Graph Theory

Adam Kelly

January 26, 2021

This set of notes is a work-in-progress account of the course 'Graph Theory', originally lectured by Dr Julian Sahasrabudhe in Lent 2020 at Cambridge. These notes are not a transcription of the lectures, but they do roughly follow what was lectured (in content and in structure).

These notes are my own view of what was taught, and should be somewhat of a superset of what was actually taught. I frequently provide different explanations, proofs, examples, and so on in areas where I feel they are helpful. Because of this, this work is likely to contain errors, which you may assume are my own. If you spot any or have any other feedback, I can be contacted at ak2316@cam.ac.uk.

Contents

1 Introduction

For many people, 'Graph Theory' is a first course in combinatorics. It's an area with a big focus on problem solving, and it can give a perspective on many other areas of mathematics.

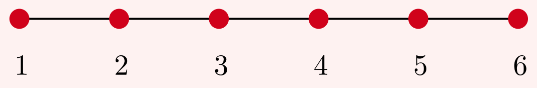
§1.1 Definitions

We will begin our course in graph theory naturally by defining what a graph is.

Definition 1.1.1 (Graph)
A **graph** is an ordered pair $G = (V, E)$ where V is the set of **vertices**, and $E \subseteq \{\{x, y\} \mid x, y \in V, x \neq y\}$ is a set of unordered pairs of vertices called **edges**.

We have a natural way of drawing a graph. For each vertex we have a point in the plane, and for each edge we draw a line between the corresponding pair of vertices.

Example 1.1.2 (Example of a Graph)
The ordered pair (V, E) where $V = \{1, 2, \dots, 6\}$ and $E = \{\{1, 2\}, \{2, 3\}, \dots, \{5, 6\}\}$ is a graph.

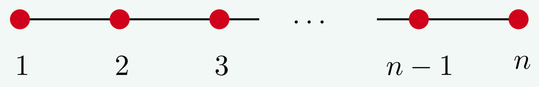


This graph is known as P_6, a path on 6 vertices.

§1.1.1 Common Graphs

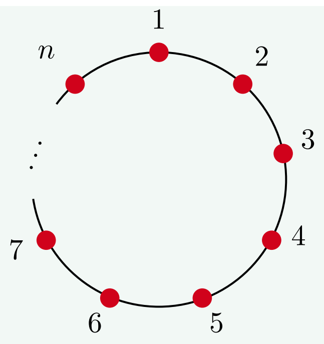
There are some graphs that will appear repeatedly throughout the course, and we will define them now.

Definition 1.1.3 (Path)
We define P_n to be the graph $V = \{1, \dots, n\}$, $E = \{\{1, 2\}, \{2, 3\}, \dots, \{n-1, n\}\}$ as shown.



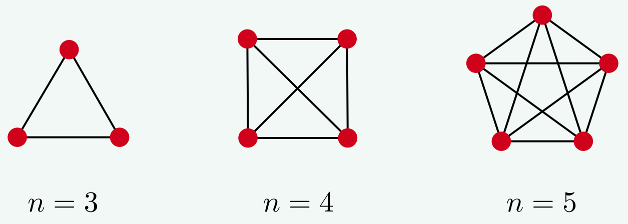
We call this a **path** on n vertices, and say it has **length** $n - 1$.

Definition 1.1.4 (Cycle)
We define C_n (for $n \geq 3$) to be the graph $V = \{1, \dots, n\}$, $E = \{\{1, 2\}, \dots, \{n - 1, n\}, \{n, 1\}\}$ as shown.



We call this the **cycle** on n vertices.

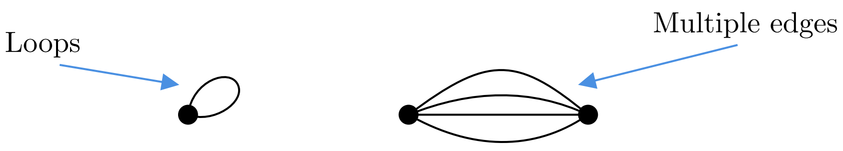
Definition 1.1.5 (Complete Graph)
The **complete graph** on n vertices K_n is the graph $\{1, \dots, n\}$ and $E = \{\{i, j\} \mid i \neq j \in V\}$.



Note that there is an edge between every pair of vertices.

Definition 1.1.6 (Empty Graph)
We define the **empty graph** on n vertices $\overline{K_n}$ to have $V = \{1, \dots, n\}$ but $E = \emptyset$.

