Práctica 11

DFS

Matú Hernández Diana

Rojo Mata Daniel

- **algoritmoDFS**
- **Entrada:** Una gráfica `G` de orden `n` y tamaño `m` tal que `G.vertices = {v1, v2, . . . , vn}` y `vi.identificador = vi` y el identificador de un vértice `s` $\in$ `G.vertices`.
- **Salida:** La gráfica `G` tal que el valor del campo visitado de los vértices alcanzados desde el vértice `s` es `true`.

```
1  public void algoritmoDFS(String s) throws Exception {
2      for (Vertice coso : this.vertices) {
3          Vertice v = darVertice(coso.identificador);
4          v.modificarVisitado(false);
5          v.modificarPendiente(false);
6      }
7
8      Pila<String> pilaVisitados = new Pila<>();
9      pilaVisitados.apilar(s);
10     modificarPendienteVertice(s, true);
11
12     while (!pilaVisitados.esVacia()) {
13         String u = pilaVisitados.darElementoCima();
14         pilaVisitados.desapilar();
15         LinkedList<String> adj_u = darVecindad(u);
16
17         while (!adj_u.isEmpty()) {
```

```
18              String v = adj_u.getFirst();
19              adj_u.removeFirst();
20              Vertice w = darVertice(v);
21              if (w.darVisitado() == false && w.darPendiente() == false) {
22                  pilaVisitados.apilar(v);
23                  modificarPendienteVertice(v, true);
24              }
25          }
26          modificarVisitadoVertice(u, true);
27      }
28  }
```

Corrección del algoritmo

El algoritmo inicia con el ciclo `for 2-5`, quien es el encargado de recorrer todos los vértices de la gráfica para asegurarse de que todo vértice, v_i, cumpla que: v_i`.visitado = false` y v_i`.pendiente = false`. Esto con el afán de no tener problemas en la ejecución del algoritmo. Este ciclo se implementa de manera correcta, para demostrar esto, se proporciona un invariante de ciclo en forma de lema.

> *LEMA: Después de la iteración i, el campo* ***visitado*** *y* ***pendiente*** *de todos los vértices anteriores al vértice* v_i*, incluido éste, refleja la condición boolena* ***false****, esto es,* v_i`.visitado = false` *y* v_i`.pendiente = false` $\forall j \in \{1, ..., i\}$

- **Inicialización:** El ciclo `for` itera sobre todos los vértices de la gráfica, lo que significa que la inicialización se lleva a cabo para cada uno de ellos. Después de la primera iteración ($i = 1$), se asegura que los campos `visitado` y `pendiente` de v_1 sean establecidos en `false`, pues se ejecutan las líneas de código 4 en donde se cambia el estado de `visitado` a `false` y la línea 5, en donde se cambia el estado de `pendiente` a `false`. Este hecho garantiza que el invariante se cumpla después de la primera iteración del ciclo `for`, ya que la condición booleana requerida por el lema se satisface para el primer vértice.

- **Mantenimiento:** Supóngase que el vértice actual es v_i en la iteración i. Hasta este punto, la instrucción 4 se ha ejecutado en los vértices $v_1, v_2, ..., v_{i-1}$, estableciendo sus campos `visitado` y `pendiente` en `false`. Luego, en la iteración actual, la instrucción establecida en la línea 4 se ejecuta en v_i, asegurando que los campos anteriores también se establezcan en `false`. Por lo tanto, al final de la iteración i, todos los vértices anteriores o iguales a v_i (incluido v_i) cumplen el invariante, ya que todos tienen sus campos `visitado` y `pendiente` en `false`.

- **Finalización:** Al recorrer todos los vértices, se ha aplicado la instrucción 4 a todos éstos, garantizando que los campos mencionados de cada uno han sido cambiados a `false`. Por lo que el invariante se cumple.

Ahora, se procede a dar un invariante para el ciclo `while 10-25` en forma de lema.

LEMA: Antes de cada iteración del ciclo `while` *exterior (líneas 12-27) y después de cada iteración del ciclo* `while` *interior (líneas 17-25), la pila* `pilaVisitados` *contiene los vértices que deben ser procesados en el nivel actual del recorrido DFS. Además, los vértices en* `pilaVisitados` *han cambiado los campos* `visitado` *y* `pendiente` *a* `true`*.*

- **Inicialización:** Antes de iniciar el ciclo exterior, se crea la pila `pilaVisitados` y es hasta la línea 9 en donde se apila el elemento `s` a ésta. Esto cumple con el hecho de que la pila mencionada contiene los vértices que deben ser procesados para el recorrido DFS, a saber, solo contiene a `s`. Se ejecutan los ciclos while solo con este elemento, y en particular, en la línea 23 y 26 se cambian los campos `pendiente` y `visitado` a `true`, cumpliéndose así el invariante.

- **Mantenimiento:** Durante cada iteración del bucle interior, se considera un vértice `u` que ha sido desapilado de `pilaVisitados`. A continuación, se exploran todos los vecinos no visitados de `u` y se realizan las siguientes acciones:

  - Para cada vecino `v` de `u` que aún no ha sido visitado, se apila en `pilaVisitados` y se marca como visitado. Esto se logra con la condición `if` presente en las líneas 21-23. Este paso garantiza que todos los vecinos no visitados de `u` (sin ser repetidos) se agregarán a la pila `pilaVisitados` y se marcarán como `true` en el campo pendiente.

- Después de este bucle interior, `pilaVisitados` contiene los vértices que deben ser procesados en el próximo nivel del recorrido DFS. Además, los vecinos de `u` se han agregado a la pila y se ha cambiado la marca de pendiente a `true` además de haber cambiado el campo visitado (línea 26). Finalmente, el marcado en ambos campos de los vértices garantiza el cumplimiento del invariante.

- **Finalización:** La condición de terminación establece que cuando la pila `pilaVisitados` está vacía, todos los vértices alcanzables desde el vértice inicial `s` han sido visitados. La pila `pilaVisitados` contiene los vértices que deben ser procesados en el nivel actual del recorrido, y cuando esta pila se vacía, significa que no hay más vértices por procesar en el nivel actual. Dado que se han marcado como visitados y ya no están pendientes todos los vértices alcanzables desde el vértice inicial hasta el nivel actual, se puede concluir que el algoritmo ha explorado todos los vértices alcanzables en toda la gráfica.
  Esta propiedad además asegura la finalización del algoritmo de manera correcta después de haber visitado todos los vértices de la gráfica, esto es, el invariante se cumple.

Por lo argumentado anteriormente, se concluye que el algoritmo dado es correcto.