7541 - 20201C - 1er Final

Version 1

1. Calcular la complejidad computacional del siguiente algoritmo

```
int sum = 0;
for (int n = N; n > 0; n /= 2)
  for (int i = 0; i < n; i++)
    sum++;
```

2. Encuentre, si es posible, una solución para la siguiente ecuación de recurrencia

$T(n) = 2\ T(n/3) + n,\ T(1) = 1$

Version 2

1. Calcular la complejidad computacional del siguiente algoritmo

```
int sum = 0;
for (int i = 1; i < N; i *= 2)
  for(int j = 0; j < i; j++)
    sum++;
```

2. Encuentre, si es posible, una solución para la siguiente ecuación de recurrencia

$T(n) = 2\ T(n/2) + 1,\ T(1) = 1$

Version 3

1. Calcular la complejidad computacional del siguiente algoritmo

```
int sum = 0;
for (int i = 1; i < N; i *= 2)
  for (int j = 0; j < N; j++)
    sum++;
```

2. Encuentre, si es posible, una solución para la siguiente ecuación de recurrencia

$T(n) = 2\ T(n/2) + n^2,\ T(1) = 1$

7541 - 20201C - 1er Final

Version 1

Defina una estructura para implementar una **lista doblemente enlazada**. Muestre la estructura e implemente las siguientes primitivas para la lista doblemente enlazada:

- lista_crear
- lista_insertar_en_posicion

Version 2

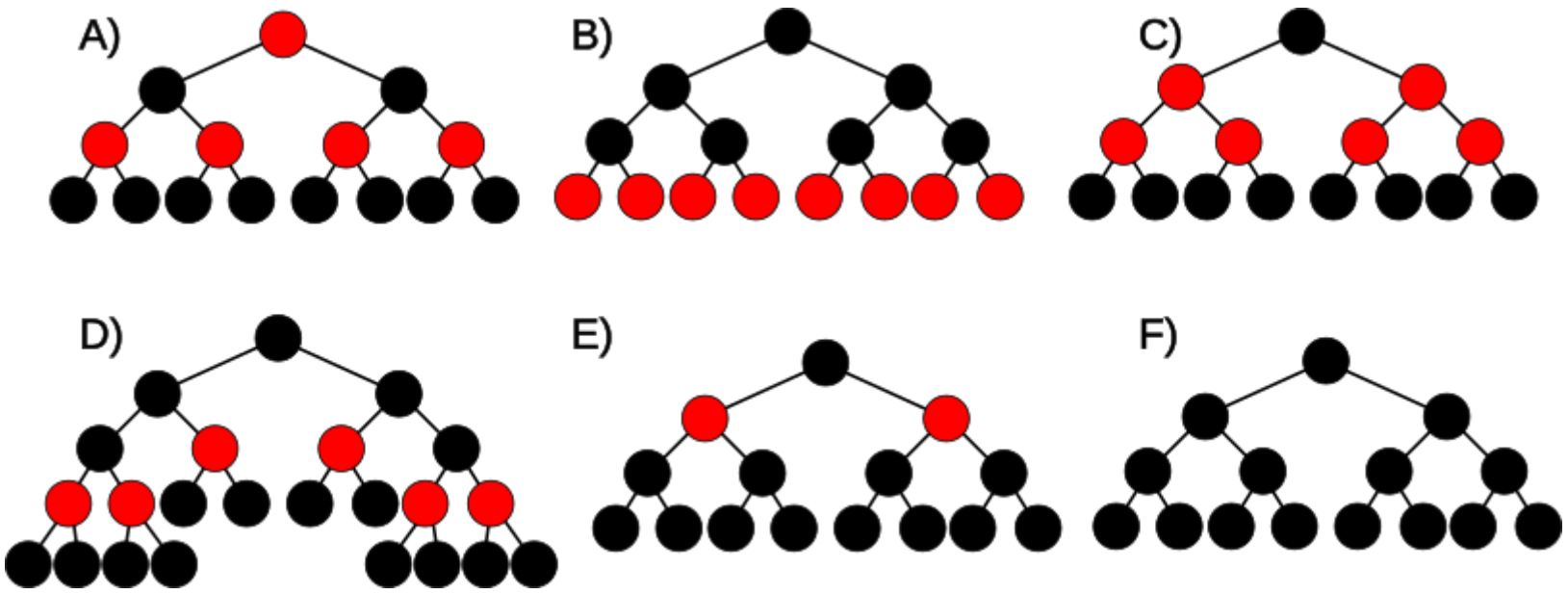
Defina una estructura para implementar una **lista doblemente enlazada**. Muestre la estructura e implemente las siguientes primitivas para la lista doblemente enlazada:

- lista_crear
- lista_eliminar_de_posicion

7541 - 20201C - 1er Final

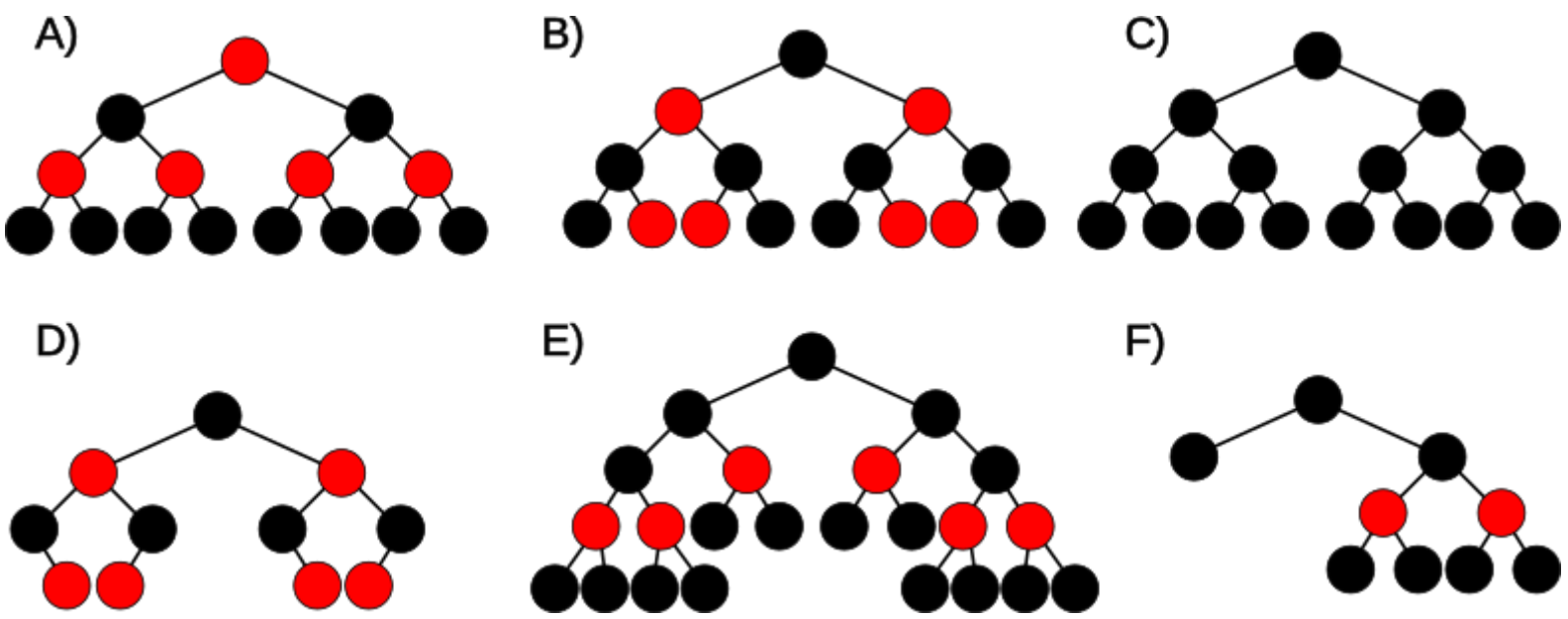
Version 1

¿Cuáles de los siguientes árboles **rojo-negro** son válidos para una determinada implementación? Justifique cada caso.



