Ayudantía 14

Grafos + DFS

Definición

Grafos

Un grafo es un par G=(V,E) en donde:

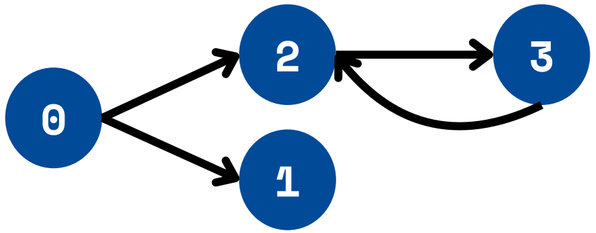
V es un conjunto de nodos

E es un conjunto de aristas

V={0,1,2,3}

E={(0,1)(0,2),(2,3),(3,2)}

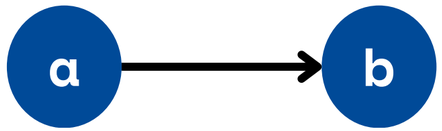
Tal que ambos generan una representación gráfica



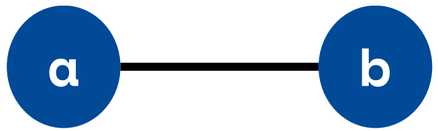
Definición

Dirigidos v/s no dirigidos

Si un grafo **es dirigido**...
Una arista (a,b) representa que desde a puedo llegar a b, no viceversa.
En general pueden tener inicio y fin dependiendo de su uso.



Si un grafo **es NO dirigido**...
Una arista (a,b) representa que desde a puedo llegar a b, como así también de b puedo llegar a a.
En general pueden representarse con flechas sin punta.



Representación

Grafos

Hay 2 tipos de representaciones de grafos:

Las **listas de adyacencia** son listas en las que el indice representa un nodo y se almacena una lista de todos los vecinos de dicho nodo

```
L=[[1],[0,3],[1,3],[0]]
```

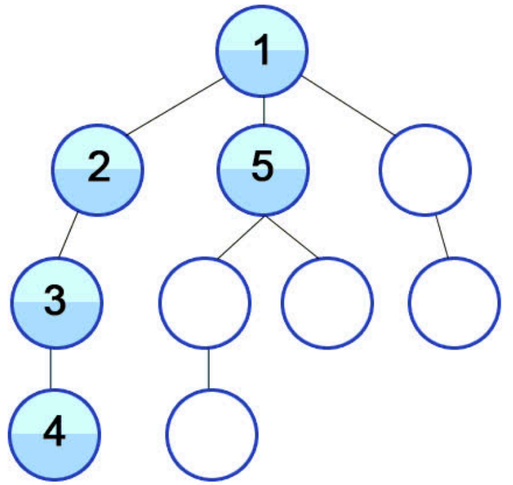
Las **matrices de adyacencia** son matrices que indican con booleanos (usualmente 0 o 1) si existe arista en una celda [i][j]

```
          (hasta)
(desde) L=[[0,1,0,0],
           [1,0,0,1],
           [0,1,0,1],
           [1,0,0,0]]
```

Definición

Depth First Search (DFS)

Es un algoritmo de búsqueda en grafos por profundidad (como dice su nombre) que recorre completamente una rama y luego se devuelve a revisar las demás.



Pseudocódigo

DFS



Blanco

Nodos **no visitados**. El punto de partida de cualquier exploración inicial.



Gris

Visitado y **en proceso**. Sus descendientes se están explorando en este momento.



Negro

Visitado y **terminado**. Todos sus vecinos han sido completamente procesados.

Pseudocódigo

DFS base

```
input : grafo G
Dfs(G):
1     for u ∈ V(G) :
2         u.color ← blanco
3     for u ∈ V(G) :
4         if u.color = blanco :
5             DfsVisit(G, u)
```

```
input : grafo G, nodo u ∈ V(G)
DfsVisit(G, u):
1     u.color ← gris
2     for v ∈ N_G(u) :
3         if v.color = blanco :
4             DfsVisit(G, v)
5     u.color ← negro
```

Definición

Tiempos de vida

Se definen dos marcas temporales para cada uno de los nodos:

u.start

Tiempo en que el nodo es descubierto (Gris).

