Algoritmos y Estructuras de Datos

Examen Final
Pregrado
2020-1

Indicaciones específicas:

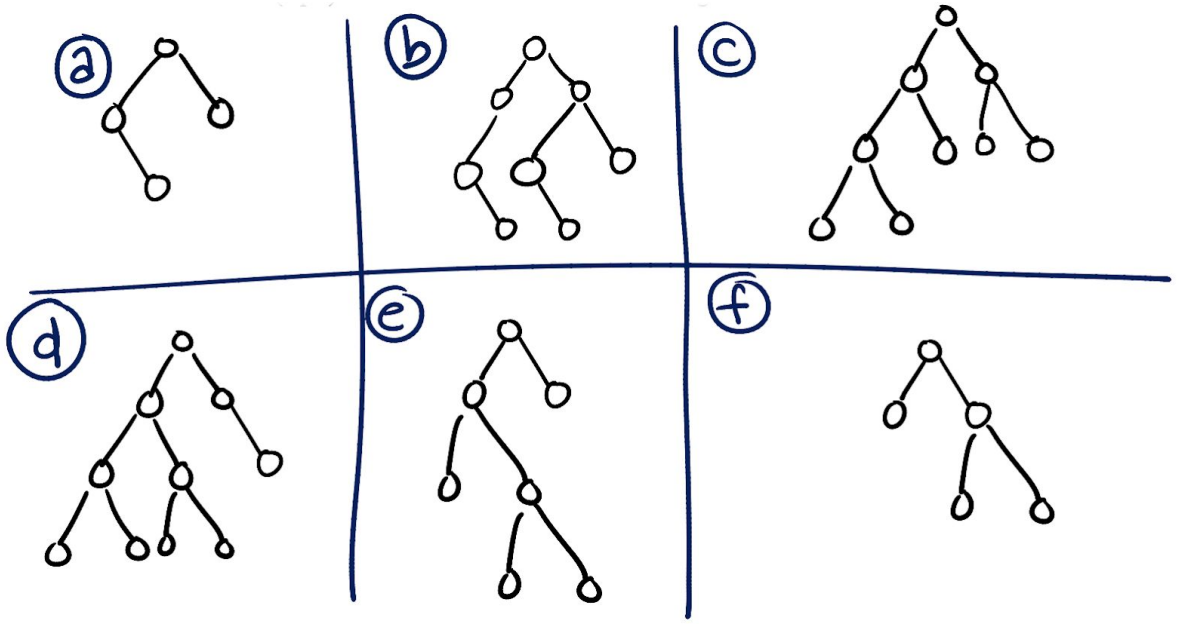
Duración: 100 minutos + 10 minutos para subir sus respuestas
Número de preguntas: 5

- Se permite el uso de calculadoras, y hojas en blanco de ayuda.
- No se permitirá el uso de notas o libros.
- Lea las preguntas cuidadosamente y responda de manera clara. Respuestas que no sean legibles o claras no tendrán ningún puntaje.
- Si no sabes la respuesta, deja el espacio en blanco y coloca "no sé la respuesta". Se concederá el 10% del puntaje. (Recuerden de todas formas resolver las preguntas que no entendieron luego de la prueba)

Pregunta 1 (7 puntos) (habilidad a)

Escriba respuestas cortas. Respuestas largas no recibirán ningún puntaje.

1. Dados los siguientes 6 árboles: (1.5 pts)



Liste las letras de todos los árboles que pertenecen a los siguientes tipos:

- Completos: c
- Llenos: c, e, f
- Perfectos:
- AVL: a, d, c, f

Notas: Es posible que ninguno de los árboles pertenezca a algún tipo, también podrían haber árboles que pertenecen a más de un tipo.

