An Introduction to Graph Homomorphisms

Davis Shurbert
Department of Mathematics and Computer Science
University of Puget Sound

May 1, 2013

Abstract

This paper aims to give a focused introduction to algebraic graph theory accessible to mathematically mature undergraduates. We will begin by giving some standard definitions, then expanding our focus to specifically study properties of graph homomorphisms in the context of constraint satisfaction problems (graph coloring in disguise). In particular, we will be discussing the idea of graph coloring by abstracting what it means for a graph to be colorable, and expanding this abstraction to deal with more complex instances of the graph coloring problem.

Introduction

Since we are going to be working in the field of graph theory, it is necessary to define precisely what we mean by a graph (for the advanced reader, we will only be working with simple graphs keeping definitions to a minimum, however the ideas presented will translate easily into more complex definitions for graphs). A *graph* G consists of a set of *vertices* $V(G)$ and a set of *edges* $E(G)$ represented by unordered pairs of vertices. It is important to note that a graph may have many different geometric representations, but we just use these as visualization tools and focus on $V(G)$ and $E(G)$ for our analysis. We define vertices x and y to be *adjacent* if $(x, y) \in E(G)$, for our purposes we will write $x \sim y$ to ease the notation but let the reader note that this is not an equivalence relation. It comes as no surprise to an algebraist that graphs have subgraphs, we say Y is a *subgraph* of X if $V(Y) \subseteq V(X)$ and $E(Y) \subseteq E(X)$. A graph is *complete* if every vertex is connected to every other vertex, and we denote the complete graph on n vertices by K_n. Finally, we define a graph with no edges to be an *independent* set of vertices. [1]

Now, we will continue our definitions by applying the notions of isomorphisms and homomorphisms to graphs. An *isomorphism* between two graphs G and H is a bijective mapping $f : G \mapsto H$ with the property

$$(x, y) \in E(G) \Leftrightarrow (f(x), f(y)) \in E(H).[1]$$

This simple definition comes as no surprise to those who have worked with isomorphisms in other contexts. We say that a graph isomorphism respects edges, just as group, field, and vector space isomorphisms respect the operations of these structures. We now consider a weakening of this definition to arrive at graph homomorphisms. A *homomorphism* from a graph G to a graph H is defined as a mapping (not necessarily bijective) $h : G \mapsto H$ such

that

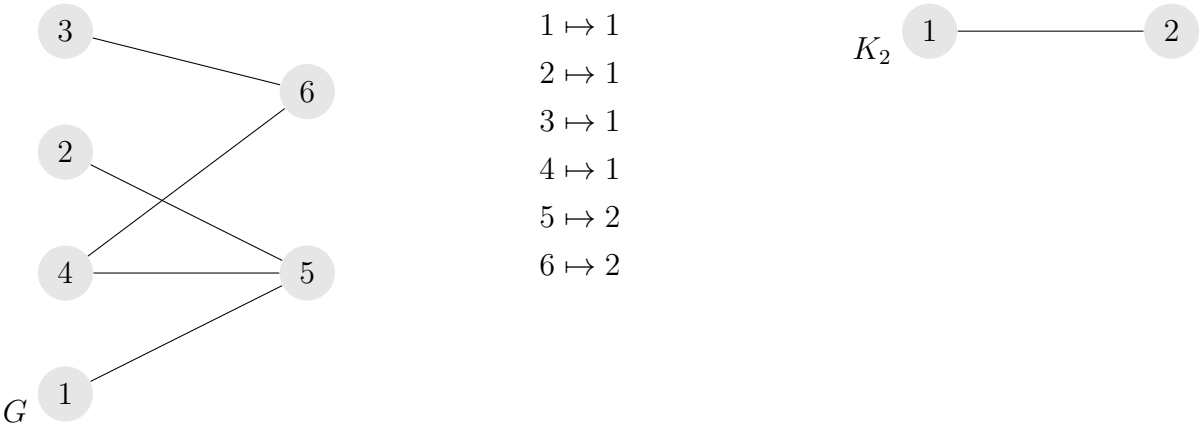
$$(x, y) \in E(G) \Rightarrow (h(x), h(y)) \in E(H).[1]$$

We say that a graph homomorphism preserves edges, and we will use this definition to guide our further exploration into graph theory and the abstraction of graph coloring.

