CS 170

1. Study Group

None

2. 2-SAT

(a) Since G_I has a strongly connected component containing both x and $\bar{x}$ for some variable x, so there is a path from x to $\bar{x}$, which means $x \Rightarrow \bar{x}$, and this just means $\bar{x}$ is true. There is also a path from $\bar{x}$ to x, which means x is true. This contradicts with the prior argue. So I has no satisfying assignment.

(b) Take any sink component, and assign variables so all the literals in this component are True. Because of how we define the graph, there is a corresponding source component which has the negations of all literals in this component. Remove this source/sink com- ponent pair, and repeat the process until the graph is empty. Since we set components to true in reverse topological order, there is no implication from a true literal to a false literal. Since no literal and its negation are in the same SCC, we never try to set a variable to be both true and false. So this produces an assignment satisfying all clauses.

(c) The graph construction can be done in $O(m + n)$, the assignment can be done in the process of SCC construction, which also can be done in linear time.

3. Perfect Matching on Trees

Algorithm Description:

```
function tree_perfect_match:
    can_match = true
    mark = [false] * n
    function subtree_match(u):
        if not can_match:
            return
        subtree_can_match = true
        mark[u] = true
        for v in E[u]:
            if mark[v]:
                continue
            if not subtree_match(v):
                if subtree_can_match:
                    subtree_can_match = false
                else:
                    can_match = false
                    break
        return not subtree_can_match
    subtree_match(0)
    return can_match
```

Correctness:

From the root, we can recursively check if there is a perfect matching in a subtree. For node x's each subtree T_i, there are three cases. First, all the subtrees of T_i has a perfect matching, then x can have at most one this kind of T_i, or there is no perfect matching for the entire tree. Second, only one T_i does not have a perfect matching, and the remaining vertect is the root of T_i, so x must be paired with it. Then node x is perfect matched. All the other cases does not have a perfect matching.

Runtime Analysis:

O($|V| + |E|$)

4. Huffman Coding

(a) Each character contains k bits, so $S(F) = mlog(n)$.

(b) If each character appears equally in the file F, the $E(F) = 1$.

(c) O($log(n)$).

5. Minimum Spanning k-Forest

(a) Similar to the Minimum Spanning Tree proof.

(b) Use Kruskal's algorithm until there are k connected component. Then runtime is O(|E|log|V|).