GEMS OF TCS

GRAPH COLORING ALGORITHMS

Sasha Golovnev
September 19, 2022
PREVIOUSLY...

• Exact Algorithms

• Randomized Algorithms

• Approximate Algorithms
PREVIOUSLY...

- Exact Algorithms
- Randomized Algorithms
- Approximate Algorithms

Today: More examples
Map Coloring
THE LAND OF OZ

Map of the Countries of the Land of Oz

- Merryland
- Land of Ev
- Kingdom of Ik
- Deserts
- Gillikin Country
- Munchkin Country
- Emerald Country
- Winkie Country
- Scarecrow Country
- Country of the gnomes
- Happy Valley
- Balsam

The Land of Oz
SWISS CANTONS
Four Color Theorem

Theorem [Appel, Haken, 1976]

Every map can be colored with 4 colors.
Theorem [Appel, Haken, 1976]

Every map can be colored with 4 colors.

• Proved using a computer.
FOUR COLOR THEOREM

Theorem [Appel, Haken, 1976]

Every map can be colored with 4 colors.

- Proved using a computer.
- Computer checked almost 2000 graphs.
The Four Color Theorem

- Proved using a computer.
- Computer checked almost 2000 graphs.
- Robertson, Sanders, Seymour, and Thomas gave a much simpler proof in 1997 (still using a computer search).

**Theorem [Appel, Haken, 1976]**

Every map can be colored with 4 colors.

**Theorem [Weak Version]**

Every map can be colored with 6 colors.
<table>
<thead>
<tr>
<th><strong>Four Color Theorem</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Theorem [Appel, Haken, 1976]</strong></td>
</tr>
<tr>
<td>Every map can be colored with 4 colors.</td>
</tr>
<tr>
<td>• Proved using a computer.</td>
</tr>
<tr>
<td>• Computer checked almost 2000 graphs.</td>
</tr>
<tr>
<td>• Robertson, Sanders, Seymour, and Thomas gave a much simpler proof in 1997 (still using a computer search).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Theorem [Weak Version]</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Every map can be colored with 6 colors.</td>
</tr>
</tbody>
</table>
Theorem [Weak Version]

Every map can be colored with 6 colors.

- **Induction** on the number of countries $n$. 
Theorem [Weak Version]

Every map can be colored with 6 colors.

- **Induction** on the number of countries $n$.
- **Base case.** $n \leq 6$: can color with 6 colors.
Theorem [Weak Version]

Every map can be colored with 6 colors.

• **Induction** on the number of countries $n$.
• **Base case.** $n \leq 6$: can color with 6 colors.
• **Induction assumption.** All maps with $k$ countries can be colored with 6 colors.
**Six Color Theorem**

**Theorem [Weak Version]**

Every map can be colored with 6 colors.

- **Induction** on the number of countries $n$.
- **Base case.** $n \leq 6$: can color with 6 colors.
- **Induction assumption.** All maps with $k$ countries can be colored with 6 colors.
- **Induction step.** We’ll show that any map with $k + 1$ countries can be colored with 6 colors.
**Lemma**

Every map contains a country $v$ with at most 5 neighbors.
Lemma

Every map contains a country $v$ with at most 5 neighbors.
Lemma

Every map contains a country $v$ with at most 5 neighbors.
SIX COLOR THEOREM. PROOF

Lemma

Every map contains a country $v$ with at most 5 neighbors.
Graph Coloring
Graph Coloring

- A graph coloring is a coloring of the graph vertices s.t. no pair of adjacent vertices share the same color.
Graph Coloring

• A graph coloring is a coloring of the graph vertices s.t. no pair of adjacent vertices share the same color.

• The chromatic number $\chi(G)$ of a graph $G$ is the smallest number of colors needed to color the graph.
Chromatic Number

The chromatic number is 3.
Chromatic number is 3
CHROMATIC NUMBER

Chromatic number is 3
CHROMATIC NUMBER

Chromatic number is 3
Chromatic number is 3
The chromatic number of $K_n$ is $n$. 

-complete graph-
For $n > 1$, the chromatic number of $P_n$ is 2.
For even $n$, the chromatic number of $C_n$ is 2.
For odd $n > 2$, the chromatic number of $C_n$ is 3.
Bipartite Graphs

The chromatic number of a bipartite graph (with at least 1 edge) is 2.
Applications
EXAM SCHEDULE

• Each student takes an exam in each of her courses
• All students in one course take the exam together
• One student cannot take two exams per day
• What is the minimum number of days needed for the exams?
EXAM SCHEDULE

• Each student takes an exam in each of her courses
• All students in one course take the exam together
• One student cannot take two exams per day
• What is the minimum number of days needed for the exams?