Example. Consider any graph G with 2 independent vertex sets V_1 and V_2 that partition $V(G)$ (a graph with such a partition is called *bipartite*). Let $V(K_2) = \{1, 2\}$, the map $f : G \mapsto K_2$ defined as

$$f(v) = \begin{cases} 1, & \text{if } v \in V_1 \\ 2, & \text{if } v \in V_2 \end{cases}$$

is a graph homomorphism because no two adjacent vertices in G map to the same vertex in K_2, and thus our edges are preserved [1]. A specific homomorphism to K_2 of this form is illustrated by



where the G is the domain and K_2 is the codomain of the homomorphism. Notice that both of the above graphs can be properly colored with two colors, as we will be returning to this idea momentarily.

Graph Coloring

One of the more famous results in graph theory is the 4 color map theorem which states that any conceivable map can be colored with 4 colors such that no two regions sharing a border have the same color, but how could we go about proving such a thing? This next section is devoted to giving a precise definition to a coloring of a graph, as well as giving a precise mathematical meaning to the phrase "this graph is n-colorable." As always, we begin with

a definition. A *proper coloring* of a graph G is an assignment of colors to $V(G)$ such that no two adjacent vertices have the same color. A graph's *chromatic number* is the minimum number of colors needed to properly color the graph. We say that a graph is n-colorable if it can be properly colored with n colors. Here we arrive at our first theorem concerning graph homomorphisms. [1]

Theorem 1. A graph G is r-colorable $\Leftrightarrow$ there exists a homomorphism from G to K_r [1].

Proof. ($\Rightarrow$)
Assume G can be properly colored with r colors labeled $\{1, 2, \ldots, r\}$. We label a generic vertex v of G by v_i if it is colored by i where $1 \leq i \leq r$. Define a mapping $h : G \mapsto K_r$ by $h(v_i) = k_i$ where k_i is the i^{th} vertex in K_r (without loss of generality we can impose an ordering on the vertices of K_r). We claim that this map is a graph homomorphism:

Let $a, b \in V(G)$ and assume $a \sim b$ (a is adjacent to b)
$\Rightarrow a$ and b do not have the same color so $a = a_i$ and $b = b_j$ where $i \neq j$.
$\Rightarrow h(a) = h(a_i) = k_i \neq k_j = h(b_j) = h(b)$ so $h(a) \neq h(b)$
$\Rightarrow h(a) \sim h(b)$ (the image of h is a complete graph)
Therefore, $a \sim b \Rightarrow h(a) \sim h(b)$ showing that h is a homomorphism from G to K_r.

($\Leftarrow$)
Let $h : G \mapsto K_r$ be a graph homomorphism. For a given $y \in V(K_r)$ define the set $h^{-1}(y) \subseteq V(G)$ to be

$$h^{-1}(y) = \{x \in V(G) \mid h(x) = y\}.$$

It is important to remember that these sets might be empty. Let $a, b \in h^{-1}(y)$,
$\Rightarrow h(a) = h(b) = y \qquad (a, b \in h^{-1}(y))$
$\Rightarrow h(a) \not\sim h(b) \qquad$ (a vertex cannot be adjacent to itself)
$\Rightarrow a \not\sim b \qquad$ (contrapositive of $a \sim b \Rightarrow h(a) \sim h(b)$).

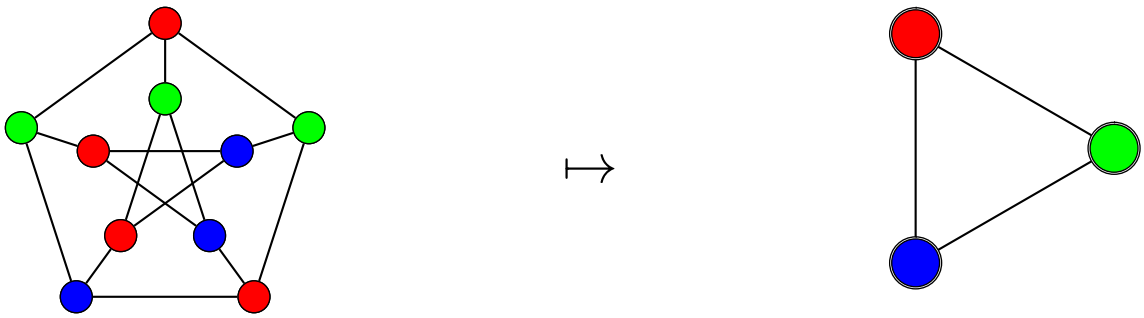
So, for every $y \in V(K_r)$ the corresponding $h^{-1}(y)$ is an independent subset of $V(G)$. Therefore, every element of $h^{-1}(y)$ can be colored using the same color. There are r vertices of K_r, and the sets $h^{-1}(y)$ form a partition of the vertices of G (partition comes from properties of a map, we can't assign one input to two different outputs and every input must map to some element of the codomain), so there are at most r colors needed to properly color G. [1] □