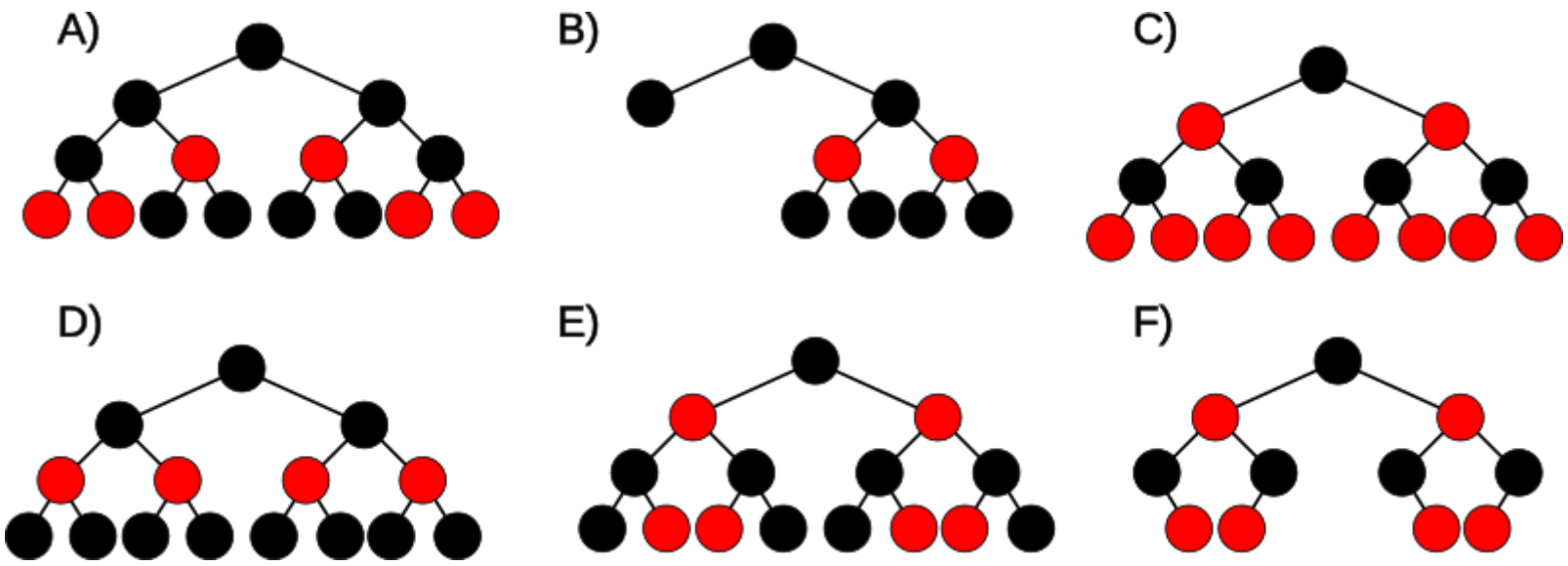
Version 2

¿Cuáles de los siguientes árboles **rojo-negro** son válidos para una determinada implementación? Justifique cada caso.



Version 3

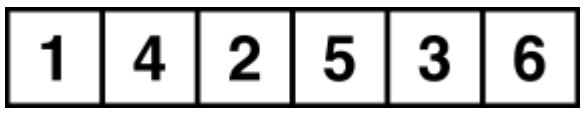
¿Cuáles de los siguientes árboles **rojo-negro** son válidos para una determinada implementación? Justifique cada caso.



7541 - 20201C - 1er Final

Version 1

Justifique si el siguiente vector de enteros representa un heap (maximal o minimal). ¿Qué condición debe cumplir para que lo sea?

