Tarea de reposición

Edgar de Jesús Vázquez Silva

3 Enero 2019

Análisis y diseño de algoritmos 2019-1

Prof. Dr. Armando Castañeda
Prof. Dr. David Flores
Aydte. Manuel Alcántara Juárez

Ejercicio 4

1. **Considera el siguiente algoritmo ineficiente que decide si existe un camino entre dos vértices s y t de una gráfica dirigida G. Demuestra que el algoritmo es correcto. Además, analiza su complejidad y compárala con la de los algoritmos que vimos en clase, y explica por que efectivamente este algoritmo es ineficiente.**

```
ALGORITHM Alcanzable(G, s, t)
  RETURN A(G,s,t,|V(G)|)
END Alcanzable

FUNCTION A(G,s,t,d)
  IF d=1 THEN
    IF s == t OR existe una arista dirigida (s,t) en G THEN
      RETURN TRUE
    ELSE
      RETURN FALSE
  ELSE

  FOR cada vértice v en G DO
    IF A(G,s,v,d/2) AND A(G,v,t,d/2) THEN
      RETURN TRUE
    ENDIF
  ENDFOR
  RETURN FALSE
END A
```

Se dará una breve descripción del funcionamiento del algoritmo, a fin de hacer más clara la demostración de corrección del mismo. El algoritmo hace uso de una función recursiva que hace dos llamadas de tamaño la mitad de la instancia original sobre el número de vértices del grafo dirigido G, sea $d = |V(G)|$. Se aprecia que el caso base sucede cuando ($d \rightarrow$ número de vértice) es igual a 1, en tal caso suceden tres opciones: 1) el origen s y el destino t sean el mismo nodo, 2) exista una arista que conecte el vértice origen s con el vértice destino t; en tal caso se retorna un *true*. 3) Que no suceda ninguna de las opciones anteriores en cuyo caso se retorna un valor *false*. De este modo en el caso base se abordan todas las opciones posibles con una instancia de tamaño $d = 1$, con d igual al número de vértices en el grafo G.

Por otro lado, la generación de llamadas recursivas se hace con dos instancias de tamaño $d/2$, donde se verifica que exista una unión de caminos desde el nodo raíz hasta un nodo v_n y a su vez exista un camino desde el mismo nodo v_n hasta el nodo destino t. Si además la generación de subinstancias del problema se realiza de forma exahustiva en el número de vértices en el grafo G (línea 13 del algoritmo), se asegura que si existe un camino entre los nodos $s \rightarrow t$, necesariamente serán encontrados por el algoritmo, como se demostrará en el apartado subsecuente.

Con un panorama más claro acerca del funcionamiento del algoritmo se procede a demostrar su corrección, posteriormente su complejidad y por ultimo se realizará una comparación con los algoritmos vistos en clase para resolver el mismo problema.

Corrección.

Por demostrar, el algoritmo considera todos los casos posibles de conexiones en el caso base.

Definición 4.1, Vértice: Sea un grafo dirigido $G = (V, E)$ donde: $V \neq \emptyset$ y $E \subseteq \{(a, b) \in V \times V : a \neq b\}$ (a, b son un conjunto de pares ordenados de elementos de V)

Lema 4.1 El caso base considera todas las posibles opciones cuando solo existe un vértice. De esta manera el caso base es correcto. (por demostrar).

Prueba. En el caso base, la llamada recursiva se realiza con dos nodos s (nodo origen), t (nodo destino), donde $s, t \in V$. Por lo tanto pueden suceder solo cuatro casos. 1) $\{a, b\}$, están conectados y además $a = s$ y $b = t$. En cuyo caso, existe un camino entre los nodos s, t formado unicamente por la arista e_i, esta condición es evaluada por el algoritmo y retorna *true* como es deseado. 2) $\{a, b\}$, están conectados y además $a \neq s$ o $b \neq t$, en tal caso no existe un camino entre el origen s y el destino b, el algoritmo considera este caso y retorna un valor de *false.* 3) $a = b$, por la definición 1, no existe arista que conecte tales nodos. Y se cumple por vacuidad. El algoritmo considera este caso y retorna un valor *true* 4 $\{a, b\}$, no están conectados. El algoritmo considera este caso y devuelve un valor *false*, ya que no existe camino entre los nodos.

