Leonard Euler 1707-1783
The Greatest Mathematician of All Time

- Analysis
- Number Theory
- Created Graph Theory

Euler

Can you take a walk, crossing each bridge exactly once?
Konigsberg

Easier to see

Euler Tour

- Each bridge is an edge
- Each part of town is a vertex
- Is there a path that crosses each edge exactly once?

If we come in on one bridge, we go out by a different bridge

Hence if degree of vertex is odd, we have to start or finish at that vertex

So if more than two odd-degree vertices, we can’t do it
What’s the fastest way from Seattle to Spokane?

What’s the cheapest inter-city network?

If we lose Wenatchee, can Seattle still talk to Spokane?
Directed Graphs

Downtown Seattle

Organize Your Life

We won’t talk much about these graphs, but there’s a homework problem on them.

Definitions

All our graphs will have at most one edge between vertices and no self-loops
More

Path

Simple Path

Simple Cycle

Cycle

Still More

Connected Graph

Disconnected Graph

Tree

Not a Tree

Why is this a tree?
Graph Searching

How do we...

- Find a path from \( u \) to \( v \)?
- Find a short path from \( u \) to \( v \)?
- Decide if \( G \) is connected?
- Decide if \( G \) has any cycles?

Representing Graphs

\[ G = V + E \]

Verticies: \( a, b, c, d, e \)

Edges: \( (a, b), (b, d), (a, d), (e, d) \)
Adjacency Matrix

How to

- find if there is an edge between \( u \) and \( v \)?
- iterate over all neighbors?
- add an edge?
- delete an edge?
- add a vertex?
- delete a vertex?

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

A Nice Representation

Adjacency List Representation:

- \( a \): \( b, d \)
- \( b \): \( a, d \)
- \( c \): 
- \( d \): \( a, b, e \)
- \( e \): \( d \)

Adjacency Lists

```c
struct Vertex {
    // Vertex structure
};

struct Graph {
    // Graph structure
};
```
Our First Graph Algorithm

Breadth First Search

Explore vertices in order of distance from the start

BFS

```c
NumberBFS(Graph G, Vertex *root)
{
    for each (v in G) {
        Encountered(v) = false;
        Number(v) = -1;
    }
    VertexQueue Q;
    Encountered(root) = true;
    Number(start) = 1;
    next_num = 2;
    Q.enQ(start);
    while(!Q.Empty()) {
        Vertex *v = Q.deQ();
        Number(v) = next_num++;
        for each (w in v->Neighbors())
            if (!Encountered(w)) {
                Encountered(w) = true;
                Q.enQ(w);
            }
    }
}
```
Using NumberBFS

How do we...

- determine if \( G \) is connected?
- find the distance from the root to a node?
- determine if \( G \) has any cycles?
- determine if \( G \) is a tree?
- find a path from the root to a node?