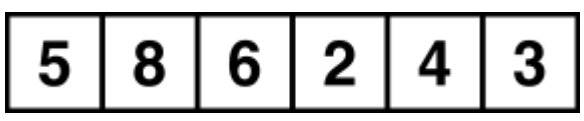


Explique (texto o código claro) cómo reordenarlo (inplace, sin requerir reservar memoria extra) para que cumpla la condición. El algoritmo descripto debe funcionar para cualquier vector de enteros.

Aplique el algoritmo al vector dado para obtener un **heap maximal** explicando cada iteración del mismo. En cada paso de algoritmo muestre como queda el array (marque con un asterisco los elementos que se movieron en el array en ese paso).

Version 2

Justifique si el siguiente vector de enteros representa un heap (maximal o minimal). ¿Qué condición debe cumplir para que lo sea?

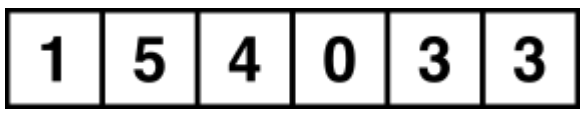


Explique (texto o código claro) cómo reordenarlo (inplace, sin requerir reservar memoria extra) para que cumpla la condición. El algoritmo descripto debe funcionar para cualquier vector de enteros.

Aplique el algoritmo al vector dado para obtener un **heap minimal** explicando cada iteración del mismo. En cada paso de algoritmo muestre como queda el array (marque con un asterisco los elementos que se movieron en el array en ese paso).

Version 3

Justifique si el siguiente vector de enteros representa un heap (maximal o minimal). ¿Qué condición debe cumplir para que lo sea?

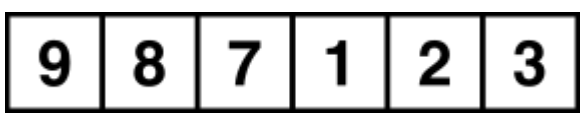


Explique (texto o código claro) cómo reordenarlo (inplace, sin requerir reservar memoria extra) para que cumpla la condición. El algoritmo descripto debe funcionar para cualquier vector de enteros.

Aplique el algoritmo al vector dado para obtener un **heap maximal** explicando cada iteración del mismo. En cada paso de algoritmo muestre como queda el array (marque con un asterisco los elementos que se movieron en el array en ese paso).

Version 4

Justifique si el siguiente vector de enteros representa un heap (maximal o minimal). ¿Qué condición debe cumplir para que lo sea?



Explique (texto o código claro) cómo reordenarlo (inplace, sin requerir reservar memoria extra) para que cumpla la condición. El algoritmo descripto debe funcionar para cualquier vector de enteros.

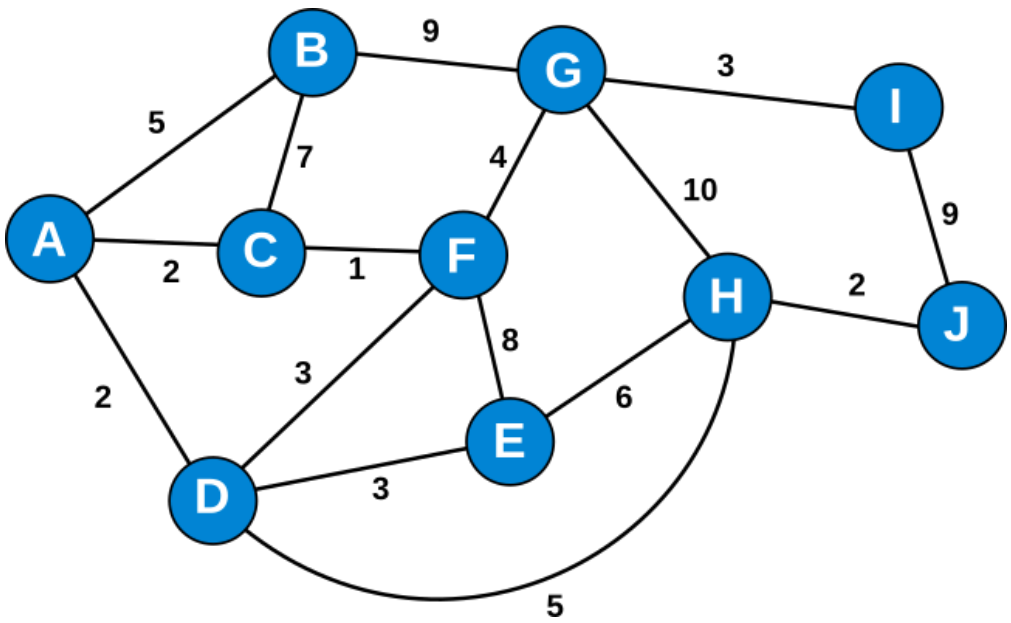
Aplique el algoritmo al vector dado para obtener un **heap minimal** explicando cada iteración del mismo. En cada paso de algoritmo muestre como queda el array (marque con un asterisco los elementos que se movieron en el array en ese paso).

