Kurt Dudek
U11108612

Homework 1

ME570
September 14, 2019

Problem 1

1.1

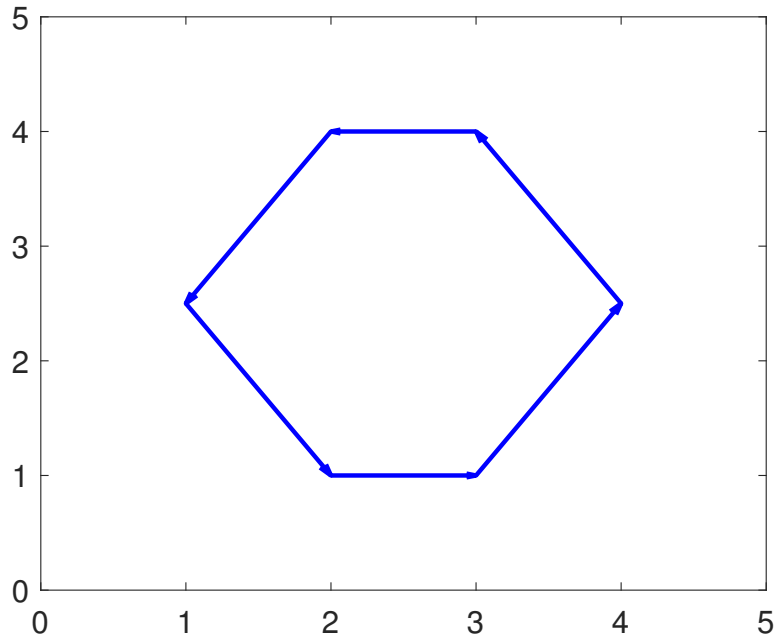

Figure 1: Polygon plotted using polygon_plot()

1.2

In the function *edge_angle()* cangle represents the cosine and sangle represents the sine. We began with three vertices

$$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \quad \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \quad \begin{bmatrix} x_2 \\ y_2 \end{bmatrix}$$

We now create vectors with v_0 as the vertex in common

$$\vec{v_2} = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} - \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \tag{1}$$

$$\vec{v_1} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} - \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \tag{2}$$

Using matlab's *norm()* function, the magnitude of the vectors are computed

$$|v_1| \ = norm(\vec{v_1}) \tag{3}$$

$$|v_2| \ = norm(\vec{v_2}) \tag{4}$$

Equations for the dot product and cross product will now be defined as the next few steps make use of them to fully explain the meaning of *cangle* and *sangle*

$$\vec{a} \bullet \vec{b} = |a| \ |b| \ cos(\theta) \tag{5}$$

$$\vec{a}x\vec{b} = |a| \ |b| \ sin(\theta) \tag{6}$$

If we rearrange the equation to give it in terms of unit vectors, it resembles the steps taken in *edge_angle()*

$$\frac{\vec{a} \bullet \vec{b}}{|a| \ |b|} = \vec{e_a} \bullet \vec{e_b} = cos(\theta) \tag{7}$$

$$\frac{\vec{a}x\vec{b}}{|a| \ |b|} = \vec{e_a}x\vec{e_b} = sin(\theta) \tag{8}$$

The unit vectors are computed in lines 26 and 27 by diving $\vec{v_1}$ and $\vec{v_2}$ by their magnitudes. The expressions are now equal to just $sin(\theta)$ and $cos(\theta)$ and are computed with the following operations

$$cangle = cos(\theta) = \begin{bmatrix} e_{v1x} & e_{v1y} \end{bmatrix} x \begin{bmatrix} e_{v2x} \\ e_{v2y} \end{bmatrix} \tag{9}$$

$$sangle = sin(\theta) = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} cross(\begin{bmatrix} e_{v1x} \\ e_{v1y} \\ 0 \end{bmatrix}, \begin{bmatrix} e_{v2x} \\ e_{v2y} \\ 0 \end{bmatrix}) \tag{10}$$

