Juan Cisneros Resendiz
Algoritmos en gráficas 2021
Tarea 2
Fecha de entrega: 6 de febrero

El objetivo de esta tarea es que te familiarices con DFS, que entiendas claramente las categorías de aristas generadas por el algoritmo y qué información respecto a la gráfica codifican, que entiendas qué es un recorrido en orden y qué es un recorrido postorden. Además que puedas convertir un problema escrito en términos no matemáticos a un problema escrito en términos no matemáticos y resolverlo con lo que hemos aprendido esta semana.

1. Tienes una colección de n cajas y m llaves. Cada llave abre exactamente una caja, sin embargo cada caja puede abrirse con una llave, más de una llave o ninguna llave si, digamos, la rompes con un martillo. Supongamos que alguien colocó las m llaves en el interior de las n cajas y después cerró cada caja, afortunadamente hizo una nota avisándote qué llave está dentro de qué caja. Podría haber cajas sin llaves adentro, o podría haber más de una llave adentro. Deseas recuperar todas las llaves, para hacer esto claramente tienes que romper al menos una caja.

a. Describe y analiza un algoritmo para determinar si es posible recuperar todas las llaves rompiendo únicamente una caja.

b. Describe y analiza un algoritmo para calcular el menor número de cajas que deben romperse para recuperar todas las llaves.

Respuesta:

a. La idea del algoritmo es la siguiente.

Para simplificar la exposición del problema, supongamos que recibimos como entrada una gráfica G en donde tenemos como vértices a las cajas y a las llaves, y hay una arista $c \to k$ si y solo sí la caja c contiene a la llave k, y hay una arista $k \to c$ si y solo sí llave k abre la caja c. Desde aquí solo trabajaremos con esta gráfica por el resto del algoritmo. Nótese que no hay ningún vértice de caja a caja, ni de llave a llave.

En el algoritmo primero calculamos la gráfica de componentes fuertemente conexas $scc(G)$, y vemos cuantos vértices fuente tiene, ese es el número de cajas mínimo que tenemos que romper. Demostremos esto último

Teorema El número de cajas mínimo que se tienen que romper es el número de fuentes en la gráfica $scc(G)$.

Sea f el número de fuentes de $scc(G)$, F el conjunto de las fuentes de $scc(G)$ y sea n el número de cajas mínimo. veamos que $f = n$

- $f \leq n$: Como $indeg(k) \geq 1$ para toda llave k, no hay ningún vértice fuente en G que sea llave, eso implica que todas las fuentes de $scc(G)$ representan a componentes que contienen al menos una caja, así que hay que romper una caja por cada componente que es fuente para abrir las cajas de esa componente, por lo tanto se necesitan al menos romper f cajas para abrir todas.
- $n \leq f$ Ya que todo vértice en una DAG tiene que ser alcanzable por al menos una fuente, el conjunto F alcanza a todos los vértices de $scc(G)$, eso implica que podemos tomar por cada fuente f_i un vértice v_i que viva en la componente conexa que f_i representa, y con esos vértices podemos alcanzar a todo G.

