Assignment 1

Sanjana S Prabhu

September 2, 2019

1 Introduction

This is an assignment to implement a traffic intersection problem from the textbook Aho,Hopcroft and Ullmann. We have 5 roads and a junction common to them. The problem is to design a traffic system between them.

2 Mathematical model of the problem

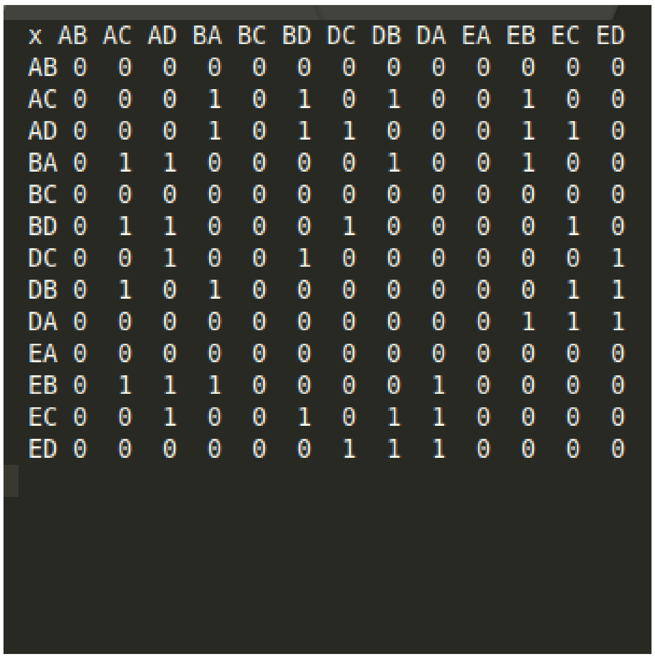
We can model this problem with a mathematical structure known as a graph. A graph consists of a set of points called vertices, and lines connecting the points, called edges. For the traffic intersection problem we can draw a graph whose vertices represent turns and whose edges connect pairs of vertices whose turns cannot be performed simultaneously. The figure is as shown below.

Figure 1: Graph showing the vertices and edges

This graph can now be converted into a table of incompatible turns where the rows and columns represent the vertices and if there is an edge between any two vertices, the entry in the table is 1. The table is shown below.

3 Algorithm to solve the problem

We can use the greedy algorithm to solve this problem. The approach is to scan all the vertices and assign them a same colour if they are compatible. By the end of the iterations we must end up with the fewest number of groups of vertices which represent the turns that can be made simultaneously.



```
x  AB AC AD BA BC BD DC DB DA EA EB EC ED
AB 0  0  0  0  0  0  0  0  0  0  0  0  0
AC 0  0  0  1  0  1  0  1  0  0  1  0  0
AD 0  0  0  1  0  1  1  0  0  0  1  1  0
BA 0  1  1  0  0  0  0  1  0  0  1  0  0
BC 0  0  0  0  0  0  0  0  0  0  0  0  0
BD 0  1  1  0  0  0  1  0  0  0  0  1  0
DC 0  0  1  0  0  1  0  0  0  0  0  0  1
DB 0  1  0  1  0  0  0  0  0  0  0  1  1
DA 0  0  0  0  0  0  0  0  0  0  1  1  1
EA 0  0  0  0  0  0  0  0  0  0  0  0  0
EB 0  1  1  1  0  0  0  0  1  0  0  0  0
EC 0  0  1  0  0  1  0  1  1  0  0  0  0
ED 0  0  0  0  0  0  1  1  1  0  0  0  0
```

Figure 2: Table of incompatible turns

4 The Pseudocode

The pseudocode is as follows:

```
procedure greedy ( var G: GRAPH; var newcolour: LIST );
{ greedy assigns to newcolour those vertices that may be
given the same colour }
var marker: boolean;
i,j: integer;
begin
newcolour := ;
i := first uncolored vertex in G;
while v<nodes do begin
found = false;
j = first vertex in newcolour;
while j<nodes do begin
if there is an edge between v and w in G then found = true;
j = next vertex in newcolour
end;
if found = false do begin
mark j colored;
add j to newclr
end;
```

```
 j = next uncolored vertex in G
 end
 end; { greedy }
```

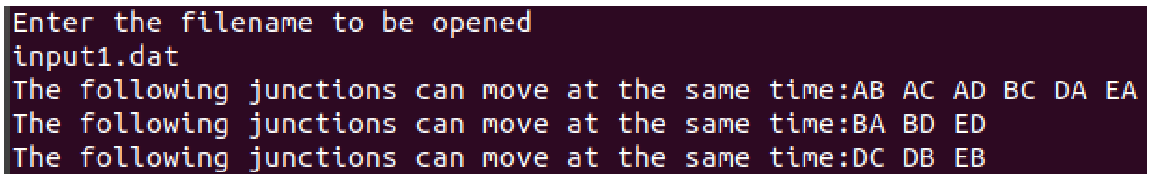
5 The C Code

The greedy algorithm is implemented in C as follows:

```
 int greedy(int *array,int nodes,int *colour){
int i;
int j;
int m;
int newcolour=1;//indicates the number of colours
int marker=2;//true
for(i=0;i<nodes;i++){
if(colour[i]==0){//assign a new colour to the first uncoloured vertex
colour[i]=newcolour;
newcolour++;
}
for(j=0;j<nodes;j++){
if(colour[j]==0){//check for next uncoloured vertex
for(m=0;m<nodes;m++){
if(colour[m]==newcolour-1){
if(*(array+j*nodes+m)==1){
/*check if previously coloured vertices of the colour in question
have an incompatibility with the next uncoloured vertex*/
marker=1;//incompatibility exists
}}}
if(marker==2){
colour[j]=newcolour-1;//if not assign the colour to this vertex}
else
marker=2;//else reinitialise the marker}}}
return newcolour-1;//return the total number of colours
}
```

6 Output

The output of the code when it is inputted a .dat file of the format corresponding to the example input file and which consists of the number of vertices, the names of the vertices and the table of incompatible turns, gives an output as follows:



```
Enter the filename to be opened
input1.dat
The following junctions can move at the same time:AB AC AD BC DA EA
The following junctions can move at the same time:BA BD ED
The following junctions can move at the same time:DC DB EB
```

Figure 3: Output

7 Conclusions