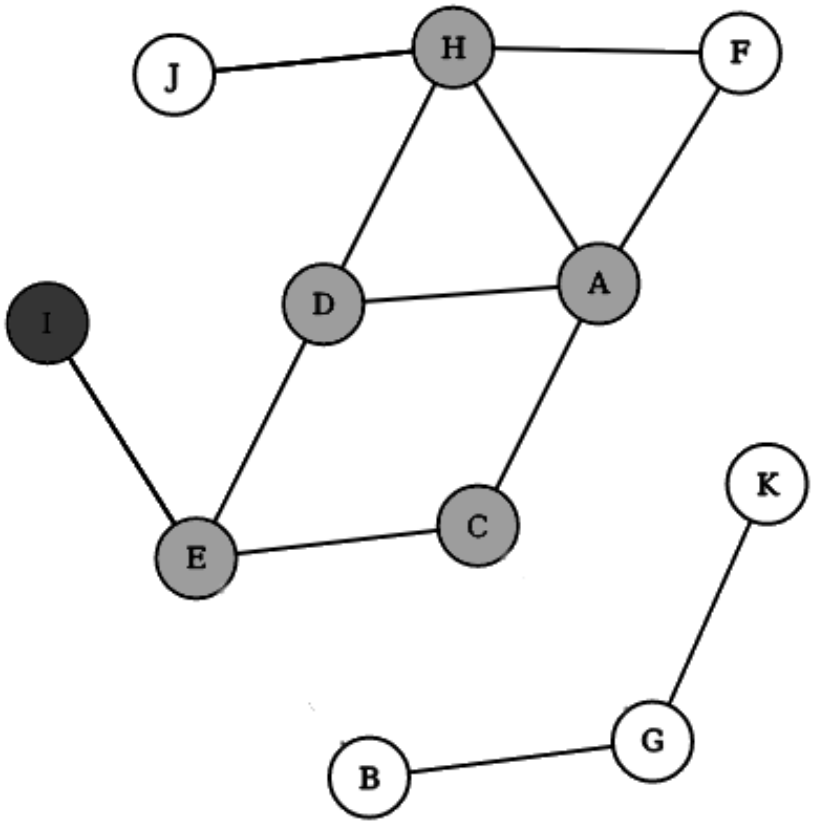
Recorrido de Grafos - DFS



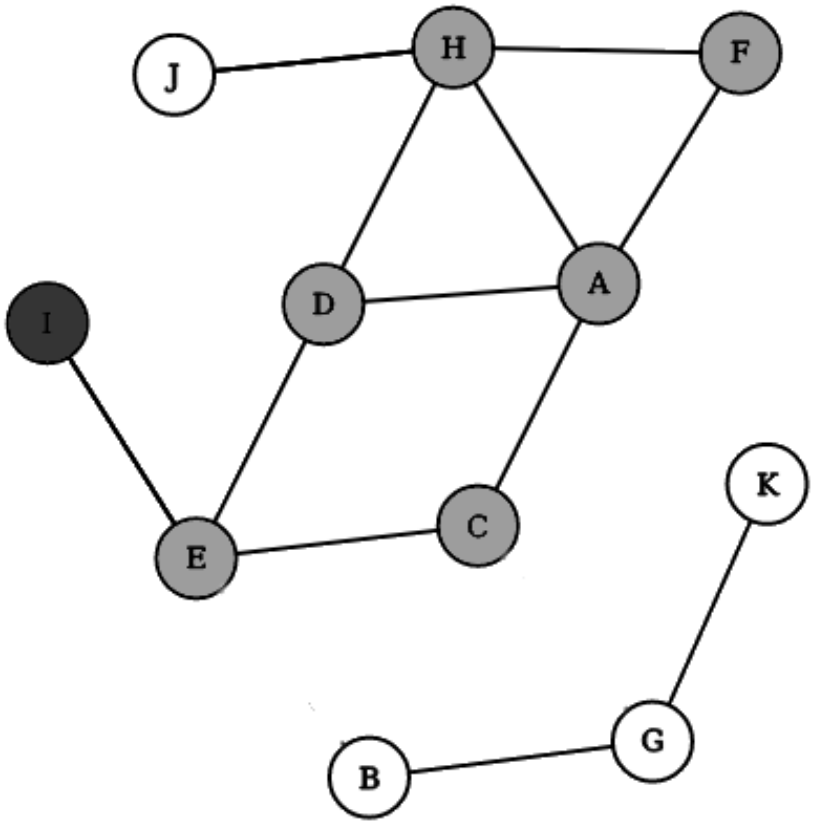
Recorrido de Grafos - DFS



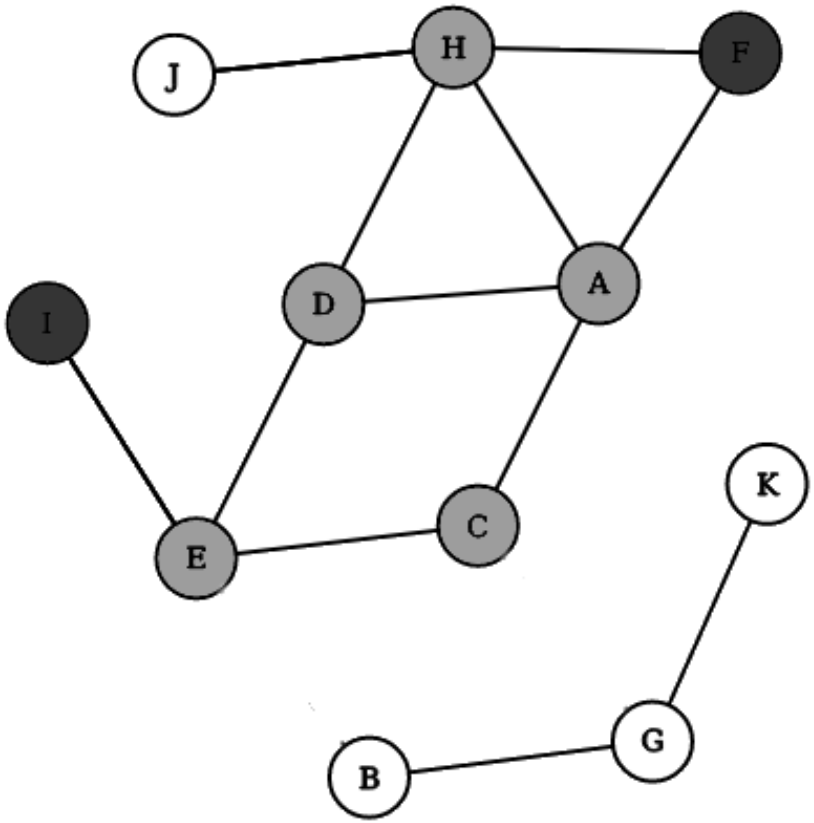
Recorrido de Grafos - DFS



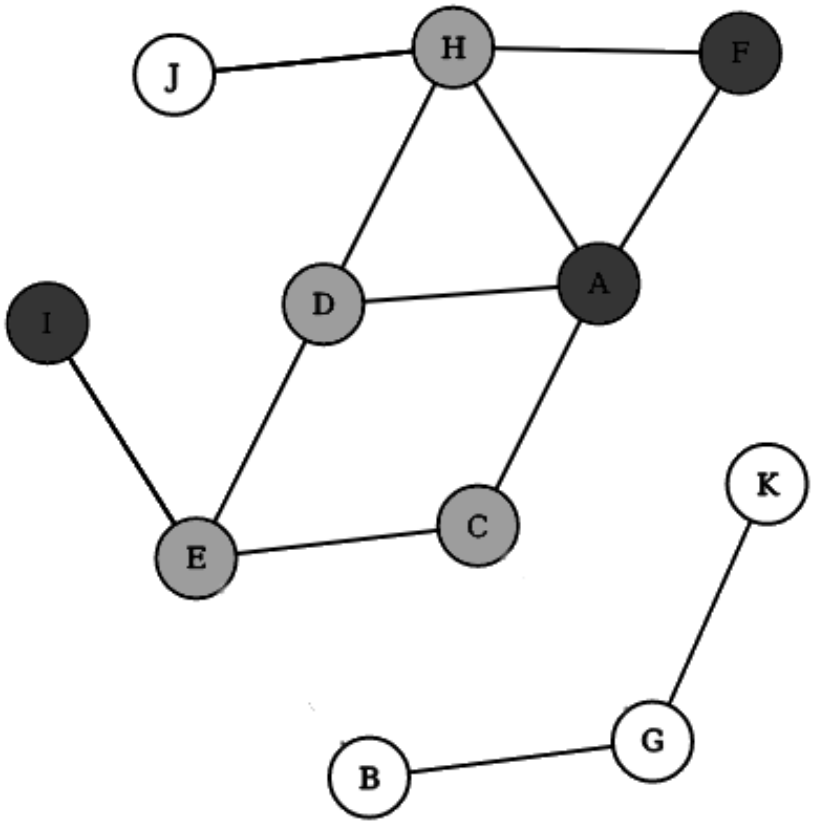
Recorrido de Grafos - DFS



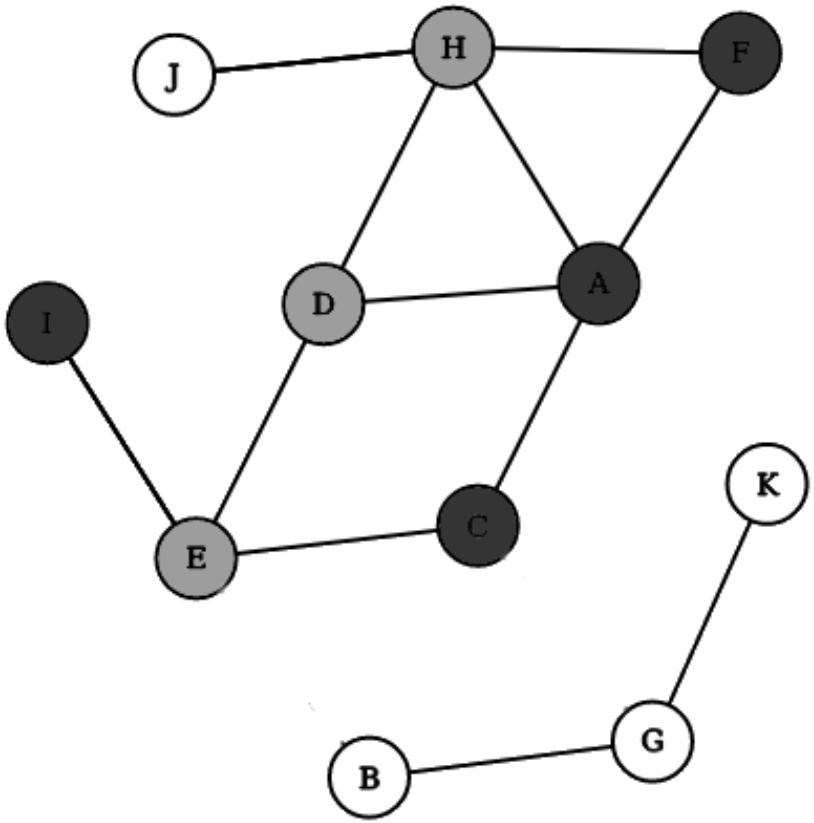
Recorrido de Grafos - DFS



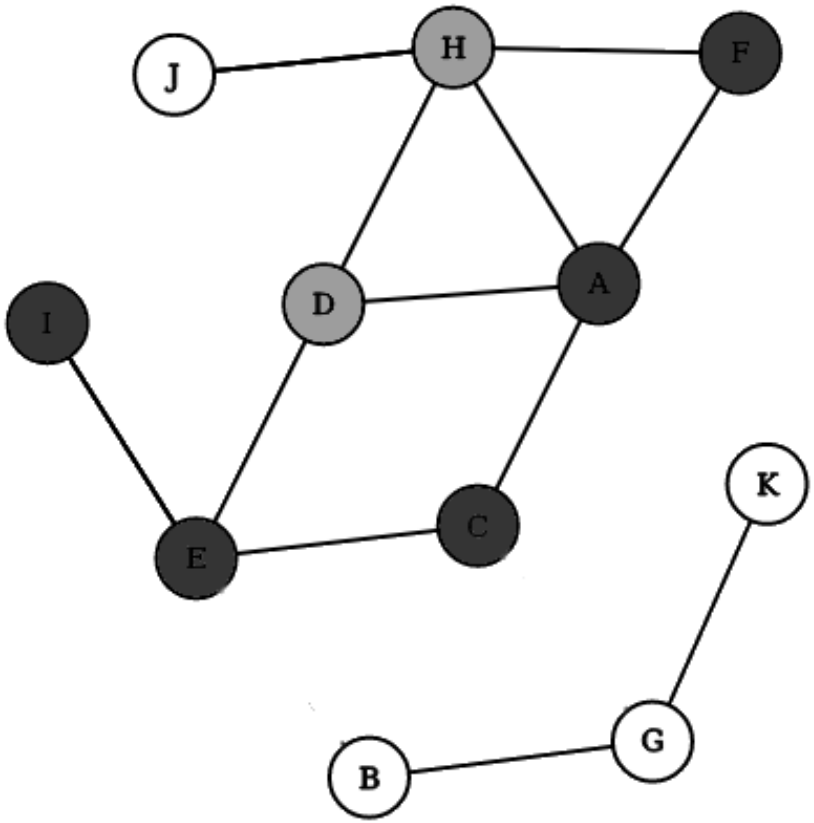
Recorrido de Grafos - DFS



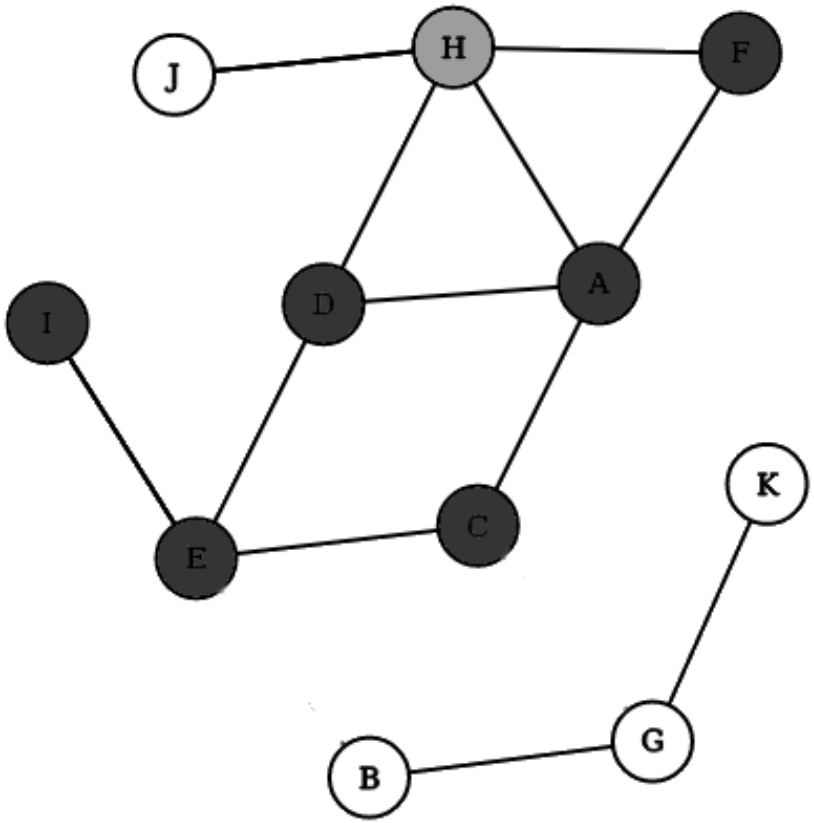
Recorrido de Grafos - DFS



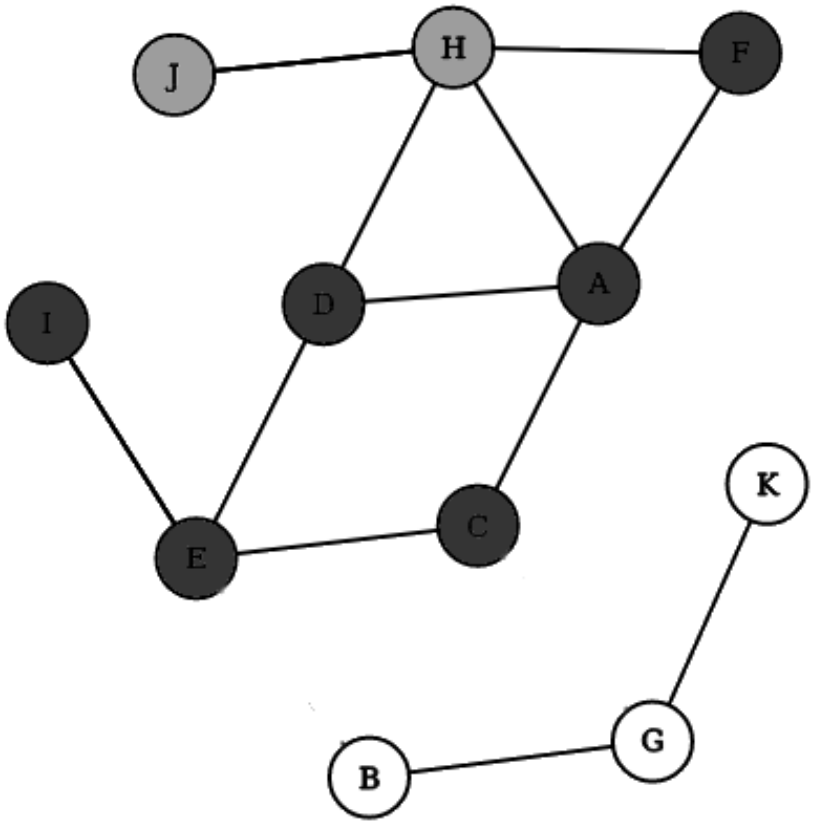
Recorrido de Grafos - DFS



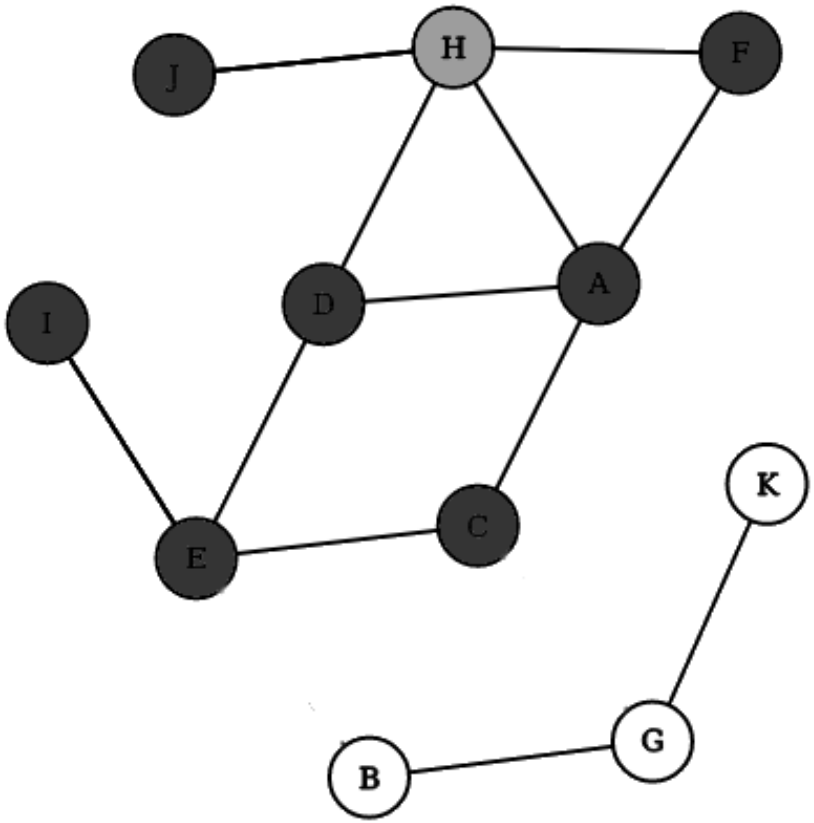
Recorrido de Grafos - DFS



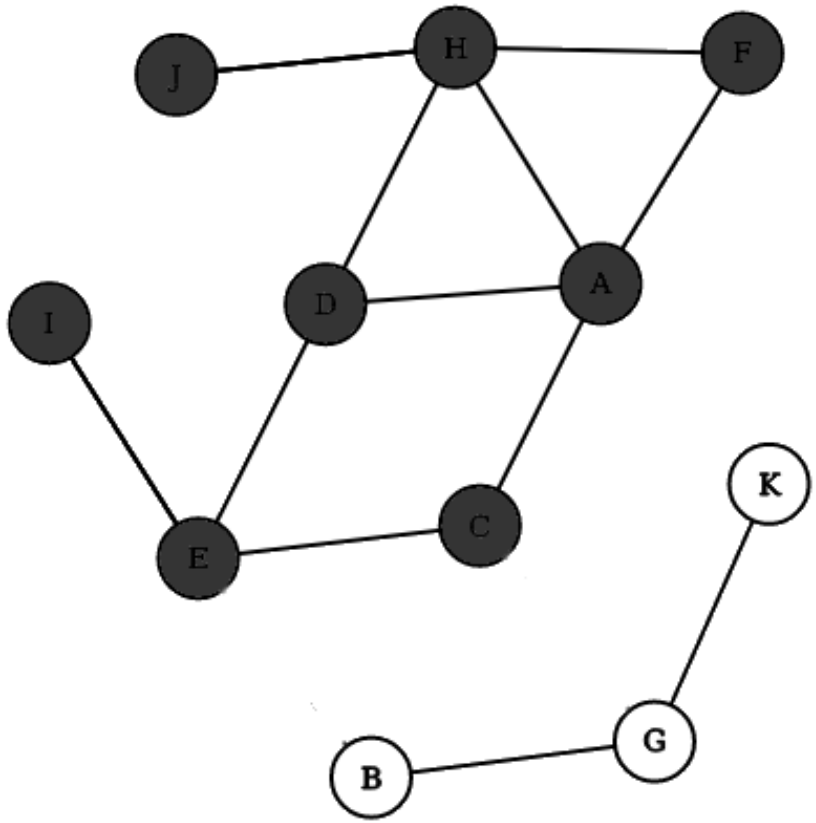
Recorrido de Grafos - DFS



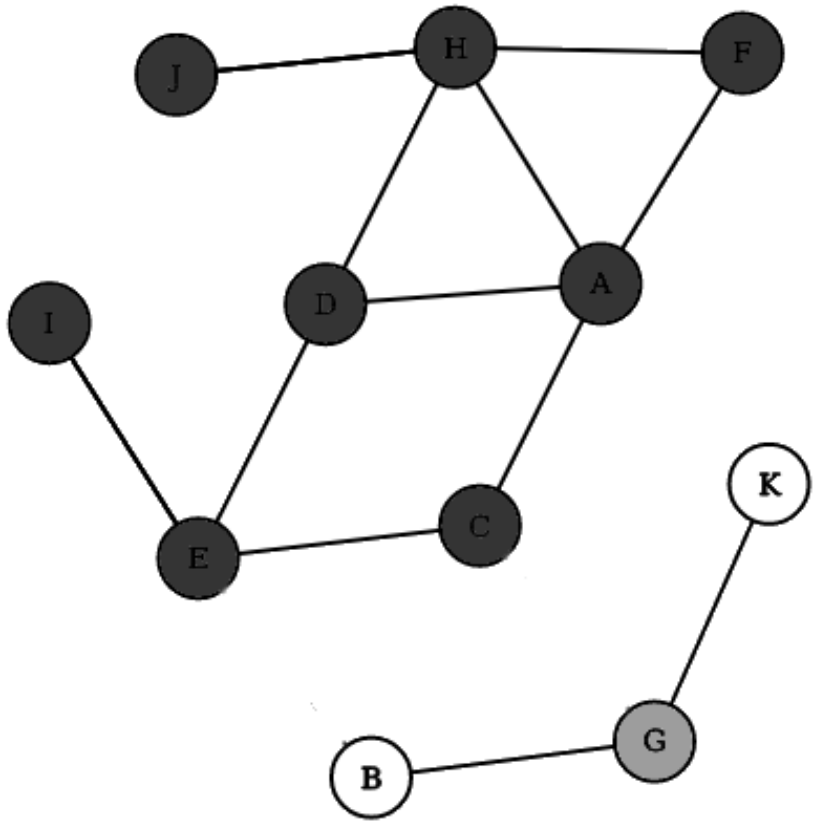
Recorrido de Grafos - DFS



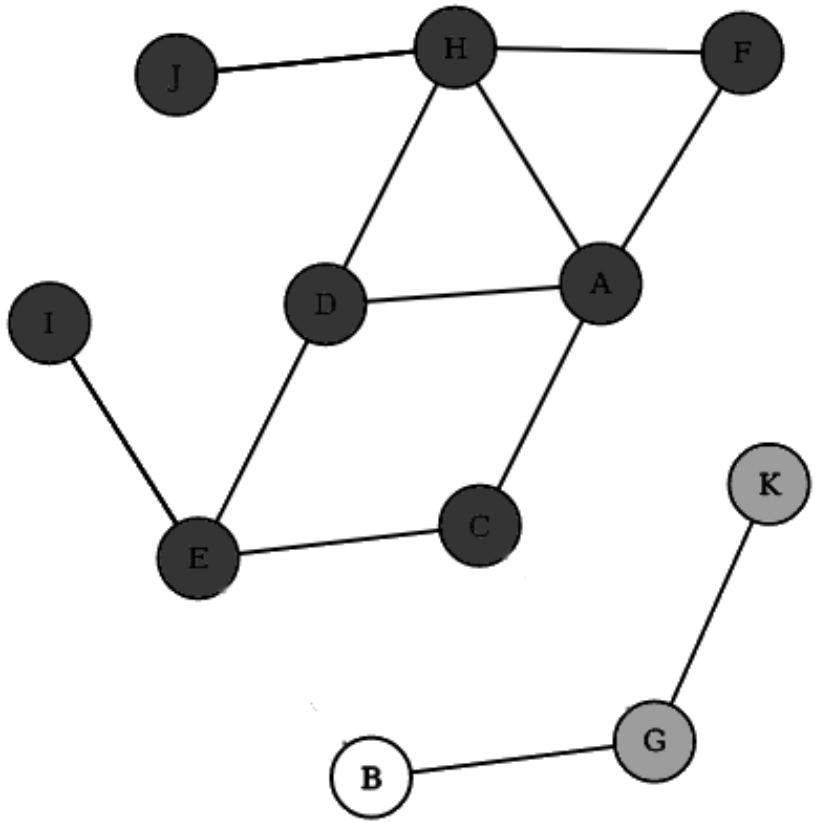
Recorrido de Grafos - DFS



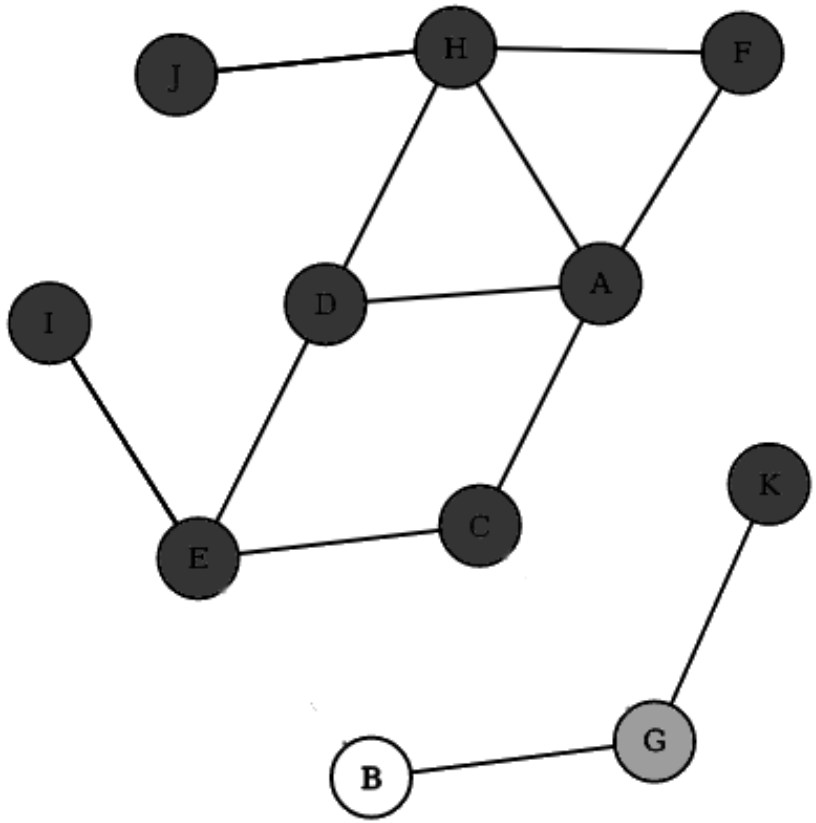
Recorrido de Grafos - DFS



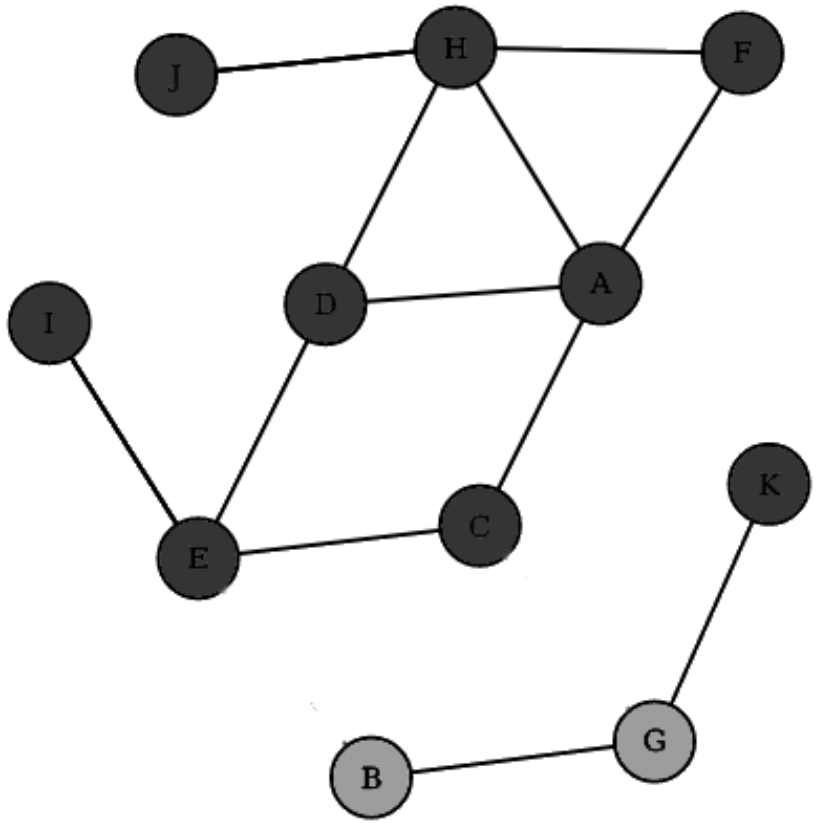
Recorrido de Grafos - DFS



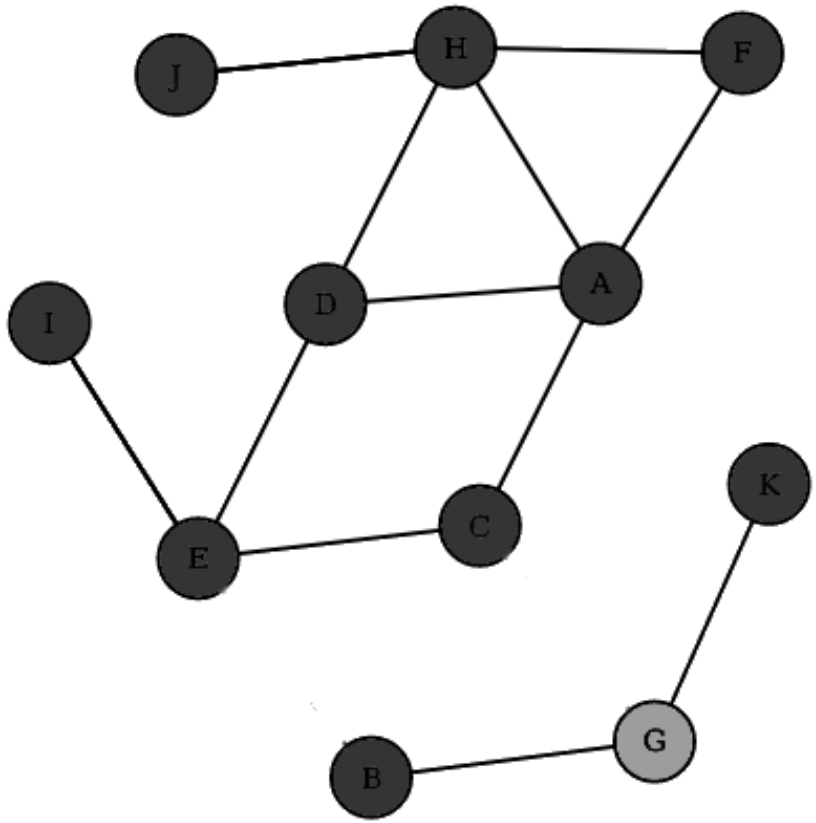
Recorrido de Grafos - DFS



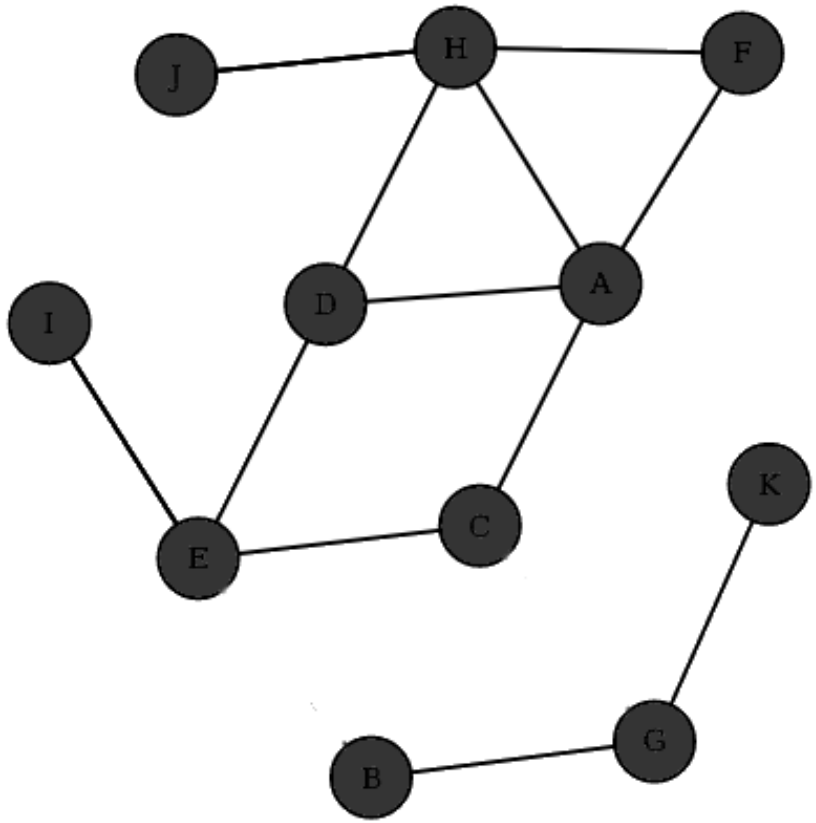
Recorrido de Grafos - DFS



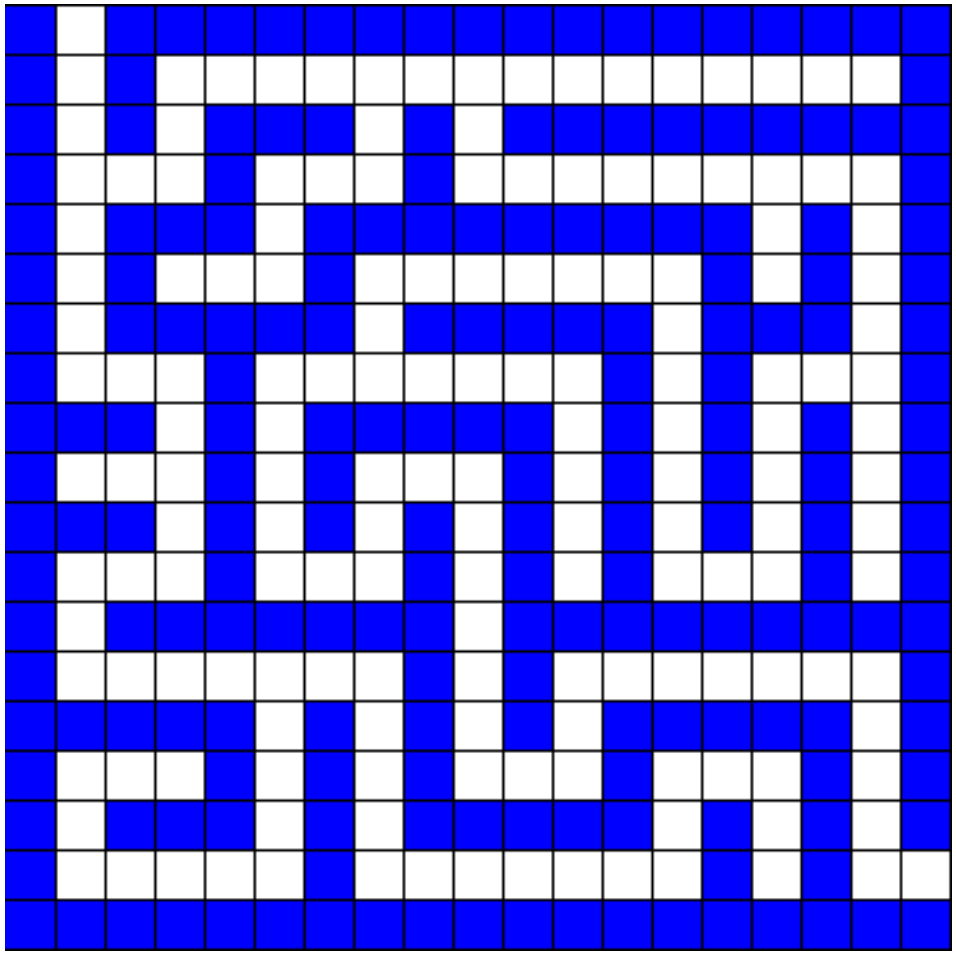
Recorrido de Grafos - DFS



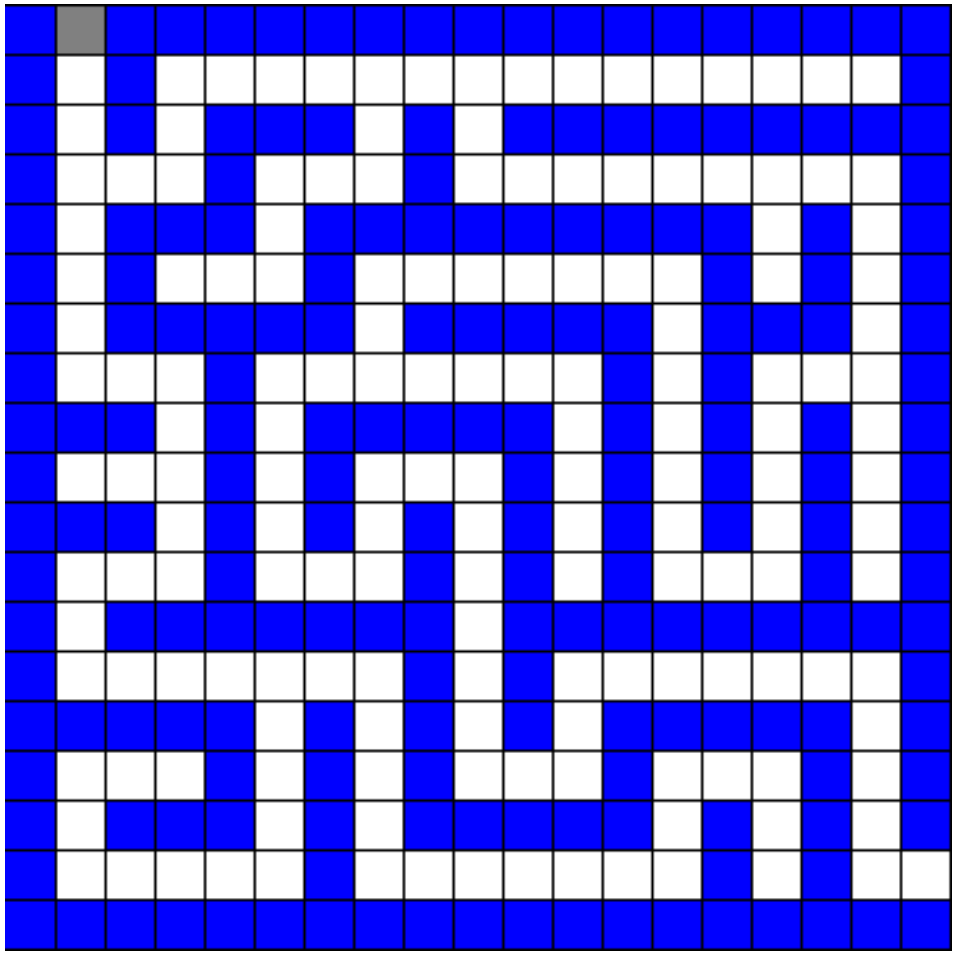
Recorrido de Grafos - DFS



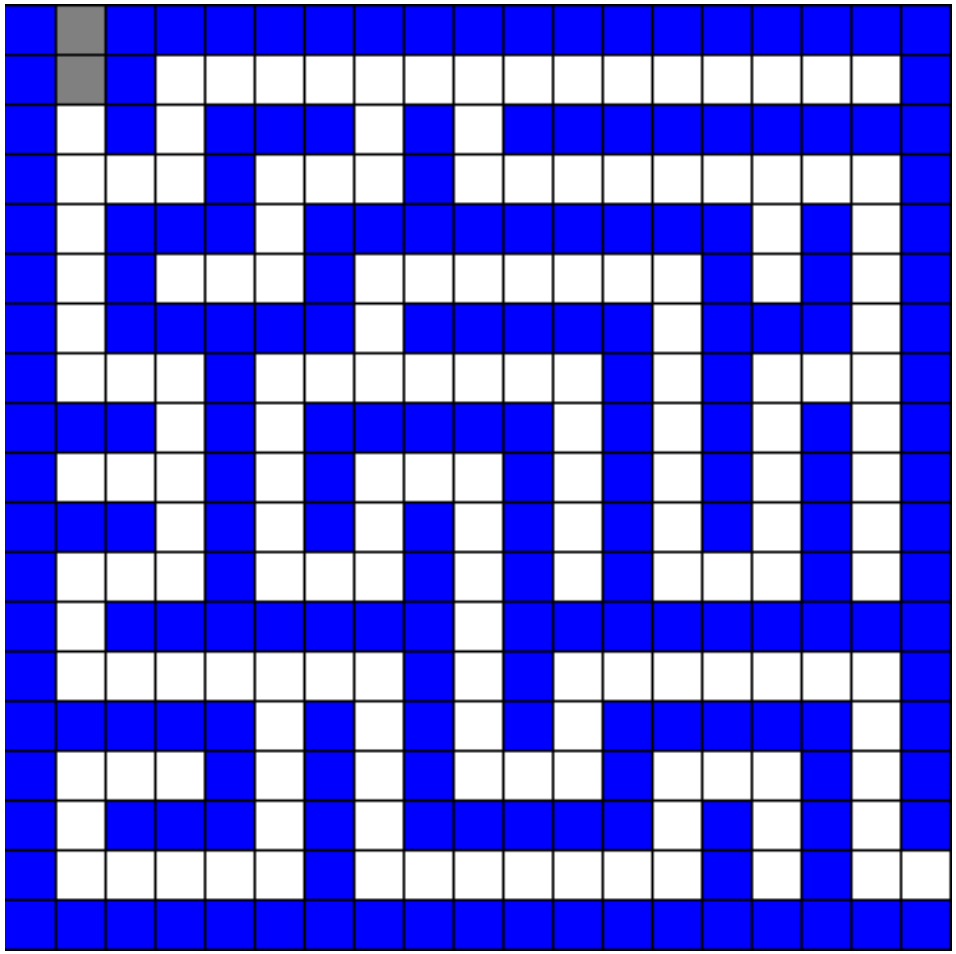
Recorrido de Grafos - DFS



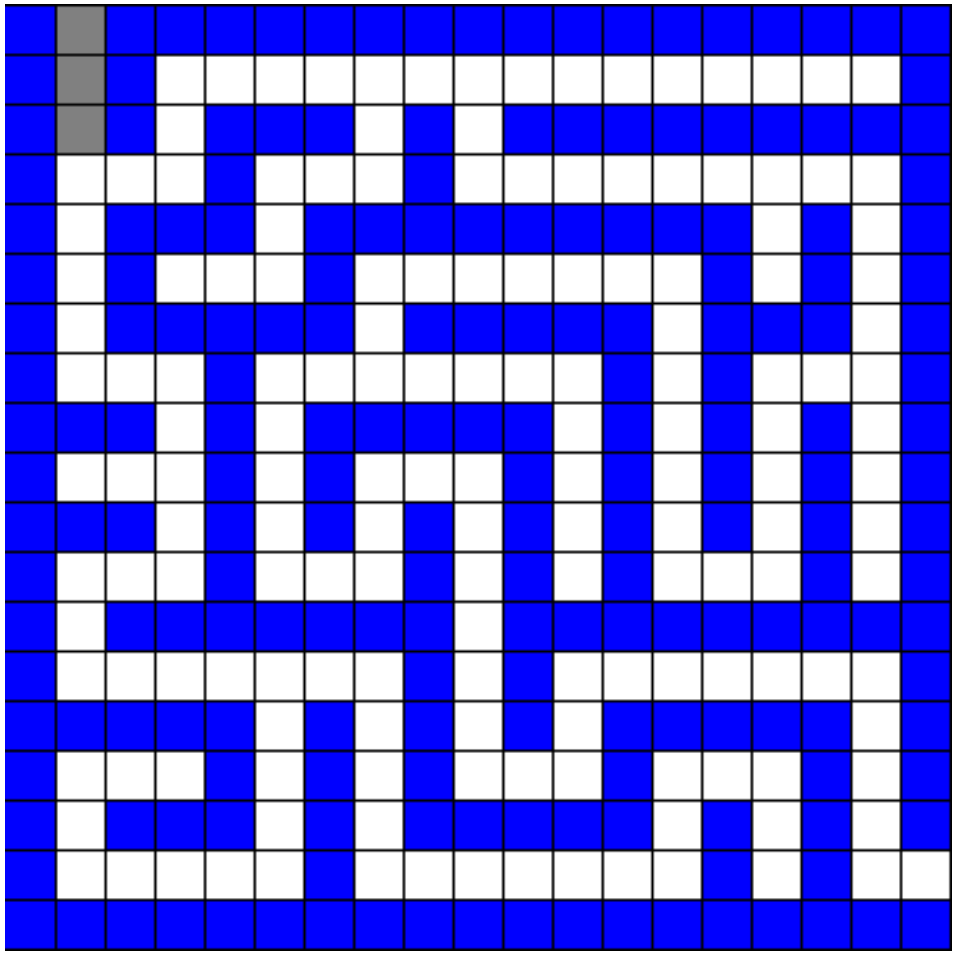
Recorrido de Grafos - DFS



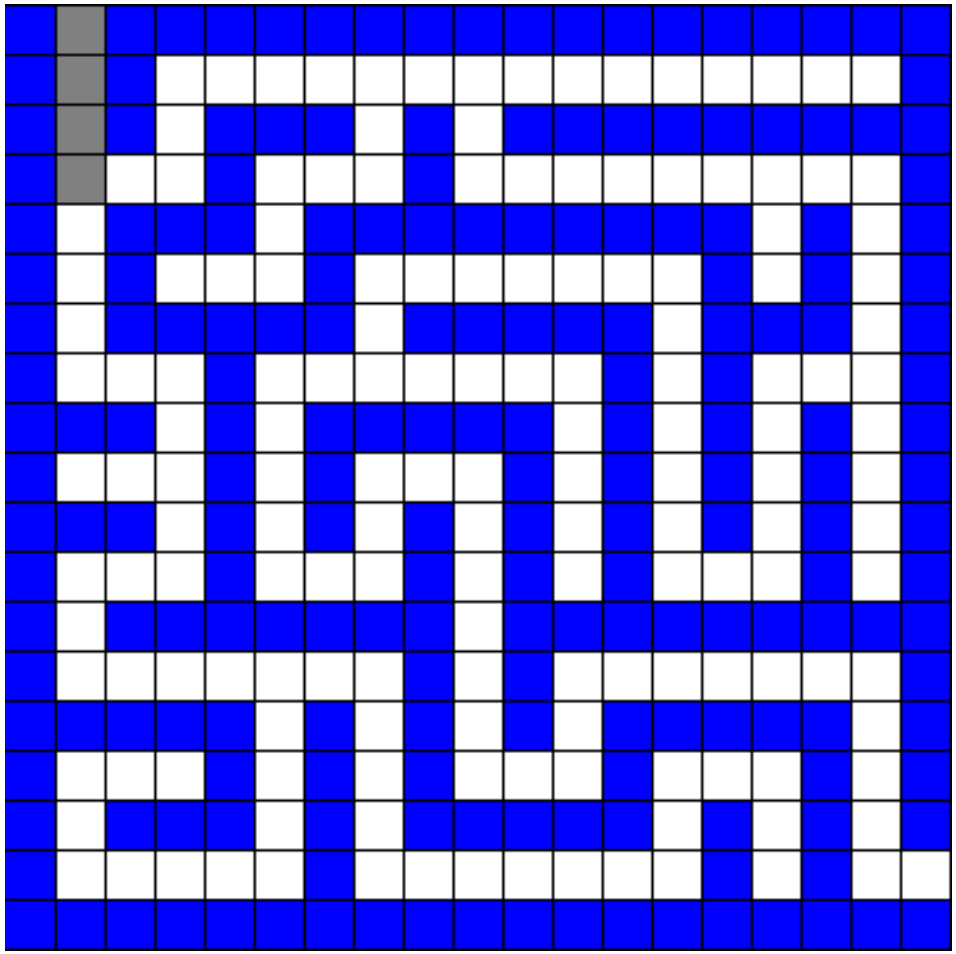
Recorrido de Grafos - DFS



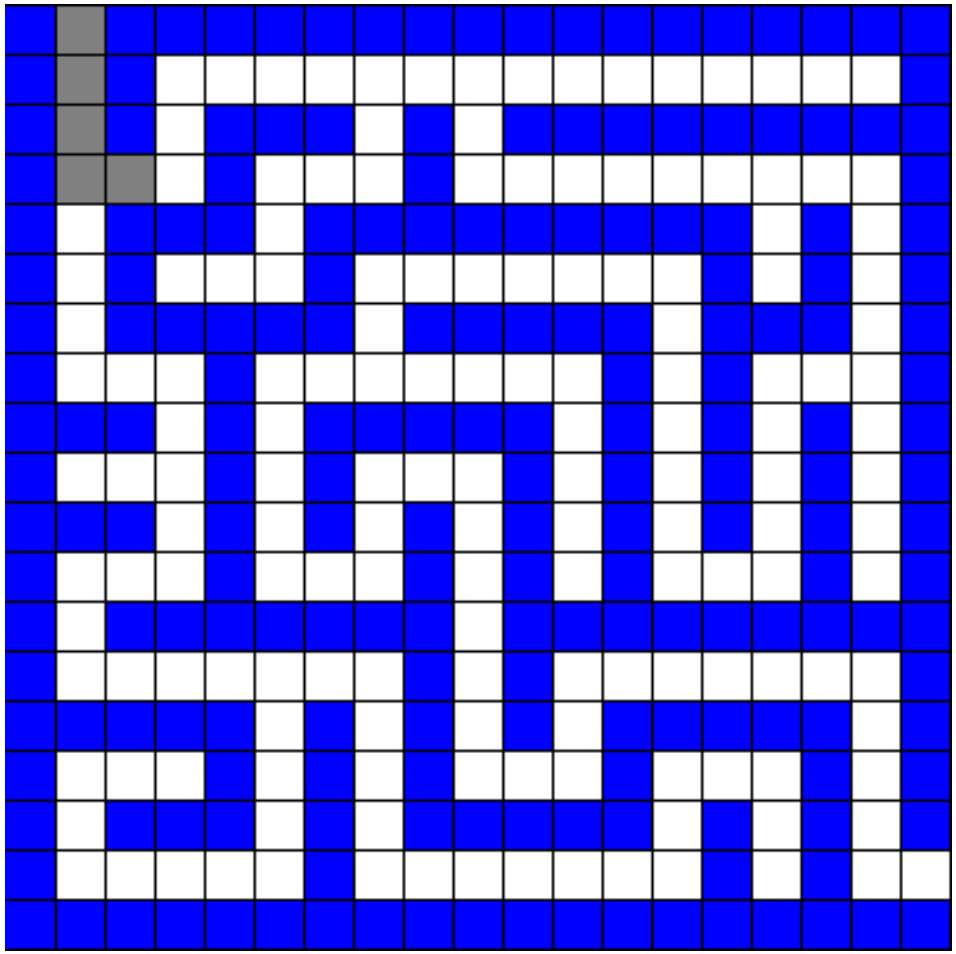
Recorrido de Grafos - DFS



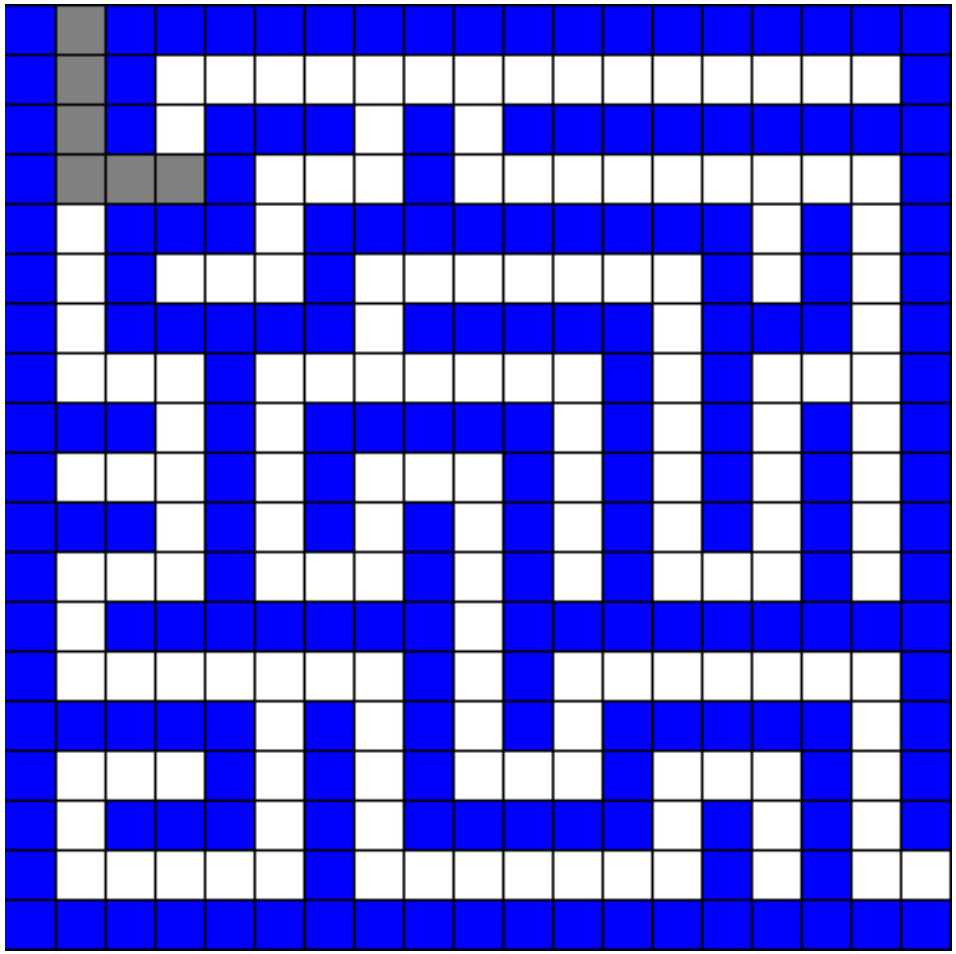
Recorrido de Grafos - DFS



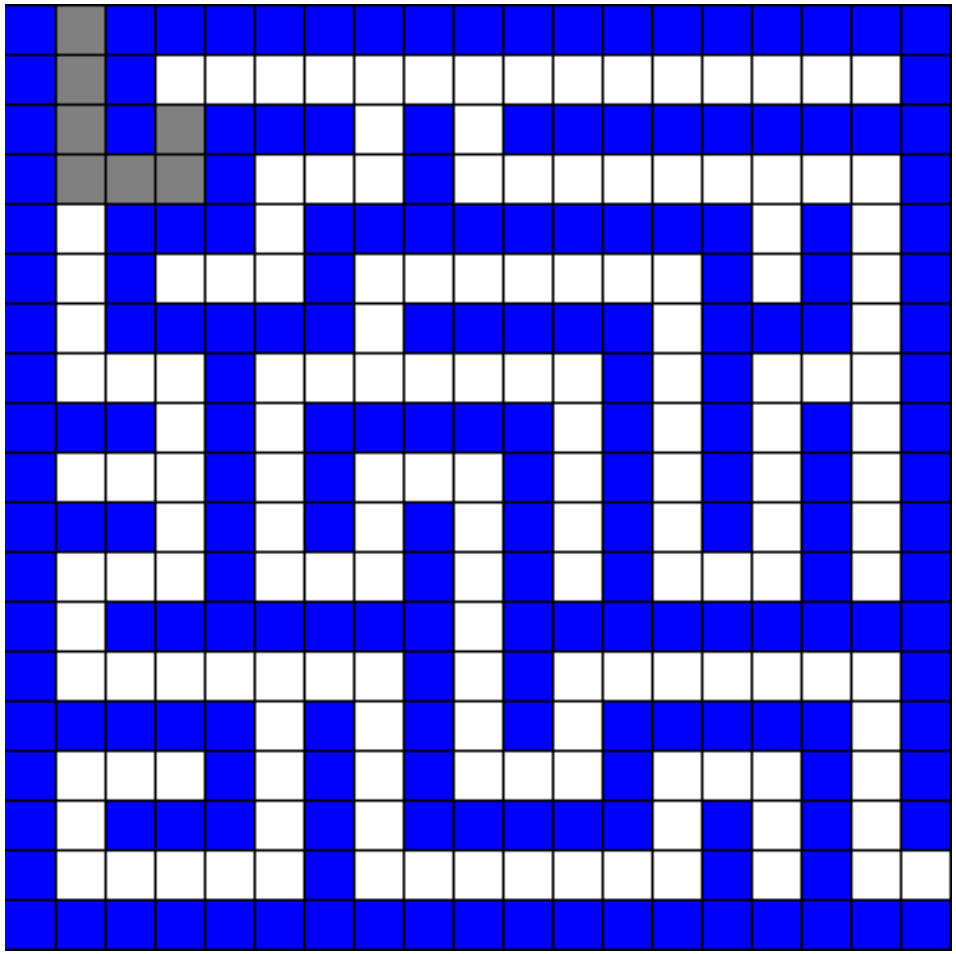
Recorrido de Grafos - DFS



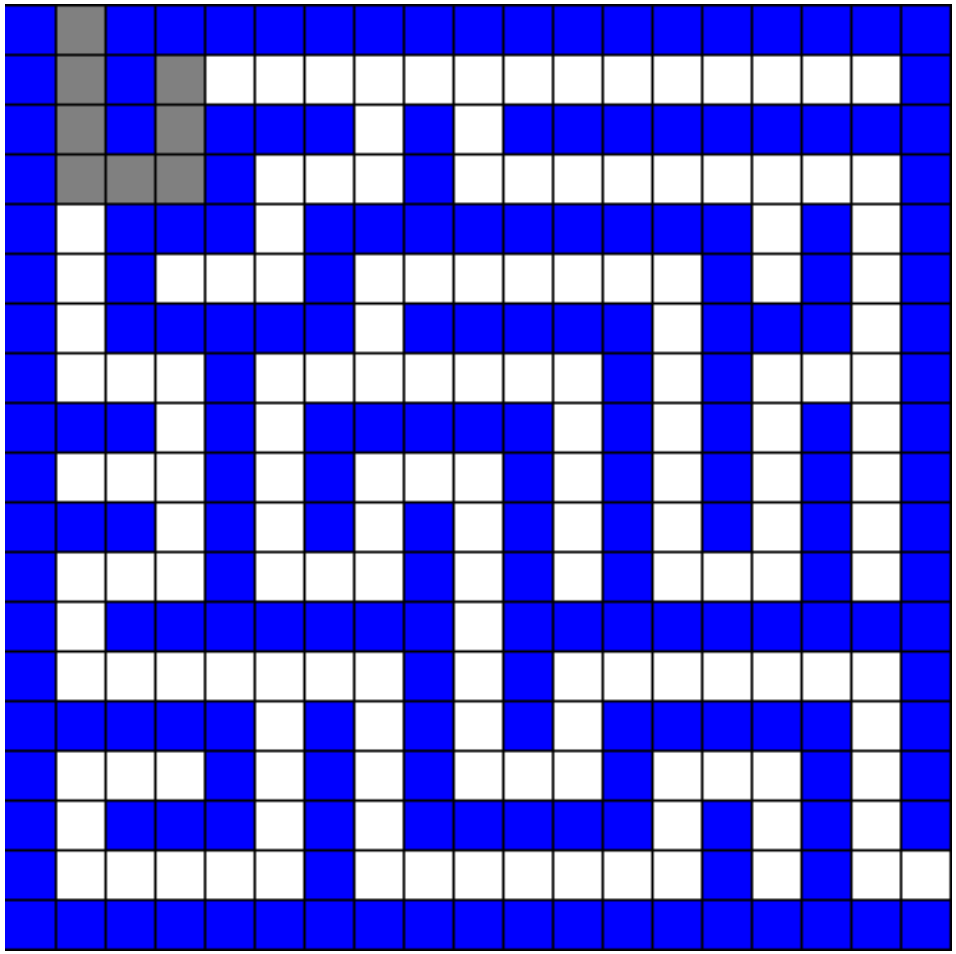
Recorrido de Grafos - DFS



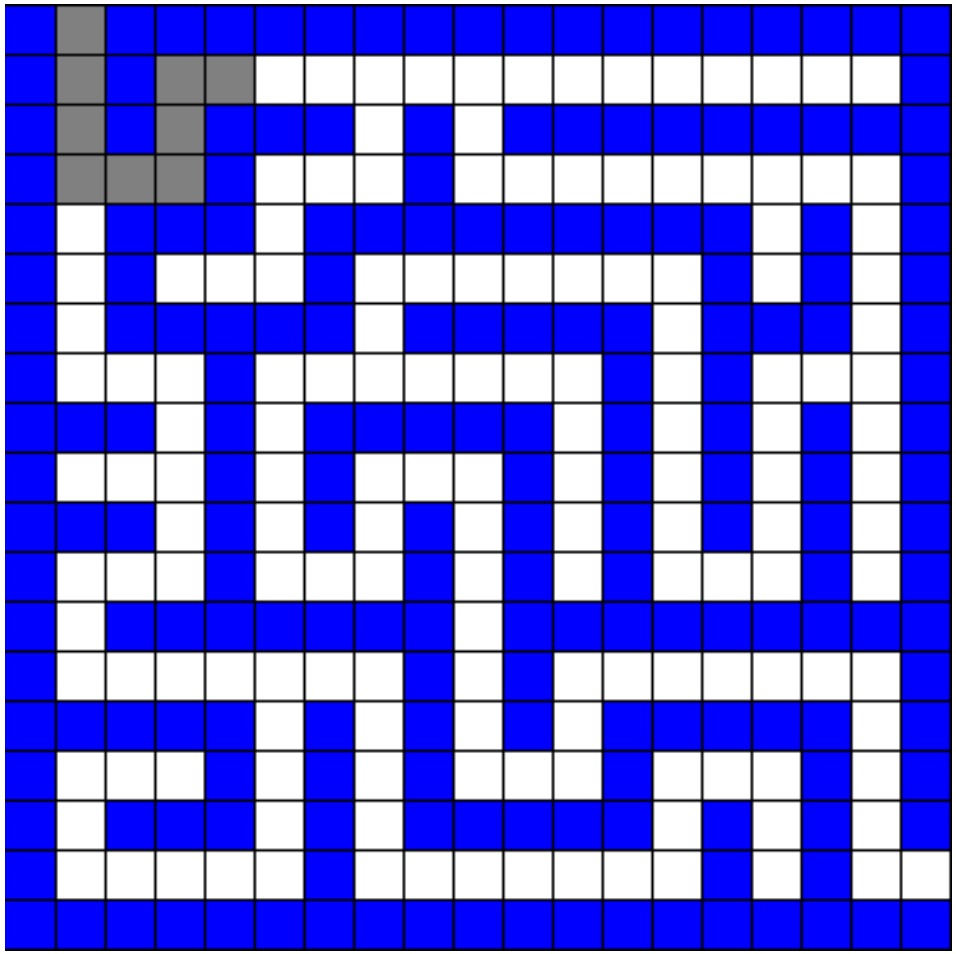
Recorrido de Grafos - DFS



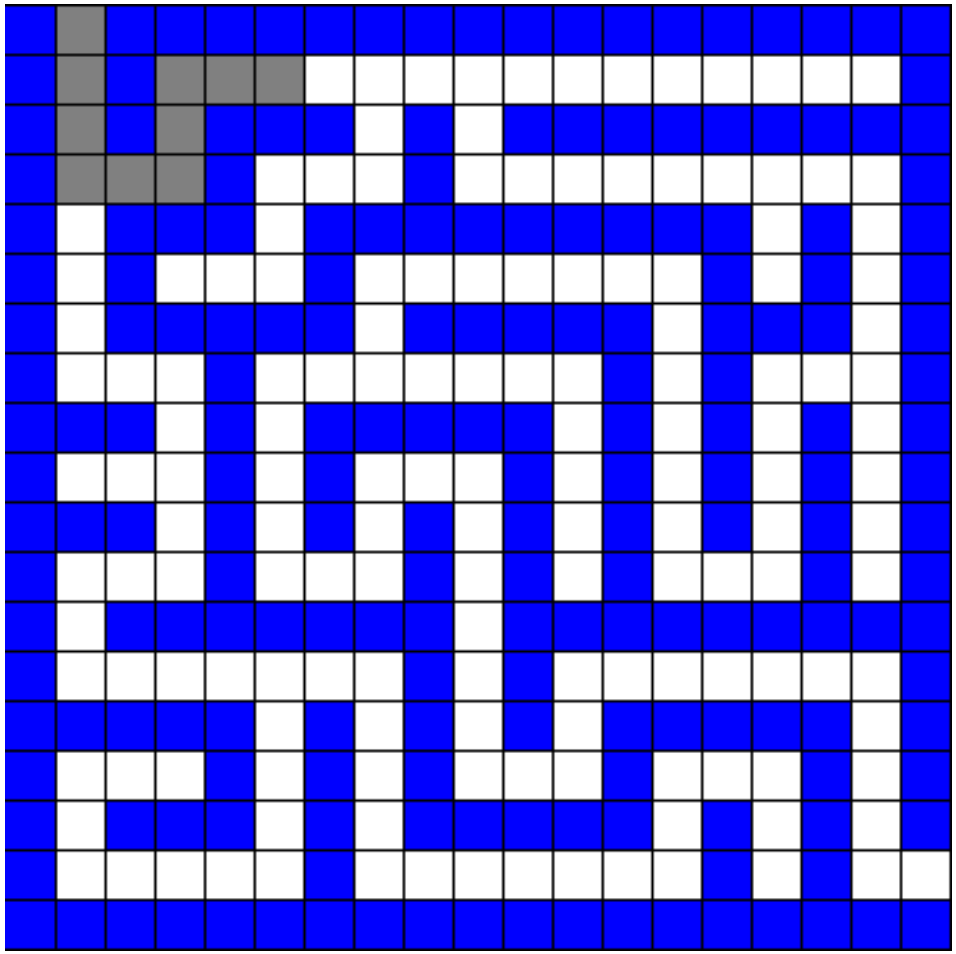
Recorrido de Grafos - DFS



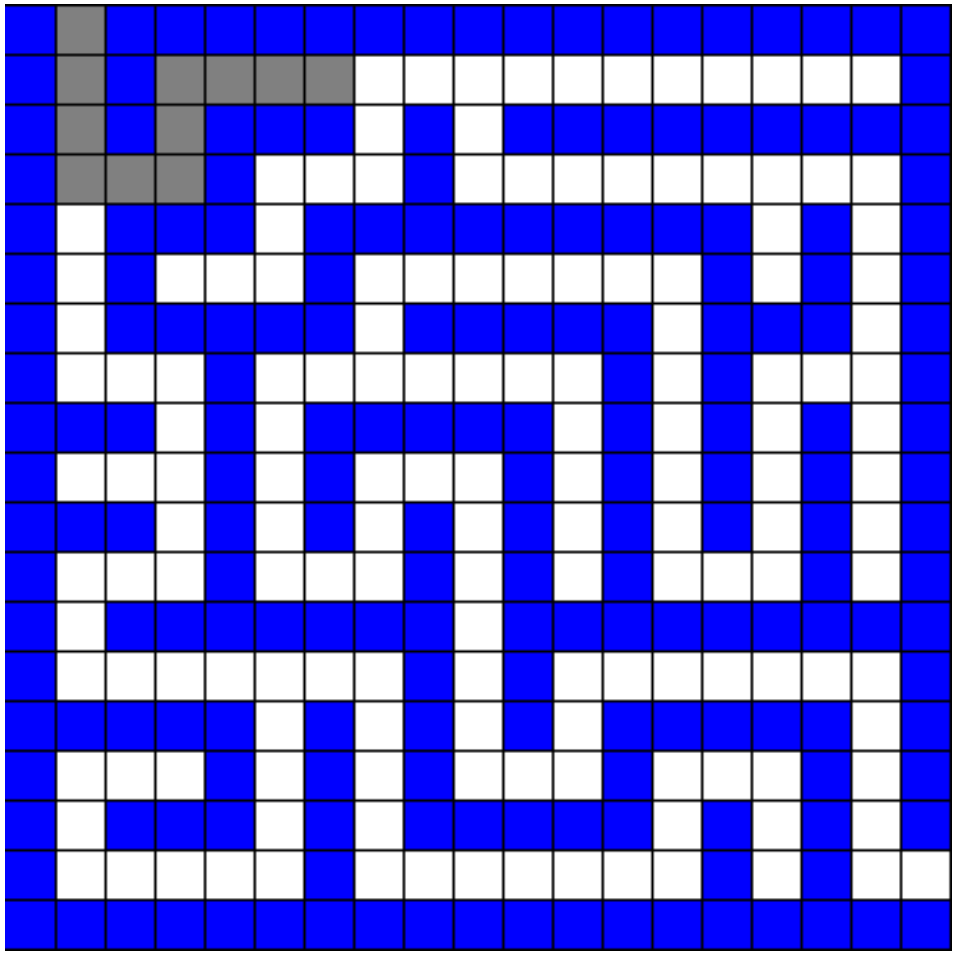
Recorrido de Grafos - DFS



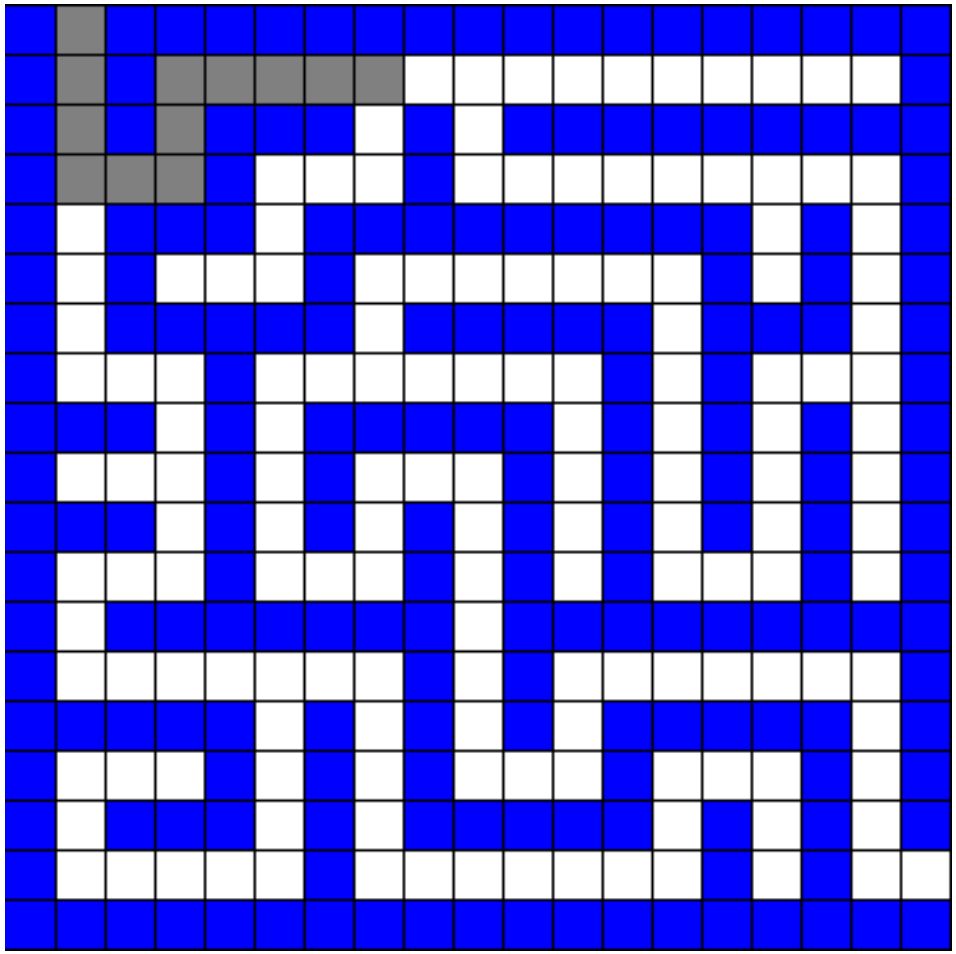
Recorrido de Grafos - DFS



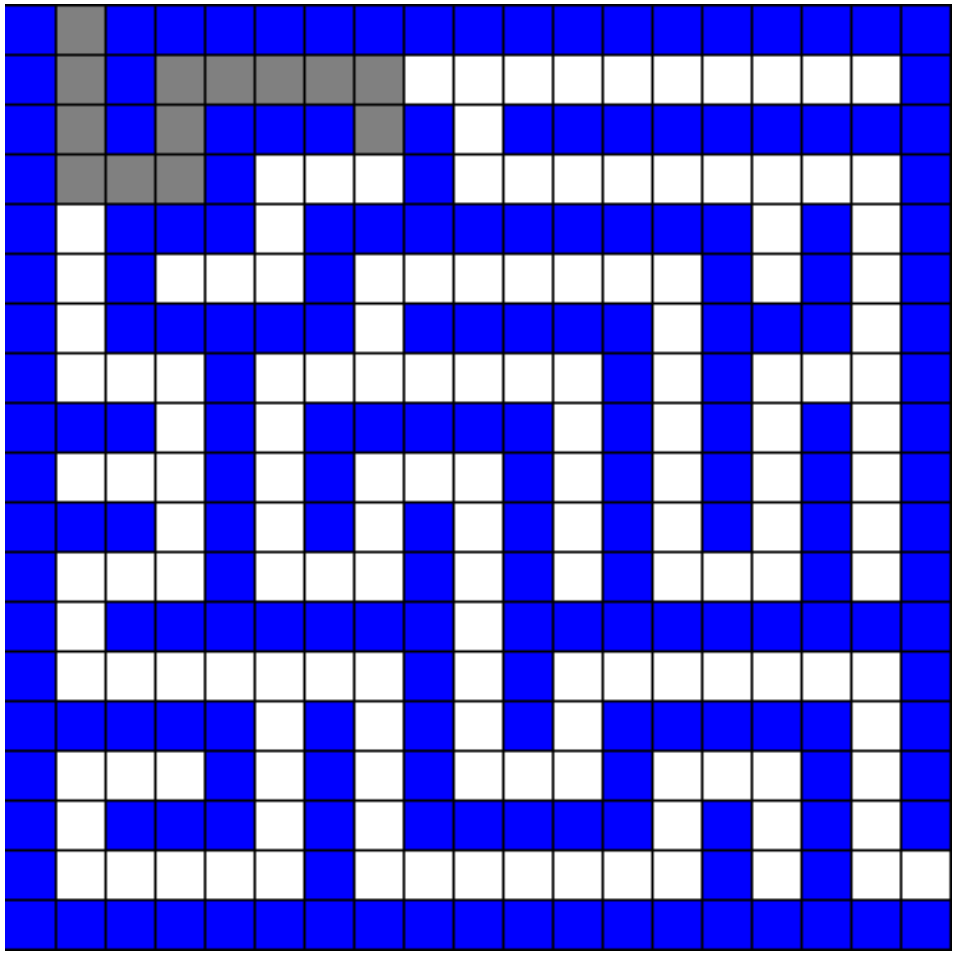
Recorrido de Grafos - DFS



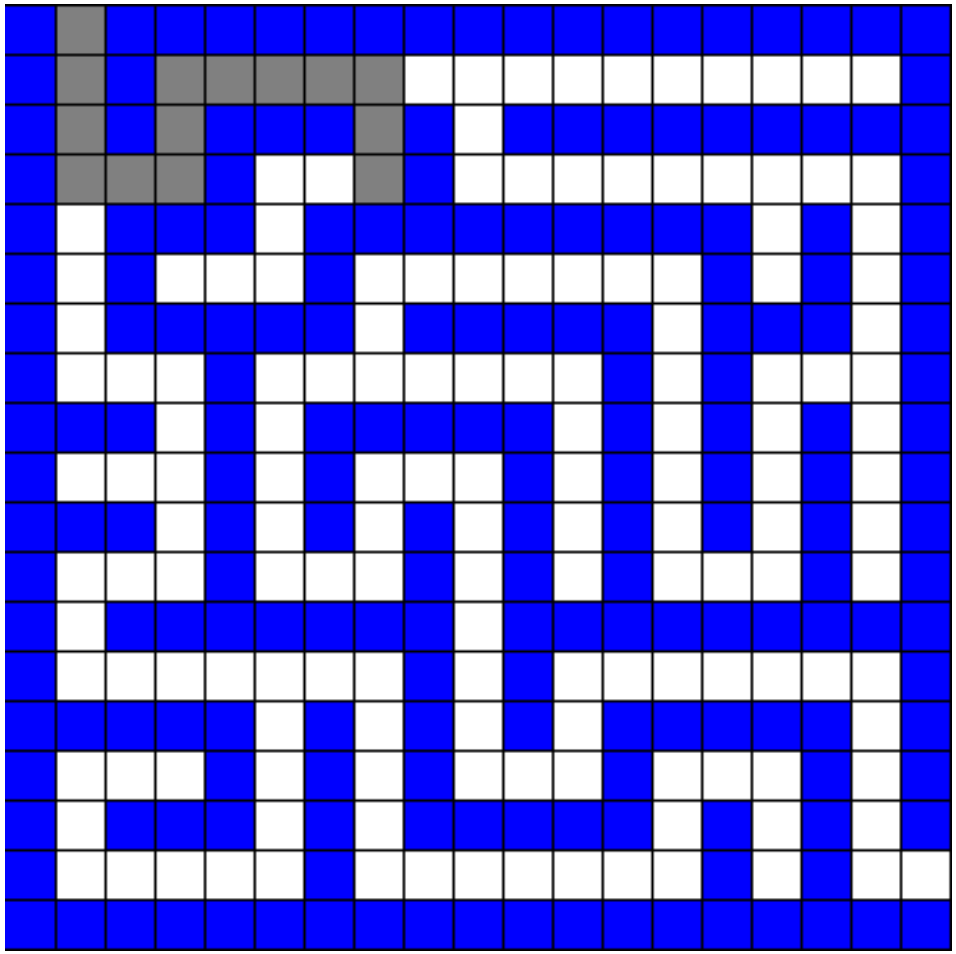
Recorrido de Grafos - DFS



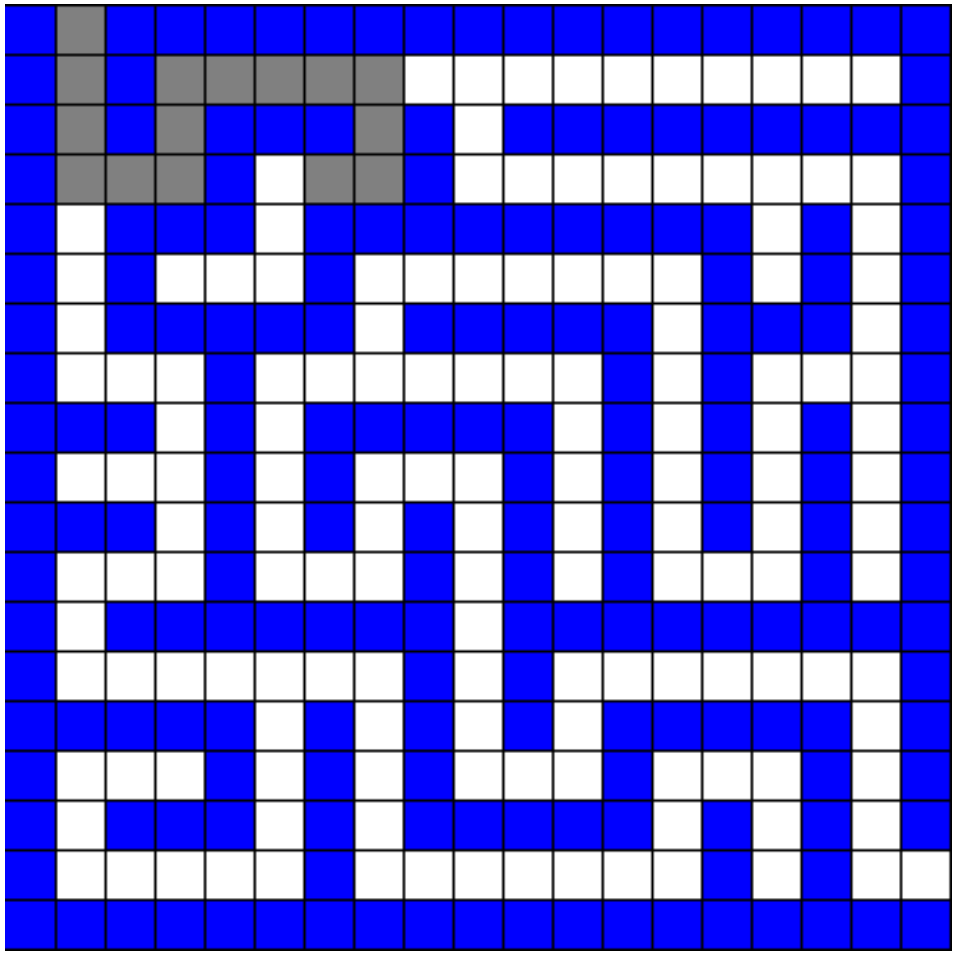
Recorrido de Grafos - DFS



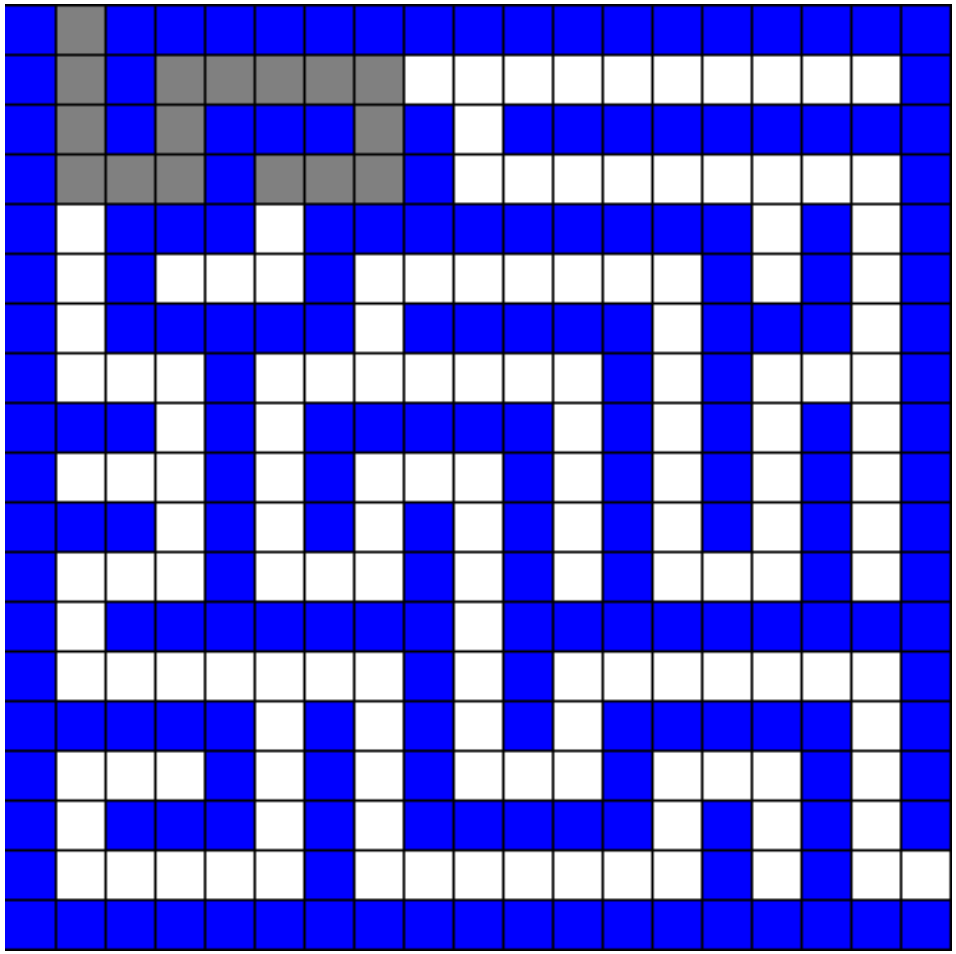
Recorrido de Grafos - DFS



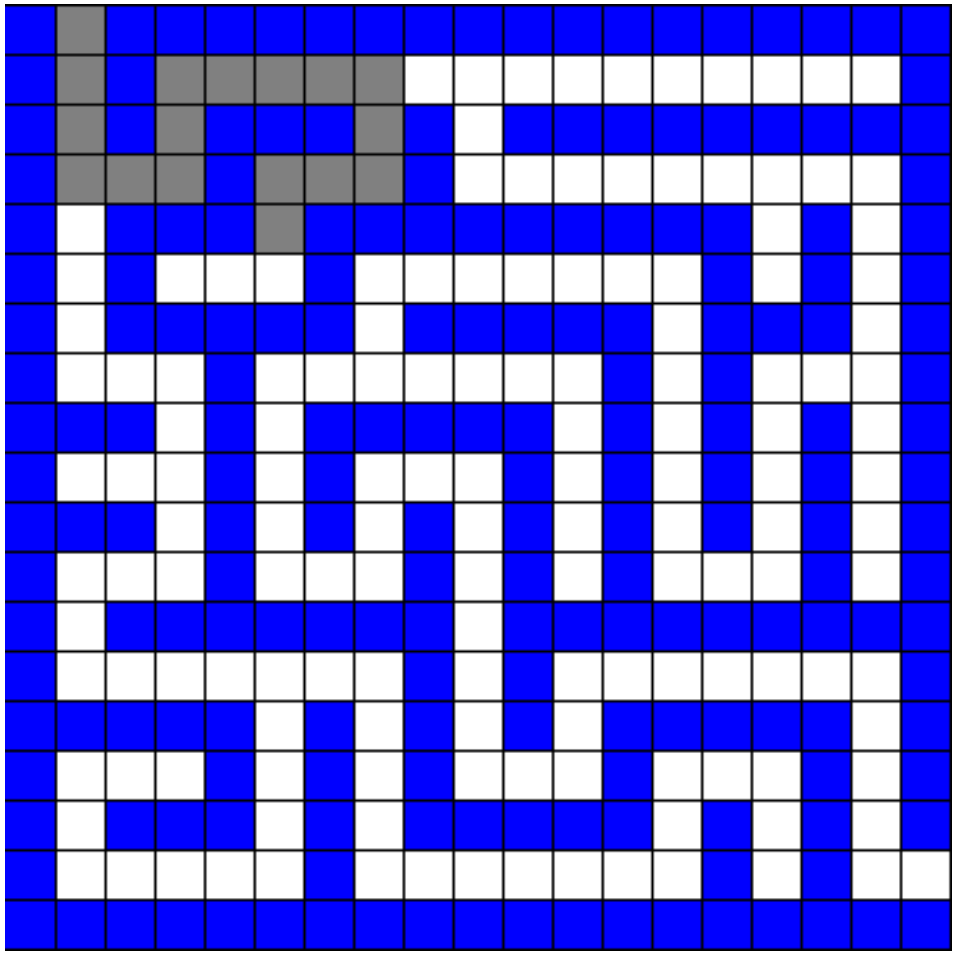
Recorrido de Grafos - DFS



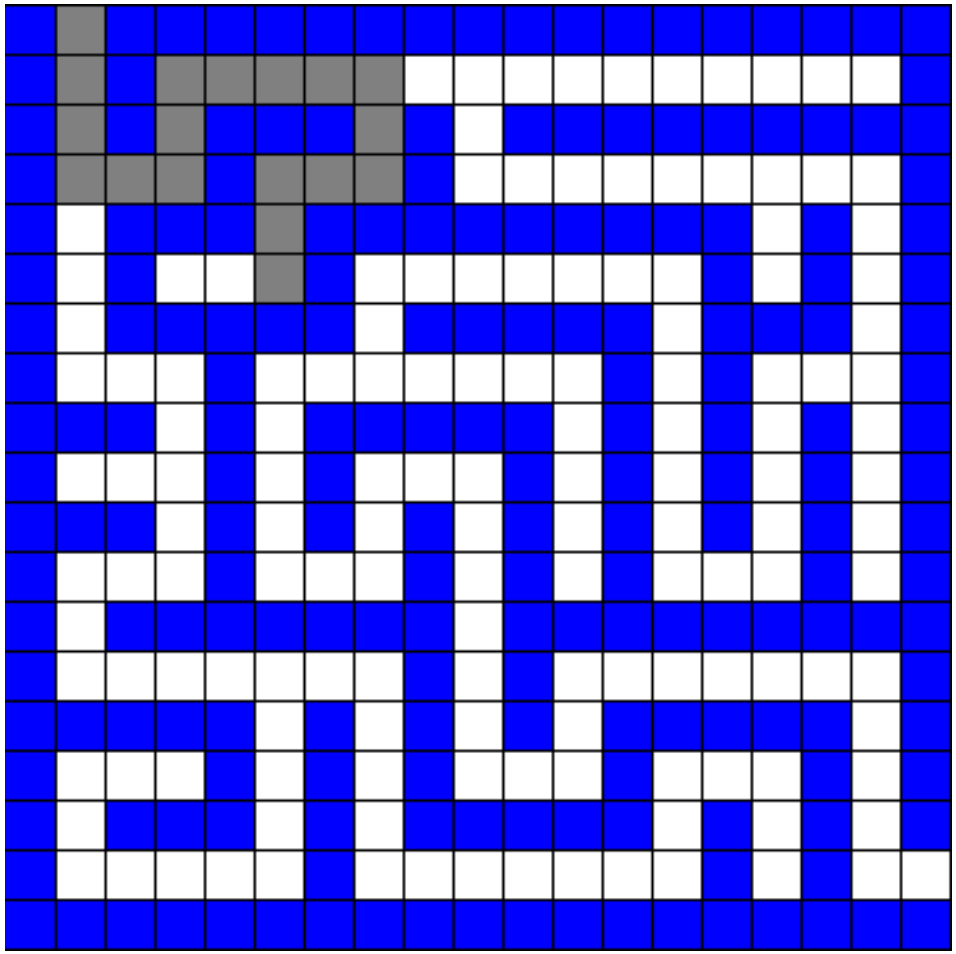
Recorrido de Grafos - DFS



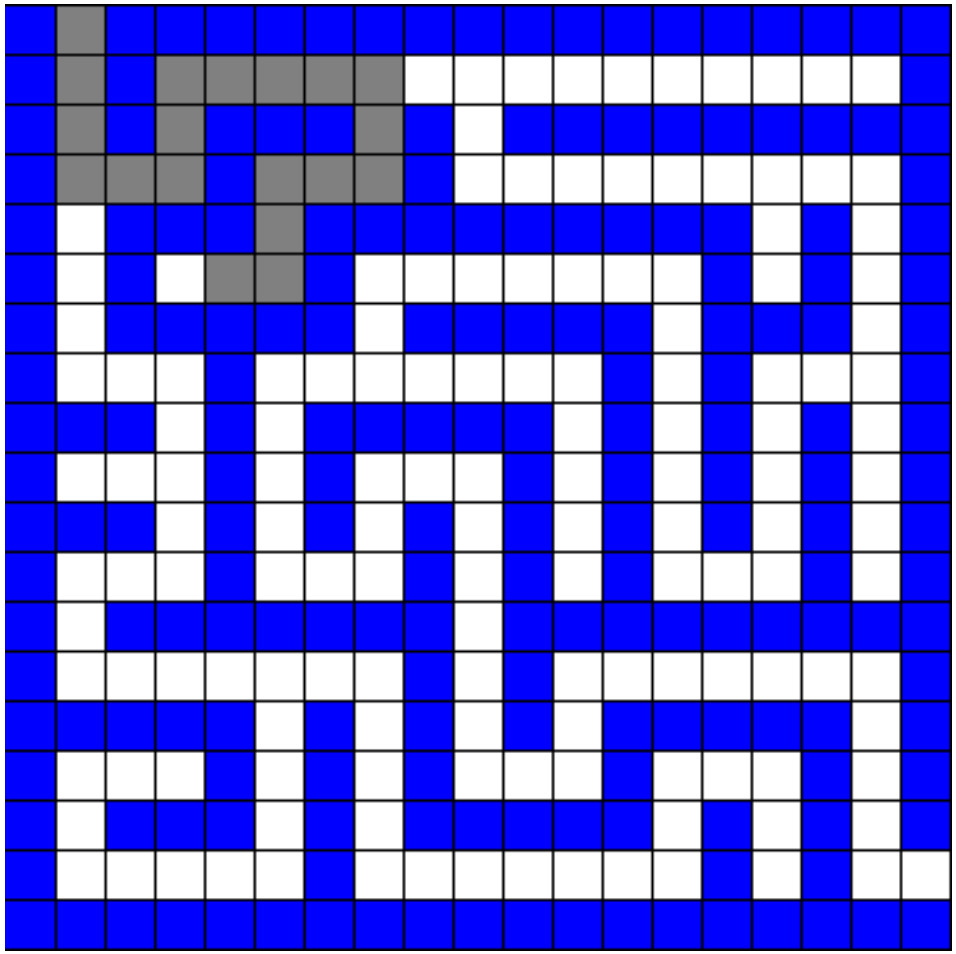
Recorrido de Grafos - DFS



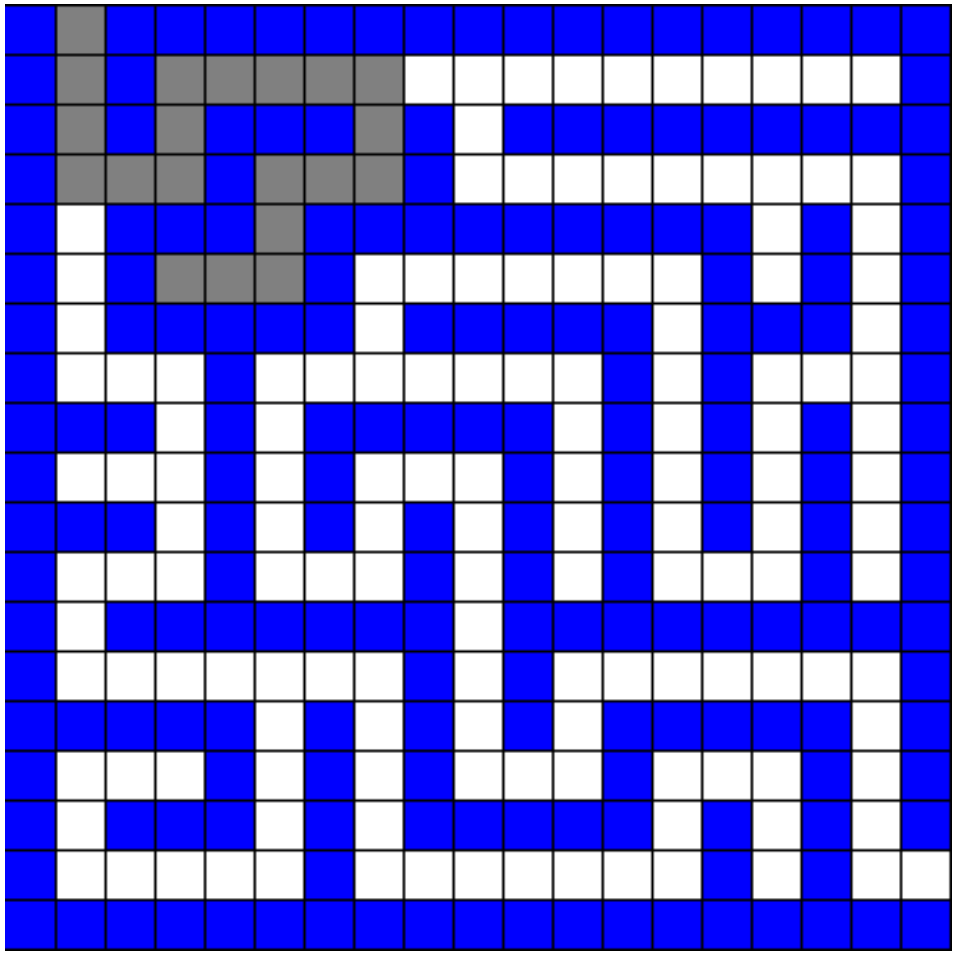
Recorrido de Grafos - DFS



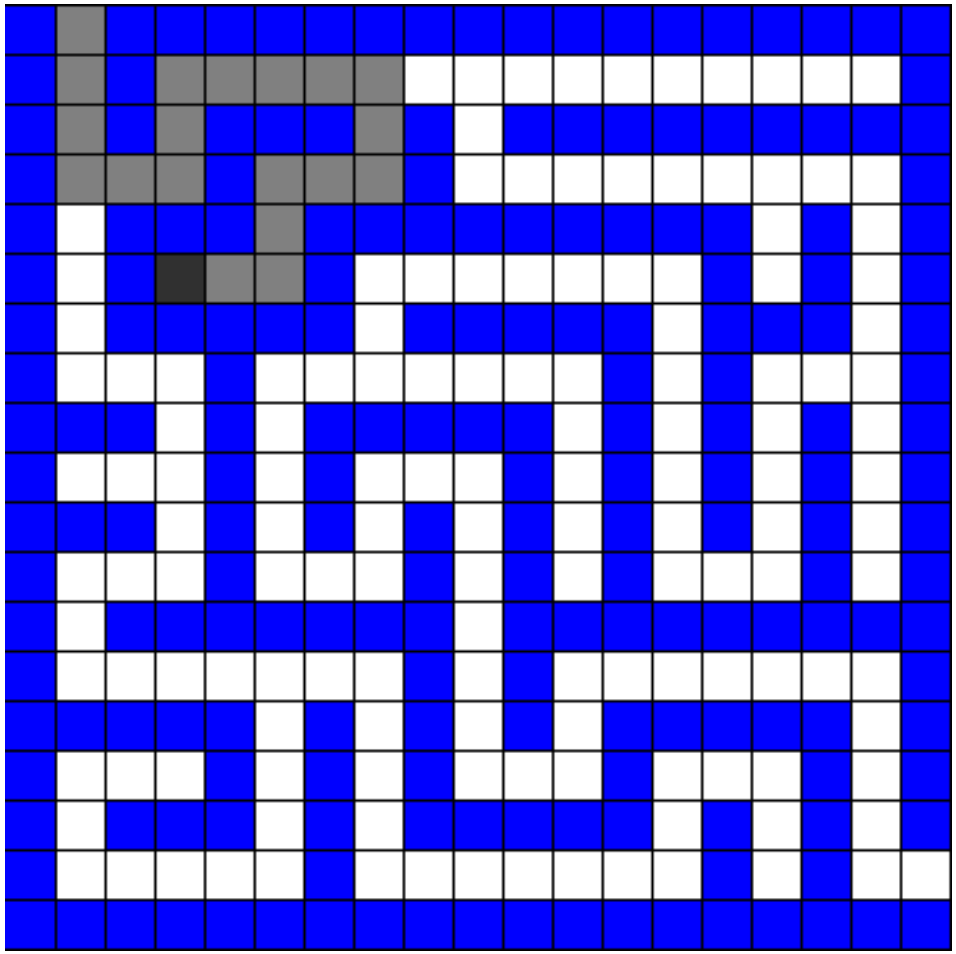
Recorrido de Grafos - DFS



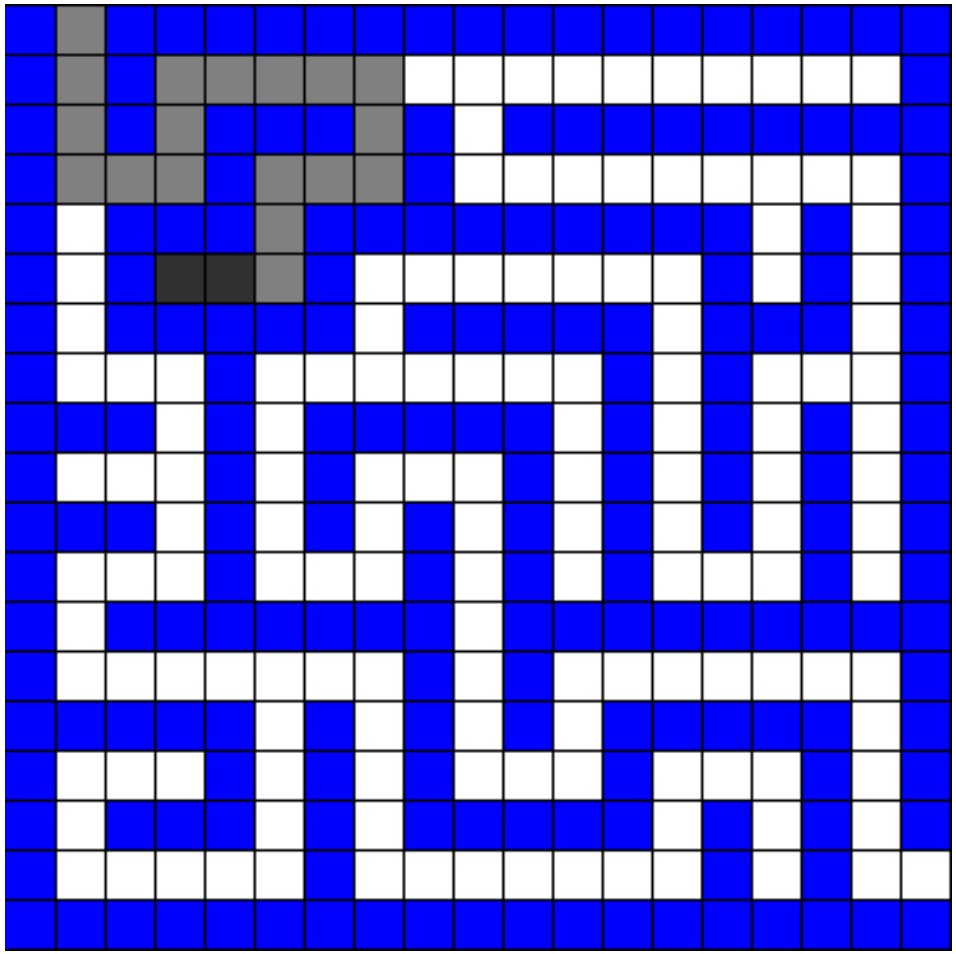
Recorrido de Grafos - DFS



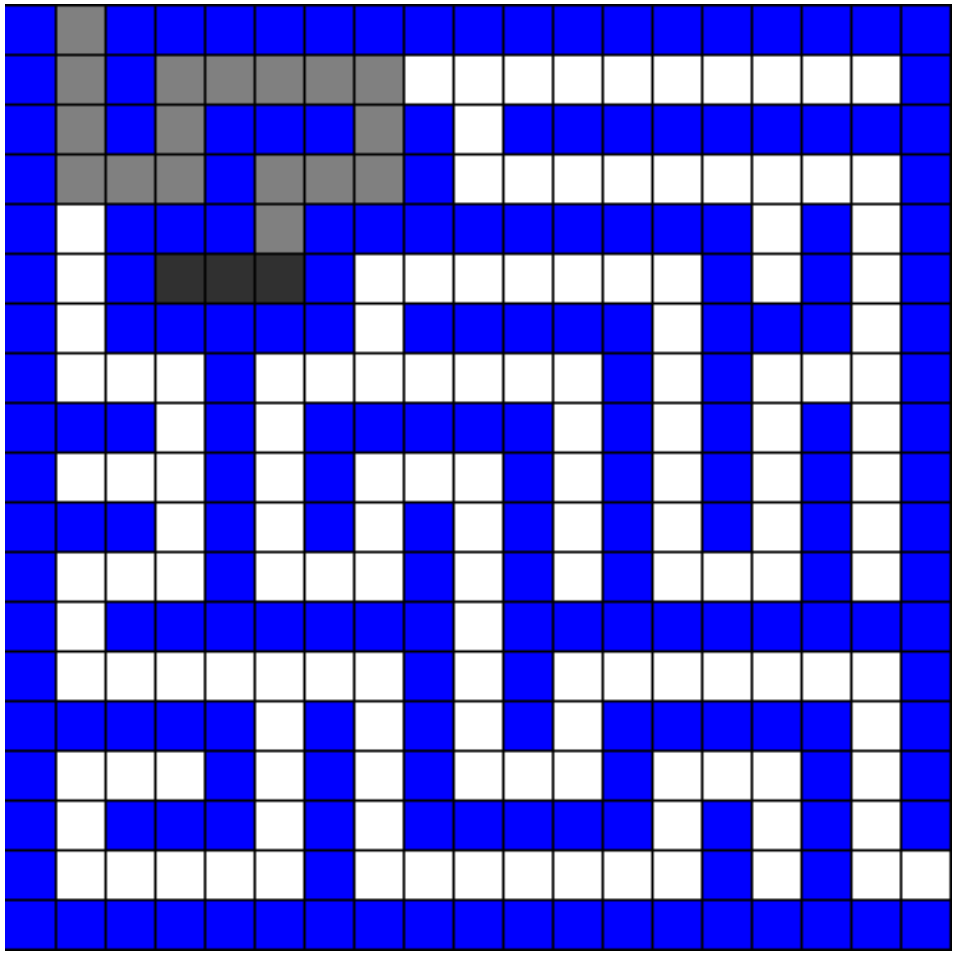
Recorrido de Grafos - DFS



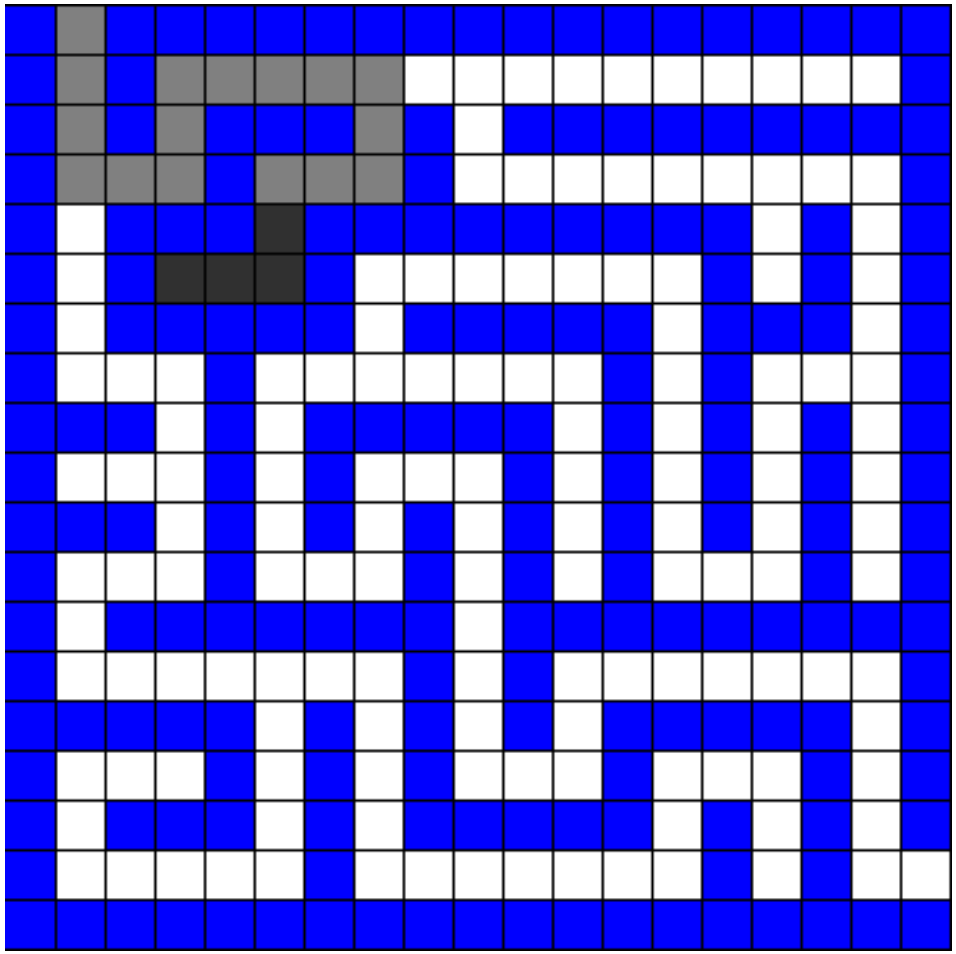
Recorrido de Grafos - DFS



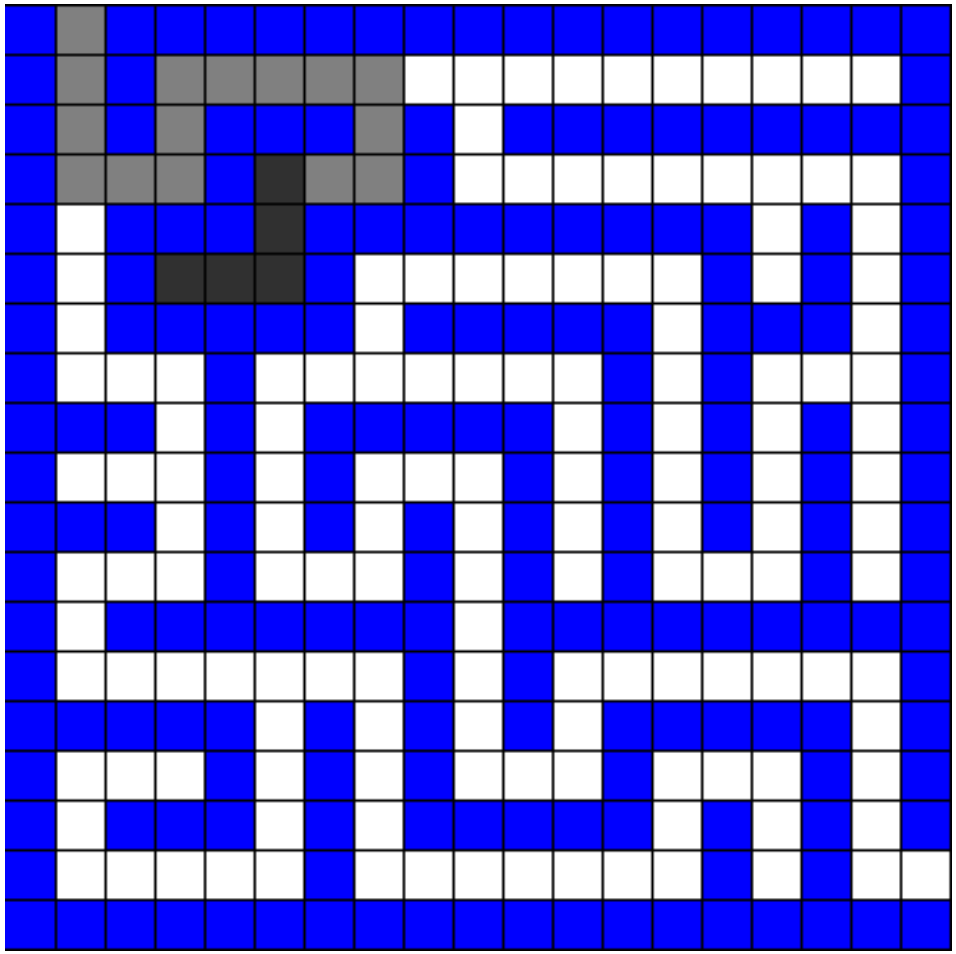
Recorrido de Grafos - DFS



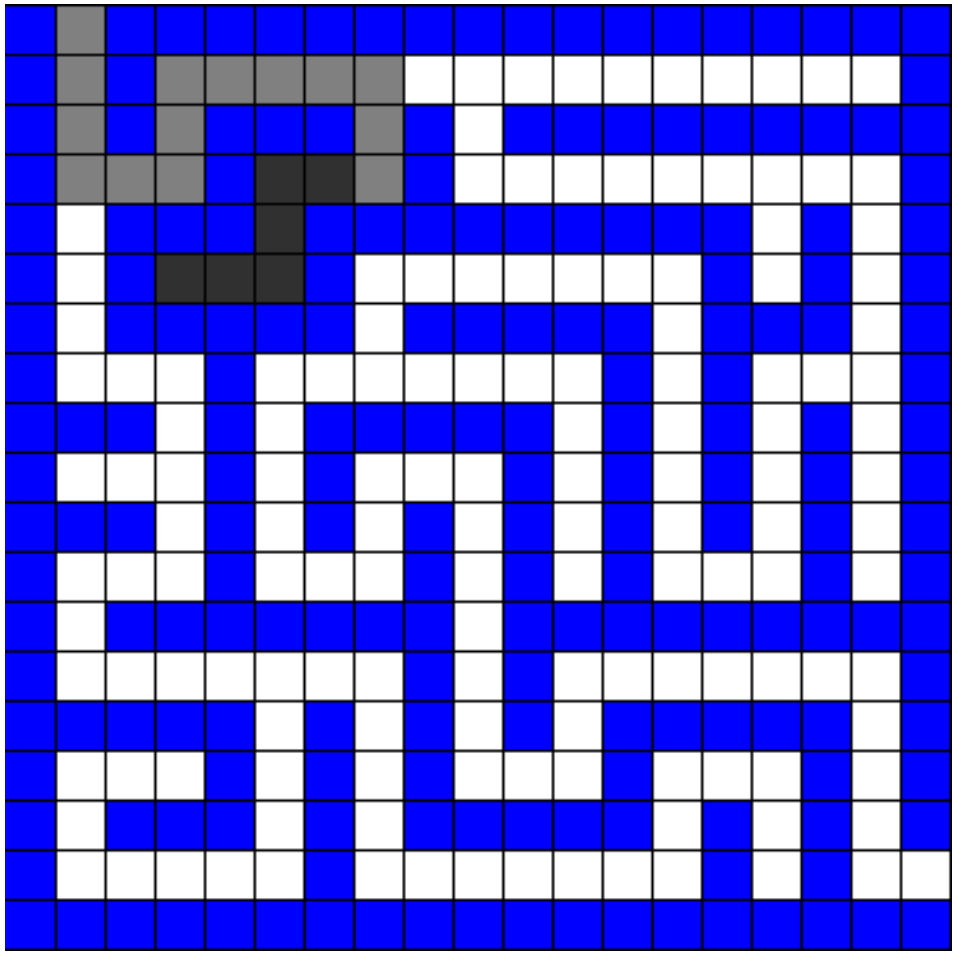
Recorrido de Grafos - DFS



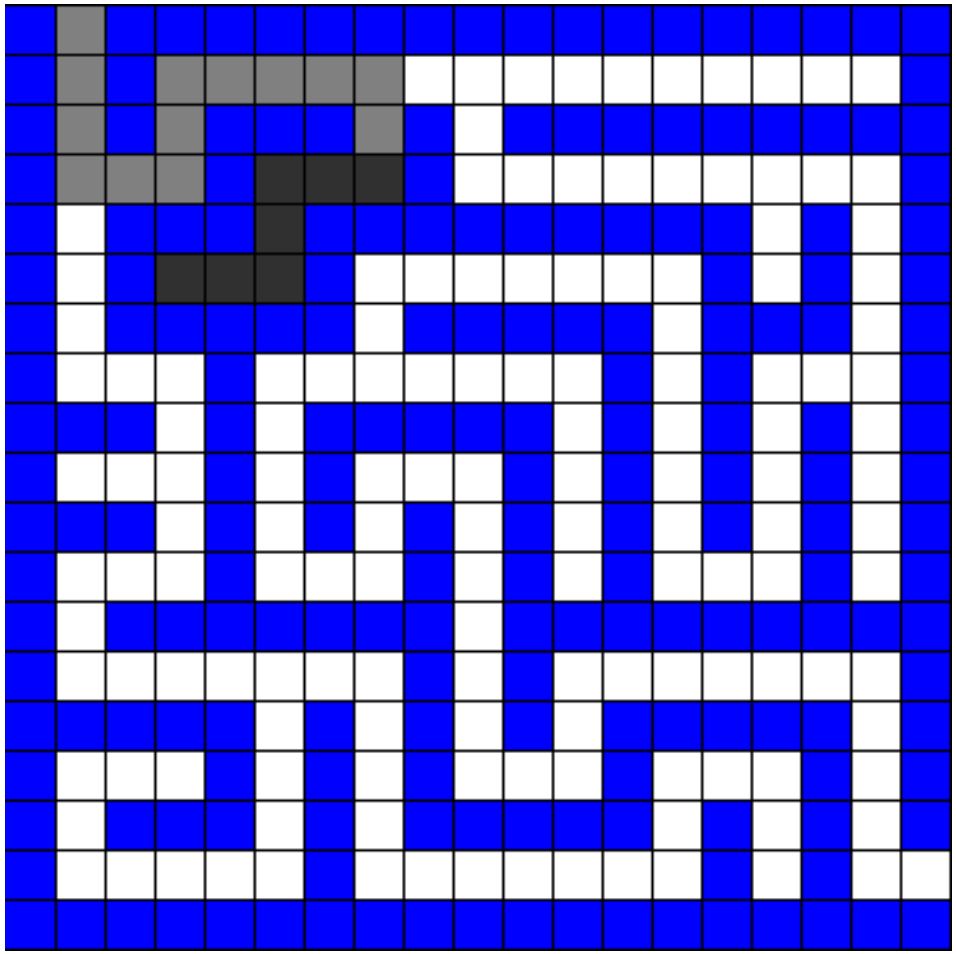
Recorrido de Grafos - DFS



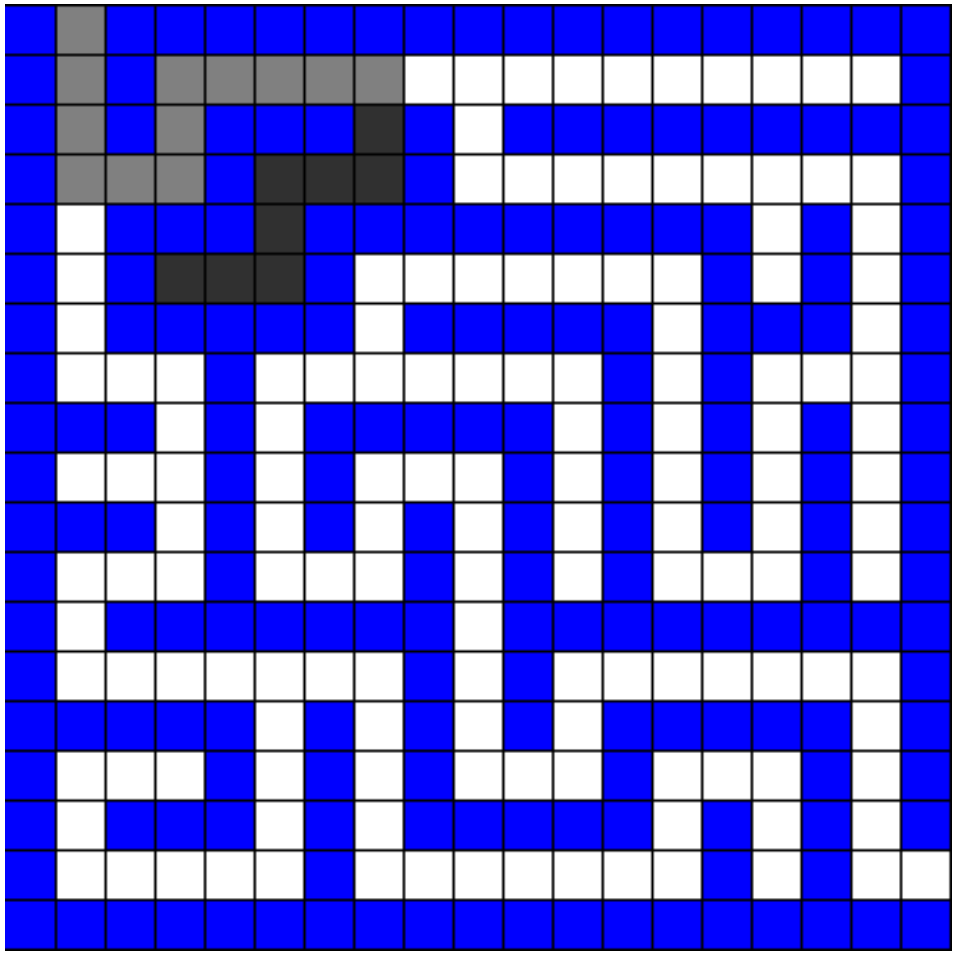