Remark. In our definition of a graph, we *don't allow* loops, and there *cannot* be multiple edges between the same set of vertices.



These limitations are inherent in our definition, where we use sets rather than multisets. You can define graphs where such things are allowed, but for now we will outlaw them. We also note that edges are *unordered pairs*, so for now edges have no direction.

To be slightly more succinct, we will use some shorthand notation.

Notation. If $G = (V, E)$ is a graph, and we have some edge $\{x, y\} \in E$, we will denote it by xy. We will also define $|G| = |V|$, and $e(G) = |E|$.

Example 1.1.7 (Vertices and Edges of K_n)
Consider the graph K_n. We have $|K_n| = n$, and $e(K_n) = \binom{k}{2}$, as there is an edge between any pair of vertices.

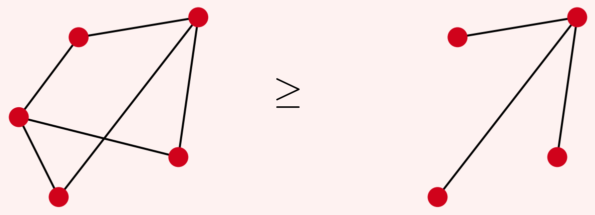
§1.1.2 Subgraphs

Now we will define the notion of a *subgraph*, in the natural way.

Definition 1.1.8 (Subgraph)
We say that $H = (V', E')$ is a **subgraph** of $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$.

Informally, H is a subgraph of G if we can remove vertices and edges from G to get H. Let's look at some examples.

Example 1.1.9 (Example of a Subgraph)
The graph on the right is a subgraph of the graph on the left.



We are also going to use some notation for removing an edge or a vertex from a graph. Of course, when removing a vertex you also have to remove the edges connecting to it.

Notation (Adding/Removing Vertices & Edges)**.** For an edge xy or a vertex x, we define $G - xy$ to be the graph G with the edge xy removed, and $G - x$ to be G with vertex x removed, along with all edges incident to x. We will also define $G + xy$ to be G with the edge xy, and $G + x$ to be G with the vertex x.

§1.1.3 Graph Isomorphism

Now that we have defined graphs, it's natural to define some notion of isomorphism.

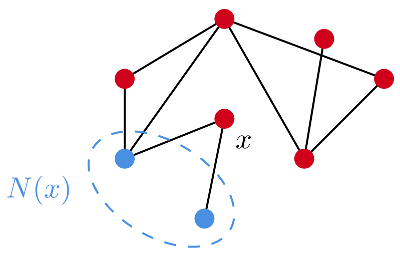
Definition 1.1.10 (Graph Isomorphism)
Let $G = (V, E)$ and $H = (V', E')$ be graphs. We say that $f : V \to V'$ is a **graph isomorphism** if $f(u)f(v) \in E' \iff uv \in E$.

If there is a graph isomorphism between G and H then we say they are **isomorphic**.

Now for the following discussion, fix some graph $G = (V, E)$, and let $x \in V$.

Definition 1.1.11 (Neighbourhood)
If $xy \in E$, then we say that x and y are **adjacent**. We define the **neighborhood** of x to be the set $N(x) = \{y \in V \mid xy \in E\}$ of all vertices adjacent to x.

Note that as in the diagram below, x is not in its own neighborhood.



Definition 1.1.12 (Degree)

We define the **degree** of a vertex x to be $d(x) = |N(x)|$. This is equal to the number of edges that are incident to x.

Definition 1.1.13 (Regularity)

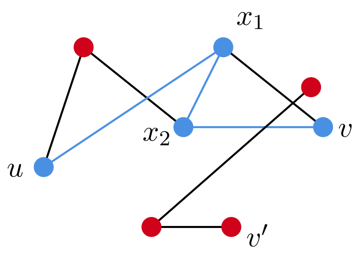
A graph G is said to be **regular** if all of the degrees are the same. We say G is k-regular if $d(x) = k$ for all $x \in V$.

Example 1.1.14 (Regular and Non-Regular Graphs)

The graphs K_n is $n-1$ regular, and C_n is 2-regular. The graph P_n is not regular.

§1.1.4 Connectivity

We now want to define some notion of *connectivity*, where a vertex u is connected to vertex v if you can follow some path in the graph to get from u to v.



For example, in the graph above we want to say somehow that u and v are connected, but u and v' are not. To do this, we will introduce some more definitions.