u.end

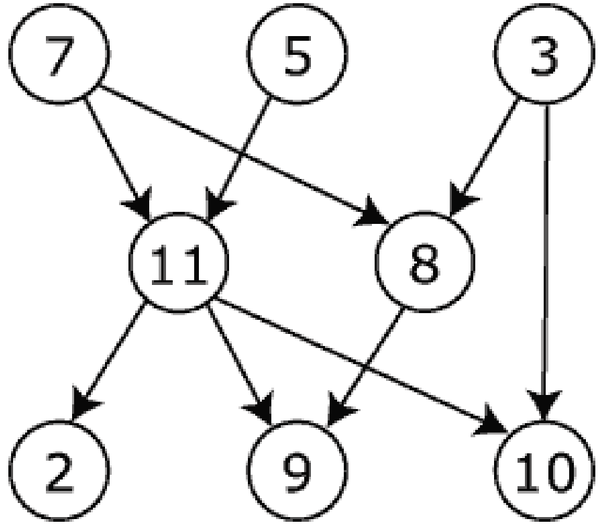
Tiempo en que el nodo es finalizado (Negro).

Propiedad: Para cualquier par u, v, sus intervalos [start, end] son disjuntos o uno está contenido en el otro.

Definición

Orden topológico

En un Grafo Acíclico Dirigido (DAG), queremos ordenar los nodos tal que si existe (u, v), u aparezca antes que v. Se busca ejecutar DFS sobre G y listar nodos en orden decreciente en base a su final (u.end)



Pseudocódigo

Orden topológico

input : grafo G
output: lista L de nodos en orden topológico

```
TopSort(G):
1    L ← lista vacía
2    Dfs(G)
3    Insertar en L nodos en orden decreciente según end
4    return L
```

Para esto el DFS normal no es suficiente...

Entonces hacemos **una versión con tiempos!**

Pseudocódigo

DFS con tiempos

```
input  : grafo G
Dfs(G):
1       t ← 1
2       for u ∈ V(G) :
3           u.start ← 0
4           u.end ← 0
5       for u ∈ V(G) :
6           if u.start = 0 :
7               t ← DfsVisit(G, u, t)
```

```
DfsVisit(G, u, t):
1       u.start ← t
2       t ← t + 1
3       for v ∈ N_G(u) :
4           if v.start = 0 :
5               t ← DfsVisit(G, v, t)
6       u.end ← t
7       t ← t + 1
8       return t
```

input : grafo G, nodo $u \in V(G)$, tiempo t
output: tiempo $t \geq 1$

Pseudocódigo

Orden topológico

De todas maneras es posible modificar dfs para no tener que agregar complejidad al ordenar :)

```
TopSort(G):
    L ← lista vacía
    t ← 1
    for u ∈ V(G) :
        u.start ← 0
        u.end ← 0
    for u ∈ V(G) :
        if u.start = 0 :
            t ← TopDfsVisit(G, L, u, t)
    return L
```

input : grafo G
output: lista de nodos L

```
TopDfsVisit(G, L, u, t):
    u.start ← t
    t ← t + 1
    for v ∈ N_G(u) :
        if v.start = 0 :
            t ← TopDfsVisit(G, L, v, t)
    u.end ← t
    Insertar u como cabeza de L
    t ← t + 1
    return t
```

Definición

Algoritmo de Kosaraju

DFS permite identificar grupos de nodos donde todos pueden llegar a todos (componentes fuertemente conexas o CFC):

1. Obtener orden decreciente de u.end en G.
2. Calcular el Grafo Transpuesto GT (invertir aristas).
3. Ejecutar DFS en GT siguiendo el orden del paso 1.

Pseudocódigo

Algoritmo de Kosaraju

input : grafo G

Kosaraju(G):

1 $L \leftarrow \text{TopSort}(G)$

2 **for** $u \in L$:

3 Assign(u, u)

input : grafo G, nodo $u \in V(G)$, nodo representante r

Assign(G, u, r):

1 **if** $u.rep = \varnothing$:

2 $u.rep \leftarrow r$

3 **for** $v \in N_{G^T}(u)$:

4 Assign(G, v, r)

Ejercicio 1

Una nueva inteligencia artificial, shutupGPT, fue usada para diseñar el sentido del tráfico vehicular de las calles de una ciudad experimental, Ciudad Crepuscular, en las afueras de Santiago. Para asegurarse de que shutupGPT había hecho un buen trabajo, el Misterio de Trasporte decidió hacer un solo test: comprobar que el grafo correspondiente a los sentidos de las calles **no fuera acíclico**;

Pregunta 4 - I2-2025-2



a) Explica detallada y precisamente cómo se construye el grafo correspondiente a los sentidos de las calles. Es decir, cuáles son los vértices (nodos); cuáles son las aristas (arcos); y qué tipo de grafo es: con costos, sin costos, direccional, no direccional.

