**ESTRUCTURA DE DATOS**

|  |  |  |  |
| --- | --- | --- | --- |
| Estudiante: | Juan Pablo Silvestre | | |
| Práctica: | Ordenamientos | Fecha: |  |

**Selección**

|  |  |
| --- | --- |
| def seleccion(lista):  for i in range(0, len(lista)-1):  minimo = i  for j in range(i+1, len(lista)):  if(lista[j] < lista[minimo]):  minimo= j  print(lista)  lista[i], lista[minimo] = lista[minimo], lista[i]    listaimpares = [11, 3, 81, 7, 45]  seleccion(listaimpares) | Análisis |
| List= [11, 3, 81, 7, 45]  I = 0; 4 | minimo = 0 | j = 1; 5  3 < 11  Minimo = 1  [3, 11, 81, 7, 45]  ===========================  List= [3, 11, 81, 7, 45]  I = 0; 4 | minimo = 1| j = 2; 5  81 < 11  Minimo = 1  [3, 11, 81, 7, 45]  ===========================  List= [3, 11, 81, 7, 45]  I = 0; 4 | minimo = 1| j = 3; 5  7 < 11  Minimo = 3  [3, 7, 81, 11, 45]  ===========================  List= [3, 7, 81, 11, 45]  I = 0; 4 | minimo = 3| j = 4; 5  45 < 11  Minimo = 3  [3, 7, 81, 11, 45]  ===========================  List= [3, 7, 81, 11, 45]  I = 0; 4 | minimo = 3| j = 5; 5  Nada que comparar  Minimo = 3  [3, 7, 81, 11, 45]  ===========================  List= [3, 7, 81, 11, 45]  I = 1; 4 | minimo = 1| j = 2; 5  81 < 7  Minimo = 1  [3, 7, 81, 11, 45]  ===========================  List= [3, 7, 81, 11, 45]  I = 2; 4 | minimo = 2| j = 3; 5  11 < 7  Minimo = 1  [3, 7, 81, 11, 45]  ===========================  List= [3, 7, 81, 11, 45]  I = 1; 4 | minimo = 1| j = 4; 5  45 < 7  Minimo = 1  [3, 7, 81, 11, 45]  ===========================  List= [3, 7, 81, 11, 45]  I = 1; 4 | minimo = 1| j = 5; 5  Nada que comparar  Minimo = 1  [3, 7, 81, 11, 45]  ===========================  List= [3, 7, 81, 11, 45]  I = 2; 4 | minimo =2 | j = 3; 5  11 < 81  Minimo = 3  [3, 7, 11, 81, 45]  ===========================  List= [3, 7, 11, 81, 45]  I = 2; 4 | minimo =3 | j = 4; 5  45 < 81  Minimo = 4  [3, 7, 11, 45, 81]  ===========================  List= [3, 7, 11, 81, 45]  I = 2; 4 | minimo =3 | j = 5; 5  Nada que comparar  Minimo = 4  [3, 7, 11, 45, 81]  ===========================  Done. |
|  |  |

**Quicksort**

|  |  |
| --- | --- |
| def quicksort(lista, primero, ultimo):  izquierda=primero  derecha=ultimo-1  pivote=ultimo  while (izquierda<derecha):  while (lista[izquierda]<lista[pivote]) and (izquierda <=derecha):  izquierda +=1  while (lista[derecha]>lista[pivote]) and (derecha >=izquierda):  derecha -=1  if(izquierda <derecha):  lista[izquierda], lista[derecha] = lista[derecha], lista[izquierda]  print(lista)  if(lista[pivote]<lista[izquierda]):  lista[izquierda], lista[pivote] = lista[pivote], lista[izquierda]  print(lista)  if(primero < izquierda):  quicksort(lista, primero, izquierda-1)  if(ultimo>izquierda):  quicksort(lista, izquierda+1, ultimo)    listaimpares = [11, 3, 81, 7, 45]  quicksort(listaimpares,0,len(listaimpares)-1) | Análisis |
|  |

