1 Ejercicios de teoría (50 puntos) Consideraciones Generales

- Se debe realizar la entrega en archivos de formato pdf.
- No se considerarán válidas entregas en archivos comprimidos.
- Fecha límite de entrega: 15/05/2022 23:59hs.

1.1 Grafos 1 (10 puntos)

- Dado un grafo no dirigido G = (V, E), y dados vertices s y t pertenecientes a G. Dar un algoritmo eficiente para computar la cantidad de caminos mas cortos desde s a t. La longitud de un camino es igual al numero de aristas en el, y dos caminos son diferentes si el conjunto de aristas que utilizan es diferente. No se debe retornar los caminos sino solamente la cantidad total. Ayuda: Mantener alguna informacion en cada vertice.

Observación: Se debe observar la siguiente rúbrica:

Rúbrica	Puntaje
Pseudocódigo correcto	4
Mostrar tiempo	1
Algoritmo eficiente	5

1.2 Grafos 2 (5 puntos)

- Usar búsqueda en profundidad (DFS) para demostrar que si cada vertice de un grafo no dirigido G tiene al menos d >= 2 vecinos, entonces G contiene un ciclo de longitud al menos d. Dar un ejemplo.

Observación: Se debe observar la siguiente rúbrica:

Rúbrica	Puntaje
Ejemplo	1
Justificacion	4

1.3 Grafos 3 (10 puntos)

Sea G = (V, E) un grafo conexo no dirigido, y suponga T_s = (V,F) es un arbol en G creado al explorar G usando DFS empezando en el vertice s. Es decir, el vertice s es la raiz de T_s. Explicar que s tiene mas de un hijo en T_s si y solamente si eliminando s de G ocasiona que G sea separado en varios componentes desconexos. Ilustre con un ejemplo.

Observación: Se debe observar la siguiente rúbrica:

Rúbrica	Puntaje

Ejemplo	2
Demostracion en un sentido del bicondicional	4
Demostracion en el otro sentido	4

1.4 Grafos 4 (15 puntos)

Sea G=(V,E) un grafo dirigido dado en la siguiente representacion por listas de adyacencia: para cada vertice v ∈ V se tiene una lista enlazada de vecinos salientes de v. Asuma que V = {1,...,n} y sea m=|E|.

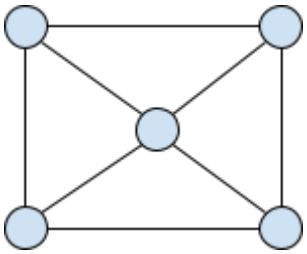
1. Escriba seudocodigo para un procedimiento que produzca una representacion de lista de adyacencia del grafo reverso (G con cada arista invertida). El procedimiento debe correr en tiempo O(m+n).
2. Escriba seudocodigo para un procedimiento que produzca la representacion con lista de adyacencia de G en donde los vecinos salientes de cada vertice se listan en orden creciente. El procedimiento debe ejecutarse en tiempo O(m+n).
3. Escriba seudocodigo para un procedimiento que verifique si G es no dirigido. Es decir para cada e ∈ E, el reverso de e tambien esta en E. El procedimiento debe correr en tiempo O(m + n).

Observación: Se debe observar la siguiente rúbrica:

Rúbrica	Puntaje
(1) correcto	5
(2) correcto	5
(3) correcto	5

1.5 Grafos 5 (10 puntos)

Asignar los pesos 1,2,3,4,1,2,3,4 a las aristas del siguiente grafo de dos maneras: A) Una asignacion tal que el arbol de recubrimiento minimo sea unico. B) Otra asignacion donde el arbol de recubrimiento minimo no sea unico. Explique ambas respuestas (por ejemplo mostrar el MST).



Observación: Se debe observar la siguiente rúbrica:

Rúbrica	Puntaje
Item A	5
Item B	5

2 Ejercicios prácticos (50 Puntos)

Consideraciones Generales

- Se debe entregar el código fuente en archivos .java separados, de acuerdo a como pide VPL.
- Fecha límite de entrega: 15/5/2022 23:59hs.

- Utilice las librerías JGraphT https://jgrapht.org/ para la resolución de los problemas propuestos. Los jar necesarios para trabajar se encuentran disponibles en: Enlace de descarga

2.1 Grafos 1 (10 puntos)

Dado un grafo $G = (V, E)$, dirigido, implemente un método Java que devuelva la transpuesta de G. Esto es, $G^T = (V, E^T)$, donde $E^T = \{(v, u) \in V \times V : (u, v) \in E\}$; por tanto, GT es G con las aristas invertidas. El método debe retornar el grafo con las aristas de manera tal que cumpla la propiedad mencionada.

