Apuntes de Teórico

PROGRAMACIÓN 3

Grafos

Versión 1.5

Contenido

Grafos

Definición Informal

Es una colección de entidades (denominadas nodos o vértices) y enlaces (denominados aristas o arcos) que conectan un subconjunto de aquellas.
Se los diagrama como un conjunto de puntos/círculos conectados por líneas.

Aplicaciones

- Sistemas de información geográfica
- Representación estructural de vínculos en páginas web.
- Representación estructural y vínculos: Química, Física, Biología, Sociología, Medicina.
- Representación estructural y flujo de redes (mediante etiquetado): telecomunicaciones, computadoras, transporte, servicios públicos.
- Representación estructural y dinámica de transición en sistemas.

Rol en informática

Desarrollo de estructuras de datos y algoritmos eficientes.

Rol en el curso

Usos específico y básico para la resolución de problemas

Definición Formal

Grafo:

Colección $G = (V, A)$ con $A \subseteq V \times V$
V: conjunto de vértices (nodos), notados v, w
A: conjunto de aristas (arcos), notados (v, w) con $v, w \in V$

Adyacencia:

El vértice w es adyacente a vértice v sii $(v, w) \in A$.

Grafo Simple:

Grafo en que a lo sumo una arista conecta todo par de vértices.

Grafo dirigido:

Grafo en el que las aristas son dirigidas. Para cada par de vértices las dos direcciones (v, w) y (w, v) son distinguibles.

Grafo etiquetado:

Grafo en el que los vértices y/o aristas tienen asignados valores.

Representación: (estructura de datos)

Dado el grafo dirigido $G = (V, A)$

Matriz de Adyacencia:

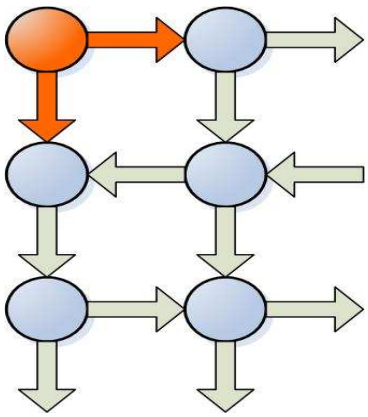
a: $V \times V \rightarrow$ Bool

donde

$$a[v,w] = \begin{cases} 1, & si\ (v,w) \in A \\ 0, & o/c. \end{cases}$$

Espacio de almacenamiento $O(|V|^2)$

Ventajas: simplicidad, facilidad de búsqueda de subgrafos.
Desventaja: baja densidad de almacenamiento para grafos dispersos.
Ej: Amanzanamiento. Manhatan.

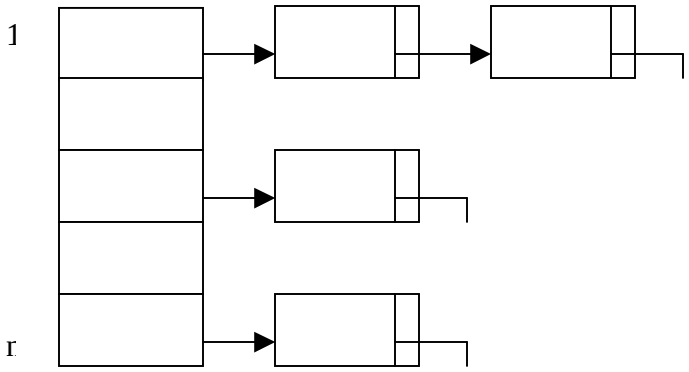


1 nodo – 2 aristas.

$\rho = \frac{2n}{n^2} = \frac{2}{n}$ (Densidad de almacenamiento)

Lista de adyacencia

Para cada nodo se tiene una lista de sus adyacentes.
Espacio: O (|V| + |A|)



Ventajas: correlación entre espacio usado y disponible, facilidad de búsquedas de adyacencia.
Desventaja: determinar la existencia de un arista (*v*, *w*); en grafos no dirigidos mantener la redundancia [(*v*, *w*) , (*w*, *v*)]

Notas:

a) La *identificación de vértices* implica un mapeo de nombres con la estructura de representación interna.

b) La *representación de etiquetas* (valores asociados) puede ser conjunta ó referenciada.

TAD Grafo

Crear Grafo: Ø → Grafo
Agregar Vértice: Grafo x Vértice → Grafo
Agregar Arista: Grafo x Arista → Grafo
Quitar Vértice: Grafo x Vértice → Grafo
Quitar Arista: Grafo x Arista → Grafo
Adyacentes: Grafo x Vértice → Conjunto de Vértices
Es Vacío: Grafo → Bool

Notar que *V* y *A* son conjuntos, cualquier representación de conjuntos puede servir para su representación.

Definiciones

SubGrafos

Dado el grafo $G = (V, A)$

1. Subgrafo ***inducido*** por vértices
   Subgrafo $G' = (V', A')$ con $V' \subseteq V$, con $A' = \{(v, w) \in A \mid v \text{ y } w \in V'\}$.

