Obligatorio II

MiMax, Anagramas y Cuadrados Latinos

Miguel A. Diab (125415)

Algoritmos y Estructuras de Datos

Universidad ORT – 20 de Mayo de 2009

1. Índice

2. MiMax

La función MiMax utiliza el paradigma **Divide para Conquistar** para calcular el mínimo y máximo de una lista de valores.

```
public static int[] miMax(int[] v1, int desde, int hasta) throws AssertionError {
    // MiMax basado en Dividir para Conquistar
    assert (v1.length>desde) : "Desde es mayor que largo de vector";
    assert (v1.length>hasta) : "Hasta es mayor que largo de vector";
    int[] miMax = new int[2];
    if (desde==hasta) {
      miMax[0] = v1[desde];
      miMax[1] = v1[desde];
    }
    else if (desde==(hasta-1)) {
      miMax[0] = (v1[desde]<v1[hasta])?v1[desde]:v1[hasta];
      miMax[1] = (v1[desde]>v1[hasta])?v1[desde]:v1[hasta];
    }
    else {
      int medio = (desde+hasta)/2;
      int[] tempMax1 = miMax(v1, desde,medio);
      int[] tempMax2 = miMax(v1, medio+1, hasta);
      miMax[0] = (tempMax1[0]<tempMax2[0])?tempMax1[0]:tempMax2[0];
      miMax[1] = (tempMax1[1]>tempMax2[1])?tempMax1[1]:tempMax2[1];
    }
    return miMax;
  }
```

3. Anagramas

Dado una palabra y una lista de palabras, busca en la segunda por anagramas de la primera.

Para eso utiliza las funciones **esAnagrama** y **ordenarPalabra**,

```
public static ArrayList<String> anagramas(String palabra, ArrayList<String> v) {
    ArrayList<String> lista = new ArrayList<String>();
    for (String tempPalabra : v) {
      if (esAnagrama(tempPalabra, palabra)) {
        lista.add(tempPalabra);
      }
    }
    return lista;
  }
```

esAnagrama verifica que al convertir a minúsculas dos palabras y ordenar sus caracteres en orden alfabético, éstas sean iguales.

```
  private static boolean esAnagrama(String palabra1, String palabra2) {
    assert(palabra1!=null);
    assert(palabra2!=null);
    if (palabra1.toLowerCase().equals(palabra2.toLowerCase())) return true;
    else {
      if
(ordenarPalabra(palabra1.toLowerCase()).equals(ordenarPalabra(palabra2.toLowerCas
e()))) {
        return true;
      }
      else {
        return false;
      }
    }
  }
```

ordenarPalabra toma una palabra y reordena sus caracteres en orden alfabético.

```
  private static String ordenarPalabra(String palabra) {
    char[] palabraOrdenada = palabra.toCharArray();
    java.util.Arrays.sort(palabraOrdenada);
    return new String(palabraOrdenada);
  }
```

4. Cuadrado Latino

Dado un número n, la función intenta encontrar un cuadrado latino aleatorio en una cantidad de reintentos dados.

```
public static int[][] cuadradoLatino(int lado) throws Exception {
    int[][] cuadradoLatino;
    int retry = 0;
    boolean esValido = false;
    do {
      cuadradoLatino = new int[lado][lado];
      boolean timeout = false;
      for (int i=0;i<lado;i++) {
        for (int j=0;j<lado;j++) {
          long start = System.currentTimeMillis();
          long elapsedTimeMillis = 0;
          do {
            cuadradoLatino = crearCuadradoLatino(cuadradoLatino, i, j);
            esValido = validarCuadradoLatino(cuadradoLatino);
            elapsedTimeMillis = System.currentTimeMillis()-start;
            if (elapsedTimeMillis>2) timeout = true;
          } while (!esValido && !timeout);
        }
      }
      if (timeout && retry>3000) throw new Exception("Timeout");
      retry++;
    } while (!esValido);
    return cuadradoLatino;
  }
```

validarCuadradoLatino verifica que cada valor i,j del cuadrado no se repita en ningún x,j ni i,x.

```
private static boolean validarCuadradoLatino(int[][] cuadradoLatino) {
    boolean esValido = true;
    for (int i=0;i<cuadradoLatino[0].length;i++) {
      for (int j=0;j<cuadradoLatino[0].length;j++) {
        for (int x=0;x<cuadradoLatino[0].length;x++) {
          if (cuadradoLatino[i][j]!=0) {
            if (x!=i) {
              if (cuadradoLatino[x][j]==cuadradoLatino[i][j]) {
                esValido = false;
              }
            }
            if (x!=j) {
              if (cuadradoLatino[i][x]==cuadradoLatino[i][j]) {
                esValido = false;
              }
            }
          }
        }
      }
    }
    return esValido;
  }
```

crearCuadradoLatino simplemente toma un número al azar entre 1 y el lado del cuadrado.

```
  private static int[][] crearCuadradoLatino(int[][] cuadradoLatino, int i, int
j) {
    int temp = (new Random()).nextInt(cuadradoLatino[0].length);
    temp++;
    cuadradoLatino[i][j] = temp;
    return cuadradoLatino;
  }
```

