CA-20221-EA - Examen parcial de complejidad algorimica desarrollado en python semestre 2022-1

Complejidad Algorítmica (Universidad Peruana de Ciencias Aplicadas)

COMPLEJIDAD ALGORÍTMICA

EVALUACIÓN PARCIAL - EA - 2022-1

Instrucciones

- Debe desarrollar el examen usando únicamente python y enviar sus soluciones en formato **.py** o **.ipynb**.
- En la duración del examen se están contemplando los 10 últimos minutos para adjuntar su solución. Tenga en cuenta este aspecto en caso de contar con dificultades de acceso a internet.
- Soluciones misteriosas o supuestas invalidan la respuesta.
- La detección de plagio parcial o totalmente será penalizado con cero de nota.
- Cada examen cuenta con un equipo académico, el cual estará conectado durante los primeros 20 minutos del examen.
- El alumno debe dedicar los primeros 20 minutos a revisar las preguntas del examen y de presentarse alguna duda enviar un correo al profesor Luis Canaval al correo pcsilcan@upc.edu.pe.
- De no recibir respuesta del equipo académico, o tener algún inconveniente adicional pasado los primeros 20 minutos, puede enviar un correo los profesores Luis Canaval al correo pcsilcan@upc.edu.pe y Patricia Reyes pcsiprey@upc.edu.pe.
- Los profesores en mención, solo recibirán correos provenientes de las cuentas UPC, de ninguna manera se recibirán correos de cuentas públicas.
- Ante problemas técnicos, debe de forma obligatoria adjuntar evidencias del mismo, como capturas de pantalla, videos, fotos, etc. Siendo requisito fundamental que, en cada evidencia se pueda apreciar claramente la fecha y hora del sistema operativo del computador donde el alumno está rindiendo el examen.
- Los problemas técnicos se recibirán como máximo 15 minutos culminado el examen.
- Estamos seguros que cada uno realizará su examen. Sin embargo, para evitar cualquier perspicacia, le recomendamos leer sus reglamentos de estudios y disciplina del alumno, en el cual se indican las faltas y las sanciones en el caso de la copia de exámenes (falta contra la probidad académica).

Parte I

Pregunta 1

(4 puntos)

Implemente una variación del algoritmo DFS, al que llamaremos RDFS, en el que se deberán acceder a los vecinos de cada nodo de manera aleatoria. Por ejemplo dado el grafo siguiente en lista de adyacencia:

0: 1, 5

1: 0, 2, 3

2: 1, 3, 4

3: 1, 2, 5

4: 2, 5

5: 0, 3, 4

El DFS tradicional iniciado desde el nodo cero siempre recorrerá primero al 1 y luego al 5, desde el 1, intentará ir al 0 (que ya está visitado) luego al 2 y luego al 3 y así sucesivamente.

Lo que se le pide es que si ejecutamos el algoritmo solicitado RDFS, partiendo del nodo 0, podría ir primero al nodo 5 antes que al 1, y en otra ocasión podría ir al 1 y luego al 5, etc.

Elabore 3 grafos distintos y realice varias ejecuciones donde se debe mostrar que los resultados son distintos cada vez.

Pregunta 2

(8 puntos)

Usted acaba de terminar una maestría en el MIT y ha decidido salir con sus amigos. Usted ha desarrollado algunos hábitos extraños y por ello ha decidido celebrar éste momento tan importante en su vida tomando muchos jugos tropicales. Usted empieza a tomar bebidas con bajo contenido de azúcar como jugo de melón, luego algo más dulce como jugo de manzana hasta que no hayan más bebidas disponibles. Una vez que empieza a tomar jugo de manzana, ya no tomará más jugo de melón de tal forma que el contenido de azúcar no disminuye con el transcurrir del tiempo.

Siendo el aplicado discípulo de su docente de complejidad algorítmica, decide utilizar python para decidir en qué orden tomar las bebidas según sus propias reglas.