Recorrido de Grafos - DFS



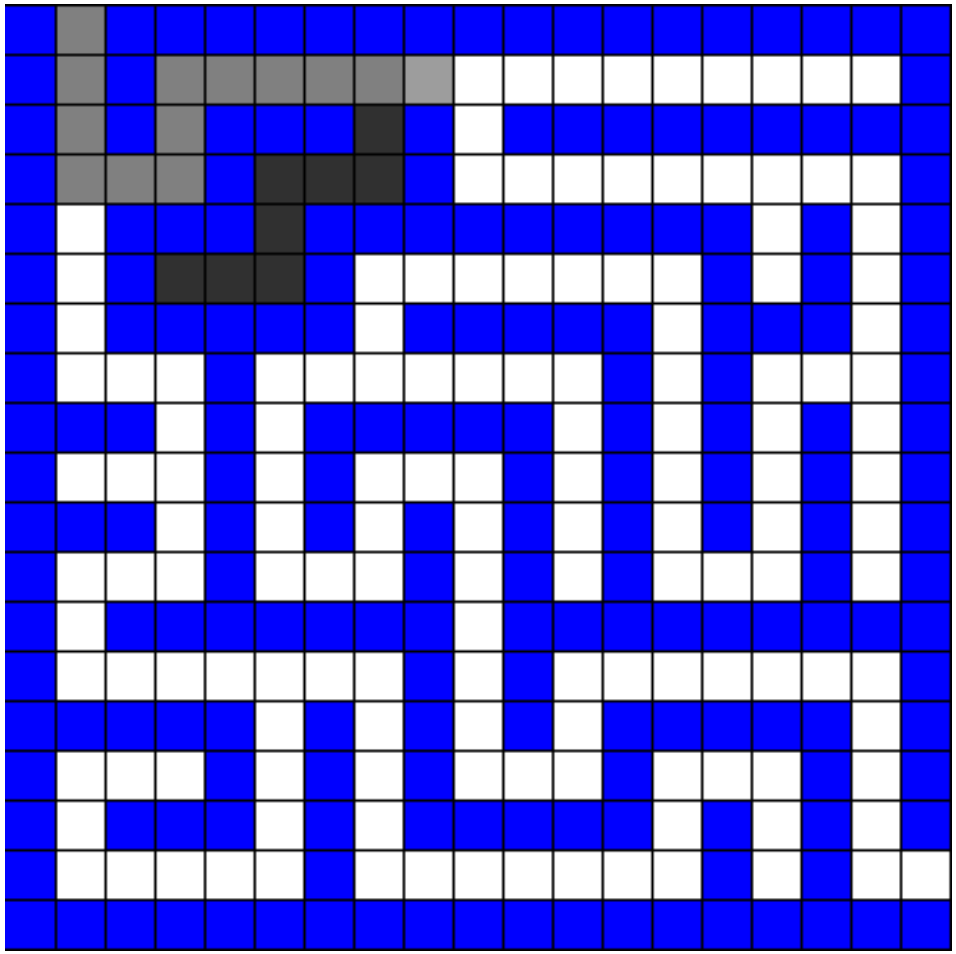
Recorrido de Grafos - DFS



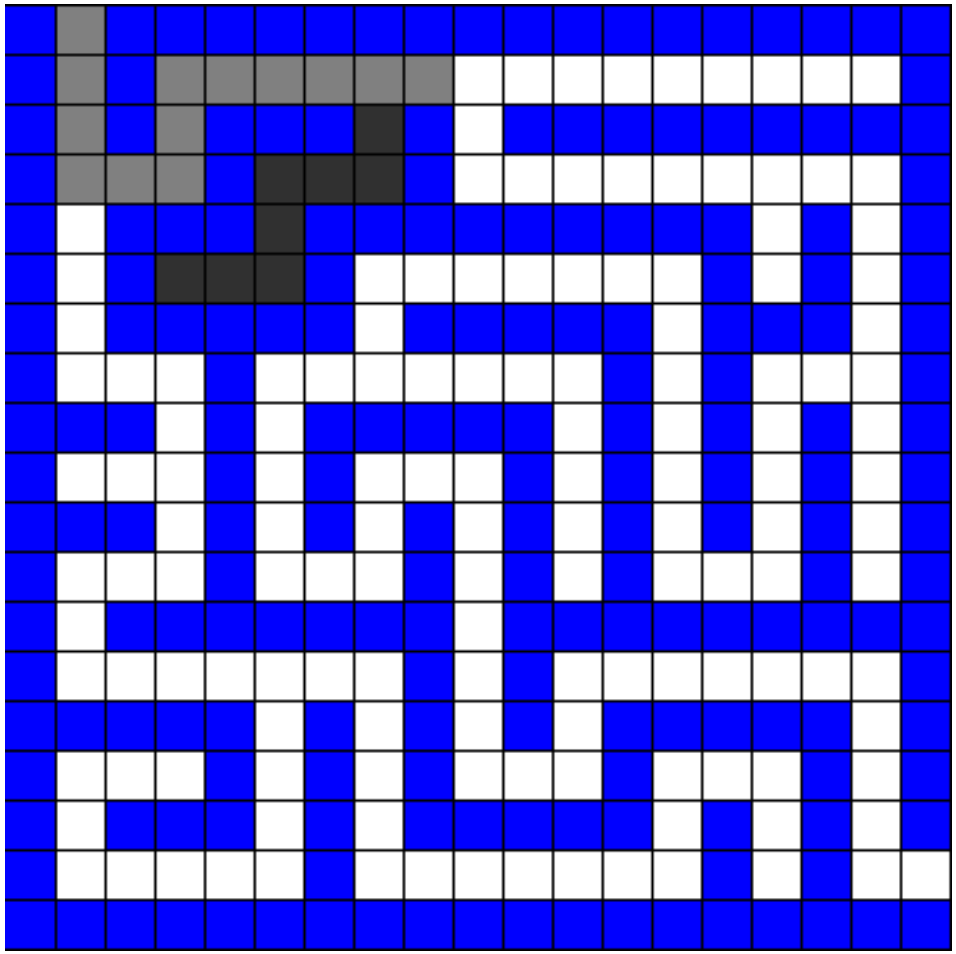
Recorrido de Grafos - DFS



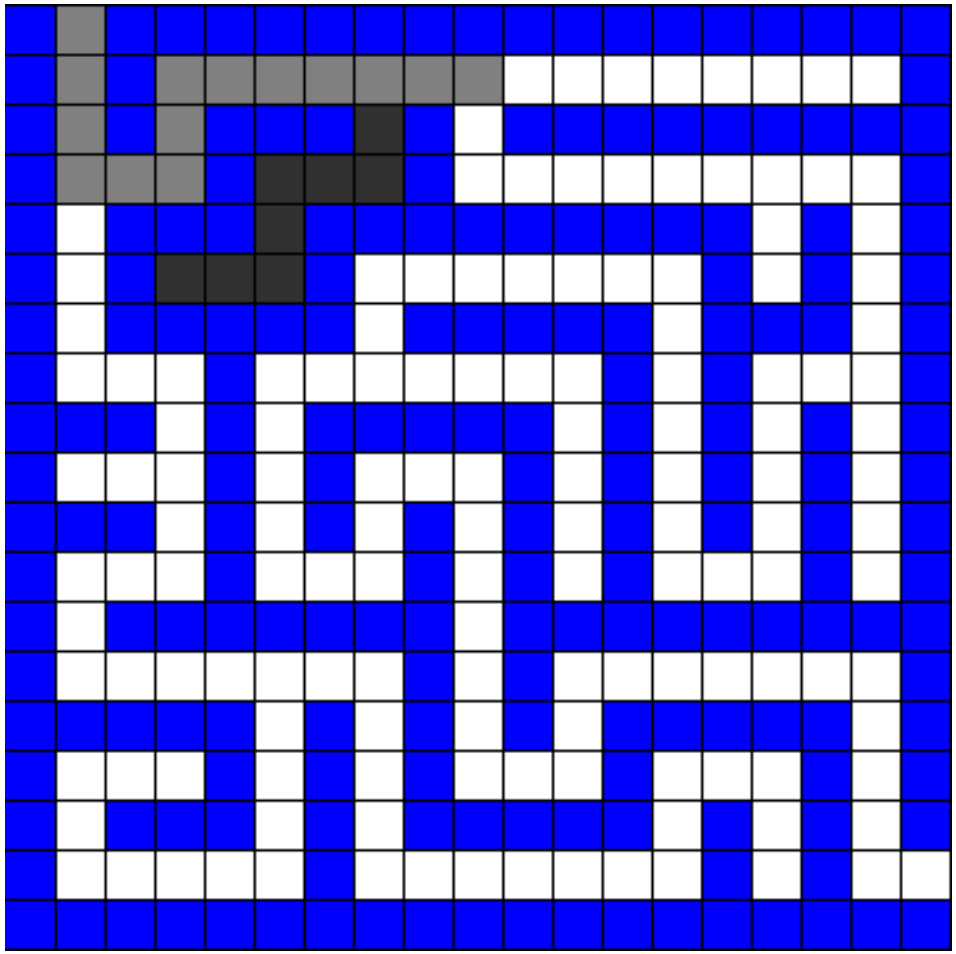
Recorrido de Grafos - DFS



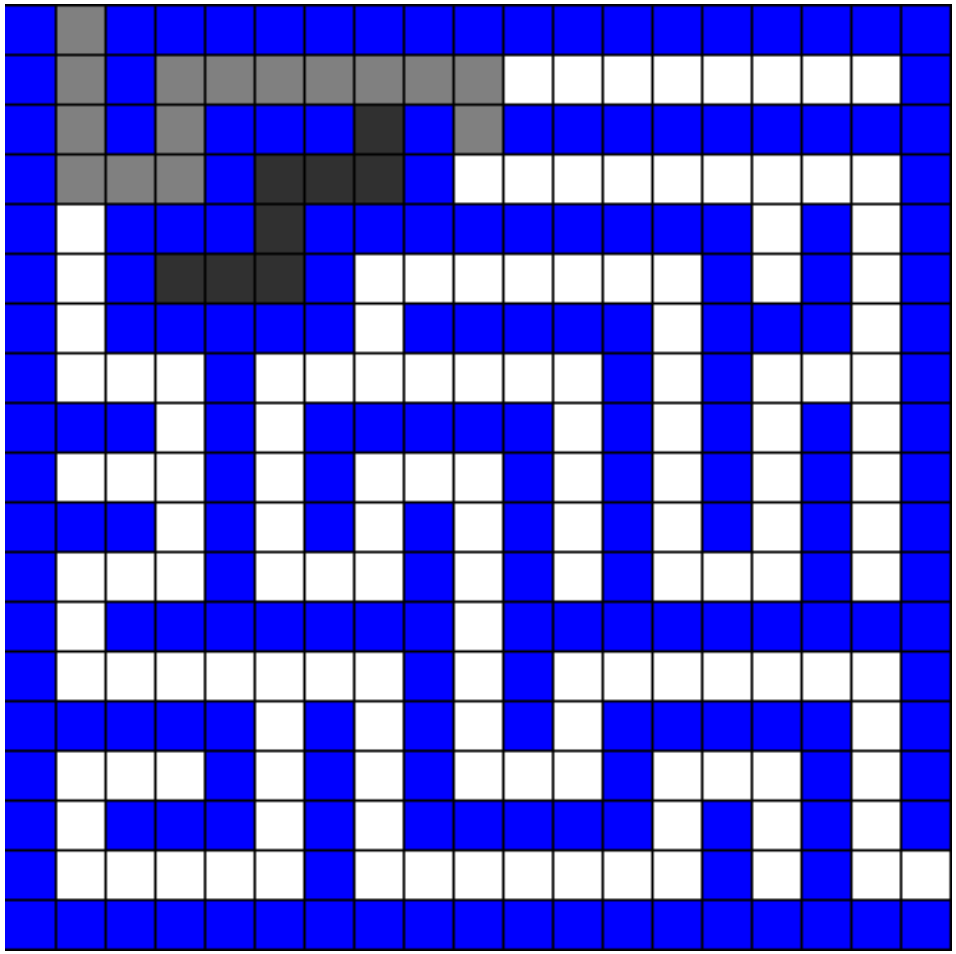
Recorrido de Grafos - DFS



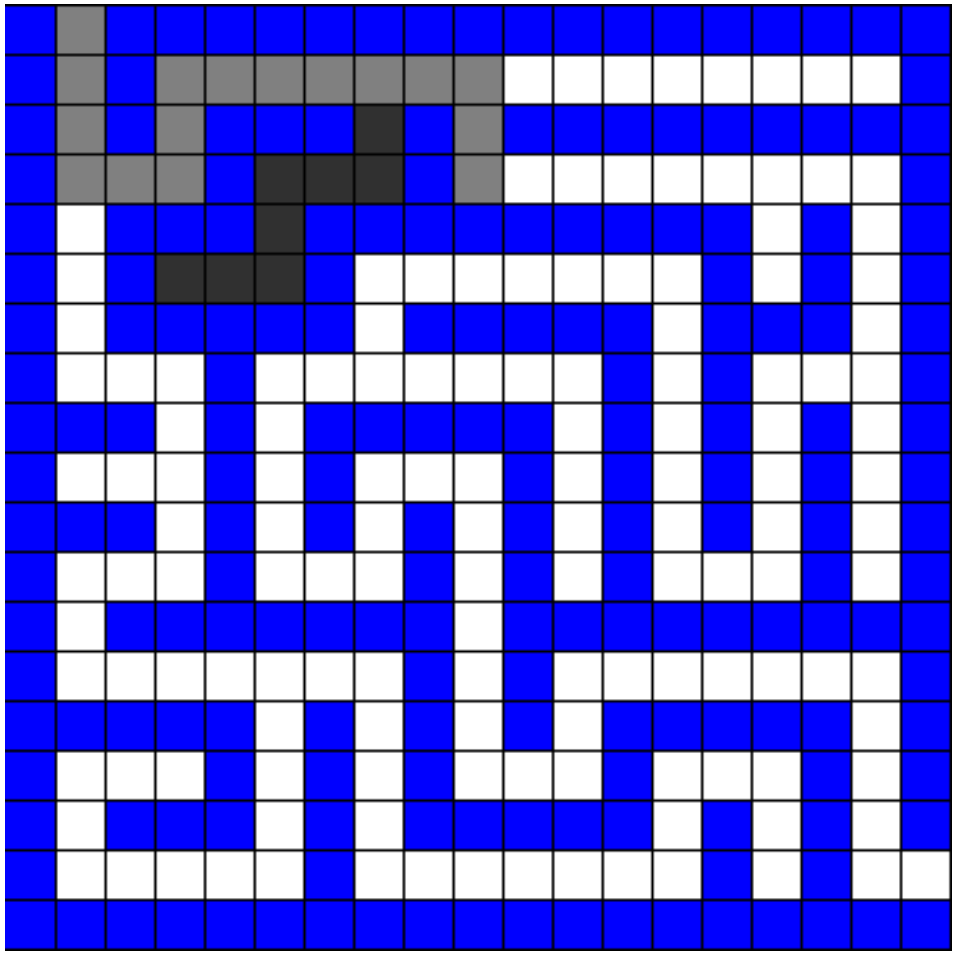
Recorrido de Grafos - DFS



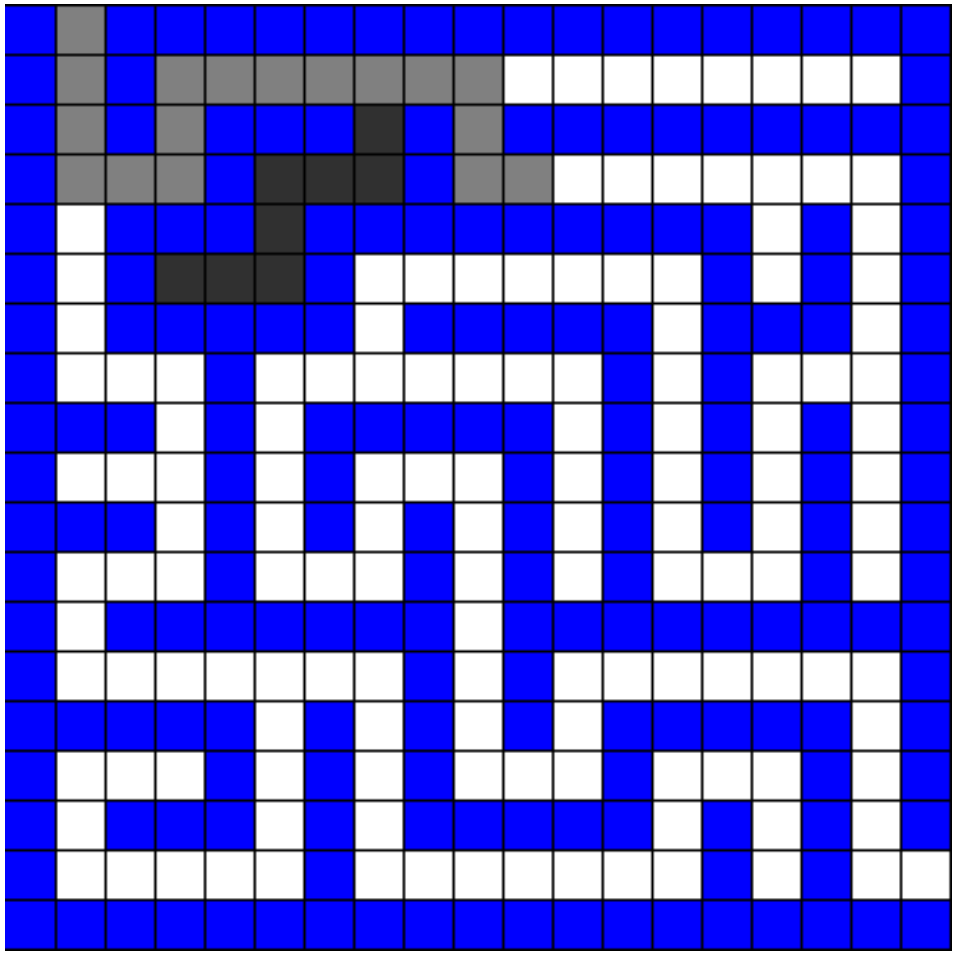
Recorrido de Grafos - DFS



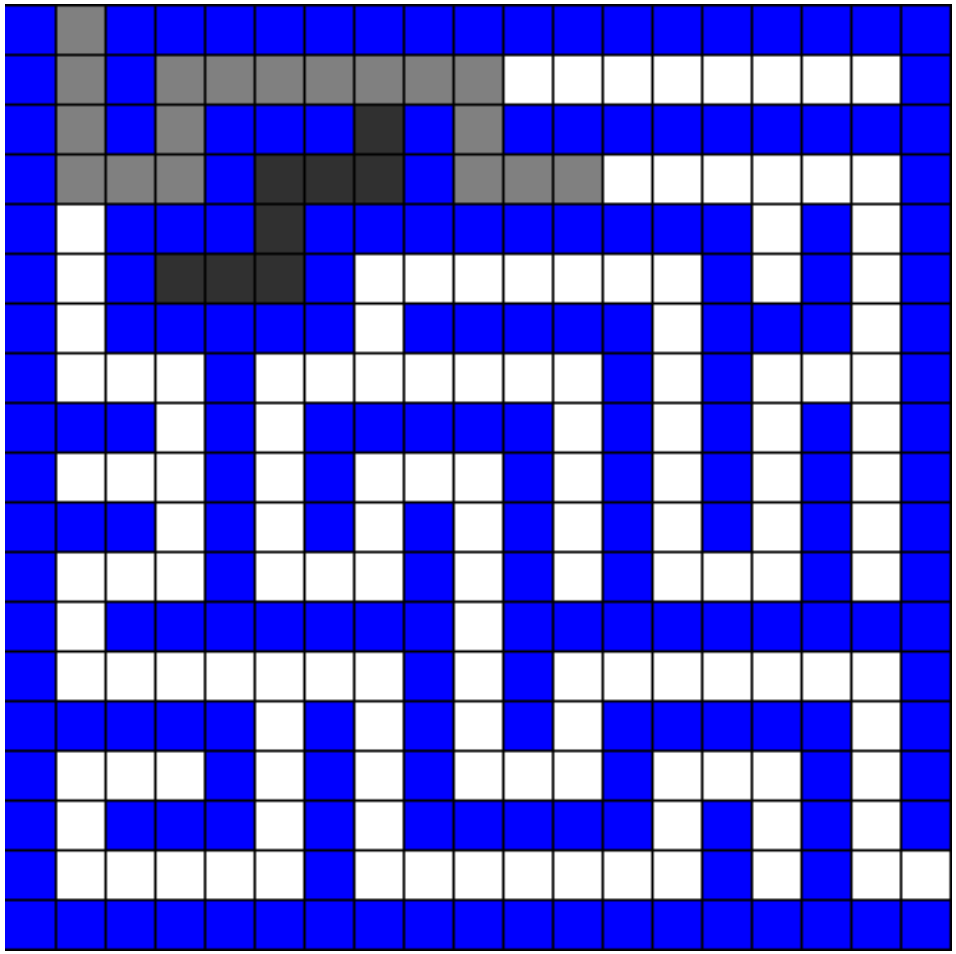
Recorrido de Grafos - DFS



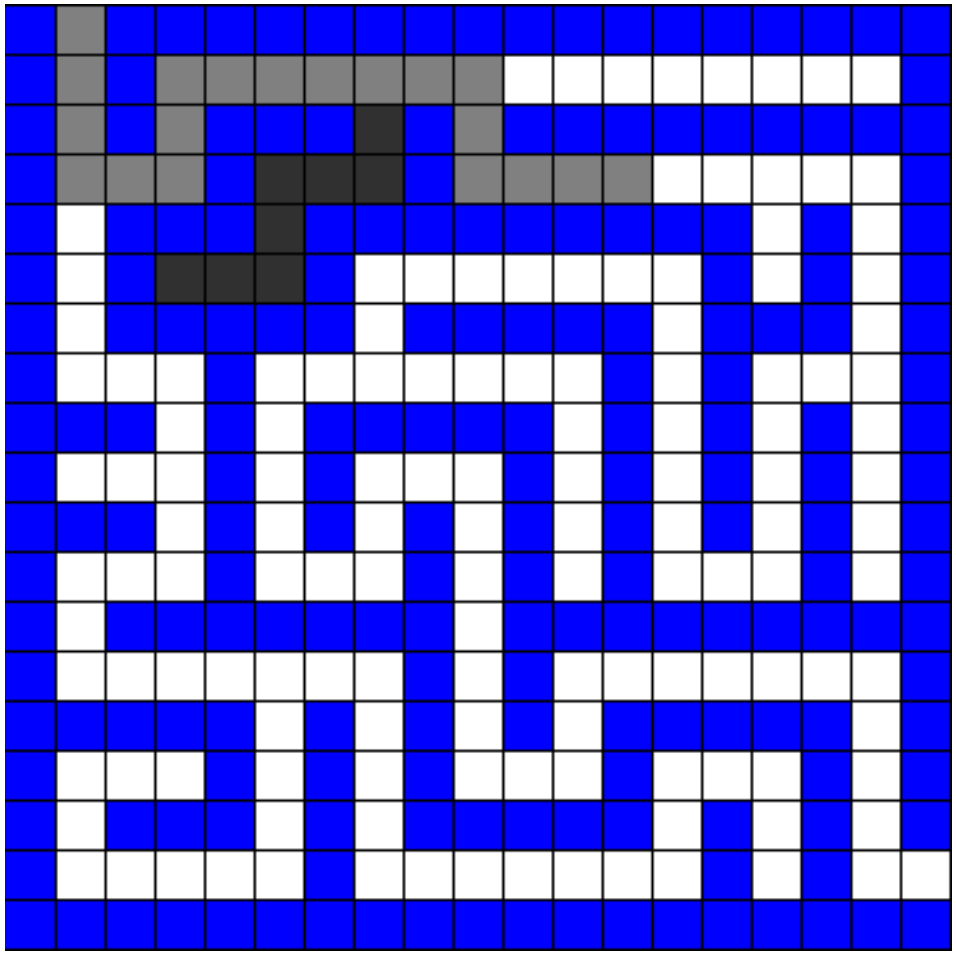
Recorrido de Grafos - DFS



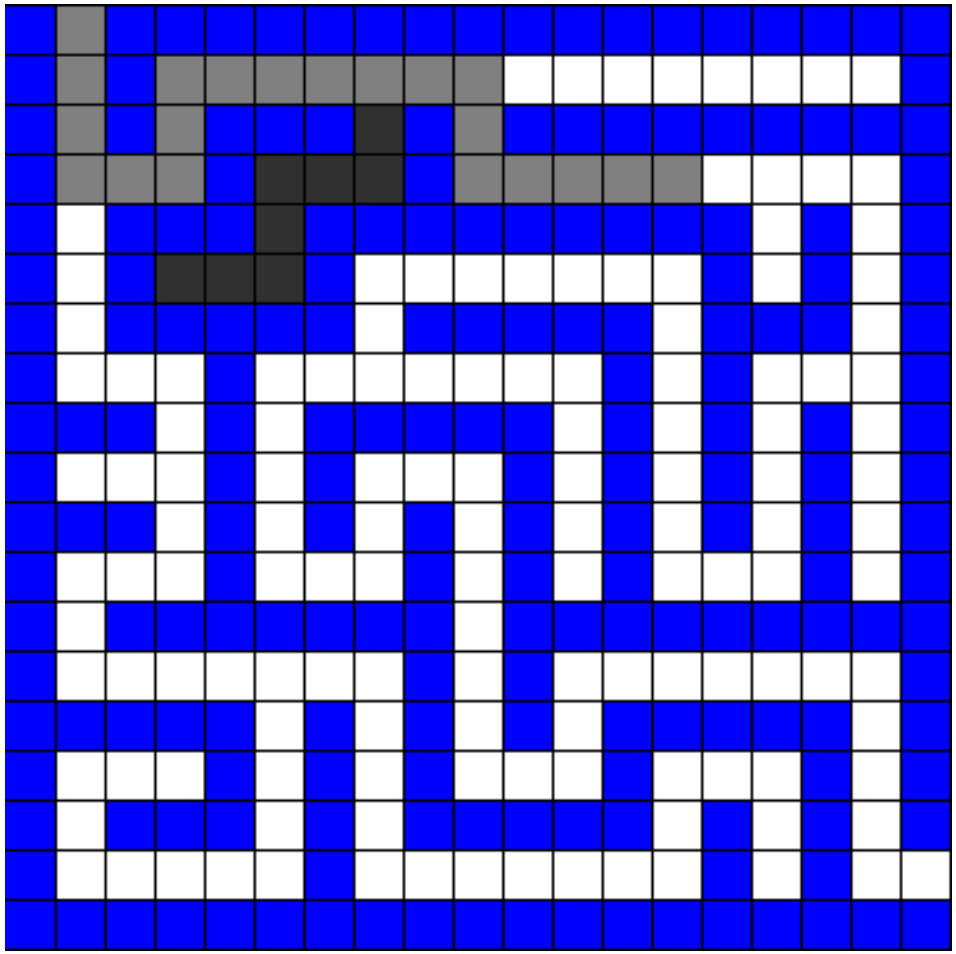
Recorrido de Grafos - DFS



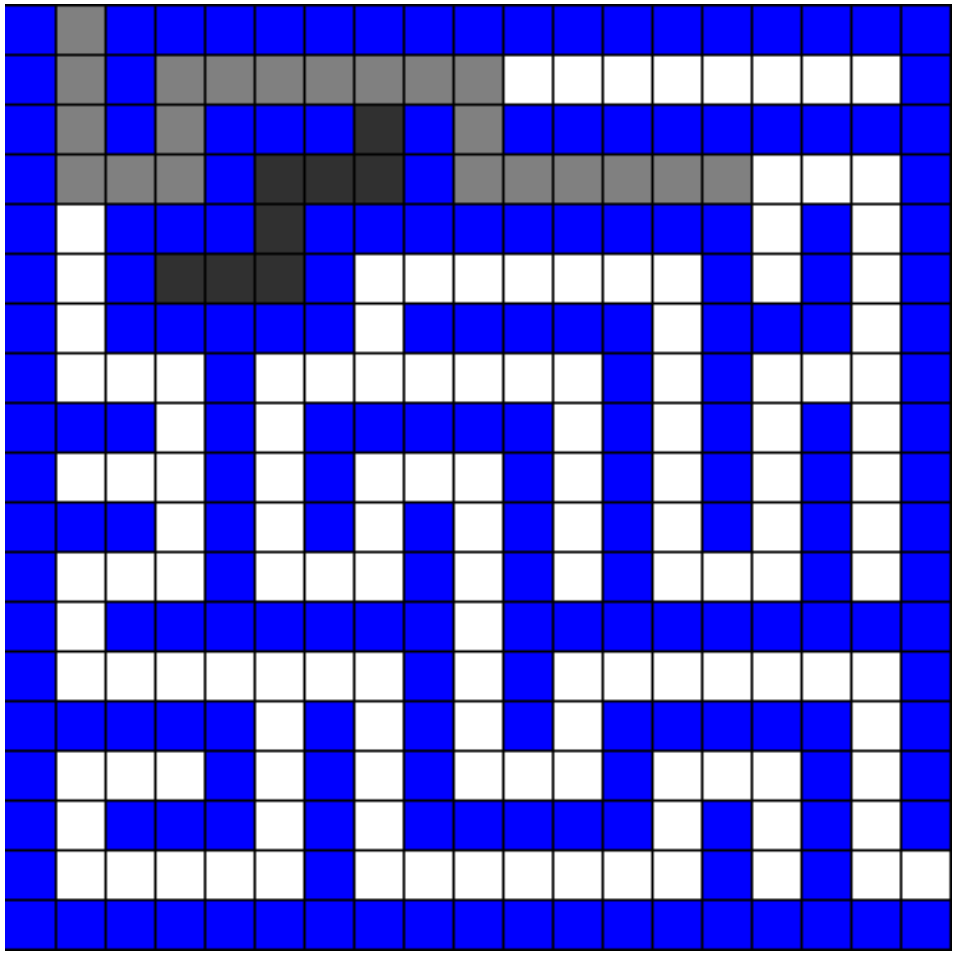
Recorrido de Grafos - DFS



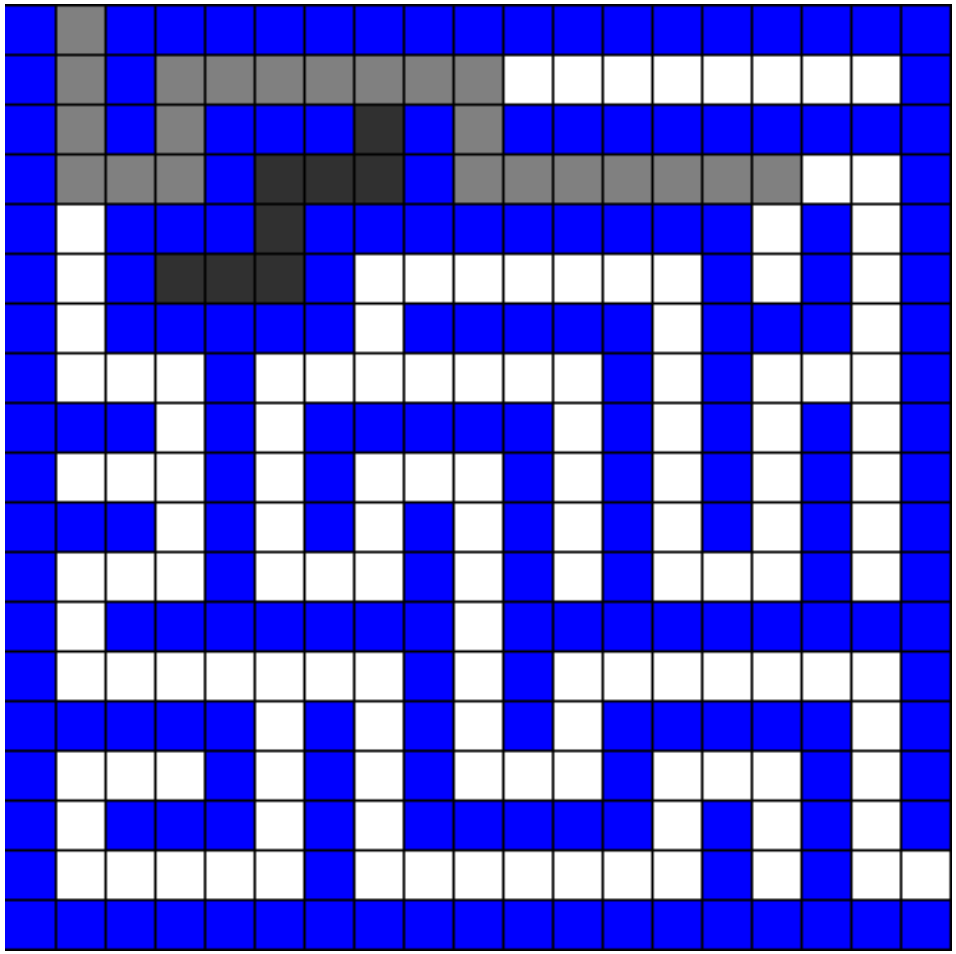
Recorrido de Grafos - DFS



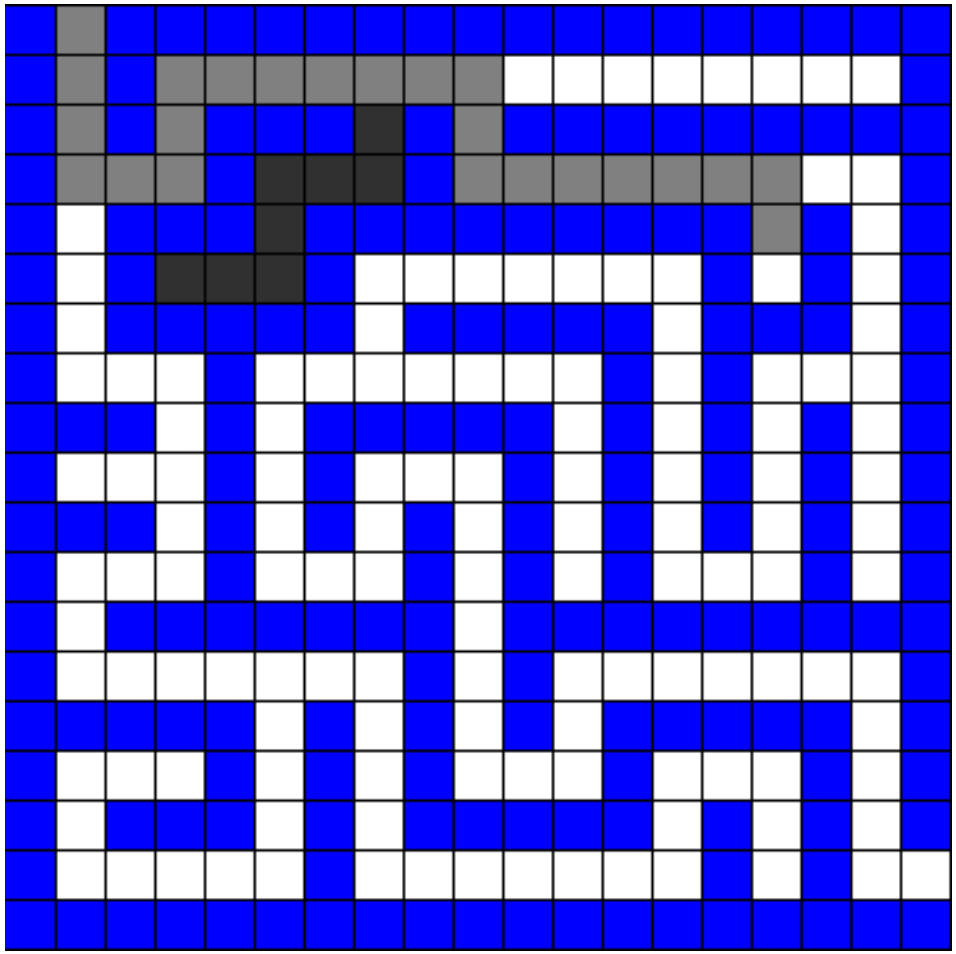
Recorrido de Grafos - DFS



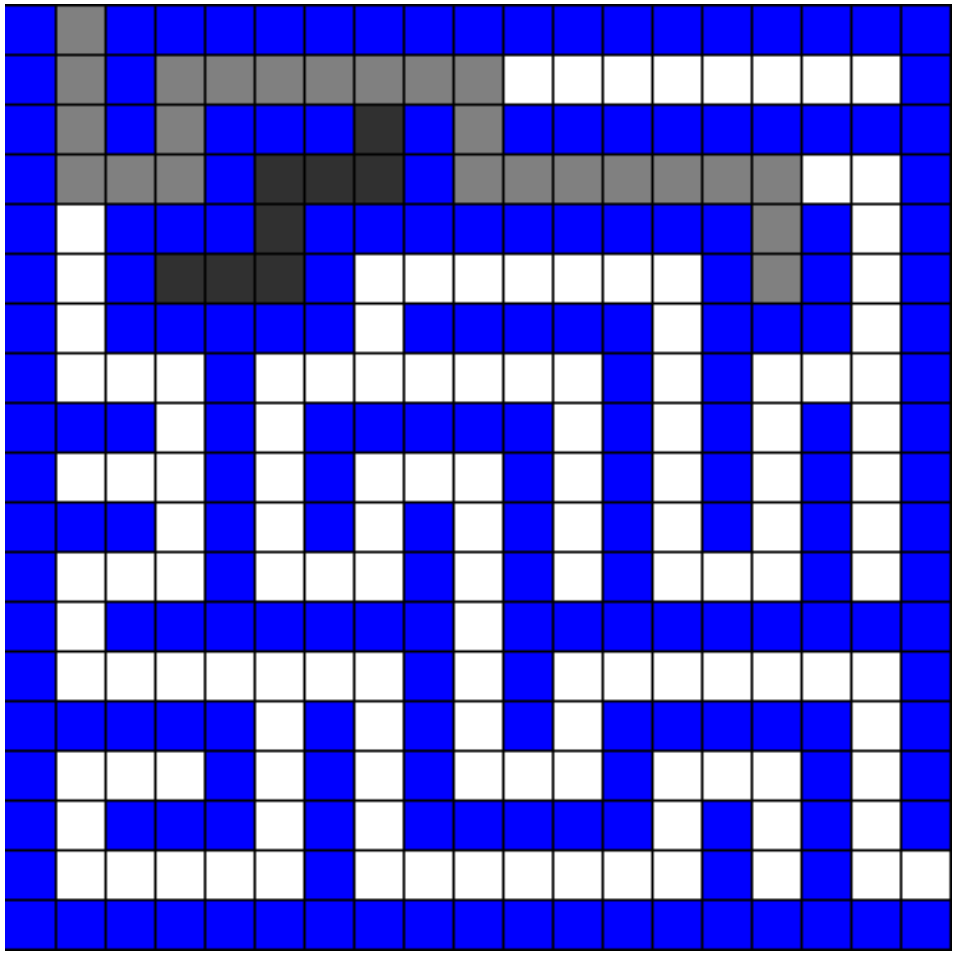
Recorrido de Grafos - DFS



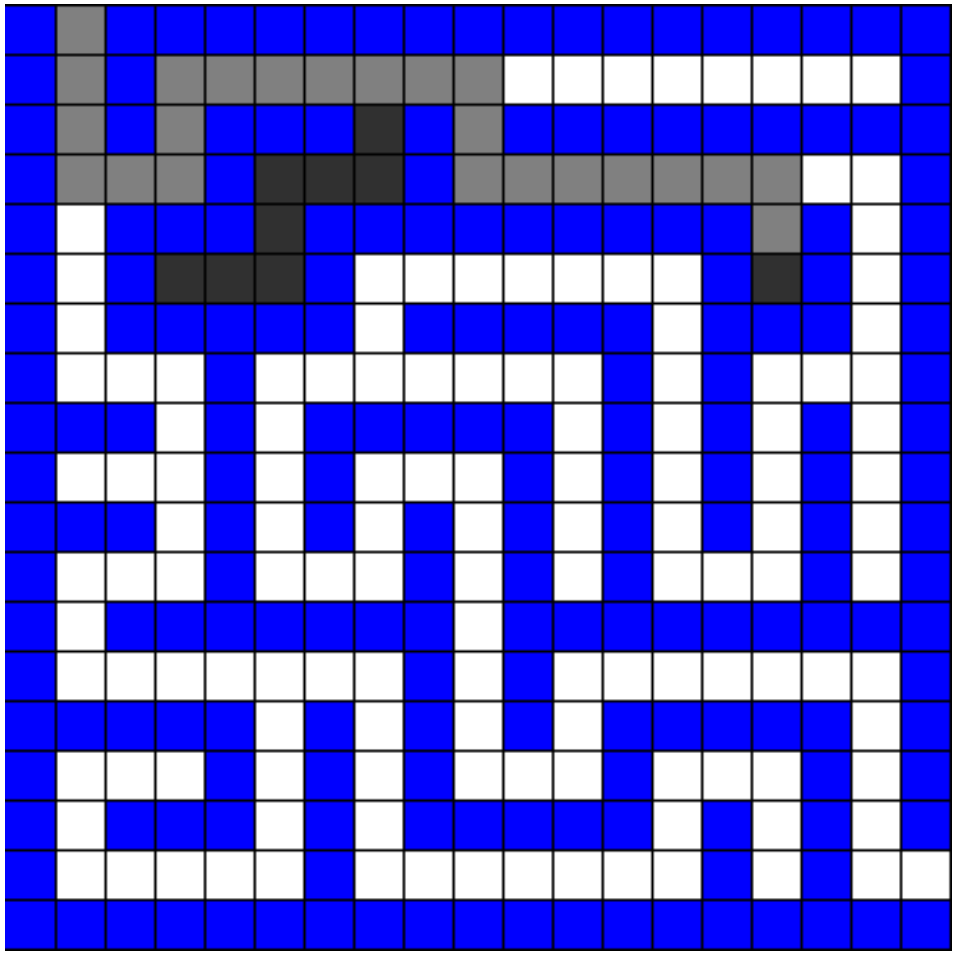
Recorrido de Grafos - DFS



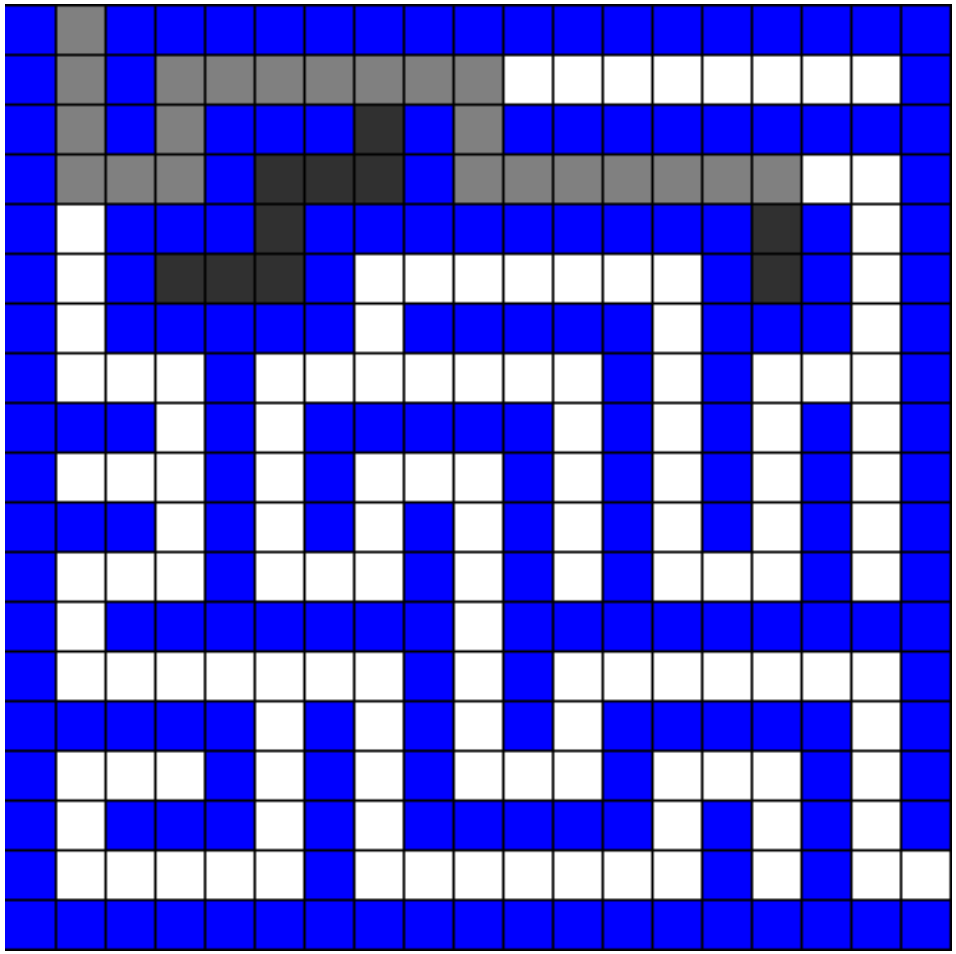
Recorrido de Grafos - DFS



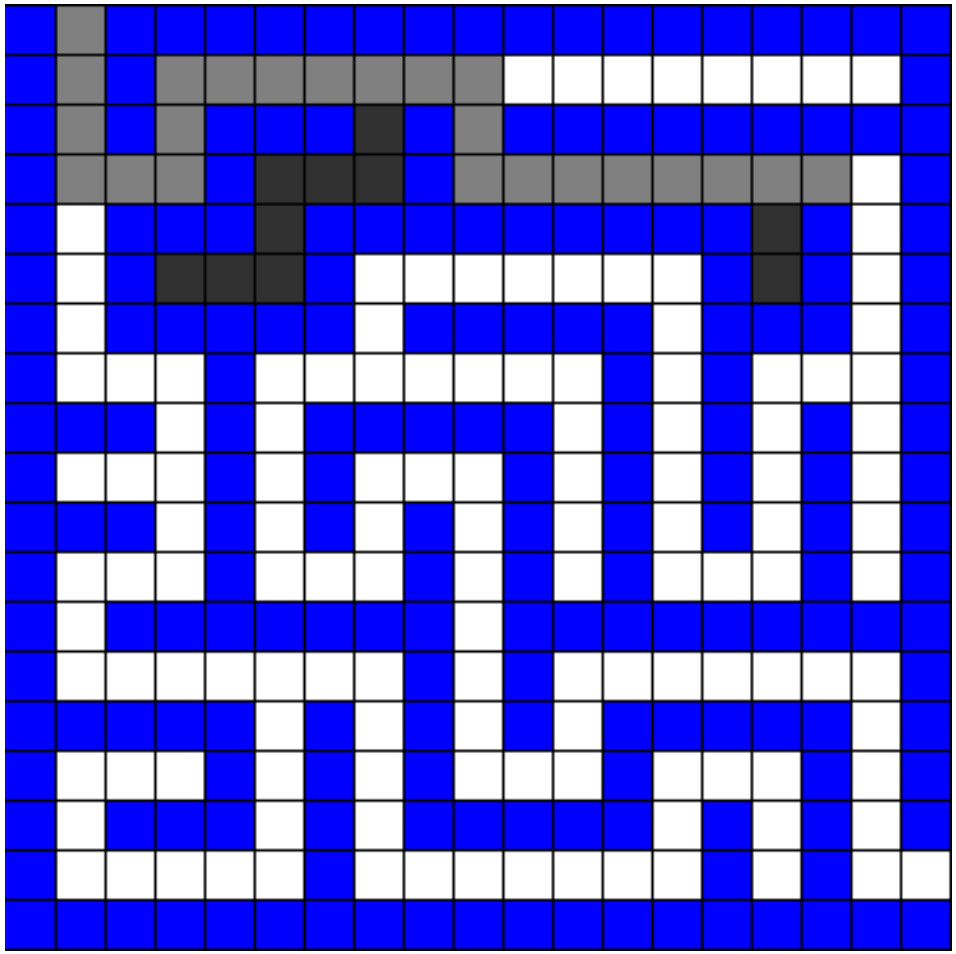
Recorrido de Grafos - DFS



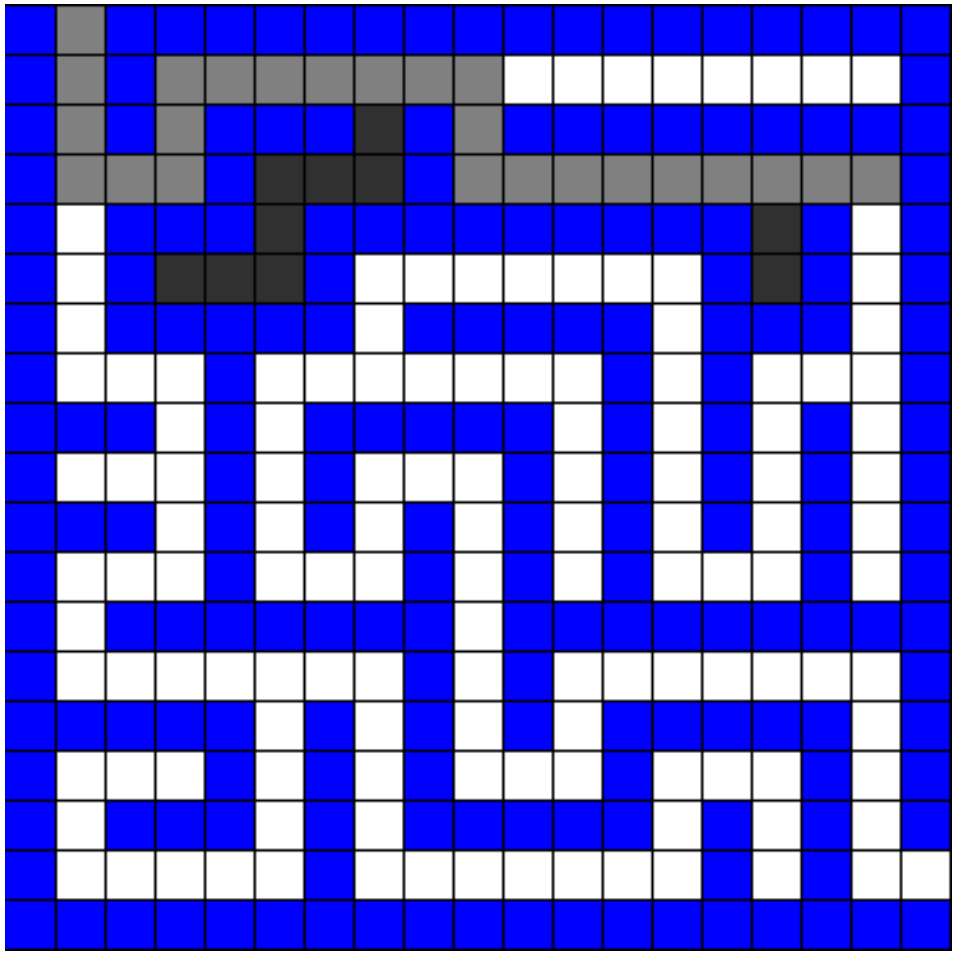
Recorrido de Grafos - DFS



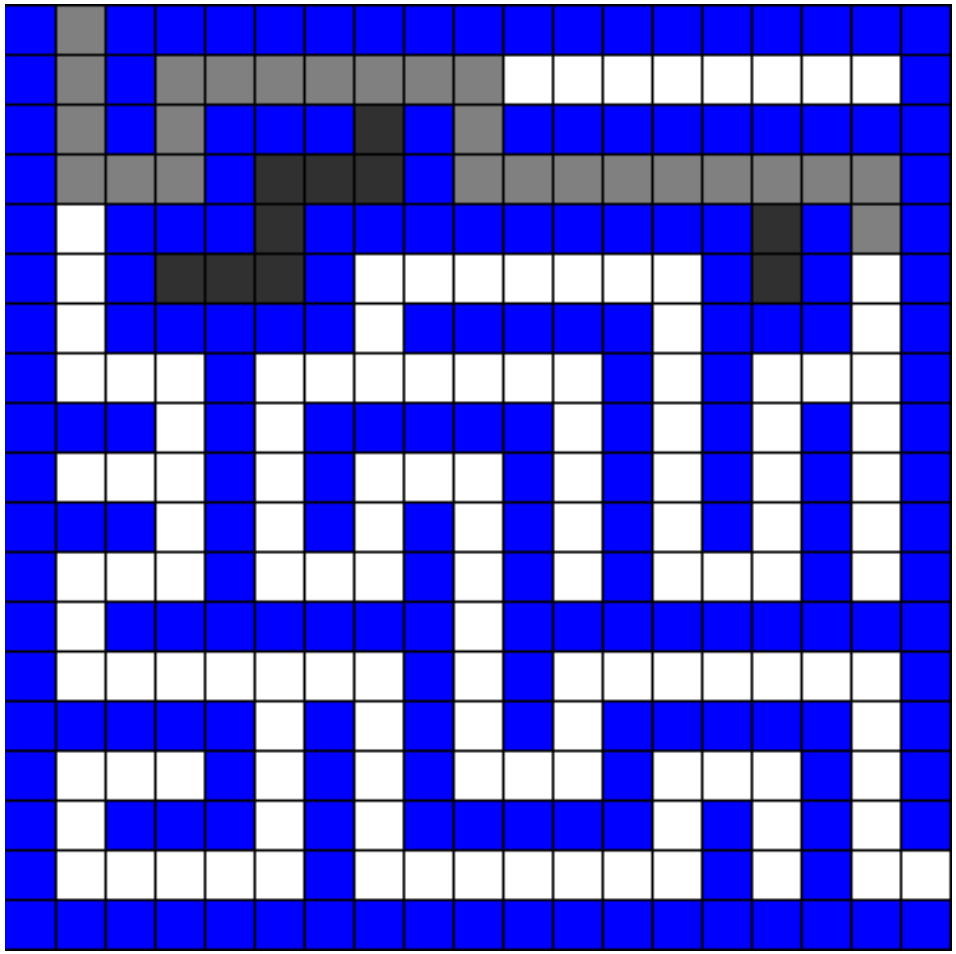
Recorrido de Grafos - DFS



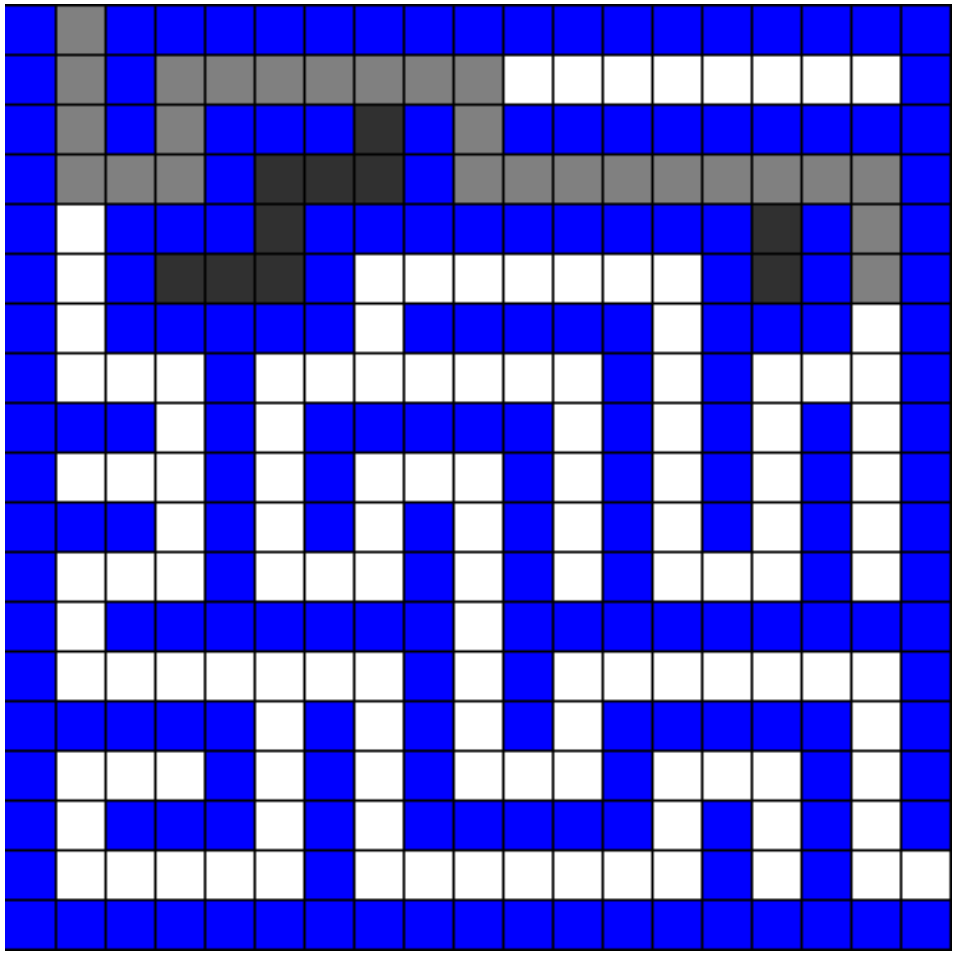
Recorrido de Grafos - DFS



Recorrido de Grafos - DFS



Recorrido de Grafos - DFS



Recorrido de Grafos - DFS

Algunos usos del algoritmo DFS para obtener información estructural del grafo incluyen:

- Componentes conexas
- Puentes, puntos de articulación, componentes biconexas
- Componentes fuertemente conexas
- Orden topológico (*toposort*) en DAGs.

Recorrido de Grafos - DFS

Algunos usos del algoritmo DFS para obtener información estructural del grafo incluyen:

- Componentes conexas
- Puentes, puntos de articulación, componentes biconexas
- Componentes fuertemente conexas
- Orden topológico (*toposort*) en DAGs.

Para más información sobre BFS, DFS, recorrido de grafos en general y muchos temas más, están invitados a visitar:
http://wiki.oia.unsam.edu.ar
Además, siempre pueden hacer consultas sobre problemas (o cualquier tema relacionado con la Olimpíada) en el Foro de la OIA:
http://foro.oia.unsam.edu.ar

En esta charla vamos a ver **cómo utilizar grafos para modelar la situación de un problema**. Una vez modelado nuestro problema, es de esperar que lo que nos pidan calcular en el problema **se transforme en una pregunta puntual** sobre el **grafo resultante**.

En esta charla vamos a ver **cómo utilizar grafos para modelar la situación de un problema**. Una vez modelado nuestro problema, es de esperar que lo que nos pidan calcular en el problema **se transforme en una pregunta puntual** sobre el **grafo resultante**. Por ejemplo, una vez modelada la situación con un grafo conveniente, luego podríamos tener que resolver ...

- Camino mínimo entre dos nodos

En esta charla vamos a ver **cómo utilizar grafos para modelar la situación de un problema**. Una vez modelado nuestro problema, es de esperar que lo que nos pidan calcular en el problema **se transforme en una pregunta puntual** sobre el **grafo resultante**. Por ejemplo, una vez modelada la situación con un grafo conveniente, luego podríamos tener que resolver ...

