CS120: Intro. to Algorithms and their Limitations Hesterberg & Vadhan

Section 6: Graph Coloring and Independent Sets

Harvard SEAS - Fall 2022 *Oct. 13, 2022*

1 Graph Coloring

The graph coloring problem is an important concept in graph theory which is a special case of graph labelling. Its applications to the real world include register allocation, team matching, and map coloring.

Definition 1.1. For an undirected graph $G = (V, E)$, a (proper) k-*coloring* of G is a mapping $f : V \to [k]$ such that for all edges $\{u, v\} \in E$, we have $f(u) \neq f(v)$.

An improper coloring is when the same color is assigned to vertices that share an edge, but we will work with proper colorings unless explicitly stated.

Example:

Input	: A graph $G = (V, E)$ and a number k
Output	: A k-coloring of G (if one exists)

Computational Problem Graph Coloring

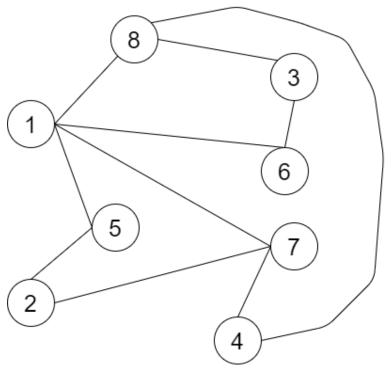
Alternatively, we are given a graph G and we wish to find a proper coloring using as *few* colors as possible. What problem is this an opposite of?

Coloring is the opposite of connected components!

- Coloring: partition V into as few sets as possible such that there are no edges within each set.
- Connected components: partition V into as many sets as possible such that there are no edges crossing between different sets.

Question 1.2. This question illustrates how the greedy graph coloring algorithm is sensitive to the order in which we process the nodes.

Consider the following graph:



1. How many colors do we need if we apply the `GreedyColoring` algorithm by processing the graphs in the following order? $\{2, 7, 6, 1, 5, 4, 8, 3\}$
2. How many colors do we need if we apply the `GreedyColoring` algorithm by processing the graphs in the following order? $\{2, 5, 6, 8, 4, 7, 1, 3\}$
3. Now process the vertices in BFS order and apply the greedy algorithm; i.e., use the `BFSColoring` algorithm from Lecture 13. How many colors do we need?
4. In general, given a graph G, what is the smallest upper bound you can give on the number of colors we will end up using if we run `GreedyColoring` using an arbitrary processing of the vertices, in terms of the maximum degree d_{max}?

Question 1.3. For a game of ultimate frisbee, a group of Disney villains are being put into two teams. However, each villain v_i has their own set of enemies $v_i.enemies$ who they do not like. The feelings are mutual. Hoping not to start a multiverse war, we wish to check if it is possible to place them into teams such that enemies are not on the same team. How can we check if there is a feasible team matching process?

1. Propose how you would represent the scenario as a graph. (1-2 sentences)
2. Explain briefly what might be some conditions that are necessary so that it would always be possible to construct two teams where no teammates are enemies. On a high level, explain how we can check if matching is possible.

Question 1.4. Show that the following are equivalent statements for a graph G:

1. G is bipartite.
2. G is 2-colorable.
3. G has no cycles of odd length.

Question 1.5. Suppose that there is an algorithm for $(k-1)$-coloring that runs in time $T(n,m) \geq n+m$. Prove that there is an algorithm for k-coloring that runs in time

$$O\left(\binom{n}{\leq (n/k)} \cdot T(n,m)\right).$$

This generalizes the SRE reduction from 3-coloring to 2-coloring to larger values of k. (Here $\binom{n}{\leq t}$ is the number of subsets of $[n]$ of size at most t, which can be bounded using the formula given in the SRE notes.)

2 Independent Sets

Recall the definition of an independent set:

Definition 2.1. Let $G = (V, E)$ be a graph. An *independent set* in G is a subset $S \subseteq V$ such that there are no edges entirely in S. That is, $\{u, v\} \in E$ implies that $u \notin S$ or $v \notin S$.

