UNIVERSITY OF LEEDS

This is a repository copy of *Queue Management Project: User and Programme Documentation.*.

White Rose Research Online URL for this paper:
http://eprints.whiterose.ac.uk/2223/

Monograph:
Clark, S.D. (1992) Queue Management Project: User and Programme Documentation. Working Paper. Institute of Transport Studies, University of Leeds , Leeds, UK.

Working Paper 343

Reuse
Unless indicated otherwise, fulltext items are protected by copyright with all rights reserved. The copyright exception in section 29 of the Copyright, Designs and Patents Act 1988 allows the making of a single copy solely for the purpose of non-commercial research or private study within the limits of fair dealing. The publisher or other rights-holder may allow further reproduction and re-use of this version - refer to the White Rose Research Online record for this item. Where records identify the publisher as the copyright holder, users can verify any specific terms of use on the publisher's website.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

White Rose university consortium
Universities of Leeds, Sheffield & York

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk/

White Rose Research Online
http://eprints.whiterose.ac.uk/

Institute of Transport Studies
University of Leeds

This is an ITS Working Paper produced and published by the University of Leeds. ITS Working Papers are intended to provide information and encourage discussion on a topic in advance of formal publication. They represent only the views of the authors, and do not necessarily reflect the views or approval of the sponsors.

White Rose Repository URL for this paper:
http://eprints.whiterose.ac.uk/2223/

Published paper

Clark, S.D. (1992) *Queue Management Project: User and Programme Documentation.* Institute of Transport Studies, University of Leeds. Working Paper 343

White Rose Consortium ePrints Repository
eprints@whiterose.ac.uk

Working Paper 343

January 1992

Queue Management Project

User and Programme Documentation

S. D. Clark

ITS Working Papers are intended to provide information and encourage discussion on a topic in advance of formal publication. They represent only the views of the authors, and do not necessarily reflect the views or approval of the sponsors

This work was sponsored by the Science and Engineering Research Council

Contents

PART A

	Page
1 Introduction	3
2 Model Specification	3
3 Data Requirements	4
3.1 Time Information	5
3.2 Lane information	5
3.3 Physical information	5
3.4 Traffic information	6
3.5 Turning information	7
3.6 Signal information	8
3.7 Summary	9
3.8 Flow file	9
4 Model Display	9
4.1 Network window	11
4.2 Time/Space window	11
4.3 Window movement	11
4.4 Retaining windows	11
5 Model Interior	12
5.1 Greenshield's model	12
5.2 All flows are uncongested	13
5.3 Left turning traffic	13
5.4 Right turning traffic	13
5.5 Lane changing	13
5.6 Blocking criterion	14
5.7 Signal movements	14
5.8 Traffic may leave a junction by an input link	14
6 Model Outputs	14
7 Restrictions	15
8 Enhancements	16

PART B

	Page
1 Introduction	18
2 Program Parts	18
3 Data Structures	19
3.1 Junctions	19
3.2 Flow	20
3.3 Map	21
3.4 Signal	21
3.5 Limit	21
3.6 Entity arrays	21
3.7 Entities	22
3.8 Junction lists	22
3.9 Data array	22
3.10 Data record	22
3.11 Junction exit clear array	23
3.12 Junction blocked start/stop times	23
3.13 Direction lookup table	23
4 Algorithms	24
4.1 Input	24
4.2 Check	25
4.3 Graph	25
4.4 Lights	26
4.5 Run	26
4.6 Stat	29
4.7 Wave	30
5 Problems	30
5.1 Input	30
5.2 Check	31
5.3 Graph	31
5.4 Run	31
5.5 Stat	31
Appendix 1	
Example model specification file	32
Appendix 2	
Results	36

1 Introduction

This document is intended for use by individuals who wish to use the Traffic model developed on the Sun386i workstation. The model is an extension of that developed by D.J.G. Ford on the IBM PC.

The first section deals with an outline discussion of the traffic flow theory used in the model. The second section demonstrates how to define a model for input into the program. The third section clarifies the arrangement of the graphics output on the screen. The next section explains some of the internal assumptions used in the model. The following section explains the form and content of the logging information collected by the model. The remaining two sections constitute a general discussion on the model, namely its restrictions and possible enhancements.

Other documentation which may be useful include:

- o Outline Specification for a Queue Management Computer Model, S.D. Clark, 07/11/90;
- o A Graphical Model to Illustrate Queue Management Techniques at Signalised Junctions within Saturated Traffic Networks, D.J.G. Ford;

2 Model Specification

The program models traffic flow through a one-way signalised network of multi-laned links. The arrangement of the network may be one-dimensional (Linear) or two-dimensional (Grid). Each junction in the network may have up to four approaches/exits, labelled North, South, East and West.

Traffic within the network behaves according to hydrodynamic traffic flow theory, using a linear (Greenshield) speed-density relationship. This theory provides us with :

The density of traffic at input approaches;
The speed of blocks of traffic;
The speed of a shock wave between traffic;
The speed of queue stopping waves;
The speed of queue starting waves;
The density of traffic leaving a queue;

Traffic in the model is represented by blocks of traffic of a certain length of a given density. These blocks of traffic may merge with others or form queues of

stationary traffic. As blocks of free moving traffic join the rear of a queue the length of the queue is increased, the amount depends upon the density of the block. The position of this queue rear, over time, is marked on a graph. When signals and receiving links allow traffic in a queue to move then blocks of free moving traffic are generated at the head of the queue. The position of the queue front is also marked on the same graph as the queue rear.

Traffic movement within the network is affected by the presence of queues and other traffic. Most importantly if a queue spills back to an upstream exit then the movement of traffic into that exit is also affected or even stopped.

3 Data Requirements

The data specifying the model is contained in two text files. These files may be created or amended using any standard text editor. The first file contains static information relating to the network. The second contains information on flows at the input junctions. The data sets required for an execution of the model are typed after the name of the program e.g.

```
traff welling welling.q
```

runs the traffic model, taking the data from the files `welling` and `welling.q`. If the program is run without specifying these data files a usage message is displayed.

The data required for the model falls into five categories:

Time information for the whole network;
Lane information for the whole network;
Physical information for each junction approach/exit;
Traffic information for each junction approach/exit;
Turning information for each junction approach/exit;
Signal information for each junction;
Flow information for each input junction.

An example network and data files are given in appendix 1.
As can be seen from the example the data is comma delimited where a list of information, on the same line, is required. Also note that spaces or tabs may be added to align values. The text following the : at the end of the line is a comment and is ignored by the model. The input lines, including comment, must be less than 99 characters.

Each of the items listed below are on a separate line. After the information title the number in brackets specifies how may items of information are required on this line. Generally each line consists of the information for each of the four directions (N S E W) for this junction.

3.1 Time information

There are three pieces of time information, each of which should be in the following order and on a separate line.

(i) Model run time (1). The total length of time that the model is to be run for.

(ii) Run in period (1). This is the amount of time judged necessary for the network to fill with traffic and reach a 'stable' state. After this period Time/Space graphs are drawn and statistics collected.

(iii) Stepping time (1). At this time the program pauses after each time unit and waits for the user to press return. In order for the return to be registered the SunView cursor (arrow) must be in the Shell/Command tool window from which the program is being run.

(iv) Cycle time (1). This is the signal cycle time for the whole network.

All times must be integers and be less than 32767 (32767 seconds ~ 9 hours).

3.2 Lane information

Only one item of lane information is required:

(i) Lane change (1). This is the percentage of traffic, by length, which moves into a short lane if there is no turning traffic (see 5.5 (ii)).

3.3 Physical information

These pieces of information allow the physical characteristics of the road network to be established. This information is required for every junction in the network.

(i) Junction number (1). This single number is the reference tag for this junction. The number must be unique and be between 1 and 12.

(ii) Junction name (1). The name of the junction is used to annotate the axis in the Time/Space graphs. The name must consist of 20 (twenty) or fewer characters.

(iii) Lanes per link (4). The number of lanes for each link is specified here. If the link is an output or is sealed (no traffic) then this value is ignored. There can at most be 4 (four) lanes per link.

(iv) Link lengths (4). A measure of the length of each lane in the link (stop line to stop line) is required. The number of lengths needed depends upon the number of lanes specified for this direction (on the line above). The lengths are taken in a left to right order, as seen by traffic on the link. Where there are unequal lane lengths, i.e. shorter lanes, then these must be either the first and/or last length specified. Once again, if the link is an output link or is sealed then no lengths are required. The units are metres.

(v) Connecting junction (4). The junction number which connects with this one in each of the four directions is given here. If the junction is an input to or an output from the network then this value MUST be 0 (zero). If the junction is sealed then this value MUST be -1 (minus one).

(vi) Stop line (4). If this link is an approach to the junction then this value MUST be 1 (one). If, on the other hand, it is an exit from the junction then this value must be 0 (zero). If the link is sealed then this value is ignored.