- Camino mínimo entre dos nodos
- La cantidad de nodos en la componente conexa de algún nodo en particular

En esta charla vamos a ver **cómo utilizar grafos para modelar la situación de un problema**. Una vez modelado nuestro problema, es de esperar que lo que nos pidan calcular en el problema **se transforme en una pregunta puntual** sobre el **grafo resultante**. Por ejemplo, una vez modelada la situación con un grafo conveniente, luego podríamos tener que resolver ...

- Camino mínimo entre dos nodos
- La cantidad de nodos en la componente conexa de algún nodo en particular
- Saber si dos nodos están conectados

En esta charla vamos a ver **cómo utilizar grafos para modelar la situación de un problema**. Una vez modelado nuestro problema, es de esperar que lo que nos pidan calcular en el problema **se transforme en una pregunta puntual** sobre el **grafo resultante**. Por ejemplo, una vez modelada la situación con un grafo conveniente, luego podríamos tener que resolver ...

- Camino mínimo entre dos nodos
- La cantidad de nodos en la componente conexa de algún nodo en particular
- Saber si dos nodos están conectados
- Chequear si el grafo es bipartito

En esta charla vamos a ver **cómo utilizar grafos para modelar la situación de un problema**. Una vez modelado nuestro problema, es de esperar que lo que nos pidan calcular en el problema **se transforme en una pregunta puntual** sobre el **grafo resultante**. Por ejemplo, una vez modelada la situación con un grafo conveniente, luego podríamos tener que resolver ...

- Camino mínimo entre dos nodos
- La cantidad de nodos en la componente conexa de algún nodo en particular
- Saber si dos nodos están conectados
- Chequear si el grafo es bipartito
- Alguna otra pregunta ad hoc que dependerá del problema

En muchas situaciones tendremos que tomar un **vértice por cada estado posible** distinto que puede haber en el problema y luego tendremos una **arista por cada transición posible entre dos estados**. En algunos casos estas aristas podrían tener un peso que indica cuánto nos cuesta dicha transición.

En muchas situaciones tendremos que tomar un **vértice por cada estado posible** distinto que puede haber en el problema y luego tendremos una **arista por cada transición posible entre dos estados**. En algunos casos estas aristas podrían tener un peso que indica cuánto nos cuesta dicha transición.
Veamos esto a través de un ejemplo.

Ahorrando gasolina

Problema:

Javier quiere viajar desde la ciudad A a la ciudad B. En total hay N ciudades y M rutas doble mano que conectan un par de ciudades. El tanque del auto de Javier puede almacenar hasta C litros de gasolina. Al comenzar el recorrido Javier llenó el tanque. Para recorrer una ruta que conecta dos ciudades (i, j) debemos gastar W_{ij} litros de gasolina.
Además, en cada ciudad podemos cargar el tanque al precio de D_i pesos por litro. ¿Cuál es la menor cantidad de dinero que debemos gastar para llegar de A a B?