Definition 1.1.15 (uv Path)

A **uv path** is a sequence $x_1, x_2, \dots, x_l$ where $x_1, \dots, x_l$ are distinct, $x_1 = u$, $x_l = v$ and $x_i x_{i+1} \in E$.

In the example above, ux_1x_2v is a uv path.

The slight subtlety in this condition is the *distinctness* condition. For example, if $x_1 \dots x_l$ is a uv path and $y_1 \dots y_{l'}$ is a vw path, then $x_1 \dots x_l y_1 \dots y_{l'}$ may *not* be a uw path since we may have reused an edge. Of course, we can just not reuse edges by avoiding cycles.

Proposition 1.1.16 (Joining Paths)

If $x_1 \dots x_l$ is a uv path and $y_1 \dots y_{l'}$ is a vw path, then $x_1 \dots x_l y_1 \dots y_{l'}$ contains a uw path.

Proof. Choose a minimal subsequence $w_1 \dots w_r$ of $x_1 \dots x_l y_1 \dots y_{l'}$ such that

1. $w_i w_{i+1} \in E$.
2. $w_1 = u$, $w_r = w$.

We now claim that $w_1 \dots w_r$ is a uw path. If this was not the case, then it must fail on distinctness, so there would exist some z such that the sequence is

$$w_1 \dots w_a z w_{a+2} \dots w_b z w_{b+2} w_r,$$

but now note that

$$w_1 \dots w_a z w_{b+2} \dots w_r$$

also satisfies the conditions for the subsequence, but is strictly shorter length. This contradicts the minimality condition. □

Now given $G = (V, E)$, let's define an equivalence relation $\sim$ on V, where

$$x \sim y \iff \text{there exists an } xy \text{ path in } G.$$

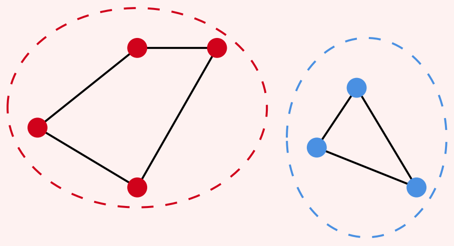
Proposition 1.1.17

$\sim$ is an equivalence relation.

Proof. Note that $\sim$ is reflexive and symmetric, and we get transitivity from our previous proposition. □

Example 1.1.18

In the graph below, the vertices that are the same colour are in the same equivalence class under $\sim$.



Definition 1.1.19 (Connected Graph)

If there is a path between any two vertices in G then we say that G is **connected**.

Definition 1.1.20 (Connected Components)

We call the equivalence classes of $\sim$ on G the **components** or **connected components** of G.

§1.1.5 Edges and Distance

We can also introduce some useful definitions relating to the edges of a graph.

Definition 1.1.21 (Minimum/Maximum Degree)

Let G be a graph. The **maximum degree** of G, $\triangle(G)$ is defined to be $\triangle(G) = \max_{x \in V} d(x)$. Similarly, we define the **minimum degree** of G, $\delta(G)$ to be $\delta(G) = \min_{x \in V} d(x)$.

In a k-regular graph as mentioned above, we have $\triangle(G) = \delta(G) = k$.

Definition 1.1.22 (Graph Distance)

Let $G = (V, E)$ be a graph. The associated **graph distance** $d : V \times V \to \mathbb{R}^{\geq 0} \cup \{\infty\}$ is defined so that $d(x, y)$ is the minimum path length from x to y if it exists, and ∞ otherwise.

Proposition 1.1.23 (Graph Distance is a Metric)

Let $G = (V, E)$ be a connected graph and let d be the associated graph distance. Then (V, d) defines a metric space.

Proof Sketch. We have $d(x, y) = 0$, and $d(x, y) = d(y, x)$ (taking the shortest path in the opposite direction), and $d(x, z) \leq d(x, y) + d(y, z)$ as we can find path from x to z by taking paths from x to y and y to z and adjoining them, and this puts an upper bound on $d(x, z)$. □

§1.2 Trees

We will now discuss a special class of graph called *trees*. This class is quite restrictive (yet is quite useful), and they have some nice properties.

