Análisis y Diseño de Algoritmos

Tarea 4

Daniel Rojo Mata
danielrojomata@gmail.com

Fecha de Entrega: 15 de septiembre de 2024

Problema 1

En la algoritmo BFS visto en clase, cada vértice tiene tres atributos: color, distancia y padre. Muestra una variante del algoritmo que calcule la distancia de s a todos los demás vertices en G pero que cada vértice tenga como atributos adicionales únicamente un entero y un bit. La complejidad de tiempo de tu algoritmo debe ser $O(|V| + |E|)$.

Solución:

BFS utiliza las siguientes propiedades para los nodos:

- $u.color = white$
- $u.color = gray$
- $u.color = black$
- $u.d = \infty$
- $u.\pi = NIL$

Se utilizan tres colores (`white`, `gray` y `black`) para clasificar el estado de cada nodo durante el proceso de exploración.

1. Blanco (`white`): Un nodo coloreado como blanco es un nodo no visitado.
2. Gris (`gray`): Un nodo cambia de blanco a gris cuando ha sido descubierto, pero su exploración completa aún no ha terminado.
3. Negro (`black`): Un nodo se colorea de negro cuando ha sido completamente explorado. Esto significa que se han revisado todas sus aristas, y el nodo no será visitado de nuevo.

El algoritmo BFS es de la siguiente forma:

Algoritmo BFS

- Entrada: $G = (V, E)$ grafo, s vértice fuente

```
1     for each vertex u en G.V - {s}
2         u.color = white
3         u.d = infinito
4         u.padre = NIL
5     s.color = gray
6     s.d = 0
7     s.padre = NIL
8     Queue = Cola Vacia
9     ENQUEUE(Q,s)
10    while Q no vacia
11        u = DEQUEUE(Q)
12        for each vertex v in G.Adj[u]
13            if v.color == white
14                v.color = gray
15                v.d = u.d + 1
16                v.padre = u
17                ENQUEUE(Q,v)
18        u.color = black
```

Ahora, el algoritmo modificado usando únicamente un entero y un bit cambia el estado de los nodos, se hace lo siguiente:

- $u.color = white \iff u.color = 0$
- $u.color = gray \iff u.color = 1$
- $u.d = \infty \iff u.d = int$ (inicializado en ∞)
- $u.\pi = NIL$

La asignación que ahora se hace es cambiar los colores por el valores 0 o 1. La nueva designación es cambiar el color *white* por el bit 0, y el color *gray* por el bit 1 quitando el color *black*.

Mientras que la distancia ahora es representada por el valor entero simbolizado por la palabra *int*, sin embargo, se inicializa con ∞.

Así pues, el algoritmo modificado se muestra a continuación:

Algoritmo BFS Modificado

- **Entrada:** $G = (V, E)$ grafo, s vértice fuente

```
1     for each vertex u en G.V - {s}
2         u.color = 0      // white
3         u.d = infinito       // int
4         u.padre = NIL
5     s.color = 1      // gray
6     s.d = 0
7     s.padre = NIL
8     Queue = Vacia
9     ENQUEUE(Q,s)
10    while Q no vacia
11        u = DEQUEUE(Q)
12        for each vertex v in G.Adj[u]
13            if v.color == 0      // white
14                v.color = 1      // gray
15                v.d = u.d + 1
16                v.padre = u
17                ENQUEUE(Q,v)
```

Equivalencia de los algoritmos

El algoritmo `BFS Modificado` y el algoritmo `BFS` generan los mismos resultados en cuanto a la determinación de las distancias desde el nodo fuente s hacia los demás nodos en el grafo G.

1. **Uso de una cola:**
   `BFS` asegura que se exploran los nodos en capas, es decir, primero se exploran los nodos a distancia 1 de s, luego los de distancia 2, y así sucesivamente. `BFS Modificado` no cambia esta propiedad, ya que sigue utilizando una cola (FIFO) para procesar los vértices. De esta manera, la exploración de s hacia los vértices más lejanos se realiza en el mismo orden que en el BFS estándar.

2. **Bits por colores:**
   En el algoritmo original, los colores `white`, `gray`, y `black` tienen la función de:

   - `white`: Nodo no descubierto.
   - `gray`: Nodo descubierto, pero aún no completamente procesado.
   - `black`: Nodo completamente procesado.

   En `BFS Modificado`, los colores `white` y `gray` han sido sustituidos por los valores de bits 0 y 1 respectivamente. El color `black` se ha eliminado, pero esto no genera un problema, ya que solo es necesario un bit para distinguir entre nodos que aún no han sido descubiertos (0) y los que están en proceso de ser descubiertos (1).

El comportamiento del algoritmo sigue siendo equivalente al original, ya que el bit 1 cumple la misma función que el color `gray`, y no es necesario marcar los nodos como completamente procesados (`black`) dado que nunca se vuelven a encolar una vez procesados.

