CS 121 Problem Set 2 (Performance Analysis)

1) Consider the search tree shown in Figure Q1, 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?

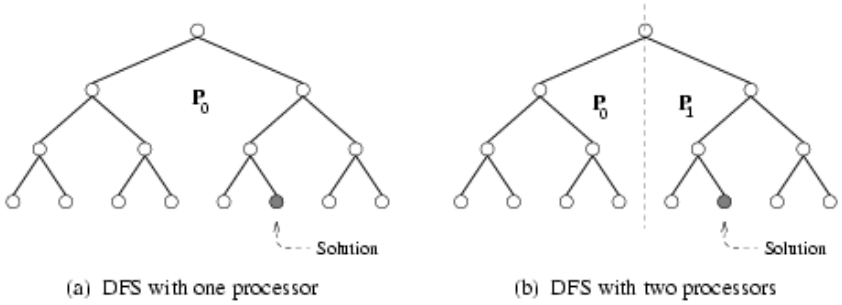

Figure Q1

   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)?

2) The task graph shown in Figure Q2 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.

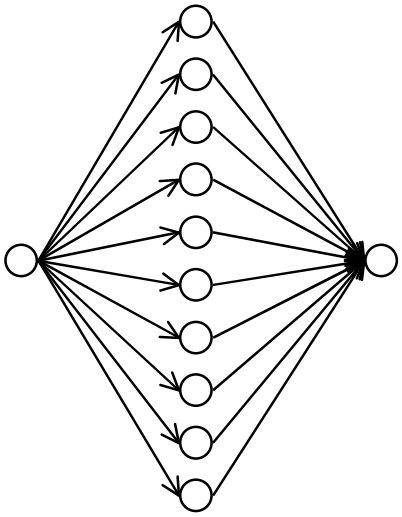

Figure Q2

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.

3) 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?

4) 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? 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 (if possible) for which the algorithm is cost-optimal.