2. Considere los siguientes algoritmos y estructuras de datos: (1.5 pts)

   A. Bellman-Ford
   B. BST
   C. Dijkstra
   D. Hashing
   E. Insertion sort
   F. Patricia Trie
   G. Quicksort

   Entre los paréntesis de la derecha de cada problema en la parte inferior, complete con la letra del algoritmo o estructura de datos más apropiada de la lista de arriba. "Más apropiada" significa el algoritmo o estructura de datos que es probable a ser la base de la solución correcta más eficiente al problema entre los de la lista.

   - Ordena una lista de elementos que ya están prácticamente ordenados. (E)
   - Ordena una larga lista de elementos en orden prácticamente aleatorio. (G)
   - Encuentra el camino más corto en un grafo dirigido con pesos, algunos pesos negativos pero sin ciclos negativos. (A)
   - Encuentra el camino más corto en un grafo dirigido con pesos, sin ninguna arista negativa. (C)
   - Mantiene una tabla de símbolos que soporta operaciones de búsqueda e inserción con llaves de tipo primitivo. (D)
   - Mantiene una tabla de símbolos que soporta operaciones de búsqueda, inserción, ordenado y selección con llaves comparables. (B)
   - Mantiene una tabla de símbolos donde las llaves son palabras. (F)

3. En tus palabras explica, qué es una función hash (hashing)? En el contexto de una tabla hash y cuales son sus propiedades: (1 pt)

Una función hash es una función matemática compleja para poder hallar un index para un arreglo en función a una key en específico. Debe. cumplir con las características de ser estable (no variar), uniforme (es ideal evitar las colisiones de dos keys con el mismo index), eficiencia y seguridad (función hash como camino de ida y no de vuelta)

4. Supongamos que tenemos los números del 1 al 100 en un árbol binario de búsqueda y queremos buscar el número 44. Cuál de las siguientes opciones (puede ser más de una) podrían ser secuencias de nodos examinados? (1 pt)

   I. 50, 25, 26, 27, 40, 45 si
   II. 5, 2, 1, 10, 39, 34, 77, 63 No
   III. 9, 8, 63, 0, 4, 3, 2, 1 No
   IV. 1, 2, 3, 4, 5, 6, 7, 10 Si
   V. 50, 25, 26, 27, 40, 43, 42 No
   VI. 8, 7, 6, 5, 4, 3, 2, 1 No

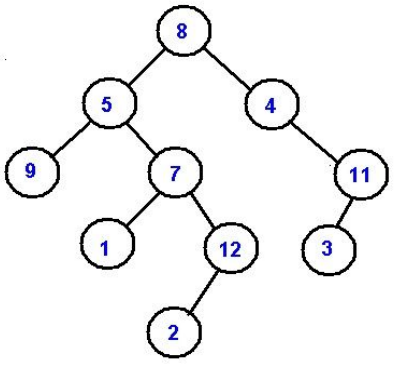
5. Explique un caso de uso para cada una de las siguientes estructuras: (2 pts)

   - Treap:
   - Red Black Tree:
   - B-Link tree: base de datos
   - Skip List: base de datos
   - Splay Tree: Cache
   - Fibonacci Heap: piority queue de dijkstra

Pregunta 2 (5 puntos) (habilidad b, c)