**Inserción**

|  |  |
| --- | --- |
| def insercion(lista):  for i in range(1, len(lista)+1):  k=i-1  while (k>0) and (lista[k]<lista[k-1]):  lista[k], lista[k-1] = lista[k-1], lista[k]  k -= 1  print(lista)  listaimpares = [11, 3, 81, 7, 45]  insercion(listaimpares) | Análisis |
| #1  List = [11, 3, 81, 7, 45]  i = 1; 6 | k = 0  nada que comparar  New List = [11, 3, 81, 7, 45]  ========================  #2  List = [11, 3, 81, 7, 45]  i = 2; 6 | k = 1  k > 0 and 3 < 11 True  k = 0  New List = [3, 11, 81, 7, 45]  ========================  #3  List = [3, 11, 81, 7, 45]  i = 3; 6 | k = 2  k > 0 and 81 < 11 False  New List = [3, 11, 81, 7, 45]  ========================  #4  List = [3, 11, 81, 7, 45]  i = 4; 6 | k = 3  k > 0 and 7 < 81 True  New List = [3, 11, 7, 81, 45]  K = 2  ========================  #5  List = [3, 11, 7, 81, 45]  i = 4; 6 | k = 2  k > 0 and 7 < 11 True  New List = [3, 7, 11, 81, 45]  K = 1  ========================  #6  List = [3, 7, 11, 81, 45]  i = 4; 6 | k = 1  k > 0 and 7 < 3 False  New List = [3, 7, 11, 81, 45]  ========================  #7  List = [3, 7, 11, 81, 45]  i = 5; 6 | k = 4  k > 0 and 45 < 81 True  New List = [3, 7, 11, 45, 81]  K = 3  ========================  #8  List = [3, 7, 11, 81, 45]  i = 5; 6 | k = 3  k > 0 and 81 < 11 False  New List = [3, 7, 11, 45, 81]  ========================  #9  List = [3, 7, 11, 45, 81]  i = 6; 6 | k = 5  k > 0 and 81 < 45 False  New List = [3, 7, 11, 45, 81]  ========================  Fin |

**Mergesort**

|  |  |
| --- | --- |
| def mergesort(lista):  if(len(lista)<=1):  return lista  else:  medio=len(lista)//2  izquierda=[]  for i in range(0, medio):  izquierda.append(lista[i])  derecha=[]  for i in range(medio, len(lista)):  derecha.append(lista[i])  izquierda=mergesort(izquierda)  derecha=mergesort(derecha)  if(izquierda [medio-1]<= derecha[0]):  izquierda+=derecha  return izquierda  resultado=merge(izquierda, derecha)  return resultado  def merge(izquierda, derecha):  lista\_mezclada=[]  while (len(izquierda)>0) and (len(derecha)>0):  if(izquierda [0] <derecha[0]):  lista\_mezclada.append(izquierda.pop(0))  else:  lista\_mezclada.append(derecha.pop(0))  if(len(izquierda)>0):  lista\_mezclada +=izquierda  if(len(derecha)>0):  lista\_mezclada +=derecha  print(lista\_mezclada)  return lista\_mezclada    listaimpares = [11, 3, 81, 7, 45]  mergesort(listaimpares) | Análisis |
|  |

**Countsort**

|  |  |
| --- | --- |
| def countsort(lista,maximo):  lista\_conteo=[0]\*(maximo+1)  lista\_ordenada=[None]\*len(lista)    for i in lista:  lista\_conteo[i]+=1    total=0  for i in range(len(lista\_conteo)):  lista\_conteo[i], total = total, total+lista\_conteo[i]    for indice in lista:  lista\_ordenada[lista\_conteo[indice]] = indice  lista\_conteo[indice]+=1  print(lista\_ordenada)  return lista\_ordenada    listaimpares = [9, 3, 1, 5, 9,2,0,1]  countsort(listaimpares, max(listaimpares)) | Análisis |
| List = [9, 3, 1, 5, 9,2,0,1] |

