15—Graphs I

§12.1–12.3

May 15, 2002

Euler

Leonard Euler 1707-1783

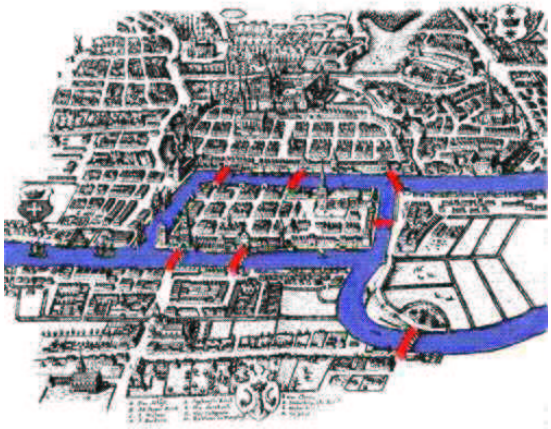
The Greatest Mathematician of All Time

- Analysis
- Number Theory
- Created *Graph Theory*

Konigsberg

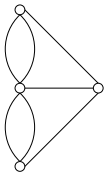
Can you take a walk, crossing each bridge exactly once?

Konigsberg



Easier to see

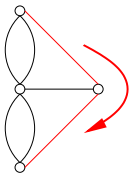
Konigsberg



Euler Tour

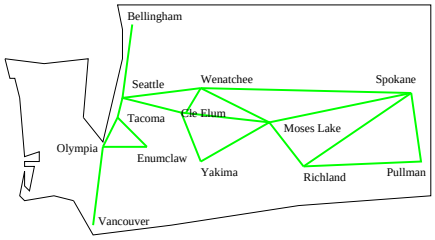
- Each bridge is an *edge*
- Each part of town is a *vertex*
- Is there a path that crosses each *edge* exactly once?

Konigsberg



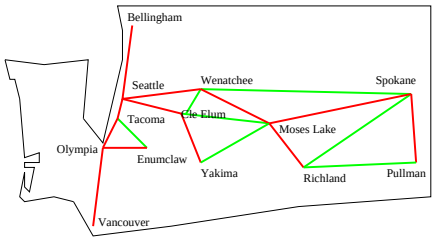
- If we come *in* on one bridge, we go *out* by a different bridge
- Hence if *degree* of vertex is *odd*, we have to start or finish at that vertex
- So if more than *two* odd-degree verticies, we can't do it

Washington



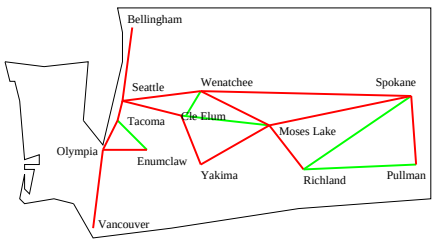
What's the fastest way from Seattle to Spokane?

Washington



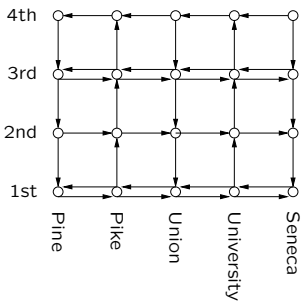
What's the cheapest inter-city network?

Washington



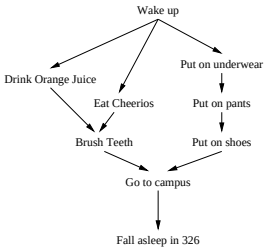
If we lose Wenatchee, can Seattle still talk to Spokane?

Directed Graphs



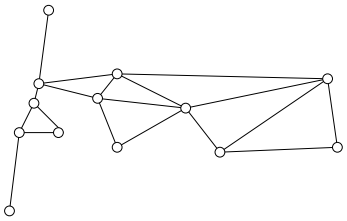
Downtown Seattle

Organize Your Life



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

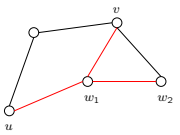
Definitions



- vertices
- edges
- degree
- neighbor

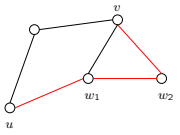
All our graphs will have at most *one* edge between verticies and no self-loops

