1 Solution to Problem 1

We prove the fact assuming one connected component, for the general proof the argument goes componentwise. Let T be the DFS tree, L be the set of leaves, n be the total number of vertices. To see that $V - L$ is a vertex cover, note that in the DFS tree, all edges are either forward edges which go from ancestor to descendant or back edges which go from descendants to ancestors. Thus there is no edges between leaves, and hence $V - L$ is a valid vertex cover.

To see that its within two times the optimal, note that any vertex cover of the graph must also be a vertex cover of the tree. Note, for a tree we can assume that none of the leaves are in the optimal vertex cover, it is always more benefecial to take the parent if a leaf is in the cover. For any vertex cover VC we have $\sum_{v \in VC} deg(v) \geq |E|$, since it must cover every edge. Also we have $\sum_{v \in VC} deg(v) + \sum_{v \notin VC} deg(v) = 2|E|$, which gives us $\sum_{v \notin VC} deg(v) \leq 2|E|$. Since all the leaves are out of the vertex cover, and for any nonleaf the degree is atleast 2, we get, $2|E| \geq |L| + 2(n - |L| - VC)$ which on rearranging gives us $VC \geq \frac{n-|L|+1}{2}$. Thus, for a tree any vertex cover must have these many vertices, which is more than half of the vertices returned by the algorithm, proving the claim of 2-approximation.

2 Solution to Problem 2

The Greedy Algorithm for the Vertex Cover problem can not achieve a 2-approximation; in fact, it cannot guarantee anything better than an $O(\log n)$-approximation ratio. This is shown by the following example:

Consider a bipartite graph with n vertices in set A, and $(n * \log n)/2$ vertices in B. Each vertex in A has degree n. B is divided into $B_0, ...B_{\log n - 1}$ groups each containing $n/2$ vertices. Degree of each vertex in B_i is $n/2^i$. Notice that the number of edges going out of A as well as B is n^2; so this bipartite graph is valid and can be constructed. The optimal vertex cover of this graph is clearly A which is n vertices. However, the greedy solution could first take all of the vertices in B_0, since each has degree n. This leaves all of the vertices in A with degree $n/2$ (since size of B_0 was $n/2$). The greedy algorithm could then select all the nodes from B_1, leaving the degree of vertices in A to be $n/4$, and so on. So the greedy algorithm could end up picking all of B which is $1/2 * n * (\log n)$ vertices; this proves the lower bound of $O(\log n)$ on the greedy algorithm for vertex cover.

3 Solution to Problem 3

T is the minimum-cost tree spanning the vertices V'. T' is the minimum spanning tree on on V'. We wish to find a Steiner Minimum Tree (SMT), which

is the minimum cost tree spanning V'. To show that T' is a 2-approximation. Construct an Euler tour over V' by doubling every edge in T'. Call this Euler tour W. So $wt(W) \leq 2 * wt(T')$. Further, due to the triangle inequality on weights, one can construct a spanning trip on V' by traversing the vertices in V' in the order they appear in W. The cost of this spanning tree is $\leq wt(W)$. Therefore, $wt(T') \leq wt(W) \leq 2 * wt(T)$. This proves the 2-approximation.

4 Solution to Problem 4

Given graph G find the smallest 2-matching of G. Note that the 2-matching is a disjoint union of cycles which span all the vertices, and each cycle is of length atleast 3. Let the cycles in the optimal 2-matching be $C_1, C_2, \cdots, C_k$. Note that the optimal TSP tour is also a 2-matching and hence $OPT \geq 2MATCHING$. Note that the cost of the 2-matching is atleast $3k$ since each cycle is of length atleast 3 and each edge is atleast 1. Now form a tour from the cycle cover as follows: Pick edges $e_i = (u_i, v_i)$ from each cycle arbitrarily. Remove these edges and add the edges $(u_1, u_2), (v_2, u_3), \cdots, (v_{k-1}, u_k), (v_k, v_1)$. Now we have the following tour: Start from v_1, go around C_1 till you reach u_1, go to u_2, go around C_2 till v_2, to u_3, around C_3 and so on till you reach v_k and then come back to v_1. Note we have added k edges and deleted k edges, thus the cost increases by atmost $2k - k = k$. Thus the cost of this tour ALG satisfied $ALG \leq 2MATCHING + k$. Thus $ALG/OPT \leq ALG/2MATCHING \leq 1 + \frac{k}{2MATCHING} \leq 1 + \frac{k}{3k} = \frac{4}{3}$

5 Solution to Problem 5

A 2-approximation algorithm for the SONET ring loading problem : For each call (i, j) in C, chose the shorter among clockwise and counterclockwise around the ring.

Proof of 2-approximation: Let the link with maximum load be L_k on link $(k, k+1)$. Let the load on this link in the optimal routing be L_{k^*}. Let $m = k + n/2 \mod n$. Let load on the link $(m, m+1)$ (the link directly opposite $(k, k+1)$) in the optimal routing be L_{m^*}. Look at the L_k paths passing through link $(k, k+1)$; none of these paths can pass through $(m, m+1)$ as we chose the shortest path and therefore the length is $\leq n/2$. Moreover, in any routing, each of these L_k calls will pass through one of the two links, $(k, k+1)$ and $(m, m+1)$. Therefore, $L_{k^*} + L_{m^*} > L_k$. It immediately follows that the optimal load is at least $0.5 * L_k$, completing the proof.