The matlab function *cross()* computes the cross product of the provided vectors. The cosine and sine of theta have been computed which allows us to solve for theta using the expression

$$tan(\theta) = \frac{sin(\theta)}{cos(\theta)} = \frac{sangle}{cangle} \tag{11}$$

and in the function, the last operation is edgeAngle=atan2(sAngle,cAngle) which equates to

$$\theta = tan^{-1}(\frac{sin(\theta)}{cos(\theta)}) = tan^{-1}(\frac{sangle}{cangle}) \tag{12}$$

The unsigned angle between the two vectors has now been computed. The last operation in the function is to convert the unsigned angle from $[0, 2\pi]$ to an angle from $[-\pi, \pi]$ if the flag is specified.

1.3

This section contains the plots outputted by *polygon_isVisible_test()*. Clearly, something is not right. My test cases for *edge_collision()* and *polygon_isSelfOccluded()* all pass and many scenarios were tested. I believe my error lies in my implementation and the way I check the points from each vertex.

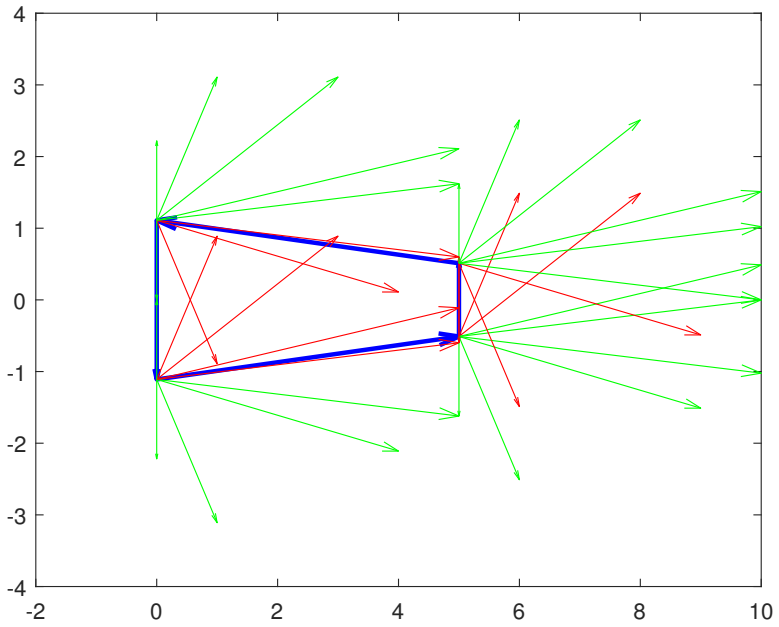

Figure 2: Results for vertices1

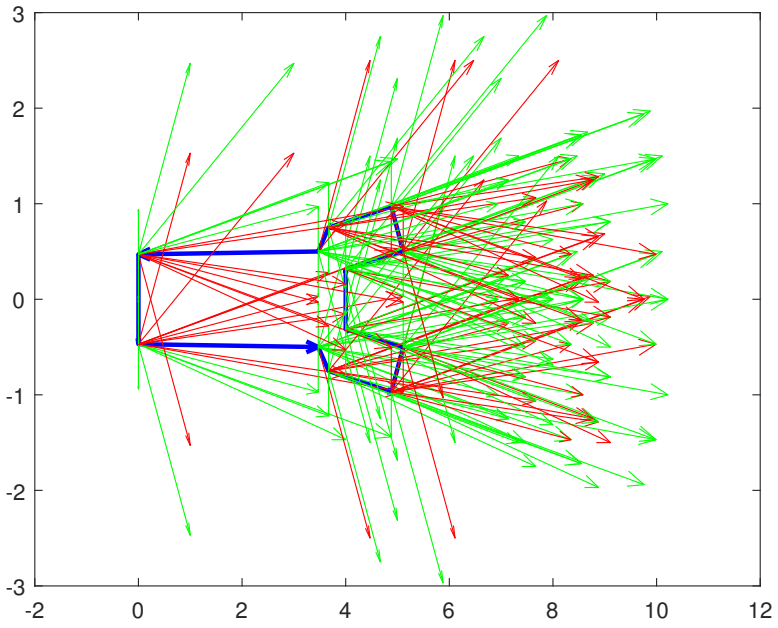

Figure 3: Results for vertices2

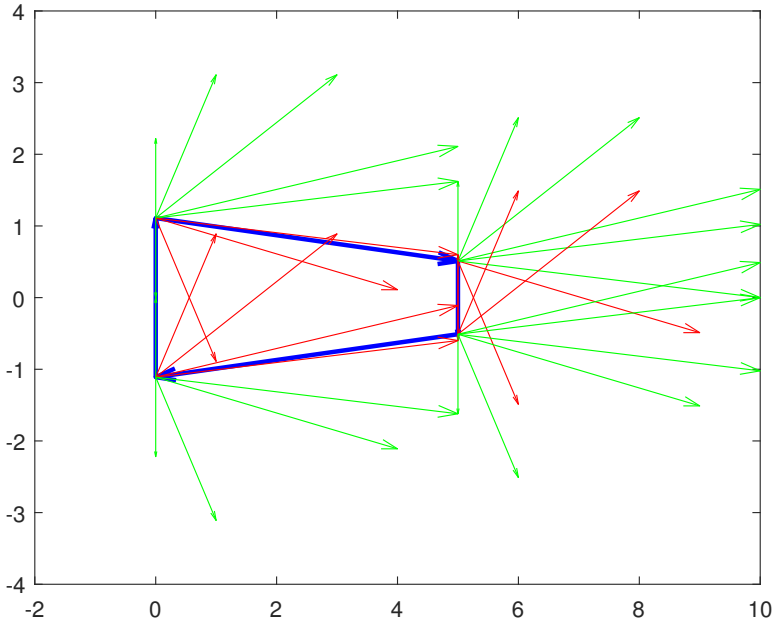

Figure 4: Results for vertices1 flipped

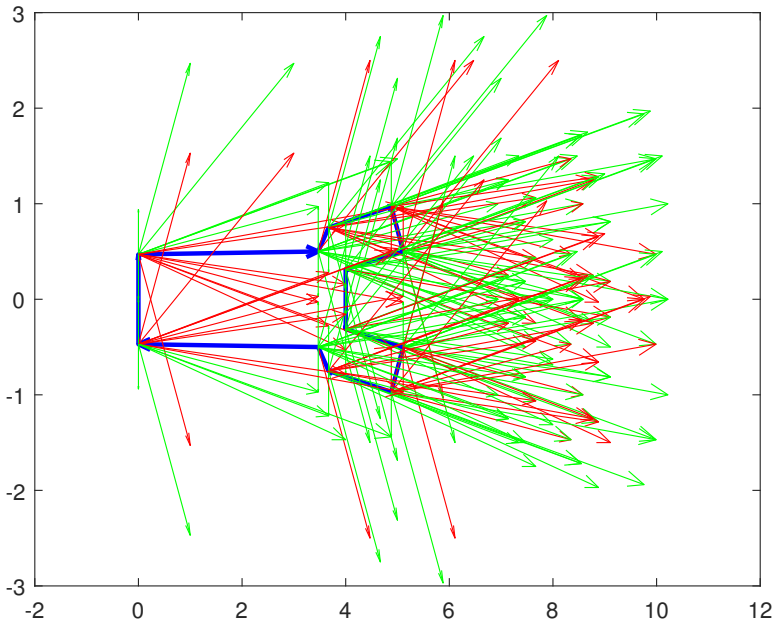

Figure 5: Results for vertices2 flipped

1.4

This section contains the plots outputted by *polygon_isCollision_test()*. Naturally, there are errors since this function is built around *polygon_isVisible*, a buggy function.

