Project TI301 - Algorithms and data structures 2

P2 2024-2025

Rover Run!

Background to the assignment

2028: A very intense solar storm has just hit... The MARC - MArs Rover Cartograph rover, which was carrying out its mission very well, has suffered a malfunction. Its programming, guidance and movement systems have been seriously affected...

The ESTF, Efrei Space Task Force, has called on programming experts to design new travel software for MARC.

Your mission is to use this travel software to bring MARC back to a base station where it can be refurbished.

The MARC rover

MARC is a slightly rustic rover, running a Linux system, and all its programs are written in C language.

It can be identified by its position, and the direction in which it is facing, from North, South, East and West (even on Mars). Fortunately, its radar and gyroscope are still working, allowing it to find its bearings.

Displacement system

MARC can perform the following movements (these are unitary movements)

- move forward 10m
- move forward 20m
- move forward 30m
- move back 10m (without turning)
- turn a quarter turn to the left
- turn a quarter turn to the right
- turn back.(U turn)

Damage to its systems means that it now moves in phases, so it cannot go directly from some point A to another point B.

For each phase, MARC will have 9 possible moves (drawn at random from table 1 - probability of available moves), and will have to choose **exactly** 5 of them to complete that phase. We therefore need to programme the "best possible choice" of move.

Type of unitary movement	Probability of draw
Move forward 10 m	22%
Move forward 20 m	15%
Move forward 30 m	7 %
Reverse 10 m	7 %
Turn a quarter turn to the left	21 %
Turn a quarter turn to the right	21 %
Turning back	7 %
TOTAL	100 %

The random draw programme is buggy - each time a movement is chosen, its probability of being drawn decreases by 1%.

Table 1 - Probability of drawing a movement

The procedure for drawing the 9 movements is as follows:

```
Repeat 9 times
    Draw a movement at random
    Reduce the probability of this movement being drawn by 1%.
```

The draw probabilities return to their initial values at the start of each phase.

Travelling to Mars

Since MARC only moves on Martian soil, the map of Mars is a two-dimensional space: this map is made up of 10 m x 10 m zones, which can be represented by a square 'box'. A square is identified by x and y coordinates on the map.

The Martian landscape

The composition and relief of the Martian soil influence the rover's movements on its surface. The types of soil identified are :

Plain: no influence on movement

Martian Erg: loose, dusty soil. **If MARC starts his move from an Erg square**, his next move is reduced by 1: moving forward 10m and moving backwards do nothing, moving forward 20m only moves forward 10m, moving forward 30m only moves forward 20m. You can't turn by a quarter-turn; a half-turn turns you either left or right by a quarter-turn.

Martian Reg: uneven ground. **If MARC finishes a move on a Reg square**, he has been shaken up and his software works less well. He will only be allowed 4 moves in total for the next phase.

Crevasse: not a very recommended area for rovers: **if MARC passes over a crevasse square**, he falls in and ends his rover life, without respawn.

Slope - the square moves downhill in a given direction: if MARC ends his move on a slope square, he is then moved one square in the direction of the downhill slope.

You can combine **slope+plain**, **slope+erg** (it's a dune in fact), **slope+reg**

Choosing the best route

On the map of Mars, the base station will be located at predefined coordinates x_b and $y_b$the MARC departure point will be located at arbitrary coordinates x_{dep} and y_{dep}.

Your mission is to ensure that MARC reaches the base station, and then to minimise the number of journeys used.

The best route is chosen by minimising a cost function.

Route cost function - this part is supplied to you

The cost function will be pre-calculated on the map.

Each square will have a value (a cost) associated with it, the lower the value the more 'interesting' the square.

For example, and by convention: the square where the base station is located will be worth 0 (the lowest possible cost, so very interesting), and the crevasse squares will be associated with a very high cost (for example: 10,000).

The MARC guidance programme will therefore seek to reach the minimum value boxes along its route.

Principle for calculating the cost function - step 1

1. Set the cost of all the boxes to a large value (e.g. 65,535).
2. Set the cost of the base station box to 0.
3. Starting from the base station (value 0), update its 4 neighbours (as MARC does not move diagonally), adding a cost according to the nature of the square (plain, erg, reg).

The final cost of a square is then :

cost associated with the type of square + min(costs of its 4 neighbours).

For example, a plain is more accessible and manoeuvrable than a reg, which is itself more 'interesting' than an erg. Here's an example of costs (you can change these values)

Type of box	plain	reg	erg	crevasse
Associated cost	1	2	4	10 000

Illustration:

The types of terrain are represented by the following drawings

We are interested in the following map, which also shows the base station and the initial position of MARC (on a plain). On the right, the initial costs:

65535	65535	65535	65535	65535	65535
65535	65535	0	65535	65535	65535
65535	65535	65535	65535	65535	65535
65535	65535	65535	65535	65535	65535
65535	65535	65535	65535	65535	65535
65535	65535	65535	65535	65535	65535
65535	65535	65535	65535	65535	65535

Step 1: base neighbours Step 2: neighbours of neighbours

65535	65535	4	65535	65535	65535
65535	1	0	1	65535	65535
65535	65535	1	65535	65535	65535
65535	65535	65535	65535	65535	65535
65535	65535	65535	65535	65535	65535
65535	65535	65535	65535	65535	65535
65535	65535	65535	65535	65535	65535

65535	5	4	2	65535	65535
5	1	0	1	3	65535
65535	2	1	3	65535	65535
65535	65535	10001	65535	65535	65535
65535	65535	65535	65535	65535	65535
65535	65535	65535	65535	65535	65535
65535	65535	65535	65535	65535	65535

Final status of stage 1

7	5	4	2	10002	7
5	1	0	1	3	5
3	2	1	3	4	5
6	3	10001	5	6	6
10	7	10003	7	7	7
12	8	10004	8	8	8
16	12	10007	10008	9	9

Principle for calculating the cost function - step 2

For squares with a cost greater than 10,000 that **are not crevasses** (numbers in red in the table), repeat the same calculation, starting with the lowest cost: the cost of the square is recalculated according to its nature + the minimum cost of its 4 neighbours.

