**1-2** *What is the diameter of a torus?*

Assuming a 2-D torus, the diameter will be the number of links required to get from the center to one of the corners (any further and we can just go in the opposite direction). Thus we will have to traverse $(\frac{\sqrt{p}-1}{2})$ over each dimension, thus the diameter is $D = \sqrt{p} - 1$ for a $p$ processor torus.

**1-3** *What is the diameter of a tree network?*

If the processors are only located at the leaves of the tree then the longest path will be a path from one leaf to another through the root node. This will be $D = 2\log_2 p$ for a binary tree. If there is a processor at each nodes and leaves of the tree then the diameter is $D = 2\log_2(\frac{p-1}{2})$ for $p$ processors.

**1-4** *What is the diameter of a d-dimensional mesh?*

The diameter will be the number of links required to get between opposite corners in the mesh. We will have to proceed over all the links in each of the $d$ dimensions. Assuming a $d \times d \times d$ topology, there will be $p^{\frac{1}{d}} - 1$ links in each dimension and $d$ dimensions. Obviously, the total number of links will be the diameter, thus the diameter is $D = dp^{\frac{1}{d}} - d$.

**1-5** *Determine the route taken in a five-dimensional hypercube network from node 7 to node 22 using the deadlock free* e*-cube routing algorithm described in the chapter. Repeat for an* $8 \times 8$ *mesh assuming that the nodes are numbered in row order (across the rows starting at the top left corner).*

For a five dimensional hypercube each processor has a 5 bit address. The address of processor 7 is `00111` and the address of processor 22 is `10110`. The bitwise exclusive or of these to addresses is `10001` so the route will traverse two links. Routing from the highest dimension first, we get `00111 -> 10111 -> 10110` or 7 routes to neighbor 23 which then routes to its neighbor 22.

For an $8 \times 8$ mesh, the processors ids will consist of 6 bits, the 3 most significant bits representing the processor row number and the three least significant bits representing the processor column number. Thus processor 7 = `000 111` is on row 0 column 7 and processor 22 = `010 110` is on row 2 column 6. Assuming we progress over rows first

then columns, we would route as follows: `000 111 -> 001 111 -> 010 111 -> 010 110`. That is the route would be as follows 7,15,23,22.

**1-11** *A Multiprocessor consists of 10 processors, each capable of a peak execution rate of 200 MFLOPS. What is the performance of the system as measured in MFLOPs when 10% of the code is sequential and 90% is parallelizable?*

Assuming Amdahl's law applies, we can define speedup as

$$S(p) = \frac{p}{1 + (p-1)f}, \tag{1}$$

where $p$ is the number of processors and $f$ is the serial fraction. For this problem the serial fraction is $f = 0.1$ so we get

$$S(10) = \frac{10}{1 + 9 * .1} = 5.26 \tag{2}$$

With this speedup and assuming that the serial algorithm achieves the peak performance, we have the MFLOPs of the parallel algorithm as $200 * S(10) = 1052$ MFLOPs.

**1-12** *Suppose the best sequential algorithm for a problem requires $2n \log \sqrt{n}$ steps for $n$ data items. What is the minimum number of steps for a parallel algorithm to be cost optimal when using $n^2$ processors?*

A parallel algorithm is cost optimal if its cost is proportional to the cost of the best serial algorithm. We are given that the cost of the serial algorithm is $t_s = 2n \log \sqrt{n}$ steps. Since the parallel cost is given by $pt_p$ where $p$ is the number of processors and $t_p$ is the time required to execute the parallel algorithm on $p$ processors. If we assume that we will use $p = n^2$ processors then our parallel time $t_p$ must be proportional to $t_s/p = \frac{\log \sqrt{n}}{n}$ steps. Therefore, to be cost optimal, the parallel algorithm must be able to execute in $k\frac{\log \sqrt{n}}{n}$ steps where $k$ is a constant of proportionality. However, note that this function asymptotically approaches zero. Since the parallel algorithm must take at least one step, it will not be feasible to develop a parallel algorithm that maintains this relationship. Therefore there will be no cost optimal parallel algorithm that can solve this problem using $n^2$ processors!