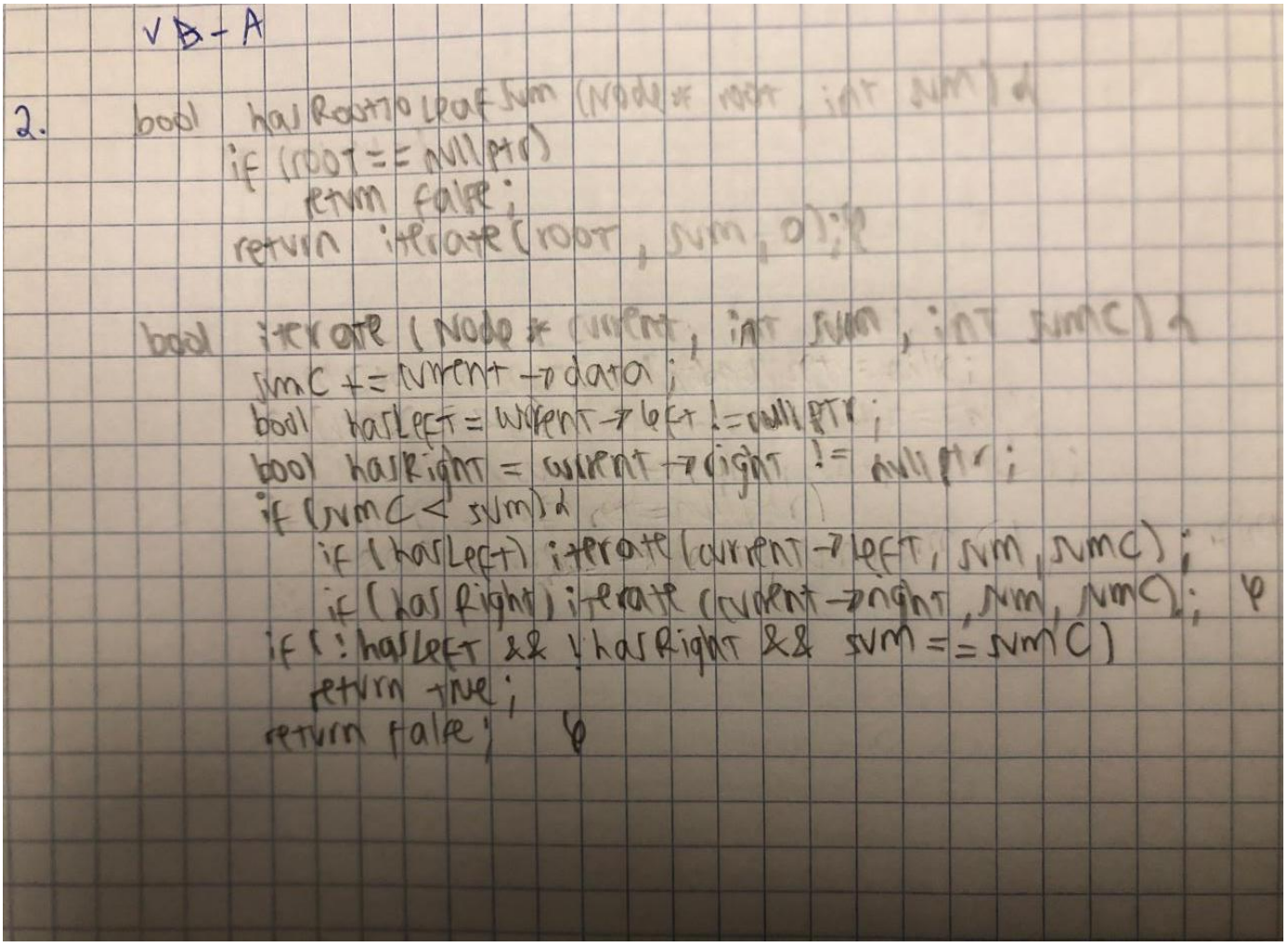
Implementa en C++. Tu respuesta será validada en correctitud, eficiencia y claridad.

A. Dado un árbol binario T y una suma S, escriba un programa para revisar si existe un camino desde el root hasta alguna hoja del árbol con una suma igual a S:



Ejemplo: Para la suma 26, existe el camino 8, 4, 11 y 3

```
struct node {
        int data;
        node* left;
        node* right;
};
```