(vii) Junction width (2). In order to establish when a queue blocks the upstream junction a measure of the widths across the junction are required. The first number is the width of the junction from North to South whilst the second is the width from East to West. The units are metres.

3.4 Traffic information

This category of information provides measures of various traffic statistics.

(i) Jam density (4). The jam density, k_j, is required for all types of link except output links and sealed links. The units are in vehicles per kilometre (veh/km).

(ii) Free flow speed (4). The free flow speed, u_f, is required for all types of link except output links and sealed links. The units are in kilometres per hour (km/h).

3.5 Turning information

Information on how traffic splits when flowing through the network is required next. This is done by specifying a percentage, i.e. a number between 0 and 100, to reflect the amount of splitting.

(i) Percentage traffic turning left (4). This value is the percentage of traffic in the left lane which wishes to turn left.

(ii) Percentage traffic turning right (4). This value is the percentage of traffic in the right lane which wishes to turn right.

For each link that receives traffic, i.e. an input approach or an internal exit, the amount of traffic which enters each lane requires specifying. Obviously if there is only one lane in this link then this percentage is 100. If, however, there is more than one lane in the receiving link then a percentage entering each lane is required. The percentages are taken in a left to right order, as seen by traffic entering the link.

(iii) North lane splits (4). This is the split of traffic entering the North link from each of the other three directions. If this is an input link then the other three percentages are zero and the percentages in position one are the lane splits for the North link.

(iv) South lane splits (4). This is the split of traffic entering the South link from each of the other three directions. If this is an input link then the other three percentages are zero and the percentages in position two are the lane splits for the South link.

(v) East lane splits (4). This is the split of traffic entering the East link from each of the other three directions. If this is an input link then the other three percentages are zero and the percentages in position three are the lane splits for the East link.

(vi) West lane splits (4). This is the split of traffic entering the West link from each of the other three directions. If this is an input link then the other three percentages are zero and the percentages in position four are the lane splits for the West link.

3.6 Signal information

This information describes the timing and contents of the signal phases for a junction.

(i) Offset (1). This is the number of time units that a junction's start of cycle stage is offset from a chosen junction. The chosen junction in the network MUST be given an offset of 0 (zero). Negative offset are acceptable.

(ii) Number of stages (1). The number of signal stages per cycle is given here. The maximum number of stages per junction is 8 (eight).

(iii) Stage description (variable). For each signal stage a character string is required to describe the signal settings. Each action starts with a direction (N S E W). There then follows characters indicating the signal aspects. These characters and their meaning are given below:

G - Green light
A - Amber light
H - Halt (red) light
L - Left turning arrow
R - Right turning arrow
D - Directly across arrow

Any direction not mentioned in the string is assumed to be Halt (red). Thus the string

NGSL

sets the North links signal to Green;
South links signal to Left turning arrow;
East links signal to Halt (red);
West links signal to Halt (red);

Valid examples of other stage strings are : NASAH, ERLSHR;
Invalid examples are : NSAH, NRNHWR;

The stage string

XXX

is shorthand for all signals Halt (red). A stage description must contain 14 (fourteen) or fewer characters.

(iv) Stage duration (variable). For each of the stages described above a duration is required. The total of these times MUST be the same as the cycle time for the network.

3.7 Summary

The order and format for all the data is summarised in table 1 overleaf.

3.8 Flow file

The flow file contains information about what input flows are to be used during the execution of the model.

(i) Time. This is the time after which the following flows are to be used. The first of these MUST be time zero.

(ii) Flows. These values are the input flows to use for each junction. Every junction in the model must have a line of information even if it does not have any inputs. In this case the values supplied are ignored.

Information	Category	Amount	Units
Time	Time	number	(s)
Flow	Traffic	N, S, E, W	(veh/h)

Table 2 Summary of flow data format

4 Model Display

The program creates a number of display windows to show the workings of the model. The largest window displays a representation of the traffic network. To the right of this window there may be other windows containing Time/Space axis for North/South routes in the network. Below the traffic network window there may also be Time/Space graphs for East/West routes in the Network.

Information	Category	Type	Units
Model-time	Time	number	(s)
Run-in-time	Time	number	(s)
Single-step-time	Time	number	(s)
Cycle-time	Time	number	(s)
Lane change	Lane	percentage	1..100
Junction number	Physical	number	1..12
Junction name	Physical	string	max 20
Lanes per link	Physical	N, S, E, W	1..4
Link length 1	Physical	N, S, E, W	m
:	:	:	:
Link length n	Physical	N, S, E, W	m
Connecting Jn	Physical	N, S, E, W	-1,0,1..12
Stop line	Physical	N, S, E, W	0 or 1
Jn distances	Physical	NS, EW	m
Jam density	Traffic	N, S, E, W	veh/km
Free-flow speed	Traffic	N, S, E, W	km/h
Percentage left	Turning	N, S, E, W	0..100
Percentage right	Turning	N, S, E, W	0..100
N Split lane (left)	Turning	N, S, E, W	0..100
:	:	:	:
N Split lane (right)	Turning	N, S, E, W	0..100
S Split lane (left)	Turning	N, S, E, W	0..100
:	:	:	:
S Split lane (right)	Turning	N, S, E, W	0..100
E Split lane (left)	Turning	N, S, E, W	0..100
:	:	:	:
E Split lane (right)	Turning	N, S, E, W	0..100
W Split lane (left)	Turning	N, S, E, W	0..100
:	:	:	:
W Split lane (right)	Turning	N, S, E, W	0..100
Offset	Signal	number	(s)
Number of stages	Signal	number	0..8
Stage descriptions	Signal	string,	max 14
Stage duration	Signal	number,	(s)

Table 1 Summary of model data format

4.1 Network window

The network of links is represented as a rectangular grid of equal length links. The actual length of the link is given, either above or to the right of the link. Traffic signals are drawn as they would be seen by the approaching traffic.

During the execution of the model blocks of freely moving traffic are represented as hollow blocks, scaled in width and length accordingly. Queues of stationary traffic are represented by scaled, solid blocks of differing colours. The colour of the queue depends upon its lane. This enables a correspondence to be made between a queue in the network and its Stopping/Starting wave profile. Below the network view is an area for text information. This area displays the current model time, a request to press return or a warning. The warning message concerns any traffic which moves into an approach (see 5.8).

4.2 Time/Space window

A time space window is drawn for each North/South or East/West route which consists of two or more junctions. The axes are drawn to scale. The X axis is a measure of time and is [model-run-time minus run-in-time] in length. The Y axis is the length of the longest link in a lane. Consecutive junctions in a route are appended to each other.

4.3 Window movement

The windows created by the program may be manipulated in a limited manner. Any operation which causes a redraw of the window contents (close/overlap/resize) results in the window being wiped clean and only new images displayed. Movement is generally permissible. Quitting a window abandons the execution of the entire program. Occasionally this puts SunView into a sleep state for a couple of minutes.

4.4 Retaining windows

At the end of the model-run-time period the windows created by the program are removed. In order to save a windows contents the program must be prevented from stopping. This can be done by following the steps below:

1. Set the time of single-stepping to a number less than or equal to the model-run-time;
2. At time model-run-time do not press return;
3. Close the Network window and any Time/Space windows not required;
4. Move the Time/Space windows which you wish to retain out of harms way;
5. Run subsequent executions of the program from another Shell/Command Tool.

As long as you do not cause the termination of the original program the required window stays on the screen.

5 Model Interior

This section attempts to describe the techniques used within the model code. These techniques include assumptions made to facilitate a speedy and accurate implementation of the theory. These assumptions are listed below and individually justified later in this section.

1 - Greenshield's model is adopted to explain traffic behaviour;
2 - All flows are uncongested;
3 - All left turning traffic is in the left hand lane;
4 - All right turning traffic is in the right hand lane;
5 - Lane changing occurs only when leaving an approach and when filling a short lane;
6 - An exit is blocked if any traffic in the exit backs upstream into a junction;
7 - Shared lanes require signals set to green for all types of traffic in the lane for it to be able to move;
8 - Traffic may leave a junction via an input link.

5.1 Greenshield's model

As mentioned in section 2, Greenshield's model is used to provide traffic behaviour characteristics. The simpler uses are:

- Speed of a block of density k.

$$u = u_f\left(1 - \frac{k}{k_j}\right)$$

- Speed of a shock wave between two blocks of density k_a and k_b.

$$u_{shock} = u_f\left(1 - \left(\frac{k_a}{k_j} + \frac{k_b}{k_j}\right)\right)$$

- Speed of a stopping wave given a block of density k.

$$u_{stop} = u_f\left(\frac{k}{k_j}\right)$$

- Speed of a starting wave.

$$u_{start} = \frac{u_f}{2}$$

Where k_j is the links jam density
and u_f is the links free flow speed

Greenshield's model is also used to convert from flows to densities. When traffic leaves an approach to join an exit the traffic is divided according to its flow. Thus as a block crosses the stop line its density is converted to a flow (or a proportion of it, depending upon how much of the block crosses the stop line). This flow is then divided according to the split percentages given by the user. These proportioned flows are then converted back to densities for the allocation of traffic blocks on the receiving links. Once again, the ratio of flow over density provides the block length.

