|UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

Facultad de Ciencias de la Ingeniería y Tecnología

Unidad Valle de las Palmas

Unidad 2: Divide y vencerás.
Meta 2.3

Análisis de algoritmos

José Humberto Moreno Mejia

Septiembre 2024

• Selecciona 2 funciones que no has usado de la lista del archivo TestFunctions Grafica las funciones en Matlab (obteniendo una imagen)
• Guarda la imagen generada (plot)
• Ordena los datos obtenidos de cada una de las funciones utilizando un método de ordenamiento visto en clase.
• Realiza un reporte que incluya las funciones seleccionadas, el código para graficarlas, la imagen generada e imagen de los datos antes y después ordenados (tomar captura en el software).
• Subir tu reporte en el espacio generado como Evidencia_M2.3

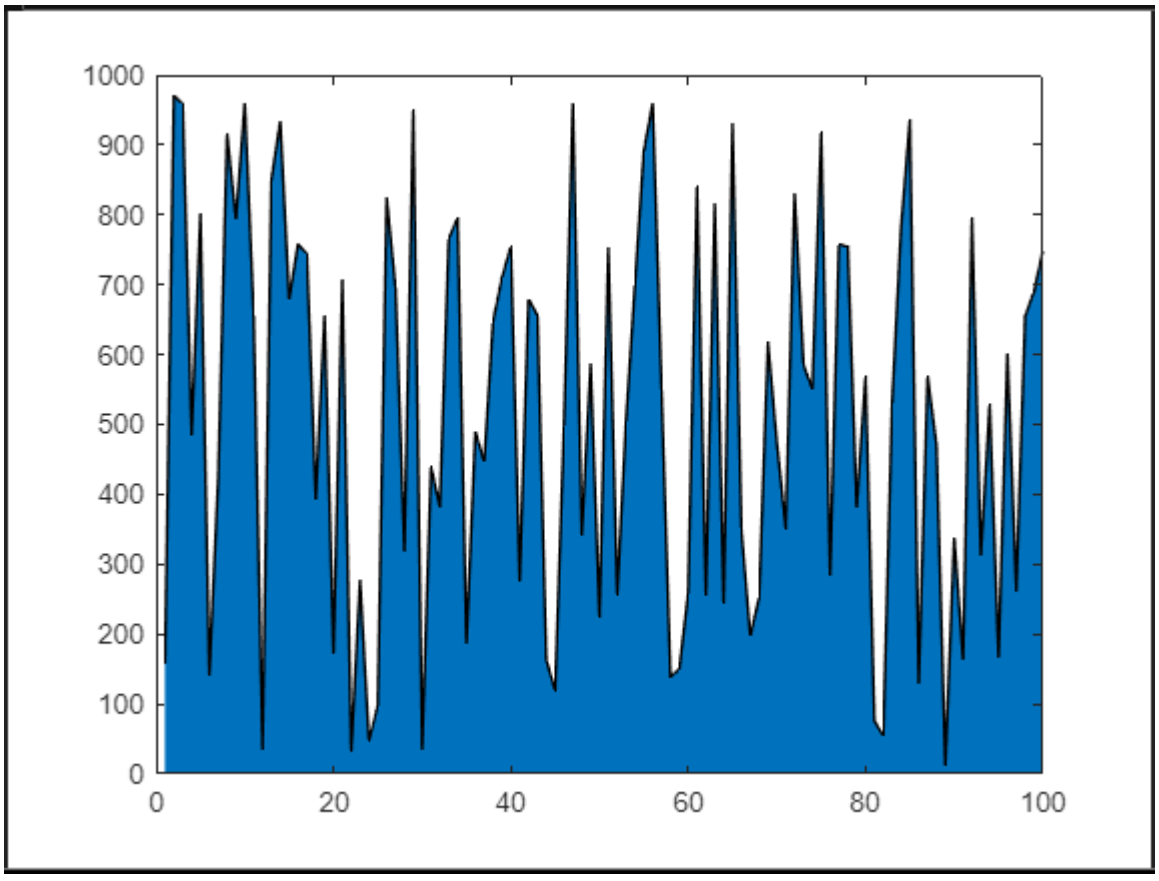
radixSort

Algoritmos utilizados

```
% Función Radix Sort
function sortedArray = radixSort(array)
    maxNum = max(array);  % Encuentra el número más grande del array
    exp = 1;  % Exp inicia en 1 para empezar a ordenar por el dígito menos significativo (unidades)
    % Mientras haya dígitos que procesar
    while floor(maxNum / exp) > 0
        array = countingSortByDigit(array, exp);  % Ordena el array por el dígito actual
        exp = exp * 10;  % Pasa al siguiente dígito (decenas, centenas, etc.)
    end
    sortedArray = array;  % El array ya está ordenado
end

% Función Counting Sort por dígito
function sortedArray = countingSortByDigit(array, exp)
    % Contador de ocurrencias para cada dígito (0-9)
    count = zeros(1, 10);
    n = length(array);
    output = zeros(1, n);
    % Contar cuántas veces aparece cada dígito en el dígito actual (exp)
    for i = 1:n
        digit = floor(array(i) / exp) - floor(array(i) / (exp * 10)) * 10;
        count(digit + 1) = count(digit + 1) + 1;  % +1 porque MATLAB no tiene índice 0
    end
    % Convertir el conteo a posiciones acumuladas
    for i = 2:10
