Universidad de La Laguna

Ejercicios de Código de Ordenación

Algoritmos y Estructuras de Datos Avanzadas
Grado en Ingeniería Informática

Ejercicios de código de ordenación

1. Escribe un procedimiento en C++ que ordene la primera mitad de la secuencia o vector por el método de **selección** de menor a mayor, luego ordene la segunda mitad también por el método de **inserción**, pero de mayor a menor, y finalmente **mezcle** ambas mitades de forma que el vector quede ordenado de mayor a menor ordenada.

2. Escribe un procedimiento en C++ que ordene de **menor a mayor** por el método de ordenación **por inserción** los elementos de las posiciones impares entre sí, y luego ordene de **mayor a menor** por el método de ordenación por **selección** los elementos de las posiciones pares, y finalmente mezcle ambas partes.

3. Escribe un procedimiento C++ que aplique el procedimiento de ordenación siguiente: se aplica una iteración del método **Quicksort** para dividir la secuencia en dos partes y se ordena, la primera parte por **inserción** y la segunda por **selección**.

4. Escribe un procedimiento C++ que aplique un procedimiento de ordenación que combine las ideas de los métodos de ordenación por inserción y Quicksort de la forma siguiente: en primer lugar se elige un pivote, entonces dos índices recorren la parte por ordenar en sentidos ascendente descendente, respectivamente. Si el recorrido ascendente encuentra un valor menor o igual que el pivote, lo inserta en orden por la izquierda pero si encuentra un valor mayor que el pivote se detiene. Si el recorrido descendente encuentra un valor mayor o igual que el pivote, lo inserta por la derecha pero si encuentra un valor menor que el pivote se detiene. Cuando ambos recorridos se detienen, se intercambian los elementos donde se han parado que se insertan en la parte correspondiente y se continúa el proceso hasta que ambos recorridos se encuentren.

Ejercicio 1

```
void Ordenar(int *sec, int N) {
   int i, j, k, min, x;
   int mitad = N / 2;
   int aux[N];

      //  Primera mitad [0, mitad]:
      //  selección de menor a mayor
   for (i = 0; i < mitad; i++) {
      min = i;
      for (j = i + 1; j <= mitad; j++)
         if (sec[j] < sec[min])
            min = j;
      x = sec[min];
      sec[min] = sec[i];
      sec[i] = x;
   }

      //  Segunda mitad [mitad+1, N-1]:
      //  inserción de mayor a menor
   for (i = mitad + 2; i < N; i++) {
      x = sec[i];
      j = i;
      while (( x > sec[j - 1]) && (j > mitad + 1)) {
         sec[j] = sec[j - 1];
         j--;
      }
      sec[j] = x;
   }

      // Mezclar las dos mitades de mayor a menor
   i = mitad;
   j = mitad + 1;
   k = 0;
   while ((i >= 0) && (j <= N-1)) {
      if (sec[i] > sec[j]) {
         aux[k] = sec[i];
         i--;
      }
      else {
         aux[k] = sec[j];
         j++;
      }
      k++;
   }

      // Se copia la subsecuencia no completa
   if (i < 0)
      while (j <= N-1) {
         aux [k] = sec[j];
         j++; k++;
      }
   else
      while (i >= 0) {
         aux[k] = sec[i];
         i--; k++;
      }

      // Se copia al array original
   for (i = 0; i < N; i++)
      sec[i] = aux[i];
}
```