Input	**:** A graph $G = (V, E)$
Output	**:** An independent set $S \subseteq V$ in G of maximum size

Computational Problem Independent Set

Like with graph coloring, we can try a greedy algorithm for Independent Set:

```
1 GreedyIndSet($G$)
  Input  : A graph $G = (V, E)$
  Output : A "large" independent set in $G$
2 Choose an ordering $v_0, v_1, v_2, \ldots, v_{n-1}$ of $V$;
3 $S = \emptyset$;
4 foreach $i = 0$ to $n - 1$ do
5     if $\forall j < i$ s.t. $\{v_i, v_j\} \in E$ we have $v_j \notin S$ then $S = S \cup \{v_i\}$;
6 return $S$
```

And, similarly to coloring, we can only prove fairly weak bounds on the performance of the greedy algorithm in general:

Theorem 2.2. *For every graph G with n vertices and m edges,* `GreedyIndSet(`G`)` *can be implemented in time $O(n + m)$ and outputs an independent set of size at least $n/(d_{max} + 1)$, where d_{max} is the maximum vertex degree in G.*

Question 2.3. Prove that the `GreedyIndependentSet` algorithm finds an independent set of size at least $n/(d_{max} + 1)$.

One particularly useful application of the IndependentSet problem is in optimizing interval scheduling – how do we schedule as many non-conflicting commitments as possible in a given period, using our GreedyIndSet algorithm as a framework?

Question 2.4. (Concept Check) How do we represent the IntervalScheduling-Optimization problem in graphical form? What are the vertices, and what are the edges? How do we connect this to the IndependentSet problem?

With this graph-theoretic modelling, we can instantiate `GreedyIndSet()` for IntervalScheduling-Optimization:

```
1 GreedyIntervalScheduling($x$)
  Input  : A list $x$ of $n$ intervals $[a, b]$, with $a, b \in \mathbb{Q}$
  Output : A "large" subset of the input intervals that are disjoint from each other
2 Choose an ordering of the input intervals $[a_0, b_0], [a_1, b_1], \ldots, [a_{n-1}, b_{n-1}]$;
3 $S = \emptyset$;
4 foreach $i = 0$ to $n - 1$ do
5     if $\forall j < i$ s.t. $j \in S$ we have $[a_j, b_j] \cap [a_i, b_i] = \emptyset$ then $S = S \cup \{i\}$;
6 return $S$
```

Theorem 2.5. *If the input intervals are sorted by increasing order of end time b_i, then* `GreedyIntervalSchedulin` *will find an optimal solution to IntervalScheduling-Optimization, and can be implemented in time $O(n \log n)$.*

Question 2.6. (Concept Check) What happens if we sort the intervals by increasing order of start time instead? What about if we sort the intervals by decreasing start time?

Question 2.7. (Maximal Independent Sets.) In class, we have studied the problem of finding an independent set of *maximum size*. An easier problem is that of finding what is called a *maximal* independent set (note the difference in terminology: maximal vs. maximum). A *maximal independent* in a graph $G = (V, E)$ is an independent set $S \subseteq V$ such that there is no independent set T that strictly contains S ($S \subsetneq T$).

Note: You won't see Maximal independent sets outside of section (in CS 120), so don't get this definition confused with Maximum-Size independent sets!

1. Prove that a set is a maximal independent set if it is an independent set and for every vertex not in the set, at least one of its neighbors is in the set.

2. Prove that every maximum-size independent set is maximal but that the converse is not true (i.e. there is a graph G with a maximal independent set that is not of maximum size).

3. Show that `GreedyIndSet(`G`)` always outputs a maximal independent set in G.

4. Prove that every independent set S is contained in a maximal independent set.

5. Show that if a graph G is k-colorable, then there is a k-coloring of G in which one of the color classes is a maximal independent set.