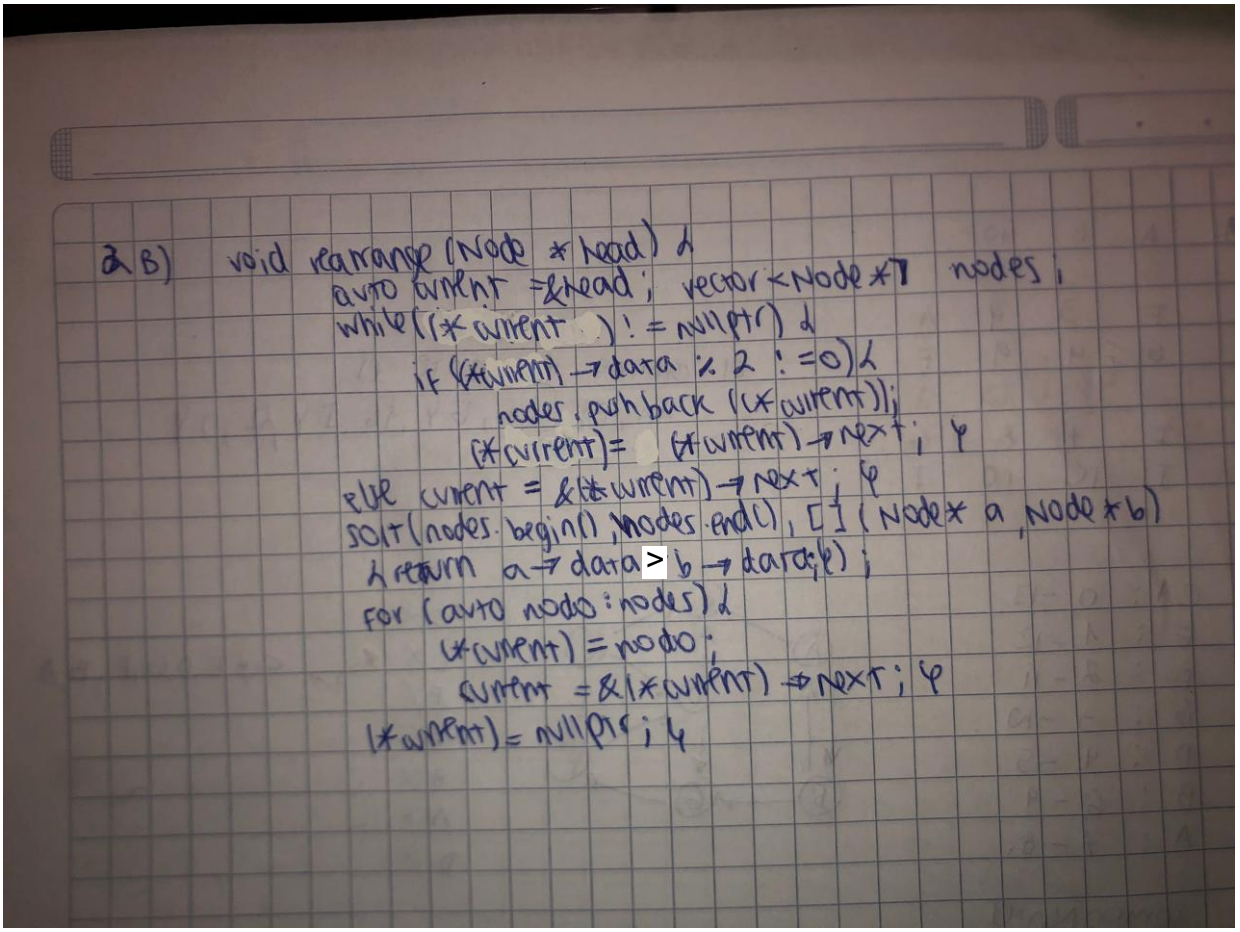
B. Reordena una lista simplemente enlazada de manera que cada nodo par sea movido al final de la lista en orden inverso.