Pregunta 4 - I2-2025-2

Pregunta 4.a) - I2-2025-2

Las calles mismas constituyen las aristas (o arcos). Todo cruce de calles es un vértice (o nodo); por lo tanto, una arista es en realidad un tramo continuo de calle sin ningún cruce. Las aristas son direccionales, según el sentido del tráfico de la calle correspondiente; si una calle tiene tráfico en ambos sentidos, entonces se la representa mediante dos aristas, cada una dirigida en una de las dos direcciones. El grafo es direccional sin costos.

b) Explica detallada y precisamente cómo se hace el test para determinar que el grafo no es acíclico, y por qué tiene sentido hacerlo.

Pregunta 4 - I2-2025-2

Pregunta 4.b) - I2-2025-2

Para determinar la presencia (o no) de ciclos en un grafo direccional, ejecutamos el algoritmo DFS; si durante la ejecución del algoritmo, llegamos a un vértice pintado de gris (o equivalentemente, identificamos una arista “hacia atrás”), entonces el grafo es cíclico.

Si el grafo direccional, que representa los sentidos del tráfico de las calles, fuera acíclico, entonces las personas que salen (en vehículo) de un lugar, no podrían volver a él.

La gente se fue a vivir a Ciudad Crepuscular y durante las primeras semanas todo funcionaba aparentemente bien. Sin embargo, después de un tiempo, algo curioso comenzó a ocurrir: se empezaron a formar “barrios”, de modo que la gente de movía (en vahículo) sólo dentro de sus respectivos barrios. Y después de algún tiempo adicional, algo verdaderamente extraño comenzó a ocurrir: algunos barrios empezaron a desocuparse, mientras que otros empezaron a sobrepoblarse. A base de lo estudiado sobre grafos en clases, responde:

c) ¿Cómo se explica la formación de barrios?

Pregunta 4 - I2-2025-2

Pregunta 4.c) - I2-2025-2

Según lo estudiado en clases, lo que ocurre es que los sentidos de las calles fueron definidos de manera tal que si bien el grafo tiene ciclos, éstos son sólo “locales,” dando origen a lo que se conoce como componentes fuertemente conectadas (CFCs); en la práctica, cada CFC pasó a ser un barrio: dentro de cada CFC, es posible ir y volver entre cualquier par de vértices.

d) ¿Cómo se explica la desocupación de ciertos barrios y el sobrepoblamiento de otros? ¿Qué tiene que hacer el Misterio de Transporte para revertir esta situación?

Pregunta 4 - I2-2025-2



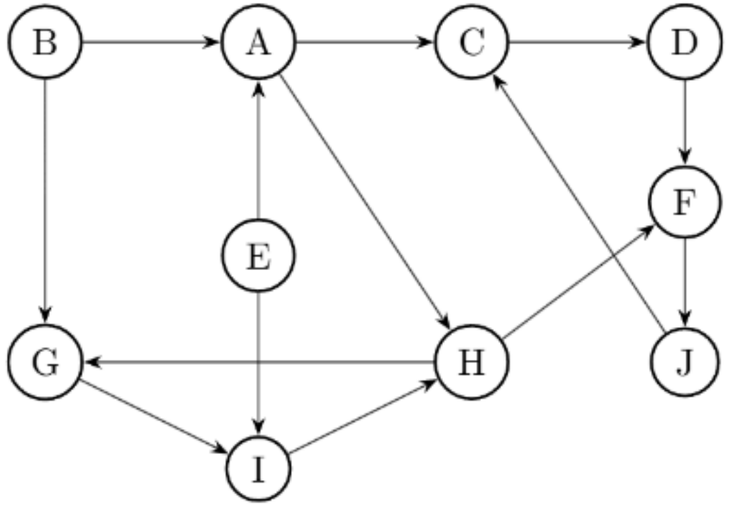
Pregunta 4.d) - I2-2025-2

El grafo de las CFCs de un grafo direccional es un grafo direccional acíclico, de modo que si hay una forma de ir (en vehículo) de un vértice *u* que está en la CFC *A* a un vértice *v* que está en la CFC *B*, **no hay ninguna forma** de volver de *v* a *u*. En términos de la ciudad, lo que ocurrió fue que las personas empezaron a viajar a otros barrios (otras CFCs) y no pudieron volver.