Ejercicio 2

```
void Ordenar(int *sec, int N) {
   int i, j, k, max, x;
   int aux[N];

      // Elementos de posiciones impares:
      // inserción de menor a mayor
   for (i = 3; i < N; i+=2) {
      x = sec[i];
      j = i;
      while (( x < sec[j - 2]) && (j > 1)) {
         sec[j] = sec[j - 2];
         j-= 2;
      }
      sec[j] = x;
   }

      //  Elementos de posiciones pares:
      //  selección de mayor a menor
   for (i = 0; i < N-1; i+=2) {
      max = i;
      for (j = i + 2; j < N; j+=2)
         if (sec[j] > sec[max])
            max = j;
      x = sec[max];
      sec[max] = sec[i];
      sec[i] = x;
   }

      // Mezclar las dos subsecuencias de menor a mayor
   if (N % 2 == 0)    // Subsecuencia posiciones pares
      i = N - 2;
   else
      i = N - 1;
   j = 1;                     // Subsecuencia posiciones impares
   k = 0;
   while ((i >= 0) && (j <= N-1)) {
      if (sec[i] < sec[j]) {
         aux[k] = sec[i];
         i-=2;
      }
      else {
         aux[k] = sec[j];
         j+=2;
      }
      k++;
   }

      // Se copia la subsecuencia no completa
   if (i < 0)
      while (j <= N-1) {
         aux [k] = sec[j];
         j+=2; k++;
      }
   else
      while (i >= 0) {
         aux[k] = sec[i];
         i-=2; k++;
      }

      // Se copia al array original
   for (i = 0; i < N; i++)
      sec[i] = aux[i];
}
```

Ejercicio 3

```
void Ordenar(int *sec, int N) {
   int i, j, k, f, min, piv, x;

      // Iteración de QuickSort:
       // dividir la secuencia en dos
   i = 0;
   f = N - 1;
   piv = sec[(i + f) / 2];

   while (i <= f) {
      while (sec[i] < piv) i++;
      while (sec[f] > piv) f--;
      if (i <= f) {
         x = sec[i];
       sec[i] = sec[f];
       sec[f] = x;
       i++;
       f--;
       }
   }
```

```
      // Primera subsecuencia [0, f]: inserción
   for (k = 1; k <= f; k++) {
      x = sec[k];
      j = k;
      while (( x < sec[j - 1]) && (j > 0)) {
         sec[j] = sec[j - 1];
         j--;
      }
      sec[j] = x;
   }

      // Segunda subsecuencia [i, N-1]: selección
   for (k = i; k < N-1; k++) {
      min = k;
      for (j = k + 1; j < N; j++)
         if (sec[j] < sec[min])
            min = j;
      x = sec[min];
      sec[min] = sec[k];
      sec[k] = x;
   }
}
```

Ejercicio 4

Ejemplo:

```
[22  21  36  42  33  38  45  31  17] Pivote: 33
[22  21  36  42  33  38  45  31  17] Inserta 22
[21  22  36  42  33  38  45  31  17] Inserta 21
[21  22  36  42  33  38  45  31  17] Encuentra 36
[21  22  36  42  33  38  45  31  17] Encuentra 17
[21  22  17  42  33  38  45  31  36] Intercambia
[17  21  22  42  33  38  45  31  36] Inserta 17
[17  21  22  42  33  38  45  31  36] Inserta 36
[17  21  22  42  33  38  45  31  36] Encuentra 42
[17  21  22  42  33  38  45  31  36] Encuentra 31
[17  21  22  31  33  38  45  42  36] Intercambia
[17  21  22  31  33  38  45  42  36] Inserta 31
[17  21  22  31  33  38  45  36  42] Inserta 42
[17  21  22  31  33  38  45  36  42] Inserta 33
[17  21  22  31  33  38  45  36  42] Encuentra 38
[17  21  22  31  33  38  36  42  45] Inserta 45
[17  21  22  31  33  36  38  42  45] Inserta 38
```

Ejercicio 4

```
void Ordenar(int sec[], int N) {
   int i, j, f, piv, x;

   i = 0;
   f = N - 1;
   piv = sec[(i + f) / 2];

   while (i <= f) {

      // Recorrido ascendente
      while (sec[i] <= piv) {
         x = sec[i];
         j = i;
         while (( x < sec[j - 1]) && (j > 0)) {
            sec[j] = sec[j - 1];
            j--;
         }
         sec[j] = x;
         i++;
      }

      // Recorrido descendente
      while (sec[f] >= piv)  {
         x = sec[f];
         j = f;
         while (( x > sec[j + 1]) && (j < N)) {
            sec[j] = sec[j + 1];
            j++;
         }
         sec[j] = x;
         f--;
      }


      // Intercambio de valores
      if (i <= f) {
         x = sec[i];
         sec[i] = sec[f];
         sec[f] = x;
      }

   }  // while (i <= f)
}  // void Ordenar
```


Universidad
de La Laguna