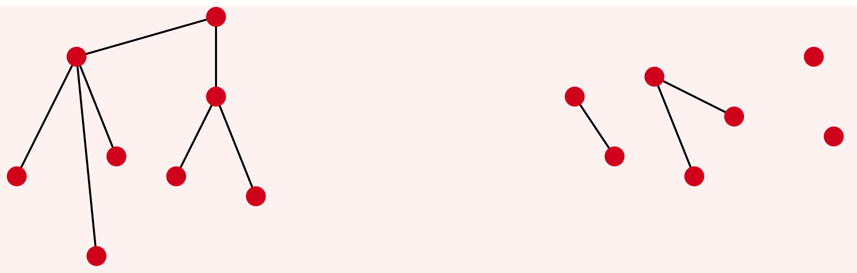
To define what a tree is, we first need a notion of when a graph is acyclic.

Definition 1.2.1 (Acyclic)

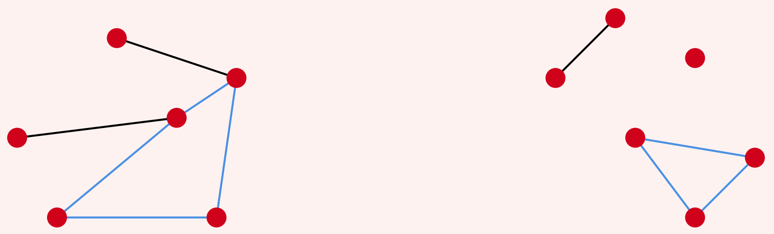
A graph G is said to be **acyclic** if it does not contain any subgraph isomorphic to a cycle, C_n.

Example 1.2.2 (Example of Acyclic/Non-Acyclic Graphs)

In the example below, the two graphs are both *acyclic*.



Two *non-acyclic* graphs are shown below. The subgraphs isomorphic to C_4 and C_3 are highlighted.

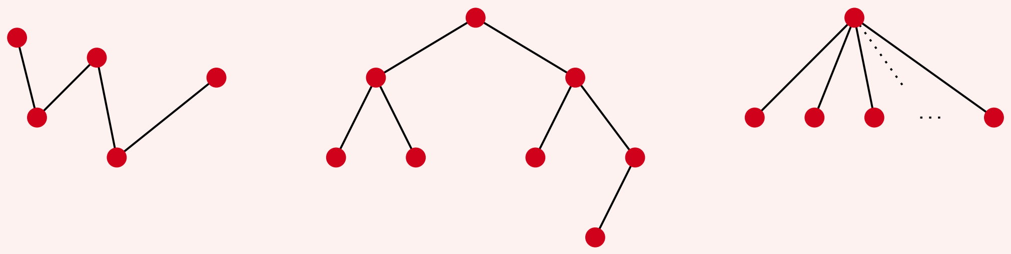


Definition 1.2.3 (Tree)

A **tree** is a connected, acyclic graph.

Example 1.2.4 (Examples of Trees)

The following three graphs are trees.



Proposition 1.2.5 (Characterising Trees)

The following are equivalent.

(a) G is a tree.

(b) G is a maximal acyclic graph (adding any edge creates a cycle).

(c) G is a minimal connected graph (removing any edge disconnects the graph).

Proof. *(a) $\implies$ (b).* By definition G is acyclic. Let $x, y \in V$ such that $xy \notin E$. As G is connected, there is an xy path P. The that xPy then defines a cycle.

(b) $\implies$ (a). By definition G is acyclic. So for a contradiction assume G is not connected and let x, y be vertices from different components. Now note $G + xy$ is acyclic, but this contradicts the claim that G is maximally acyclic.

(a) $\implies$ *(c)*. By definition G is connected. Suppose, for a contradiction, that there exists some vertices $x, y \in E$ with $x \neq y$ and $G - xy$ is connected. But then there is some xy path P that does not use the edge xy, so xPy is then a cycle, contradicting that G is acyclic.

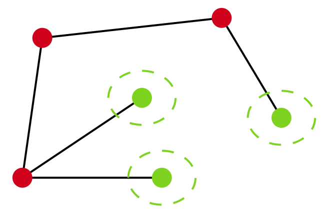
(c) $\implies$ *(a)*. By definition G is connected. Again for a contradiction, assume that G contains a cycle C. Then let xy be an edge on C. We claim $G - xy$ is still connected. If $u, v \in V(G - xy)$ then let P be a path in G from u to v. If xy does not appear as consecutive vertices on this path, then u is connected to v. Otherwise, we can consider a new path where we replace x, y with the other vertices in $C - xy$ in order. Thus u and v are still connected. This contradicts the minimal connectedness of G.