2. Subgrafo ***recubridor***
   Subgrafo $G' = (V, A')$ con $A' \subseteq A$.

Abreviaturas de operaciones

Remoción de un vértice del grafo G: $G - \{v\}$
Remoción de una arista del grafo G: $G - \{(v, w)\}$

Camino:

Secuencia de vértices $(v_1, \ldots, v_p)$ tal que $(v_i, v_{i+1}) \in A$ con $i = 1,\ldots, p\text{-}1$.

Longitud de Camino:

Cantidad de aristas que forman el camino (por definición hay camino de un vértice a sí mismo y se considera de largo cero).

Ciclo:

Un camino donde el primer y último nodo coinciden: $v_1 = v_p$

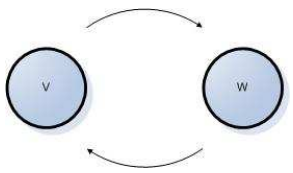
Camino Simple:

Todos los vértices del camino son distintos

Ciclo Simple:

Si el camino contenido es simple. Una arista (v,v) es un ciclo o lazo; tiene longitud de camino uno.
En **grafos dirigidos** el sentido cuenta:

El camino (v, w, v) es un ciclo si $\begin{cases} (v, w) \in A \\ y \\ (w, v) \in A \end{cases}$



En **grafos no dirigidos:** la arista (v, w) es indistinguible de (w, v); por lo que no hay ciclo en la secuencia de vértices (v, w, v)

Grafo acíclico:

Si no contiene ciclos.
Ejemplos: Representación de previaturas de las asignaturas de una carrera, camino crítico, problemas de redes de flujos.

Conectividad:

- Un grafo no dirigido es ***conexo*** si hay un camino entre todo par de vértices.
- Un grafo dirigido es ***fuertemente conexo*** si hay camino (dirigido) entre todo par de vértices.

Grafo completo:

Si hay arista entre todo par de vértices.

Punto de articulación:

Sea el grafo G conexo, si existe $v \in V$ tal que $G - \{v\}$ es desconexo entonces v se denomina ***punto de articulación***.

Observación: si G no tiene puntos de articulación entonces es ***biconexo***

Árbol:

Es un grafo acíclico, conexo y no dirigido.
Equivalente: grafo no dirigido en el cual existe un solo camino entre todo par de vértices.
Se denomina ***bosque*** a una colección de árboles.

Propiedades:

- Un árbol con n vértices tiene n -1 aristas.
- Si se agrega una arista a un árbol, entonces se obtiene un grafo con un ciclo
- Si se elimina una arista a un árbol, entonces se obtiene un grafo no conexo.

Árbol de cubrimiento:

Sea el grafo $G = (V, A)$ conexo y no dirigido, el subgrafo $T = (V, A')$ es un árbol de cubrimiento de G, si $A' \subseteq A$ es tal que $|A'|$ es mínima para que T sea conexo.

Observación: Se define ***árbol de cubrimiento de costo mínimo***, cuando las aristas están ponderadas, la suma de los valores de las elegidas debe ser la mínima posible.

Recorrido en Profundidad: Depth-first search (DFS)

Dado un vértice *v* de partida, se lo marca como visitado. Luego para todo vértice adyacente a *v* que no haya sido visitado, se lo toma como nuevo punto de partida y se invoca recursivamente al procedimiento.
Cuando están marcados todos los vértices adyacentes a *v*, el recorrido que comenzó en *v* ha finalizado, si queda algún vértice que no haya sido visitado, se lo toma como nuevo punto de partida hasta visitar todos los vértices.

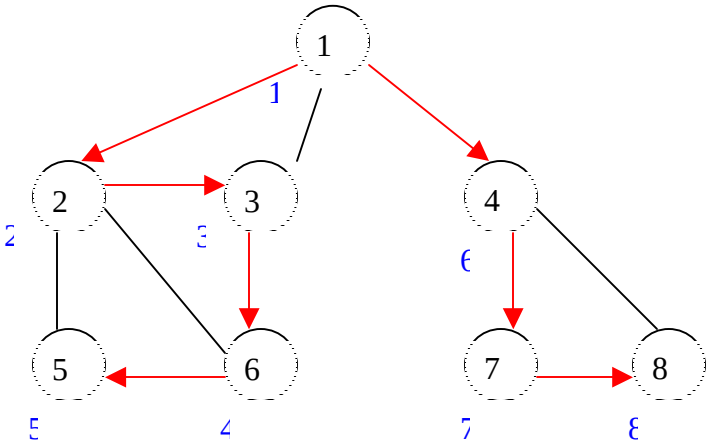
Procedimiento DFS (v: vértice)

```
Comienzo
     marcar v
     <Preprocesamiento>
     Para cada w adyacente a v
          Si w no marcado entonces DFS(w)
     Fin Para
     <Posprocesamiento>
Fin
```

Procedimiento Recorrido_DFS(G: grafo) {G = (V, A)}

```
     Var v: vértice
Comienzo
     Para cada v ∈ V: inicializar v como no-marcado
     Para cada v ∈ V:
          Si v no-marcado entonces DFS(v)
Fin
```

Ejemplo:



Vértice de *partida*: 1
***Adyacencia*:** secuencia numérica del id. de vértice.
Marcas: número de recorrido en preorden.

