Problem Set 1 - Solutions

1. **Solution:** Since every face has size at least three, and each edge is in exactly two faces, $3f \leq 2m$ (here, f is the number of faces). Substituting into Euler's formula we get $2 = n - m + f \leq n - m/3$. Or, equivalently, $m \leq 3n - 6$.

2. **Solution:** The crucial observation is that in *any* planar graph with no self loops and no parallel edges, there always exists a node whose degree is at most 5. To see this, note that by the sparsity of planar graphs, $\sum_{v \in V} \text{degree}(v) = 2m \leq 6n - 12$.

   The algorithm is:

   1: $T \leftarrow \emptyset$
   2: **while** G is not empty **do**
   3: choose an arbitrary node v s.t. $deg(v) \leq 5$
   4: let e be the minimal weight edge incident to v
   5: $T \leftarrow T \cup \{e\}$
   6: contract e, eliminating any parallel edges that occur by only keeping the one with minimal weight.
   7: **end while**

   First, we may assume that G contains no self loops and parallel edges since we can detect and delete them in linear time (for parallel edges delete all but the lightest edge). The correctness of the algorithm follows from the two claims (a) and (b) by a simple induction on the number of nodes n.

   The number of iterations is at most n since the contraction at each iteration reduces the number of nodes of the graph by one. We next argue that each iteration takes constant time. We can maintain a list L of nodes with degree at most 5 and a table with the degree of each vertex. Finding the node v in line 3 takes constant time (take any element in the list L). Since the degree of v is constant, finding the minimum edge incident to it also takes constant time. In our representation contraction takes constant time. The degree table only changes for v and its neighbors (constant number of them). Similarly, parallel edges are only created between neighbors of v and can be identified in constant time per edge by maintaining the edges incident to each node in a hash table with endpoints of the edges as keys. Therefore, all executions of line 6 require total linear time.

3. **Solution:** Let T^* be the spanning tree of G^* comprising of the edges not in T. Consider f_∞ as the root of T^*. We first argue that, for a non-tree edge e, the subtree of T^* rooted at the endpoint (face) of e further from f_∞ in T^* consists of the faces of G enclosed by $C_T(e)$, the elementary cycle of e with respect to T. To see this note that, by cycle-cut duality, the edges of $C_T(e)$ form a cut in G^*, whose only edge not in T is e. Hence, since T^* is a spanning tree of G^*, all faces enclosed by C must be descendants of e in T^*.

   By the claim above, it suffices to compute, for each face f, the weight of the subtree of T^* rooted at v. Clearly this can be done in linear time by accumulating the weights working up from the leaves of T^* towards the root.