Input

Cada caso de prueba empieza con N, el número de bebidas disponibles, Luego siguen N líneas con los nombres de cada bebida, los nombres no tienen espacios en blanco. Luego sigue M, y luego M líneas de la forma B1 B2 indicando que la bebida B2 tiene más azúcar que la bebida 1. De modo que usted puede tomar la bebida B1 antes de empezar a tomar la B2. Asegúrese que la relación es transitiva, de modo que hay una relación B2 B3, B1 debe tomarse antes que B3. Hay una línea en blanco al final de cada caso de prueba. En caso de no relación entre las bebidas, Usted puede tomar aquella que aparezca primero en el Input.

Output

Por cada caso de prueba usted puede imprimir el mensaje ‘caso #C: usted debería tomar las bebidas en éste orden: B1 B2 … Bn.’, donde C es el número de caso empezando desde 1 y B1, B2 y Bn son las bebidas de modo que el contenido de azúcar de Bi+1 es mayor o igual al contenido de Bi. Luego de cada caso de prueba se debe imprimir una línea en blanco.

Ejemplo de Input

```
3
platano
acai
manzana
2
acai platano
manzana acai


5
acai
manzana
cereza
melon
papaya
6
manzana papaya
melon manzana
melon cereza
manzana cereza
manzana acai
acai papaya


10
papaya
cereza
melon
mango
cocona
acai
platano
manzana
```

```
capuli
durazno
11
manzana cocona
melon durazno
cereza papaya
platano mango
melon capuli
cereza durazno
acai cocona
melon manzana
manzana cereza
acai platano
manzana mango
```

Ejemplo de output

```
Caso #1: usted debería tomar las bebidas en éste orden: manzana acai platano.


Caso #2: usted debería tomar las bebidas en éste orden: melon manzana acai cereza papaya.


Caso #3: usted debería tomar las bebidas en éste orden: melon acai platano manzana cereza
papaya mango cocona capuli durazno.
```

Parte II

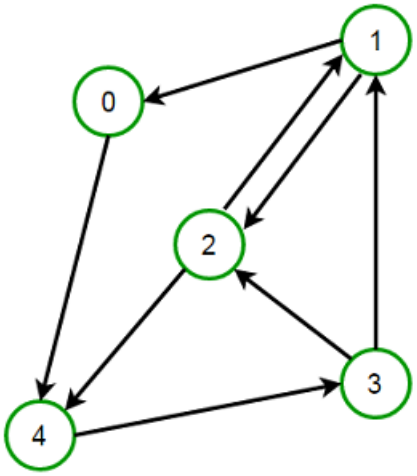
(8 puntos)

En esta parte podrá elegir entre resolver dos, y máximo dos, de las 3 preguntas electivas a continuación. Cada pregunta que responda tiene un puntaje de 4, en caso resuelva más de 2, solo se revisará las 2 primeras que el profesor encuentre. No es permitido resolver las 3 y exigir 12 puntos o exigir que el profesor revise las mejores 2.

Electiva 1: Comprobar si un grafo está fuertemente conectado o no

Dado un grafo dirigido, comprueba si es fuertemente conexo o no. Se dice que un grafo dirigido es fuertemente conexo si todos los vértices son accesibles desde todos los demás vértices.

Por ejemplo, el siguiente gráfico está fuertemente conectado ya que existe un camino entre todos los pares de vértices:



Explique a través de un ejemplo utilizando los algoritmos: búsqueda primero en profundidad (DFS) o una búsqueda primero en amplitud (BFS) y guárdelo como archivo Pregunta1.py

Datos de entrada del programa:

Lista de aristas según el grafo

aristas = [(0, 4), (1, 0), (1, 2), (2, 1), (2, 4), (3, 1), (3, 2), (4, 3)]

total de nodos en el grafo

n = 5

Salida del programa:

1 (si es fuertemente conectado SCC)