Árbol asociado al recorrido DFS(1)

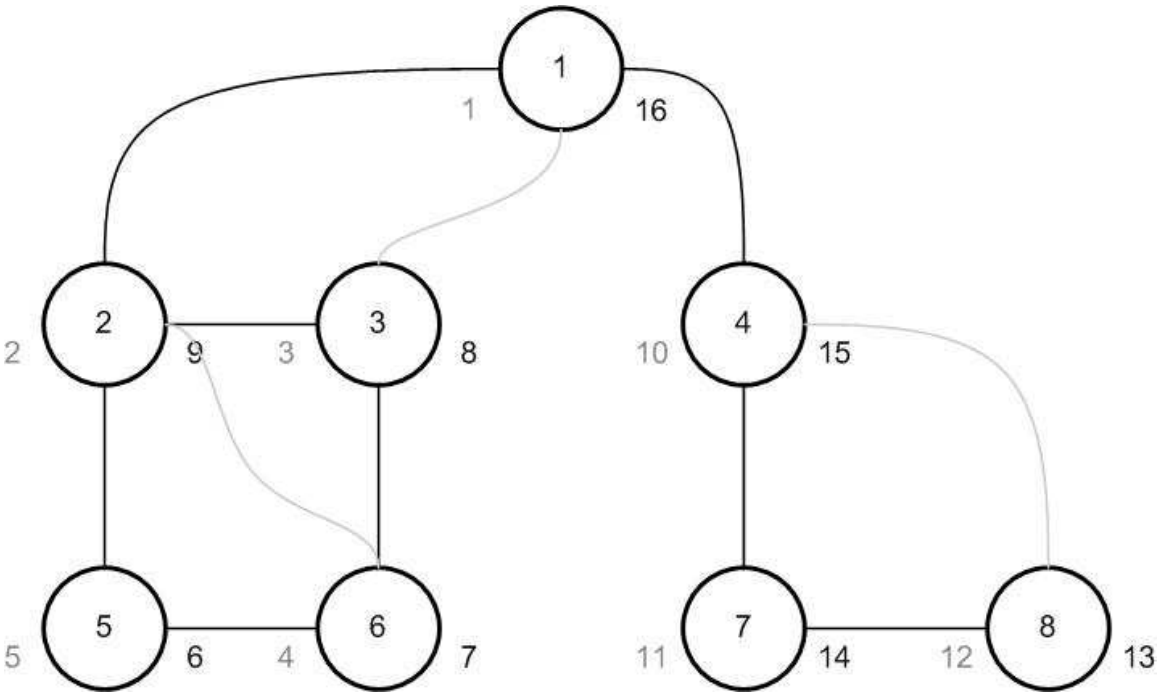
El recorrido DFS genera un árbol de cubrimiento, cuyas aristas se denomina A_T
Las aristas A-A_T no cruzan ramas del árbol (van de un vértice a un ancestro)

Etapa	*Nivel de recurrencia*	
1	DFS(1)	Inic.
2	DFS(2)	Rec.
3	DFS(3)	Rec.
4	DFS(6)	Rec.
5	DFS(5)	Rec.
6	DFS(4)	Ady.
7	DFS(7)	Rec.
8	DFS(8)	Rec.

Observaciones:

a) El recorrido DFS de un grafo conexo, genera un subgrafo que es un árbol de recubrimiento, *T*. En un grafo en general, genera un bosque de árboles (uno por cada componente conexa)
b) Complejidad tiempo de recorrido: O (|*V*| + |*A*|)
c) Un recorrido DFS permite marcar los vértices del grafo en el momento de comienzo de procesamiento (preorden) y/o en el momento de fin del procesamiento (posorden).

Ejemplo de recorrido DFS con marcado de comienzo y fin de procesamiento
Vértice de *partida*: 1
Adyacencia: secuencia numérica del id. de vértice.
Marcas de tiempo: tiempo de comienzo de procesamiento, subíndice izquierdo (c_v), tiempo de fin de procesamiento, subíndice derecho (f_v).



Si *v* es visitado antes que *w*, $c_v < c_w$, solo se dan los casos:

a. $c_v < c_w < f_w < f_v$
b. $c_v < f_v < c_w < f_w$

Recorrido en Amplitud: Breadth-first search (BFS)

El recorrido BFS comienza en un vértice dado y visita primero a todos sus vértices adyacentes. Luego para cada vértice adyacente, recorre los vecinos respectivos no visitados. Así sucesivamente hasta alcanzar todos los vértices. (Recorrido exhaustivo ó búsqueda no-informada)

Observaciones:

a) No es naturalmente recursivo
b) La expansión de los vértices adyacentes a uno dado puede representarse con una cola (FIFO).

