NAME

getValueFileInfo − Read information from the header of a differential cost value file

SYNOPSIS

#include <qnet.h>

int getValueFileInfo(string *path***, int** *maxdim***, int*** *ndim***, int** *nval***, int** *trunc[]***);**

DESCRIPTION

The **getValueFileInfo**() function reads the header information from the file whose name is *path* and that is expected to contain differential cost value information for a state space that has dimension less than or equal to *maxdim*.

Upon return, *ndim* contains the dimension of the state space (the number of classes or queues) and *nval* contains the number of cost values per state. The state space truncation values are returned in the array *trunc[]*, which should contain space for at least *maxdim* elements.

RETURN VALUE

On success, the value **QNET_NORMAL** (defined to be zero) is returned, otherwise one of the following negative values is returned:

QNET_BAD_MAGIC
The input file is not recognized as a value data file.

QNET_BAD_DIMENSION
The state space dimension stored in the file exceeds *maxdim* or *maxdim* is negative.

QNET_BAD_FILE
The input file could not be opend or read from.

EXAMPLE

The following sequence can be used to read the header information from a value file:

```
int maxdim = 3;    // largest state space dimension expected
int ndim;          // actual dimension of state space
int nval;          // number of cost values per state
int N[maxdim + 1]; // state space truncations

getValueFileInfo(string("value.h"), maxdim, &ndim, &nval, &N[1]);
```

Here N[0] is not used; this follows the convention used in most of the QNET DP code. Once the header information is available an array to hold the value data can be allocated and the data read in. If *ndim*=3 and *nval*=1 then the code to do this would be

```
double*** h = (double***) makeValueArray(ndim, nval, &N[1]);
readValueFile(string("value.h"), ndim, nval, &N[1], h);
```

but if *nval*=2 (or any number greater than 1), an additional level of indirection is required:

```
double**** h = (double****) makeValueArray(ndim, nval, &N[1]);
readValueFile(string("value.h"), ndim, nval, &N[1], h);
```

In this case h[x1][x2][x3][0] is the first value for state (x1,x2,x3) and h[x1][x2][x3][1] is the second value for the same state.

SEE ALSO

readValueFile(3), **writeValueFile**(3), **makeValueArray**(3)

AUTHOR