This theorem gives us a concrete mathematical interpretation of coloring a graph, but the above result answers much more than the simple question of whether or not a graph is n-colorable. As a simple consequence of this theorem, the chromatic number of a graph G is the smallest integer r such that there exists a homomorphism from G to K_r. In addition to this small logical step, a number of facts become apparent from the following corollary.

Corollary. A coloring of a graph G is **precisely** a homomorphism from G to some complete graph.

If we view a homomorphism $h : G \mapsto K_r$ for some graph G as an assignment of colors to the vertices of G, then h directly tells us *how* to create this coloring. For any $a \in V(G)$, if $h(a) = k_i$ then we simply assign color i from a set of r colors to vertex a. This process gives us a unique coloring of G for every h. If we want to know how many proper colorings of G we can create with r colors, we simply count the number of distinct homomorphisms from G to K_r.

Example. Consider the following coloring of the Petersen graph P represented as a homomorphism from $P \mapsto K_3$ where vertices of P map to their corresponding color in K_3. Given our previous discussion, it should be fairly obvious that this map is indeed a graph homomorphism.



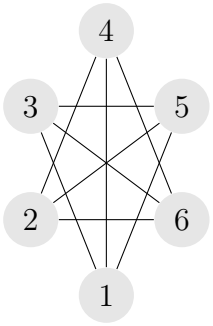
At this point, it is natural to wonder why graph coloring is important enough to warrant such a careful study. One would hope that there are more applications of this process than simply assigning colors to vertices, and indeed there are. Consider the problem of scheduling final exams such that no student is scheduled to take two exams during the same period. We can construct a graph G which has classes as its vertices, where two vertices are adjacent if there is a student taking both classes. We now have a graphical representation G of the current state our classes, and we wish to assign final periods to the vertices of G such that no two adjacent vertices have the same period assignment (otherwise this would constitute a scheduling conflict). Finding such a scheduling of our classes is equivalent to finding a proper coloring of G, where each color represents a different exam period. [2]

There are seemingly infinite variations of the above problem. We can assign frequencies to radio stations without causing interference arising from two geographically close radio stations having the same frequency. We can help an airline company assign pilots to their flights where each pilot is a color, each flight is a vertex, and each edge represents two flights that cannot be piloted by the same person. We can make as many examples of these types

of problems as we want by simply inventing probable scenarios of this form. You might be tempted to say that these scenarios are all the same problem worded differently, but instead think of them as a family of problems that can all be solved using the same technique.

Our family of problems solvable by graph coloring is indeed expansive, but it feels quite constrained because all of our solutions only consist of assigning n colors to a given graph. Let us revisit our exam scheduling example by adding constraints to our optimal schedule. If we want to ensure that no student is scheduled to take 2 exams in consecutive periods, we must change the desired homomorphic image from a complete graph to a more restrictive graphical representation of our optimal schedule. The rest of this section will discuss homomorphisms in the context of this problem, as to make comprehension of the material easier. [2]

Example. If we have 6 final periods with exams already scheduled such that no student must take consecutive exams, we can model this schedule as

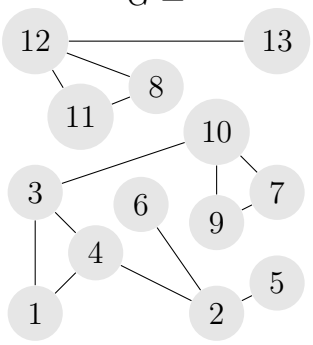


where each vertex is a final exam period and there exists an edge between vertices if a student can be scheduled to to take an exam in both time slots.