Para cada calle que va de una CFC a otra (en un solo sentido), hay que agregar otra calle en el sentido inverso.

Ejercicio 2

Dado el siguiente grafo dirigido:



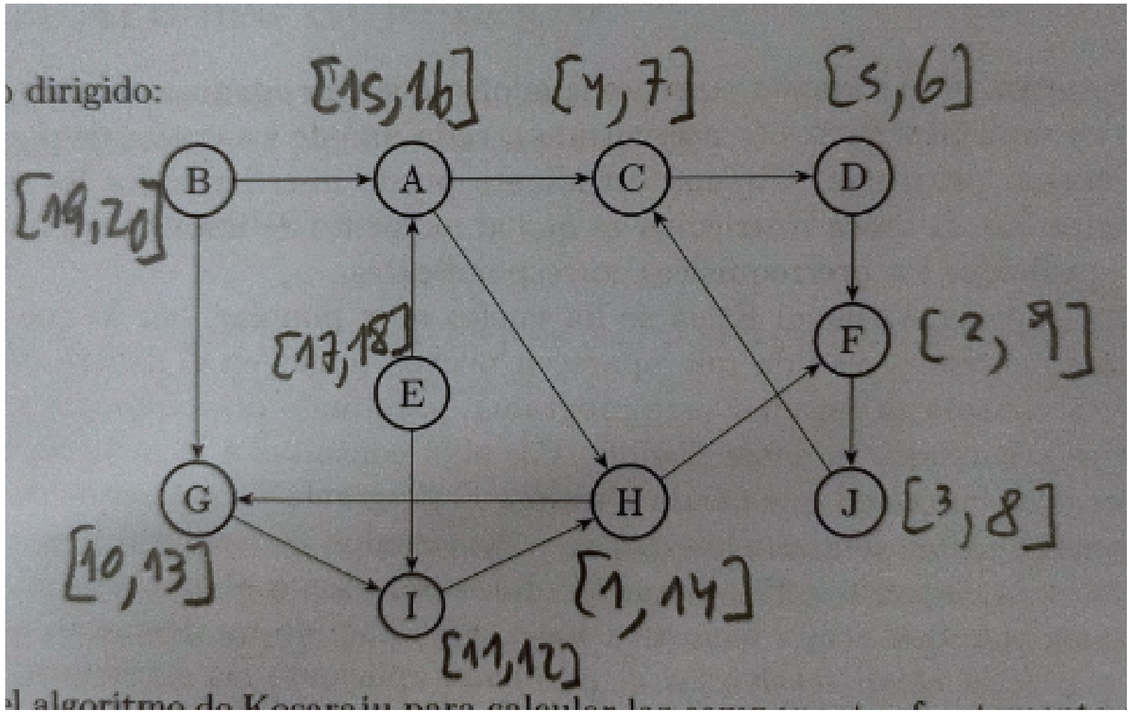
Pregunta 4) - EX-2025-1

a) (3 pts.) Ejecute el algoritmo de Kosaraju para calcular las componentes fuertemente conectadas. Muestre detalles como: intervalos DFS de los nodos, grafo transpuesto, orden topológico de los vértices, y los representantes (arbitrarios) de cada componente.