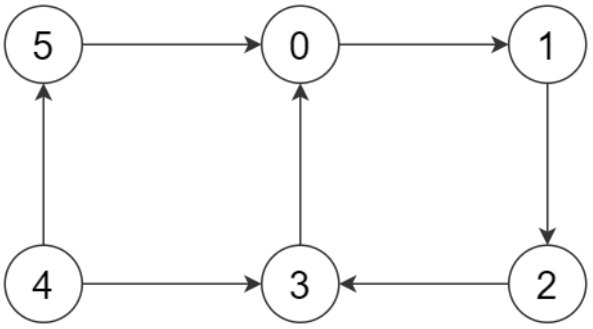
0 (si no lo es)

Electiva 2: Encontrar el vértice raíz de un grafo

Un vértice raíz de un gráfico dirigido es un vértice **u** con un camino dirigido de **u** hacia **v** para cada par de vértices **(u, v)** en el grafo. En otras palabras, todos los demás vértices del gráfico se pueden alcanzar desde el vértice raíz.

Un grafo puede tener múltiples vértices de raíz. Por ejemplo, cada vértice en un componente fuertemente conectado es un vértice raíz. En tales casos, la solución debería devolver cualquiera de ellos. Si el grafo no tiene vértices de raíz, la solución debería devolver -1. Presentar su solución en el archivo Pregunta2.py

El vértice raíz es 4 porque tiene una ruta a cada otro vértice en el siguiente gráfico:



Datos de entrada del programa:

Lista de aristas del grafo

aristas = [(0, 1), (1, 2), (2, 3), (3, 0), (4, 3), (4, 5), (5, 0)]

Número total de nodos en el grafo (0 a 5)

n = 6

Salida del programa:

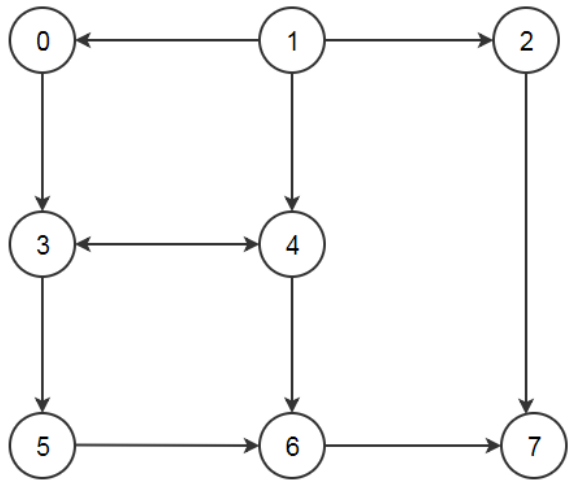
4

(-1 si no encuentra un nodo raíz).

Electiva 3: Encuentra el camino entre los vértices dados en un grafo dirigido

Dado un grafo dirigido y dos vértices (por ejemplo, vértice de origen y de destino), determine si el vértice de destino es accesible desde el vértice de origen o no. Si existe una ruta desde el vértice de origen hasta el vértice de destino, imprímala. Implemente su respuesta en Pregunta3.py

Por ejemplo, existen dos caminos [0—3—4—6—7] y [0—3—5—6—7] de vértice 0 a vértice 7 en el siguiente grafo. Por el contrario, no hay camino desde el vértice 7 a ningún otro vértice.



Datos de entrada del programa:

Lista de aristas del grafo

aristas = [(0, 3), (1, 0), (1, 2), (1, 4), (2, 7), (3, 4), (3, 5), (4, 3), (4, 6), (5, 6), (6, 7)]

Número total de nodos en el grafo (etiquetados de 0 a 7)

n = 8

nodo_inicial = 0

nodo_final = 7

Salida del programa:

1 (si existe un camino desde el nodo_inicial al nodo_final)

0 (si no existe un camino desde el nodo_inicial al nodo_final)