Ejercicio 6 – Programación Dinámica con Reducción

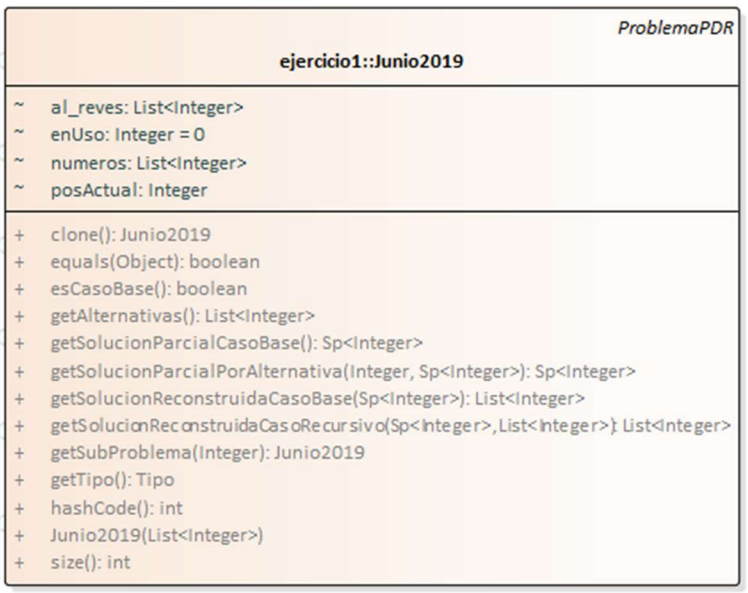
Se tiene una lista de números enteros mayores o iguales que 0, donde cada número representa el número máximo de saltos hacia la derecha que se pueden realizar desde ese elemento. Un índice I, desea viajar desde el inicio de la lista hasta su final.
Se desea calcular la secuencia que incluya el número mínimo de saltos necesarios para que I llegue al final de la lista (a partir del primer elemento).
La lista contiene portales indicados por el número 0. Cuando I aterriza en un portal, viaja a una realidad alternativa e invertida en la que la lista contiene los mismos números en orden inverso. En esta realidad alternativa I continuará viajando hacia la derecha hasta el final de la lista y, si cae en otro portal, volverá a la lista original.
En el siguiente ejemplo, si I aterriza en un portal de la lista Normal, aparece en la lista Al revés (justo en la posición de la flecha) y continúa avanzando por dicha lista hacia la derecha.

```
                ↓
  Normal: 2,4,2,0,2,3,2,4,3,3,4
Al revés: 4,3,3,4,2,3,2,0,2,4,2
                  ↑
```

Si I llega al final de la lista en la realidad alternativa, entonces I estará atrapado para siempre en esta realidad y el problema no tendrá solución.

SE PIDE:

a) Implemente el método public Tipo getTipo().
b) Implemente el método public boolean esCasoBase().
c) Implemente el método Sp<Integer> getSolucionParcialCasoBase()
d) Implemente el método public List<Integer> getAlternativas().



Solución:

```
@Override
public Tipo getTipo() {
	return Tipo.Min;
}


@Override
public boolean esCasoBase() {
	return ( (this.numeros.size()-1) == this.posActual );

}


@Override
public Sp<Integer> getSolucionParcialCasoBase() {
	if (this.enUso == 0)
		return Sp.create(0, .0);
	return null;
}



@Override
public List<Integer> getAlternativas() {
	List<Integer> alternativas = new ArrayList<Integer>();
	List<Integer> aUsar;
	if (enUso == 0)
		aUsar = this.numeros;
	else
		aUsar = this.al_reves;

	if (aUsar.get(this.posActual) == 0) {
		if (enUso == 0) {
			enUso = 1;
			aUsar = this.al_reves;
		}
		else {
			enUso = 0;
			aUsar = this.numeros;
		}
	}

	for (int i= 1; i <= aUsar.get(this.posActual); i++) {
		if ((this.posActual+i) == aUsar.size()) {
			break;
		}
		alternativas.add(i);
	}
	return alternativas;
}
```