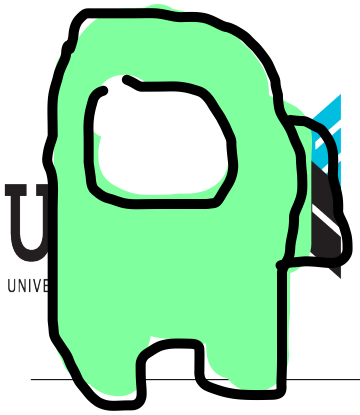

Intro. a Ciencia de la Computación
BÚSQUEDA
Pregrado
2019-II
Ciencia de la Computación
Laboratorio

Las siguientes preguntas se recogen de prácticas pasadas y están acompañadas de un indicador de dificultad que va desde el 1 al 5.

1. (Nivel 1) Tenemos un programa que contiene una lista cuyos elementos son dicccionarios de la siguiente forma:

   - actor, nacionalidad, película

   Ejemplo de una posible lista:

```
actores=[{"actor": "Felipe","nacionalidad":"peruana","pelicula":"loco␣por␣
    mary"},
        {"actor": "Gorilon","nacionalidad":"brasilera","pelicula":"
            asesinos␣locos"},
        {"actor": "Grecia","nacionalidad":"cubana","pelicula":"salvando␣al
            ␣soldado␣Jacinto"},
        {"actor": "Teresita","nacionalidad":"venezolana","pelicula":"tres␣
            y␣dos␣son␣ocho"}]
```

   Se pide lo siguiente:

   - Pedir el usuario que nos diga el actor a buscar.
   - Encontrar el actor usando un algoritmo de búsqueda binaria y mostrar todos sus datos.

   Algunos ejemplos de diálogo de este programa serían:

```
Ingrese nombre de actor a buscar: Felipe

Resultado:
nombre: Felipe
nacionalidad: peruana
película: loco por mary
```

```
Ingrese nombre de actor a buscar: Teresita

Resultado:
nombre: Teresita
nacionalidad: venezolana
película: tres y dos son ocho
```

2. (Nivel 1) Elabore un programa que permita al usuario ingresar un número entero positivo N. Valide que sea un número positivo mayor o igual a 5. Luego ingresará N números enteros a una lista ordenada. El programa imprime la lista. A continuación, el usuario ingresa un número a buscar en la lista. Asuma que el usuario ingresa un número entero positivo.

   Si encuentra el número, nos informa que lo encontró y si no lo encontró, crea e imprima una nueva lista con la misma cantidad de elementos que la otra lista, pero donde cada elemento es igual a cero.

   Utilice el algoritmo de *Búsqueda Binaria* para encontrar el número buscado.

   Algunos ejemplos de diálogo:

```
Ingrese un número entero: -3
Ingrese un número entero: -343
Ingrese un número entero: 5
Ingrese los números:
11
44
66
87
90
Lista: [11, 44, 66, 87, 90]
Ingrese número a buscar: 33
No se encontró 33.  -> [0, 0, 0, 0, 0]
```

```
Ingrese un número entero: -4
Ingrese un número entero: 6
23
78
134
345
444
Lista: [23, 78, 134, 345, 444]
Ingrese número a buscar: 345
Se encontró 345 en la posición 3
```

3. (Nivel 1) Implemente una función, utlizando el algoritmo de búsqueda binaria, para buscar datos que están en una lista de diccionarios sobre películas, con la siguiente estructura: título y año. La función debe recibir un parámetro (el año a buscar) y debe devolver el título de la película.

   Un ejemplo de una lista sería:

```
peliculas = [
        {'titulo': 'Shrek', 'año': 2001},
        {'titulo': 'El viaje de Chihiro', 'año': 2002},
        {'titulo': 'Buscando a Nemo', 'año': 2003},
```

```
        {'titulo': 'Los Increíbles', 'año': 2004},
        {'titulo': 'Wallace y Gromit', 'año': 2005},
        {'titulo': 'Happy Feet', 'año': 2006},
        {'titulo': 'Ratatouille', 'año': 2007},
        {'titulo': 'WALL-E', 'año': 2008},
        {'titulo': 'Up', 'año': 2009},
        {'titulo': 'Toy Story 3', 'año': 2010},
        {'titulo': 'Rango', 'año': 2011},
        {'titulo': 'Valiente ', 'año': 2012},
        {'titulo': 'Frozen', 'año': 2013},
        {'titulo': 'Grandes Héroes', 'año': 2014} ]
```

