Problem 1. Construct a smallest graph where the approximate algorithm we discussed produce a solution that is twice the minimum. In notes we gave you an example graph on 4 vertices, you should construct an even smaller example.

Problem 2. Design an efficient greedy algorithm to find vertex cover in a tree on n vertices. The runtime of your algorithm should be $O(n)$.*Hint:* Use the fact that a tree must have a leaf.

Problem 3.

1. Let G be a connected graph. Prove that in $G - v$ (the graph obtained by removing the vertex v and its incident edges), every connected component has a vertex w_i that is adjacent to v.

2. Let G be a connected graph with all its vertices having degree $\leq k$. Furthermore, suppose there is a vertex in G which has degree $< k$. Prove by induction that we can properly color G with k colors.

3. Let G be a connected graph such that all its vertices have degrees $\leq k$ (for $k \geq 0$) except one. That one vertex could have degree $> k$. Show that we can properly color G with $k + 1$ colors.

Problem 4. Let $\Delta(G)$ be the maximum degree of the graph G. Prove by induction on the number of vertices, that any graph G has a proper vertex coloring with $\Delta(G) + 1$ colors.

Problem 5. Given a directed graph $G = (V, E)$. Our goal is to find the largest subset of edges $E' \subseteq E$, such that the subgraph $G' = (V, E')$ (the graph on the same set of vertices as G induced by the subset of edges E') has no directed cycle.

Design a 2-approximate algorithm for this maximum acyclic subgraph problem. Also prove that this approximation ratio is tight (i.e. construct a small graph on like 3 edges, where your algorithm actually is only half as good).

Hint. Arbitrarily order vertices and consider forward and backward edges ((v_i, v_j) is a forward edge if $i < j$ in the fixed order). Find a subset of edges containing at least half of the edges that doesn't induce a cycle.

Problem 6. The maximum clique problem is that of finding the largest subset of vertices in the graph such that every pair of them is adjacent. As we saw earlier this an NP-hard problem (by a simple reduction from the maximum independent set problem.

In class we proved the following theorem (see lecture notes)

Theorem 1. If $P \neq NP$, then there is no k-absolute approximation algorithm for the maximum independent set (MIS) problem.

Prove that if $P \neq NP$, then there is no k-absolute approximation algorithm for the maximum clique problem. *Hint:* The proof is almost identical to the proof of the above theorem in notes.

Problem 7. In class we gave a 2-approximation algorithm for the minimum vertex cover problem. We also noted that vertex cover is a special case of the set cover problem. But here since each element of U (each edge) appears in at most 2 sets (the sets of edges incident on each vertex), instead of using the $\log n$-approximation for vertex cover we could do better. Now suppose that every element of U appears in at most 3 sets what happens. Formally let U be a set of n elements and $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ such that each S_i is a subset of U. Suppose every element of U appears in at most 3 subsets in $\mathcal{S}$. Design a 3-approximation algorithm for this problem.

You may want to look at this problem as the vertex cover problem in a hypergraph. A hypergraph is $H = (V, E)$, where V is a vertex sets and E is a collection of hyperedges, each hyperedge is a subset of V (instead of 2-subset of V, it could contain more than 2 vertices). In this problem we can think of U as a set of hyperedges where each hyperedge can be covered by choosing at least one of the vertices in that hyperedge. Note that in this case our hyperedges contain at most 3 vertices.

Problem 8. We give another 2-approximate algorithm for the vertex cover problem.

Definition 1. A **matching** in an undirected graph $G = (V, E)$ is a subset of edges such that no two edges in the set are incident on the same vertex. In other words a matching is a set of disjoint edges.

There are polynomial time algorithms to find maximum matching in G.

Definition 2. A **maximal matching** is a matching that is not proper subset of any other matching. In other words a maximal matching is one that you cannot add any other edge to extend it.

It is easy to see that a maximum matching matching must be a maximal matching, but a maximal matching need not be a maximum matching.

1. Construct a small example graph, where a maximal matching is not a maximum matching. You can make a graph on 4 vertices.

2. Design an $O(|E|)$ algorithm to find a maximal matching in G.

3. Let M be a maximum matching in a graph G. Prove that for any maximal matching M', $|M'| \geq \frac{|M|}{2}$. In other words prove that above algorithm is 2-approximate algorithm for the maximum matching problem. Note that maximum matching problem is not NP-hard there is a polynomial time algorithm.

Problem 9. In this problem we will develop another 2-approximate algorithm for the minimum vertex cover problem. The algorithm is quite simple and given below.

Algorithm 1 : VertexCoverViaMaximalMatching

```
C ← ∅                                          ▷ Initialize an empty vertex cover C
Compute a maximal matching M' in G             ▷ Call the above algorithm
for each edge e = (u, v) ∈ M' do
    C ← C ∪ {u} ∪ {v}                          ▷ Add both u and v to C
return C
```

1. Let OPT be the minimum vertex cover in G and let M' be a maximal matching in G. Prove that $OPT \geq |M'|$.

2. Let C be the cover returned by the above algorithm, i.e. $C = \{u, v \;:\; (u, v) \in M'\}$. Prove that C is a vertex cover.

3. Prove that the above algorithm based on maximal cover is 2-approximate.

Problem 10. content...