■

MinCajas(G):
    return the number of source vertices in $scc(G)$

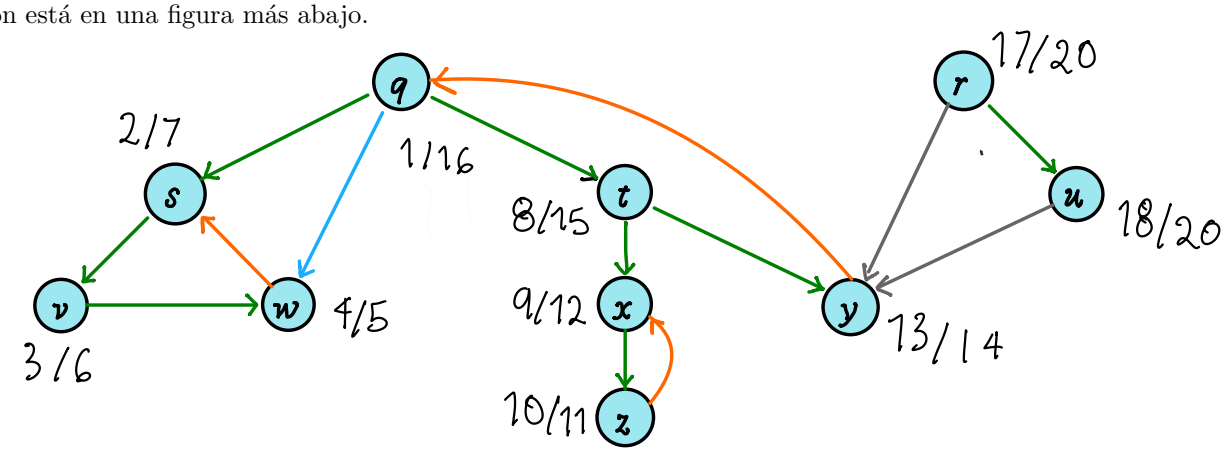
Podemos calcular la gráfica $scc(G)$ en tiempo $O(V+E)$ con el algoritmo de Kosaraju, y podemos contar el número de vértices fuente de $scc(G)$ también en tiempo $O(V+E)$, así que el obtener el número mínimo de cajas que se necesitan romper se puede realizar en tiempo $O(V+E)$

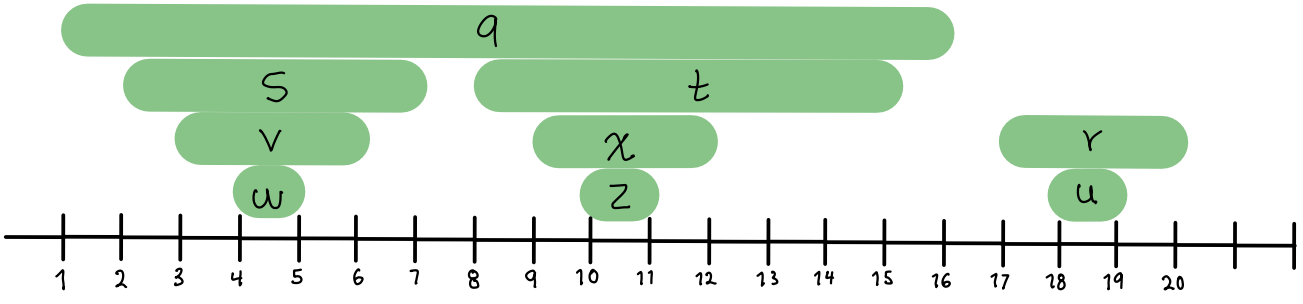
b. Está resuelto en el inciso a

■

2. Muestra cómo funciona DFS en la gráfica de la Figura 1. Supón que el algoritmo considera los vértices en orden alfabético, y supón que cada lista de adyacencia se encuentra ordenada también en orden alfabético. Para cada vértice v, muestra el valor de $v.pre$ y $v.post$, muestra también la clasificación de cada arista de la gráfica y gráfica los segmentos $[v.pre, v.post]$.

Respuesta:
La solución está en una figura más abajo.





■

3. Sea G una gráfica dirigida acíclica con una única fuente s y un único pozo t, como la que se muestra en la Figura 2. Un vértice $v \notin \{s,t\}$ es llamado un vértice de $(s,t)-$corte si cada camino de s a t pasa a través de v, o equivalentemente, si al borrar a v de la gráfica t deja de ser alcanzable desde s. La gráfica en la Figura 2 tiene tres vértices de $(s,t)-$corte. Describe y analiza un algoritmo para encontrar cada vértice de $(s,t)-$corte en G.