The flow to density equation is the solution of :

$$0 = u_f k - \frac{u_f}{k_j} k^2 - q$$

5.2 All flows are uncongested

From the form of the flow to density equation above, the solution is clearly the roots of a quadratic. This provides us with two measures of density. Since we are operating in uncongested conditions with regard to flows then the lower root is chosen. This lower root corresponds to using the left hand side of the flow-density curve. Problems may exist when, given the values of u_f, k_j and q, there are no real roots for this quadratic. If this event should occur then an error diagnostic message is displayed and half the link's jam density is returned.

5.3 Left turning traffic

This condition requires that all the traffic that wishes to turn left should be placed in the left hand lane.

5.4 Right turning traffic

This condition requires that all the traffic that wishes to turn right should be placed in the right hand lane.

5.5 Lane changing

Lane changing occurs in two situations:

(i) When leaving a junction. As a block of traffic crosses an approach stop line, it is converted into a flow and proportions assigned to appropriate destination links. For each destination link the flow is proportioned amongst the lanes within the link. Thus traffic leaving a lane within an approach to a junction is spread over all receiving lanes. These flows are accumulated for all blocks of traffic

leaving the approach and converted back into blocks, of a given density and length, in the receiving lanes.

(ii) Filling short lanes. Only short lanes which causes the link to 'fan out' are implemented, i.e. short lanes extending back from a stop line. Traffic is moved into these lanes according to the following criterion. If there is turning traffic of the appropriate type then the amount of turning traffic is taken, by length, from the neighbouring lane. Once again if there should prove to be insufficient traffic in the neighbouring lane then all the traffic is moved into the short lane and a warning message issued. If there is no turning traffic of the appropriate type then the lane change percentage as specified by the user, is taken from the neighbouring lane. In this latter case the traffic in the short lane is straight ahead traffic.

5.6 Blocking criterion

Blocking back is caused when any lane within the link fills the upstream junction. If a lane is shared or the movement is directly across the junction then any blocking back of traffic in any exit link causes movement out of the approach to stop. If the lane is not shared and consists of only left or right turning traffic then movement will only be prevented if the receiving link is blocked..

5.7 Signal movements

The signals provide for movement out of the lane when the signals provide permission for ALL traffic in the lane to move over the stop line. Thus, for example, if the lane consists of both left and direct traffic then the signals need to be set for left and direct traffic movement.

5.8 Traffic may leave a junction via an input

Normally in a one way traffic system vehicles are not permitted to cross into an approach link. In order to increase the number of situations which may be modelled by the program such movement is allowed. The consequence of this, however, is that such traffic is no longer modelled. With a linear network, so long as the approach is an input link, i.e. traffic beyond it is not normally modelled once the traffic has left the junction, then this is not a problem.

6 Model Outputs

The model outputs are in two forms, graphical and textual. The graphical form has been covered in section 4, so this section deals with the textual output. When the model has finished, the windows holding the traffic network and the Time/Space graphs are closed. The information collected during the execution of

the model is processed. The model then writes this information to a text file. The text file has the same base name as the static model specification file but with the extension `.out` added. Thus if the model specification file was called `welling` then the output file is called `welling.out`. The contents of this text file are also listed by the program using the Unix `more` utility. This utility pauses every screen full of information and moves on a line if return is pressed or one page if the space bar is pressed. The Q key quits the listing. For help press the H key.

For an example of the output see appendix 2. The information provided is described below:

(i) Histograms of Queue lengths

For each lane in the link a histogram showing the distribution of the queue lengths is provided. The class widths are a multiple of 5 (five). The frequencies represent the number of lengths equal to or less than the given frequency.

(ii) Table of statistics

For each link a table of summary statistics, one entry for each lane, is provided. These statistics are:

The mean queue length;
The standard deviation of the queue lengths;
The variance of the queue lengths;
The maximum queue rear;
The maximum queue rear at the start of green.

(iii) Blocking times

This table reports the start and finish times for any blocking back occurring in junction exits. Only blocking which starts after the run-in-time is reported. If the blocking has a finish time of -1 (minus one) then at the end of the model execution the blocking back was still occurring.

7 Restrictions

Most of the restrictions in the model have been covered elsewhere. Here I shall detail a few of the more important ones and introduce a few more. It should be noted that these restrictions were adopted for two reasons:

They reduce the complexity of the programming task;
They reduce the complexity of the model specification.

(i) Pseudo one-way systems only. This is perhaps the severest restriction and the most difficult to remove. The restriction means that most grid type networks can not be accurately modelled.

(ii) Maximum of four approaches/exits in a junction. If this restriction was relaxed a number of problems would have to be addressed. The representation of the network in the window would be cumbersome, requiring diagonal as well as horizontal and vertical links at least. If this was implemented then the network may as well be displayed geometrically exact, rather than rectangular. Movements directly across/left/right would have to be redefined so as to enable more than one lane to receive the traffic.

(iii) Left/Right turning traffic originating from only one lane. At the moment left/right turning traffic originates from one lane. It may be useful to have turning traffic originating from two or more lanes.

(iv) Lane turning occurs in the junction. It would be more realistic if the changing of lanes occurred in the link itself.

(v) No pedestrian crossings. Partway along the length of a link a pedestrian crossing with stochastic demand may be desirable.

(vi) No platoon dispersion. At the moment all traffic in a block travels at the same speed. This is acceptable in short links. In longer links, however, the block may lengthen to reflect differing traffic speeds within the block.

(vii) Uniform junction cycle times. Rather than assuming the same cycle time for all junctions in the network, cycle times for each junction could be used.

(viii) Uniform starting wave speeds. In real life we might expect the position of the rear of the queue in the receiving link to affect the starting wave speed of traffic leaving a queue.

8 Enhancements

As seems reasonable, the main enhancement would be to relax some of the restrictions mentioned in section 7. I shall grade each restriction in terms of my estimation of the effort required to overcome it (10 - a great deal, 1 - little).

(i) Full two way network (10);
(ii) An 'unlimited' number of approaches/exits (10);

(iii) Multiple turning lanes (4);
(iv) Lane changing (4);
(v) Pedestrian crossings (4);
(vi) Platoon dispersion (2);
(vii) Independent cycle times (1).
(viii) Non uniform starting wave speeds (1)

One additional enhancement would be to record the total number of vehicles hours spent in the network. Since we have both a density and a length for the block then the product of these two should give us the number of vehicles. If the blocks of traffic are stamped with their time of input into the model (creation), on leaving the network the difference between the current and creation time is the time the vehicles in the block have spent in the network. The only problem with this is when two blocks, of different creation times, merge and when blocks form queues and visa-versa. Some aggregate creation time would need to be constructed in these cases. This measure would enable some over-all performance measure to be usefully constructed. This could take about 3 units of effort.

1 Introduction

This document provides additional information for those who want or need a greater technical understanding of the Traffic model.

Undoubtedly the best way to try and understand program code is to read it. Most of the important ideas in the code are commented within the code.

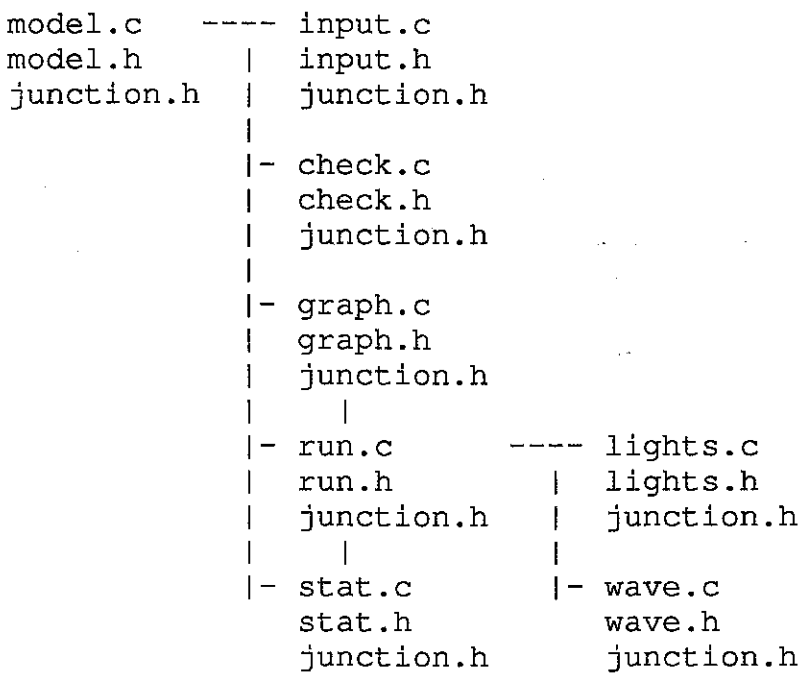
The first section contains a discussion of the program 'building blocks' (source, header and make files). The second section describes the data structures used in the model. The third section describes some of the reasoning behind key algorithms in the model. The remaining section deals with potential problem areas.