        count(i) = count(i) + count(i - 1);
    end
    % Construir el array ordenado por el dígito actual
    for i = n:-1:1
        digit = floor(array(i) / exp) - floor(array(i) / (exp * 10)) * 10;
        output(count(digit + 1)) = array(i);
        count(digit + 1) = count(digit + 1) - 1;
    end
    % Copiar el resultado en el array original
    sortedArray = output;
end
```

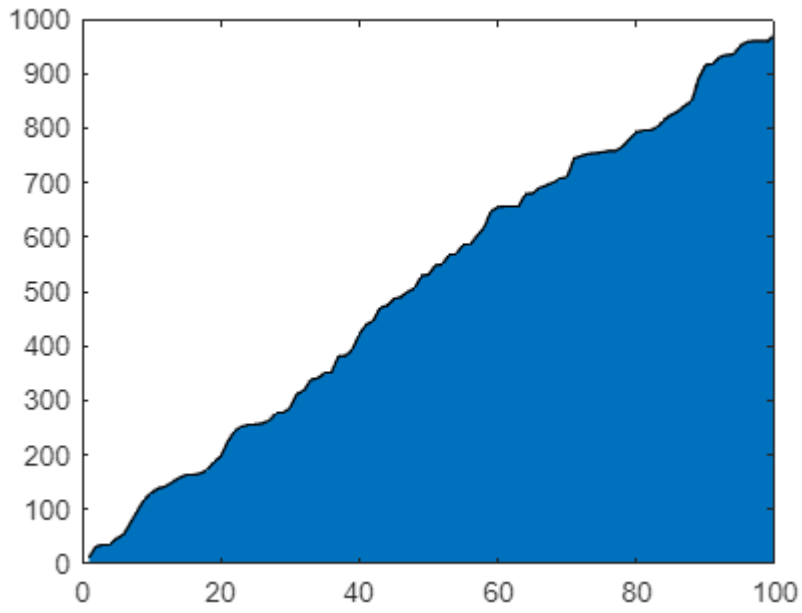
Array desordenado

```
Array original:
  Columns 1 through 32

   158   971   958   486   801   142   422   916   793   960   656    36   850   934   679   758   744   393   656   172   707    32   277    47    98   824   695   318   951    35   439   382

  Columns 33 through 64

   766   796   187   490   446   647   710   755   277   680   656   163   119   499   960   341   586   224   752   256   506   700   891   960   548   139   150   258   841   255   815   244

  Columns 65 through 96

   930   350   197   252   617   474   352   831   586   550   918   286   758   754   381   568    76    54   531   780   935   130   569   470    12   338   163   795   312   529   166   602

  Columns 97 through 100

   263   655   690   749
```



Array Ordenado

```
Array ordenado:
  Columns 1 through 32

    12    32    35    36    47    54    76    98   119   130   139   142   150   158   163   163   166   172   187   197   224   244   252   255   256   258   263   277   277   286   312   318

  Columns 33 through 64

   338   341   350   352   381   382   393   422   439   446   470   474   486   490   499   506   529   531   548   550   568   569   586   586   602   617   647   655   656   656   656   679

  Columns 65 through 96

   680   690   695   700   707   710   744   749   752   754   755   758   758   766   780   793   795   796   801   815   824   831   841   850   891   916   918   930   934   935   951   958

  Columns 97 through 100

   960   960   960   971
```