Por ejemplo:

Input: 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> null
Output: 1 -> 3 -> 5 -> 7 -> 6 -> 4 -> 2 -> null

```
struct Node {
        int data;
        Node* next;
};

void rearrange(Node* head) {
```



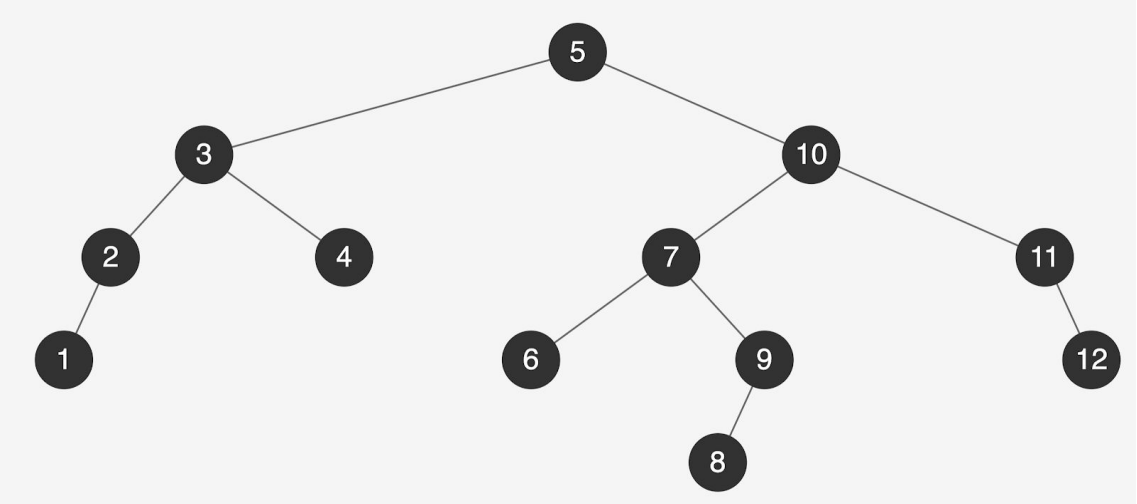
Pregunta 3 (2.5 puntos) (habilidad a, b, c)

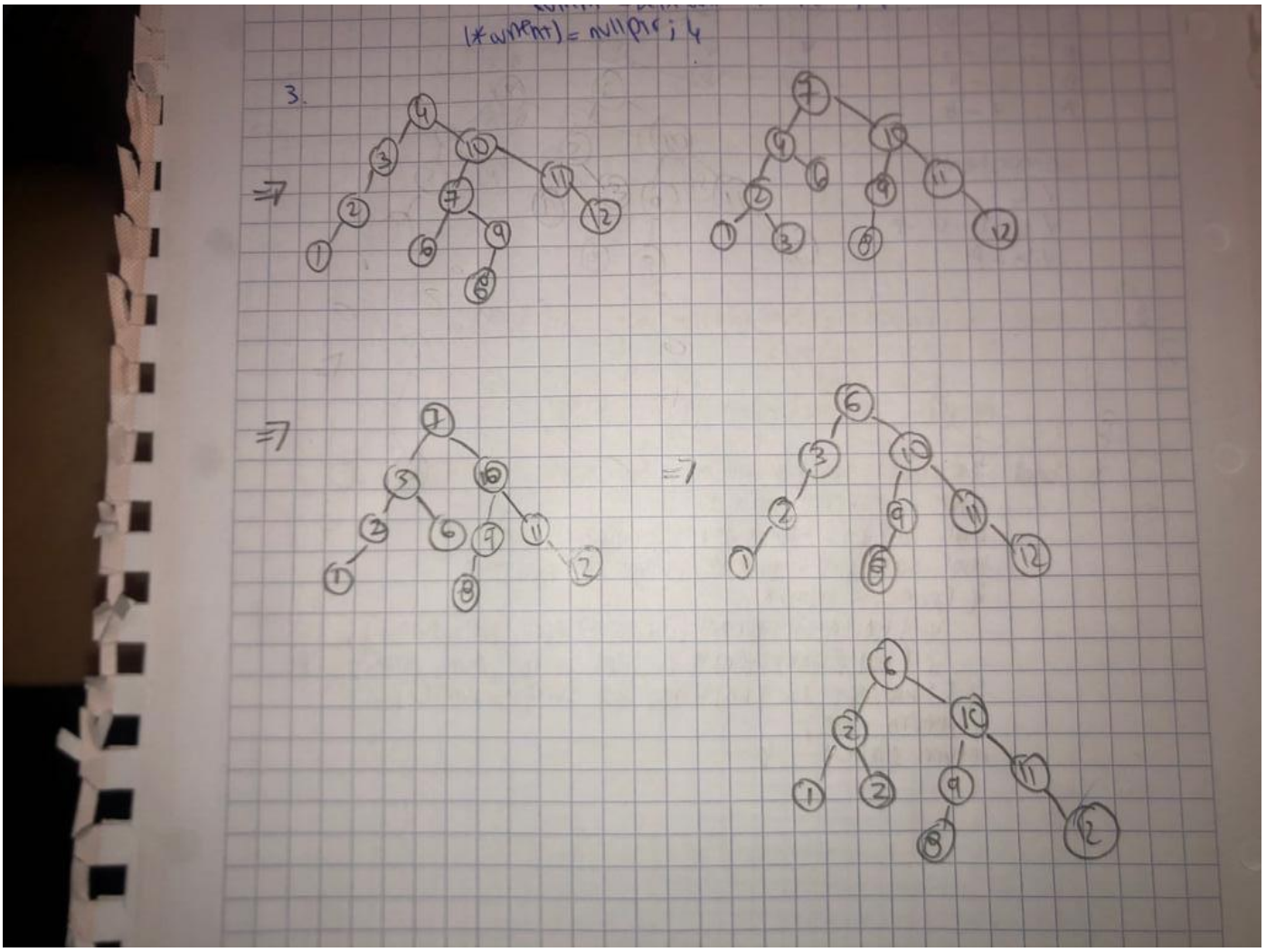
Notas:

- Tener en cuenta que para cualquier operación de borrado, se tomará el caso de reemplazar con el elemento anterior. Osea, *el de la izquierda más a la derecha*. Caso que se decida reemplazar con el siguiente (lo cual no lo hace incorrecto) se tendrá una penalidad sobre el puntaje.

- Solo se necesita mostrar el resultado final de cada una de las operaciones, sin embargo cualquier árbol parcial se tendrá en cuenta para puntaje por procedimiento.

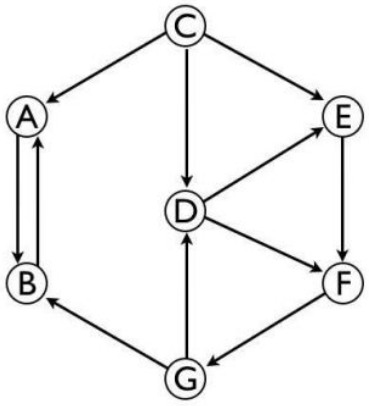
Muestre los árboles resultantes de remover el **5**, luego el **4** y finalmente el **7** del siguiente árbol AVL:



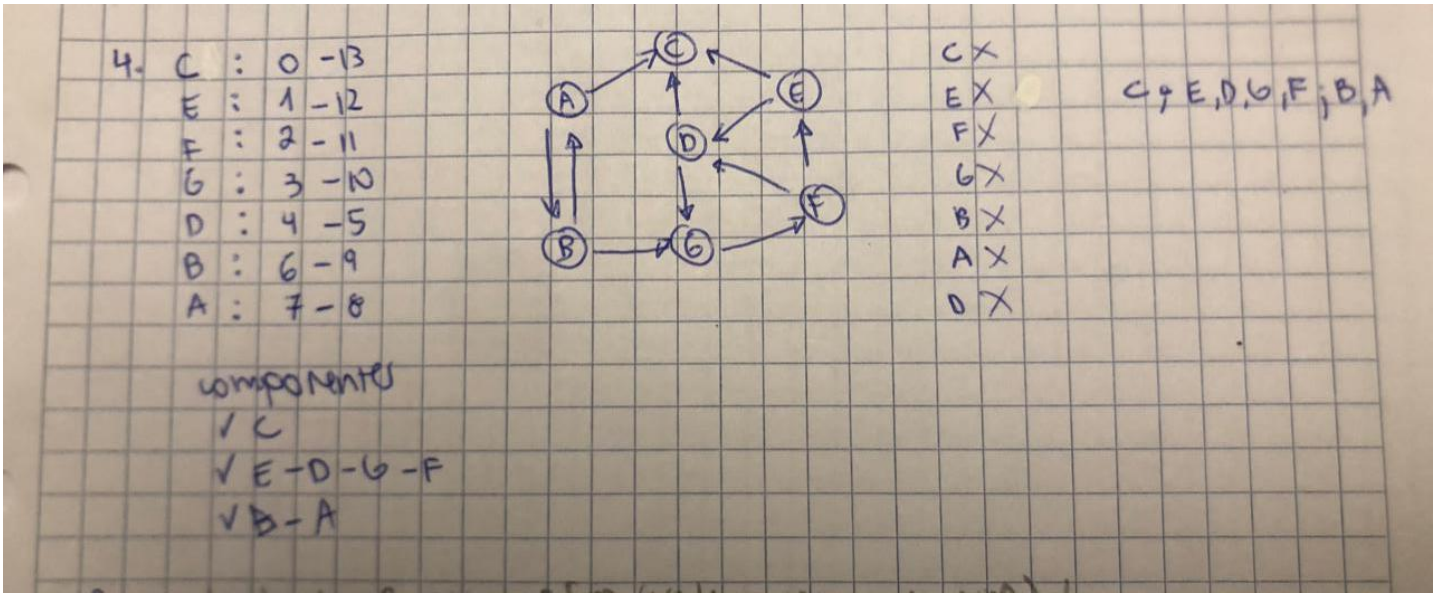


Pregunta 4 (2.5 puntos) (habilidad a, b)