7541 - 20201C - 1er Final

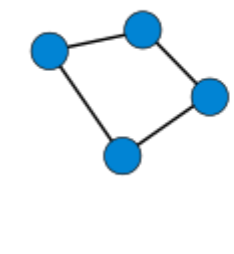
EJERCICIO 1 | EJERCICIO 2 | EJERCICIO 3 | EJERCICIO 4 | EJERCICIO 5

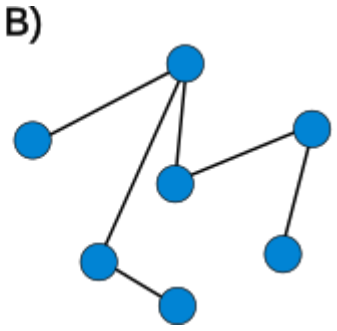
Version 1

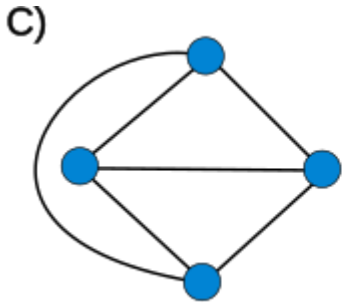
1. Aplique el **algoritmo de Prim** en el siguiente grafo. Explique cada paso del algoritmo.

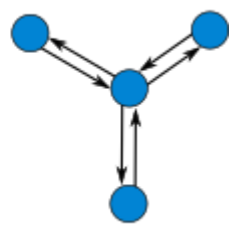


2. Clasifique los siguentes grafos en: Dirigido / no dirigido, simple / no simple, completo / incompleto, cíclico / acíclico, conexo / fuertemente conexo / débilmente conexo / no conexo, árbol. Justifique.

A) 

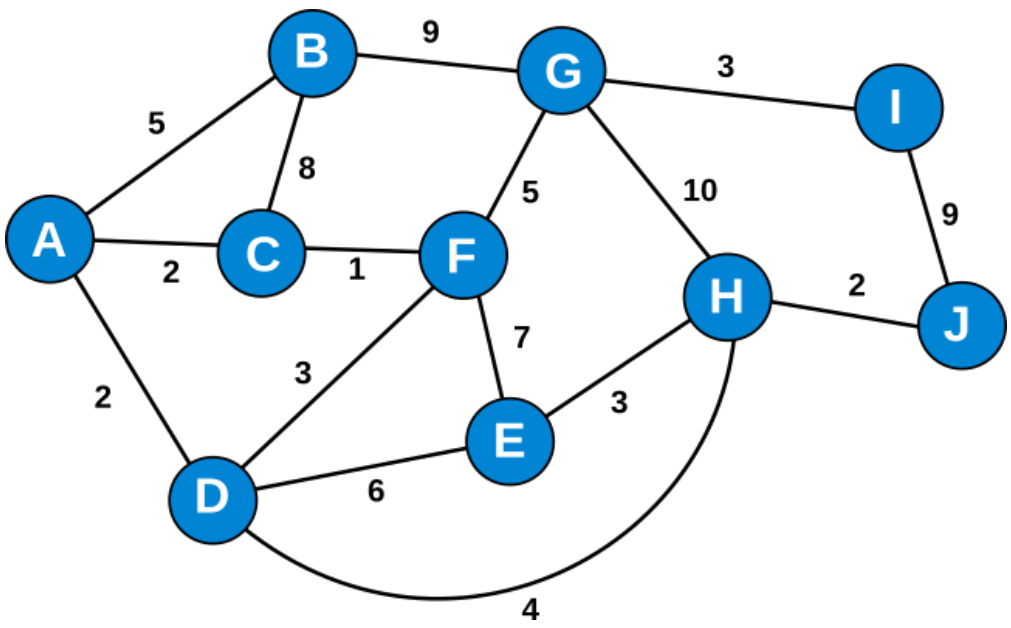
B) 

