Lab 1 :

Summer Orienteering

By Prakhar Gupta (pg9349)

The problem at its core is a graph problem. Hence we have used A* algorithm.

Following are the steps/decision taken to build the solution

1. **Reading image**
   The input image is read using Pillow library of python and we save the pixels into a list of list in python. This reverses the coordinates which are changed accordingly in later part of the lab. A list of pixel is created using the input image which uses objects of pixel class

2. **Reading elevation and checkpoint files**
   Checkpoint and elevation files are read and stored as list. Elevation file has extra columns which are sliced off

3. **Creating adjacency list for graph**
   The adjacency list is created after increments and decrements in x and y of the pixel location we have also checked if the pixel value is equal to (205, 0, 101) which is out of bound and in that case we will not added it in the graph.

4. **Create graph and cost function**
   Graph is created using adjacency list. Terrain graph and pixelvertex classes are used to make the graph.
   The cost of travelling from one note to next(pixel a to pixel b) is calculated using below formula

   $$x1, y1 = coordinates\ of\ pixel_a \qquad x2, y2 = coordinates\ of\ pixel_b$$

   Distance is calculated using Euclidian formula scaled along with the constants of those pixels in meters

   $$distance = \sqrt{((x1 - x2) * 7.55)^2 + ((y1 - y2) * 10.29)^2 + (z1 - z2)^2}$$

   Using the distance cost between neighbours is calculated represented by cost function. Value of speed of adjacent pixel is taken as its assumed we are on the edge of pixel a. This gives us g(x)

$$cost_{neighbours} = distance/speed_{pixelb}$$

5. Heuristic Calculations

$$cost_{heuristic} = distance/speed_{road}$$

Heuristic is calculated using the same Euclidian formula as above by using a node along with the checkpoint/end node.
However we change the dividing factor to speed of road which we have taken as 10 (highest in our terrain world). Hence its not possible to travel from any given node to the checkpoint node is less than the cost of the heuristic. This gives us h(x).

Both our **cost function and heuristic** work due to the following reasons.

- Heuristic cost is monotonic in nature i.e. it decreases as we move closer to the destination/checkpoint node. Hence our A* algorithm converges faster.
- The distance between two node is divide by the speed which makes sure we are capturing time. It may be the case that the distance may be small but the speed to travel would be much less. The ration of both(time) capture this relationship.

6. A* Algorithm

A* algorithm is an informed search algorithm which is coded using priority queue, sets and dictionary in python.

$$g(x) = cost_{neighbours} \qquad h(x) = cost_{heuristic}$$

$$f(x) = g(x) + h(x)$$

The algorithm continues to go over nodes and add them into priority queue bases on the f(x) value which is the cost of travel with heuristic value added. The nodes which have lowest f(x) are popped from the

priority queue. Since priority queue has a time complexity of log(n). We are able to determine the path in less time.

The algorithm will terminates in two cases-

- When priority queue is empty
- When it has found the node its was looking for

Hence our algorithm will work to find the most optimal path across a terrain.

Since it A* is not a brute force algorithm it converges faster as we only pick nodes which lead to lowering the cost to our destination node.