1. A graph $G = (V, E)$ is called *bipartite* if the vertex set can be partitioned into sets X, Y such that no edge in E has both endpoints in the same set X or Y. Give a linear time algorithm to check if a graph is bipartite.

2. A graph is *2 vertex connected* if it has no *cut-vertex*. A cut-vertex is a vertex whose removal disconnects the graph. Give a linear time algorithm, using DFS, to check if a graph is 2-vertex connected.

3. A *strongly connected component* is a maximal set of vertices which are strongly connected. Give a linear time algorithm to identify all the strongly connected components of a graph. Your procedure should label each vertex so that vertices have the same label iff they are strongly connected.

4. Suppose you are given a parallel-processing subroutine which can find the median of n numbers in $O(\sqrt{n})$ time. How much time would you need to sort the n numbers?

5. Devise a divide and conquer algorithm for multiplying n complex numbers using $3(n-1)$ real multiplications.

6. Given an array $A[1..n]$ of numbers find i, j, $1 \leq i < j \leq n$ such that the sum $\sum_{k=i}^{j} A[k]$ is maximized.

7. Devise a divide-and-conquer algorithm for finding a closed knights tour which visits all the squares of an $n \times n$ chess board for all even $n \geq 6$.

Problems on Matchings and Flows

1. Show how to formulate the maximum bipartite matching problem as the problem of computing a maximum $s-t$ flow in a suitable graph. Why would the flow in this graph give you a matching in the original bipartite graph?

2. A graph is called k-regular if every vertex has degree k. Show that every k-regular bipartite graph has a perfect matching.

3. A *cut* in an undirected graph is a partition of the vertex set, V, into two sets U, W. The capacity of a the cut (U, W) is the sum of the capacities of the edges with one end-point in U and the other in W. How will you find the cut of minimum capacity?

4. Given a directed graph $G = (V, E)$, we want to find a subgraph $H = (V, E')$ so that every vertex in H has k incoming and k out going edges. Show how to do this.

5. Two paths are said to be edge (resp. vertex) disjoint if they do not share any edge (resp. vertex). You have to find the maximum number of edge (resp. vertex) disjoint paths between two given vertices u, v in a directed graph $G = (V, E)$. Show how to do this.

6. Place a maximum number of rooks on a $m \times n$ chess-board with some squares cut out.

7. Recall the maximum flow algorithm in which we routed flow along the shortest path in each step. A *phase* was defined as a sequence of steps in which the length of the shortest path remains the same. Show how to implement a phase in $O(n^2)$ time, where n is the number of vertices.

8. Consider a graph in which all edge capacities are 1. Show how to implement a phase in $O(m)$ time, where m is the number of edges.

9. Use the solution of the above problem to compute a maximum flow in a unit-capacity graph in $O(m\sqrt{m})$ time.

10. Use the solution to the above problem to compute a maximum matching in a bipartite graph on n vertices and m edges in $m\sqrt{n}$ time.

NP-completeness

1. In the *bin packing problem* we are given a set of n objects which have sizes $s_1, s_2, \ldots s_n$ where $0 \leq s_i \leq 1$. A subset of these objects can fit into a bin if the sum of the sizes of the objects in this subset is at most 1. The objective is to minimize the number of bins required to pack all objects. Show that this problem is NP-hard.

2. In the *machine scheduling problem* we are given m machines and a set of n jobs with processing times $p_1, p_2, \ldots p_n$. A job can be scheduled on any machine. The *makespan* of a schedule is defined as the time by which all jobs complete.Show that finding a schedule of minimum makespan is NP-hard.

3. Given a graph with edge weights (may be negative) and two vertices s, t show that the problem of finding the shortest simple path between s and t is NP-hard.

4. Give a polynomial time algorithm to find a minimum vertex cover in a bipartite graph.