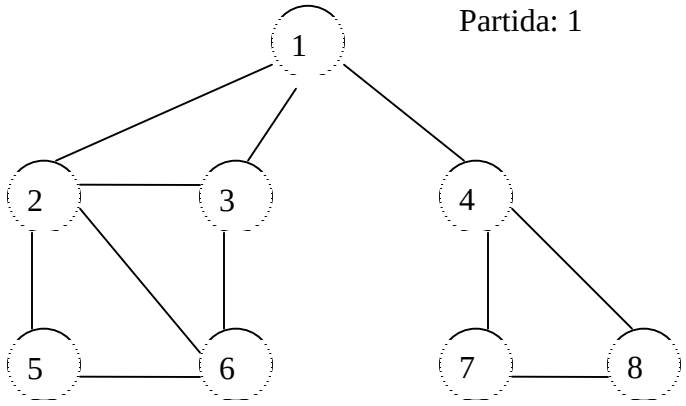
Procedimiento BFS (v: vértice)

```
      Var   Q: cola de vértices
            u,w: vertice
Comienzo
     CrearCola(Q)
     Marcar(v)
     InsBack(Q,v) //"encolar"
     Mientras No-Vacia(Q)
          u = primero(Q)
          <Procesar(u)>
          Q = resto(Q)
          Para cada w adyacente a u
               Si w no marcado
                    Marcar(w)
                    InsBack(Q,w)
               Fin Si
          Fin Para
     Fin Mientras
Fin
```

Procedimiento Recorrido_BFS(G: grafo)

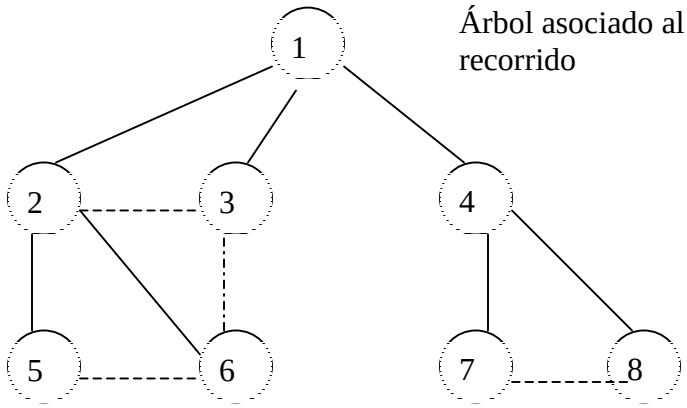
```
     Var v: vértice
Comienzo
     Para cada v ∈ V: inicializar v como no marcado
     Para cada v ∈ V:
          Si v no marcado entonces BFS(v)
Fin
```

Ejemplo:



Comienzo en vértice 1, adyacencia orden numérico ascendente de identificador.

Vértice visitado	Q
1	2, 3 , 4
2	3, 4, 5, 6
3	4, 5, 6
4	5, 6, 7, 8
5	6, 7, 8
6	7, 8
8	-



Árbol asociado al recorrido

El recorrido BFS genera un subgrafo que es un árbol de cubrimiento, *T*.

Las aristas de A-A_T se clasifican en 2 tipos, tomando como rector los niveles en los que se pueden clasificar los vértices según el árbol generado:

a. aristas entre vértices del *mismo nivel*
b. aristas entre vértices de *niveles adyacentes*

Tiempo de ejecución: O (max(|*V*|,|*A*|))

Observaciones:

a) BFS se usa especialmente cuando el grafo es infinito (camino infinito o muy grande)
b) Para hallar el camino más corto entre un par de vértices. El árbol BFS es un árbol de camino más corto comenzando desde la raíz (con largo igual a su nivel).

Puntos de Articulación (PA)

Un vértice de un grafo no dirigido es un **punto de articulación**, si el subgrafo que se obtiene, borrándolo junto a todos sus aristas incidentes contiene más componentes conexas que el original.

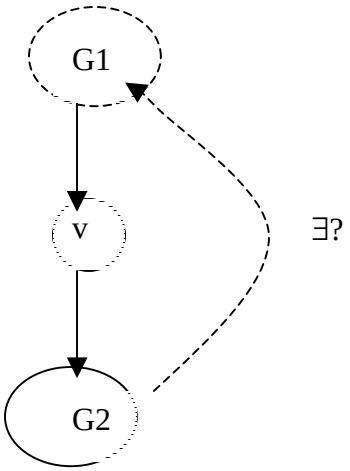
Determinación de si un vértice es PA

Para verificar si un vértice *v* es PA, se obtiene un árbol de cubrimiento (DFS) con raíz *v*, si *v* tiene más de un descendiente entonces *v* es PA.