Considere, que los datos ya están en el programa, no necesita ingresar la lista de películas, y realize el siguiente proceso:

- Solicite al usuario que ingrese un año,
- Buscar la película por el año, utilizando la función implementada,
- Imprimir el título de la película.

```
Ingrese año: 2002
Pelicula: El viaje de Chihiro
```

```
Ingrese año: 2013
Pelicula: Frozen
```

4. (Nivel 1) Implemente una función, utlizando el algoritmo de búsqueda binaria, para buscar datos que están en una lista de diccionarios sobre ganadores de premios nobel, con la siguiente estructura: año y ganadores. La función debe tomar un parámetro: el año a buscar y debe devolver los ganadores.

   Un ejemplo sería, la lista de ganadores de premios noble de Física:

```
peliculas = [
{'año':2010, 'nombre':'Andre Geim y Konstantín Novosiólov'},
{'año':2011, 'nombre':'Perlmutter, Schmidt y Adam Riess'},
{'año':2012, 'nombre':'Serge Haroche y David Wineland'},
{'año':2013, 'nombre':'Peter Higgs y François Englert'},
{'año':2014, 'nombre':'Akasaki, Hiroshi Amano y Nakamura'},
{'año':2015, 'nombre':'Takaaki Kajita y Arthur B. McDonald'},
{'año':2016, 'nombre':'Thouless, Haldane y Kosterlitz'},
{'año':2017, 'nombre':'Rainer Weiss, Barry Barish y Thorne'},
{'año':2018, 'nombre':'Donna Strickland, Mourou y Ashkin'},
{'año':2019, 'nombre':'James Peebles, Mayor y Queloz'} ]
```

   Considere, que los datos ya están en el programa, no necesita ingresar la lista, y realize el siguiente proceso:

   - Solicite al usuario que ingrese un año,
   - Buscar en la lista de diccionarios por el año, utilizando la función implementada,

- Imprimir los nombres de los ganadores.

```
Ingrese año: 2011
Ganadores: Perlmutter, Schmidt y Adam Riess
```

```
Ingrese año: 2018
Ganadores: Donna Strickland, Mourou y Ashkin
```

5. (Nivel 2) Elabore un programa que pida el nombre de usuario y clave desde teclado y guardarlos en sistema. Luego, preguntar al usuario si desea cambiar su clave.

   - Si responde que si: solicite el usuario y la clave nueva y verifique si el cambio es válido; si lo es se muestra el mensaje CORRECTO y, en caso contrario, se muestra el mensaje ERROR.
   - No se debe permitir ingresar una clave anteriormente usada.
   - Si responde que no: acabar el programa.

   Ejemplo:

```
Usuario: admin
Clave : utec
CORRECTO
Desea cambiar clave[S/N]: S
Usuario: admin
Clave: 1234
CORRECTO
Desea cambiar clave[S/N]: S
Usuario: admin
Clave: utec
ERROR
Desea cambiar clave[S/N]: N
FIN
```

6. (Nivel 2) Implemente una función que realice el algoritmo "búsqueda binaria" de manera `recursiva` para determinar si un número está o no en una lista de números. Asuma que la lista que ingresará a la función se encuentra ordenada de **mayor a menor**.

7. (Nivel 2) Implemente una función que realice el algoritmo "búsqueda binaria" de manera `iterativa` para determinar si un número está o no en una lista de números. Asuma que la lista que ingresará a la función se encuentra ordenada de **mayor a menor**.

8. (Nivel 3) Escribe un programa que realice lo siguiente:

   - Generar la siguiente lista de números ordenados por comprensión:

```
numeros =   [15, 30, 45, 60, 75, 90, 105, 120, 135, 150,
             165, 180, 195, 210, 225, 240, 255, 270, 285,
             300, 315, 330, 345, 360, 375, 390, 405, 420,
             435, 450, 465, 480, 495, 510, 525, 540, 555,
             570, 585, 600, 615, 630, 645, 660, 675, 690,
             705, 720, 735, 750, 765, 780, 795, 810, 825,
             840, 855, 870, 885, 900, 915, 930, 945, 960,
             975, 990]
```

- Solicitar al usuario que ingrese un número.
- Buscar el número ingresado en la lista
- Imprime "`El número está en la posición X de la lista`" cuando el número este en la lista y "`El número NO está en la lista`" en caso contrario.

