CS 330 Movement exercises

For these exercises use the ‘graph paper’ from the ‘cellNumberSheet.docx’ file.

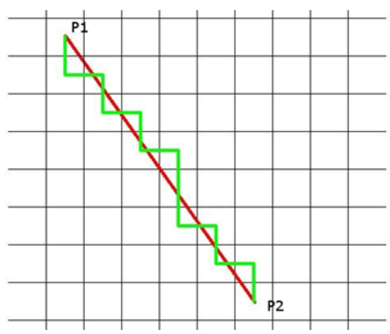
We will assume that each cell in the table is a location. Note that for our current purposes each cell could be considered to be 0.25 inches on each side, 1.0 inches on each side or vanishingly small on each side. The representation on the graph paper is built on cells that are 0.25 inches on a side.

Exercise 1

Let’s start with a simple exercise. You have to move one object from the (0,0) cell to the (37,28) cell. We will continue to designate cells from the top and left. The origin of the user space is the upper-left corner of the component’s occupied area. The *x* coordinate increases to the right and the *y* coordinate increases downward. Thus a position is labeled as (y, x) the upper left coordinate. (If you prefer, it can be called row –column order.) We will assume for this exercise that the object to be moved covers one cell and is only one cell in area.

We will assume that P1 is (0,0) and P2 is (37,28).

Now, how can the object move? This will determine the path. We could use straight line Euclidian movement (red line) or the block-wise Manhattan movement (green line) for 4-directions movement. (You might want to try 8 directions of movement as well.)



Consider two points. P1 is (a,b) and P2 is (c,d). The Euclidian distance between P1 and P2 is

$$D = \sqrt{(a-c)^2 + (b-d)^2}$$

The Manhattan distance is different The Manhattan distance between P1 and P2 is

$$D = |a-c| + |b-d|$$

So now we have an idea that the distance between the points will be different. We also know that if the object could move in a Euclidian way that the problem would be ‘easy’. Count how many cells would be visited using the Euclidian path. Now, generate a Manhattan path from P1 to P2. In fact generate three paths. The first path moves along the x axis and then drops down the y axis. Path 2 goes down the y axis and then across the x axis. The third path approximates the diagonal path used in the Euclidian version. For each of the Manhattan paths, how many cells are visited? What is the length of the path?

Exercise 2

Let us suppose that we have a programing language where there is an instruction similar to the following:

```
move_object (object, next_cell);
```

or in a more object oriented way

```
<object>.move_object (next_cell);
```

The `object` is the object that we are moving and the `next_cell` is the cell to move to. The `move_object` function/method only allows the object to move one cell at a time. Sketch a program (pseudo code) that would move the object in a Euclidian way and a Manhattan way. Feel free to add any reasonable data structures and helper methods.

Exercise 3

This one is similar to the first two except for one significant difference. There are two impenetrable blocks. These are cells that cannot be penetrated by the object you are moving. The first barrier extends from (30,10) to (30,28) and the second barrier extends from (5,0) to (5,15). Determine the path that would go from P1 to P2. Record the steps in solving the problem. How many cells were visited? Use the `move_object` function/method and the steps you recorded to generate pseudocode for your solution.

Exercise 4 (shared exercise)

In this exercise there is a barrier. However, unlike exercise 3, your first task is to move around the barrier and return to the place that you started, which is again P1. So if the barrier is a circle, you would want to travel along the circumference without entering the area inside of the circle. Let's think of this as a collection of stacked barriers with the following start and end cells:

(14,10) to (14,20)

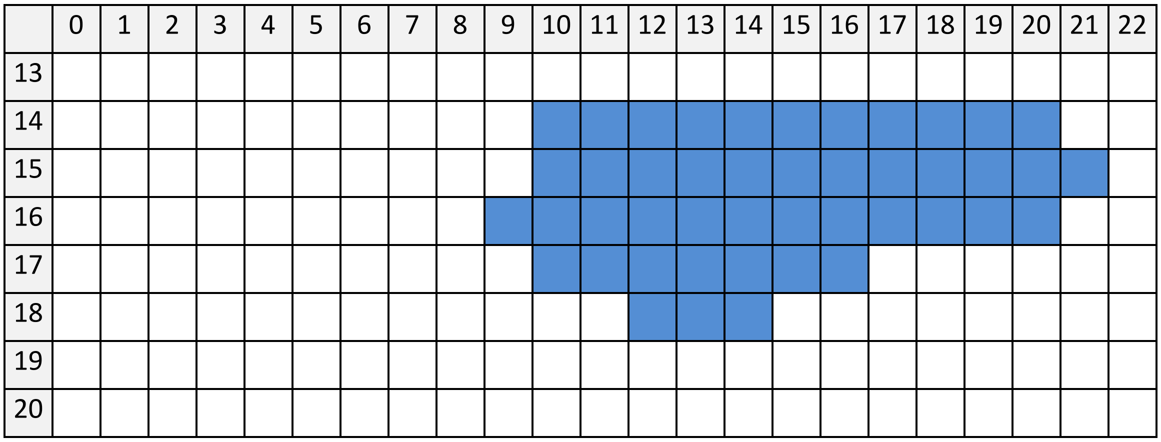
(15,10) to (15,21)

(16,9) to (16,20)

(17,10) to (17,16)

(18,12) to (18,14)

This would produce the excerpted portion of the table illustrated below. So you would want to go from (0,0) to, perhaps, (13,9) and then around the blue cells back to (13,9) and then to (0,0).



Write down the steps in the procedure that you would use to do this. Count the number of cells visited. Use the `move_object` function/method and the steps you recorded to generate pseudocode for your solution. Your second task is to determine the steps and the pseudo code to move from P1 to P2 but avoid the barrier. There will be more than one solution. Why did you pick one rather than another? What did you need to know to pick one rather than another?