ACT\_5.1\_OPERACIONES\_SOBRE\_CONJUNTOS

Casos de prueba:

Antes de mostrar los casos de prueba cabe resaltar que se utilizaron dos tablas con una longitud de 10 valores para llevar a cabo la técnica escogida de hashing y por su parte, para implementar las estrategias de control de colisiones en estas mismas.

Se implementaron dos principales estrategias de control de colisiones: cuadrática y de encadenamiento.

1) Que los datos ingresados con la técnica de hashing en la tabla de 10 espacios, se acomoden acorde a la estrategia de control de colisiones utilizada.

Estrategia: Cuadrática.

Input:

Texto

Descripción generada automáticamente

Output:

Texto

Descripción generada automáticamente

2) Que los datos ingresados con la técnica de hashing en la tabla de 10 espacios, se acomoden acorde a la estrategia de control de colisiones utilizada.

Estrategia: Encadenamiento.

Input:

Texto

Descripción generada automáticamente

Output:

Texto

Descripción generada automáticamente

3) Que los datos ingresados con la técnica de hashing en la tabla de 10 espacios, se acomoden acorde a la estrategia de control de colisiones utilizada.

Estrategia: Cuadrática.

Input:



Output:

Texto

Descripción generada automáticamente

4) Que los datos ingresados con la técnica de hashing en la tabla de 10 espacios, se acomoden acorde a la estrategia de control de colisiones utilizada.

Estrategia: Encadenamiento.

Input:



Output:

Texto

Descripción generada automáticamente

Complejidades Computacionales:

1) disp\_tab ():

Texto

Descripción generada automáticamente

La complejidad computacional de esta función es de O (n2), ya que, contamos con dos bucles for anidados y por su parte siempre que queramos imprimir nuestra tabla debemos de recorrer todo el vector y a su vez todas las listas que conforman a este mismo.

2) quadratic ():

Texto

Descripción generada automáticamente

La complejidad computacional de esta función es de O (n), ya que, en el peor de los casos el bucle while tiene que recorrer todos los elementos de la tabla. Es decir, que en el peor de los casos se tendrán que recorrer los n elementos de la tabla.

3) chain ():

Texto

Descripción generada automáticamente

La complejidad computacional de esta función es de O (1), debido a que, simplemente contamos con líneas de código sencillas que solo llevan a cabo funciones independientes. No contamos con ningún bucle ni ninguna otra cosa que pueda afectar la complejidad de nuestra función.

4) hash\_tech():

Texto

Descripción generada automáticamente

La complejidad computacional de esta función es de O (1), debido a que, simplemente contamos con líneas de código sencillas que solo llevan a cabo funciones independientes. No contamos con ningún bucle ni ninguna otra cosa que pueda afectar la complejidad de nuestra función.