Discrete Optimization
Lecture-15

Ngày 29 tháng 11 năm 2011

Vertex and edge coloring and their computational complexity

Recall:

- $Delta(G) = \max deg(v) \mid v \in V(G)$.

Vertex and edge coloring and their computational complexity

Recall:

- $Delta(G) = \max deg(v) \mid v \in V(G)$.
- $\chi(G) =$ the chromatic number of G.

Vertex and edge coloring and their computational complexity

Recall:

- $Delta(G) = \max deg(v) \mid v \in V(G)$.
- $\chi(G) =$ the chromatic number of G.
- $\chi_1(G) =$ the chromatic index of G (edge colors).

Vertex and edge coloring and their computational complexity

Recall:

- $Delta(G) = \max deg(v) \mid v \in V(G)$.
- $\chi(G) =$ the chromatic number of G.
- $\chi_1(G) =$ the chromatic index of G (edge colors).

Vertex and edge coloring and their computational complexity

Recall:

- $Delta(G) = \max deg(v) \mid v \in V(G)$.
- $\chi(G) =$ the chromatic number of G.
- $\chi_1(G) =$ the chromatic index of G (edge colors).

We proved:

Theorem: Deciding whether a graph G is bipartite ($2-$ vertex-colorable) can be done in polynomial time.

Vertex and edge coloring and their computational complexity

Recall:

- $Delta(G) = \max deg(v) \mid v \in V(G)$.
- $\chi(G) =$ the chromatic number of G.
- $\chi_1(G) =$ the chromatic index of G (edge colors).

We proved:

Theorem: Deciding whether a graph G is bipartite ($2-$ vertex-colorable) can be done in polynomial time.

Theorem: Deciding whether a given graph G is $3-$ vertex colorable is in the class **NP-Complete**

Vertex and edge coloring and their computational complexity

Recall:

- $Delta(G) = \max deg(v) \mid v \in V(G)$.
- $\chi(G) =$ the chromatic number of G.
- $\chi_1(G) =$ the chromatic index of G (edge colors).

We proved:

Theorem: Deciding whether a graph G is bipartite ($2-$ vertex-colorable) can be done in polynomial time.

Theorem: Deciding whether a given graph G is $3-$ vertex colorable is in the class **NP-Complete**

König: If G is bipartite then $\chi_1(G) = \Delta(G)$.

Theorem (Brooks Theorem)

If a connected graph $G \neq K_{n+1}$ and $\Delta(G) = n > 2$ then $\gamma(G) \leq \Delta(G)$.

Theorem (Brooks Theorem)

If a connected graph $G \neq K_{n+1}$ and $\Delta(G) = n > 2$ then $\gamma(G) \leq \Delta(G)$.

Chứng minh.

Watch the blackboard. □

Theorem (Brooks Theorem)

If a connected graph $G \neq K_{n+1}$ and $\Delta(G) = n > 2$ then $\gamma(G) \leq \Delta(G)$.

Chứng minh.

Watch the blackboard. □

Remark

Brooks' theorem gives us an effcient algorithm that will guarantee a coloring by no more than $\Delta(G)$ colors. This can be quite far from the chromatic number. It is not difficult to construct a sequencing for a bipartite graph for which a particular sequence-coloring will yield a coloring by close to $\Delta(G)$ colors.

Theorem (Brooks Theorem)

If a connected graph $G \neq K_{n+1}$ and $\Delta(G) = n > 2$ then $\gamma(G) \leq \Delta(G)$.

Chứng minh.

Watch the blackboard. □

Remark

Brooks' theorem gives us an effcient algorithm that will guarantee a coloring by no more than $\Delta(G)$ colors. This can be quite far from the chromatic number. It is not difficult to construct a sequencing for a bipartite graph for which a particular sequence-coloring will yield a coloring by close to $\Delta(G)$ colors.

Theorem (Brooks Theorem)

If a connected graph $G \neq K_{n+1}$ and $\Delta(G) = n > 2$ then $\gamma(G) \leq \Delta(G)$.

Chứng minh.

Watch the blackboard. □

Remark

Brooks' theorem gives us an effcient algorithm that will guarantee a coloring by no more than $\Delta(G)$ colors. This can be quite far from the chromatic number. It is not difficult to construct a sequencing for a bipartite graph for which a particular sequence-coloring will yield a coloring by close to $\Delta(G)$ colors.

Surprisingly, for edge coloring, which is NP-Complete, there is an effcicient algorithm that will color the edges of a simple graph G by no more than $\Delta(G) + 1$ colors. That is by using no more than one extra color than the absolute necessary number of colors.

Vizing's Theorem

Theorem

If G is a simple graph then $\chi_1(G) \leq \Delta(G) + 1$

Vizing's Theorem

Theorem

If G is a simple graph then $\chi_1(G) \leq \Delta(G) + 1$

Remark

The odd cycles C_{2k+1}, Petersen's graph are examples of graphs for which $\chi_1(G) = \Delta(G) + 1$.