Once we have found a graphical representation of a schedule satisfying the constraints of our problem, we must ask if such a scheduling is possible, and if so how can we achieve it. To answer these questions we can again construct a graphical representation G of our current classes (classes are vertices of G while edges represent classes with mutual students) and find a homomorphism from G to our desired schedule represented by a graph H.

Example. Consider the following schedule assignment as a graph homomorphism $h : G \mapsto H$ where

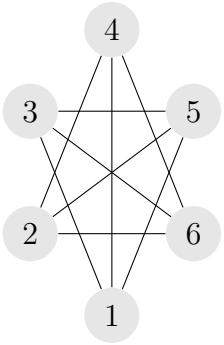
$G =$



h :

$1 \mapsto 1$
$2 \mapsto 2$
$3 \mapsto 3$
$4 \mapsto 5$
$5 \mapsto 6$
$6 \mapsto 5$
$7 \mapsto 2$
$8 \mapsto 1$
$9 \mapsto 3$
$10 \mapsto 6$
$11 \mapsto 5$
$12 \mapsto 3$
$13 \mapsto 6$

$H =$



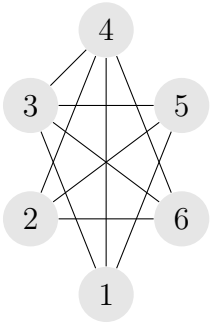
This homomorphism creates a scheduling that satisfies our constraints because two adjacent vertices in G (i.e. two classes that share at least one student) cannot be mapped to vertices of the form $(i, i + 1 \bmod 6)$ due to the definition of graph homomorphisms. This is equivalent to saying that no two classes that share at least one student can have their final exams scheduled in consecutive time slots. Our original requirement that no two classes that share students can be scheduled at the same time is still satisfied, because adjacent vertices in G must be mapped to adjacent in H and no vertex is adjacent to itself.

Notice how period 4 is never used, showing that that H does not represent an optimal schedule of final exam periods. With these specific constraints on our desired schedule, we must find the smallest n for which there exists a homomorphism from G to H_n. We define H_n to be the subgraph of K_n with edges $(i, i+1 \bmod n)$ removed. It is useful to think of this definition of H_n as if we have labeled the vertices of K_n as $\{1, 2, \ldots, n\}$ in increasing order around the perimeter of K_n. However this interpretation is not necessary as all labelings of K_n are isomorphic.

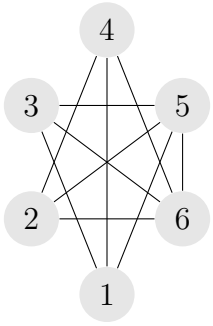
With the tools we have built so far, it now becomes easy to modify our constraints on our optimal schedule. We can add and remove edges and vertices to our graph H (the graphical representation of our desired schedule) depending on our new constraints.

Example. If our desired schedule has 6 periods, again with the constraint that no student is scheduled to take two exams consecutively, we can modify this to include a lunch break (so exams can be schedule on either side of this break without conflict), or a dinner break, or both.

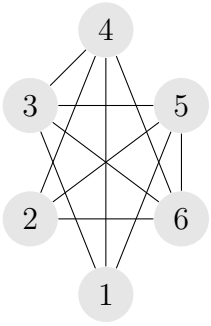
Lunch break
(between periods 3 and 4)



Dinner break
(between periods 5 and 6)



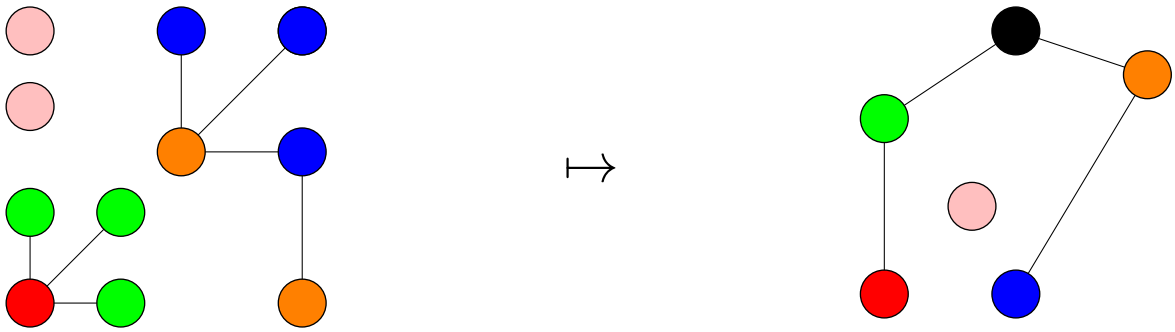
Both



To drive home the point, we arrive at our final example.