Considere el siguiente grafo dirigido no ponderado:

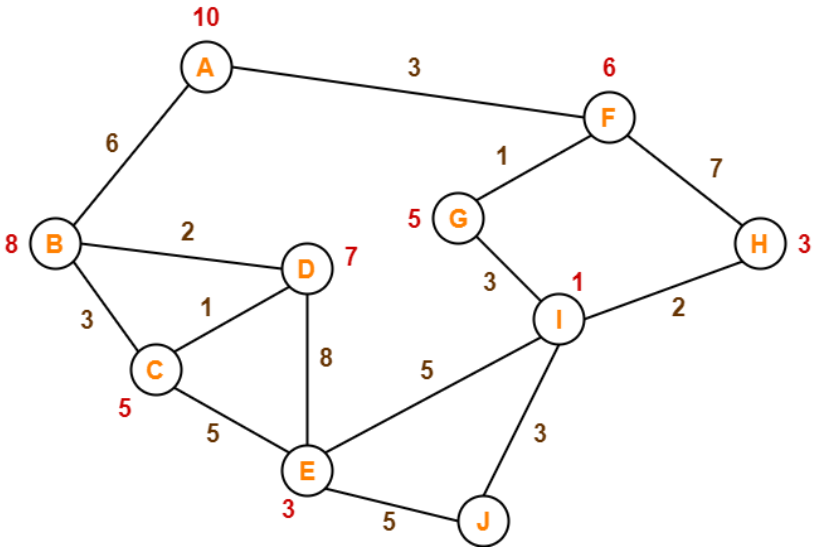


- **Encuentre los componentes fuertemente conexos y listelos con los vértices que los conforman.**
- **Muestre el proceso completo para obtener los componentes, no se aceptarán respuestas sin procedimiento.**



Pregunta 5 (3 punto) (habilidad a, b)

Considere el siguiente grafo no dirigido:



- **Encuentre el camino más eficiente para llegar del vértice 'A' al vértice 'J' utilizando el algoritmo de A*.**
- **Los números en las aristas representan la distancia entre vértices.**
- **Los números en los nodos representan el valor de la heurística.**
- **Muestre paso por paso la ejecución del algoritmo y el camino resultante.**