3. **Distancia iguales:**
   En ambos algoritmos, la distancia de un vértice v desde la fuente s se calcula de la misma manera. Cada vez que se descubre un nodo v adyacente a u, su distancia se actualiza como $v.d = u.d + 1$. Esta operación no ha cambiado en `BFS Modificado`, ya que usa un entero para representar $v.d$. Por lo tanto, las distancias calculadas por ambos algoritmos son exactamente las mismas.

4. **Padres de los nodos:**
   En ambos algoritmos cada nodo tiene un padre $u.\pi$, que se utiliza para almacenar el nodo desde el cual fue descubierto. Este atributo no cambia en `BFS Modificado`, lo que significa que ambos algoritmos preservan la información necesaria para reconstruir los caminos más cortos desde el nodo fuente s a los demás nodos.

5. **Complejidad temporal:**
   La complejidad del algoritmo `BFS` original es $O(|V| + |E|)$, donde $|V|$ es el número de nodos y $|E|$ es el número de aristas. Esto se debe a que:

   `BFS Modificado` no altera esta complejidad. Los nodos siguen siendo encolados y desencolados exactamente una vez, y todas las aristas adyacentes son exploradas una vez. El hecho de haber reducido el uso de colores a un bit y mantener las distancias como enteros no afecta el número de operaciones realizadas por el algoritmo. Por lo tanto, la complejidad sigue siendo $O(|V| + |E|)$.

Así pues, ambos algoritmos imprimen los mismos resultados porque mantienen el mismo enfoque de exploración de nodos, calculan las distancias de la misma manera, y preservan la estructura de los caminos más cortos usando los padres ($u.\pi$). La única diferencia es el uso de los colores por bits, que no afecta al funcionamiento del algoritmo.

Problema 2

Considera una grafica no dirigida y conexa G y un vértice s en ella. Supongamos que tanto $BFS(G, s)$ como $DFS(G)$ generan el mismo árbol T. Demuestra que $G = T$.

Demostración:

Sea G una gráfica conexa y no dirigida y s un vértice en $G.V$, tales que se cumple $BFS(G, s) = T = DFS(G)$, en donde lo anterior significa que $DFS(G)$ y $BFS(G, s)$ generan al árbol T.

Se quiere probar que $G = T$.

Por doble contención se tiene lo siguiente:

$\subseteq$ En este caso, se tiene que $T \subseteq G$, pues $T.E \subseteq G.E$, es decir, las aristas del árbol (que fueron procesadas por los algoritmos) necesariamente deben ser aristas de la gráfica y $T.V \subseteq G.V$ pues necesariamente los vértices del árbol fueron tomados de la gráfica.

$\supseteq$ Se usa el siguiente teorema:

`Teorema: En toda ejecución de BFS(G,s)`

1. `El algoritmo termina`
2. `Todo vértice` v `alcanzable desde` s `termina con` $v.d = \delta(s, v)$
3. $\forall\, v$ `alcanzable desde` s`, existe un camino más corto de` s `a` v `siguiendo las aristas de` $(u.\pi, u)$`,` $u \in G.V$

Nótese que gracias a que G es conexa y al teorema anterior, se puede decir que $G.V \subseteq T.V$; esto es, se visitan todos los vértices de la gráfica para construir el árbol.

Ahora, se prueba que $G.E \subseteq T.E$.

Por contradicción suponga que no es cierto lo anterior, esto quiere decir que existe una arista, llámese $(u, v) = e$, perteneciente a $G.E$ tal que $(u, v) \notin T.E$, pero u y v si son procesados por los algoritmos.

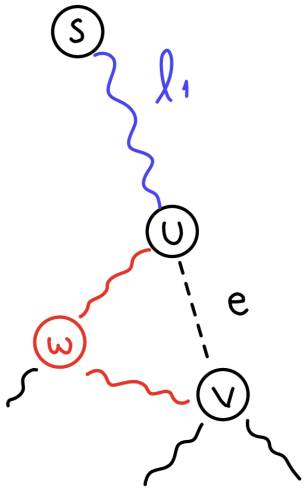

Figura 1: Existe al menos un vértice ω tal que DFS se "desvía" para llegar a v.

Puesto que T ha sido generado con ayuda de DFS, desde s se puede formar un camino hasta u, llámese ℓ_1 a este camino. Al llegar a u el algoritmo no siguió el camino (arista) $u - v$ puesto que se supuso que $(u, v) \notin T.E$, por lo que necesariamente, DFS tomó otra ruta.

Esto último se traduce a decir que existe al menos un nodo ω (o un camino pintado de rojo en 1) tal que DFS siguió para poder llegar a v. Lo anterior se ilustra en la imagen 1 como apoyo visual.

En palabras, el algoritmo siguió la ruta (l_1, u, ω, v).

