Práctica 2

EDD

Matú Hernández Diana

Rojo Mata Daniel

Recordemos que un algoritmo es una secuencia de pasos que solucionan un problema, el cual debe tener:

- Especificación de entrada
- Especificación de cada instrucción
- Integridad
- Número finito de pasos
- Terminación
- Descripción de resultado

Veamos cada caso:

```
Algoritmo esVacio
Entrada: Conjunto (Arreglo)
Salida: boolean

if (this.elementos.length == 0) {
    return true;
    } else {
    return false;
    }
```

```
Algoritmo pertenencia
Entrada: Conjunto (Arreglo) y un entero x
Salida: boolean

for (int i = 0; i < this.elementos.length; i++) {
    if (this.elementos[i] == x) {
        return true;
    }
}
    return false;
    }
```

```
Algoritmo subConjunto
Entrada: 2 conjuntos (Arreglos)
Salida: boolean

int cardinalidad_1 = c.elementos.length;
int cardinalidad_2 = this.elementos.length;

if (cardinalidad_2 <= cardinalidad_1) {
  for (int i = 0; i < cardinalidad_2; i++) {
    this.pertenencia(i);
  }
  return true;
} else {
  return false;
}
```

```
Algoritmo diferencia
```

```
Entrada: 2 conjuntos (Arreglos)
Salida: Conjunto (Arreglo)

int[] conjunto_prueba = new int[this.elementos.length];
int tamanio = 0;

for (int elemento : this.elementos) {
  if (!c.pertenencia(elemento)) {
    conjunto_prueba[tamanio] = elemento;
    tamanio++;
  }
}

int[] conjunto_diferencia = new int[tamanio];

for (int i = 0; i < tamanio; i++) {
  conjunto_diferencia[i] = conjunto_prueba[i];
}

return new Conjunto(conjunto_diferencia);
```

```
Algoritmo unión
Entrada: 2 conjuntos (Arreglos)
Salida: Conjunto (Arreglo)

int tamanio_1 = c.elementos.length;
int tamanio_2 = this.elementos.length;
int tamanio = tamanio_1 + tamanio_2;
int[] conjunto_union = new int[tamanio];

```

```
for (int i = 0; i < tamanio_1; i++) {
  conjunto_union[i] = c.elementos[i];
}
for (int i = tamanio_1; i < tamanio; i++) {
  conjunto_union[i] = this.elementos[i - tamanio_1];
}
return new Conjunto(conjunto_union);
```

```
Algoritmo intersección
Entrada: 2 conjuntos (Arreglos)
Salida: Conjunto (Arreglo)

int[] conjunto_prueba = new int[this.elementos.length];
int tamanio = 0;
for (int elemento : this.elementos) {
    if (c.pertenencia(elemento)) {
        conjunto_prueba[tamanio] = elemento;
            tamanio++;
          }
   }

int[] conjunto_final = new int[tamanio];
for (int i = 0; i < tamanio; i++) {
  conjunto_final[i] = conjunto_prueba[i];
}

return new Conjunto(conjunto_final);
```

Como podemos ver, cada algoritmo tiene entrada, salida y pasos específicos bien definidos, además tenemos la seguridad de que son funcionales pues resuelven los problemas dados retornando lo necesario (ejecútese el archivo .java) y puesto que se está trabajando con estructuras finitas, es decir, que presentan un número finito de elementos, los algoritmos operan en un número finito de pasos, en especial para los que ejecutan algún tipo de ciclo, esto último garantiza su terminación pues no hay bucles sin salida.

Por lo tanto, cada uno de ellos satisface las 6 características que deben tener los algoritmos así que estos son correctos.