Example. Consider a group of displaced peoples whose entire country is now uninhabitable. This country was often at civil war, with many different factions forming a complex network of allies and enemies. We construct a graph G with the towns of this country as vertices, and add edges between enemy towns. Now construct a graph H representing the open land of the country which plans to adopt the refugees. Let the vertices of H be habitable regions of the new county, all having enough resources to share with friendly neighbors. Place edges between regions that have some sort of barrier between them. How can we assign groups of refugees to the new land such that enemy factions have barriers between each other's homes, minimizing potential violence?

A homomorphism from G to H acts as such an assignment. Enemies in G (adjacent vertices) cannot be moved to the same location in H, or to two locations that are easily traveled between. If two groups of refugees are enemies in G then they must be placed with barriers between them in H. In the language of our graphs, our proposed map must preserve edges and is thus a homomorphism. A simple instance of this problem is modeled by



where each vertex in G maps to its corresponding vertex in H. Notice than this is not an optimal arrangement of our refugees because the black vertex in H is never used and we can

find a homomorphism from G to smaller subgraphs of H (namely, the subgraph consisting of vertices {green, red, black} and edges {(green, red), (green, black)}).

The above example should convince you that our desired homomorphic image can take on almost any form, depending on the complexity of the problem we are trying to solve. We already know that the our original state G can be just as complex. From this observation, it becomes natural to ask if there exists a homomorphism $h : G \mapsto H$ given graphs G and H. If we can answer this decision problem (whether h exists or not), we might want to determine an explicit h and count how many distinct homomorphisms are possible. Here we will end our discussion of specific problems and arrive at these questions lying at the heart of this area of mathematical research.

Homomorphisms

So, how can we find $h : G \mapsto H$ given arbitrary graphs G and H as inputs? We will not be explicitly answering this question as it is still an open area of research, and any discussion of an exact algorithm for this problem would be beyond the scope of this paper. Instead, we will now be discussing the mathematics of graph homomorphisms in order to further understand the complexity of this problem. As a result, we will discover specific instances of G and H in which we can easily determine that there is no homomorphism from G to H. However, we must first present a few more definitions.

To ease notation, we write $G \to H$ if there exists a graph homomorphism from G to H. A *clique* of a graph G is a subgraph of G which is complete. We say that a graph has *clique number* n, denoted $\omega(G)$, if the largest clique of G has n vertices. Just as $\chi(G)$ is the smallest r such that $G \to K_r$, $\omega(G)$ is the smallest n such that $K_n \to G$. [2] We state this as a theorem for consistency.

Theorem 2. For any graph G, $\omega(G) = r \Leftrightarrow$ r is the largest integer such that $K_r \to G$. [2]

Proof. If G has a clique of size n labeled C, then any isomorphism from K_n to C induces a homomorphism from K_n to G (the fact that C is isomorphic to K_n should be obvious, considering they are both complete graphs on n vertices). Conversely, if $K_n \to G$ then each vertex of K_n must have a unique image in G, where all of these images must be connected in order to preserve the edges of K_n. These images must then form a clique of size n in G. Given this necessary and sufficient condition, it directly follows that $\omega(G) = r \Leftrightarrow K_r \to G$ where r is maximal. □

We have already defined the chromatic number of a graph G, denoted $\chi(G)$, as the minimum number of colors needed to properly color G. The following theorem describes a

useful relationship between both $\chi(G)$ and $\chi(H)$, and $\omega(G)$ and $\omega(H)$ respectively, where H is the codomain of a homomorphism from G.

Theorem 3. If $G \to H$ then $\omega(G) \leq \omega(H)$ and $\chi(G) \leq \chi(H)$. [2]

Proof. The compositions of homomorphisms is again a homomorphism, i.e. $G \to H \to K$ implies $G \to K$. Given this fact,
$K_{\omega(G)} \to G \to H$ implies $K_{\omega(G)} \to H$, so $\omega(H)$ is at least $\omega(G)$, and
$G \to H \to K_{\chi(H)}$ implies $G \to K_{\chi(H)}$, so $\chi(G)$ is at most $\chi(H)$. [2] □

Even with these simple results, we are beginning to make significant process in determining if, given graphs G and H, there is no homomorphism from G to H. We can compute $\omega(G)$ and $\chi(G)$ for any graph G, although these computations might be difficult. Comparing our results for G and H we can immediately tell if G does not map homomorphically to H.