6	5	4	2	10002	7
5	1	0	1	3	5
3	2	1	3	4	5
6	3	10001	5	6	6
10	7	10003	7	7	7
12	8	10004	8	8	8
16	12	10007	10008	9	9

➡

6	5	4	2	10002	7
5	1	0	1	3	5
3	2	1	3	4	5
6	3	10001	5	6	6
10	7	8	7	7	7
12	8	9	8	8	8
16	12	13	10008	9	9

Phase displacement selection - this part is not supplied

A phase therefore consists of 5 moves (4 in the case of a **Reg** square) chosen by MARC.

To make the 'best' possible choice for a phase, we use an **N-ary tree**, with each level representing one of the phase's moves. Level 0 (the root) is not a move, but is used to start the tree. The 'optimal' path in relation to a given phase will be the leaf with the minimum value in this tree.

N-ary tree structure

For level 1 (choice of first movement), there are 9 possibilities.

For level 2 (choice of the second movement), there are 8 possibilities for each first choice made, i.e. 8*9 nodes.)

For level 3, we have 7 possibilities for each node in the previous level, i.e. 7*8* nodes...

In all, we have: 1 + 9 +9*8 +9*8*7+9*8*7*6+9*8*7*6*5 nodes = 18,730 nodes (of which 9*8*7*6*5 = 1,520 leaves).

Values stored in nodes

The values stored in this tree will be: for the root, the value of the square from which the MARC movement phase starts. For each descendant of a node, the value is that of the square reached after application of the movement associated with the chosen move.

Method number 1- compulsory

Construct the corresponding integer tree. Find, among the leaves, the minimum value obtained. Find the leaf (or leaves) that stores this value. Find (by a means to be defined by yourself) the sequence of movements associated with this leaf.

Method number 2 - optional

Take the 'most promising' path first: rather than building the whole tree, the choices are made from the node with the lowest value found at a given moment. Eventually, a leaf with the minimum value is found. Find the sequence of movements associated with this leaf (using a method to be defined by yourself).

Exceptional cases - stop before end of phase

If, at the end of any movement (so not necessarily at the end of the phase), MARC arrives at the base station, you can stop the phase (if you find a node with a value of 0, you've won!).

If MARC 'goes off' the map, we lose contact with it, and it ends its life as a rover on Mars.

Update of MARC's position and orientation following choice of movements - MARC's updated coordinates are provided

Apply the chosen sequence of movements to update the position and orientation of MARC.

Illustrations for building the tree

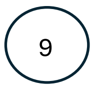
To simplify, the example will be taken for a choice of 3 movements out of 5. The starting situation is that of the following card (already used as an example). MARC is positioned at the bottom right of the card and faces 'NORTH', towards the top of the card. His starting square has a value of 9.

6	5	4	2	10002	7
5	1	0	1	3	5
3	2	1	3	4	5
6	3	10001	5	6	6
10	7	8	7	7	7
12	8	9	8	8	8
16	12	13	10008	(M ↑) 9	9

Random movements available

10 m forward
Turn Left
Turn Right
20 m forward
10 m backward

Initial state of the tree: the root contains the starting value



First choice of move: 5 possibilities. The left-most son corresponds to the score of the square reached by the 'Move 10 m' move, i.e. the square worth 8

Note:☠ will be worth 10,000, as for a crevasse. So, in the tree, if the value of the node is greater than 9999, we don't go any lower for this node.

Second choice: illustration starting from node '8': the 'Move forward 10 m' move has been used. Try it with the 4 remaining moves.

After 5 choices, you will have constructed all the possible move sequences. The best sequence(s) will be the one(s) that lead(s) to the square with the lowest cost.

A leaf with a minimum value must therefore be selected.

Then, from this leaf, you need to determine the path taken from the root of the tree to get there, which will give you the sequence of movements to perform.

To succeed in your mission

Proceed in successive stages

Stage 1- mandatory

- Test the code provided for MARC maps and individual movements;
- Define the data structures for the trip selection tree;
- Start simple: create a tree with 3 choices from 4 or 5 possible values, to fine-tune the algorithms for **building** the tree, **finding a leaf with the minimum value**, and the **path from the root to that leaf**.

Stage 2 - mandatory

Scaling up: gradually increase the number of choices to be made, and the number of possible values, until you arrive at a choice of 5 movements from 9 possibilities.

Complexity: indicate the execution time of your programme for

- The tree construction phase ;
- The search for a leaf with the minimum value among all the leaves in the tree ;
- Calculating the path from the root to this leaf ;
- A complete example of MARC guidance from its home position to the base station.

Make up your own geographical maps of Mars to see how MARC gets on, thanks to your program

Stage 3 - optional

Complexity study: now we want to find the best compromise between time and performance for the MARC route. Continue to increase the number of movement choices for a phase, as well as the number of possible choices.

Indicate how long it takes to calculate the algorithms as the tree grows

Indicate whether these increases give better results for the route taken by MARC on a map.

Ideas for extensions: implement slopes for terrains, add new terrain types, improve the graphical interface.