PARTE 1

Ejercicio 1

Ejemplificar que pasa cuando insertamos las llaves 5, 28, 19, 15, 20, 33, 12, 17, 10 en un **HashTable** con la colisión resulta por el método de chaining. Permita que la tabla tenga 9 slots y la función de hash:

$$H(k) = k \bmod 9 \qquad (1)$$

Lo que pasaría en este ejemplo es:

	Key	Key	Key
0			
1	28	19	10
2	20		
3	12		
4			
5	5		
6	15	33	
7			
8	17		

Ejercicio 2

A partir de una definición de diccionario como la siguiente:

dictionary = Array(m,0) # una sugerencia de implementacion,se puede usar una lista de python

Crear un módulo de nombre **dictionary.py** que **implemente** las siguientes especificaciones de las operaciones elementales para el **TAD diccionario** .

Nota: puede **dictionary** puede ser redefinido para lidiar con las colisiones por encadenamiento

insert(D, key, value)

Descripción: Inserta un key en una posición determinada por la función de hash (1) en el diccionario (dictionary). Resolver colisiones por encadenamiento. En caso de keys duplicados se anexan a la lista.
Entrada: el diccionario sobre el cual se quiere realizar la inserción

y el valor del key a insertar.
Salida: Devuelve D

search(D,key)
Descripción: Busca un key en el diccionario
Entrada: El diccionario sobre el cual se quiere realizar la búsqueda (dictionary) y el valor del key a buscar.
Salida: Devuelve el value de la key. Devuelve **None** si el key no se encuentra.

delete(D,key)
Descripción: Elimina un key en la posición determinada por la función de hash (1) del diccionario (dictionary)
Poscondición: Se debe marcar como None el **key** a eliminar.
Entrada: El diccionario sobre el se quiere realizar la eliminación y el valor del key que se va a eliminar.
Salida: Devuelve **D**

PARTE 2

Ejercicio 3

Considerar una tabla hash de tamaño m = 1000 y una función de hash correspondiente al método de la multiplicación donde A = (sqrt(5)-1)/2). Calcular las ubicaciones para las claves 61, 62, 63, 64 y 65.

61 -> 700
62 -> 318
63 -> 936
64 -> 554
65 -> 172

Ejercicio 4

Implemente un algoritmo lo más eficiente posible que devuelva **True** o **False** a la siguiente proposición: dado dos strings $s_1...s_k$ y $p_1...p_k$, se quiere encontrar si los caracteres de $p_1...p_k$ corresponden a una permutación de $s_1...s_k$. Justificar el costo en tiempo de la solución propuesta.

Ejemplo 1:
Entrada: S = ‘hola’ , P = ‘ahlo’
Salida: True, ya que P es una permutación de S

Ejemplo 2:

Entrada: S = ‘hola’ , P = ‘ahdo’
Salida: Falso, ya que P tiene al carácter ‘d’ que no se encuentra en S por lo que no es una permutación de S

Ejercicio 5

Implemente un algoritmo que devuelva True si la lista que recibe de entrada tiene todos sus elementos únicos, y Falso en caso contrario. Justificar el costo en tiempo de la solución propuesta.

Ejemplo 1:
Entrada: L = [1,5,12,1,2]
Salida: Falso, L no tiene todos sus elementos únicos, el 1 se repite en la 1ra y 4ta posición

Ejercicio 6

Los nuevos códigos postales argentinos tienen la forma cddddccc, donde c indica un carácter (A - Z) y d indica un dígito 0, . . . , 9. Por ejemplo, C1024CWN es el código postal que representa a la calle XXXX a la altura 1024 en la Ciudad de Mendoza. Encontrar e implementar una función de hash apropiada para los códigos postales argentinos.

Ejercicio 7

Implemente un algoritmo para realizar la compresión básica de cadenas utilizando el recuento de caracteres repetidos. Por ejemplo, la cadena ‘*aabccccccaaa’* se convertiría en ‘a2blc5a3’. Si la cadena "comprimida" no se vuelve más pequeña que la cadena original, su método debería devolver la cadena original. Puedes asumir que la cadena sólo tiene letras mayúsculas y minúsculas (a - z, A - Z). Justificar el costo en tiempo de la solución propuesta.

Mi solución es O(n), ya que tengo que iterar por cada caracter del string, realizando operaciones O(1) dentro de ese for loop.

Ejercicio 8

Se requiere encontrar la primera ocurrencia de un string $p_1...p_k$ en uno más largo $a_1...a_L$. Implementar esta estrategia de la forma más eficiente posible con un costo computacional menor a $O(K*L)$ (solución por fuerza bruta). Justificar el coste en tiempo de la solución propuesta.