2 Program Parts

The program is written in the C programming language. The graphics routines are supplied by the SunGKS libraries. Thus in theory the program is eminently portable to any platform which has a K&R compatible C compiler and at least a level 1a implementation of GKS. The only Sun specific portion of the code is that which establishes the SunWindows for the network and Time/Space windows (functions `ws1_draw_view`, `draw_time_space_ns` and `draw_time_space_ew` in graph.c)

The execution of the program progresses via the following phases:
Input - Read in the model specification from file;
Check - Validate the specification and report all errors;
Graph - Establish SunWindows and draw graphic images;
Run - Run the model from start to finish;
Stat - Produce output;

The interdependence between the file elements of the program is explained below,



3 Data Structures

The program uses some data structures to enable the encapsulation of data to take place. Other structures are used to permit information to be transmitted more easily between functions. Structures are arranged in two forms, either as an array or a linked list.

3.1 Junctions (junction)

This data structure is held in an array which consists of MAX_JN elements (13 initially). For programming convenience element zero is not used, so MAX_JN-1 elements are available. Each element holds the junction data for the correspondingly numbered junction, as given by the user in the data file.

no.	The junction number, same as the array element number;
name.	The junction name, NAME_LEN-1 characters at most;
lanes[].	The number of lanes per link, NO_JN elements;
length[][].	The lengths of each lane. The first subscript is the link, the second is the lane. NO_JN links and MAX_LANE lanes at most;
connecting[].	The connecting junction for each link, NO_JN elements;
stop[].	1 for an approach link, 0 otherwise, NO_JN elements;
NSwidth.	The North to South junction width;

EWwidth.	The East to West junction width;
jam_density[].	The jam density for each link, NO_JN elements;
free_flow[].	The free flow speed for each link, NO_JN elements;
turn_l[].	The percentage of traffic turning left out of each link, NO_JN elements;
turn_r[].	The percentage of traffic turning right out of each link, NO_JN elements;
split[][][].	The percentage split between lanes in each link. The first subscript is the destination link, the second is the source link whilst the third is the lane. NO_JN links and MAX_LANE lanes at most;
offset.	The signal phase offset;
stages.	The number of signal stages, MAX_STAGE at most;
stage[][].	The stage description strings. The first subscript is the stage number, MAX_STAGE at most. The second subscript is the characters, MAX_STAGE_LEN at most;
time[].	Each stages duration. MAX_STAGE elements at most;
longest[].	The length index of the first longest lane for each link. NO_JN elements;
lastlong[].	The length index of the last longest lane for each link. NO_JN elements.

3.2 Flow (flows)

This structure is held as an array of MAX_JN elements. This array holds the flows of traffic, per hour, at each of the input links along with the time after which they are to be used.

time.	The time after which this flow is to be used;
p1[][].	The flow at the link. MAX_JN entries and NO_JN approaches.

In addition, element zero of this structure (which is never used to hold flow information since we do not have a zero junction) is used to hold count information. The variable flow[0].time holds the number of junctions in the network so that we can, for each time, read in the correct number of junction lines. The variable flow[0].p1[0][0] holds the number of different flows to be used in the mode.

3.3 Map

This is a two dimensional array of integers. Its purpose is to hold a spatial 'map' of the network. This map gives information on where a junction is located in relation to all others. This information is primarily used when displaying information in the Windows. The first and second dimensions are 2*MAX_JN making $(2*MAX\_JN)^2$ elements in total. The first subscript is the Y (North/South) direction, whilst the second is the X (East/West) direction.

3.4 Signal (signal)

This structure is held as an array of MAX_JN elements. The first element is ignored (see junction). The structure records the current signal stage number for the whole junction and its start time.

stage.	The stage index in junction of the current stage;
time.	The time at which the current stage started.

3.5 Limit (limit)

This structure holds the limits of the map in one direction. This gives us the extent of the map given in 3.2. There are two instances of this structure, one for the Y direction and one for the X direction.

max.	The maximum index of the map;
min.	The minimum index of the map;
dist.	The maximum real distance in the map.

3.6 Entity arrays

These arrays hold pointers to the list of entities within the model. Within each list the entities for a given junction and link are held. The elements of the list must be in lane front position order.

block_list[][].	The list of block entities. The first subscript is the junction number whilst the second is the link. The first dimension is MAX_JN, the second is NO_JN. Element zero of block_list is not used;
queue_list[][].	The list of queue entities. The first subscript is the junction number whilst the second is the link. The first dimension is MAX_JN, the second is NO_JN. Element zero of queue_list is not used;

3.7 Entities (entity)

The entities are the basic elements in the model. They represent both blocks of free moving traffic and queues. They are held in linked lists, with the first element tied to an array of pointers (see 3.6).

moved.	TRUE if the entity has been moved during the current cycle, FALSE otherwise. This prevents traffic which has moved from one link into another being moved in the receiving lane during the current time period. Has no meaning for a stationary queue;
lane.	The lane of this entity. MAX_LANE at most;
k.	The density of this entity in vehicles per metre;
pos1.	The front position for this entity. Given as a distance in metres from the stop line;
pos2.	The rear position for this entity. Given as a distance in metres from the stop line. Must, by definition, be no less than pos1;
pos3.	The previous value of pos1;
pos4.	The previous value of pos2;
next.	A pointer to the next entity in the list, NULL if end of list.

3.8 Junction lists (jn_list)

This structure is held as a single linked list. The list contains all those junctions which are free to receive traffic.

jn.	The junction number;
d.	The junction link;
next.	A pointer to the next junction in the list, NULL if end of list;

3.9 Data array

These arrays hold pointers to the list of data records kept by the model. One copy of the data record is held for every lane within the approach link.

data_log[][].	The first subscript is the junction number whilst the second is the link. The first dimension is MAX_JN, the second is NO_JN. Element zero of data_log is not used;

3.10 Data record (data_log)

This structure holds the logging information collected by the model during its last cycle. They are held in linked lists, with the first element tied to an array of pointers (see 3.9).

lane. The lane of this data record. MAX_LANE at most;
width. The width of a class on the X axis;
xlen. The number of X classes necessary;
prev_stop. TRUE if the during the previous time interval the signal was set to stop, FALSE otherwise. Used to say when a switch from stop to go occurred so that the queue rear at start of green can be recorded;
gmax. The maximum queue rear at start of green for the last cycle;
max. The maximum queue rear during the last cycle;
data[]. The frequency count of queue lengths. Each element corresponds to one X class. There are at most MAX_COL classes;
next. A pointer to the next data record in the list, NULL if end of list;

3.11 Junction exit clear array

These arrays hold pointers to the list of blocked exits start and stop times. A list of stop/start times is kept for each junction and exit link.

jn_exit[][]. The first subscript is the junction number whilst the second is the link. The first dimension is MAX_JN, the second is NO_JN. Element zero of jn_exit is not used;

3.12 Junction blocked start/stop times (blocked)

These structures hold the start and stop times of any exit blockages that may take place. The stop time is marked as -1 when the structure is first created with a start time. They are held in linked lists, with the first element tied to an array of pointers (see 3.11).

start. The time at which blocking started;
stop. The time at which blocking finished;
next. A pointer to the next element in the list, NULL if end of list;

3.13 Directions lookup table (z[][])

This structure permits the block movement routine to be more succinctly coded by enabling the same code to be used irrespective of the source links direction. In essence z is a lookup table which given a current direction and the desired turning movement returns the destination direction. The first subscript is the current direction, the second is the turning movement. For each direction the following turning movements are available:

P Previous direction (opposite of current)
L Left junctions direction
R Right junctions direction
D Directly across junctions direction

The full table is thus

	movement			
current	P	L	R	D
N	S	E	W	S
S	N	W	E	N
E	W	S	N	W
W	E	N	S	E

4 Algorithms

Its is difficult to talk about program code in the abstract thus this section contains an overview only. More detailed descriptions may be found in comments within the code. Each program element is taken in turn.

4.1 Input

There are a number of routines which read in a set of information from the static model specification and flow files. All information is read a line at a time and then split into individual items using one of the routines below. The routine to use depends upon where we are in the specification file.

