Identification of letter A using graph theory

Akhila Jetty (IRM2014006)
Arun Kumar Reddy(IRM2014005)
Swarnima(IWM2014003)

Introduction

- Connectivity: The number of neighbours of a point which are equal to one in the binary image gives its connectivity. Thus, Maximum connectivity of a point can be 8.
- Skeleton: A Skeleton of the image gives a one pixel length figure which approximates the shape of the original figure, This is an important pre-processing step necessary to extract the branch points and endpoints of the image.
- Endpoints : These are the positions in the binary image where the structure ends ie. where the structure made by one's ends as binary image is made by zeros and ones.
- Branch points : These are the positions where the direction changes in the skeleton.
- Edge: An edge is said to be present between two points if there is a skeleton without any major curves connecting the two points.

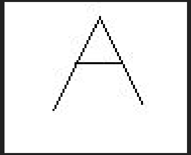
Implementation

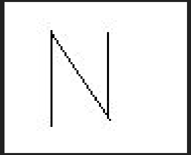
1. Two assumptions on the input image :
   a. The image should be clean.
   b. The letter in the image should be thin.
2. We came up with two different methods for the recognition of letter ”A”.
   a. In the first approach, we attempt to find the properties of the letter ’A’ in the figure.
   b. In the second approach, we try to the model the figure as a graph and validate its properties.

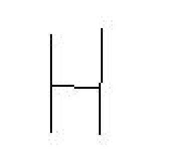
Approach 1

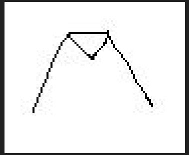
- Here we divide the approach in 3 parts
  - Preprocessing
  - Segmentation
  - Checking
- Preprocessing : We check for branch points and endpoints of the image and get the skeleton of image .
- Segmentation : Segment the image in three parts as the middle part of “A” contains the branching.
- Checking : Now we are checking the properties of given image with the properties of “A”.
  - The main properties we considered here are
    - The starting vertex (i,j) should be connected to point (i+1,j+1) and (i+1,j-1).
    - The middle segmented part should have one row in which number of ones are greater ie. branch point (k,l) so at least (j-l) ones should be present in row k.
    - At least 3 endpoints and 3 branch points should be present in the image.

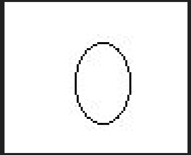
Results :

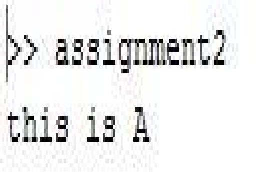




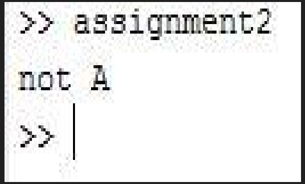




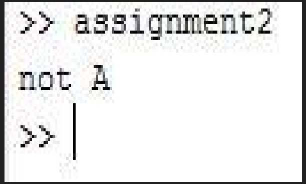




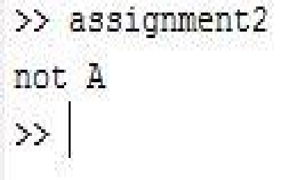
```
>> assignment2
this is A
```



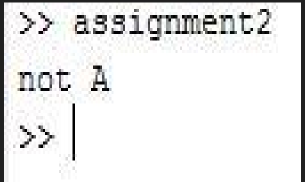
```
>> assignment2
not A
>> 
```



```
>> assignment2
not A
>> 
```



```
>> assignment2
not A
>> 
```



```
>> assignment2
not A
>> 
```

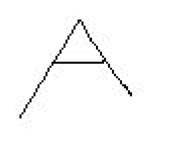
Approach 2

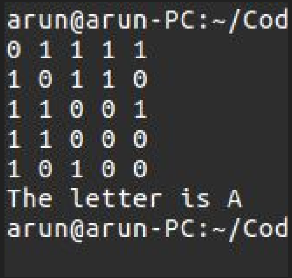
- The motivation behind this approach to apply the concepts of graph theory to character recognition.
- The first step of this process is to clean the image to remove any outlier points that may affect the outcome.
- Then, A few morphological operations are applied on the image to obtain the skeleton of the character in the image. The skeleton is usually one pixel thick.
- The end points and the branch points in the skeleton are obtained by checking the neighbourhood of each pixel in the skeleton. The pixels with only one neighbour are marked end points. The pixels with three or more neighbours are marked as branch points.

Approach 2

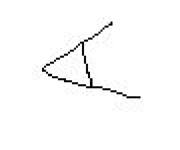
- The image is once again cleaned to remove duplicates and to merge connected branch points into one.
- Now, the remaining endpoints and branch points are assumed to be vertices of a graph. An edge is added between the two points if they are connected by the skeleton. This connectivity is checked by applying a constrained breadth first search.
- The letter ‘A’ can be viewed as a graph. This graph has five vertices of which two are pendent. The non-pendent ends of the pendent edges are connected. Also, these ends are connected to a common vertex which is top-most point in the image. The pendent vertices are the left-most and right-most points in the image.
- These properties are checked for in the graph which was obtained and if they are satisfied, the image is of the letter ‘A’.