Para buscar el número ingresado en la lista, implemente y use la siguiente función:

- `buscar_numero` recibe como parámetro una `lista` y el número a buscar, y utilizando el algoritmo de `Búsqueda Binaria` retorne la posición del número buscado, en caso de no estar el elemento retornar -1.

Algunos ejemplos de diálogo son:

```
Ingrese numero a buscar: 645
El número está en la posición 42 de la lista
```

```
Ingrese numero a buscar: 153
El número NO está en la lista
```

9. (Nivel 3) Escribe un programa que realice lo siguiente:

   - Generar la siguiente lista de números múltiplos de 7 por comprensión:

```
numeros =   [  7,  14,  21,  28,  35,  42,  49,  56,  63,
              70,  77,  84,  91,  98, 105, 112, 119, 126,
             133, 140, 147, 154, 161, 168, 175, 182, 189,
             196, 203, 210, 217, 224, 231, 238, 245, 252,
             259, 266, 273, 280, 287, 294, 301, 308, 315,
             322, 329, 336, 343, 350, 357, 364, 371, 378,
             385, 392, 399, 406, 413, 420, 427, 434, 441,
             448, 455, 462, 469, 476, 483, 490, 497]
```

   - Solicitar al usuario que ingrese un número.
   - Buscar el número ingresado en la lista de números
   - Imprime "`El número está en la posición X de la lista`" cuando el número este en la lista y "`El número NO está en la lista`" cuando no esté en la lista.

   Para buscar el número ingresado en la lista, implemente y use la siguiente función:

- `buscar_numero` recibe como parámetro una `lista` y el número a buscar, y utilizando el algoritmo de `Búsqueda Binaria` retorne la posición del número buscado, en caso de no estar el elemento retornar -1.

Algunos ejemplos de diálogo son:

```
Ingrese numero a buscar: 125
El número NO está en la lista
```

```
Ingrese numero a buscar: 385
El número está en la posición 54 de la lista
```

10. (Nivel 3) Escribe una función en Python que permita saber si un número se encuentra en una lista ordenada. Si el número a buscar se encuentra o no en la lista, entonces se deberá mostrar un mensaje apropiado. Adicionalmente, deberá contar el número de comparaciones hechas por su algoritmo. El número total de comparaciones debe ser menor al tamaño de la lista de números y puede asumir que la lista tiene como mínimo 5 elementos.

    Por ejemplo:

```
Lista de numeros: [1,2,3,4,5]
Numero a buscar: 10
El numero 10 no se encuentra despues de 3 comparacion(es)
```

```
Lista de palabras: [10, 20, 30, 40, 50, 60, 70, 80, 90]
Numero a buscar: 50
El numero 50 se encuentra despues de 1 comparacion(es)
```

11. (Nivel 3) Crea una función que reciba una lista ordenada con 100 números y un número a buscar, y luego por medio de búsqueda binaria encuentre la posición de dicho número. Además deberá devolver cuántas búsquedas tuvo que realizar para hallar el resultado.

```
Lista: [ 11,  23,  25,  27,  37,  39,  41,  50,  51,  52,
         58,  66,  71,  76,  83, 111, 117, 129, 143, 163,
        164, 167, 171, 174, 199, 211, 226, 250, 259, 296,
        299, 303, 304, 315, 322, 328, 334, 340, 342, 348,
        363, 374, 384, 392, 408, 411, 411, 420, 441, 448,
        468, 472, 475, 480, 497, 508, 516, 522, 526, 527,
        533, 537, 590, 602, 602, 604, 626, 628, 636, 667,
        684, 695, 699, 702, 707, 717, 726, 731, 772, 779,
        793, 798, 800, 813, 816, 845, 852, 883, 883, 894,
        897, 912, 919, 922, 933, 949, 952, 996, 999, 999]
Ingrese número a buscar: 602
Output:
El número 602 se encuentra en la posición: 63
Se realizó la búsqueda: 4 veces
```

12. (Nivel 3) En UTEC se ha definido un criterio para comparar listas: una lista será mayor, menor o igual que otra dependiendo si el promedio de sus elementos es mayor, menor o igual que el promedio de la segunda lista. Implemente las funciones necesarias de tal forma que sea posible realizar búsquedas binarias iterativamente y así determinar si una lista determinada se encuentra o no en una lista de listas bajo el criterio definido por UTEC. Considere que su algoritmo debe funcionar para una lista de cualquier tamaño que contenga listas de cualquier tamaño. No es necesario implementar el ingreso de la información hacia la lista o matrices. Asuma que la lista ya está ordenada.

