Laboratorio No. 3

Tipos de Algoritmos

(Programación Dinámica, Divide y Vencerás, Algoritmos Probabilísticos)

Introducción

En este laboratorio el/la estudiante deberá desarrollar algoritmos utilizando técnicas de diseño y determinando los tipos de algoritmos. A saber: algoritmos voraces, backtracking, programación dinámica, divide y vencerás, algoritmos probabilísticos

Objetivos

Al finalizar este laboratorio, el/la estudiante deberá ser capaz de:

a. Utilizar la técnica de programación dinámica para desarrollar algoritmos
b. Desarrollar algoritmos del tipo "divide y vencerás"
c. Desarrollar algoritmos probabilísticos
d. Aplicar conocimientos discutidos en clase

Contexto

1. Trabaje con un modelo de n capas (domain, controller, test, util)
2. Cree un nuevo proyecto llamado "Laboratory3" utilizando la tecnología javaFX, la cual permitirá trabajar en un entorno gráfico.
3. Defina una clase llamada "Dynamic" y resuelva los siguientes algoritmos utilizando la técnica de Programación Dinámica. Recuerde seguir los siguientes pasos: (1) determinar la ecuación recurrente, (2) determinar los casos base, (3) definir las tablas, (4) encontrar la solución:
   a. Factorial de un número no recursvio
      public long factorial (int n))
   b. Sucesión de Fibonacci no recursivo
      public long fibonacci (int n))

4. Defina una clase llamada "DivideAndConquer" y resuelva los siguientes algoritmos utilizando la técnica Divide y vencerás:
   a. Búsqueda binaria iterativa
      public int binarySearch (int sortedArray[], int value)
      Uso: Iterative binarySearch (sortedArray, value)
   b. Búsqueda binaria recursiva
      public int binarySearch (int sortedArray[], int value, int low, int high)
      Uso: Recursive binarySearch (sortedArray, value, 0, sortedArray.length

5. Defina una clase llamada "Vector" según lo siguiente:
   Atributos:
   a. public int n; tamaño máximo del vector
   b. public int data []; array de elementos tipo enteros
   c. public int count; cantidad de elementos agregados
   Métodos de la interface VectorList:
   d. **int size();** // devuelve el número de elementos en el vector
   e. **void clear();** //remueve todos los elementos del vector
   f. **boolean isEmpty();** // true si el vector está vacío
   g. **boolean contains(Object element);** //true si el elemento existe
   h. **void add (Object element);** // inserta un elemento al final
   i. **void add(int index, Object element);** //inserta un elemento en la posición indicada
   j. **boolean remove(Object element);** //true si el elemento es suprimido
   k. **Object remove(int index);** //suprime y retorna el elemento
   l. **void sort();** //ordena el vector
   m. **int indexOf(Object element);** //devuelve la posición del elemento (primera ocurrencia)

n. **Object get(int index);** //devuelve el elemento en la posición indicada
o. **toString():** redefine el método toString y muestra el contenido del vector por consola

6. Tome como referencia la clase vector creada anteriormente y defina una clase llamada VectorE genérica (E=element), que permita determinar el tipo de contenido del vector (Integer, String, Character).
7.
8. Compruebe el funcionamiento de la clase "DivideAndConquer" a través de una clase de testeo de la siguiente forma:
   a. Utilice las siguientes clases para comprobar el funcionamiento de los métodos de búsqueda binaria:
      domain.DivideAndConquer.binarySearch (iterative and recursive)
      java.util.Arrays.binarySearch()
      java.util.Collections.binarySearch()
   b. Cree una instancia de la clase Vector con tamaño 50 y llene todo el vector con valores aleatorios entre 0 y 99.
   c. Muestre el contenido del vector por consola (no ordenado y ordenado)
   d. Utilice un bucle "for" hasta 20 para generar valores aleatorios y probar las búsquedas binarias utilizando la instancia del vector creada anteriormente. Un ejemplo de la salida por consola es el siguiente:
      JAVA.UTIL.ARRAYS CLASS BS... The element 65 does not exist in java Arrays
      JAVA.UTIL.COLLECTIONS BS... The element 65 does not exist in java Collections
      ITERATIVE BS... The element 65 does not exist in vector
      RECURSIVE BS... The element 65 does not exist in vector

