COMPONENTES DEL EQUIPO –> B1_06

- Dídimo Javier Negro Castellanos
- Óscar Jiménez Jiménez
- Jorge Alberto Gómez León

NOMBRE DEL REPOSITORIO -> B1_06

https://github.com/didimoj/B1_06

HITO 4 – JUSTIFICACIÓN DE LA TAREA REALIZADA

En este último hito, añadimos la creación de la heurística asociada a un estado. Además, implementamos el mecanismo de poda del árbol de búsqueda, para que si encontramos un estado cuyo nodo tiene menor valoración añadamos este nuevo y borremos el antiguo (el cual tiene una mayor valoración). De otro modo, si encontramos un estado cuyo nodo tiene una mayor valoración directamente no hacemos nada con él, ni lo insertamos en la frontera.

Con respecto a la poda, hemos implementado una tabla hash para identificar cada nodo y, así, saber cuando son el mismo nodo y realizar la técnica de poda sobre uno de ellos (el que mayor valoración tenga). Código:

```
Hashtable<String, Nodo> visitados = new Hashtable<String, Nodo>();
```

Adicionalmente, implementamos una clase (getId) con la cual, pasándole como parámetro el estado del nodo, genera una clave hash, con la que creamos el id, a partir de la posición del tractor y las distribuciones en el terreno. Código:

```
public String getId(Estado estado) throws NoSuchAlgorithmException {
    String clear = estado.toString();
    MessageDigest md = MessageDigest.getInstance("MD5");
    byte[] b = md.digest(clear.getBytes());

    int size = b.length;
    StringBuffer h = new StringBuffer(size);
    for (int i = 0; i < size; i++) {
        int u = b[i] & 255;
        if (u < 16) {
            h.append("0" + Integer.toHexString(u));
        } else {
            h.append(Integer.toHexString(u));
        }
    }
    return h.toString();

}
```

Por último, hemos añadido a nuestro problema la estrategia A*, además de las que ya implementábamos (anchura, profundidad, voraz).