Lenguaje Natural:

* Hacer una bpf desde un vértice v del grafo
* Luego verificamos que todos los vértices estén visitados
* Creamos otro grafo con las aristas invertidas, ejecutamos una bpf desde el mismo vértice anterior y repetimos la verificación.
* Si todos los vértices estuvieron visitados en los dos casos es conexo fuerte

Seudo:

TGrafoDirigido.esFuertementeConexo(): booleano

Comienzo

DesvisitarMaricones()

vertice 🡨 vertices.agarrarCualquieraJAJAXDSALUDOSBIGOTEEEE

vertice.bpf()

ParaCada bertice en vertices Hacer O(V)

Si (bertice.esVisitado() = falso) Entonces

Devolver falso

FinSi

FinPara

aristasInvertidas 🡨 nueva lista

ParaCada bertic3 en vertices Hacer O(V)

ParaCada adyacencia en adyacencias O(A)

aristaInvertida 🡨 nuevaarista(adyacencia.destino,bertic3,adyacencia.costo)

aristasInvertidas.add(aristaInverida) O(1) (porque es enlazada pro)

FinPara

FinPara

grafoInvertido 🡨 nuevo TGrafoDirigido(vertices, aristasInvertidas)

vertic 🡨 grafoInvertido.buscar(vertice.etiqueta)

vertic.bpf()

ParaCada berti en grafoInvertido.vertices Hacer O(V)

Si (berti.esVisitado() = falso) Entonces

Devolver falso

FinSi

FinPara

Devolver Verdadero

Fin

El orden es O(A\*V)

1. Efectúese una búsqueda en profundidad de G y numérense los vértices en el orden

de terminación de las llamadas recursivas; esto es, asígnese un número al

vértice *v* después de la línea (4) de la figura 6.24.

1. Constrúyase un grafo dirigido nuevo G, invirtiendo las direcciones de todos los

arcos de G.

1. Realícese una búsqueda en profundidad en G~ partiendo del vértice con nume·

raci6n más alta de acuerdo con la numeraci6n asignada en el paso (1). Si la bús·

queda en profundidad no llega a todos los vértíces, iníciese la siguiente búsqueda

a partir del vértice restante con numeraci6n más alta.

1. Cada árbol del bosque abarcador resultante es un componente fuertemente conexo

de G.

Seudocódigo

TGrafoDirigido.componentesFoeeertes()

Comienzo

verticeDeGrafo 🡨 vertices.sacarCualquiera

verticeDeGrafo.numerar()

verticeMaximo 🡨 nuevo vertice

aristasInvertidas 🡨 nueva lista

aristas 🡨 nueva lista

ParaCada vértice en vertices Hacer

Si vertice.numBP > verticeMaximo.numBP

verticeMaximo 🡨 vertice

FinSi

ParaCada adyacencia en vertice.adyacencias O(A)

arista 🡨 nueva arista(vertice,adyacencia.destino,adyacencia.costo)

aristas.add(arista)

aristaInvertida 🡨 nueva arista(adyacencia.destino,vertice ,adyacencia.costo)

aristasInvertidas.add(aristaInverida) O(1) (porque es enlazada pro)

FinPara

grafoInvertido 🡨 nuevo TGrafoDirigido(vertices, aristasInvertidas)

comienzoDeBpf 🡨 grafoInvertido.buscar(verticeMaximo.etiqueta)

comienzoDeBpf.bpf()

noVisitados 🡨 nueva Lista

componentes 🡨 nueva Lista

primerComponente 🡨 nueva Lista

Para Cada vertice en grafoInvertido Hacer

Si vertice no visitado entonces

noVisitados.insertar(vertice)

SiNo

primerComponente.insertar(vertice)

FinSi

FinPara

primerGrafo 🡨 nuevo Grafo(primerComponentes,aristas)

componentes.add(PrimerGrafo)

Mientras ¡noVisitados.esVacio() hacer

verticeMaximo 🡨 noVisitados.primero

bpMaximo 🡨 vertices.buscar(verticeMaximo.etiqueta).numBP

Para cada vertice en noVisitados Hacer

Si vertice.visitado() = false Entonces

Si vertices.buscar(vertice.etiqueta).numBP > bpMaximo Entonces

verticeMaximo 🡨 vertice

FinSi

SiNo

noVisitados.eliminar(vertice)

FinSi

FinPara

noVisitados.eliminar(verticeMaximo)

verticesConexo 🡨 nueva lista

verticeMaximo.bpfEnLista(verticesConexo)

componenteConexo 🡨 nuevo grafo(verticesConexo,aristas)

FinMientras

Retornar componentes

FIN!