Single floating point number (`split_no`)

This function is used to split a line consisting of a single number, e.g. model time information and signal offsets. The function returns a floating point number which, if appropriate, is caste down to an integer;

Single character string (`split_name`)

This function takes a string of up to NAME_LEN-1 characters or until the first colon from the input string. This is used to get the junction name;

A set of four integers (`split_int_nsew`)

This function returns four integers from in a comma delimited form in a string. The assumption that there are four values is made;

A set of four floating point numbers (`split_flt_nsew`)

This function returns four floating point numbers. These numbers are comma delimited and in a string. The assumption that there are four values is made;

A variable, but known, number of floating point numbers (`split_time`)

This function takes an input string, a position to start in that string and returns the floating point number following that position. This number can be caste down to an integer if necessary.

A variable, but known, number of strings (`split_str`)

This function takes an input string, a position to start in that string and returns the string (terminating in a comma or colon) following that position.

4.2 Check

The purpose of check is to ensure that the data file supplied by the user describes a physically correct network. This includes checking that values are consistent across links and that each junction has at least one approach and one exit. The junction map of the network is also constructed as a check for overlapping junctions and for spatial use later.

4.3 Graph

The functions in graph establish the network and time space windows on the Sun monitor.

Initially the function `get_map_limits` establishes the extents of the map in the X and Y directions. Functions `x_size` and `y_size` find the minimum and maximum map indices along either a single X row or Y column. The minimum and maximum values for all these rows or columns then give global minimums and maximums.

Next the network window is drawn. Function `ws1_draw_view` establishes the physical window on the monitor. Function `ws1_set_trans` sets the transform from World to Device coordinates. The extent of this transform is determined by the X and Y limits. In any case the X and Y values are in the range 0 to 24. The rectangular grid network is then drawn by the function `draw_network`. Each junction in the network is drawn using `draw_jn`. The length of the longest lane in a North or East link is draw either above or to the right of the link by function `draw_scale`.

The final stage is to draw the Time/Space windows. The functions to draw both North/South type and East/West type windows are similar. The function `draw_time_space_##` establishes the physical window. Function `set_time_space_##` sets the transform from World to Device coordinates. The extent of the World transformation is given by model run time minus run in time (for the time axis) and by the maximum junction lengths along a route (space axis). The function `draw_time_space_axis_##` then draws both the axis and labels them with the junction name.

The function `time_space_##` is used during model execution to draw the stopping and starting wave profiles during the final signal cycle.

NB `##` denotes either `ns` or `ew`.

4.4 Lights

These functions control the setting and displaying of the traffic signals.

The function `init_stage` sets the signal stage for the start of the model. This uses the offsets from a previous junction and the signal durations. The time at which this stage would have commenced, in the past, is also established. This is achieved by counting down the stage durations until the offset has been reached. This time is always a number less than or equal to zero.

The `update_phase` function checks to see if the current stage has come to an end. If it has then it returns TRUE and also the new stage string. If this is time zero then TRUE and the stage string are always returned. If an update of the stage is required then the new stage string is decoded and the lights for that junction drawn. During signal drawing the NE and SE coordinates of the signal box and the red, amber and green lights are reflected for use in drawing the SW and NW signals.

4.5 Run

The run module is the most complex and important element of the model. This module does some necessary initialisation and then advances the model from time zero to completion. The stages that it executes each time interval are:

Ensures the signals are set correctly;
Adds blocks to the input junctions;
Merges the blocks just input;
Establishes which exits are blocked;
Moves the blocks in the model;
Merges the internal blocks;
Update queue stopping waves;
Update queue starting waves;
If appropriate, draw stopping and starting waves;
If appropriate, collect statistics.

4.5.1 Initialisation

The signal stages are set to the appropriate stage and start time, given previous signal time offsets. The data records used to hold information collected after the run in time are created. The lists of block and queue entities are set to empty.

4.5.2 Ensure signal set correctly

This involves a call to a routine in lights. This routines goes through all the junctions in the network checking for a signal change. If a change has occurred then the signals for that junction are redrawn.

4.5.3 Adds blocks to the input junctions

At the start of each cycle, the flows at an input junction are used to generate input traffic. The flow per time unit is converted into a density per metre and this density forms a block at the rear of all traffic in the queue. The length of the block is given by the ratio of flow over density. It is perhaps worth mentioning here that whenever an entity (queue or block) is added to a list then the entities in a lane MUST be in order of their front position. It doesn't matter if different lanes are mixed together but as you travel down the list the front positions for each lane must increase. This is to ensure that when block movement occurs, all blocks are moved in order, starting with the one nearest the stop line.

4.5.4 Merges the blocks just input

So that full advantage of the recently input traffic blocks may take place, they are merged with any other blocks at the rear of the input link. It should be noted that this merge only deals with traffic at the input links.

4.5.6 Establishes which exits are blocked

A linked list is compiled of all junctions which are free to receive traffic. All links which are not internal exit links are clear. Thus approach links, input links, output links and sealed links are free to receive traffic at all times. An internal exit link is only clear if there is no traffic (queue or block) in the junction. This means that if any traffic has a rear position in the range [link length] to [link length minus junction width] then the exit is blocked for incoming traffic.

4.5.7 Moves the blocks in the model

This is the core of the model. It performs the functions of moving blocks in a link, filling short lanes and moving traffic from one link into another. The movement goes in two phases for each approach in the model. The first phase does the following:

- o Establish the distance this block can move given other traffic and Greenshield's speed-density relationship.
- o If movement does not cause traffic to cross the stop line then advance it and move some into a short lane if necessary.
- o If movement does cause traffic to cross the stop line then accumulate the flows into running totals.

The second phase takes the flows generated in the first phase and creates blocks in the appropriate receiving links.

4.5.8 Merges the internal blocks

This time the internal blocks only are merged. This is because some of the previous movement may have caused some blocks to abut each other.

4.5.9 Update queue stopping waves

For stage one, if there is a block with a zero front position and either the signals or a blockage prevents it from moving then a queue is formed. This queue is initially set to empty. The second stage takes any queues, including the empty one possibly created during the first phase, and advances their rear if there is a block of traffic at the rear. The speed of this stopping wave is given by Greenshield's equation.

4.5.10 Update queue starting wave

If there is a queue with a zero front position, the signals are set to green and the receiving link is clear, then the queue starts to discharge traffic. The amount of traffic to discharge is given by Greenshield's equation. Queues with a non zero front position should always generate traffic at their head.

4.5.11 If appropriate, draw stopping waves, starting waves and collect statistics

After the run in time of the model the state of the traffic network is reported to the user. The positions of the queue fronts and rears are graphed on the appropriate Time/Space diagrams, if they are not zero. Performance statistics are also collected.

4.6 Stat

The functions in stat initialise, collect and report the output statistics.

The function `init_stat` creates the data records to hold the information for each lane in an approach link. The maximum values and frequency counts are also initialised.

During the execution of the model the performance statistics of the model are collected. For a queue in a specific lane the corresponding data record is found. The cell count appropriate to the queues length is increased by one. The current queue rear is then compared with the queue rear maximum value. If the queue rear is greater than this maximum then the rear is recorded as the new queue maximum. Similarly if this is the start of green for the queue then the current queues rear is compared with the maximum queue rear at start of green to, perhaps, record a new maximum. Also the duration of any blocking back from an exit junction is recorded. If the exit is clear then a search is made for a previously unfinished blockage. If one is found then the current time is recorded as the stop time for the blockage. If the exit is not clear then a search is made to see if this blockage has already been recorded. If it is recorded then no action is taken, otherwise this is the start of a new blockage. A new blockage record is created with a start time of the current time. This new record is added to the end of the list of blockage durations for this junction exit.

When the model has run to completion, the information collected is written to file and then displayed using the Unix `more` command.

4.7 Wave

These functions provide the traffic characteristics, according to Greenshield's model. The functions to convert from flow to density and vice-versa along with speed functions are given here. Perhaps the most important aspect of these functions is the units for the various measures. Where the measure is directly supplied by the user, e.g. free flow speed or jam density, then the measures are in the units supplied. Where the measure is generated by the model, e.g. a blocks density or speed, then the distance measure is metres and the time measure is seconds. The table below gives a summary of the units used:

Function	Input Flow	Density	Speed	Jam density	Output
qtok	veh/h	.	km/hr	veh/km	veh/m
U	.	veh/m	km/hr	veh/km	m/s
U_stop	.	veh/m	km/hr	veh/km	m/s
U_start	.	veh/m	km/hr	veh/km	m/s
U_shock	.	veh/m	km/hr	veh/km	m/s

The only other point worth mentioning is that due to rounding errors the discriminant in the quadratic may go to a very small negative number rather than zero. To overcome this problem, if the value is small and negative then it is reset to zero.

5 Problems

During the execution of the model various problems may manifest themselves. For each stage given in section 2 potential problem areas are suggested. When a major fault occurs during the execution of the model the program halts with a diagnostic error message.

5.1 Input

The problems here will mainly concern an incorrectly formatted data file. Commas may be missing from lists of items. If this is the case then the remaining items in the list may contain uninitialised rogue values. These are difficult to detect and may only come to light because of strange behaviour in the run stage of the model. The user may also supply insufficient or too much information, in this case the program will abandon with an invalid end of file message. Ensure that there are the correct number of lane lengths and split percentages.