We will finish our discussion of graph homomorphisms with a brief introduction to the concept of homomorphism classes. If $G \leftarrow H$ and $H \leftarrow G$ we say that $G \equiv H$ [2]. As one might suspect, '$\equiv$' produces an equivalence relation. The relation is symmetric because the identity map from G to G is an isomorphism and thus a homomorphism, transitive because composition of homomorphisms is again a homomorphism, and reflexive by nature of our definition. This equivalence relation forms equivalence classes of graphs, which we can use to solve our decision problem. [2]

Since we are now dealing with equivalence classes, it is natural to ask if there are canonical representatives of these classes. We define a *core* of a graph G to be the element of $[G]$ (homomorphism equivalence class of G) with the least number of vertices [2]. Given this definition, we get some interesting results involving these cores.

1) G is a core $\Leftrightarrow$ every endomorphism of G is an automorphism. [2]

2) Any graph (or homomorphism equivalence class) has a unique core unto isomorphism. [2]

These results are certainly nontrivial, but we will not discuss the proofs here. [2] gives sufficient proofs of these facts and a discussion of the topic of cores and homomorphism equivalence classes in the context of classifying graphs, but these results are not crucial to our discussion. Instead, let us examine why the above results might prove useful.

Since the number of vertices of the core of a graph G must be less than or equal to $|V(G)|$, we assume it is more computationaly simple to find homomorphism from this core to a desired H than using our original G. If we denote the core of G by $C(H)$, it should be clear that $G \to H$ if and only if $C(G) \to H$. Therefore, we can refine our search for $G \to H$

for all graphs homomorphically equivalent to G. This is useful because cores have specific properties that they do not share with generic graphs (fact 1 from above for example).

These results do not explicitly help us solve the problem of determining whether $G \to H$ because we are still faced with the problem of finding a core. This new problem may be just as complex as our original, and if we solve it we still have not fully answered our question. If we could easily find a core, we could then determine if $H \equiv G$ (cores of homomorphism classes are unique unto isomorphism) but even then we would be faced with an isomorphism problem, which is again nontrivial. However, deciding whether or not $G \to H$ is true may still be more challenging than deciding if $C(G) \to C(H)$ is true, and the preceding are equivalent. This ends our discussion of graph homomorphisms and their applications. However, the reader is encouraged to pursue further reading on the subject through some suggested texts bellow.

Further Reading

Readers who are new two graph theory are encouraged to pick up [1], as I found it an incredibly helpful and enlightening read focusing on algebraic graph theory in general. Advanced readers (or those specifically interested in graph homomorphisms) should consider reading *Graphs and Homomorphisms (Oxford Lecture Series in Mathematics and Its Applications)*, written by Pavol Hell and Jaroslav Nesetril. The authors are well known for their contributions to this area of research, and the text is written at the master's level. Note that this is a fairly new area of research, only starting to become widely accepted during the past decade or so. Here is a list of *open problems* as of 2008 on the subject of graph homomorphisms.

References

[1] Chris Godsil, Gordon Royle, *Algebraic Graph Theory*. Graduate Texts in Mathematics, 2001. ISBN: 0 387 95241 1

[2] Peter J. Cameron *Graph homomorphisms* Combinatorics Study Group Notes, September 2006 , Paper (*Link*)

[3] Norman Biggs, *Algebraic Graph Theory*. Cambridge Mathematical Library, 2nd edition,1993. ISBN: 0 521 45897 8

[4] Ronald C. Read, Robin J. Wilson, *An Atlas of Graphs*. Oxford Science Publications, 1998. ISBN: 0 19 853289

[5] Martin Dyer, Catherine Greenhill *The complexity of counting graph homomorphisms* , Paper (*Link*)

[6] William Stein, Graph Theory in Sage ,Talk (*Link*)

[7] Rob Beezer, Chris Godsil Explorations in Algebraic Graph Theory with Sage: , Practice Problems (*Link*)

Author: Davis W. Shurbert

http://creativecommons.org/licenses/by/3.0/