Ejemplo 1:
Entrada: S = ‘abracadabra’ , P = ‘cada’
Salida: 4, índice de la primera ocurrencia de P dentro de S (abracadabra)

Mi algoritmo tiene una complejidad de O(n), siendo n la longitud de S.
Ya que lo que hago es ver el hash de P, y luego, buscar eso hash, tomando substrings de S, de a longitud len(P).
Luego, recorro todo S una vez y el resto de operaciones son O(1)

Ejercicio 9

Considerar los conjuntos de enteros $S = \{s1, \ldots, sn\}$ y $T = \{t1, \ldots, tm\}$. Implemente un algoritmo que utilice una tabla de hash para determinar si $S \subseteq T$ (S subconjunto de T). ¿Cuál es la complejidad temporal del caso promedio del algoritmo propuesto?

O(s+t)
Ya que, lo que hago, es insertar cada elemento de T en un diccionario (operación O(1)), y luego, busco cada elemento de s en el diccionario creado (siendo la búsqueda O(1) en caso promedio, asumiendo que no hay colisiones).

Parte 3

Ejercicio 10

Considerar la inserción de las siguientes llaves: 10; 22; 31; 4; 15; 28; 17; 88; 59 en una tabla hash de longitud m = 11 utilizando direccionamiento abierto con una función de hash *h'(k)* = *k*. Mostrar el resultado de insertar estas llaves utilizando:

1. Linear probing
   h(k,i) = (k+i)mod 11

0	22
1	88
2	
3	
4	4
5	15
6	28
7	17
8	59

9	31
10	10

2. Quadratic probing con *c1 = 1* y c2 = 3
   h(k,i) = (k + i + 3(i^2)) mod 11

0	22
1	
2	88
3	17
4	4
5	
6	28
7	59
8	15
9	31
10	10

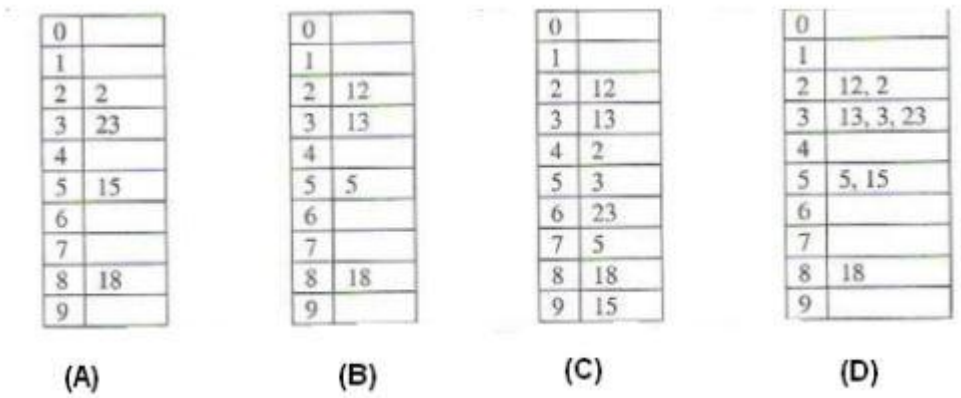
3. Double hashing con *h1(k) = k* y *h2(k) = 1 +(k mod (m - 1))*
   *h(k,i) = (k+ i * (1 + k mod(10)))mod11*

0	*22*
1	*17*
2	
3	*59*
4	*4*
5	*15*
6	*28*
7	*88*
8	
9	*31*

10	*10*

Ejercicio 12

Las llaves 12, 18, 13, 2, 3, 23, 5 y 15 se insertan en una tabla hash inicialmente vacía de longitud 10 utilizando direccionamiento abierto con función hash h(k) = k mod 10 y exploración lineal (linear probing). ¿Cuál es la tabla hash resultante? Justifique.



0	
1	
2	2
3	23
4	
5	15
6	
7	
8	18
9	

(A)

0	
1	
2	12
3	13
4	
5	5
6	
7	
8	18
9	

(B)

0	
1	
2	12
3	13
4	2
5	3
6	23
7	5
8	18
9	15

(C)

0	
1	
2	12, 2
3	13, 3, 23
4	
5	5, 15
6	
7	
8	18
9	

(D)

C, ya que al insertar 2, tenemos como h(2,0) = 2, que está ocupado por el 12, h(2,1)= 3, que está ocupado por el 13, y lo logramos insertar en la posición 4

Ejercicio 13

Una tabla hash de longitud 10 utiliza direccionamiento abierto con función hash h(k)=k mod 10, y exploración lineal (linear probing). Después de insertar 6 valores en una tabla hash vacía, la tabla es como se muestra a continuación.

0	
1	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

¿Cuál de las siguientes opciones da un posible orden en el que las llaves podrían haber sido insertadas en la tabla? Justifique
(A) 46, 42, 34, 52, 23, 33 No es esta opción, ya que insertando en este orden, 52 va en la posición 3
(B) 34, 42, 23, 52, 33, 46 No es, ya que al insertar 33,debería estar insertado en la posición 6
(C) 46, 34, 42, 23, 52, 33
(D) 42, 46, 33, 23, 34, 52 No es, ya que el 23 se insertará en la posición 4

A tener en cuenta:

1. **Usen lápiz y papel primero**
2. **~~No se puede utilizar otra Biblioteca mas allá de algo1.py y las bibliotecas desarrolladas durante Algoritmos y Estructuras de Datos I.~~**