5.2 Check

There are not likely to be any major faults here. The program reports all inaccuracies in the data file before abandoning.

5.3 Graph

Given a correctly specified model (ensured to a large extent by Check) there are no instances of error conditions here.

5.4 Run

Since this phase contains most of the code for the model it is most prone to faults.

Dealing with signal settings first, the program only checks the validity of signal stage descriptions at run time. During the first signal cycle is when these errors are likely come to light. The program prints the signal stage description which is in error. It should be noted that any reference to an approach direction which is not an approach is treated as an error.

The next problem with Run is when the flow to density (`qtok`) equation gives non-real roots. The only solution to this is to lessen the flow by altering the turning and lane split percentages. Another problem occurs if a zero jam density is passed to any of the speed functions, since all these functions have the jam density as a divisor.

To accommodate a nearly unlimited number of entities within the network the memory to hold information about them is taken from the heap using malloc. If there should prove to be insufficient memory for an entity then the malloc will fail and the program abandons. The program must be re-compiled using a larger heap size with the makefile.

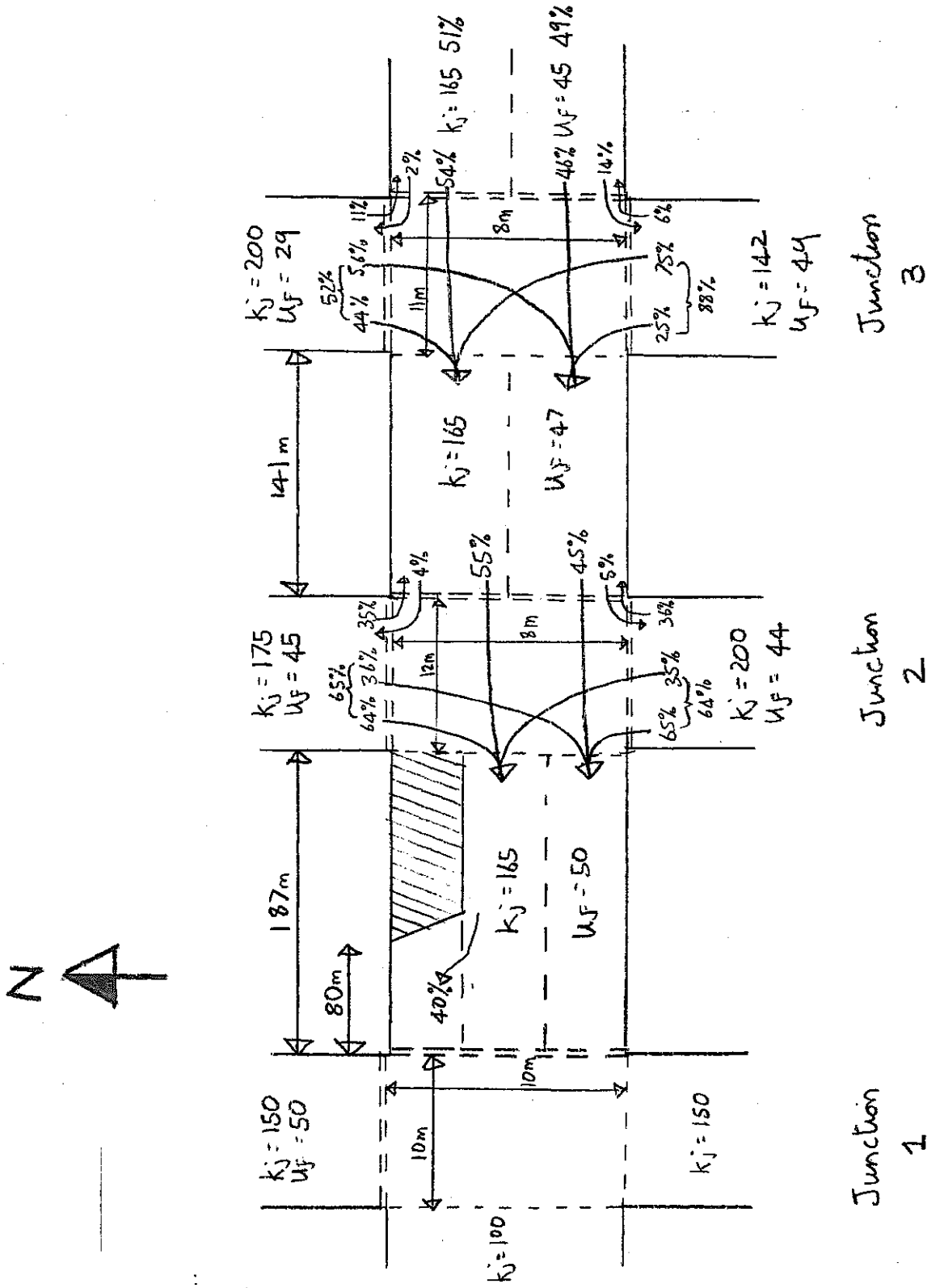
Inside the function to remove an entity (`free_entity`) checks are made for impossible (?) situations which mean that an entity can not be removed. If you get these messages then there is a serious problem with maintaining linked lists in the program. This problem did occur in the initial stages of development because memory freed was immediately being reused to create a new entity, resulting in important information disappearing.

5.5 Stat

The only problem likely to occur in stat is an inability to take memory from the heap to hold the logging information (see 6.4).

Appendix 1 - Example model specification file

Network



Static file

```
1200                                    : Model run time
360                                     : Run in time
1200                                    : Start stepping
90                                      : Cycle time
40                                      : LANECHANGE
1
Yorkshire Post
     1,   1,   3,   1,                 : NSEW direction ( lanes )
     200, 100, 187, 100,               : NSEW lane 1 length
     0,   0,   187, 0,                 : NSEW lane 2 length
     0,   0,   80,  0,                 : NSEW lane 3 length
     0,   0,   2,   0,                 : NSEW connecting jn
     1,   0,   1,   0,                 : NSEW stop line Y/N
     10,  10,                          : NS EW junction widths
     150, 150, 165, 100,               : NSEW jam density
     50,  0,   50,  0,                 : NSEW free flow speed
     0,   0,   0,   0,                 : NSEW % turning left
     0,   0,   0,   0,                 : NSEW % turning right
     100, 0,   0,   0,                 : NSEW lane split
22                                      : Offset
4                                       : Stages
EG,EANHA,NG,NAEHA,                      : Movements
40,17,27,6,                             : Time
2
MFI
     1,   1,   2,   2,                 : NSEW direction ( lanes )
     232, 128, 141, 187,               : NSEW lane 1 length
     0,   0,   141, 187,               : NSEW lane 2 length
     0,   0,   3,   1,                 : NSEW connecting jn
     1,   1,   1,   0,                 : NSEW stop line Y/N
     8,   12,                          : NS EW junction widths
     175, 200, 165, 165,               : NSEW jam density
     45,  44,  47,  50,                : NSEW free flow speed
     36,  65,  5,   0,                 : NSEW % turning left
     64,  35,  4,   0,                 : NSEW % turning right
     100, 0,   0,   0,                 : North   NSEW lane split
     0,   100, 0,   0,                 : South   NSEW lane split
     30,  64,  45,  0,                 : West    NSEW lane split
     70,  36,  55,  0,                 :         NSEW lane split
82                                      : Offset
4                                       : Stages
EG,EANAHSAH,NGSG,EAHNASA,               : Movements
65,4,14,7,                              : Time
```

```
3
Post Office
     1,   1,   2,   2,        : NSEW direction ( lanes )
     186, 180, 200, 141,      : NSEW lane 1 length
     0,   0,   200, 141,      : NSEW lane 2 length
     0,   0,   0,   2,        : NSEW connecting jn
     1,   1,   1,   0,        : NSEW stop line Y/N
     8,   11,                 : NS EW junction widths
     200, 142, 165, 165,      : NSEW jam density
     29,  49,  44,  47,       : NSEW free flow speed
     11,  88,  14,  0,        : NSEW % turning left
     52,  6,   2,   0,        : NSEW % turning right
     100, 0,   0,   0,        : North NSEW lane split
     0,   100, 0,   0,        : South NSEW lane split
     0,   0,   49,  0,        : East  NSEW lane split
     0,   0,   51,  0,        :       NSEW lane split
     56,  25,  46,  0,        : West  NSEW lane split
     44,  75,  54,  0,        :       NSEW lane split
0                             : Offset
4                             : Stages
EG,EANAHSAH,NGSG,EAHNASA,     : Movements
39,6,41,4,                    : Time
```