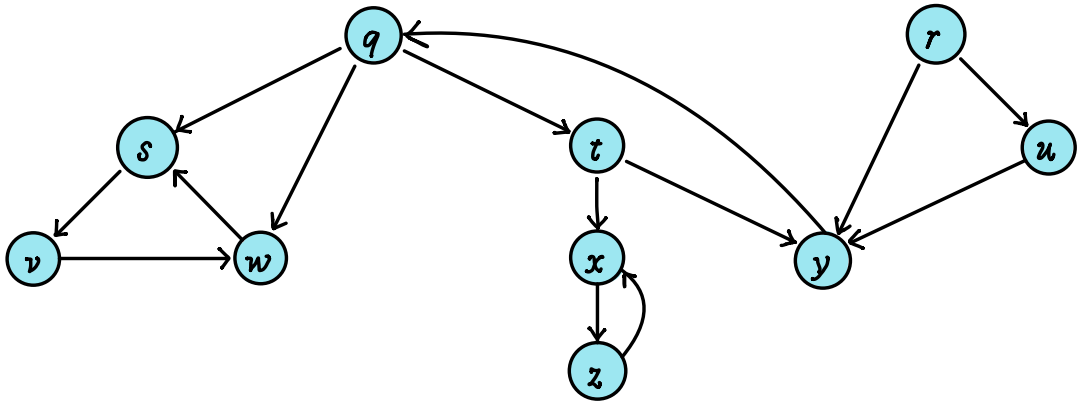


Figura 1: Ejecutar DFS en esta gráfica

Respuesta:
Para saber si un vértice v es de corte podemos remover ese vértice de la gráfica y a todas las aristas que inciden a v en G, y hacemos un recorrido DFS comenzando desde s sobre la gráfica sin v, como s es la única fuente de la DAG entonces s tiene que alcanzar a todos los vértices para que sea una gráfica conexa, si hay vértices no marcados después del recorrido entonces v es de corte. Para remover el v podemos simplemente marcar a v antes de realizar el recorrido, eso hace que el recorrido DFS no lo tome en cuenta.

```
VerticeCorte(v):
  for all vertices u
    unmark u
  mark v
  DFS(v)
  for all vertices u
    if u is unmarked
      return false
  return true
```

Para encontrar todos los vértices de corte simplemente ejecutamos el algoritmo VerticeCorte sobre todos los vértices de la gráfica.

```
VerticesDeCorte(G):
  C ← una lista vacía
  for all vertices u
    if VerticeCorte(u)
      //Agrega ese vértice al inicio de la lista C
      C ← u : C
  return C
```

■

La complejidad del algoritmo VerticesDeCorte es la siguiente: cada llamada a VerticeCorte(v) toma tiempo $O(V + E)$, y hacemos V llamadas, y el agregar un vértice al inicio de la lista C toma tiempo $O(1)$, así que el algoritmo tiene complejidad $O(V(V + E))$.

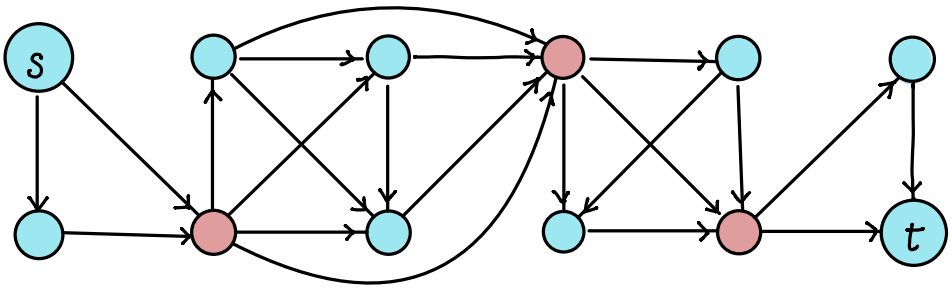


Figura 2: Una DAG con una fuente y un pozo, los vértices de $(s,t)-$corte se muestran en rojo.