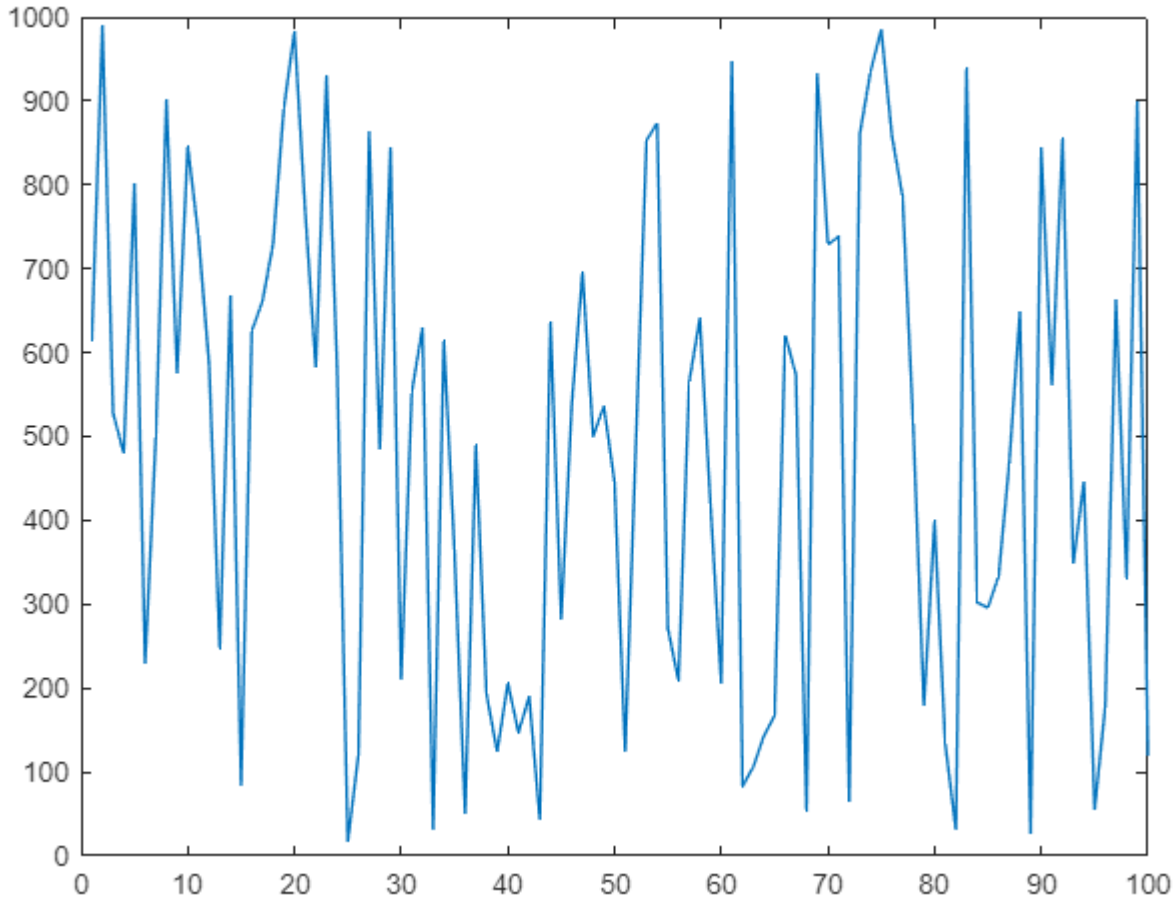
heapsort

Algoritmos utilizados

```
% Función Heap Sort
function sortedArray = heapSort(array)
    n = length(array);
    % Construir el Max-Heap
    for i = floor(n/2):-1:1
        array = heapify(array, n, i);
    end
    % Extraer elementos del heap
    for i = n:-1:2
        % Intercambiar el elemento máximo con el último elemento
        array([1 i]) = array([i 1]);
        % Volver a aplicar heapify para mantener la propiedad del heap
        array = heapify(array, i-1, 1);
    end
    sortedArray = array;  % El array ya está ordenado
end

% Función Heapify
function array = heapify(array, n, i)
    % Inicializar el nodo actual como el más grande
    largest = i;
    left = 2 * i;  % Hijo izquierdo
    right = 2 * i + 1;  % Hijo derecho

    % Si el hijo izquierdo es mayor que el nodo actual
    if left <= n && array(left) > array(largest)
        largest = left;
    end
    % Si el hijo derecho es mayor que el nodo más grande hasta ahora
    if right <= n && array(right) > array(largest)
        largest = right;
    end
    % Si el nodo más grande no es el nodo actual
    if largest ~= i
        % Intercambiar el nodo actual con el más grande
        array([i largest]) = array([largest i]);
        % Aplicar heapify de manera recursiva en el subárbol afectado
        array = heapify(array, n, largest);
    end
end
```

Array desordenado

```
Array original:
  Columns 1 through 30

   531   655   408   820   719   969   532   326   106   611   779   424    91   267   154   282   441   528   458   876   519   944   638   958   241   677   290   672   696    68

  Columns 31 through 60

   255   225   668   845   345   781   676     7   603   387   916     2   463   425   461   771   323   785   472    36   176   722   474   153   342   608   192   739   243   918

  Columns 61 through 90

   270   766   189   288    92   577   684   547   426   645   648   680   636   946   209   710   237   120   608   451   459   662   771   351   663   417   842   833   257   614

  Columns 91 through 100

   583   541   870   265   319   120   940   646   480   640
```



Array Ordenado

```
Array ordenado:
  Columns 1 through 30

    2     7    36    68    91    92   106   120   120   153   154   176   189   192   209   225   237   241   243   255   257   265   267   270   282   288   290   319   323   326

  Columns 31 through 60

  342   345   351   387   408   417   424   425   426   441   451   458   459   461   463   472   474   480   519   528   531   532   541   547   577   583   603   608   608   611

  Columns 61 through 90

  614   636   638   640   645   646   648   655   662   663   668   672   676   677   680   684   696   710   719   722   739   766   771   771   779   781   785   820   833   842

  Columns 91 through 100

  845   870   876   916   918   940   944   946   958   969
```