A walk in a graph \( G = (V,E) \) is a finite alternating sequence of vertices and edges, \( W = v_0 e_1 v_1 e_2 v_2 \ldots e_k v_k \) where each \( e_i = (v_i,v_{i+1}) \). \( W \) is a \( v_0 - v_k \) walk of length \( k \).

\( v_0, v_1, v_2, \ldots, v_{k-1} \) are the internal vertices of walk \( W \).

An \( a-b \) walk of length 8.

Edges can be omitted in a simple graph.

A trivial walk is a walk of length 0.

A non-trivial \( u-v \) walk is closed if \( u = v \).

A non-trivial \( u-v \) walk is a trail if no edge is repeated.

A \( u-v \) walk is a path if no vertex is repeated.

A cycle is a non-trivial closed trail in which the internal vertices are distinct from each other and the first vertex.

\( G \) is acyclic if it contains no cycles.

**Thm 1.3** Every \( x-y \) walk contains as a subsequence an \( x-y \) path.

**Proof:** By induction on \( k \), the length of the \( x-y \) walk.

\( W = v_0 e_1 v_1 e_2 v_2 \ldots e_k v_k \)

**Base:** \(|W|=0\). Then \( x=y \) and \( W \) is a trivial path.

**Step:** \(|W|=k>0\) and assume the theorem is true for walks of length less than \( k \).

If no vertex is repeated in \( W \) then \( W \) is a path.

Otherwise there are repeated vertices. That is, \( v_i = v_j \) for some indices \( i \) and \( j \) with \( 0 \leq i < j \leq k \).

Let \( W' = v_0 e_1 v_1 e_2 v_2 \ldots e_{i-1} v_i e_i \ldots e_j v_j \).

\(|W'|=k-(i-j)<k\).

So by induction \( W' \) contains as a subsequence an \( x-y \) path \( P \).

Since \( W' \) is a subsequence of \( W \), \( P \) is also a subsequence of \( W \).

So \( W \) has as a subsequence an \( x-y \) path, \( P \).
Connectivity

Vertices, \(u\) and \(v\), are connected in \(G\) if there is a \(u-v\) walk (path) in \(G\).

Connectivity is an equivalence relation on vertices of a graph.

1. Reflexive: \(u\) is connected to \(u\) by a trivial walk.
2. Symmetric: A \(u-v\) walk can be reversed to obtain a \(v-u\) walk.
3. Transitive: A \(u-v\) walk can be concatenated with a \(v-z\) walk to obtain a \(u-z\) walk.

The vertices of a graph can be divided into equivalence classes called connected components.

The vertices of a graph can be divided into equivalence classes called connected components \(V = C_1 \cup C_2 \cup \ldots \cup C_k\), \(k \leq V\) are the number of connected components of \(G\).

Finding Connected Components

\[
\begin{align*}
&\text{begin} \\
&S = V; \ k = 0 \\
&\text{while} \ S \neq \emptyset \\
&\quad \ k = k + 1 \\
&\quad \ v \leftarrow \text{a vertex from } S \\
&\quad \ Q \leftarrow (v); \ C[k] \leftarrow (v) \\
&\quad \text{while} \ Q \neq \emptyset \\
&\quad \quad \ u \leftarrow \text{a vertex from } Q \\
&\quad \quad \text{for each edge } e = (u, w) \text{ of } u \\
&\quad \quad \quad \text{if } w \text{ in } S \\
&\quad \quad \quad \text{then } S \leftarrow S \cup \{w\}; \ Q \leftarrow Q \cup \{w\}; \ C[k] \leftarrow C[k] \cup \{w\} \\
&\quad \quad \end{align*}
\]