Hallar todos los PA de un grafo

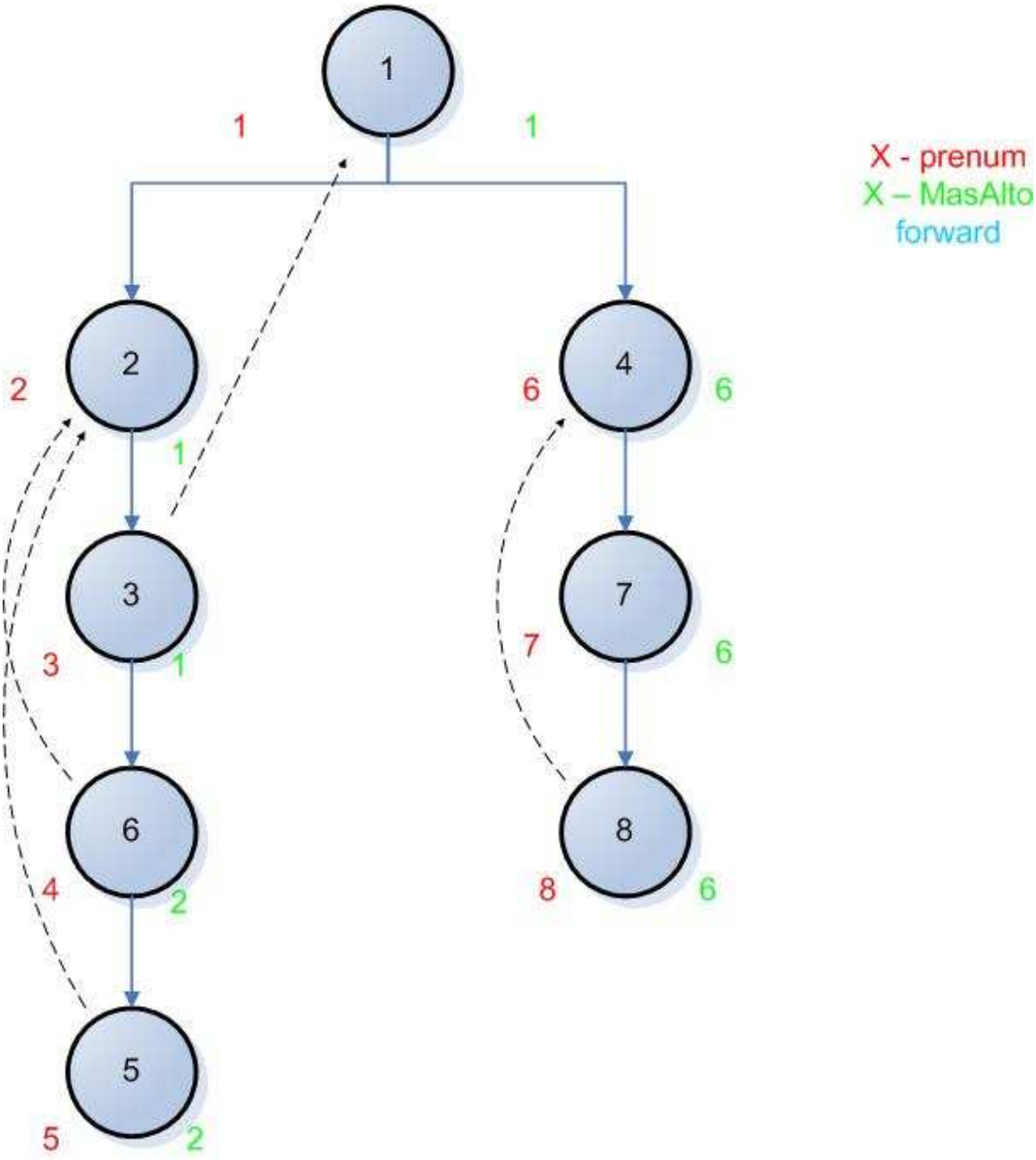
1. Se puede realizar recorridas DFS para cada vértice.
   Tiempo de ejecución: O (|*V*| (|*V*|+|*A*|))

2. Dado el árbol de cubrimiento generado por DFS. Un vértice *v* (no raíz, ni hoja) será PA si no hay arista de retorno de sus sucesores a sus antecesores.



Solución 1
Ver si en G2 hay algún vértice, *w*, con arista back a un vértice visitado antes que *v*: `prenum`(*w*) < `prenum`(*v*).
Esto llevaría a hacer algoritmos especiales para cada vértice

Solución 2
Introduce un nuevo número para cada vértice: `masalto`(*v*). Sea *w* el nodo más alto del árbol que se puede alcanzar desde *v* con un camino de cero o más aristas de *T*, y al final una sola arista back.
Se define `masalto`(*v*) := `prenum`(*w*)



Las aristas back no cruzan de una rama a la otra.
Para un vértice dado, el vértice más alto alcanzable tiene que ser un antecesor.