- Observación: Se debe observar la siguiente rúbrica:

Rúbrica	Puntaje
Implementación correcta, sin errores de compilación y de acuerdo a restricciones	5
Provisión de datos de prueba y comentarios aclaratorios	5

2.2 Grafos 2 (15 puntos)

Dada una matriz 2D, donde un '#' representa un obstáculo y un '.' representa un espacio libre. Escriba un algoritmo Java que encuentre el área de cada área libre, además de la cantidad de áreas libres. Debe considerar que cada área libre representa un componente desconectado, además de que las celdas (i + 1, j), (i, j + 1), (i -1, j), (i, j-1), son celdas adyacentes a (i, j). Se deberá leer los datos de un archivo de texto, el cual tendrá´ el siguiente formato:

tamanhox
tamanhoy

```
# # .
. . #
# . #
```

Entonces, las siguientes entradas de prueba deberían dar las siguientes salidas:

Entrada	salida
3 3 # # . . . # # . #	2 1 3
3 5 # . # . # . # . # . # . # . #	7 1 1 1 1 1 1 1
4 4 . . # # . . # # # # . . # # . .	2 4 4

Observación: Se debe observar la siguiente rúbrica:

Rúbrica	Puntaj

	e
Implementación correcta, sin errores de compilación y de acuerdo a restricciones	10
Comentarios aclaratorios	5

2.3 Grafos 3 (10 puntos)

Existen muchas islas, las cuales están conectadas por puentes de una sola vía, es decir, si un puente conecta las islas α y β, entonces este puente solamente puede usarse para ir desde α hasta β, pero no se puede ir de β a α a través del mencionado puente. Te quedas atrapado en una isla si no puedes usar algún puente para avanzar a partir de ahí. Escriba un algoritmo Java que liste todas las islas en las que puedes quedar atrapado. El algoritmo debe calcular en tiempo lineal con respecto a la cantidad de islas y puentes. Si en algún trayecto se detecta algún camino que representa a un ciclo, se debe lanzar una excepción.

La entrada está compuesta por los siguientes elementos: En la primera línea, tres enteros que representan al número de islas, al número de puentes, y el índice de la isla inicial.

Siguientes líneas: dos enteros que representan a un puente desde una isla a otra.

Entonces, las siguientes entradas de prueba deberían dar las siguientes salidas:

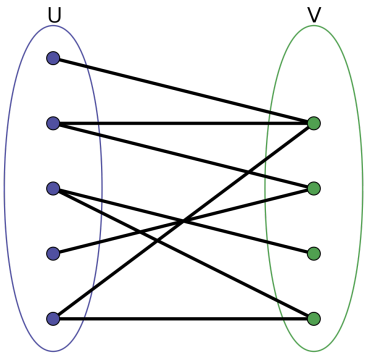
Entrada	salida
5 4 1 1 3 1 0 1 4 0 2	2 4 3

Observación: Se debe observar la siguiente rúbrica:

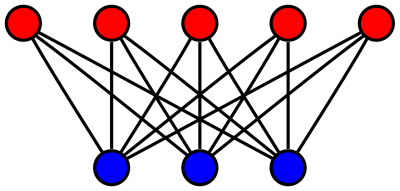
Rúbrica	Puntaje
Implementación correcta, sin errores de compilación y de acuerdo a restricciones de las operaciones solicitadas de la implementación	6
Provisión de datos de prueba	2
Comentarios aclaratorios	2

2.4 Grafos 4 (15 puntos)

Un grafo $G = (V, E)$ no dirigido se dice que es bipartito si sus vértices pueden ser particionados en dos conjuntos V_1 y V_2 tales que (u, v) E implica o bien $u \in V_1$ y $v \in V_2$ o $v \in V_1$ y $u \in V_2$. Esto es, todas las aristas 'cruzan' los conjuntos V_1 y V_2. Este es un ejemplo genérico de un grafo bipartito:



Además, un grafo bipartito se dice que es completo cuando cualquier vértice u de V_1 es adyacente a todos los vértices de V_2, y cualquier vértice v de V_2 es adyacente a todos los vértices de V_1. Este es un ejemplo genérico de un grafo completo bipartito:



Entonces, dadas las definiciones mencionadas arriba:

- Escriba un algoritmo Java que determine si un grafo es bipartito, y si es así liste los elementos de V_1 y de V_2.

- Escriba un método que determine si un grafo bipartito es además un grafo bipartito completo.

Observación: Se debe observar la siguiente rúbrica:

Rúbrica	Puntaje
Implementación correcta, sin errores de compilación y de acuerdo a restricciones de las operaciones solicitadas para determinar si el grafo es bipartito	5
Implementación correcta, sin errores de compilación y de acuerdo a restricciones de las operaciones solicitadas para determinar si el grafo bipartito es un grafo completo	4
Provisión de datos de prueba	3
Comentarios aclaratorios	3