    Algunos ejemplos de diálogo de este programa serían:

```
Input:[ [ [1, 1, 2, 2] ], [ [1, 2, 3, 4, 4, 4] ], [ [4, 4, 5, 5] ] ]
Buscar: [ [3, 3] ]
Output: True
```

```
Input:[ [ [1, 1, 2, 2] ], [ [1, 2, 3, 4, 4, 4] ], [ [4, 4, 5, 5] ] ]
Buscar: [ [3, 1] ]
Output: False
```

13. (Nivel 3) En UTEC se ha definido un criterio para comparar matrices: una matriz será mayor, menor o igual si la suma de sus elementos es mayor, menor o igual que la suma de los elementos de la segunda matriz. Implemente las funciones necesarias de tal forma que sea posible realizar búsqueda binaria recursivamente y así determinar si una matriz determinada se encuentra o no en una lista de matrices bajo el criterio definido por UTEC. Considere que su algoritmo debe funcionar para una lista de cualquier tamaño que contenga matrices de cualquier tamaño. No es necesario implementar el ingreso de la información hacia la lista o matrices. Asuma que la lista ya está ordenada.

    Algunos ejemplos de diálogo de este programa serían:

```
Input:[ [ [1, 1], [2, 2] ], [ [0, 2, 3], [4, 4, 4] ], [ [4,
   4], [5, 5] ] ]
Buscar: [ [0, 0], [3, 3] ]
Output: True
```

```
Input:[ [ [1, 1], [2, 2] ], [ [0, 2, 3], [4, 4, 4] ], [ [4,
   4], [5, 5] ] ]
Buscar: [ [1, 4], [3, 3] ]
Output: False
```

14. (Nivel 4) Dado los siguientes diccionarios relacionados a los planetas del sistema planetario solar, el cual contiene los siguientes atributos:

    - planeta
    - diámetro

```
mercurio = {
    "planeta": "mercurio",
    "diametro": 4.878
}

venus = {
    "planeta": "venus",
    "diametro": 12.104
}

tierra = {
    "planeta": "tierra",
    "diametro": 12.756
}

marte = {
    "planeta": "marte",
    "diametro": 6.794
}

jupiter = {
    "planeta": "jupiter",
    "diametro": 142.800
}

saturno = {
    "planeta": "saturno",
    "diametro": 120.660
}

urano = {
    "planeta": "urano",
    "diametro": 51.800
}

neptuno = {
    "planeta": "neptuno",
    "diametro": 49.500
}
```

Se pide lo siguiente:

- Agregar los diccionarios a una lista
- Ordenar la lista por diámetro usando un algoritmo de ordenamiento.
- Encontrar el planeta con diámetro igual a `12.104` usando el algoritmo de búsqueda binaria.

Algunos ejemplos de diálogo de este programa serían:

```
Lista:
[{'diametro': 4.878, 'planeta': 'mercurio'},
 {'diametro': 12.104, 'planeta': 'venus'},
```

```
 {'diametro': 12.756, 'planeta': 'tierra'},
 {'diametro': 6.794, 'planeta': 'marte'},
 {'diametro': 142.8, 'planeta': 'jupiter'},
 {'diametro': 120.66, 'planeta': 'saturno'},
 {'diametro': 51.8, 'planeta': 'urano'},
 {'diametro': 49.5, 'planeta': 'neptuno'}]

Output:
Lista ordenada:
[{'diametro': 4.878, 'planeta': 'mercurio'},
 {'diametro': 6.794, 'planeta': 'marte'},
 {'diametro': 12.104, 'planeta': 'venus'},
 {'diametro': 12.756, 'planeta': 'tierra'},
 {'diametro': 49.5, 'planeta': 'neptuno'},
 {'diametro': 51.8, 'planeta': 'urano'},
 {'diametro': 120.66, 'planeta': 'saturno'},
 {'diametro': 142.8, 'planeta': 'jupiter'}]

El planeta con diámetro igual a 12.104 es venus
```