Pero por otra parte, T también es generado por BFS, así que se cumple el siguiente lema.

`Lema: Toda arista conecta a vértices de capas/niveles adyacentes,`
`o en la misma capa/nivel.`

Así pues, al ejecutar BFS se tendría una situación como la que se ve en la figura 2.

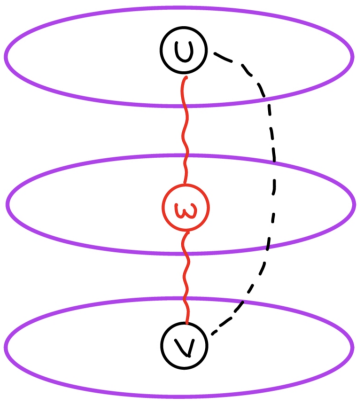

Figura 2: El acomodo de capas hecho con BFS no es consistente.

Entonces, BFS divide en capas a los vértices u, v, ω, sin embargo, por la construcción de DFS no es posible que u y v estén en capas adyacentes o en la misma capa, lo cual viola el Lema anterior, llegando a una contradicción, pues no se cumple que u y v estén en capas adyacentes. Esto quiere decir que los algoritmos procesan aristas distintas.

El error fue suponer que $(u, v) \notin T.E$, por lo que, por contradicción es cierto que si $(u, v) \in G.E$ entonces $(u, v) \in T.E$.

Finalmente, al cumplirse ambas contenciones se tiene que $G = T$.

□

Problema 3

Escribe un resumen de no más de una página del siguiente video: https://www.youtube.com/watch?v=HeQX2HjkcNo&t=1237s

Resumen:

A lo largo de la historia, se ha creído que las matemáticas tienen la capacidad de probar cualquier afirmación verdadera. Durante siglos, esta idea era indiscutible para muchos (incluyéndome). Sin embargo, hubo quienes no compartían esta creencia.

Con el avance de las matemática modernas, personas como Georg Cantor, Kurt Gödel y Alan Turing desafiaron esta visión que se había desarrollado con el paso de los años. Sus trabajos revelaron importantes limitaciones sobre lo que es posible probar y decidir dentro de las matemáticas.

Georg Cantor, en sus estudios sobre los números y el infinito, observó que no todos los infinitos son iguales. Demostró que hay más números reales en el intervalo $(0, 1)$ que en los números naturales $\mathbb{N}$. Este resultado ocasionó que los matemáticos se dividieran en dos grupos; los intuicionistas quienes creían que las matemáticas eran la creación pura de la mente humana y los formalistas, quienes creían que las matemáticas podían ser cimentadas sobre la teoría de los conjuntos trabajada y desarrollada por Cantor.

Dentro de los formalistas, destaca David Hilbert, quien trabajó en varias ramas de las matemáticas, además de que estaba convencido de que un sistema formal basado en la teoría de los conjuntos podía resolver todos los problemas abiertos de aquel siglo.

Es así que, a principios del siglo XX, Hilbert formuló tres preguntas que buscaban determinar los fundamentos de las matemáticas:

1. **¿Es la matemática completa?** ¿Hay alguna manera de probar toda afirmación verdadera?
2. **¿Son las matemáticas consistentes?** ¿Las matemáticas son libres de contradicciones?
3. **¿Es la matemática decidible?** ¿Existe un algoritmo que permita decidir si cualquier afirmación matemática es verdadera o falsa a partir de un conjunto de axiomas?

Hilbert creía firmemente que la respuesta a estas tres preguntas era afirmativa. Sin embargo, los descubrimientos de Gödel y Turing mostrarían que esta noción no era la correcta.

En 1931, Kurt Gödel demostró que la respuesta a la primera pregunta es "no". En su primer teorema de incompletitud, Gödel probó que cualquier sistema matemático que cuente con aritmética básica siempre tendrá afirmaciones dentro de él que son verdaderas pero que no tienen pruebas.

Además, su segundo teorema de incompletitud reveló que un sistema matemático no puede demostrar su propia consistencia. Es decir, si un sistema es consistente, esta propiedad no puede ser probada dentro de él mismo.

Poco después, Alan Turing respondió la tercera pregunta de Hilbert y demostró que la respuesta también era "no". Turing se preguntó si es posible decidir, para cualquier algoritmo, si este se detendrá o continuará ejecutándose indefinidamente.

Turing demostró que no existe un algoritmo general que pueda resolver este problema en todos los casos. Concluyó que no existe un algoritmo que pueda determinar la verdad o falsedad de todas las afirmaciones matemáticas.

Los trabajos de Gödel y Turing mostraron que siempre habrá afirmaciones verdaderas sobre las matemáticas que no podrán ser demostradas, y algunos problemas seguirán siendo indecidibles.

Como resultado, no es posible conocerlo todo con certeza: **simplemente, siempre habrá afirmaciones que no podremos probar ni decidir**.