C) 

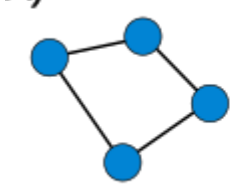
D) 

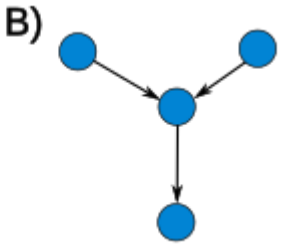
Version 2

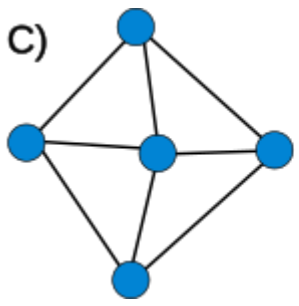
1. Aplique el **algoritmo de Kruskal** en el siguiente grafo. Explique cada paso del algoritmo.

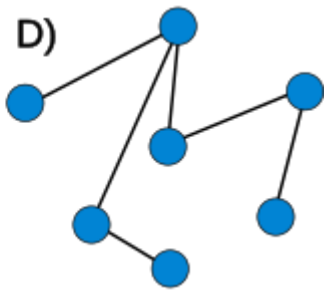


2. Clasifique los siguentes grafos en: Dirigido / no dirigido, simple / no simple, completo / incompleto, cíclico / acíclico, conexo / fuertemente conexo / débilmente conexo / no conexo, árbol. Justifique.

A) 

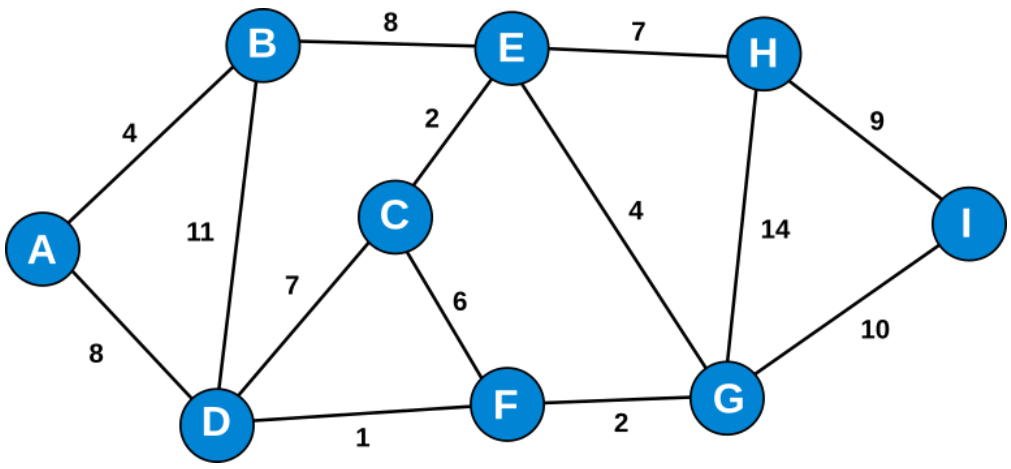
B) 

C) 

