Lindsey Bieda and Joe Frambach
Reduction and Parallel Problems
11.02.2011

19. Show by reduction that if the decision version of the SAT-CNF problem has a polynomial time algorithm then the decision version of the 3-coloring problem has a polynomial time algorithm.

20. In the dominating set problem the input is an undirected graph G, the problem is to find the smallest dominating set in G. A dominating set is a collection S of vertices with the property that every vertex v in G is either in S, or there is an edge between a vertex in S and v. Show that the dominating set problem is NP-hard using a reduction from the vertex cover problem.

23. In the disjoint paths problem the input is a directed graph G and pairs $(s_1, t_1), \ldots, (s_k, t_k)$ of vertices. The problem is to determine if there exist a collection of vertex disjoint paths between the pairs of vertices (from each s_i to each t_i). Show that this problem is NP-hard by a reduction from the 3SAT problem. Note that this problem is not easy.
HINT: Construct one pair (s_i, t_i) for each variable x_i in your formula F . Intuitively there will be two possible paths between s_i and t_i depending on whether x_i is true or false. There will be a component/subgraph D_j of G for each clause C_j in F. There will be three possible paths between the (s_i, t_i)'s pairs for each D_j . You want that it is possible to route any two of these paths (but not all three) through D_j.

2. You know that lots of famous computer scientists have tried to find a fast efficient parallel algorithm for the following Boolean Formula Value Problem:
INPUT: A Boolean formula F and a truth assignment A of the variables in F.
OUTPUT: 1 if A makes F true, and 0 otherwise.
Moreover, most computer scientists believe that there is no fast efficient parallel algorithm for the Boolean Value Problem. You want to find a fast efficient parallel algorithm for some new problem N. After much effort you can not find a fast efficient parallel algorithm for N, nor a proof that N does not have a fast efficient parallel algorithm. How could you give evidence that finding a fast efficient parallel algorithm for N is at least a hard of a problem as finding a fast efficient parallel algorithm for Boolean Formula Value problem? Be as specific as possible, and explain how convincing the evidence is.
Note that "fast and efficient" means poly-log time with a polynomial number of processors. The term "poly-log" means bounded by O($log^k n$) for some constant k.

3. Consider the problem of taking as input an integer n and an integer x, and creating an array A of n integers, where each entry of A is equal to x.

Give an algorithm runs in time O($log\ n$) on a EREW PRAM using n processors. What is the efficiency of this algorithm?

Give an algorithm that runs in time O($log\ n$) on a EREW PRAM using $n/log\ n$ processors. What is the efficiency of this algorithm?

Give an algorithm that runs in time O(1) on a CRCW PRAM using n processors. What is the efficiency of this algorithm?