Pregunta 4) - EX-2025-1

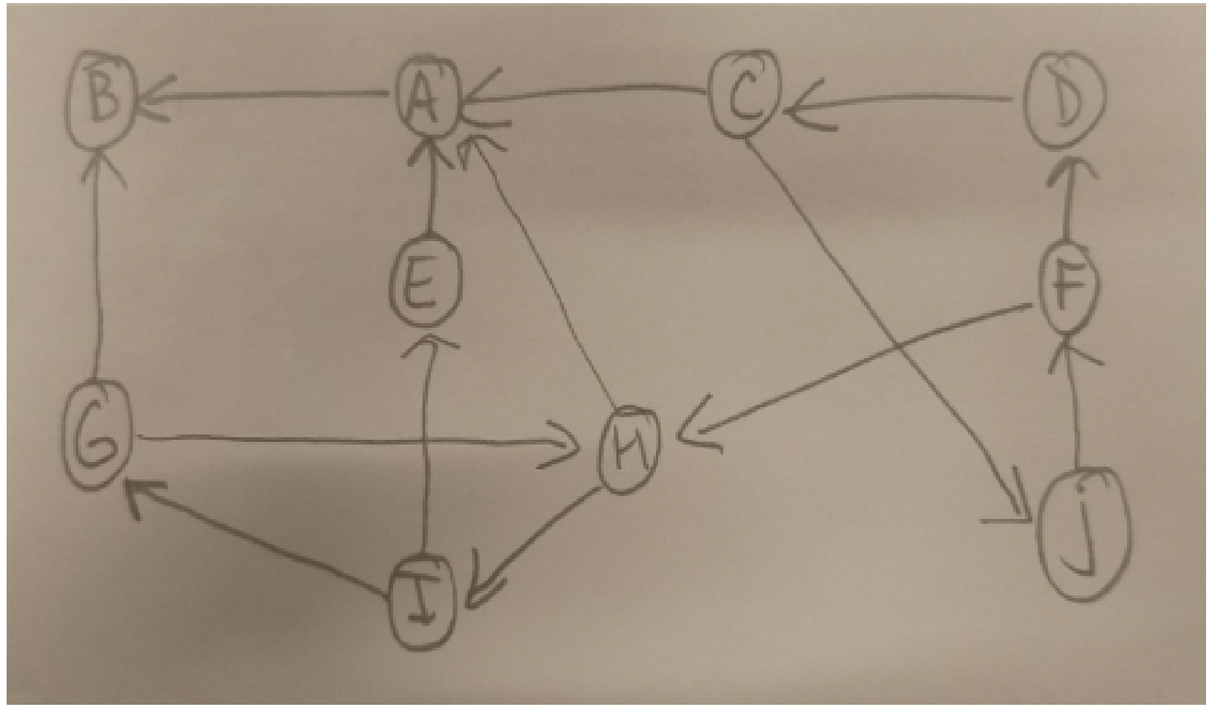
Pregunta 4) - EX-2025-1

DFS comenzando en H (no es relevante el dónde partir)