Ahorrando gasolina

Problema:

Javier quiere viajar desde la ciudad A a la ciudad B. En total hay N ciudades y M rutas doble mano que conectan un par de ciudades. El tanque del auto de Javier puede almacenar hasta C litros de gasolina. Al comenzar el recorrido Javier llenó el tanque. Para recorrer una ruta que conecta dos ciudades (i, j) debemos gastar W_{ij} litros de gasolina.
Además, en cada ciudad podemos cargar el tanque al precio de D_i pesos por litro. ¿Cuál es la menor cantidad de dinero que debemos gastar para llegar de A a B?

Lo primero que vamos a hacer es identificar los distintos estados posibles por los que podemos transitar al viajar de A a B.

- Claramente en nuestro estado tendrá que estar reflejado en **qué ciudad estamos actualmente**.

- Claramente en nuestro estado tendrá que estar reflejado en **qué ciudad estamos actualmente**.
- La otra información que necesitamos saber es **cuánta gasolina nos queda** en el tanque

- Claramente en nuestro estado tendrá que estar reflejado en **qué ciudad estamos actualmente**.
- La otra información que necesitamos saber es **cuánta gasolina nos queda** en el tanque

¿Cuáles son nuestras posibles transiciones si actualmente estamos en la ciudad i con l litros de gasolina en el tanque? ¿Qué decisiones podemos tomar?

- Claramente en nuestro estado tendrá que estar reflejado en **qué ciudad estamos actualmente**.
- La otra información que necesitamos saber es **cuánta gasolina nos queda** en el tanque

¿Cuáles son nuestras posibles transiciones si actualmente estamos en la ciudad *i* con *l* litros de gasolina en el tanque? ¿Qué decisiones podemos tomar?

- Podemos *movernos a otra ciudad* (si es que nos da la gasolina en el tanque):

- Claramente en nuestro estado tendrá que estar reflejado en **qué ciudad estamos actualmente**.
- La otra información que necesitamos saber es **cuánta gasolina nos queda** en el tanque

¿Cuáles son nuestras posibles transiciones si actualmente estamos en la ciudad i con l litros de gasolina en el tanque? ¿Qué decisiones podemos tomar?

- Podemos *movernos a otra ciudad* (si es que nos da la gasolina en el tanque): $(i, l) \xrightarrow{0} (j, l - W_{ij})$.
- Podemos *cargar un litro de gasolina* en la ciudad actual (si el tanque no está lleno):

- Claramente en nuestro estado tendrá que estar reflejado en **qué ciudad estamos actualmente**.
- La otra información que necesitamos saber es **cuánta gasolina nos queda** en el tanque

¿Cuáles son nuestras posibles transiciones si actualmente estamos en la ciudad i con l litros de gasolina en el tanque? ¿Qué decisiones podemos tomar?

- Podemos *movernos a otra ciudad* (si es que nos da la gasolina en el tanque): $(i, l) \xrightarrow{0} (j, l - W_{ij})$.
- Podemos *cargar un litro de gasolina* en la ciudad actual (si el tanque no está lleno): $(i, l) \xrightarrow{C_i} (i, l + 1)$

- Claramente en nuestro estado tendrá que estar reflejado en **qué ciudad estamos actualmente**.
- La otra información que necesitamos saber es **cuánta gasolina nos queda** en el tanque

