1 Colorings [30 points]

For a simple undirected graph $G = (V, E)$ and positive integer $l$, a $l$-vertex coloring is a map $C : V \rightarrow \{1, 2, \ldots, l\}$ such that for all edges $(v_i, v_j) \in E$, $C(v_i) \neq C(v_j)$.

An $l$-edge coloring is a map $F : E \rightarrow \{1, 2, \ldots, l\}$ such that for any two distinct edges $e_i, e_j$ incident on a common vertex $F(e_i) \neq F(e_j)$.

We now define two decision problems for coloring. An input instance to both the problems is a simple undirected graph $G$ and a positive integer $k$.

**Vertex Coloring Decision Problem**: Is there a $k$-vertex coloring of $G$?

**Edge Coloring Decision Problem**: Is there an $k$-edge coloring of $G$?

We can also define the search versions of these problems. An input instance for both the following problems is a simple undirected graph $G$.

**Vertex Coloring Search Problem**: Find a vertex coloring of $G$ that uses as few colors as possible.

**Edge Coloring Search Problem**: Find an edge coloring of $G$ that uses as few colors as possible.

(a) [20 points] Given an algorithm to the **Vertex Coloring Decision Problem** that runs in time $f(n)$ on a graph with $n$ vertices for some polynomial $f$, construct an algorithm for the **Vertex Coloring Search Problem** that runs on a graph with $n$ vertices in $g(n)$ time for some polynomial $g$.

(b) [10 points] Given an algorithm to the **Edge Coloring Decision Problem** that runs in time $f(n)$ on a graph with $n$ vertices for some polynomial $f$, construct an algorithm to the **Edge Coloring Search Problem** that runs on a graph with $n$ vertices in $g(n)$ time for some polynomial $g$.

**SOLUTION**: (a) To figure out the minimum number of colors (chromatic number)
required to vertex color $G$, call the Vertex Coloring Decision Problem with every color count from 1 to $n$ (or do a binary search). Suppose that $G$ has chromatic number $k$.

Now add as many missing edges as possible to $G$ with out increasing its chromatic number (add an edge, keep it if the chromatic number stays the same or toss it away if it does increase). We end up with a graph $G'$ with chromatic number $k$ such that adding any edge increases the chromatic number. Note that any $k$-coloring of $G'$ is also a valid $k$-coloring for $G$.

Pick any maximal subset $S_1$ of vertices with no edges between them. Any $k$-coloring of $G'$ should color all the vertices in $S_1$ with the same color (if there is a $k$-coloring with $\{u, v\} \in S_1$ colored differently, then $(u, v)$ could have been added to $G'$ with out increasing the chromatic number). Similarly pick another maximal subset $S_2$ of vertices from $V \setminus S_1$ such that there are no edges between vertices in $S_2$. Repeat this procedure until all the vertices have been picked. Say we end up with $k'$ subsets $S_1, S_2, \ldots, S_{k'}$. Any $k$-coloring of $G'$ will color all vertices in $S_i$ with the same color.

Claim 1: $k' \geq k$. Otherwise we have an obvious $k'$-coloring of $G'$ and therefore a $(< k)$-coloring of $G$.

Claim 2: $k' \leq k$. Suppose a $k$-coloring assigns color $c_1$ to set $S_1$. This forces $S_2$ to be colored with a color $c_2 \neq c_1$ because there is at least one edge from every vertex in $S_2$ to a vertex in $S_1$. $S_3$ is forced to be colored with color $c_3 \in \{c_1, c_2\}$ because every vertex in $S_3$ is connected to at least one vertex in each of $S_1$ and $S_2$. By induction, we can prove that every set $S_i$ needs to be colored differently as every vertex in $S_i$ is connected to at least one vertex in each of $\{S_1, S_2, \ldots, S_{i-1}\}$. Since the graph is $k$-colorable, it follows that $k' \leq k$.

We now have a $k$-partition of vertices such that vertices with in each partition are non-adjacent. Color each partition with a different color and we are done.

(b) Edge Coloring Decision Problem → Vertex Coloring Decision Problem → Vertex Coloring Search Problem → Edge Coloring Search Problem

Given a polynomial time algorithm for Vertex Coloring Search Problem, we can construct a polynomial time algorithm Edge Coloring Search Problem as follows. To edge color $G$, construct graph $G'$: for every edge in $G$, create a vertex in $G'$. If edges $e_1, e_2$ are incident on a common vertex in $G$, then add an edge between the corresponding vertices in $G'$. $G$ is $k$-edge colorable if and only if $G'$ is $k$-vertex colorable. Further, a $k$-coloring for $G'$ can be used to easily construct a $k$-edge coloring for $G$: color an edge in $G$ with the color of the corresponding vertex in $G'$.

We have already seen how to use a poly time algorithm for Vertex Coloring Decision Problem to construct a poly time algorithm for Vertex Coloring Search Problem.

Since Edge Coloring Decision Problem is NP-complete (http://dx.doi.org/10.1137/0210055), Edge Coloring Decision Problem can be used to solve 3SAT (using polynomial num-
number of invocations) which can be used to solve Vertex Coloring Decision Problem.

2 Luby's algorithm [30 points]

Chain of credit for this line of analysis: Guy Blelloch→ Kanat Tangwongsan→ Anupam Gupta→ Roger Wattenhoffer (ETH Zurich)→ (self?)

For a simple undirected graph \( G = (V, E) \), a subset \( S \subseteq V \) of vertices is independent if for any pair of vertices \( v_i, v_j \in S \), \( E \) does not contain the edge \( (v_i, v_j) \). In other words, an independent set of a graph is a subset of its vertices such that no two vertices are adjacent. An independent set \( S \) is maximal if for all \( v \in V \setminus S \), \( S \cup \{v\} \) is not an independent set.