**Bucker sort**

|  |  |
| --- | --- |
| def bucketSort(array):  bucket = []  for i in range(len(array)):  bucket.append([])  for j in array:  index\_b = int(10 \* j)  bucket[index\_b].append(j)  for i in range(len(array)):  bucket[i] = sorted(bucket[i])  k = 0  for i in range(len(array)):  for j in range(len(bucket[i])):  array[k] = bucket[i][j]  k += 1  return array  list = [.42, .32, .33, .52, .37, .47, .51]  print(bucketSort(list)) | Análisis |
|  |

**Radix sort**

|  |  |
| --- | --- |
| # Radix sort in Python  # Using counting sort to sort the elements in the basis of significant places  def countingSort(array, place):  size = len(array)  output = [0] \* size  count = [0] \* 10  # Calculate count of elements  for i in range(0, size):  index = array[i] // place  count[index % 10] += 1  # Calculate cumulative count  for i in range(1, 10):  count[i] += count[i - 1]  # Place the elements in sorted order  i = size - 1  while i >= 0:  index = array[i] // place  output[count[index % 10] - 1] = array[i]  count[index % 10] -= 1  i -= 1  for i in range(0, size):  array[i] = output[i]  # Main function to implement radix sort  def radixSort(array):  # Get maximum element  max\_element = max(array)  # Apply counting sort to sort elements based on place value.  place = 1  while max\_element // place > 0:  countingSort(array, place)  place \*= 10  data = [121, 432, 564, 23, 1, 45, 788]  radixSort(data)  print(data) | Análisis |
|  |

**Shell sort**

|  |  |
| --- | --- |
| # Shell sort in python  def shellSort(array, n):  # Rearrange elements at each n/2, n/4, n/8, ... intervals  interval = n // 2  while interval > 0:  for i in range(interval, n):  temp = array[i]  j = i  while j >= interval and array[j - interval] > temp:  array[j] = array[j - interval]  j -= interval  array[j] = temp  interval //= 2  data = [9, 8, 3, 7, 5, 6, 4, 1]  size = len(data)  shellSort(data, size)  print('Sorted Array in Ascending Order:')  print(data) | Análisis |
|  |

**Timsort**

|  |  |
| --- | --- |
| MINIMUM= 32    def find\_minrun(n):        r = 0      while n >= MINIMUM:          r |= n & 1          n >>= 1      return n + r    def insertion\_sort(array, left, right):      for i in range(left+1,right+1):          element = array[i]          j = i-1          while element<array[j] and j>=left :              array[j+1] = array[j]              j -= 1          array[j+1] = element      return array    def merge(array, l, m, r):        array\_length1= m - l + 1      array\_length2 = r - m      left = []      right = []      for i in range(0, array\_length1):          left.append(array[l + i])      for i in range(0, array\_length2):          right.append(array[m + 1 + i])        i=0      j=0      k=l        while j < array\_length2 and  i < array\_length1:          if left[i] <= right[j]:              array[k] = left[i]              i += 1            else:              array[k] = right[j]              j += 1            k += 1        while i < array\_length1:          array[k] = left[i]          k += 1          i += 1        while j < array\_length2:          array[k] = right[j]          k += 1          j += 1    def tim\_sort(array):      n = len(array)      minrun = find\_minrun(n)        for start in range(0, n, minrun):          end = min(start + minrun - 1, n - 1)          insertion\_sort(array, start, end)        size = minrun      while size < n:            for left in range(0, n, 2 \* size):                mid = min(n - 1, left + size - 1)              right = min((left + 2 \* size - 1), (n - 1))              merge(array, left, mid, right)            size = 2 \* size          array = [-1,5,0,-3,11,9,-2,7,0]    print("Array:")  print(array)    tim\_sort(array)    print("Sorted Array:")  print(array) | Análisis |
|  |