□

Definition 1.2.6 (Leaf)

Let G be a graph. A vertex $v \in V(G)$ is a **leaf** if $d(v) = 1$.

For example, the tree below has three leaves.



In general, trees has a leaf.

Proposition 1.2.7 (Trees Have Leaves)

Every tree T with $|T| \geq 2$ has a leaf.

Proof. Let T be a tree with $|T| \geq 2$, and let P be a path of maximum length in T, with $P = x_1 \dots x_k$. We claim that $d(x_k) = 1$. Observe that $\deg(x_k) \geq 1$, since $x_k x_{k-1} \in E$. If x_k is adjacent to another vertex $y \neq x_{k-1}$, then either $y \in \{x_1, \dots, x_{k-2}\}$, which would imply thar T contains a cycle, or $y \notin \{x_1, \dots, x_{k-2}\}$, then $x_1 \dots x_k y$ is a path longer than P, which violates its maximality. □

Remark. This proof gives us two leaves in T, which is the best we can hope for considering P_n is a tree with exactly two leaves.

Proposition 1.2.8 (Edges of a Tree)

Let T be a tree. Then $e(T) = |T| - 1$.

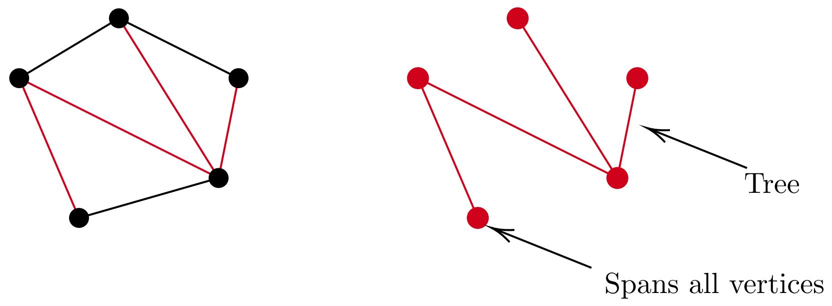
Proof. We will do induction on $n = |T|$. If $n = 1$, this is trivial as there is only one edge. Now given T with at least 2 vertices, let x be a leaf in T, and define $T' = T - x$.

T' must be acyclic, since we have only removed vertices. T' must also be connected since for all $u, v \in V(T')$ there exists a path from u to v in T that does not use x, so it is also a path from u to v in T'. Thus T' is a tree. Thus by induction, T' has $n-2$ edges, and $e(T) = e(T') + 1 = |T| - 1$. □

Now lets think about trees as subgraphs of other graphs.

Definition 1.2.9 (Spanning Tree)

Let G be a graph. We say T is a **spanning tree** of G if T is a tree on $V(G)$ and is a subgraph of G.



Spanning trees are useful in a number of contexts, one of which is giving a sensible ordering to the vertices of a graph. They are particularly useful because of the following result.

Proposition 1.2.10 (Connected Graphs have Spanning Trees)

Every connected graph contains a spanning tree.

Proof. A tree is a minimal connected graph. So take the connected graph and remove edges until it becomes a minimal connected graph. Then this will be a subgraph of the original graph, and will thus be a spanning tree. □

§1.3 Bipartite Graphs

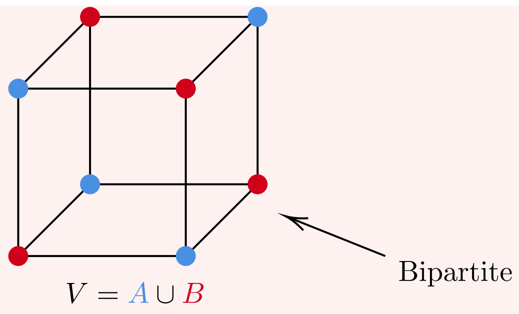
The next type of graph we will look at is *bipartite* graphs.

Definition 1.3.1 (Bipartite Graphs)

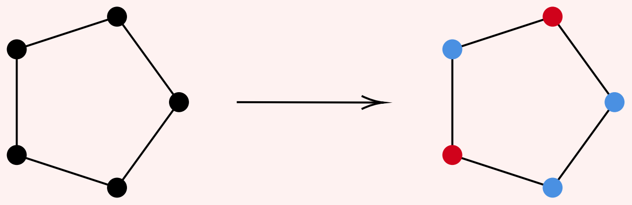
A graph $G = (V, E)$ is **bipartite** if $V = A \cup B$ where $A \cap B = \emptyset$ and all edges $xy \in E$ have either $x \in A, y \in B$ or $x \in B, y \in A$.