¿Cuáles son nuestras posibles transiciones si actualmente estamos en la ciudad i con l litros de gasolina en el tanque? ¿Qué decisiones podemos tomar?

- Podemos *movernos a otra ciudad* (si es que nos da la gasolina en el tanque): $(i, l) \xrightarrow{0} (j, l - W_{ij})$.
- Podemos *cargar un litro de gasolina* en la ciudad actual (si el tanque no está lleno): $(i, l) \xrightarrow{C_i} (i, l + 1)$

Luego, el problema se reduce a encontrar un camino mínimo de (A, C) a algún nodo de la forma (B, l) para algún l.

¡La bici no dobla!

Problema:

Melanie vive en una ciudad que tiene forma de grilla, con H calles horizontales y V calles verticales. Además, algunas esquinas están obstaculizadas por diversos motivos, y no puede pasar por allí. Recientemente Melanie se compró una bicicleta y ya aprendió a andar sin rueditas, pero doblar todavía le cuesta. Le gustaría poder ir en bici de su casa a la facultad *minimizando la cantidad de veces que tiene que doblar en una esquina*. ¿Cómo la podemos ayudar?

¡La bici no dobla!

Problema:

Melanie vive en una ciudad que tiene forma de grilla, con H calles horizontales y V calles verticales. Además, algunas esquinas están obstaculizadas por diversos motivos, y no puede pasar por allí. Recientemente Melanie se compró una bicicleta y ya aprendió a andar sin rueditas, pero doblar todavía le cuesta. Le gustaría poder ir en bici de su casa a la facultad *minimizando la cantidad de veces que tiene que doblar en una esquina*. ¿Cómo la podemos ayudar?

Lo primero que vamos a hacer es identificar los distintos estados posibles por los que podemos transitar al viajar de su casa (C) a la facultad (F).

- En nuestro estado tendrá que estar reflejado en **qué calles estamos actualmente**.

- En nuestro estado tendrá que estar reflejado en **qué calles estamos actualmente**.
- La otra información que necesitamos saber es **en qué dirección nos estamos moviendo** (para saber en qué direcciones estaríamos doblando y en cuál seguimos derecho).

- En nuestro estado tendrá que estar reflejado en **qué calles estamos actualmente**.
- La otra información que necesitamos saber es **en qué dirección nos estamos moviendo** (para saber en qué direcciones estaríamos doblando y en cuál seguimos derecho).

¿Cuáles son nuestras posibles transiciones si actualmente estamos en las calles (h, v) y nos estamos moviendo en la dirección d? ¿Qué decisiones podemos tomar?

A la hora de implementar las transiciones, puede sernos útil tener dos **arreglos auxiliares de movimiento** `dx` **y** `dy` de forma que `dx`$[d]$ nos diga cómo se modifican las calles horizontales al movernos en la dirección d. De la misma forma con `dy`$[d]$ para las calles verticales.

A la hora de implementar las transiciones, puede sernos útil tener dos **arreglos auxiliares de movimiento** `dx` **y** `dy` de forma que `dx`$[d]$ nos diga cómo se modifican las calles horizontales al movernos en la dirección d. De la misma forma con `dy`$[d]$ para las calles verticales.

- Podemos *seguir derecho*: $(h, v, d) \xrightarrow{0} (h + \texttt{dx}[d], v + \texttt{dy}[d], d)$