15. (Nivel 4) Escribir un programa que realize la búsqueda de un número entero dentro de una matriz de tamaño $m \times n$. La matriz tiene las siguientes propiedades:

    - Los enteros de cada fila están ordenados de izquierda a derecha.
    - El primer entero de cada fila es mayor que el último entero de la fila anterior.

```
Ejemplo de matriz en Python:
A=[[1,2,3,4],[5,6,7,8],[9,10,11,12]]
```

    Indicaciones:

    - Usar **búsqueda binaria**.
    - El usuario ingresa el número que desea buscar en la matriz.
    - Imprimir la ubicación (fila, columna) en caso el elemento fuera encontrado.
    - Caso contrario, mostrar el mensaje "El número no existe en la matriz".

16. (Nivel 4) Se está implementando un programa para ESSALUD. Usted debe implementar el algoritmo de búsqueda binaria que devuelve la posición en la lista. Complete la sección de la función `busbin`. La posición en la lista es la llave en el diccionario.

```
dni = [11489834, 12832055, 12869666, 13869179, 16019959, 16426202,
    18947201, 25480016, 28678360, 28717605, 29600532, 29718806, 32719777,
    32843423]

expedientes = {
  0:{'nombre':'juan', 'tipo':'rh-'},
  1:{'nombre':'maria', 'tipo':'a+'},
  2:{'nombre':'jose', 'tipo':'a-'},
  3:{'nombre':'paola', 'tipo':'o+'},
  4:{'nombre':'gabriel', 'tipo':'o+'},
```

```
  5:{'nombre':'maria', 'tipo':'o+'},
  6:{'nombre':'susana', 'tipo':'rh-'},
  7:{'nombre':'sofia', 'tipo':'rh-'},
  8:{'nombre':'manuel', 'tipo':'b-'},
  9:{'nombre':'elene', 'tipo':'b+'},
  10:{'nombre':'javier', 'tipo':'rh-'},
  11:{'nombre':'rosario', 'tipo':'rh-'},
  12:{'nombre':'jhuly', 'tipo':'rh-'},
  13:{'nombre':'samuel', 'tipo':'rh-'}
}

n = int(input("ingrese dni a buscar:"))

def busbin(dni, n):
    k = -1
    #Completar aquí

    return k

pos = busbin(dni, n)

info = expedientes[pos]
print("nombre:", info['nombre'], ", tipo de sangre:", info['tipo'])
```

```
input:
29718806

output:
nombre: rosario, tipo de sangre:rh-
```

17. (Nivel 4) La página web de un banco le ha pedido que mejore su algoritmo de búsqueda sobre una lista de DNIs de clientes. El algoritmo de búsqueda debe encontrar la posición en la lista de un dni ingresado por el usuario. En caso no se encuentre devolver -1. Una vez encontrado imprimir el nombre y el número de cuenta. Note que la posición en la lista es la llave para un diccionario de cuentas.

```
dni = [11489834, 12832055, 12869666, 13869179, 16019959, 16426202,
   18947201, 25480016, 28678360, 28717605, 29600532, 29718806, 32719777,
   32843423]

cuuentas = {
  0:{'nombre':'juan', 'cuenta':'223402'},
  1:{'nombre':'maria', 'cuenta':'235543'},
  2:{'nombre':'jose', 'cuenta':'3434343'},
  3:{'nombre':'paola', 'cuenta':'334564'},
  4:{'nombre':'gabriel','cuenta':'337564'},
  5:{'nombre':'maria', 'cuenta':'334764'},
  6:{'nombre':'susana', 'cuenta':'334564'},
  7:{'nombre':'sofia', 'cuenta':'324564'},
  8:{'nombre':'manuel', 'cuenta':'134564'},
  9:{'nombre':'elene', 'cuenta':'334364'},
  10:{'nombre':'javier', 'cuenta':'934563'},
  11:{'nombre':'rosario', 'cuenta':'334562'},
```

```
  12:{'nombre':'jhuly', 'cuenta':'334567'},
  13:{'nombre':'samuel', 'cuenta':'334565'}
}

n = int(input("ingrese dni a buscar:")

def busbin(dni, n):
    k = -1
    #Completar aquí

    return k

pos = busbin(dni, n)

info = cuentas[pos]

print("nombre:", info['nombre'],
    ", cuenta de ahorro:", info['cuenta'])
```

```
input:
29718806

output:
nombre: rosario, cuenta de ahorro: 334562
```