Example 1.3.2 (Example of Bipartite Graphs)

The graph below is bipartite, with vertices in the set A being coloured red and vertices in the set B being coloured blue.



An example of a non-bipartite graph is C_5. To see this, we can start by choosing a vertex to be in A (without loss of generality), then the adjacent vertices must be in B, but then their adjacent vertices must be in A, but then there is an edge between two vertices in A. This is shown below.



C_5 can't be bipartite

The argument given for C_5 works in general.

Proposition 1.3.3 (Bipartite Cyclic Graphs)

The cycle C_{2k+1} is *not* bipartite, and the cycle C_{2k} *is* bipartite.

Proof. Assume that C_{2k+1} is bipartite. Then there must be disjoint sets A and B, and as $2k+1$ is odd, we must have (without loss of generality), $|A| > |B|$. Now let's count the edges between A and B. This must be $2|A|$ and also $2|B|$, as every vertex has degree 2. But then $|A| = |B|$, which is a contradiction.

For C_{2n}, we can let $v_i \in A$ if i is even and $v_i \in B$ if i is odd. Then a vertex i only has edges to vertices $i-1$ and $i+1 \pmod 2$, which have opposite parity. Thus all edges are between A and B, as required. □

There is then a natural question: given some arbitrary graph G, how do we determine if a given graph is bipartite? It turns out that there is a nice check for 'bipartness'. We will state the result and then do some setup before we prove it.

Proposition 1.3.4 (Bipartite Criterion)

A graph G is bipartite if and only if G contains no odd cycles.

We need to first develop some theory regarding *circuits*. Informally, a circuit is like a cycle where we can revisit vertices.

Definition 1.3.5 (Circuit)

A circuit is a sequence $x_1 \dots x_l$ where $x_1 = x_l$ and $x_i x_{i+1} \in E$. The **length** of the circuit is $l-1$, the number of edges traversed in the circuit.

Definition 1.3.6 (Odd Circuits)

If the length of a circuit is odd, then we say it is an **odd circuit**.

Proposition 1.3.7

An odd circuit contains an odd cycle.

Proof. We will prove this by induction on the length of the circuit. For a circuit of length 3, the circuit must be a cycle. In general, let $C = x_1 \dots x_l$ be our circuit. If $x_1, \dots, x_{l-1}$ are distinct, then C is a cycle and we are done.

Otherwise, there exists some $z \in C$ that is repeated. We write

$$C = x_1 \dots x_a z x_{a+2} \dots x_b z x_{b+2} \dots x_l.$$

We define $C' = x_1 \dots x_a z x_{b+2} \dots x_l$ and $C'' = z x_{a+2} \dots x_b z$. The length of C' and C'' is strictly less than the length of C. One of these circuits must have odd length, and by induction that odd circuit contains an odd cycle. □

We can now prove our original bipartness criterion, that a graph is bipartite if and only if it contains no odd cycles.

Proof (Bipartite Criterion). If G was bipartite and contained an odd cycle, then there exists an odd cycle that is bipartite. But this is a contradiction.

Now if G is not bipartite, we can induct on the number of vertices. For $|G| = 1$, this holds. Now if G is not connected, let $C_1, \dots, C_k$ be the components of G. We may now apply our induction to each component of G to obtain a bipartitian $V(C_i) = A_i \cup B_i$ for each $i \in 1, \dots, k$. Then $A = A_1 \cup \dots A_k$ and $B = B_1 \cup B_k$ is a bipartitian for the whole graph.

We may now assume without loss of generality that G is connected. Fix some vertex $v \in V$, and define

$$A = \{u \in V \mid d(u, v) \text{ is odd}\}$$
$$B = \{u \in V \mid d(u, v) \text{ is even}\}$$

We claim that $A \cup B$ is a bipartitian.

Assume (for a contradiction) that u_1 is adjacent to u_2 and $d(u_1, v) \cong d(u_2, v) \pmod 2$. Then there exists paths P_1 from v to u_1 and P_2 from u_2 to v with $|P_1| \equiv |P_2|$. But this implies that $vP_1u_1u_2P_2$ defines a odd circuit in G. Therefore, by our previous proposition, G contains an odd cycle, which is a contradiction. □