More



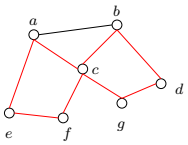
$u\ w_1\ w_2\ w_1\ v$

Path



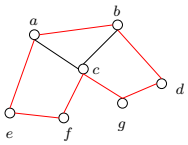
$u\ w_1\ w_2\ v$

Simple Path



$a\ c\ g\ d\ b\ c\ f\ e\ a$

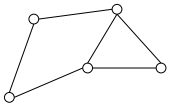
Cycle



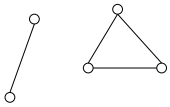
$a\ b\ d\ g\ c\ f\ e\ a$

Simple Cycle

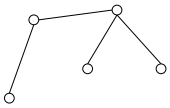
Still More



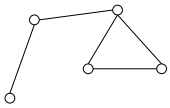
Connected Graph



Disconnected Graph

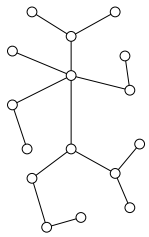


Tree

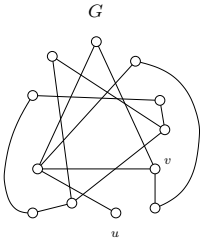


Not a Tree

Why is this a tree?



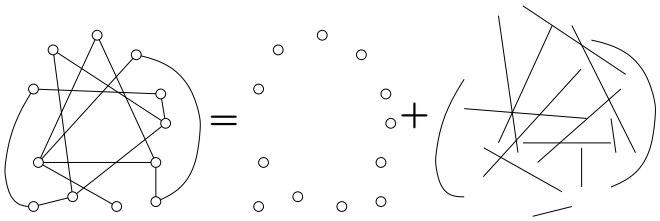
Graph Searching



How do we. . .

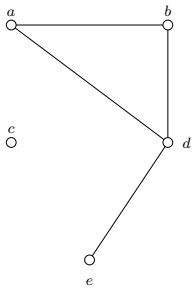
- Find a path from u to v?
- Find a short path from u to v?
- Decide if G is connected?
- Decide if G has any cycles?

Representing Graphs



$$G = V + E$$

Representing Graphs



Verticies: a, b, c, d, e

Edges: $(a, b), (b, d), (a, d), (e, d)$

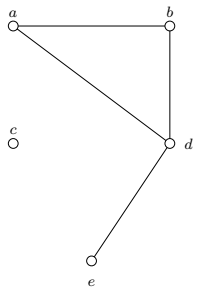
Adjacency Matrix

	a	b	c	d	e
a	0	1	0	1	0
b	1	0	0	1	0
c	0	0	0	0	0
d	1	1	0	0	1
e	0	0	0	1	0

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?

A Nice Representation



Adjacency List Representation:

a: b, d

b: a, d

c:

d: a, b, e

e: d

Adjacency Lists

```
struct Vertex {




};
struct Graph {




};
```

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?

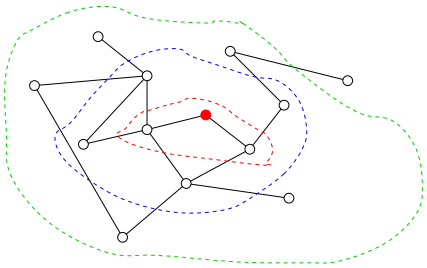
Adjacency Lists

```
class Vertex {

};
```

```
class Graph {

};
```


Our First Graph Algorithm

Breadth First Search

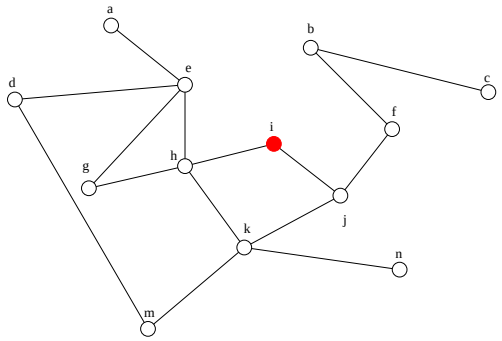


Explore vertices in order of distance from the start

BFS

```
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.Emtpy()) {
    Vertex *v = Q.deQ();
    Number(v) = next_num++;
    for each (w in v->Neighbors())
      if (!Encountered(w)) {
        Encountered(w) = true;
        Q.enQ(w);
      }
  }
}
```

NumberBFS in Action



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?

The BFS Tree