# CS121 Problem Set 1

1. Consider the problem of adding a list of $n$ numbers. Assume that one person can add two numbers in time $t_c$. How long will one person take to add $n$ numbers?

   Now assume eight people are available for adding these $n$ numbers and that the list has already been divided into eight equal parts. Each person can add two numbers in time $t_c$. Furthermore, a person can pass on the result of an addition (in the form of a single number) to the person sitting next to him or her in time $t_w$. How long will it take to add $n$ numbers if

   a) All eight people sat in a circle.
   b) The eight people are sitting in two rows of four people each.

   If you are free to seat the eight people in any configuration, how would you seat them as to minimize the time taken to add the list of numbers? Is there a best configuration independent of $t_c$ and $t_w$?

2. Compare the execution of the following code to calculate the greatest common divisor of $x$ and $y$ as performed on a SIMD versus an MIMD architecture.

```
gcd(x,y) {
while (x != y) {
      if (x > y) x = x-y;
      else y = y-x;
      }
}
```

   Suppose two arrays $x = (x_0, ..., x_{n-1})$ and $y = (y_0, ..., y_{n-1})$ are each distributed across $n$ processors with $x_i$ and $y_i$ on processor i. We wish to find $z = (z_0, ..., z_{n-1})$, where $z_i = \gcd(x_i, y_i)$. Assuming that a compare or a subtract instruction each takes one time step, how many time steps would it take to calculate the result when $x = (52,24)$ and $y = (12, 64)$ using

   a) A two processor SIMD architecture.
   b) A two processor MIMID architecture.

   In each case, write a trace for the two processors, indicating clearly which instruction is executed at which time step.

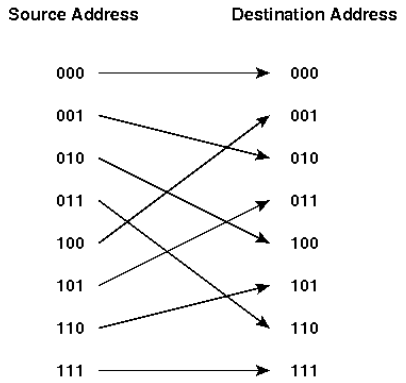
3. The *distance* between nodes $u$ and $v$ in a static network is the length of the shortest path from $u$ to $v$. Suppose the distance between $u$ and $v$ in a hypercube is $h$. Show the following.

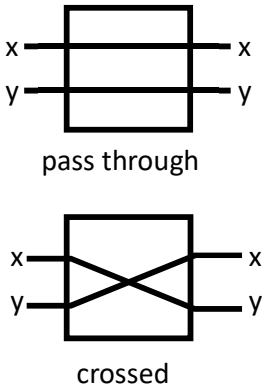
   a) There are $h!$ different paths of length $h$ from $u$ to $v$, where we allow some edges to appear in more than one path.

b) If we have a set of paths in which no edge appears in more than one path, then the set contains at most $h$ paths.

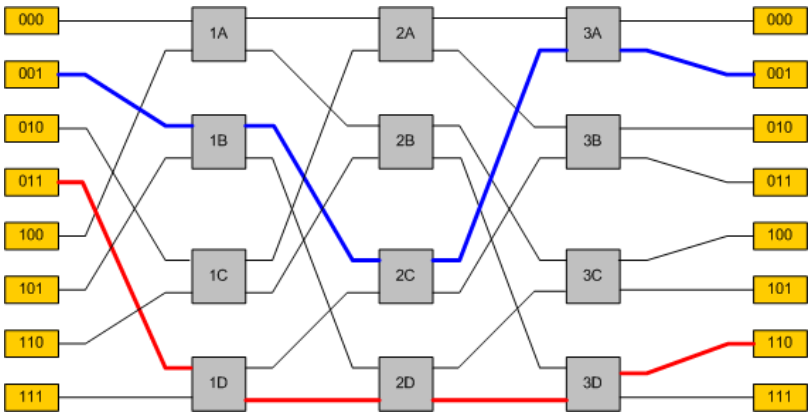
4. A *perfect shuffle* for a set of $2^n$ processors consists of connecting each processor $i$ to another processor $j$ whose ID in binary is a left rotation of $i$'s ID in binary. A perfect shuffle on 8 processors is shown below. For example, the left rotation of 100 is 001 (the 1 bit wraps around to the right side), so there is an edge between 100 and 001.

A *switch* consists of two inputs $x$ and $y$ and two outputs. If the inputs are *passed through* the switch, then the outputs are also $x$ and $y$. Otherwise, the inputs are *crossed*, and the outputs are $y$ and $x$. A switch is shown below.




An *Omega network* for a set of $2^n$ processors is an indirect network consisting of $n$ layers of perfect shuffles of the processors. Switches in the network can be set to either pass through or crossed in order to perform routing. For example, in the Omega network below, switches 1B, 2C and 3A are set to pass through to route from processor 001 to 001, while switch 2D is set to pass through and switches 1D and 3D are set to crossed to route from 011 to 110.