      JAVA.UTIL.ARRAYS CLASS BS... The element 6 exists at position [5]
      JAVA.UTIL.COLLECTIONS BS... The element 6 exists at position [5]
      ITERATIVE BS... The element 6 exists at position [5]
      RECURSIVE BS... The element 6 exists at position [5]

9. Compruebe el funcionamiento de las clases "Vector" y "VectorE" a través de clase de testeo llamada VectorTest, de la siguiente forma:
   a. Cree un método de testeo para la clase Vector llamado testVector().
   b. Cree e instancie un vector de tamaño 50 y llene el vector con valores aleatorios
   c. Orden el vector con el método "sort" y muestre el contenido por consola con el método toString()
   d. Pruebe los métodos size, isEmpty
   e. Agregue elementos en las siguientes posiciones del vector: 10, 5, 0. Luego muestre por consola el contenido del vector
   f. Utilice un bucle "for" hasta 30 para generar valores aleatorios y probar el método contains.
   g. Elimine el elemento en la posición 10 del vector. Luego agregue el 81 en la posición 10.
   h. Elimine el elemento en la posición 5 y muestre el contenido del vector.
   i. Agregue el 40 en la posición 5, elimine el elemento de la posición 0, agregue el 70 en la posición 0. Al final muestre el contenido del vector por consola.
   j. Utilice un bucle "for" hasta 30 para generar valores aleatorios y probar el método remove.
   k. Muestre por consola el contenido del vector.

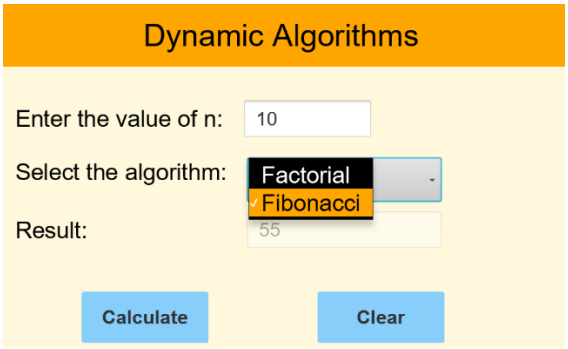
   l. Cree un método de testeo para la clase VectorE llamado testVectorE().
   m. Cree e instancie VectorE<String> vector = new VectorE(50);
   n. Llene el vector en forma aleatoria con nombres de países (se pueden repetir).
   o. Ordene el vector y realice pruebas de los métodos (similar a lo indicado en la parte de arriba para la clase Vector).

10. Define una clase llamada “Probabilistic” y resuelva el siguiente algoritmo utilizando la técnica de algoritmos probabilísticos:
    Problema de la “Paradoja del cumpleaños” (Birthday paradox). Esta paradoja establece que, de un conjunto de 23 personas, hay una probabilidad del 50,7% de que al menos dos personas de ellas cumplan años el mismo día. Para 57 o más personas la probabilidad es mayor del 99,666%
    Fórmula: probability = probability * (366-i) / 365, i=1, 2, ….n
    n = número de personas

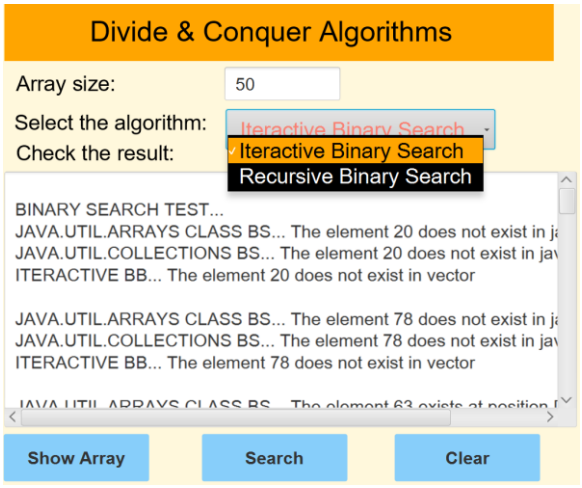
11. Compruebe el funcionamiento de la clase Probabilistic a través de una clase de testeo que permita comprobar la “paradoja del cumpleaños” para las siguientes poblaciones: 30, 23, 57, 10, 85, 5.