18. (Nivel 4) **Sistema Solar:** crear un código que realize una búsqueda de datos del Sistema Solar, basado en la tabla: `Pista:` Están ordenados por año. E imprima los siguientes

planeta	**diámetro**	**año**	**día**
Mercurio	4878	88	59
Venus	12104	225	5784
Tierra	12760	365	24
Marte	6787	687	24
Júpiter	139822	4343	10
Saturno	120500	10767	11
Urano	51120	30660	18
Neptuno	49530	60225	19

resultados:

- Planetas con diámetro mayor que la Tierra.
- Planetas con años más largos que la Tierra.
- Planetas con diámetro menor que la Tierra y días iguales o más cortos que los de la Tierra.

```
Planetas con diámetro mayor que la Tierra: Júpiter, Saturno, Urano,
   Neptuno
```

```
Planetas con años más largos que la Tierra: Marte, Júpiter, Saturno, Urano
    , Neptuno
Planetas con diámetro menor que la Tierra y días iguales o más cortos que
    los de la Tierra: Marte
```

19. (Nivel 4) Crea una función que reciba una lista con 25 números y un número a buscar, y luego ordene la lista y realice la búsqueda del número y devuelva la posición del número en la lista ya ordenada e imprima la misma. En caso no encuentre el número deberá devolver un -1. Deberás usar búsqueda binaria.

```
Lista: [41, 38, 77, 0, 51, 46, 60, 78, 42, 32, 78, 38, 25, 64, 9, 96, 41,
    39, 32, 7, 26, 43, 43, 87, 7]
Ingrese número a buscar: 60
Output:
Lista Ordenada: [0, 7, 7, 9, 25, 26, 32, 32, 38, 38, 39, 41, 41, 42, 43,
    43, 46, 51, 60, 64, 77, 78, 78, 87, 96]
Se encontró el número 60 en la posición: 18
```

20. (Nivel 5) Se plantea el siguiente juego: en una mesa se tiene una fila de n vasos y dentro de cada vaso hay una bola enumerada. Se pide hallar la mínima cantidad de vasos que se necesita levantar para verificar la existencia de un número dado. Asumir que las bolas están ordenadas de forma descendente por lo tanto, lo mejor es aplicar la estrategia de búsqueda binaria que minimice el número de vasos a levantar.

Un ejemplo de los datos de entrada y salida del programa son:

```
Bolitas: 21,18,15,11,9,8,5,2
Buscar: 5
Vasitos levantados: 3
```

El prototipo del programa lleva la siguiente estructura:

```
def contarVasosLevantados(lista, search):
  cont = 0

  # Use el algoritmo de búsqueda binaria para
  # buscar el numero "search" y contabilice
  # la cantidad de vasitos a levantar en la variable "cont"

  return cont
```

```
bolitasStr=input("").split(",")
search=int(input(""))
bolitasInt = [int(i) for i in bolitasStr]
cont = contarVasosLevantados(bolitasInt, search)
print(cont)
```

21. (Nivel 5) Existe el requerimiento de programar una aplicación móvil de magia en la cual se encuentra el truco "Adivina las cartas" por lo que desea crear un programa en python que le permite ordenar 52 cartas.

    Las cartas están ordenadas de manera que el primer mazo es corazones, el segundo espadas, el tercero diamantes y el cuarto treboles.

    Generar un programa en python que cumpla con lo siguiente:

    - El usuario debe ingresar dos números e indicar el mazo (Ejemplos de input: 'K, espadas', 'J, treboles')
    - Crear una función que genere el diccionario de cartas respetando el orden de mazo propuesto en el enunciado.
    - Crear otra función que reciba el input del usuario, busque en forma binaria las posiciones y devuelva como resultado un solo string donde la posición 1 (que corresponde al input 1) es seguida de la posición 2 (que corresponde al input 2).

    Ejemplo:

```
Carta 1: '10, corazones'
Carta 2: 'Q, treboles'
Resultado: 1051
```