Vizing's Theorem

Theorem

If G is a simple graph then $\chi_1(G) \leq \Delta(G) + 1$

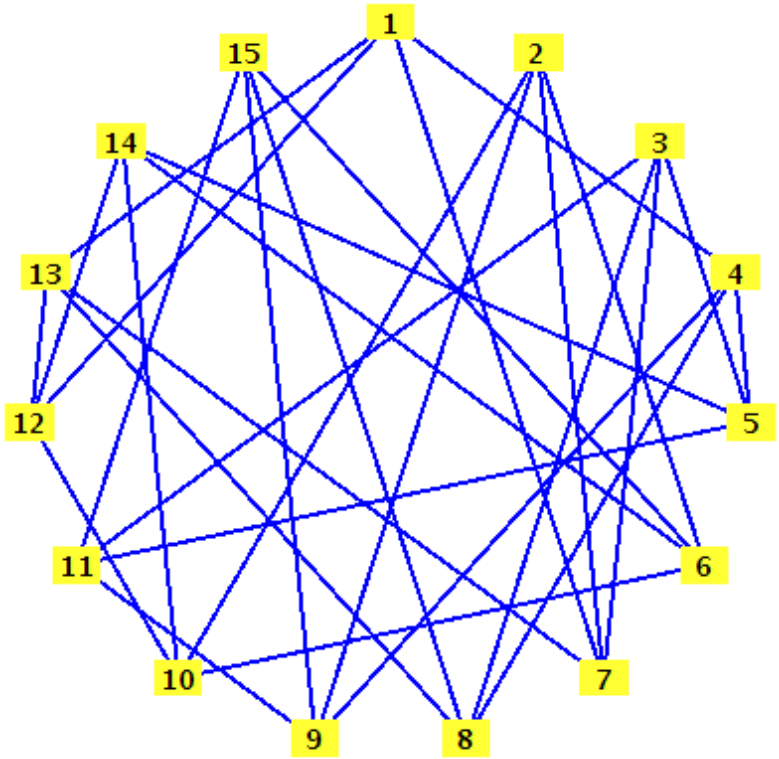
Remark

The odd cycles C_{2k+1}, Petersen's graph are examples of graphs for which $\chi_1(G) = \Delta(G) + 1$.

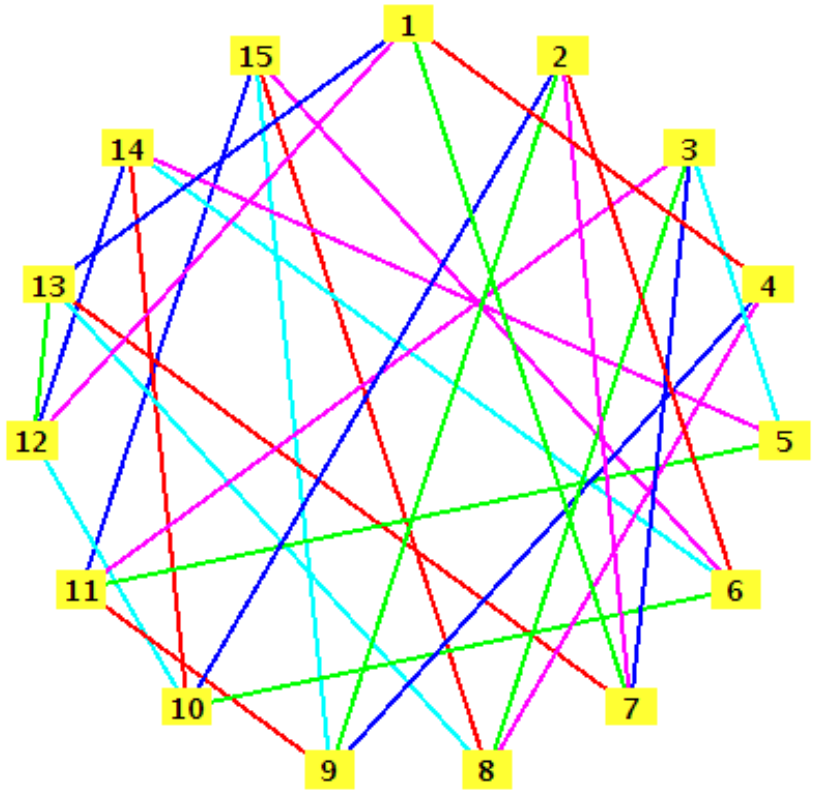
Discussion

To understand the proof, we shall start with an example. Consider the following graph of order 15 regular of degree 4. We shall try to color the edges using 5 colors.

