LABORATORY ASSIGNMENT

Subtask 1

LABORATORY GROUP: A1 – 07

COMPONENTS:

AMANDA SÁNCHEZ GARCÍA

FERNANDO VELASCO ALBA

GITHUB REPOSITORY: A1-07

11/10/2017

INDEX

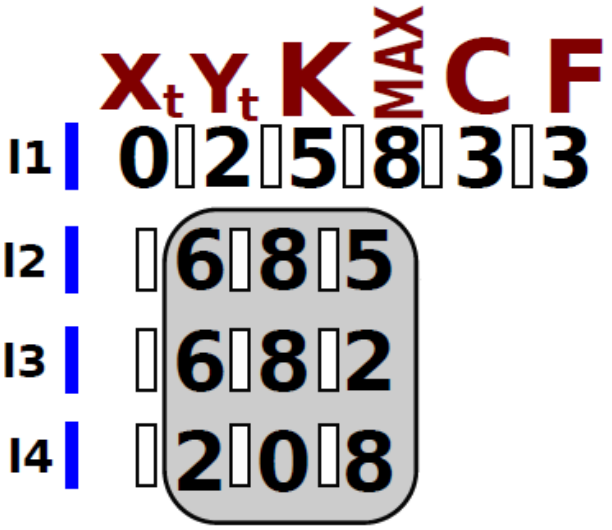
I. A BRIEF RESUME OF THE PROBLEM

The main goal of this laboratory assignment consists of defining, designing and developing an agent program to find the sequence of actions to be taken by a tractor to ensure that all the sand in a field is evenly distributed on the ground. So, all the boxes will have an equal amount of sand K.

The first things to be done are:

- Implement an internal representation of the field.
- Create a field.
- Reading and writing a field from/to a file.
- Generate all possible actions from a field with the tractor in the (X_y, Y_t) box.
- Get a new field after applying an action to a given one.

It is necessary to take into account the format of the file where the provided information is going to be:



2. OUR IMPLEMENTATION OF THE PROBLEM

2.1 PROGRAMMING LANGUAGE CHOSEN

The programming language that we are going to use is Java. We have decided to use it because it is the programming language that we know best, also Java is a very complete language so we will have available all the data structures we are going to need.

2.2 CLASSES IMPLEMENTED

The field is going to be represented as a bidimensional array which boundaries are defined through the file. The file defines the position (x,y) of the tractor in the field, the desired quantity of sand in each square (k), the maximum sand that can be placed in a square, and the number of columns and rows of the field.

We have implemented five classes:

- **Tractor**. Where the tractor is defined, with the current sand it's carrying, and its position in the field (row, column). The method which identifies the possible movements from the current position is implemented in this class.
- **Field.** Definition of the field where the tractor is in, providing the number of rows and columns, the desired amount of sand for each square and the maximum sand permitted per square.
- **FileHandler.** This class is used to manage the file, that is, to have the possibility to read it and write on it. It checks that the file is read correctly, taking into account the format, throwing the corresponding errors.
- **InputExceptions.** Class to check that the format of the file is the correct one (checking that there are only positive integers, the correct use of blank spaces or other error).
- **MainClass.** Main class of the program. Here the field is created and the file is read. The invocation of the principal methods is done in the main class too.
- **Action.** Class where an action is defined. The main attributes of this class are the next movement of the tractor and the quantity of sand to be redistributed in each possible move. This class implements the method where all the possible actions that can be taken are identified.

2.3 MAIN METHODS

- MOVE TRACTOR

```
public static List<Movement> moveTractor (Tractor t, Field f) {
    List <Movement> pos_moves = new ArrayList <Movement>();

    if (0 <= t.getX() && t.getX() < f.getN_rows()) {
        if(t.getX() + 1 < f.getN_rows()) {
            Movement s = new Movement (t.getX() +1, t.getY());
            pos_moves.add(s);
        }

        if(t.getX() - 1 >= 0) {
            Movement n = new Movement(t.getX() -1, t.getY());
            pos_moves.add(n);
        }
    }

    if (0 <= t.getY() && t.getY() < f.getN_cols()) {
        if(t.getY() + 1 < f.getN_cols()) {
            Movement e = new Movement(t.getX(), t.getY()+1);
            pos_moves.add(e);
        }

        if(t.getY() - 1 >= 0) {
            Movement w = new Movement (t.getX(), t.getY()-1);
            pos_moves.add(w);
        }
    }

    return pos_moves;
}
```

The method moveTractor is located in the class Tractor. The main goal of the method is to generate a list with all the possible movements the tractor can perform. The next movement of the tractor depends on the current position of it. This is, the tractor can only moved within the field. If the next position is out of the boundaries of the field, it will not be considered as a movement.

Diagonal movements are not considered, so the tractor can only moved to the right or to the left, and to the north or to the south.

- DECIDE ACTIONS

```
public static List<Action> decideActions(Tractor t, Field f) {
    List<Movement> moves = Tractor.moveTractor (t,f);
    List<Action> actions = new ArrayList<Action>();
    int n_combs = f.getMax()*1000 + f.getMax()*100 + f.getMax()*10 + f.getMax();
    List <int[]> combinations = new ArrayList<int[]> ();
    int [][] field = f.getField();
    int [] aux = new int[4];

    for (int i=0; i<=n_combs; i++) {
        combinations.add(generate_combinations(i));
    }

    for (int i=0; i<moves.size(); i++) {
        for (int j=0; j<combinations.size(); j++) {
            int sum_sand = 0;

            for (int s=0; s<4; s++) {
                aux = combinations.get(j);
                sum_sand = sum_sand + aux[s];
            }

            t.setCurrent_sand(field[t.getX()][t.getY()] - f.getK());
            int sand = t.getCurrent_sand();

            if(sum_sand==sand && sum_sand>0) {
                Movement mv = new Movement (moves.get(i).getX(), moves.get(i).getY());
                Action ac = new Action (mv, aux[0], aux[1], aux[2], aux[3]);
                actions.add(ac);
            }
        }
    }

    return actions;
}
```

The most important method is in Action class and it is called decideActions. This method is crucial because the distribution of the sand is decided through the actions considered in this method.

In order to generate all the possible actions, we have to take into account the number of possible combinations. For example, if the maximum quantity of sand for each box is 8 we will use 8888 possible combinations, checking later that a higher number will not be used.

Let’s explain this graphically:

N	S	E	W
8	8	8	8

The maximum quantity of sand to be moved for each position is that one, but we have to assure that the next situation is not going to happen:

N	S	E	W
8	8	7	9

As we have 8888 possible combinations, this combination can take place, so we have to control that 9 quantities of sand are not going to be moved.

We have to check the possible combinations of sand movements for each new position. To select the correct combinations, we have to compare the total amount of sand to be redistributed. This figure has to be equal to the quantity of sand removed from the initial position of the tractor. The sand removed from this position is the initial amount of sand minus the desired amount of sand in each box, called K.

If this condition takes place, we generate a new action. An action is composed of the next movement of the tractor and the sand for each adjacent box.

Once the action is created, we add it to the list of all possible actions.