Flow file

```
0
500, 0,   0,   0,   : NSEW q jn 1
330, 360, 0,   0,   : NSEW q jn 2
300, 420, 840, 0,   : NSEW q jn 3
120
500, 0,   0,   0,   : NSEW q jn 1
150, 180, 0,   0,   : NSEW q jn 2
240, 270, 690, 0,   : NSEW q jn 3
240
500, 0,   0,   0,   : NSEW q jn 1
90,  360, 0,   0,   : NSEW q jn 2
270, 330, 450, 0,   : NSEW q jn 3
360
500, 0,   0,   0,   : NSEW q jn 1
60,  390, 0,   0,   : NSEW q jn 2
360, 330, 780, 0,   : NSEW q jn 3
480
500, 0,   0,   0,   : NSEW q jn 1
270, 90,  0,   0,   : NSEW q jn 2
240, 90,  0,   0,   : NSEW q jn 3
```

Appendix 2 - Results

Histogram for Yorkshire Post, N junction, lane 1.

```
Freq
  70 |
  65 |  *
  60 |  *  *    *
  55 |* *  *    *
  50 |***********
  45 |***********
  40 |***********
  35 |************
  30 |************
  25 |************
  20 |************
  15 |************
  10 |************
   5 |************
     |------------------------------------------ Queue length
                          11111111111111111112
       11223344556667888900112233445566678889O
      50505050505050505050505050505050505050
```

lane	Mean	Sd	Variance	Q Max	G Rear
1	28.58	16.849	283.878	66.7	59.3

Histogram for Yorkshire Post, E junction, lane 1.

```
Freq
 150 |*
 140 |*
 130 |*
 120 |*
 110 |*
 100 |*
  90 |*
  80 |*
  70 |*
  60 |*   **
  50 |*   **
  40 |*  ***
  30 |* ****  * *
  20 |***********
  10 |***********
     |------------------------------------------ Queue length
                          11111111111111111111
       11223344556667888900112233445566678888
      50505050505050505050505050505050505050
```

Histogram for Yorkshire Post, E junction, lane 2.

```
Freq
 180 |
 168 |*
 156 |*
 144 |*
 132 |*
 120 |*
 108 |*
  96 |*
  84 |*   *
  72 |*  **
  60 |* ***
  48 |* ***
  36 |* ***
  24 |*******
  12 |********
     |---------------------------------------- Queue length
                          1111111111111111111
       11223344556667888900112233445566678888
      505050505050505050505050505050505050
```

Histogram for Yorkshire Post, E junction, lane 3.

```
Freq
 180 |
 168 |*
 156 |*
 144 |*
 132 |*
 120 |* *
 108 |* *
  96 |***
  84 |***
  72 |***
  60 |****
  48 |****
  36 |****
  24 |*****
  12 |******
     |------------------ Queue length
       112233445566678
      5050505050505050
```

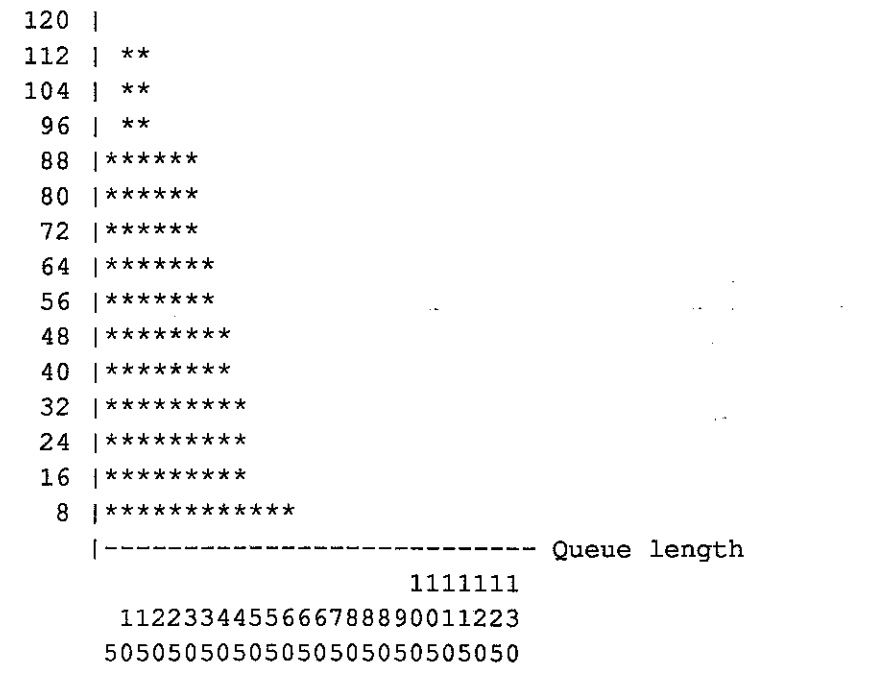
lane	Mean	Sd	Variance	Q Max	G Rear
1	21.42	16.670	277.895	68.0	61.9
2	14.61	11.571	133.877	54.1	52.5
3	9.29	6.819	46.499	34.4	34.0

Histogram for MFI, N junction, lane 1.

```
Freq
 240 |
 224 |*
 208 |*
 192 |*
 176 |**
 160 |**
 144 |**
 128 |**
 112 |***
  96 |****
  80 |****
  64 |*****
  48 |*****
  32 |******
  16 |*******
     |------------------------------------------------- Queue length
                          1111111111111111111122222222
       1122334455666788890011223344556667888900112233
      5050505050505050505050505050505050505050505050505
```

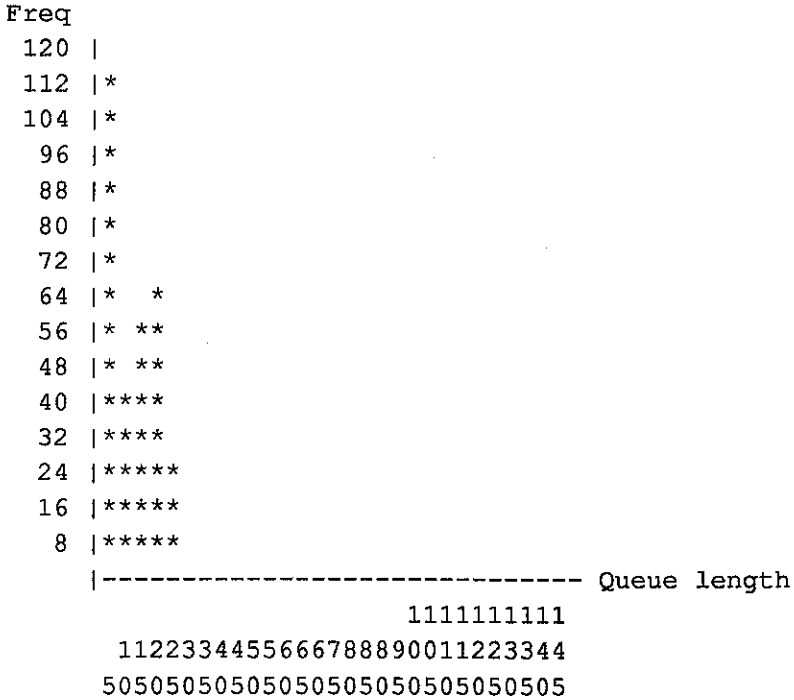
lane	Mean	Sd	Variance	Q Max	G Rear
1	10.67	8.146	66.356	34.7	33.0

Histogram for MFI, S junction, lane 1.

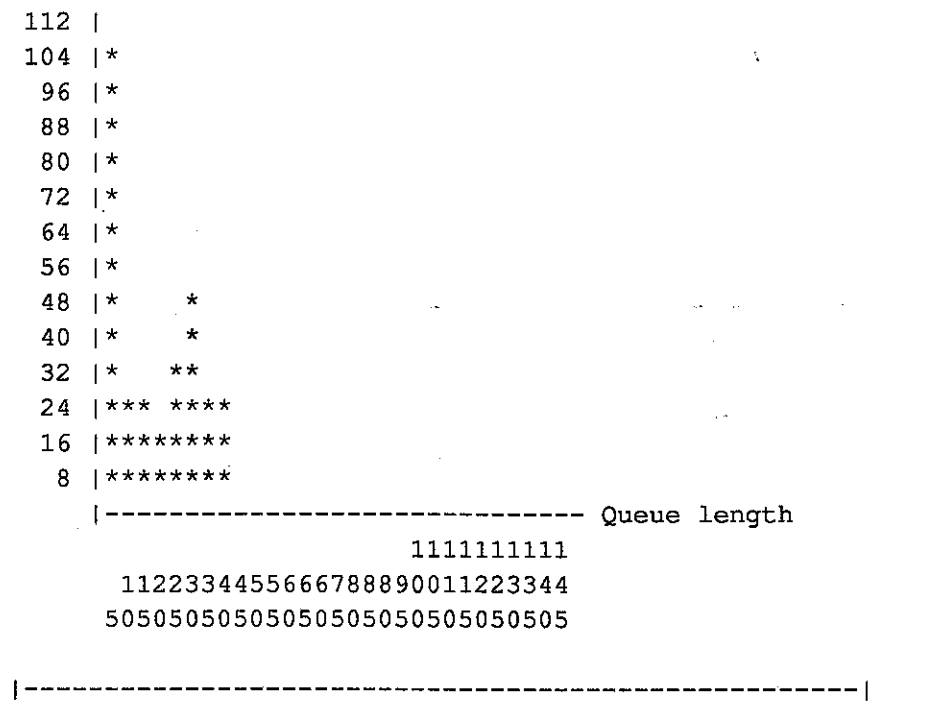


lane	Mean	Sd	Variance	Q Max	G Rear
1	20.40	13.838	191.486	68.3	59.2

Histogram for MFI, E junction, lane 1.