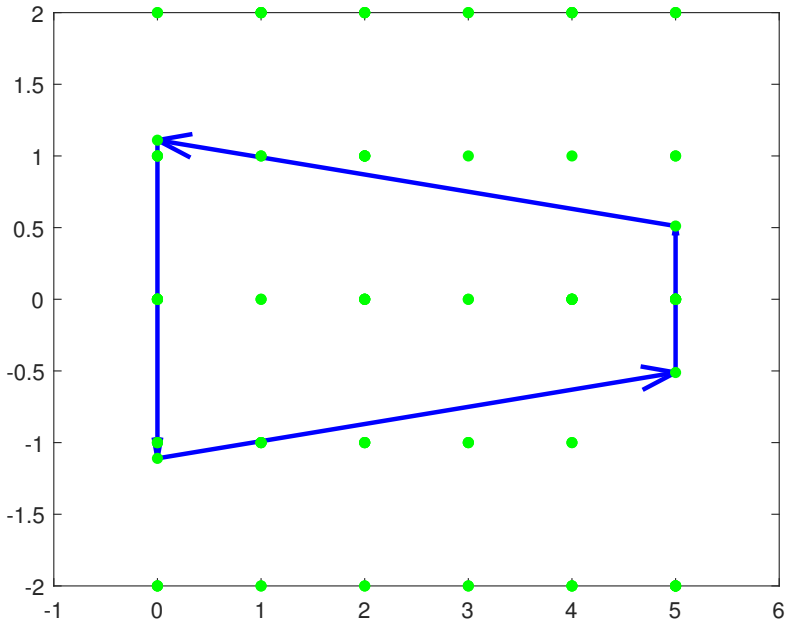

Figure 6: Results for vertices1

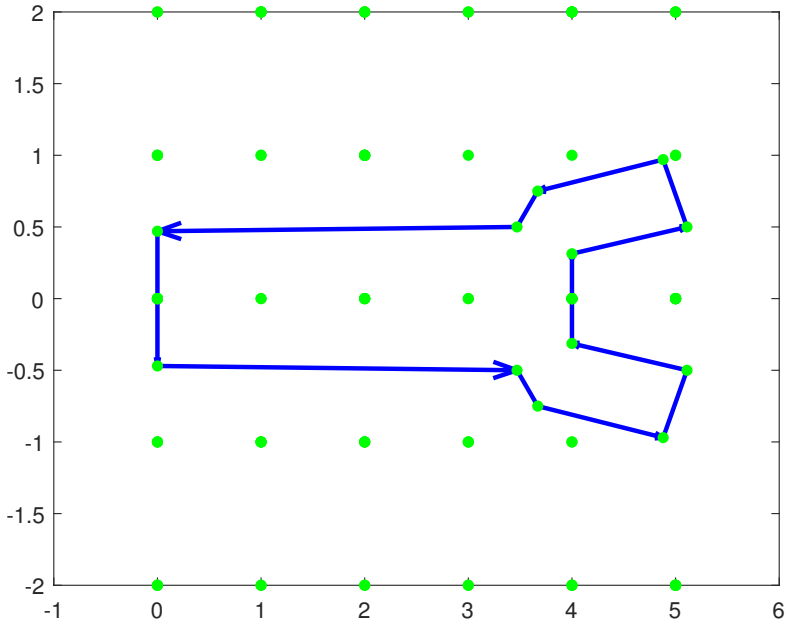

Figure 7: Results for vertices2

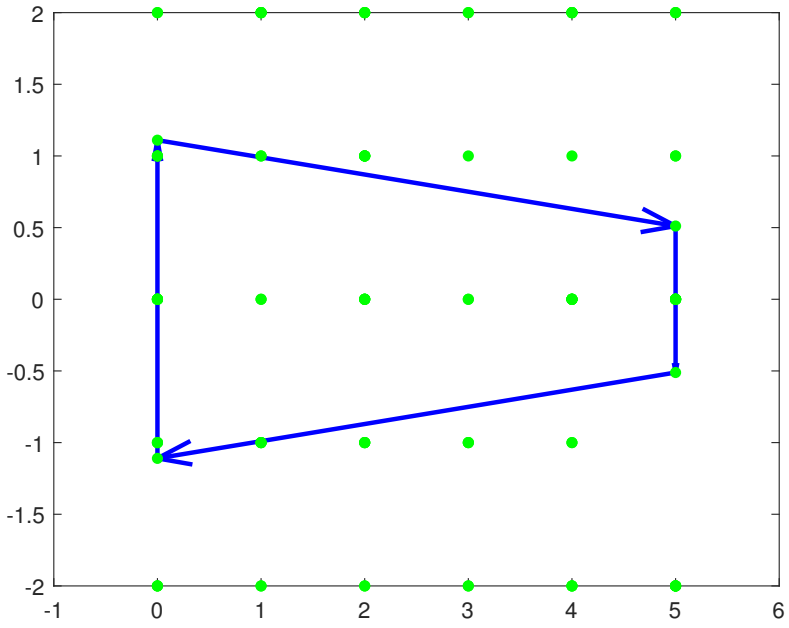

Figure 8: Results for vertices1 flipped

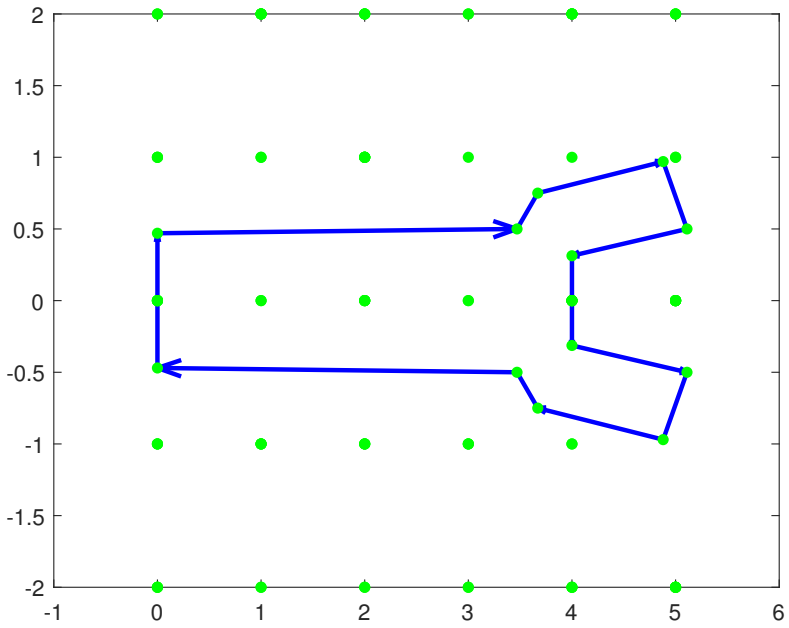

Figure 9: Results for vertices2 flipped

Problem 2

2.1

```
Following is the terminal output from the priority queue manipulation.
>> pQueue = priority_prepare()
pQueue =
  0  1  empty struct array with fields:
    key
    cost
>> pQueue = priority_insert(pQueue,'k1',10)
pQueue =
  struct with fields:
     key: 'k1'
    cost: 10
>> pQueue = priority_insert(pQueue,'k2',2)
pQueue =
  1  2  struct array with fields:
    key
    cost
>> pQueue = priority_insert(pQueue,'k3',30)
pQueue =
  1  3  struct array with fields:
    key
    cost
>> pQueue = priority_minExtract(pQueue)
pQueue =
  1  2  struct array with fields:
    key
    cost
>> pQueue = priority_insert(pQueue,'k4',25);
>> priority_isMember(pQueue,'34')
ans =
  logical
   0
>> priority_isMember(pQueue,'k4')
ans =
  logical
   1
>> pQueue = priority_minExtract(pQueue)
pQueue =
  1  2  struct array with fields:
    key
    cost
>> pQueue = priority_minExtract(pQueue)
pQueue =
  struct with fields:
     key: 'k3'
    cost: 30
```

```
>> pQueue = priority_minExtract(pQueue)
pQueue =
  1  0  empty struct array with fields:
    key
    cost
>>
```