A la hora de implementar las transiciones, puede sernos útil tener dos **arreglos auxiliares de movimiento** `dx` **y** `dy` de forma que $\texttt{dx}[d]$ nos diga cómo se modifican las calles horizontales al movernos en la dirección d. De la misma forma con $\texttt{dy}[d]$ para las calles verticales.

- Podemos *seguir derecho*: $(h, v, d) \xrightarrow{0} (h + \texttt{dx}[d], v + \texttt{dy}[d], d)$
- Podemos *doblar*: $(h, v, d) \xrightarrow{1} (h, v, \hat{d})$ $\quad$ $(h, v, d) \xrightarrow{1} (h, v, \tilde{d})$

A la hora de implementar las transiciones, puede sernos útil tener dos **arreglos auxiliares de movimiento** `dx` **y** `dy` de forma que $\texttt{dx}[d]$ nos diga cómo se modifican las calles horizontales al movernos en la dirección d. De la misma forma con $\texttt{dy}[d]$ para las calles verticales.

- Podemos *seguir derecho*: $(h, v, d) \xrightarrow{0} (h + \texttt{dx}[d], v + \texttt{dy}[d], d)$
- Podemos *doblar*: $(h, v, d) \xrightarrow{1} (h, v, \hat{d}) \quad (h, v, d) \xrightarrow{1} (h, v, \tilde{d})$

Finalmente ¿qué debemos hacer en este grafo para resolver el problema?

A la hora de implementar las transiciones, puede sernos útil tener dos **arreglos auxiliares de movimiento** `dx` **y** `dy` de forma que `dx`$[d]$ nos diga cómo se modifican las calles horizontales al movernos en la dirección d. De la misma forma con `dy`$[d]$ para las calles verticales.

- Podemos *seguir derecho*: $(h, v, d) \xrightarrow{0} (h + \texttt{dx}[d], v + \texttt{dy}[d], d)$
- Podemos *doblar*: $(h, v, d) \xrightarrow{1} (h, v, \hat{d}) \quad (h, v, d) \xrightarrow{1} (h, v, \tilde{d})$

Finalmente ¿qué debemos hacer en este grafo para resolver el problema?
Debemos encontrar el camino más corto que va de algún nodo (C, d) para toda dirección d posible, y llega a algún nodo de la forma $(F, \hat{d})$ para alguna dirección $\hat{d}$

Este problema es una joya

Este problema es una joya

El proceso de manufactura de una joya comienza con un pedazo de plata en bruto. La plata se tiene que *tallar*, *pulir*, *agujerear* y *marcar*:

- Hay que *tallarla* antes que *agujerearla*, y *pulirla* antes que *marcarla*

Además, cada proceso demora una cierta cantidad de tiempo.

- El proceso de tallado y el de pulido demoran 1 unidad de tiempo.
- El proceso de marcado requiere de 2 unidades de tiempo si la pieza no está tallada y 4 unidades de tiempo si la pieza está tallada.
- Agujerearla requiere de 3 unidades de tiempo si la pieza no está pulida. Si la pieza está pulida, pero no marcada, entonces agujerearla requiere de 5 unidades de tiempo. Por otro lado, si la pieza está pulida y también marcada, se requieren 7 unidades de tiempo para agujerearla.

¿Cuál es el menor tiempo en el que podemos manufacturar la joya?

Lo primero que vamos a hacer es identificar los distintos estados posibles del problema.

Lo primero que vamos a hacer es identificar los distintos estados posibles del problema.
En todo momento, cada uno de los 4 procesos puede haber sido realizado en la joya o no. Por lo tanto, en total tenemos 2^4 estados posibles. Vamos a identificar un estado de la joya con una tira binaria de largo 4.