Dejando de lado la raíz:

- Si *v* no tiene descendientes entonces no es PA
- Si *v* tiene descendientes, sea *x* un hijo de *v* en el árbol *T*:
  - Caso 1: `masalto`(*x*) < `prenum`(*v*)
    hay un camino (que excluye al camino (*v*,*x*)) que partiendo de *x* nos lleva a un vértice de *T* más alto que *v*

Si se borra *v*, los vértices del subárbol cuya raíz es *x* no quedaron desconectados del resto del árbol.

- Caso2: `masalto`(*x*) >= `prenum`(*v*)
  No hay camino (excluyendo (*v*,*x*)) que parta de *x* y que esté encima de *v*.
  Si se borra *v*, los vértices del subárbol cuya raíz es *x* quedaron desconectados del resto del árbol

Un vértice v que no sea raíz de *T* es un PA de G, si tiene al menos un hijo *x* con `masalto`(*x*) >= `prenum`(*v*).

`masalto`(*v*) se calcula en `postorden`; y es el mínimo entre:

- `prenum`(*v*) quedándose en el vértice
- `prenum`(*w*) donde (*v*,*w*) arista back (ascenso back)
- `masalto`(*x*) para todo hijo *x* de *v* (descenso forward)

Algoritmo (resumido) para hallar los PA (grafo dirigido)

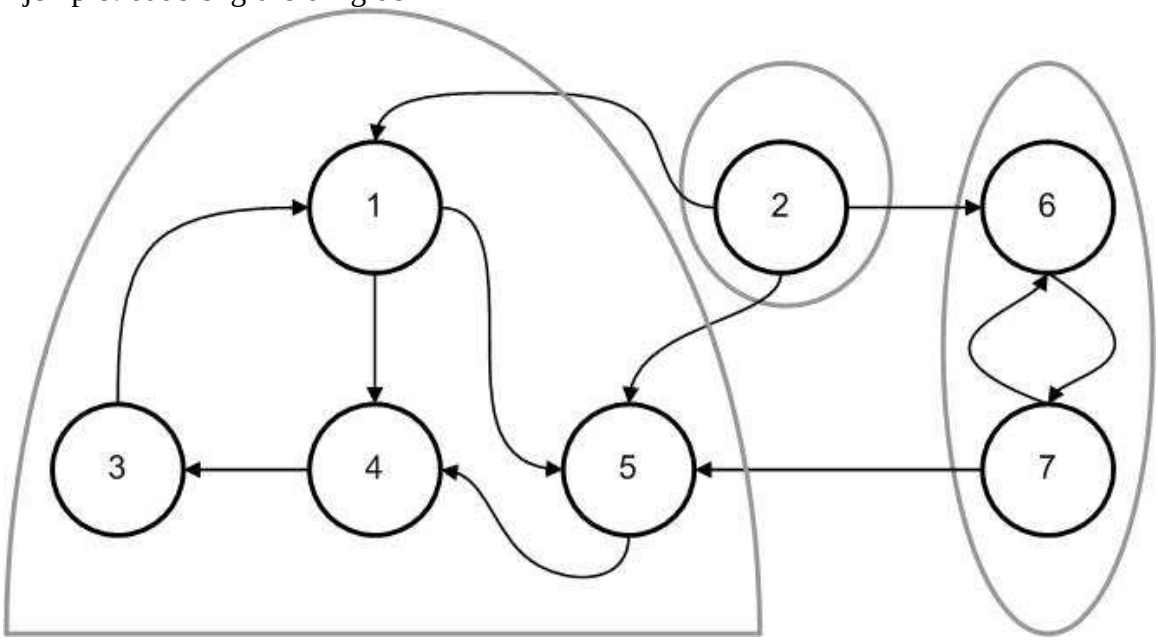
1. Se recorre DFS de G numerando en preorden, generando árbol $T = (V, A_T)$ $(\forall v \in V, prenum(v))$
2. Se recorre DFS de G procesando en postorden.
   Para cada vértice v visitado se calcula `masalto`(v) como mínimo de:
   - `prenum`(v)
   - `prenum`(w) $\forall w$ si $(v,w) \in A - A_T$
   - `masalto`(x) para todo hijo x de v
3. Se determinan los PA de G:
   - La raíz de T es PA si tiene más de un hijo
   - Cualquier otro vértice v es PA si tiene un hijo x tal que `masalto`(x) >= `prenum`(v)

Componentes Fuertemente Conexas (CFC)

(Grafos dirigidos)
Un grafo dirigido es fuertemente conexo si hay un camino (dirigido) entre todo par de vértices.
Esto establece una relación de equivalencia que particiona a *V* en CFC.

