Week 07: Graphs

FanFly

April 19, 2020

The Seven Bridges of Königsberg

The **Seven Bridges of Königsberg** is considered to be the first problem of graph theory.

The Seven Bridges of Königsberg

The **Seven Bridges of Königsberg** is considered to be the first problem of graph theory.

In this problem, people would like to know whether a citizen can take a walk through the town in such a way that each bridge would be crossed exactly once.

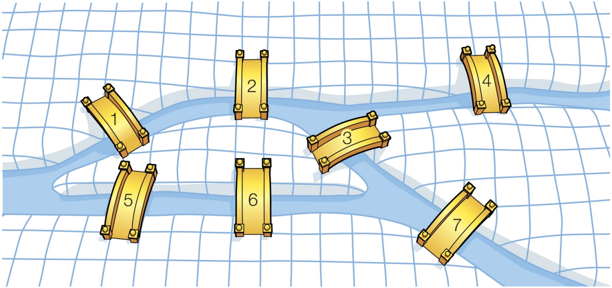
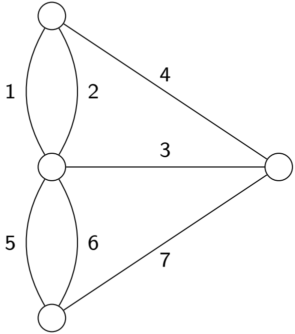


Figure: The Seven Bridges of Köingsberg

Abstraction

The towns and bridges can be seen as vertices and edges in a graph.



Graphs

In the following weeks, we are going to introduce some problems related to graphs.

Graphs

In the following weeks, we are going to introduce some problems related to graphs.

The definition of a graph is as follows.

Definition

A **simple graph** is a pair $G = (V, E)$, where each component is as follows.

Graphs

In the following weeks, we are going to introduce some problems related to graphs.

The definition of a graph is as follows.

Definition

A **simple graph** is a pair $G = (V, E)$, where each component is as follows.

- V is a finite collection of **vertices**.

Graphs

In the following weeks, we are going to introduce some problems related to graphs.

The definition of a graph is as follows.

Definition

A **simple graph** is a pair $G = (V, E)$, where each component is as follows.

- V is a finite collection of **vertices**.
- E is a collection of **edges**, where each edge is of the form $e = \{u, v\}$ for some $u, v \in V$.

Graphs

In the following weeks, we are going to introduce some problems related to graphs.

The definition of a graph is as follows.

Definition

A **simple graph** is a pair $G = (V, E)$, where each component is as follows.

- V is a finite collection of **vertices**.
- E is a collection of **edges**, where each edge is of the form $e = \{u, v\}$ for some $u, v \in V$.

If we allow a graph to have more than one edges that have the same endpoints, then the graph is called a **multigraph**.

Eulerian Path Problem

Now we can formalize the Seven Bridges of Königsberg with new terminology.

Eulerian Path Problem

Now we can formalize the Seven Bridges of Königsberg with new terminology.

- A **path** is a sequence of edges that joins a sequence of vertices.

Eulerian Path Problem

Now we can formalize the Seven Bridges of Königsberg with new terminology.

- A **path** is a sequence of edges that joins a sequence of vertices.
- An **Eulerian path** is a path visiting each edge exactly once.

Eulerian Path Problem

Now we can formalize the Seven Bridges of Königsberg with new terminology.

- A **path** is a sequence of edges that joins a sequence of vertices.
- An **Eulerian path** is a path visiting each edge exactly once.

Then a walk through the town in such a way that each bridge would be crossed exactly once is exactly an Eulerian path.

Eulerian Path Problem

Now we can formalize the Seven Bridges of Königsberg with new terminology.

- A **path** is a sequence of edges that joins a sequence of vertices.
- An **Eulerian path** is a path visiting each edge exactly once.

Then a walk through the town in such a way that each bridge would be crossed exactly once is exactly an Eulerian path.

Thus, we have the following problem.

Eulerian Path Problem

Now we can formalize the Seven Bridges of Königsberg with new terminology.

- A **path** is a sequence of edges that joins a sequence of vertices.
- An **Eulerian path** is a path visiting each edge exactly once.

Then a walk through the town in such a way that each bridge would be crossed exactly once is exactly an Eulerian path.

Thus, we have the following problem.

Eulerian Path Problem

- Input: A multigraph $G = (V, E)$.
- Output: Whether G has an Eulerian path or not.

The Solution to the Eulerian Path Problem

In fact, we have the following theorem, which simply solves the Eulerian path problem.

The Solution to the Eulerian Path Problem

In fact, we have the following theorem, which simply solves the Eulerian path problem.

Theorem

Let $G = (V, E)$ be a multigraph.

The Solution to the Eulerian Path Problem

In fact, we have the following theorem, which simply solves the Eulerian path problem.

Theorem

Let $G = (V, E)$ be a multigraph.

- If each vertex in G has an even degree, then G has an Eulerian path that starts and ends on the same vertex.

The Solution to the Eulerian Path Problem

In fact, we have the following theorem, which simply solves the Eulerian path problem.

Theorem

Let $G = (V, E)$ be a multigraph.

- If each vertex in G has an even degree, then G has an Eulerian path that starts and ends on the same vertex.
- If there are exactly two vertices of odd degree, then G has an Eulerian path that starts on one of them and ends at the other.