For example:

\[
\begin{align*}
\text{S = \{b,c,d,e,f,g,h,i\}} \\
\text{V = a} \\
\text{Q = \{a\}} \\
\text{C[1] = \{a\}}
\end{align*}
\]
C[1] = {a}
S = {a,b,c,d}
Q = {}

v = a
u = a
Q = {b,f}
C[1] = {a,b,f}

k=2
S = {e,g,h,i}
Q = {f,d}
C[1] = {a,b,d,f}
C[2] = {c}

k=2
S = {e,g,h,i}
Q = {e,g}
C[1] = {a,b,d,f}
C[2] = {c,e,g,h}

\[ k=2 \quad S = \{i\} \]
\[ v = C \quad u = h \]
\[ Q = \{\} \]
\[ C[1] = \{a,b,d,f\} \quad C[2] = \{c,e,g,h\} \]

\[ k=3 \quad S = \{\} \]
\[ v = i \quad u = i \]
\[ Q = \{\} \]
\[ C[1] = \{a,b,d,f\} \quad C[2] = \{c,e,g,h\} \quad C[3] = \{i\} \]

**Analysis**

<table>
<thead>
<tr>
<th># times</th>
<th># cost</th>
<th>( n )=# vertices</th>
<th>( m )=# edges</th>
<th>( k )=# components</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \Theta(1) )</td>
<td>\begin{align*} &amp; S \leftarrow V; \ k \leftarrow 0 \ &amp; k \leftarrow 1 \end{align*}</td>
<td>( \Theta(1) )</td>
<td>( \Theta(1) )</td>
</tr>
<tr>
<td>( k+1 )</td>
<td>( \Theta(1) )</td>
<td>while ( S \neq \emptyset )</td>
<td>( \Theta(1) )</td>
<td>( \Theta(1) )</td>
</tr>
<tr>
<td>( k )</td>
<td>( \Theta(1) )</td>
<td>\begin{align*} &amp; k \leftarrow k+1 \ &amp; v \leftarrow \text{remove a vertex from } S \ &amp; k \leftarrow \Theta(1) \end{align*}</td>
<td>( \Theta(1) )</td>
<td>( \Theta(1) )</td>
</tr>
<tr>
<td>( k )</td>
<td>( \Theta(1) )</td>
<td>\begin{align*} &amp; Q \leftarrow {v}; C[k] \leftarrow {v} \ &amp; k \leftarrow \Theta(1) \end{align*}</td>
<td>( \Theta(1) )</td>
<td>( \Theta(1) )</td>
</tr>
<tr>
<td>( k+n )</td>
<td>( \Theta(1) )</td>
<td>while ( Q \neq \emptyset )</td>
<td>( \Theta(1) )</td>
<td>( \Theta(1) )</td>
</tr>
<tr>
<td>( n )</td>
<td>( \Theta(1) )</td>
<td>\begin{align*} &amp; u \leftarrow \text{remove a vertex from } Q \ &amp; 2m \ ? \end{align*}</td>
<td>( \Theta(1) )</td>
<td>( \Theta(1) )</td>
</tr>
<tr>
<td>( 2m )</td>
<td>( \Theta(1) )</td>
<td>\begin{align*} &amp; \text{for each edge } e = (u,w) \text{ of } u \ &amp; 2m \ ? \end{align*}</td>
<td>( \Theta(1) )</td>
<td>( \Theta(1) )</td>
</tr>
<tr>
<td>( n )</td>
<td>( \Theta(1) )</td>
<td>\begin{align*} &amp; \text{if } w \text{ in } S \ &amp; \text{end} \end{align*}</td>
<td>( \Theta(1) )</td>
<td>( \Theta(1) )</td>
</tr>
</tbody>
</table>

**Representations**

**Edge List**

\[ e=e(v,v) \]
\[ e=e(v,v) \]
\[ e=e(v,v) \]
\[ e=e(v,v) \]

The number of vertices, \( n \), must also be supplied.