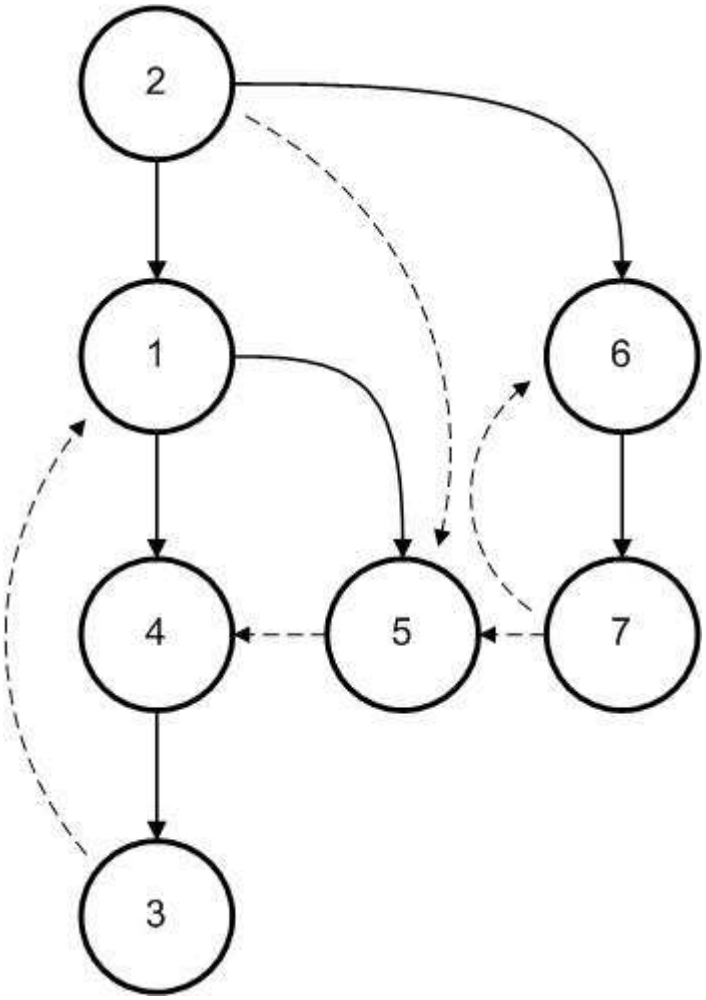
El recorrido DFS para grafos dirigidos
Es similar al recorrido para grafos no dirigidos con la diferencia de el vértice *w* que es adyacente a *v* si existe la arista dirigida (*v*,*w*). El resultado es algo diferente con respecto a las aristas del grafo que pertenecen al árbol.

Ejemplo: dado el grafo dirigido



Se comienza la recorrida en el vértice 2, la adyacencia se determina a según la secuencia numérica creciente de identificadotes.

El resultado es un árbol de cubrimiento: $T = (V, A_T)$


2
1
6
4
5
7
3

Tipos de aristas: A - A_T según vínculos:

Regreso (back): de descendientes a ancestros
Cruzadas (cross): de subárbol derecho a subárbol izquierdo (*)
Directa (forward): de ancestros a descendientes (*)

(*) Aristas que apuntan a vértices procesados.

Observación:

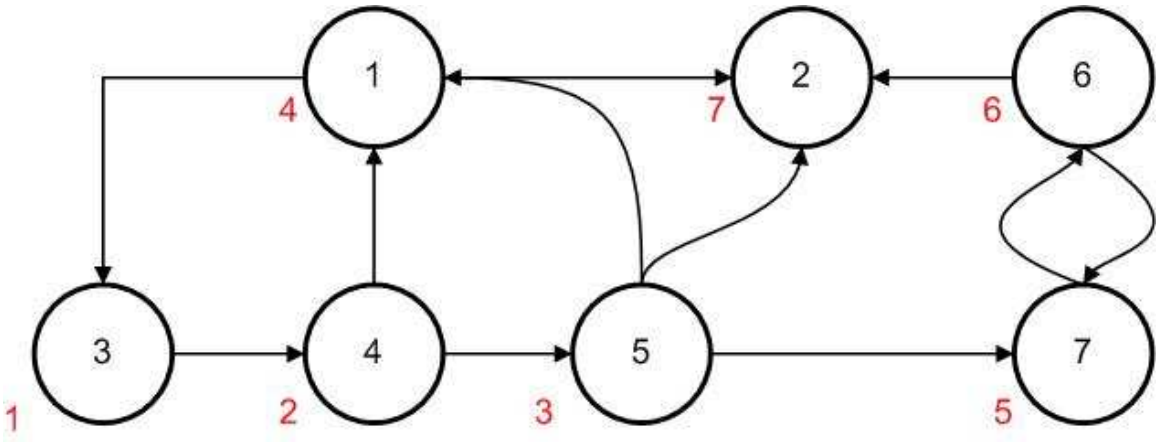
Un grafo dirigido es acíclico, si y solo si no tiene aristas de regreso (back)

Procedimiento para encontrar CFC

1. Recorrida DFS de G numerado en postorden
2. Construir el grafo transpuesto de G, G^T, con los mismos vértices de G y las mismas aristas pero con el sentido invertido
3. Hacer DFS en G^T. Siempre que haya que comenzar un nuevo árbol comenzar por el vértice que tenga postnum (el del paso 1) mayor entre los no marcados.