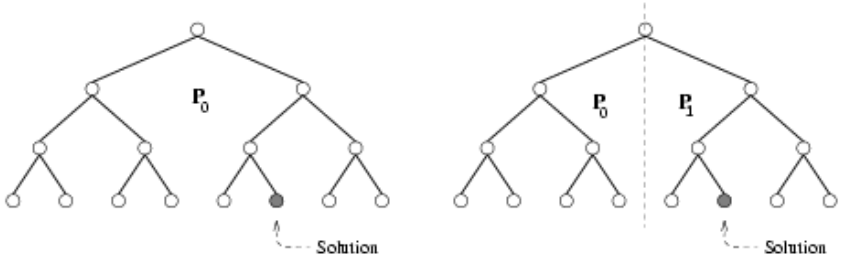

*Source*: http://www.xatlantis.ch/education/graphics/bus_design2.png

For this problem, devise an algorithm to route a message between two processors in an Omega network. Explain why your algorithm is correct.

5. Consider the search tree shown in the figure below, in which the dark node represents the solution.

   a) If a sequential search of the tree is performed using the standard depth-first-search (DFS) algorithm as shown in Figure Q1(a), how much time does it take to find the solution if traversing each link down the tree takes one unit of time? You may ignore the time for backtracking up the tree.

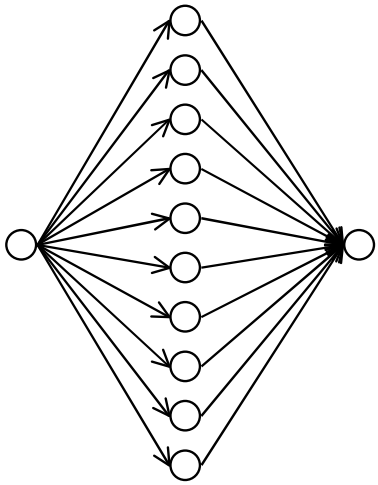
   b) Assume that the tree is partitioned between two processors as shown in Figure Q1(b). If both processors perform a DFS on their respective halves of the tree, how much time does it take for the solution to be found? What is the speedup? Is there a speedup anomaly? If so, can you explain the anomaly?




(a) DFS with one processor (b) DFS with two processors

   c) Suppose we have a large set of identical processors. Show that one processor can simulate the steps of p processors in O(p) time. Conclude that it is impossible to achieve superlinear absolute speedup. Why does this not contradict the phenomenon observed in Q1(b)?

6. The task graph shown in the figure below represents a parallel application. Each circle represents an indivisible task. There are 12 tasks: an initialization task, 10 computation tasks, and a finalization task. Each of the 12 tasks takes exactly one unit of time on one processor. The initialization task must complete before any of the computation tasks begin. Similarly, all 10 computation tasks must complete before the finalization task begins. The computation tasks can be executed in any order.



a) What is the maximum speedup that can be achieved if this problem is solved on a parallel computer with 2 processors? Give a diagram showing how the tasks would be allocated to processors.

b) What is the maximum speedup that can be achieved if this problem is solved on any parallel computer?

c) What is the minimum number of processors required to achieve the speedup given in part (b)? Give a diagram showing how the tasks would be allocated to processors.

7. Three students taking the Parallel Computing course have written parallel programs for their lab assignments.

a) Amanda's parallel program achieves a speedup of 9 on 10 processors. What is the maximum fraction $f$ of the computation that can be inherently sequential in her program?

b) Brian's parallel program executes in 225 seconds on 16 processors. By timing parts of his program, he determines that 9 seconds is spent performing initialization and finalization on one processor, and for the remaining 216 seconds all 16 processors are executing. What is the *scaled speedup* achieved by Brian's program?

c) Cindy times her parallel program on 10 processors and finds that for 270 seconds, all processors are active, and for 30 seconds, one processor is executing inherently sequential code. Assuming that the size of the sequential code does not increase as the problem size increases, what is the *scaled speedup* she can expect for $p$ processors, where $p$ = 20, 30, 40?

8. Consider the problem of adding a list of $n$ numbers on a $d$-dimensional hypercube with $p$ processors ($p = 2^d$). Initially, each processor holds a sublist of the numbers to be added and at the end of the computation, one processor has the sum of all the numbers. Assume that adding two numbers and communicating a number between two directly connected processors *each* takes one unit of time.

   a) Suppose $n = p$ and each processor initially holds one number. What is the parallel execution time, speedup, and cost?

      The *cost* of a parallel algorithm is defined as the product of its execution time and the number of processors it uses. We say the algorithm is *cost-optimal* if its cost is asymptotically equal to the running time of the best sequential algorithm to solve the problem. Is your algorithm cost-optimal?

   b) Suppose $n > p$ and each processor initially holds a sublist of $n/p$ numbers. Find a relationship between $n$ and $p$ for which the algorithm is cost-optimal.