**Adjacency Lists**

If the graph is simple, edges can be replaced by vertices.
Incidence Matrix $M(G)$

An $n \times m$ matrix where $n$ = #vertices and $m$ = #edges

$m_{ij} = \#$ times edge $e_j$ has vertex $v_i$ as an endpoint

$$
\begin{array}{cccccc}
M(G) & e_1 & e_2 & e_3 & e_4 & d(v) \\
\hline
v_1 & 1 & 0 & 0 & 1 & 2 \\
v_2 & 0 & 1 & 0 & 0 & 1 \\
v_3 & 0 & 0 & 2 & 0 & 1 \\
v_4 & 0 & 1 & 0 & 0 & 1 \\
v_5 & 0 & 0 & 0 & 0 & 0 \\
v_6 & 1 & 0 & 0 & 1 & 1 \\
\end{array}
$$

Rows sum to the degrees

Each column sums to 2

Adjacency Matrix $A(G)$

An $n \times n$ matrix where $n$ = #vertices

$a_{ij} = \#$ edges between $v_i$ and $v_j$

$$
\begin{array}{cccccccc}
A(G) & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \\
\hline
v_1 & 0 & 0 & 0 & 0 & 0 & 2 \\
v_2 & 0 & 0 & 0 & 1 & 0 & 0 \\
v_3 & 0 & 0 & 1 & 0 & 0 & 1 \\
v_4 & 0 & 1 & 0 & 0 & 0 & 0 \\
v_5 & 0 & 0 & 0 & 0 & 0 & 0 \\
v_6 & 2 & 0 & 1 & 0 & 0 & 0 \\
\end{array}
$$

Symmetric

For simple graphs, all entries are 0 or 1 and diagonal is 0.

Comparison of Representations

Space

Memory usage

Time for operations

1. Given $v$, report the edges of $v$
2. Given $v$, report the neighbors of $v$
3. Given $u$ and $v$, report the # of edges between $u$ and $v$

Comparison of Representations

<table>
<thead>
<tr>
<th>Representation</th>
<th>$n$=# vertices</th>
<th>$m$=# edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
<td>$\Theta(m)$</td>
<td>$\Theta(m)$</td>
</tr>
<tr>
<td>Find $v$'s edges</td>
<td>$\Theta(m)$</td>
<td>$\Theta(m)$</td>
</tr>
<tr>
<td>Find $N(v)$</td>
<td>$\Theta(m)$</td>
<td>$\Theta(m)$</td>
</tr>
<tr>
<td>Find $u,v$ edges</td>
<td>$\Theta(m)$</td>
<td>$\Theta(m)$</td>
</tr>
<tr>
<td>Edge List</td>
<td>$\Theta(m)$</td>
<td>$\Theta(m)$</td>
</tr>
<tr>
<td>Adjacency Lists</td>
<td>$\Theta(m+n)$</td>
<td>$\Theta(m)$</td>
</tr>
<tr>
<td>Incidence Matrix</td>
<td>$\Theta(mn)$</td>
<td>$\Theta(m)$</td>
</tr>
<tr>
<td>Adjacency Matrix</td>
<td>$\Theta(n^2)$</td>
<td>$\Theta(n)$</td>
</tr>
</tbody>
</table>

$\Theta(m)$ if endpoints stored with edges.
begin
S: V; k: 0
while S≠Ø
k:=k+1
v: remove a vertex from S
Q:=v; C[k]:={v}
while Q≠Ø
u: remove a vertex from Q
for each edge e = (u,w) of u
if w in S
then S:=S-{w}; Q:=Q∪{w}; C[k]:=C[k]∪{w}
end
end
end

Final Analysis

<table>
<thead>
<tr>
<th></th>
<th>Line 8 per u</th>
<th>Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge List</td>
<td>$\Theta(n)$</td>
<td>$\Theta(nm)$</td>
</tr>
<tr>
<td>Adjacency Lists</td>
<td>$\Theta(d(u))$</td>
<td>$\Theta(nm)$</td>
</tr>
<tr>
<td>Incidence Matrix</td>
<td>$\Theta(m)$</td>
<td>$\Theta(nm)$</td>
</tr>
<tr>
<td>Adjacency Matrix</td>
<td>$\Theta(n)$</td>
<td>$\Theta(n^3)$</td>
</tr>
</tbody>
</table>

Time is $\Theta(n+m)$ + $n \times \text{(time to get u's edges)}$

- $n$: # vertices
- $m$: # edges
- $k$: # components