Lo primero que vamos a hacer es identificar los distintos estados posibles del problema.

En todo momento, cada uno de los 4 procesos puede haber sido realizado en la joya o no. Por lo tanto, en total tenemos 2^4 estados posibles. Vamos a identificar un estado de la joya con una tira binaria de largo 4.

Por ejemplo, si la joya en este momento está *tallada* y *agujereada*, pero no *pulida* ni *marcada*, entonces el estado actual de la joya será: $(\overset{T}{1}, \overset{P}{0}, \overset{A}{1}, \overset{M}{0})$.

Lo primero que vamos a hacer es identificar los distintos estados posibles del problema.

En todo momento, cada uno de los 4 procesos puede haber sido realizado en la joya o no. Por lo tanto, en total tenemos 2^4 estados posibles. Vamos a identificar un estado de la joya con una tira binaria de largo 4.

Por ejemplo, si la joya en este momento está *tallada* y *agujereada*, pero no *pulida* ni *marcada*, entonces el estado actual de la joya será: $(\overset{T}{1}, \overset{P}{0}, \overset{A}{1}, \overset{M}{0})$.

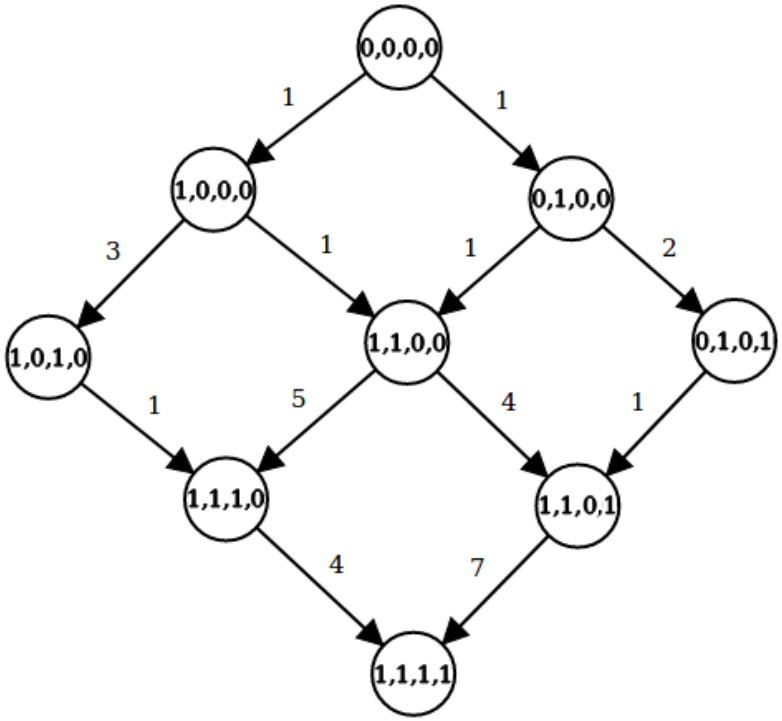
En realidad, el enunciado nos asegura que algunos estados no son posibles.

Hay que tallarla antes que agujerearla

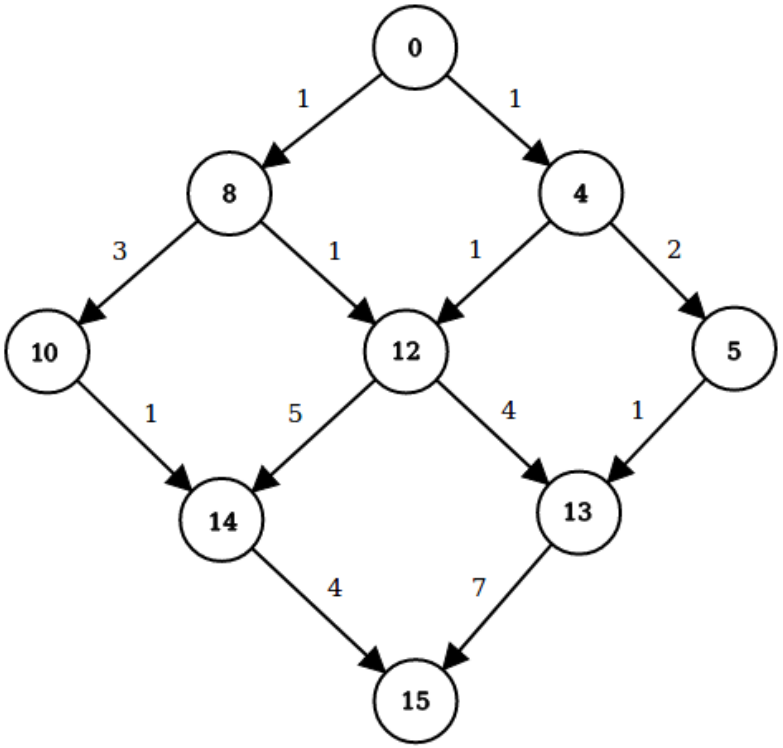
Eso quiere decir que todas los estados de la forma $(\overset{T}{0}, \overset{P}{\bigstar_1}, \overset{A}{1}, \overset{M}{\bigstar_2})$ para cualquier valor de $\bigstar_1, \bigstar_2$ no son posibles.

¿Cuáles son las transiciones? ¿Qué problema debemos resolver?

¿Cuáles son las transiciones? ¿Qué problema debemos resolver?



¿Cuáles son las transiciones? ¿Qué problema debemos resolver?



Muchas veces, las operaciones de bits nos ayudan a implementar estas cosas de forma relativamente sencilla.

http://wiki.oia.unsam.edu.ar/cpp-avanzado/operaciones-de-bits

La comunidad del anillo

Problema:

Se tienen n ciudades que forman el llamado "anillo de la plata". Las ciudades i e $i+1$ están conectadas por un camino (para $1 \leq i < n$) y de la ciudad n a la 1 también. El gobierno decidió construir M nuevos caminos que conectan un par de ciudades. Cada uno de estos caminos debe estar completamente dentro del anillo o completamente por fuera del anillo. ¿Es posible construir estas M rutas de forma que ninguna de estas se corten entre ellas (salvo en los extremos)?¿Qué rutas deberían estar por dentro y cuáles por fuera?

La comunidad del anillo

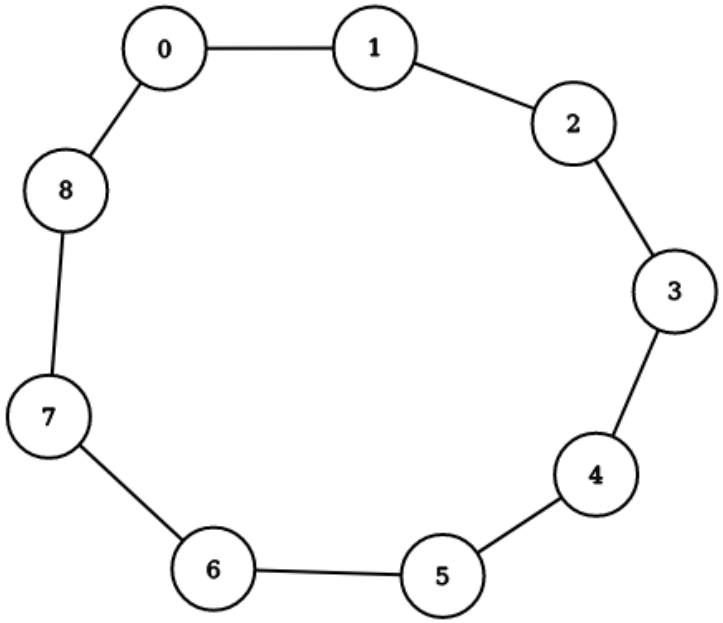
Problema:

Se tienen n ciudades que forman el llamado "anillo de la plata". Las ciudades i e $i+1$ están conectadas por un camino (para $1 \leq i < n$) y de la ciudad n a la 1 también. El gobierno decidió construir M nuevos caminos que conectan un par de ciudades. Cada uno de estos caminos debe estar completamente dentro del anillo o completamente por fuera del anillo. ¿Es posible construir estas M rutas de forma que ninguna de estas se corten entre ellas (salvo en los extremos)?¿Qué rutas deberían estar por dentro y cuáles por fuera?

Analicemos el grafo que nos viene dado . . .

Imaginemos que tenemos que ubicar las aristas:

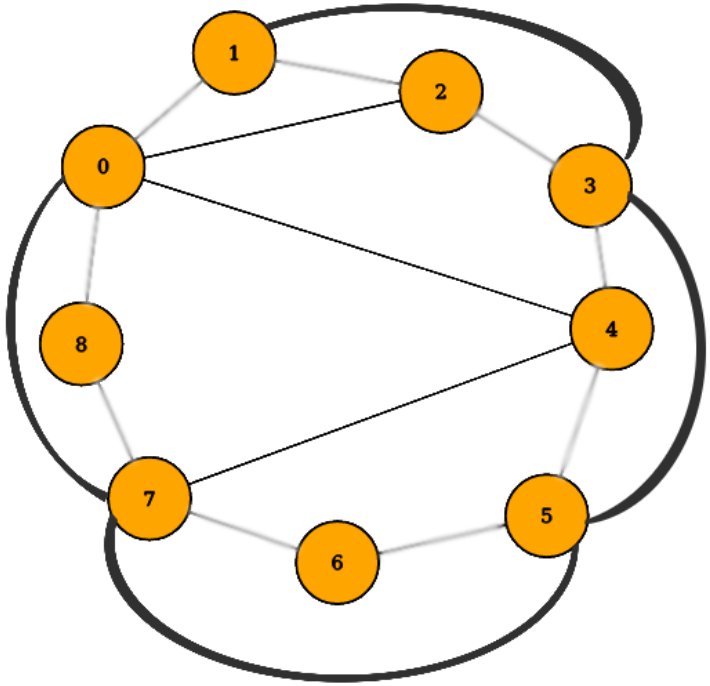
1. (0,2)
2. (0,4)
3. (0,7)
4. (1,3)
5. (3,5)
6. (4,7)
7. (5,7)



Imaginemos que tenemos que ubicar las aristas:

1. (0,2)
2. (0,4)
3. (0,7)
4. (1,3)
5. (3,5)
6. (4,7)
7. (5,7)

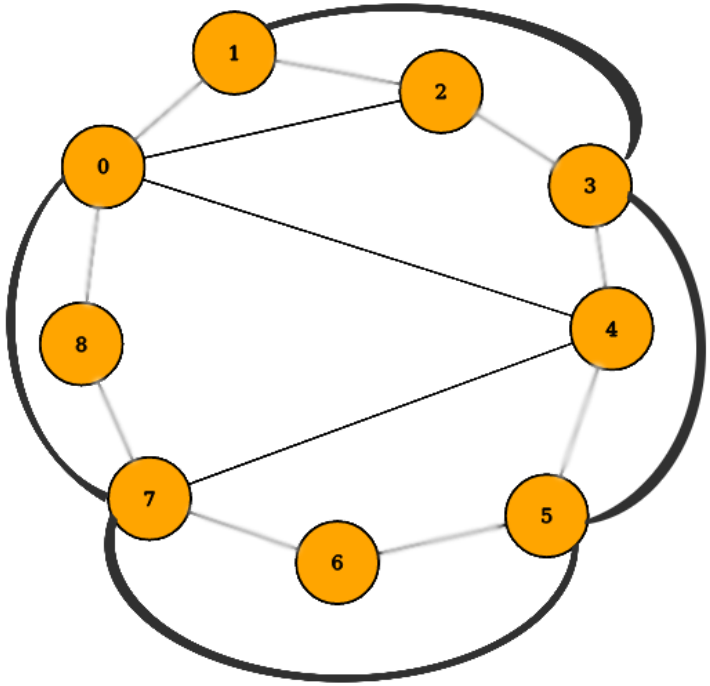
Una forma posible sería la siguiente



Imaginemos que tenemos que ubicar las aristas:

1. (0,2)
2. (0,4)
3. (0,7)
4. (1,3)
5. (3,5)
6. (4,7)
7. (5,7)

Una forma posible sería la siguiente



¿Qué nos importa realmente a la hora de resolver el problema?
¿Nos importa el grafo del anillo que nos dan?

¿Cuándo podemos afirmar que dos aristas necesariamente van en lados distintos?

¿Cuándo podemos afirmar que dos aristas necesariamente van en lados distintos?
Cada arista nos define un intervalo (abierto, pues si se tocan en los nodos no hay problema), y dos aristas **pueden ir del mismo lado, si los intervalos son disjuntos o un intervalo está contenido en el otro**.

¿Cuándo podemos afirmar que dos aristas necesariamente van en lados distintos?
Cada arista nos define un intervalo (abierto, pues si se tocan en los nodos no hay problema), y dos aristas **pueden ir del mismo lado, si los intervalos son disjuntos o un intervalo está contenido en el otro**.
Entonces, podemos generarnos el grafo que tiene un vértice por cada arista que nos dan, y en este nuevo grafo vamos a ubicar una arista entre dos vértices si las aristas que corresponden no pueden ir del mismo lado.

¿Cuándo podemos afirmar que dos aristas necesariamente van en lados distintos?
Cada arista nos define un intervalo (abierto, pues si se tocan en los nodos no hay problema), y dos aristas **pueden ir del mismo lado, si los intervalos son disjuntos o un intervalo está contenido en el otro**.
Entonces, podemos generarnos el grafo que tiene un vértice por cada arista que nos dan, y en este nuevo grafo vamos a ubicar una arista entre dos vértices si las aristas que corresponden no pueden ir del mismo lado.
¿Qué problema debemos resolver en este nuevo grafo para resolver el problema original?

¿Cuándo podemos afirmar que dos aristas necesariamente van en lados distintos?
Cada arista nos define un intervalo (abierto, pues si se tocan en los nodos no hay problema), y dos aristas **pueden ir del mismo lado, si los intervalos son disjuntos o un intervalo está contenido en el otro**.
Entonces, podemos generarnos el grafo que tiene un vértice por cada arista que nos dan, y en este nuevo grafo vamos a ubicar una arista entre dos vértices si las aristas que corresponden no pueden ir del mismo lado.
¿Qué problema debemos resolver en este nuevo grafo para resolver el problema original? **Debemos ver si podemos pintarlo de dos colores sin que haya aristas con extremos del mismo color** (o sea, si es bipartito). Cada color, nos dice de qué lado dibujamos a cada arista.

¿Cuándo podemos afirmar que dos aristas necesariamente van en lados distintos?
Cada arista nos define un intervalo (abierto, pues si se tocan en los nodos no hay problema), y dos aristas **pueden ir del mismo lado, si los intervalos son disjuntos o un intervalo está contenido en el otro**.
Entonces, podemos generarnos el grafo que tiene un vértice por cada arista que nos dan, y en este nuevo grafo vamos a ubicar una arista entre dos vértices si las aristas que corresponden no pueden ir del mismo lado.
¿Qué problema debemos resolver en este nuevo grafo para resolver el problema original? **Debemos ver si podemos pintarlo de dos colores sin que haya aristas con extremos del mismo color** (o sea, si es bipartito). Cada color, nos dice de qué lado dibujamos a cada arista. http://codeforces.com/contest/27/problem/D

Babel

Juan y María son fanáticos de aprender idiomas nuevos, y últimamente se divierten con palabras que se escriben igual en dos idiomas distintos pero tienen distinto significado. Por ejemplo, en castellano una "red" puede utilizarse para pescar, pero "red" en inglés hace referencia al color rojo.

El juego que inventaron consiste en **elegir dos idiomas**: I_1 e I_2. Luego deben **encontrar una secuencia de palabras** de forma que la primera palabra tenga algún significado en I_1, la última palabra tenga algún significado en I_2, y que las palabras adyacentes en la secuencia tengan significado en algún idioma común. De todas las secuencias de palabras buscan la de **menor cantidad de letras**.

Para hacerlo más interesante, decidieron **no permitir utilizar dos palabras consecutivas que comiencen con la misma letra**.

Solo utilizan palabras con significado en *exactamente* dos idiomas.

- Si los idiomas que eligen son Portugués y Francés, entonces : "amigo actual date" sería una secuencia posible (Portugués/Español, Español/Inglés, Inglés/Francés) de no ser porque "amigo" y "actual" comienzan ambas con la letra *a*

- Si los idiomas que eligen son Portugués y Francés, entonces : "amigo actual date" sería una secuencia posible (Portugués/Español, Español/Inglés, Inglés/Francés) de no ser porque "amigo" y "actual" comienzan ambas con la letra *a*
- "amigo red date" es una secuencia posible (Portugués/Español, Español/Inglés, Inglés/Francés)

- Si los idiomas que eligen son Portugués y Francés, entonces : "amigo actual date" sería una secuencia posible (Portugués/Español, Español/Inglés, Inglés/Francés) de no ser porque "amigo" y "actual" comienzan ambas con la letra *a*
- "amigo red date" es una secuencia posible (Portugués/Español, Español/Inglés, Inglés/Francés)

Imaginemos que en la entrada nos dan una lista de M palabras, cada una de ellas de la forma $\mathtt{I}_1, \mathtt{I}_2, \mathtt{p}_i$, diciendo que la i-ésima palabra tiene significados distintos en los idiomas I_1 e I_2.

- Si los idiomas que eligen son Portugués y Francés, entonces : "amigo actual date" sería una secuencia posible (Portugués/Español, Español/Inglés, Inglés/Francés) de no ser porque "amigo" y "actual" comienzan ambas con la letra *a*
- "amigo red date" es una secuencia posible (Portugués/Español, Español/Inglés, Inglés/Francés)

Imaginemos que en la entrada nos dan una lista de M palabras, cada una de ellas de la forma $\mathtt{I}_1, \mathtt{I}_2, \mathtt{p}_i$, diciendo que la i-ésima palabra tiene significados distintos en los idiomas I_1 e I_2.
¿Cuáles son nuestros estados posibles?

- Si los idiomas que eligen son Portugués y Francés, entonces : "amigo actual date" sería una secuencia posible (Portugués/Español, Español/Inglés, Inglés/Francés) de no ser porque "amigo" y "actual" comienzan ambas con la letra *a*
- "amigo red date" es una secuencia posible (Portugués/Español, Español/Inglés, Inglés/Francés)

Imaginemos que en la entrada nos dan una lista de M palabras, cada una de ellas de la forma $\mathtt{I}_1, \mathtt{I}_2, \mathtt{p}_i$, diciendo que la i-ésima palabra tiene significados distintos en los idiomas I_1 e I_2.
¿Cuáles son nuestros estados posibles? (*Idioma*, *letraProhibida*)

- Si los idiomas que eligen son Portugués y Francés, entonces : "amigo actual date" sería una secuencia posible (Portugués/Español, Español/Inglés, Inglés/Francés) de no ser porque "amigo" y "actual" comienzan ambas con la letra *a*
- "amigo red date" es una secuencia posible (Portugués/Español, Español/Inglés, Inglés/Francés)

Imaginemos que en la entrada nos dan una lista de M palabras, cada una de ellas de la forma $\mathtt{I}_1, \mathtt{I}_2, \mathtt{p}_i$, diciendo que la i-ésima palabra tiene significados distintos en los idiomas I_1 e I_2.
¿Cuáles son nuestros estados posibles? (*Idioma*, *letraProhibida*)
¿Cuáles son nuestras transiciones?

- Si los idiomas que eligen son Portugués y Francés, entonces : "amigo actual date" sería una secuencia posible (Portugués/Español, Español/Inglés, Inglés/Francés) de no ser porque "amigo" y "actual" comienzan ambas con la letra *a*
- "amigo red date" es una secuencia posible (Portugués/Español, Español/Inglés, Inglés/Francés)

Imaginemos que en la entrada nos dan una lista de M palabras, cada una de ellas de la forma $\mathtt{I}_1, \mathtt{I}_2, \mathtt{p}_i$, diciendo que la i-ésima palabra tiene significados distintos en los idiomas I_1 e I_2.
¿Cuáles son nuestros estados posibles? $(Idioma, letraProhibida)$
¿Cuáles son nuestras transiciones?
$(I_1, letraProhibida) \xrightarrow{\texttt{largo(p)}} (I_2, \mathtt{p}[0])$, siempre que $\mathtt{p}$ tenga significados distintos en I_1 e I_2 y $\mathtt{p}[0] \neq letraProhibida$

- Si los idiomas que eligen son Portugués y Francés, entonces : "amigo actual date" sería una secuencia posible (Portugués/Español, Español/Inglés, Inglés/Francés) de no ser porque "amigo" y "actual" comienzan ambas con la letra *a*
- "amigo red date" es una secuencia posible (Portugués/Español, Español/Inglés, Inglés/Francés)

Imaginemos que en la entrada nos dan una lista de M palabras, cada una de ellas de la forma $\mathtt{I}_1, \mathtt{I}_2, \mathtt{p}_i$, diciendo que la i-ésima palabra tiene significados distintos en los idiomas I_1 e I_2.
¿Cuáles son nuestros estados posibles? (*Idioma*, *letraProhibida*)
¿Cuáles son nuestras transiciones?
$(I_1, \mathit{letraProhibida}) \xrightarrow{\texttt{largo(p)}} (I_2, \mathtt{p}[0])$, siempre que $\mathtt{p}$ tenga significados distintos en I_1 e I_2 y $\mathtt{p}[0] \neq \mathit{letraProhibida}$
Nos queda un problema de camino mínimo en ese grafo.
https://uva.onlinejudge.org/ (Problema 11492)

Camino mínimo de largo par

Problema:

Dado un grafo pesado y dos nodos distinguidos A y B. Se busca el menor peso que tiene un camino de A a B que utiliza una cantidad par de aristas.

Camino mínimo de largo par

Problema:

Dado un grafo pesado y dos nodos distinguidos A y B. Se busca el menor peso que tiene un camino de A a B que utiliza una cantidad par de aristas.

¿Cuáles son nuestros estados posibles?

Camino mínimo de largo par

Problema:

Dado un grafo pesado y dos nodos distinguidos A y B. Se busca el menor peso que tiene un camino de A a B que utiliza una cantidad par de aristas.

¿Cuáles son nuestros estados posibles? $(i, \textit{paridad})$

Camino mínimo de largo par

Problema:

Dado un grafo pesado y dos nodos distinguidos A y B. Se busca el menor peso que tiene un camino de A a B que utiliza una cantidad par de aristas.

¿Cuáles son nuestros estados posibles? $(i, \textit{paridad})$
¿Cuáles son nuestras transiciones posibles?

Camino mínimo de largo par

Problema:

Dado un grafo pesado y dos nodos distinguidos A y B. Se busca el menor peso que tiene un camino de A a B que utiliza una cantidad par de aristas.

¿Cuáles son nuestros estados posibles? $(i, \mathit{paridad})$
¿Cuáles son nuestras transiciones posibles?
$(i, \mathit{paridad}) \xrightarrow{w_{ij}} (j, 1 - \mathit{paridad})$

Camino mínimo de largo par

Problema:

Dado un grafo pesado y dos nodos distinguidos A y B. Se busca el menor peso que tiene un camino de A a B que utiliza una cantidad par de aristas.

¿Cuáles son nuestros estados posibles? $(i, paridad)$
¿Cuáles son nuestras transiciones posibles?
$(i, paridad) \xrightarrow{w_{ij}} (j, 1 - paridad)$
Comenzamos en (A, par) o $(A, 0)$ de cualquier forma que lo querramos representar.

Camino mínimo de largo par

Problema:

Dado un grafo pesado y dos nodos distinguidos A y B. Se busca el menor peso que tiene un camino de A a B que utiliza una cantidad par de aristas.

¿Cuáles son nuestros estados posibles? $(i, \mathit{paridad})$
¿Cuáles son nuestras transiciones posibles?
$(i, \mathit{paridad}) \xrightarrow{w_{ij}} (j, 1 - \mathit{paridad})$
Comenzamos en (A, par) o $(A, 0)$ de cualquier forma que lo querramos representar.
¿Cómo se modifica si queremos utilizar una cantidad de aristas que sea de la forma $3k + 1$ para algún k?

Camino mínimo de largo par

Problema:

Dado un grafo pesado y dos nodos distinguidos A y B. Se busca el menor peso que tiene un camino de A a B que utiliza una cantidad par de aristas.

¿Cuáles son nuestros estados posibles? $(i, paridad)$
¿Cuáles son nuestras transiciones posibles?
$(i, paridad) \overset{w_{ij}}{\rightarrow} (j, 1 - paridad)$
Comenzamos en (A, par) o $(A, 0)$ de cualquier forma que lo querramos representar.
¿Cómo se modifica si queremos utilizar una cantidad de aristas que sea de la forma $3k + 1$ para algún k? $(i, r_3(\#aristasUtilizadas))$

Escapando del Laberinto

Problema:

Estamos atrapados en un laberinto. Cada casilla está vacía, tiene una llave de un color o tiene un puerta de un color. Si pasamos sobre una llave, nos la quedamos y nos permitirá abrir toda puerta de ese color. ¿Podemos salir del laberinto?

Escapando del Laberinto

Problema:

Estamos atrapados en un laberinto. Cada casilla está vacía, tiene una llave de un color o tiene un puerta de un color. Si pasamos sobre una llave, nos la quedamos y nos permitirá abrir toda puerta de ese color. ¿Podemos salir del laberinto?

Usando lo que vimos hasta ahora, ¿qué estados posibles tenemos?

Escapando del Laberinto

Problema:

Estamos atrapados en un laberinto. Cada casilla está vacía, tiene una llave de un color o tiene un puerta de un color. Si pasamos sobre una llave, nos la quedamos y nos permitirá abrir toda puerta de ese color. ¿Podemos salir del laberinto?

Usando lo que vimos hasta ahora, ¿qué estados posibles tenemos? En todo momento deberíamos tener en nuestro estado la **posición** y el **subconjunto de llaves** que tenemos a disposición actualmente.

Escapando del Laberinto

Problema:

Estamos atrapados en un laberinto. Cada casilla está vacía, tiene una llave de un color o tiene un puerta de un color. Si pasamos sobre una llave, nos la quedamos y nos permitirá abrir toda puerta de ese color. ¿Podemos salir del laberinto?

Usando lo que vimos hasta ahora, ¿qué estados posibles tenemos? En todo momento deberíamos tener en nuestro estado la **posición** y el **subconjunto de llaves** que tenemos a disposición actualmente. $(i, j, mask)$

Escapando del Laberinto

Problema:

Estamos atrapados en un laberinto. Cada casilla está vacía, tiene una llave de un color o tiene un puerta de un color. Si pasamos sobre una llave, nos la quedamos y nos permitirá abrir toda puerta de ese color. ¿Podemos salir del laberinto?

Usando lo que vimos hasta ahora, ¿qué estados posibles tenemos?
En todo momento deberíamos tener en nuestro estado la **posición** y el **subconjunto de llaves** que tenemos a disposición actualmente. $(i, j, mask)$
Esta es una solución posible, falta recorrer el grafo aumentado y ver si llegamos a la salida con algún subconjunto de llaves.
Aumentamos el grafo de forma exponencial en la cantidad de llaves distintas.

Escapando del Laberinto

Problema:

Estamos atrapados en un laberinto. Cada casilla está vacía, tiene una llave de un color o tiene un puerta de un color. Si pasamos sobre una llave, nos la quedamos y nos permitirá abrir toda puerta de ese color. ¿Podemos salir del laberinto?

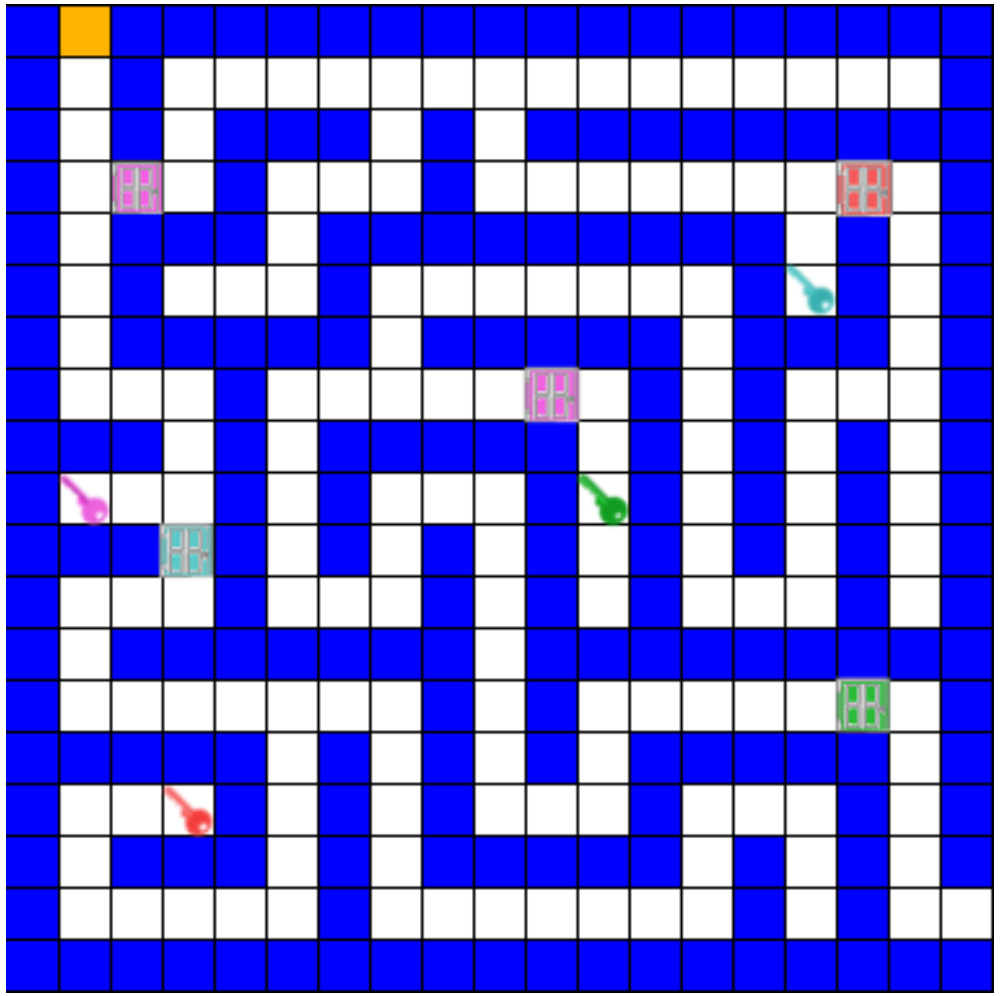
Usando lo que vimos hasta ahora, ¿qué estados posibles tenemos? En todo momento deberíamos tener en nuestro estado la **posición** y el **subconjunto de llaves** que tenemos a disposición actualmente. $(i, j, mask)$
Esta es una solución posible, falta recorrer el grafo aumentado y ver si llegamos a la salida con algún subconjunto de llaves.
Aumentamos el grafo de forma exponencial en la cantidad de llaves distintas. ¿Podemos hacer algo mejor?