12. Utilice la tecnología “javaFX” para crear un entorno gráfico que muestre un menú principal y la solución a los algoritmos anteriores, de la siguiente forma:

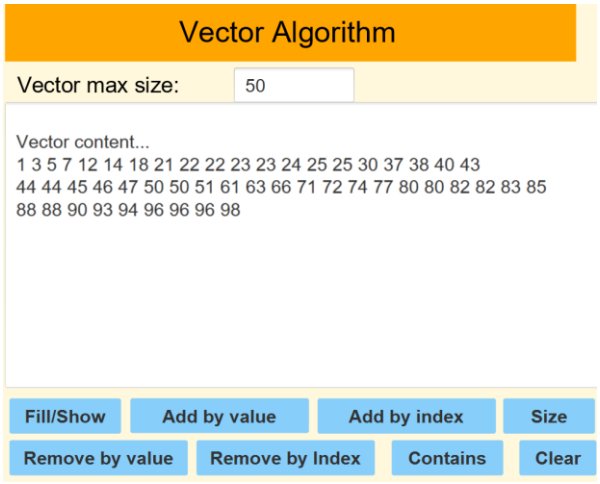
    a. Dynamic: Utilice 2 objetos gráfico tipo TextField y 1 objeto gráfico tipo ChoiceBox (Fibonacci, Factorial) para probar los algoritmos según corresponda.



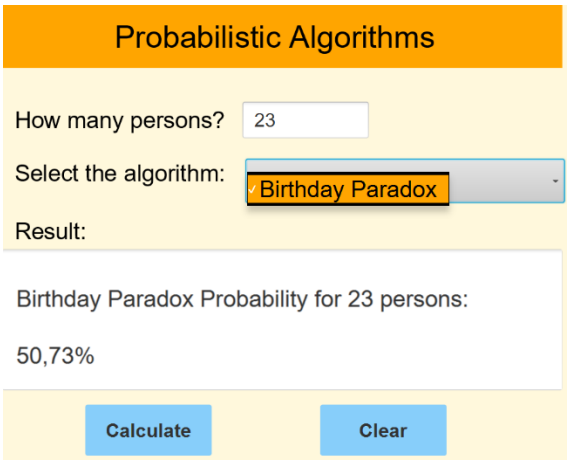
    b. Divide & Conquer: Utilice objetos gráficos tipo TextField y ChoiceBox. El botón “Show Array” permite mostrar el contenido del vector con el tamaño indicado. El botón “Search” permite realizar la búsqueda binaria con los algoritmos java.util.Arrays, java.util.Collections y el algoritmo que el usuario seleccione (iterative, recursive binary search)



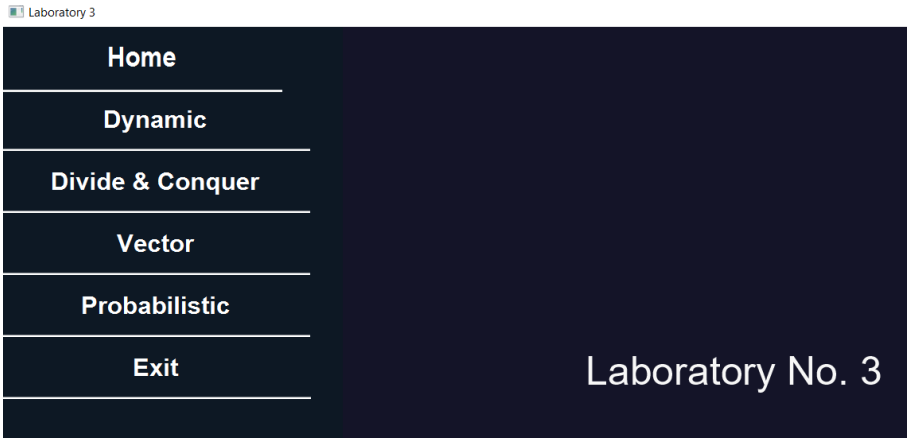
c. Vector



d. Probabilistic



Un ejemplo del menú gráfico es el siguiente:



Resuelva y publique el laboratorio en el entorno del curso de la plataforma de mediación virtual (METICS). Verifique la fecha límite para el envío del informe.
URL: https://mv1.mediacionvirtual.ucr.ac.cr/course/view.php?id=7513