2.2

In this grid, the coordinates of each element correspond to the *key* in the pQueue. Naturally, the cost of the element in the grid corresponds to the *cost* in the pQueue.

pQueue is an array of structs, so the grip must be looped through and stored into the pQueue. Then, the minExtract function could be called until the queue is empty. Every time the function is called, the output is stored into a new queue. The new queue will be in ascending order, since it is filled by sequentially calling min extract which outputs the next lowest cost. If you want the elements back in the grid but in ascending order, you can simply loop through the length of the queue and store each element in the corresponding grid location.

3

3.1

The function *matlabStyleHinter()* was run in the directory. It checked the optional files and reuturned errors saying that output arguments were not used. Those return values are not shown below.

```
>> matlabStyleHinter()
File: edge_angle.m
* Programming style report
    No problems found
* Matlab Code Analyzer report
    No problems found
File: edge_angle_test.m
* Programming style report
    No problems found
* Matlab Code Analyzer report
    No problems found
File: edge_isCollision.m
* Programming style report
    No problems found
* Matlab Code Analyzer report
    No problems found
File: edge_isCollision_test.m
* Programming style report
```

```
    No problems found
* Matlab Code Analyzer report
    No problems found
File: polygon_isCollision.m
* Programming style report
    No problems found
* Matlab Code Analyzer report
    No problems found
File: polygon_isCollision_test.m
* Programming style report
    No problems found
* Matlab Code Analyzer report
    No problems found
File: polygon_isSelfOccluded.m
* Programming style report
    No problems found
* Matlab Code Analyzer report
    No problems found
File: polygon_isSelfOccluded_test.m
* Programming style report
    No problems found
* Matlab Code Analyzer report
    No problems found
File: polygon_isVisible.m
* Programming style report
    No problems found
* Matlab Code Analyzer report
    No problems found
File: polygon_isVisible_test.m
* Programming style report
    No problems found
* Matlab Code Analyzer report
    No problems found
File: polygon_plot.m
* Programming style report
    No problems found
* Matlab Code Analyzer report
    No problems found
File: polygon_plot_test.m
* Programming style report
    No problems found
* Matlab Code Analyzer report
    No problems found
File: priority_insert.m
* Programming style report
    No problems found
* Matlab Code Analyzer report
    No problems found
File: priority_isMember.m
* Programming style report
    No problems found
```

```
 * Matlab Code Analyzer report
    No problems found
File: priority_minExtract.m
 * Programming style report
    No problems found
 * Matlab Code Analyzer report
    No problems found
File: priority_prepare.m
 * Programming style report
    No problems found
 * Matlab Code Analyzer report
    No problems found
File: priority_test.m
 * Programming style report
    No problems found
 * Matlab Code Analyzer report
    No problems found
File: twolink_polygons.m
 * Programming style report
    No problems found
 * Matlab Code Analyzer report
    No problems found
File: twolink_polygons_test.m
 * Programming style report
    No problems found
 * Matlab Code Analyzer report
    No problems found
>>
```

4

4.1

This homework took me roughly 10 hours. About the length of an instrumentation lab. I think this is a fair homework length assuming there is 2 weeks to complete it. With three other courses having a heavy workload I would be extremely pushed to get this done in a week. I spend roughly 95% of my time on problem one. It wasn't exceedingly difficult, but it was very time intensive to get it right. Even now, I know that it is not the best work possible, but I made a select few decision early on that would make going back and fixing it take just as long as doing the problem set in the first place.