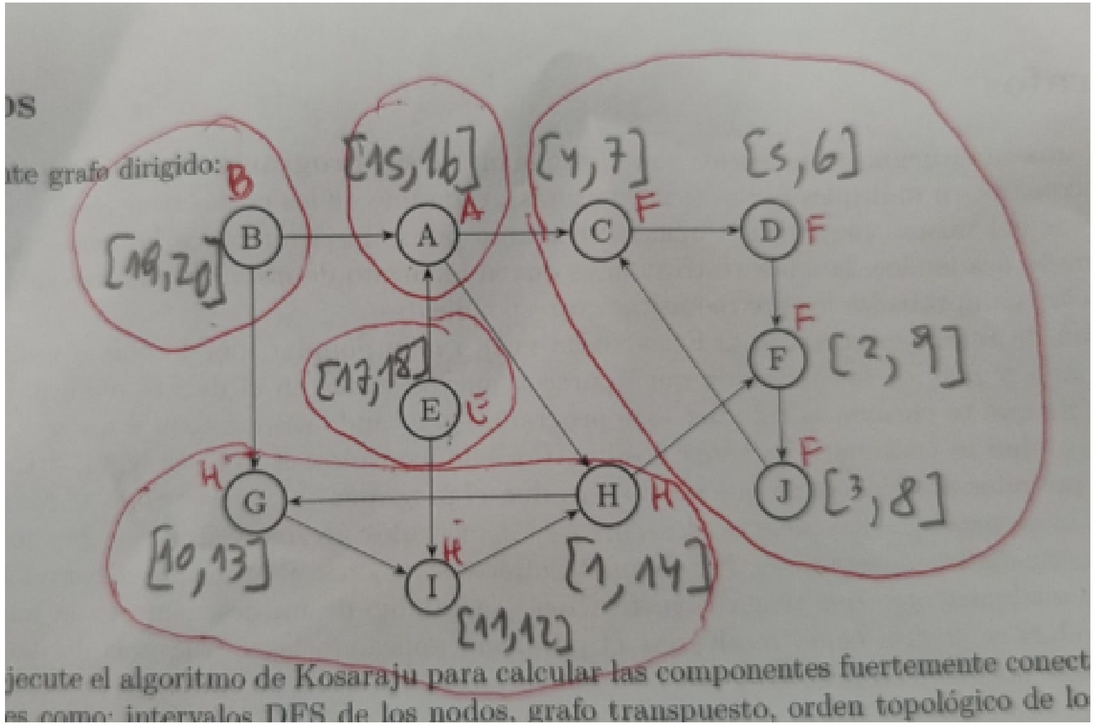
Pregunta 4) - EX-2025-1

Grafo transpuesto



Pregunta 4) - EX-2025-1

Grafo post ejecución de Kosaraju

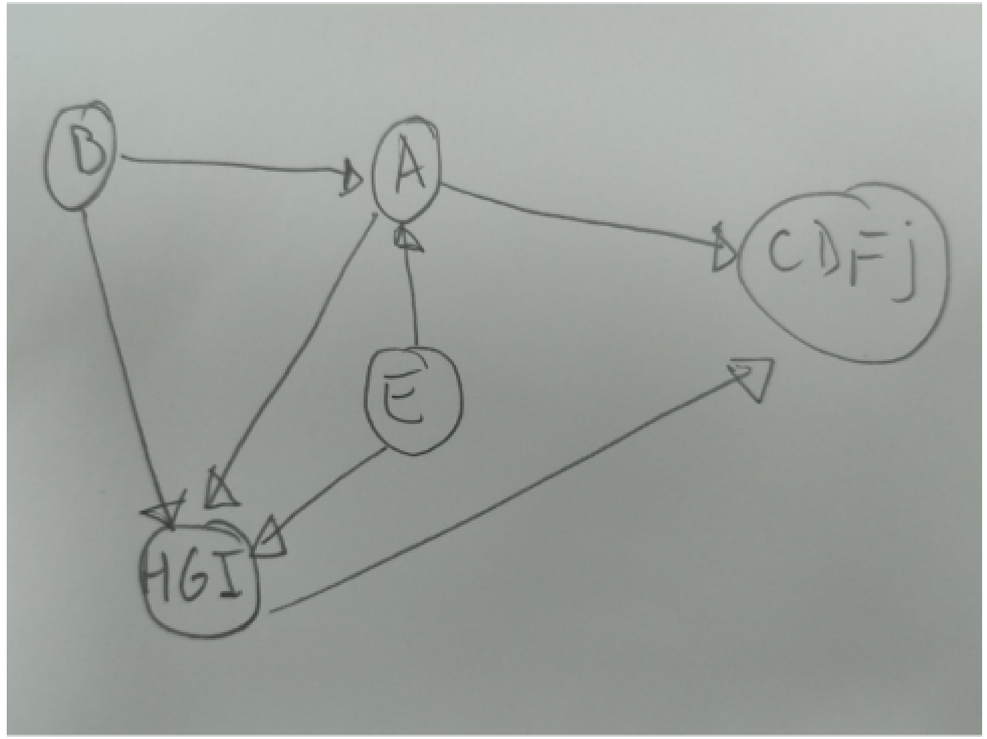


b) (1 pt.) Muestre el grafo de componentes G^{CFC} del grafo.