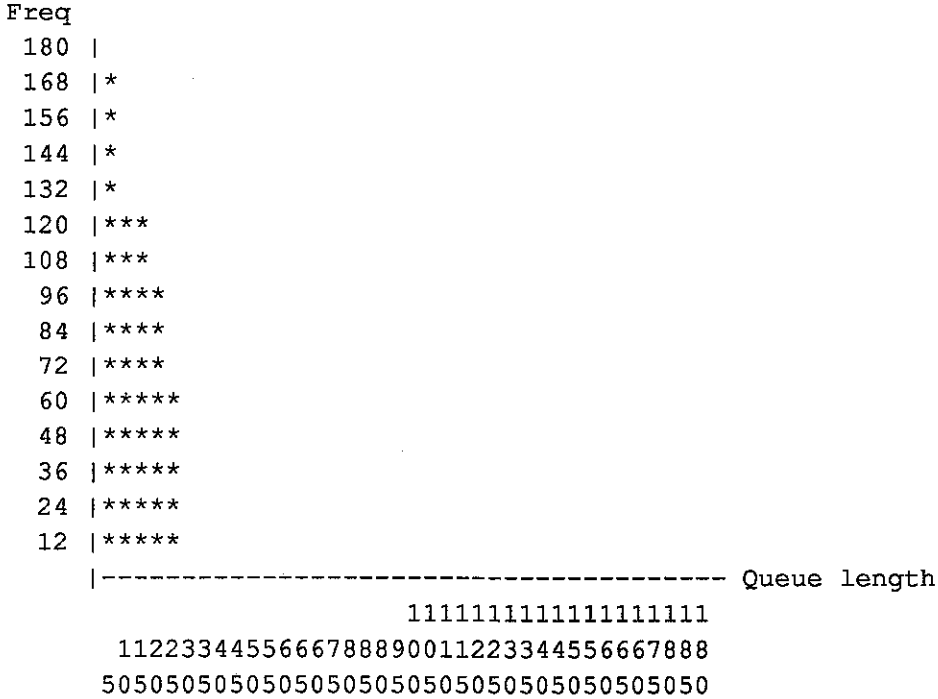


Histogram for MFI, E junction, lane 2.



lane	Mean	Sd	Variance	Q Max	G Rear
1	9.47	6.761	45.705	25.4	25.1
2	16.58	12.815	164.221	50.0	44.9

Histogram for Post Office, N junction, lane 1.



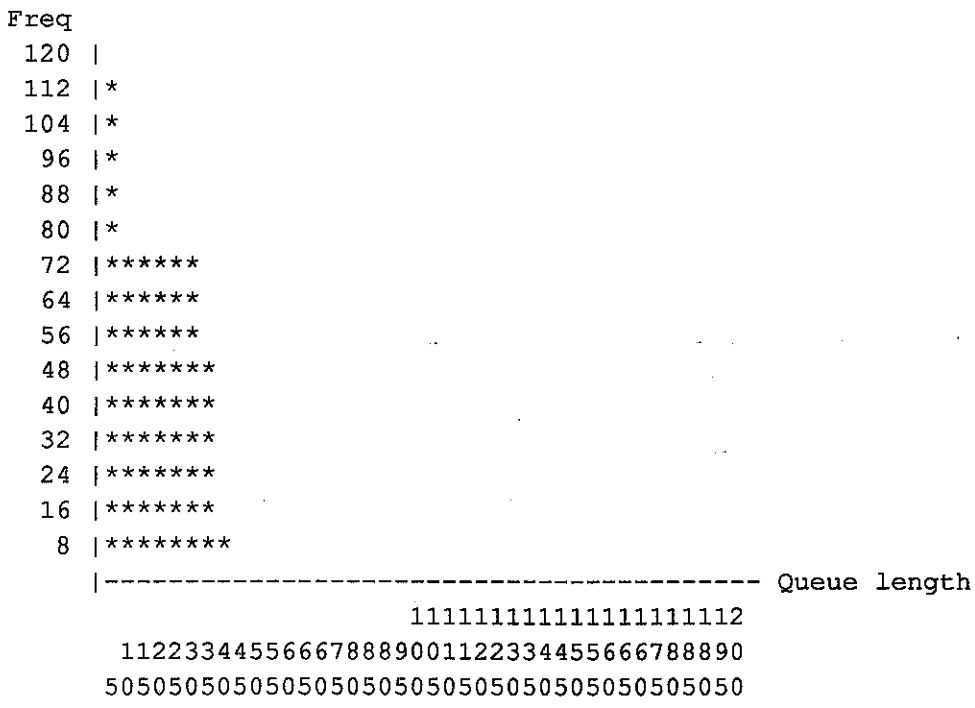
lane	Mean	Sd	Variance	Q Max	G Rear
1	9.95	6.769	45.817	30.3	27.1

Histogram for Post Office, S junction, lane 1.

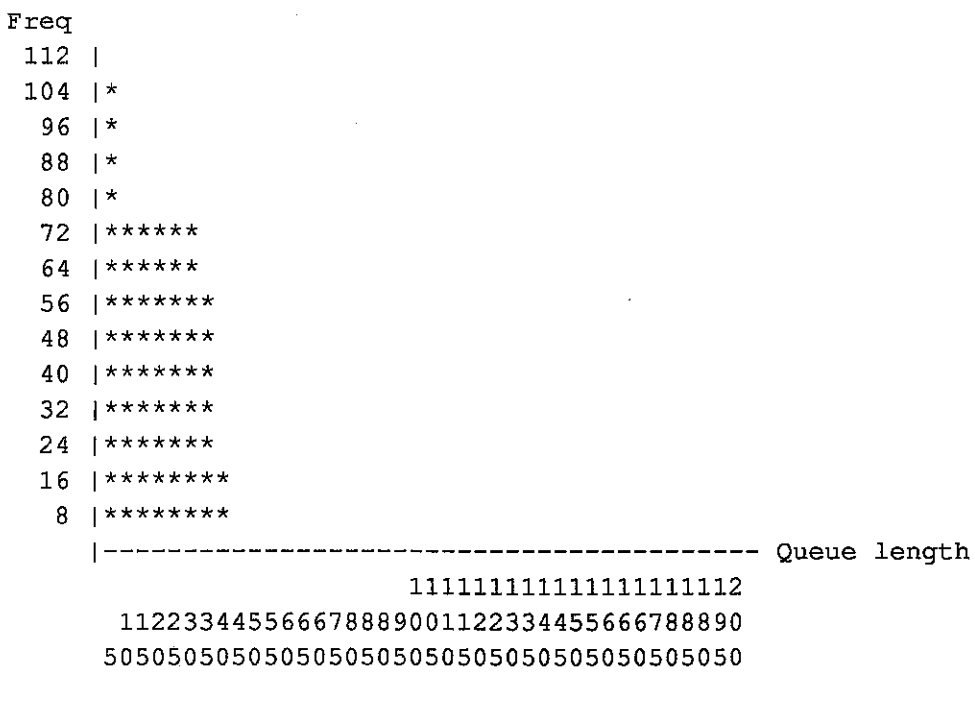
```
Freq
 135 |
 126 |*
 117 |*
 108 |*
  99 |*
  90 |*
  81 |*
  72 |***
  63 |******
  54 |******
  45 |*******
  36 |********
  27 |********
  18 |********
   9 |*********
     |------------------------------------ Queue length
                          11111111111111111
       112233445566678889001122334455666 78
      505050505050505050505050505050505050
```

lane	Mean	Sd	Variance	Q Max	G Rear
1	17.24	12.546	157.404	66.7	57.3

Histogram for Post Office, E junction, lane 1.



Histogram for Post Office, E junction, lane 2.



lane	Mean	Sd	Variance	Q Max	G Rear
1	15.91	10.453	109.260	49.6	43.8
2	16.55	10.827	117.231	51.7	45.5

Amount of time an exit is blocked

Junction	Direction	Start	Stop	Dur
MFI	W	366	367	1
		486	487	1
		494	496	2
		672	673	1
		837	839	2
Post Office	W	371	372	1
		382	383	1