5. Casos de Prueba MiMax

Para un único elemento. El máximo y el mínimo son el mismo.

```
  @Test
public void testMiMaxUnElemento() throws Exception {
  System.out.println("testMiMaxUnElemento");
  int[] v = {5};
  int[] expResult = {5,5};
  int[] result = Algebra.miMax(v, 0, 0);
  assertArrayEquals(expResult, result);
}
```

Para un vector, con un rango de un único elemento, el resultado es ese elemento.

```
@Test
public void testMiMaxUnElementoVector() throws Exception {
  System.out.println("testMiMaxUnElemento");
  int[] v = {5,3,4,5,2,2};
  int[] expResult = {3,3};
  int[] result = Algebra.miMax(v, 1, 1);
  assertArrayEquals(expResult, result);
}
```

Para vector solamente de números negativos.

```
@Test
public void testMiMaxNegativos() throws Exception {
  System.out.println("testMiMaxUnElemento");
  int[] v = {-6,-9,-12,-14,-3,-4,-7};
  int[] expResult = {-14,-3};
  int[] result = Algebra.miMax(v, 0, v.length-1);
  assertArrayEquals(expResult, result);
}
```

Para caso genérico

```
@Test
public void testMiMaxGenerico() throws Exception {
  System.out.println("testMiMaxGenerico");
  int[] v = {6,-9,12,14,-3,4,-7};
  int[] expResult = {-9,14};
  int[] result = Algebra.miMax(v, 0, v.length-1);
  assertArrayEquals(expResult, result);
}
```

Para Mínimo y Máximo en bordes del vector.

```
  @Test
  public void testMiMaxBordeA() throws Exception {
    System.out.println("testMiMaxBordeA");
    int[] v = {-20,-9,12,14,-3,4,30};
    int[] expResult = {-20,30};
    int[] result = Algebra.miMax(v, 0, v.length-1);
    assertArrayEquals(expResult, result);
  }
```

Para Máximo y Mínimo en bordes del vector.

```
  @Test
  public void testMiMaxBordeB() throws Exception {
    System.out.println("testMiMaxBordeB");
    int[] v = {20,-9,12,14,-3,4,-30};
    int[] expResult = {-30,20};
    int[] result = Algebra.miMax(v, 0, v.length-1);
    assertArrayEquals(expResult, result);
  }
```

6. Casos de Prueba Anagrama

Para una única coincidencia de un anagrama

```
/**
 * Test of anagramas method, of class Texto.
 */
@Test
public void testAnagramaUnico() throws Exception {
  System.out.println("testAnagramaUnico");
  String palabra = "ROMA";
  ArrayList<String> v = new ArrayList<String>();
  v.add("AMOR");
  int expResult = 1;
  ArrayList<String> result = Texto.anagramas(palabra, v);
  assertEquals(expResult, result.size());
}
```

Para ninguna coincidencia de un anagrama

```
/**
 * Test of anagramas method, of class Texto.
 */
@Test
public void testAnagramasSinCoincidencia() throws Exception {
  System.out.println("testAnagramasSinCoincidencia");
  String palabra = "ROMA";
  ArrayList<String> v = new ArrayList<String>();
  v.add("PRUEBA");
  int expResult = 0;
  ArrayList<String> result = Texto.anagramas(palabra, v);
  assertEquals(expResult, result.size());
}
```

Para varias coincidencias de un anagrama, en toda la lista

```
/**
 * Test of anagramas method, of class Texto.
 */
@Test
public void testAnagramasTodasCoincidencias() throws Exception {
  System.out.println("testAnagramasTodasCoincidencias");
  String palabra = "ROMA";
```

```
    ArrayList<String> v = new ArrayList<String>();
    v.add("AMOR");
    v.add("ROMA");
    v.add("MORA");
    v.add("RAMO");
    int expResult = 4;
    ArrayList<String> result = Texto.anagramas(palabra, v);
    assertEquals(expResult, result.size());
  }
```

Para algunas coincidencias de un anagrama en la lista

```
  /**
   * Test of anagramas method, of class Texto.
   */
  @Test
  public void testAnagramasAlgunasCoincidencias() throws Exception {
    System.out.println("testAnagramasAlgunasCoincidencias");
    String palabra = "ROMA";
    ArrayList<String> v = new ArrayList<String>();
    v.add("AMOR");
    v.add("PRUEBA");
    v.add("ROMA");
    v.add("RELOJ");
    int expResult = 2;
    ArrayList<String> result = Texto.anagramas(palabra, v);
    assertEquals(expResult, result.size());
  }
```

Para coincidencias independiente de la capitalización de las letras

```
  /**
   * Test of anagramas method, of class Texto.
   */
  @Test
  public void testAnagramasCapitalizacion() throws Exception {
    System.out.println("testAnagramasAlgunasCoincidencias");
    String palabra = "Roma";
    ArrayList<String> v = new ArrayList<String>();
    v.add("amor");
    v.add("rOmA");
    v.add("RaMo");
    int expResult = 3;
    ArrayList<String> result = Texto.anagramas(palabra, v);
```

```
    assertEquals(expResult, result.size());
  }
```

7. Casos de Prueba Cuadrado Latino

La función de Cuadrado Latino intenta construir el mismo por fuerza bruta. Existe la posibilidad que no encuentre un cuadrado latino dado un tiempo de espera razonable. Por lo tanto los casos de prueba fueron omitidos.

Se desea encontrar un algoritmo mas eficiente para generar los mismos, pero no se ha logrado al momento de publicar este documento.