Por la definición 1, necesariamente toda pareja de nodos $\{a, b\}$ debe caer en alguno de estos casos. En caso de una eventualidad o un caso no considerado dentro de las opciones anteriormente enumeradas, se retorna un valor *false*, lo cual es correcto para nuestro algoritmo, puesto que ya se consideran todas los posibles eventos en los cuales **existe un camino entre los nodos** $\{a, b\}$. Por lo tanto podemos decir que el algoritmo es correcto en el caso base. ∎

Lema 4.2 Si existe un camino entre los nodos s, t necesariamente está formado con el conjunto de aristas $A = \{\ E_1 = \{a, v_1\}, E_2 = \{v_1, v_2\}, E_3 = \{v_2, v_3\}...E_n = \{v_{n-1}, t\}\ \}$. Donde, el conjunto A tiene al menos un elemento.

Prueba. En este sentido, la ejecución de cada subinstancia del problema, explora exhaustivamente todos los nodos v_i del grafo para un nodo padre s y verifica que exista una arista e_j que los conecte, posteriormente verifica que exista una arista e_k que conecte el mismo nodo v_i con el destino t. Visto de otra manera el algoritmo evalua que exista la unión de $s \rightarrow v_i \cup v_i \rightarrow t$, lo cual es la definición que el camino que intenta encontrar el algoritmo, en el caso que exista ese camino el algoritmo devuelve *true*.

Por el lema 4.1, sabemos que en el caso base el algoritmo es correcto (cuando existe una arista que conecta el nodo s con el nodo t). Además, el agoritmo explora la conexión de un nod fuente s con todos los nodos existentes en la grafica (línea 13), eso por un lado. En adición, el algoritmo explora todos los nodos existentes en el grafo G en busca de una conexión entre los nodos $\{v_i, v_{i+1}, v_{i+2}..., v_n\}$ con el destino origen t.

Por contradicción, existe un camino solución entre los nodos $s \to t$ que contiene las aristas $A = \{ E_1 = \{a, v_1\}, E_2 = \{v_1, v_2\}, E_3 = \{v_2, v_3\}...E_n = \{v_{n-1}, t\} \}$, que no fue encontrado por el algoritmo. Para la primer llamada del algoritmo, se exploró la conexión del nodo origen s con todos los nodos existentes en el grafo, se exploraron todas las posibles conexiones con el nodo fuente t, y el algoritmo toma como solución la unión de ambas soluciones de las respectivas subintancias. Cada una de estas llamadas, genera a su vez todas las subinstancias posibles tomando ahora como los nodos origenes a los nodos destino de la subinstancia anterior.

Si existe un camino solución $A = \{ E_1 = \{a, v_n\}, E_n = \{v_n - 1, t\} \}$, que no fue encontrado por el algoritmo, donde v_n es el cunjunto de todos los nodos que pertenecen a G y que ademas conectan a $s \to t$, quiere decir que el algoritmo aún no ha explorado todos los nodos existentes en el grafo G, para cualquier llamada a una subinstancia de tamaño menor que $|V(G)|$, por lo tanto el ciclo *for* nunca terminaría, si el ciclo for nunca termina. Si se considera que el número de vértices en el grafo G es finito, esto es un absurdo. Lo cual contradice a la naturaleza del problema. ∎

Por el lema 4.1 el algoritmo es correcto en el caso base, por el lema 4.2 el algoritmo explora exhaustivamente todos los nodos existentes en el grafo G, y encuentra la solución si es que existe, por lo tanto el algoritmo es correcto.

Complejidad. Para el analisis de complejidad del algoritmo recursivo tenemos que cada llamada de una subinstancia, produce dos subinstancias de tamaño $d/2$, donde d es el número de nodos existentes en el grafo.

Prueba. ∎