Graphs
Proofs
Numbers
Combs
Project
EXAM SCHEDULE

- Each student takes an exam in each of her courses
- All students in one course take the exam together
- One student cannot take two exams per day
- What is the minimum number of days needed for the exams?
EXAM SCHEDULE

• Each student takes an exam in each of her courses
• All students in one course take the exam together
• One student cannot take two exams per day
• What is the minimum number of days needed for the exams?
Bandwidth allocation

Different stations are allowed to use the same frequency if they are far apart. What is an optimal assignment of frequencies to stations?
Bandwidth allocation

Different stations are allowed to use the same frequency if they are far apart. What is an optimal assignment of frequencies to stations?
Different stations are allowed to use the same frequency if they are far apart. What is an optimal assignment of frequencies to stations?
OTHER APPLICATIONS

• Scheduling Problems

• Register Allocation

• Sudoku puzzles

• Taxis scheduling

• ...
Exact Algorithm for Coloring
Dynamic Programming

- Given graph $G$ on $n$ vertices, find $\chi(G)$—minimum number of colors in a valid coloring of $G$
Dynamic Programming

• Given graph $G$ on $n$ vertices, find $\chi(G)$—minimum number of colors in a valid coloring of $G$

• Dynamic programming is one of the most powerful algorithmic techniques
Dynamic Programming

- Given graph $G$ on $n$ vertices, find $\chi(G)$—minimum number of colors in a valid coloring of $G$
- Dynamic programming is one of the most powerful algorithmic techniques
- Rough idea: express a solution for a problem through solutions for smaller subproblems
• For a subset of vertices $S \subseteq \{1, \ldots, n\}$ compute $\chi(S)$—the minimum number of colors needed to color vertices $S$. 
For a subset of vertices $S \subseteq \{1, \ldots, n\}$ compute $\chi(S)$ — the minimum number of colors needed to color vertices $S$.

Consider $S$. For any subset $U \subseteq S$, if there are no edges between vertices from $U$, we can color them all in one color, and use $\chi(S \setminus U)$ to color the rest.

$$\chi(S) = \min \chi(U)$$
For a subset of vertices $S \subseteq \{1, \ldots, n\}$ compute $\chi(S)$—the minimum number of colors needed to color vertices $S$.

Consider $S$. For any subset $U \subseteq S$, if there are no edges between vertices from $U$, we can color them all in one color, and use $\chi(S \setminus U)$ to color the rest.
For a subset of vertices $S \subseteq \{1, \ldots, n\}$ compute $\chi(S)$—the minimum number of colors needed to color vertices $S$

Consider $S$. For any subset $U \subseteq S$, if there are no edges between vertices from $U$, we can color them all in one color, and use $\chi(S \setminus U)$ to color the rest

$$\chi(S) = \min_{U \text{ without edges}} 1 + \chi(S \setminus U)$$
Order of Subproblems

- Need to process all subsets $S \subseteq \{1, \ldots, n\}$ in order that guarantees that when computing the value of $\chi(S)$, the values of $\chi(S \setminus U)$ have already been computed.
ORDER OF SUBPROBLEMS

• Need to process all subsets $S \subseteq \{1, \ldots, n\}$ in order that guarantees that when computing the value of $\chi(S)$, the values of $\chi(S \setminus U)$ have already been computed.

• For example, we can process subsets in order of increasing size.
Algorithm

\[
\chi(\emptyset) = 0
\]

for \( s \) from 1 to \( n \):

for all \( S \subseteq \{1, \ldots, n\} \) of size \( s \):

for all \( U \subseteq S \), \( U \) without edges

\[
\chi(S) \leftarrow \min\{\chi(S), \chi(S \setminus U) + 1\}
\]

return \( \chi(\{1, \ldots, n\}) \)
χ(∅) = 0

for s from 1 to n:
    for all S ⊆ {1, . . . , n} of size s:
ALGORITHM

\[
\chi(\emptyset) = 0
\]

for s from 1 to n:

for all \( S \subseteq \{1, \ldots, n\} \) of size s:

for all \( U \subseteq S, \ U \) without edges

\[
\chi(S) \leftarrow \min\{\chi(S), \chi(S \setminus U) + 1\}
\]

return \( \chi(\{1, \ldots, n\}) \)
Algorithm

\[ \chi(\emptyset) = 0 \]

for \( s \) from 1 to \( n \):

\[ \text{for all } S \subseteq \{1, \ldots, n\} \text{ of size } s: \]

\[ \text{for all } U \subseteq S, \ U \text{ without edges} \]

\[ \chi(S) \leftarrow \min\{\chi(S), \chi(S \setminus U) + 1\} \]

return \( \chi(\{1, \ldots, n\}) \)
$\chi(\emptyset) = 0$

FOR $s$ FROM 1 TO $n$:

FOR ALL $S \subseteq \{1, \ldots, n\}$ OF SIZE $s$:

FOR ALL $U \subseteq S$, $U$ WITHOUT EDGES

$\chi(S) \leftarrow \min\{\chi(S), \chi(S \setminus U) + 1\}$

RETURN $\chi(\{1, \ldots, n\})$
Randomized Algorithm for 3-Coloring
RANDOMIZED ALGORITHM

- Given a 3-colorable graph, find a 3-coloring
Randomized Algorithm

• Given a 3-colorable graph, find a 3-coloring

• This problem is $\textbf{NP}$-hard, we’ll give an exponential-time algorithm
RANDOMIZED ALGORITHM

• Forbid one random color at each vertex

Solve 2-SAT in polynomial time

Repeat the algorithm \(\left(\frac{3}{2}\right)n\) times
RANDOMIZED ALGORITHM

- Forbid one random color at each vertex
- Solve 2-SAT in polynomial time
- Repeat the algorithm \((\frac{3}{2})^n\) times
Randomized Algorithm

- Forbid one random color at each vertex
- Solve 2-SAT in polynomial time
RANDOMIZED ALGORITHM

• Forbid one random color at each vertex

• Solve 2-SAT in polynomial time

• Repeat the algorithm \((3/2)^n\) times
Approximate Algorithm for 3-Coloring
Approximate Coloring

- Given a 3-colorable graph, finding a 3-coloring is \textbf{NP-hard}
Approximate Coloring

• Given a 3-colorable graph, finding a 3-coloring is \textbf{NP-hard}

• Given a 3-colorable graph, finding an $n$-coloring is \textbf{trivial}
Approximate Coloring

• Given a 3-colorable graph, finding a 3-coloring is \textbf{NP-hard}

• Given a 3-colorable graph, finding an $n$-coloring is \textit{trivial}

• We’ll see how to find an $O(\sqrt{n})$-coloring in polynomial time
<table>
<thead>
<tr>
<th><strong>Greedy Coloring</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>A graph $G$ where each vertex has degree $\leq \Delta$ can be colored with $\Delta + 1$ colors.</td>
</tr>
</tbody>
</table>
Greedy Coloring

A graph $G$ where each vertex has degree $\leq \Delta$ can be colored with $\Delta + 1$ colors.
A graph $G$ where each vertex has degree $\leq \Delta$ can be colored with $\Delta + 1$ colors.
APPROXIMATE ALGORITHM

While there is vertex $v \in G$ of degree $\geq \sqrt{n}$:

Color the neighbors of $v$ in 2 new colors, remove them from the graph.

All remaining vertices have degree $< \sqrt{n}$. Color the rest of the graph using $\sqrt{n}$ new colors.
While there is vertex \( v \in G \) of degree \( \geq \sqrt{n} \):
APPROXIMATE ALGORITHM

While there is vertex $v \in G$ of degree $\geq \sqrt{n}$:

- Color the neighbors of $v$ in 2 new colors,
- remove them from the graph.

All remaining vertices have degree $< \sqrt{n}$. Color the rest of the graph using $\sqrt{n}$ new colors.
While there is vertex $v \in G$ of degree $\geq \sqrt{n}$:

Color the neighbors of $v$ in 2 new colors, remove them from the graph.
**APPROXIMATE ALGORITHM**

While there is vertex $v \in G$ of degree $\geq \sqrt{n}$:

- Color the neighbors of $v$ in 2 new colors, remove them from the graph

All remaining vertices have degree $< \sqrt{n}$. Color the rest of the graph using $\sqrt{n}$ new colors
ANALYSIS