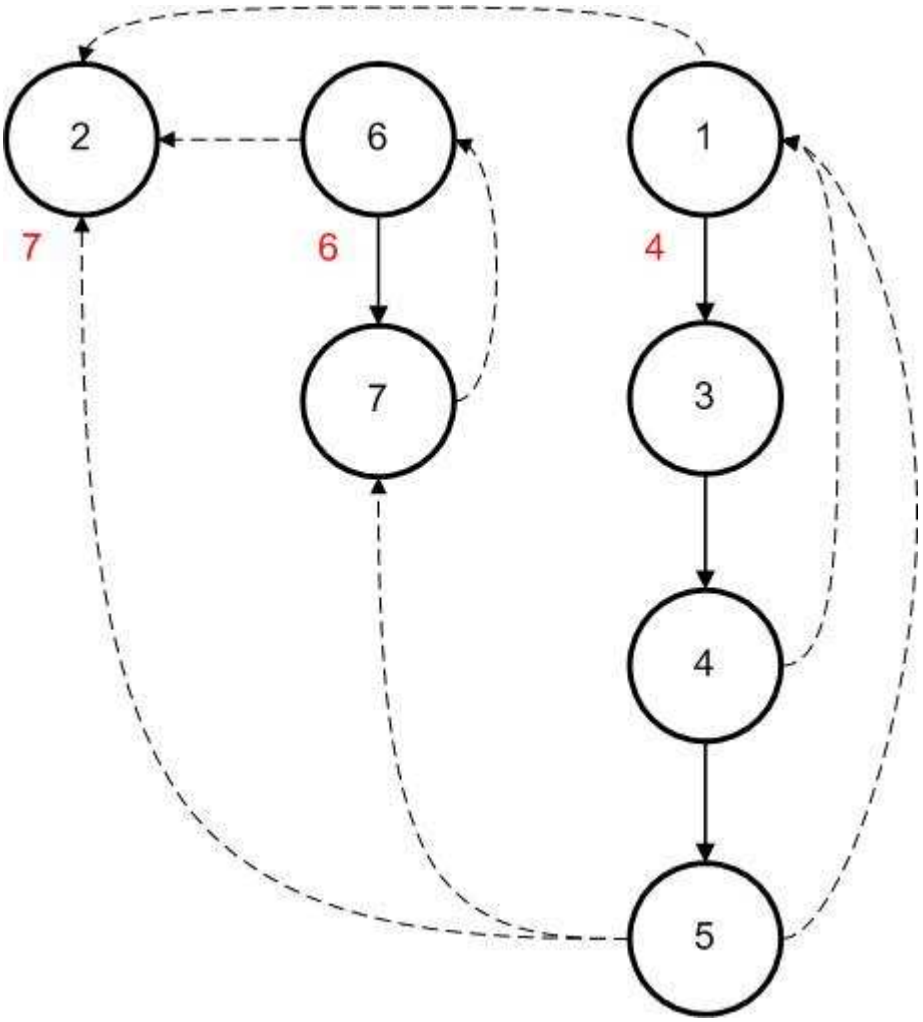
Cada árbol resultado del procedimiento, corresponde a un CFC de G.

G^T con postnum del recorrido DFS del grafo G comenzando en vértice 2



Resultado de recorrido DFS de G^T a partir de mayor postnum

En caso de que haya que recomenzar el recorrido se empieza en el vértice que tenga el mayor posnum entre los no marcados.



El resultado es un bosque de tres árboles, cada uno corresponde a una componente fuertemente conexa del grafo *G*.

Correctitud

Dos vértices *v* y *w* están en el mismo árbol – DFS de G^T sii *v* y *w* están en la misma CFC en *G*.

1. *v* y *w* están en diferentes árboles. Todo arco entre cualquier vértice del árbol de *v* y cualquier vértice del árbol de *w* es un arco cross. Dado que los arcos cross van en un solo sentido, *v* y *w* no están fuertemente conectados.
2. *v* y *w* están en el mismo árbol.
   Suponer que *r* es la raíz de dicho árbol.
   a. Existe un camino [*v*, *r*] en *G*, por construcción dado que existe un camino [*r*, *v*] en G^T
   b. `postnum`(*r*) >`postnum` (*v*) (orden de selección)
   c. *r* es un ancestro de *v* en el árbol de *G* (por a. y b.)
   d. Existe un camino de *r* a *v* en *G* (por c.)
   e. Existen caminos [*r*, *v*] y [*v*, *r*] G^T (por a. y d.)

   Por lo tanto *v* y *r* están fuertemente conectados en G^T

   El razonamiento puede generalizarse a otros vértices usando la relación de equivalencia de CFC. Si el vértice *v* está fuertemente conectado a *r* y *r* está fuertemente conectado a *w*, entonces *v* está fuertemente conectado a *w*.