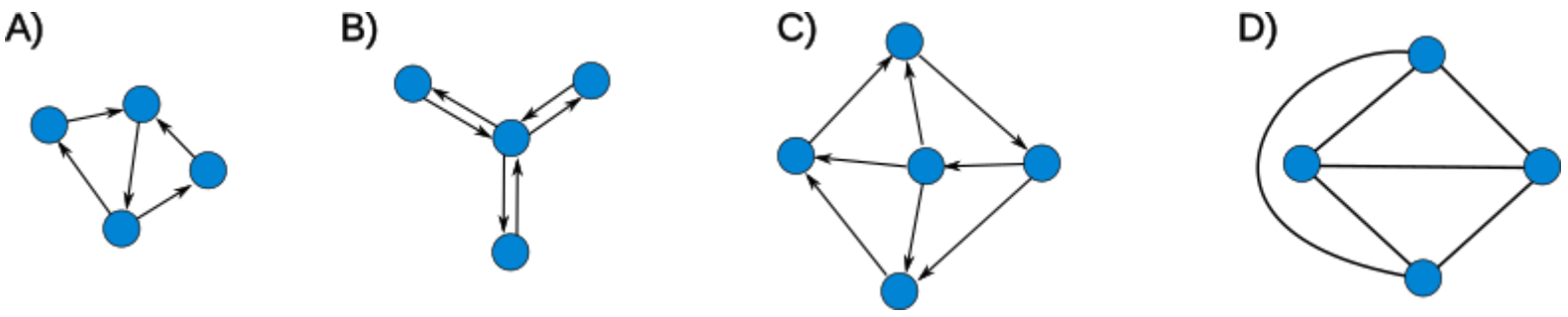
D) 

Version 3

1. Aplique el **algoritmo de Dijkstra** en el siguiente grafo. Explique cada paso del algoritmo.

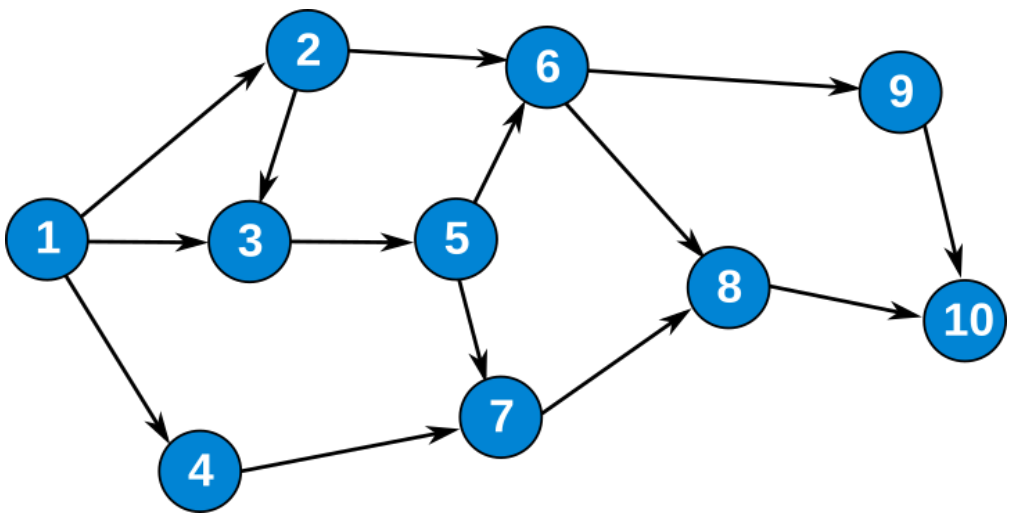


2. Clasifique los siguentes grafos en: Dirigido / no dirigido, simple / no simple, completo / incompleto, cíclico / acíclico, conexo / fuertemente conexo / débilmente conexo / no conexo, árbol. Justifique.



Version 4

1. Aplique el algoritmo de **ordenamiento topológico** en el siguiente grafo. Explique cada paso del algoritmo.



2. Clasifique los siguentes grafos en: Dirigido / no dirigido, simple / no simple, completo / incompleto, cíclico / acíclico, conexo / fuertemente conexo / débilmente conexo / no conexo, árbol. Justifique.