The following is a version of Luby's algorithm for computing maximal independent set in parallel. The set \( S \) is initially \( \emptyset \) and will eventually become the maximal independent set.

1. Each vertex picks a number uniformly at random from \([0, 1]\) \(^1\).
2. In parallel, for each vertex, check whether the number picked by the vertex is greater than the the numbers picked by all its neighbors. If so, select such a vertex for inclusion in the set \( S \) and removal from \( G \). Also, mark all the neighboring vertices of such a vertex for removal.
3. Pack all the nodes selected for inclusion in \( S \) in into the set \( S \). Create a smaller graph \( G' \) induced by vertices that have not been removed in the earlier step.
4. If \( G' \) has at least one vertex, recursively compute a maximal independent set \( S' \) of \( G' \). Else, let \( S' \) be \( \emptyset \). Return \( S \cup S' \).

(a) [5 points] Argue that this algorithm returns a maximal independent set in (expected) polynomial time, and indicate how each of the steps can be parallelized.

(b) [20 points] To evaluate the work and depth of the algorithm, we want to find the reduction in the size of the graph at each level of the recursion. You will show that this algorithm eliminates at least half the edges on average at each level of the recursion. Let \( N(u) \) denote the set of vertices adjacent to vertex \( u \), and let \( d(u) = |N(u)| \). For a pair of adjacent vertices \( u, v \), define the event \( E_{u \rightarrow v} \) to be the one in which the number picked by \( u \) is greater than all of the numbers picked by vertices in the set \( N(u) \cup N(v) \setminus \{u\} \). If \( I(\cdot) \) denotes the indicator random variable, show that

\[
d(v)E[I(E_{u \rightarrow v})] + d(u)E[I(E_{v \rightarrow u})] \geq 1.
\]

\(^1\)While it is not clear how this can be done, this selection procedure can be replaced by one that selects a random integer from the set \( \{1, 2, \ldots, |V|^4\} \). The analysis a bit messy.
Let $X$ denote
\[ \sum_{(u,v) \in E} d(v)I(E_{u \to v}) + d(u)I(E_{v \to u}). \]

It follows that $\mathbb{E}[X] \geq m$. Complete the proof from here with precise arguments.

**Hint:** Show that for a particular choice of random numbers selected by vertices, deleted edges are not over counted by more than a certain factor in $X$.

(c) [5 points] Analyse the expected work and depth of this algorithm.

**SOLUTION:**

(a) Proof by induction on the cardinality of vertex set: The algorithm is correct when the graph has just one vertex. Suppose that the algorithm is correct for all graphs with less than $n$ vertices. Let $G$ be a graph on $n$ vertices. According to inductive hypothesis, $S'$ is a maximal independent set of $G'$. By construction, $S \cup S'$ is an independent set since no vertex in $S$ is adjacent to any in $S'$. Suppose if possible that $S \cup S'$ is not maximal. Then there exist a vertex $v$ such that $S \cup S' \cup \{x\}$ is an independent set. Clearly, $v$ can not be one of the vertices discarded in step 2. Therefore $v$ is a vertex in $G'$. This would imply that $S' \cup \{x\}$ is an independent set which contradicts the fact that $S'$ is a maximal independent set.

Parallelizing the algorithm:

1. Obviously parallel (Work: $O(|V|)$, Depth: $O(1)$)

2. For each vertex, use a binary tree over the array containing its adjacency list to figure out the maximum number picked by the neighbors. (Work: $O(|E| + |V|)$, Depth: $O(\log |V|)$)

3. Constructing the set $S$ would require the parallel pack operation described in class. Constructing $G'$ would require a pack operation on the adjacency list of each vertex of $G$. (Work: $O(|E| + |V|)$, Depth: $O(\log |V| + \log |E|)$)

4. Requires compacting the representation of $S$ and $S'$. Assuming they are both represented as arrays, it is easy to create a new array of size $|S| + |S'|$ and copying the elements in $S$ and $S'$ over in to this new array in parallel (Work: $O(|V|)$, Depth: $O(1)$).

(b) The probability of $E_{u \to v}$ is exactly the probability that a certain number is greatest among $|N(u) \cup N(v)|$ numbers picked independently and uniformly randomly, which is exactly $1/(|N(u) \cup N(v)|) \geq 1/(d(u) + d(v))$. Therefore,
\[ d(v)\mathbb{E}[I(E_{u \to v})] + d(u)\mathbb{E}[I(E_{v \to u})] \geq 1. \]

For a choice $C$ of random numbers numbers picked by the vertices, suppose that the set of events $\{E_{u \to v} \mid u \to v \in S_C\}$ happens. If we interpret the term $d(v)I(E_{u \to v})$
to mean that the event $E_{u \rightarrow v}$ removes $d(v)$ edges incident on $v$, we can argue that $X$ overcounts the number of edges removed by at most a factor of 2. Consider an edge $(x, y)$. For any choice $C$, there exists at most one vertex $w \in N(x)$ such that $I(E_{w \rightarrow x}) = 1$. Similar is the case for $y$ (say $I(E_{z \rightarrow y}) = 1$). Then the edge $(x, y)$, if removed, is counted at most twice: at the events corresponding to $w \rightarrow x$ and $z \rightarrow y$. Therefore, for a choice $C$, the number of edges removed is at least $X/2$. And since the expected value of $X$ is $m$, the claim follows.

(c) Expected work is $O(|E| + |V| \log |E|)$ and the depth is $O(\log^2 |E|)$ with high probability (since there are $O(\log |E|)$ rounds of recursion in expectation, and the number of edges is halved in expectation each round).