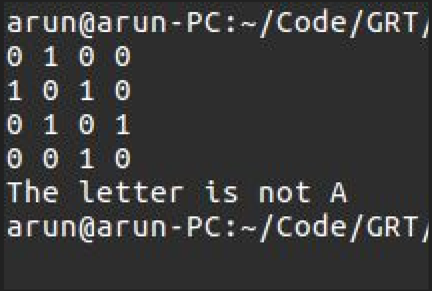
Results



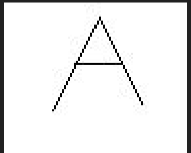


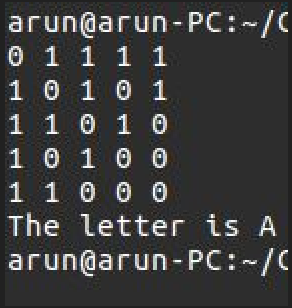
```
arun@arun-PC:~/Cod
0 1 1 1 1
1 0 1 1 0
1 1 0 0 1
1 1 0 0 0
1 0 1 0 0
The letter is A
arun@arun-PC:~/Cod
```



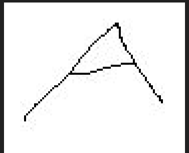


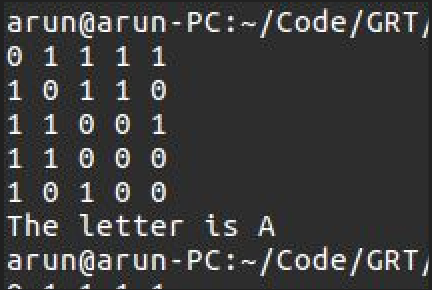
```
arun@arun-PC:~/Code/GRT/
0 1 0 0
1 0 1 0
0 1 0 1
0 0 1 0
The letter is not A
arun@arun-PC:~/Code/GRT/
```





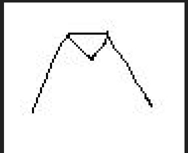
```
arun@arun-PC:~/C
0 1 1 1 1
1 0 1 0 1
1 1 0 1 0
1 0 1 0 0
1 1 0 0 0
The letter is A
arun@arun-PC:~/C
```

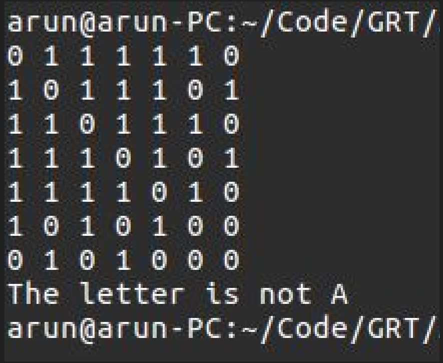




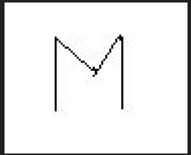
```
arun@arun-PC:~/Code/GRT/
0 1 1 1 1
1 0 1 1 0
1 1 0 0 1
1 1 0 0 0
1 0 1 0 0
The letter is A
arun@arun-PC:~/Code/GRT/
```

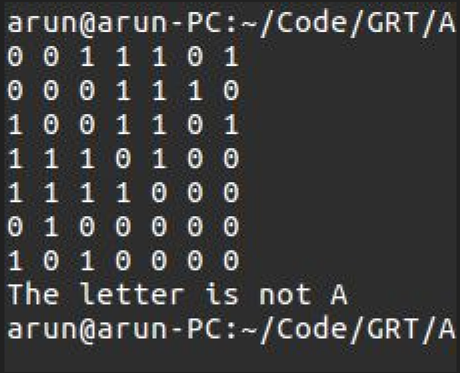
Results



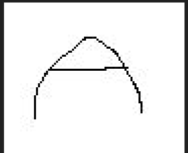


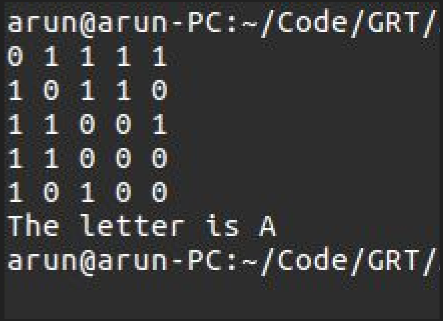
```
arun@arun-PC:~/Code/GRT/
0 1 1 1 1 1 0
1 0 1 1 1 0 1
1 1 0 1 1 1 0
1 1 1 0 1 0 1
1 1 1 1 0 1 0
1 0 1 0 1 0 0
0 1 0 1 0 0 0
The letter is not A
arun@arun-PC:~/Code/GRT/
```



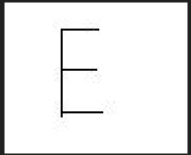


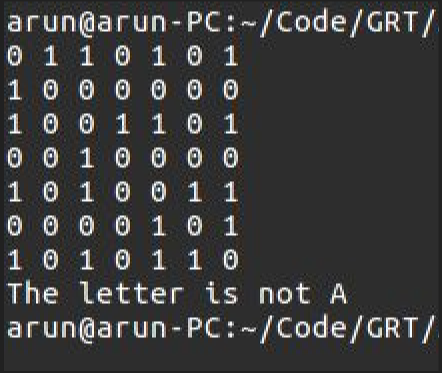
```
arun@arun-PC:~/Code/GRT/A
0 0 1 1 1 0 1
0 0 0 1 1 1 0
1 0 0 1 1 0 1
1 1 1 0 1 0 0
1 1 1 1 0 0 0
0 1 0 0 0 0 0
1 0 1 0 0 0 0
The letter is not A
arun@arun-PC:~/Code/GRT/A
```





```
arun@arun-PC:~/Code/GRT/
0 1 1 1 1
1 0 1 1 0
1 1 0 0 1
1 1 0 0 0
1 0 1 0 0
The letter is A
arun@arun-PC:~/Code/GRT/
```





```
arun@arun-PC:~/Code/GRT/
0 1 1 0 1 0 1
1 0 0 0 0 0 0
1 0 0 1 1 0 1
0 0 1 0 0 0 0
1 0 1 0 0 1 1
0 0 0 0 1 0 1
1 0 1 0 1 1 0
The letter is not A
arun@arun-PC:~/Code/GRT/
```