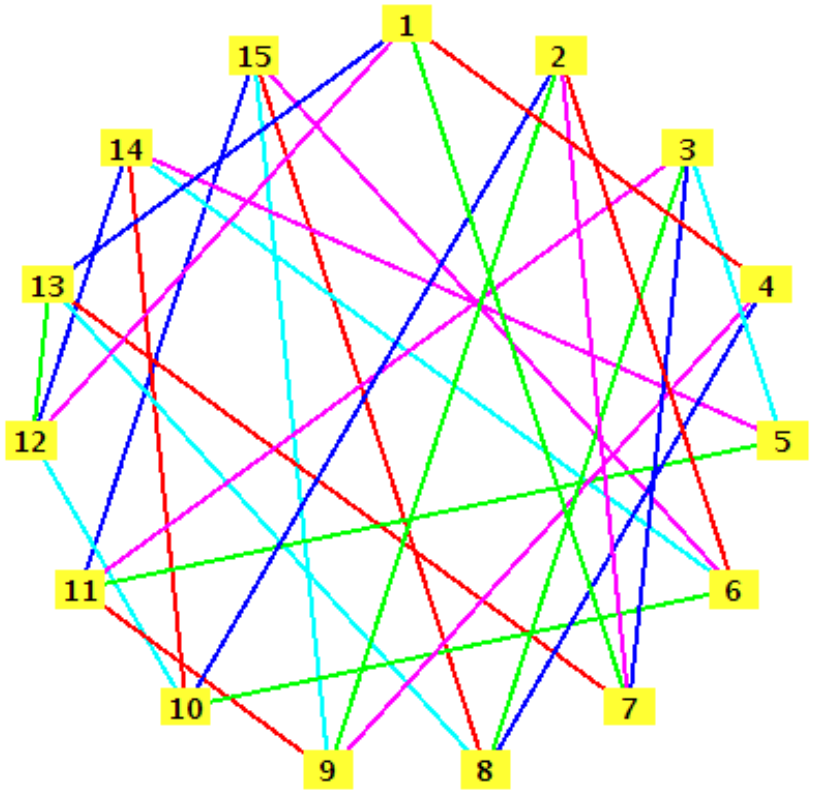
My program had difficulties coloring the edges of G in five colors.



But it had no difficulties coloring the edges of $G - (4, 5)$ in five colors.



Changing the colors prior to building the path.



Computational complexity

The core issues of Discrete Optimization is efficient computations. Most of the problems we encountered are conceptually simple. There is a finite number of feasible solutions. All we have to do is generate all of them and choose the optimal. As we studied a variety of problems, we realized that this approach is almost never feasible. So we needed to find some more clever ways to do it.

Computational complexity

The core issues of Discrete Optimization is efficient computations. Most of the problems we encountered are conceptually simple. There is a finite number of feasible solutions. All we have to do is generate all of them and choose the optimal. As we studied a variety of problems, we realized that this approach is almost never feasible. So we needed to find some more clever ways to do it.

The key question is feasibility, that is: how long will it take to solve problems. The accepted approach to measure feasibility of an algorithmi s by the number of “steps” it takes to solve an instance as a function of its “size”.

Computational complexity

The core issues of Discrete Optimization is efficient computations. Most of the problems we encountered are conceptually simple. There is a finite number of feasible solutions. All we have to do is generate all of them and choose the optimal. As we studied a variety of problems, we realized that this approach is almost never feasible. So we needed to find some more clever ways to do it.

The key question is feasibility, that is: how long will it take to solve problems. The accepted approach to measure feasibility of an algorithmi s by the number of “steps” it takes to solve an instance as a function of its “size”.

Convetional wisdom is that if the number of steps is bounded by $p(n)$ a plynomial of a fixed degree k then the algorithm is feasible.

Computational Complexity

A decision problem is a problem for which the answer is **YES** or **NO** (true or false, 1 or 0 etc.).

Example

Computational Complexity

A decision problem is a problem for which the answer is **YES** or **NO** (true or false, 1 or 0 etc.).

Example

- *Is G 3− colorable?*

Computational Complexity

A decision problem is a problem for which the answer is **YES** or **NO** (true or false, 1 or 0 etc.).

Example

- *Is G 3− colorable?*
- *Is n a compound integer?*

Computational Complexity

A decision problem is a problem for which the answer is **YES** or **NO** (true or false, 1 or 0 etc.).

Example

- *Is G 3− colorable?*
- *Is n a compound integer?*
- *Is p a prime number?*

Computational Complexity

A decision problem is a problem for which the answer is **YES** or **NO** (true or false, 1 or 0 etc.).

Example

- *Is G 3− colorable?*
- *Is n a compound integer?*
- *Is p a prime number?*
- *Does the given assignment have an assignment whose cost is* $\leq 10,000,000,000$ *VND?*

Computational Complexity

A decision problem is a problem for which the answer is **YES** or **NO** (true or false, 1 or 0 etc.).

Example

- *Is G 3− colorable?*
- *Is n a compound integer?*
- *Is p a prime number?*
- *Does the given assignment have an assignment whose cost is* $\leq 10,000,000,000$ *VND?*
- *Does the given netwrok have a cut of size* $\leq m$*?*

Computational Complexity

A decision problem is a problem for which the answer is **YES** or **NO** (true or false, 1 or 0 etc.).

Example

- *Is G 3− colorable?*
- *Is n a compound integer?*
- *Is p a prime number?*
- *Does the given assignment have an assignment whose cost is* $\leq 10,000,000,000$ *VND?*
- *Does the given netwrok have a cut of size* $\leq m$*?*
- *Does G have a vertex cover of size k?*