The Solution to the Eulerian Path Problem

In fact, we have the following theorem, which simply solves the Eulerian path problem.

Theorem

Let $G = (V, E)$ be a multigraph.

- If each vertex in G has an even degree, then G has an Eulerian path that starts and ends on the same vertex.
- If there are exactly two vertices of odd degree, then G has an Eulerian path that starts on one of them and ends at the other.
- If there are more than two vertices of odd degree, then G has no Eulerian path.

The Solution to the Eulerian Path Problem

In fact, we have the following theorem, which simply solves the Eulerian path problem.

Theorem

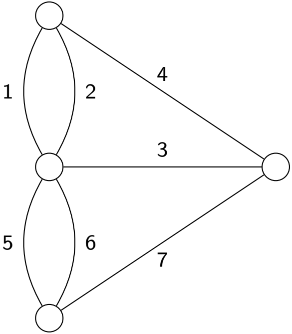
Let $G = (V, E)$ be a multigraph.

- If each vertex in G has an even degree, then G has an Eulerian path that starts and ends on the same vertex.
- If there are exactly two vertices of odd degree, then G has an Eulerian path that starts on one of them and ends at the other.
- If there are more than two vertices of odd degree, then G has no Eulerian path.

The **degree** of a vertex is the number of its incident edges.

The Solution to the Eulerian Path Problem (cont.)

Thus, no one can take a walk through the town in such a way that each bridge would be crossed exactly once.



Representation of Graphs

Now we can solve the Eulerian path problem.

Representation of Graphs

Now we can solve the Eulerian path problem.

But how can we “store” a graph in an computer?

Representation of Graphs

Now we can solve the Eulerian path problem.

But how can we “store” a graph in an computer? We need a **data structure**!

Representation of Graphs

Now we can solve the Eulerian path problem.

But how can we "store" a graph in an computer? We need a **data structure**!

We will focus on the efficiency of the following operations.

Representation of Graphs

Now we can solve the Eulerian path problem.

But how can we “store” a graph in an computer? We need a **data structure**!

We will focus on the efficiency of the following operations.

- Initialize.

Representation of Graphs

Now we can solve the Eulerian path problem.

But how can we "store" a graph in an computer? We need a **data structure**!

We will focus on the efficiency of the following operations.

- Initialize.
- Check if an edge exists.

Representation of Graphs

Now we can solve the Eulerian path problem.

But how can we "store" a graph in an computer? We need a **data structure**!

We will focus on the efficiency of the following operations.

- Initialize.
- Check if an edge exists.
- List all neighbors of a vertex.

Representation of Graphs

Now we can solve the Eulerian path problem.

But how can we "store" a graph in an computer? We need a **data structure**!

We will focus on the efficiency of the following operations.

- Initialize.
- Check if an edge exists.
- List all neighbors of a vertex.
- List all edges.

Attempt #1: Edge List

Edge List

```
n = 6
edges = [[0, 1], [0, 2], [1, 3], [2, 3], [4, 5]]
```

Attempt #2: Adjacency List

Adjacency List

```
n = 6
neighbor = [
    [1, 2],     # neighbor of vertex 0
    [0, 3],     # neighbor of vertex 1
    [0, 3],     # neighbor of vertex 2
    [1, 2],     # neighbor of vertex 3
    [5],        # neighbor of vertex 4
    [4]         # neighbor of vertex 5
]
```

Attempt #3: Adjacency Matrix

Adjacency Matrix

```
n = 6
matrix = [
    [False, True, True, False, False, False],
    [True, False, False, True, False, False],
    [True, False, False, True, False, False],
    [False, True, True, False, False, False],
    [False, False, False, False, False, True],
    [False, False, False, False, True, False]
]
```

Exercise #1

A **complete graph** is a graph in which each pair of vertices is connected by an edge.

Exercise #1

A **complete graph** is a graph in which each pair of vertices is connected by an edge.

What is the number of edges of an n-vertex complete graph?
(By the way, the n-vertex complete graph is denoted by K_n).

Exercise #1

A **complete graph** is a graph in which each pair of vertices is connected by an edge.

What is the number of edges of an n-vertex complete graph?
(By the way, the n-vertex complete graph is denoted by K_n).

Solution

There are $n(n-1)/2$ edges in K_n.

Exercise #2

Let $G = (V, E)$ be a graph with $|V| = n$.

Exercise #2

Let $G = (V, E)$ be a graph with $|V| = n$.

How much time does it take to find the degree of a vertex in V if G is stored as an adjacency matrix?

Exercise #2

Let $G = (V, E)$ be a graph with $|V| = n$.

How much time does it take to find the degree of a vertex in V if G is stored as an adjacency matrix?

Solution

It takes $\Theta(n)$ time.

Exercise #3

A **coloring** of a graph is a labeling of vertices with colors such that no two adjacent vertices have the same color.

Exercise #3

A **coloring** of a graph is a labeling of vertices with colors such that no two adjacent vertices have the same color.

What is the minimum number of colors needed to color K_5?

Exercise #3

A **coloring** of a graph is a labeling of vertices with colors such that no two adjacent vertices have the same color.

What is the minimum number of colors needed to color K_5?

Solution

We need 5 colors to color K_5.