Ejercicio 4 – Backtracking

Un coche eléctrico quiere viajar a una ciudad europea y para ello ha de pararse a recargar en cargadores eléctricos a lo largo del camino. Sin embargo, el dueño solo va a esperar 10 minutos en cada uno de los cargadores.
Los cargadores son de distinta potencia, por lo que algunos pueden cargar más el coche en esos 10 min, y por tanto llegará más lejos, y otros cargarán menos el coche.
En *List<Integer> cargadores* se guarda todos los cargadores del viaje, junto con la carga que le transmiten al coche. Por ejemplo, una carga de 1 permitirá al coche ir al siguiente cargador, una carga de 3 permitirá al coche ir a cualquiera de los tres siguientes cargadores y una carga de 0 significa que en 10 minutos no se puede cargar lo suficiente al coche para llegar al siguiente cargador y se queda abandonado en medio de la carretera.
Utilice Backtracking para descubrir cuál es el mínimo tiempo total que el coche debe permanecer cargando para llegar a su destino. Por ejemplo, si el mínimo para completar el viaje es parar en cuatro cargadores, el tiempo mínimo de carga será de 40 minutos.
Los posibles atributos de la clase *EstadoViajePorEuropa* que implementa la interfaz *EstadoBT* son:

- *List<Integer> cargadores* se guarda todos los cargadores del viaje.
- *Integer cargador* indica cuál es el cargador actual
- *Integer tiempoAcumulado* indica cuál es el tiempo actual que el coche ha estado cargando.

SE PIDE:

a) Implemente el método public Tipo getTipo().
b) Implemente el método public boolean esCasoBase().
c) Implemente el método CargaCocheElectricoBT avanza(Integer a)
d) Implemente el método public List<Integer> getAlternativas().
e) Implemente el método **public** Double getObjetivo()

```
public class CargaCocheElectricoBT implements EstadoBT<Integer, Integer,
CargaCocheElectricoBT> {
	List<Integer> solucion;
	List<Integer> conjuntoNumeros;
	int index;

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

	@Override
	public CargaCocheElectricoBT getEstadoInicial() {
		return new CargaCocheElectricoBT(this.conjuntoNumeros);
	}

	@Override
	public CargaCocheElectricoBT avanza(Integer a) {
		Integer next = index + a;
		if (next >= this.conjuntoNumeros.size()) {
			this.solucion.add(this.conjuntoNumeros.size()-1);
		} else {

			this.solucion.add(next);
		}
		index = next;
		return this;
	}

	@Override
	public CargaCocheElectricoBT retrocede(Integer a) {
		this.index-=a;
		this.solucion.remove(this.solucion.size()-1);
		return this;
	}

	@Override
	public boolean esCasoBase() {
			return this.index >= this.conjuntoNumeros.size();
	}

	@Override
	public List<Integer> getAlternativas() {
		List<Integer> alt = new ArrayList<>();
		Integer val = this.conjuntoNumeros.get(this.index);
		for (int i = 1; i <= val; i++) {
			alt.add(i);
		}

		return alt;
	}

	@Override
	public Double getObjetivo() {
		return (double)this.solucion.size();
	}
```