Pregunta 4) - EX-2025-1

Pregunta 4) - EX-2025-1

Grafo de componentes conexas resultantes de Kosaraju

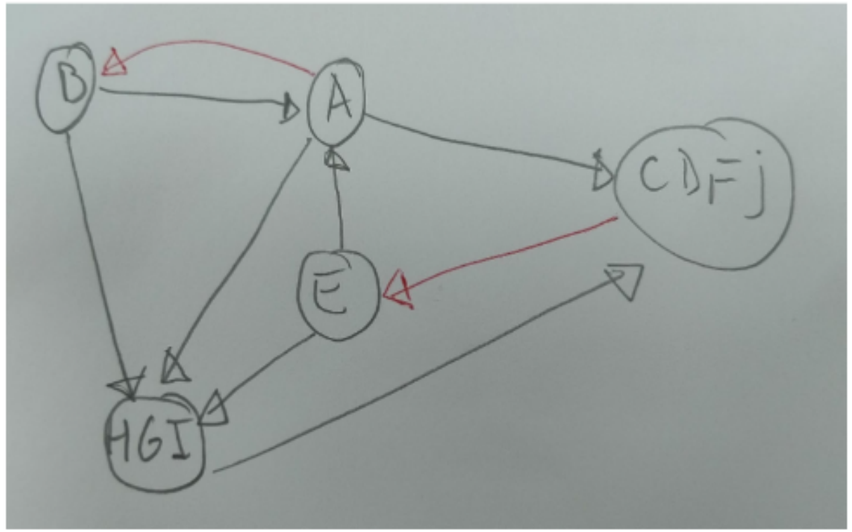


c) (2 pts.) ¿Cuál es la cantidad mínima de aristas que se deben agregar al grafo dirigido original para que tenga una única componente fuertemente conectada? Indique qué aristas agregaría y justifique.

Pregunta 4) - EX-2025-1

Pregunta 4) - EX-2025-1

Solución: El número mínimo de aristas que deben agregarse para que el grafo tenga una única componente fuertemente conectada (es decir, para que todo par de vértices sea mutuamente alcanzable entre sí) es 2. Esto se podía analizar mejor con el grafo de componentes, en donde hay dos vértices fuente (B y E) y un vértice sumidero (CDFJ). Para que CDFJ deje de ser sumidero y E una fuente, hay que agregar la arista CDFJ $\to$ E (alternativamente podían agregar CDFJ $\to B$, también es correcto). Para que B deje de ser una fuente, hay que agregar una arista que llegue a B desde cualquier otro vértice, por ejemplo A $\to$ B. Si agrega aristas sin justificar, el puntaje máximo es de 1 punto. Las aristas en rojo son las agregadas:



Ejercicio 3

Considera una institución de educación superior que ofrece sus programas en la modalidad “on-line.” Un programa consiste en múltiples cursos (o asignaturas), cada uno de los cuales tiene como prerrequisitos cero o más cursos del mismo programa. Un/una estudiante puede inscribir todos los cursos que quiera en un mismo período académico; la única restricción es que al momento de inscribir un curso al inicio del período académico, tenga aprobados los perrequisitos correspondientes.

El programa de Inteligencia Artificial Ética se ha vuelto muy popular, por lo que la institu-ción le ha ido agregando más y más cursos a medida que aparecen nuevos temas en el debate público. De pronto, tú te das cuenta de que te encantaría estudiar este programa, y, estando dispuesto/a a hacer el esfuerzo que sea necesario, tu plan es terminarlo lo antes posible. Por ello, consultas a la institución, “¿Cuál es el mínimo número de períodos académicos en que puedo terminar el programa?”

Pregunta 4) - EX-2025-1

La institución se complica un montón para darte la respuesta —hay demasiados cursos y demasiados prerrequisitos— y por lo tanto distintas personas te dan distintas respuestas. Por lo que tú ofreces solucionar el problema eficientemente —para este programa y cual-quier otro que tenga la institución— a cambio de un descuento en la matrícula.

Explica de manera clara y precisa cómo resolverías el problema aplicando las estructuras de datos y algoritmos estudiados este semestre en iic2133.



Pregunta 4) - EX-2025-1

Pregunta 4) - EX-2025-1

[**2 pts**] Lo primero es plantearse el problema aplicando las estructuras de datos estudiadas; en este caso, un grafo direccional (acíclico, o *DAG*): Un curso es un nodo [**0.7**], y hay una arista direccional del nodo *a* al nodo *b* si *a* es prerrequisito de *b* [**0.7**]; de esta manera, cada programa de la institución es un DAG [**0.6**].

[**2 pts**] Como los estudiantes pueden inscribir todos los cursos que quieran en un mismo período —si cumplen con todos los prerrequisitos correspondientes— entonces el mínimo número de períodos en que puedo terminar un programa es el número de cursos en la secuencia más larga de cursos en que un curso es prerrequisito de otro y éste es prerrequisito de otro y así sucesivamente [**1**]. En la representación del programa como un DAG, la secuencia más larga de cursos correspon-de a la ruta (direccional) con el mayor número de nodos, que también es la ruta con el mayor número de aristas [**1**] (que también es la ruta más larga cuando todas las aristas tienen el mismo costo).

Feedback