Puede sernos útil nuevamente el **arreglo auxiliar de movimientos**:

- `dx = {1,-1,0,0}`
- `dy = {0,0,1,-1}`
- $\downarrow, \uparrow, \rightarrow, \leftarrow$
- Para cada k de 1 a 4:
  - `new_i = i + dx[k]`
  - `new_j = j + dy[k]`

Llaves: {}

Puertas: {}

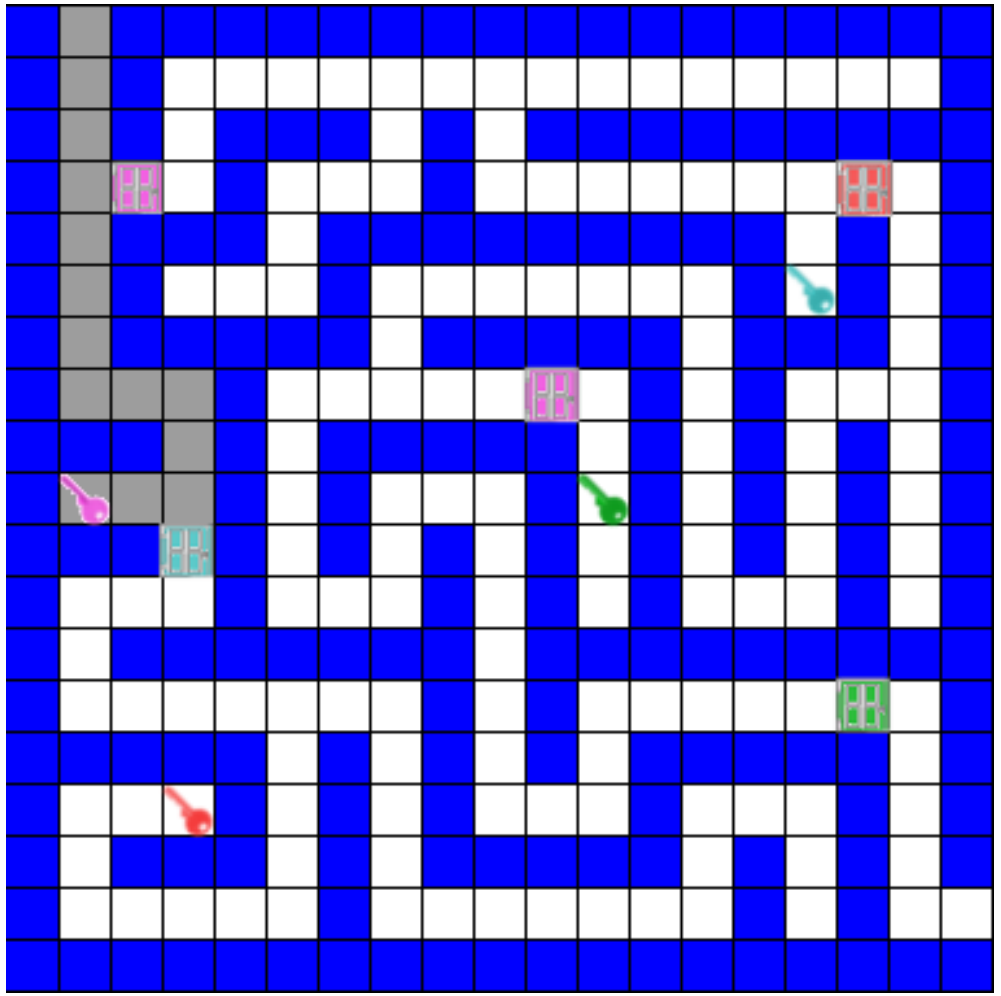


Puede sernos útil nuevamente el **arreglo auxiliar de movimientos**:

- `dx = {1,-1,0,0}`
- `dy = {0,0,1,-1}`
- $\downarrow, \uparrow, \rightarrow, \leftarrow$
- Para cada k de 1 a 4:
  - `new_i = i + dx[k]`
  - `new_j = j + dy[k]`

Llaves: {*Magenta*}

Puertas: {*Magenta*, *Celeste*}



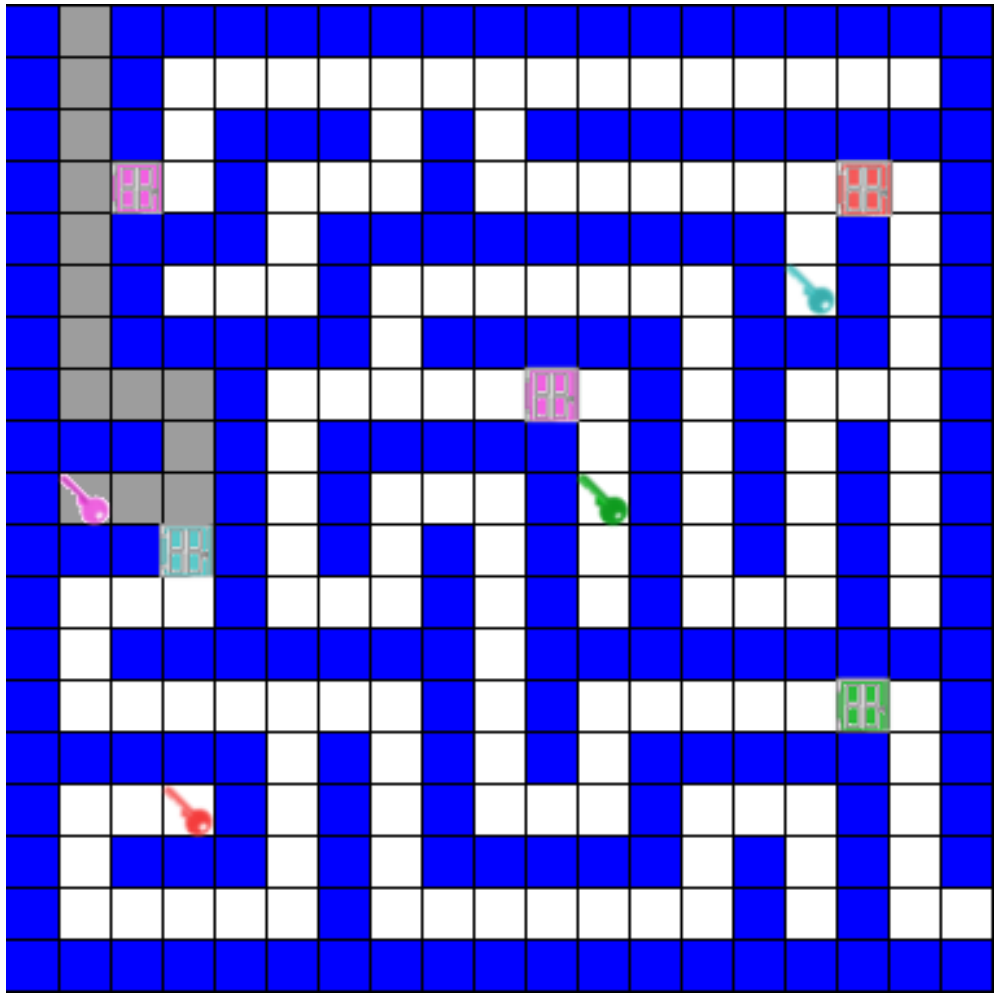
Puede sernos útil nuevamente el **arreglo auxiliar de movimientos**:

- `dx = {1,-1,0,0}`
- `dy = {0,0,1,-1}`
- $\downarrow, \uparrow, \rightarrow, \leftarrow$
- Para cada k de 1 a 4:
  - `new_i = i + dx[k]`
  - `new_j = j + dy[k]`

Llaves: {}

Puertas: {*Celeste*}

Las puertas color Magenta se abren

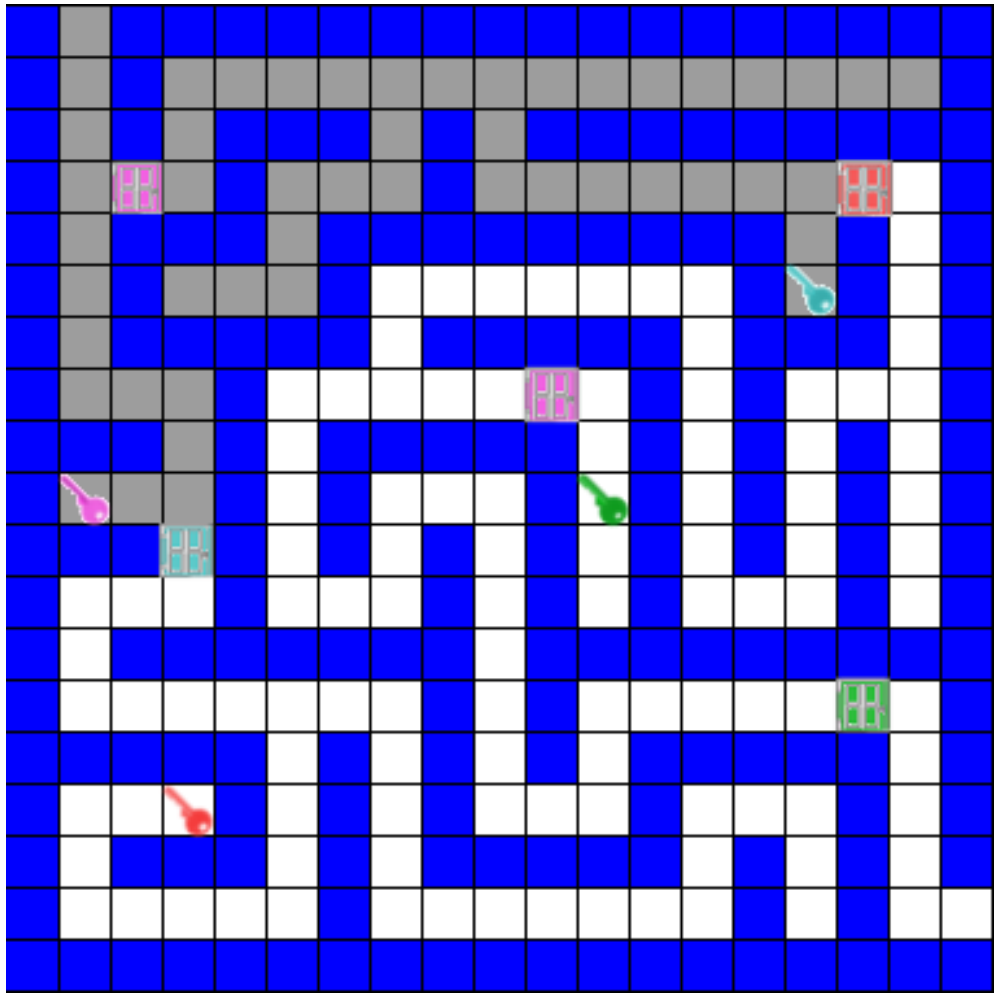


Puede sernos útil nuevamente el **arreglo auxiliar de movimientos**:

- `dx = {1,-1,0,0}`
- `dy = {0,0,1,-1}`
- $\downarrow, \uparrow, \rightarrow, \leftarrow$
- Para cada k de 1 a 4:
  - `new_i = i + dx[k]`
  - `new_j = j + dy[k]`

Llaves: {*Celeste*}

Puertas: {*Celeste*, *Rojo*}



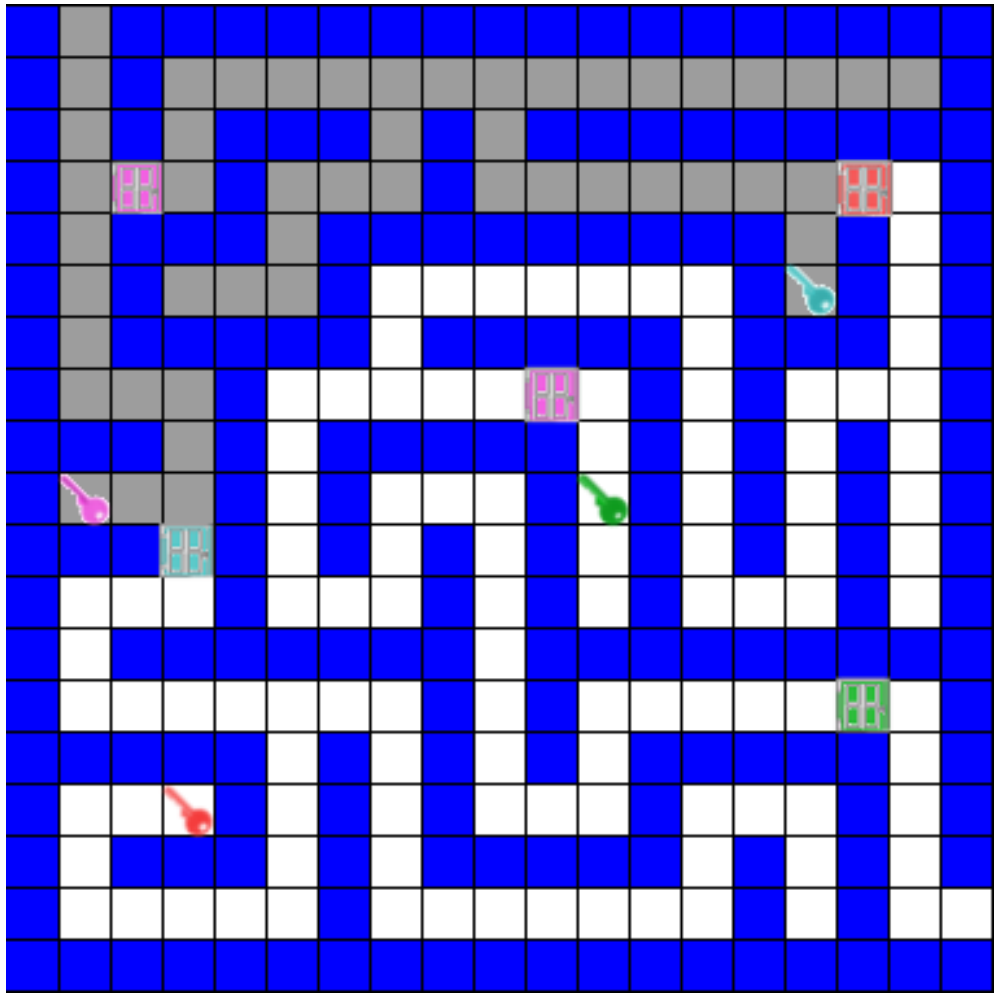
Puede sernos útil nuevamente el **arreglo auxiliar de movimientos**:

- `dx = {1,-1,0,0}`
- `dy = {0,0,1,-1}`
- $\downarrow, \uparrow, \rightarrow, \leftarrow$
- Para cada k de 1 a 4:
  - `new_i = i + dx[k]`
  - `new_j = j + dy[k]`

Llaves: {}

Puertas: {*Rojo*}

Las puertas color Celeste se abren

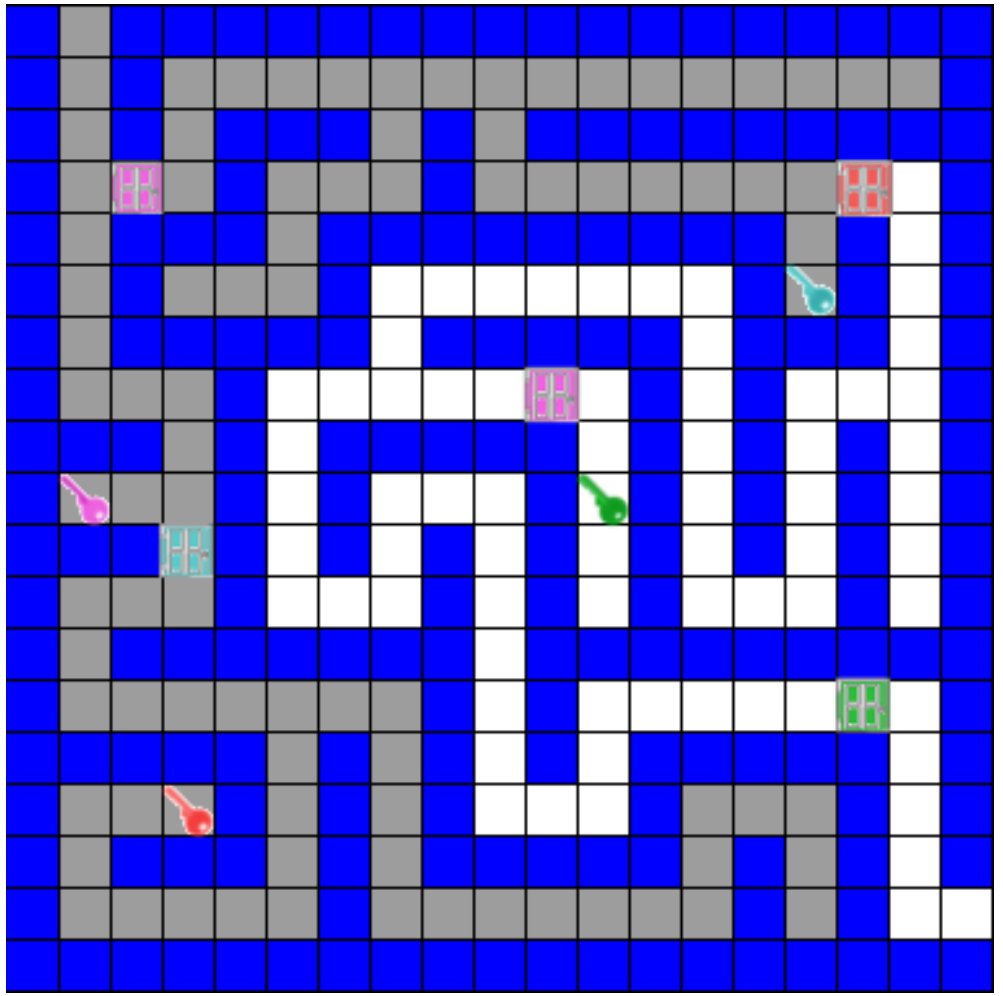


Puede sernos útil nuevamente el **arreglo auxiliar de movimientos**:

- `dx = {1,-1,0,0}`
- `dy = {0,0,1,-1}`
- $\downarrow, \uparrow, \rightarrow, \leftarrow$
- Para cada k de 1 a 4:
  - `new_i = i + dx[k]`
  - `new_j = j + dy[k]`

Llaves: $\{Rojo\}$

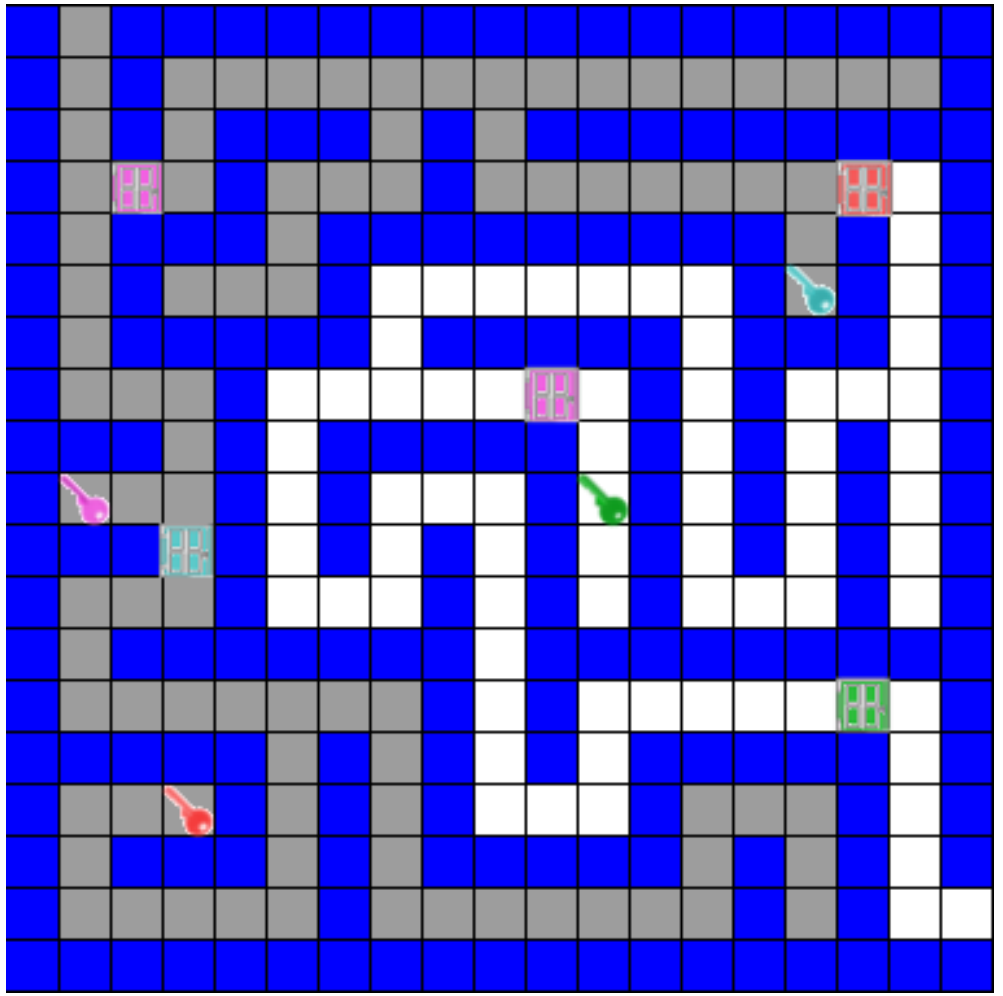
Puertas: $\{Rojo\}$



Puede sernos útil nuevamente el **arreglo auxiliar de movimientos**:

- `dx = {1,-1,0,0}`
- `dy = {0,0,1,-1}`
- $\downarrow, \uparrow, \rightarrow, \leftarrow$
- Para cada k de 1 a 4:
  - `new_i = i + dx[k]`
  - `new_j = j + dy[k]`

Llaves: {}
Puertas: {}
Las puertas color Rojo se abren

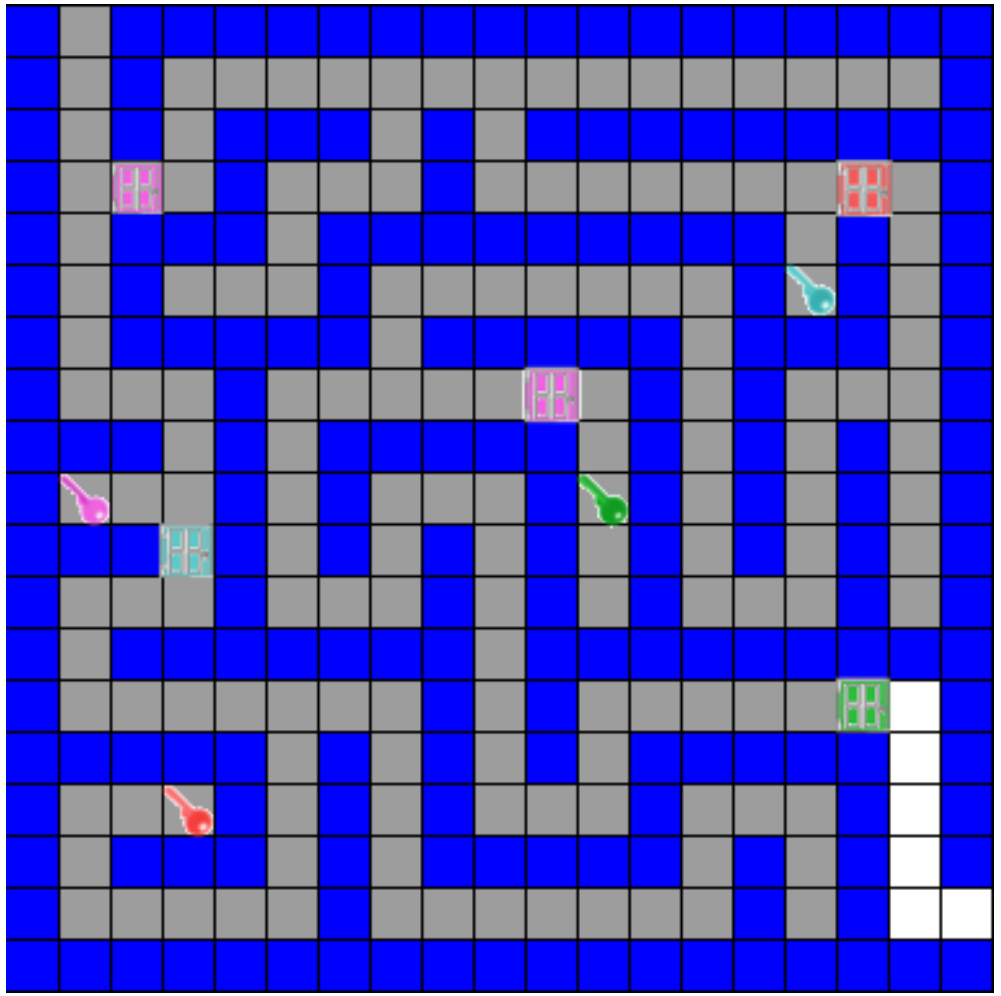


Puede sernos útil nuevamente el **arreglo auxiliar de movimientos**:

- `dx = {1,-1,0,0}`
- `dy = {0,0,1,-1}`
- $\downarrow, \uparrow, \rightarrow, \leftarrow$
- Para cada k de 1 a 4:
  - `new_i = i + dx[k]`
  - `new_j = j + dy[k]`

Llaves: $\{Verde\}$

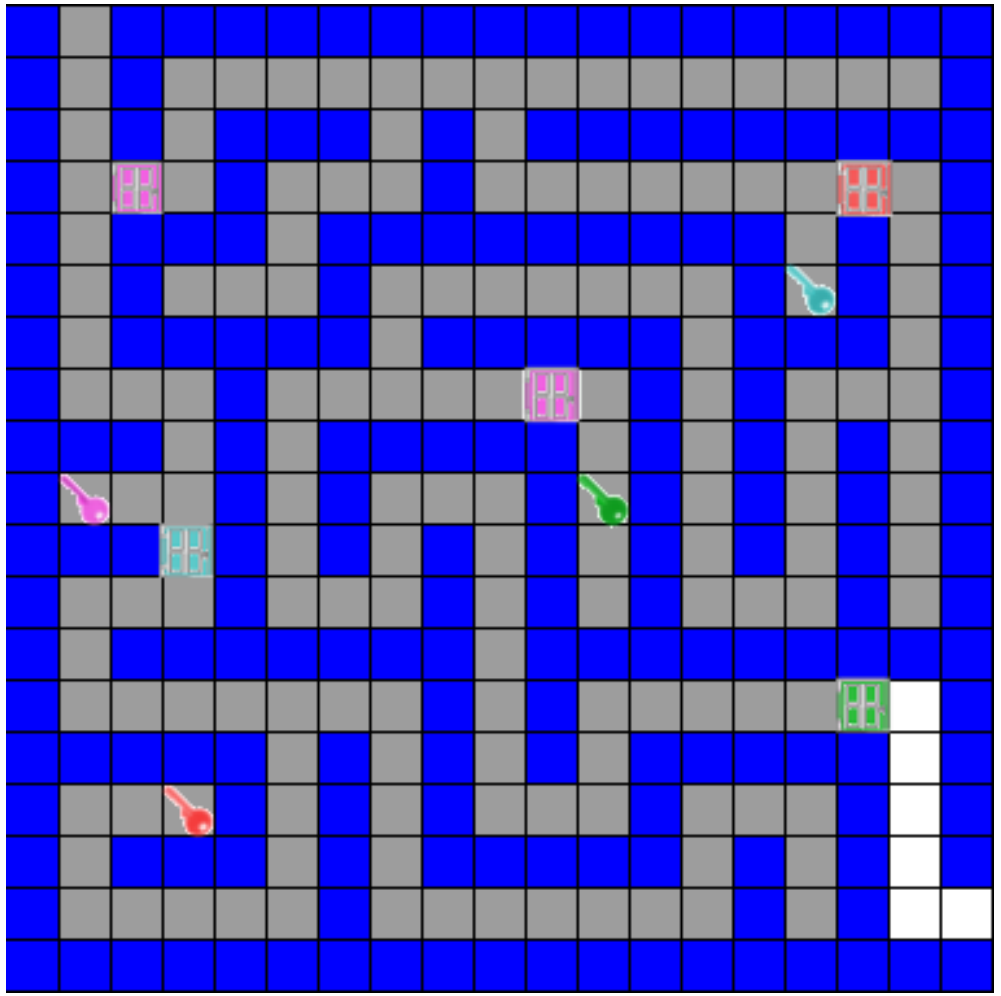
Puertas: $\{Verde\}$



Puede sernos útil nuevamente el **arreglo auxiliar de movimientos**:

- `dx = {1,-1,0,0}`
- `dy = {0,0,1,-1}`
- $\downarrow, \uparrow, \rightarrow, \leftarrow$
- Para cada k de 1 a 4:
  - `new_i = i + dx[k]`
  - `new_j = j + dy[k]`

Llaves: {}
Puertas: {}
Las puertas color Verde se abren

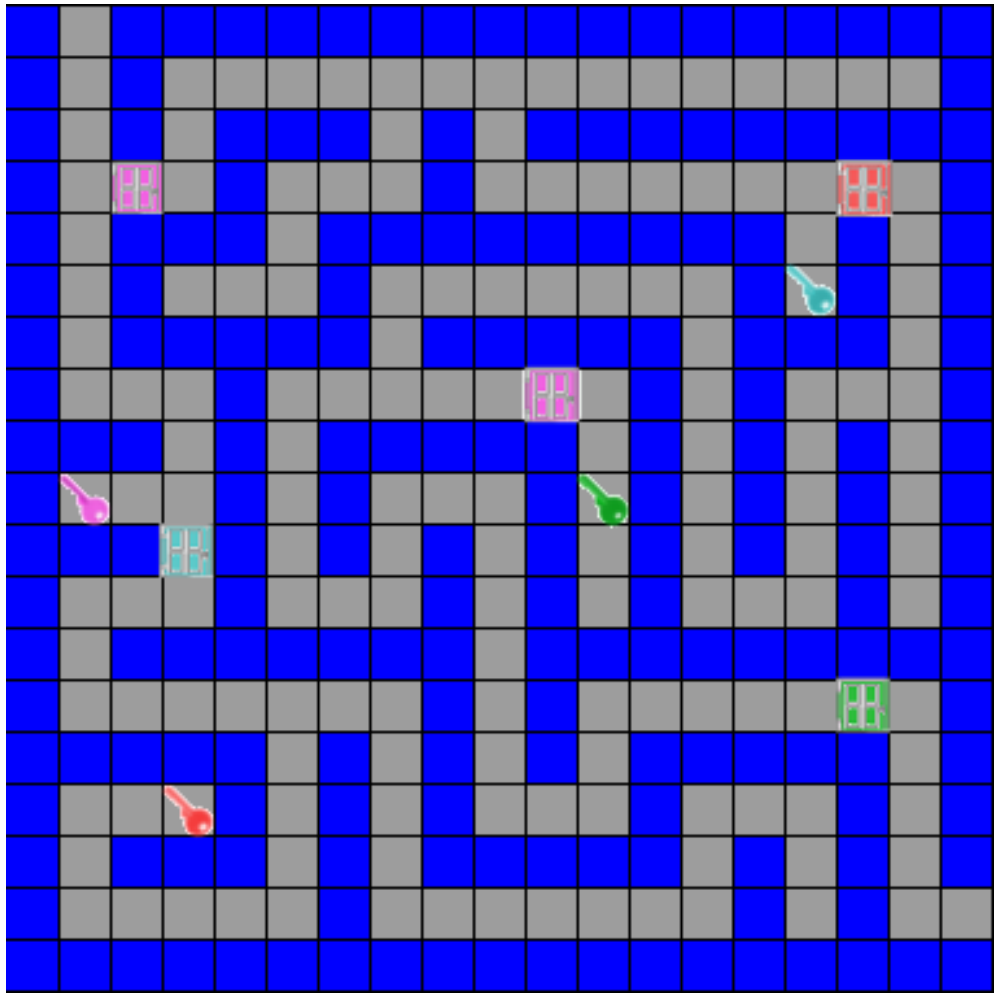


Puede sernos útil nuevamente el **arreglo auxiliar de movimientos**:

- `dx` = $\{1,-1,0,0\}$
- `dy` = $\{0,0,1,-1\}$
- $\downarrow, \uparrow, \rightarrow, \leftarrow$
- Para cada k de 1 a 4:
  - `new_i = i + dx[k]`
  - `new_j = j + dy[k]`

Llaves: $\{\}$

Puertas: $\{\}$



Problemas ordenados por dificultad (muy subjetivamente)

- "Tesoro" (castellano)
  http://juez.oia.unsam.edu.ar/#/task/tesoro/statement
- "Indiana" (castellano)
  http://juez.oia.unsam.edu.ar/#/task/indiana/statement
- "Escape From Jail Again" (inglés)
  https://www.spoj.com/problems/ESCJAILA/
- "Protesta" (difícil) (castellano)
  http://juez.oia.unsam.edu.ar/#/task/protesta/statement
- "Escape from Jail" (difícil) (inglés)
  https://www.spoj.com/problems/ESJAIL/

Problemas ordenados por dificultad (muy subjetivamente)

- "Tesoro" (castellano)
  http://juez.oia.unsam.edu.ar/#/task/tesoro/statement
- "Indiana" (castellano)
  http://juez.oia.unsam.edu.ar/#/task/indiana/statement
- "Escape From Jail Again" (inglés)
  https://www.spoj.com/problems/ESCJAILA/
- "Protesta" (difícil) (castellano)
  http://juez.oia.unsam.edu.ar/#/task/protesta/statement
- "Escape from Jail" (difícil) (inglés)
  https://www.spoj.com/problems/ESJAIL/

Siempre pueden hacer consultas sobre problemas (o cualquier tema relacionado con la Olimpíada) en el Foro de la OIA:

http://foro.oia.unsam.edu.ar

En particular, **en el foro estará esta charla subida a disposición de todos**.