CS4311
Design and Analysis of Algorithms

Lecture 22:
Elementary Graph Algorithms I
About this lecture

• Representation of Graph
  • Adjacency List, Adjacency Matrix

• Breadth First Search
Graph

undirected
directed
Adjacency List (1)

- For each vertex $u$, store its neighbors in a linked list
Adjacency List (2)

- For each vertex \( u \), store its neighbors in a linked list
Adjacency List (3)

• Let $G = (V, E)$ be an input graph
• Using Adjacency List representation:
  • Space: $O(|V| + |E|)$
    ➔ Excellent when $|E|$ is small
  • Easy to list all neighbors of a vertex
  • Takes $O(|V|)$ time to check if a vertex $u$ is a neighbor of a vertex $v$
• can also represent weighted graph
Adjacency Matrix (1)

- Use a $|V| \times |V|$ matrix $A$ such that
  
  $A(u,v) = 1$ if $(u,v)$ is an edge
  
  $A(u,v) = 0$ otherwise

$$
\begin{array}{c|ccccc}
  & 1 & 2 & 3 & 4 & 5 \\
  \hline
  1 & 0 & 1 & 0 & 0 & 1 \\
  2 & 1 & 0 & 1 & 0 & 0 \\
  3 & 0 & 1 & 0 & 1 & 1 \\
  4 & 0 & 0 & 1 & 0 & 1 \\
  5 & 1 & 0 & 1 & 1 & 0 \\
\end{array}
$$
### Adjacency Matrix (2)

- Use a $|V| \times |V|$ matrix $A$ such that:
  
  $A(u,v) = 1$ if $(u,v)$ is an edge
  
  $A(u,v) = 0$ otherwise

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Adjacency Matrix (3)

• Let $G = (V, E)$ be an input graph

• Using Adjacency Matrix representation:
  • Space: $O(|V|^2)$
    ➔ Bad when $|E|$ is small
  • $O(1)$ time to check if a vertex $u$ is a neighbor of a vertex $v$
  • $\Theta(|V|)$ time to list all neighbors

• can also represent weighted graph
Transpose of a Matrix

• Let $A$ be an $n \times m$ matrix

**Definition:**

The transpose of $A$, denoted by $A^T$, is an $m \times n$ matrix such that

$$A^T(u,v) = A(v,u) \quad \text{for every } u, v$$

† If $A$ is an adjacency matrix of an undirected graph, then $A = A^T$
Breadth First Search (BFS)

• A simple algorithm to find all vertices reachable from a particular vertex \( s \)
  • \( s \) is called source vertex

• Idea: Explore vertices in rounds
  • At Round \( k \), visit all vertices whose shortest distance (\#edges) from \( s \) is \( k-1 \)
  • Also, discover all vertices whose shortest distance from \( s \) is \( k \)
The BFS Algorithm

1. Mark $s$ as discovered in Round 0
2. For Round $k = 1, 2, 3, ...,$
   For (each $u$ discovered in Round $k-1$)
   {
     Mark $u$ as visited;
     Visit each neighbor $v$ of $u$;
     If ($v$ not visited and not discovered)
     Mark $v$ as discovered in Round $k$;
   }

Stop if no vertices were discovered in Round $k-1$
Example \((s = \text{source})\)

- **Visited** \((? = \text{discover time})\)
- **Discovered** \((? = \text{discover time})\)
- Direction of edge when new node is discovered
Example ($s = \text{source}$)

- **visited**: $r$, $s$, $t$, $u$ ($\uparrow$ = discover time)
- **discovered**: $v$, $w$, $x$, $y$ ($\uparrow$ = discover time)
- **direction of edge when new node is discovered**
Example ($s = \text{source}$)

- Visited node: $\text{visited}$ ( $? = \text{discover time} $)
- Discovered node: $\text{discovered}$ ( $? = \text{discover time} $)
- Direction of edge when new node is discovered:

$$
\begin{align*}
\text{r} & \quad \text{s} & \quad \text{t} & \quad \text{u} \\
1 & \quad 0 & \quad 2 & \quad 3 \\
2 & \quad 1 & \quad 1 & \quad 3 \\
\text{v} & \quad \text{w} & \quad \text{x} & \quad \text{y} \\
\end{align*}
$$
Example \((s = \text{source})\)

The directed edges form a tree that contains all nodes reachable from \(s\)

Called \textbf{BFS tree of} \(s\)

Done when no new node is discovered
Correctness

• The correctness of BFS follows from the following theorem:

Theorem: A vertex $v$ is discovered in Round $k$ if and only if shortest distance of $v$ from source $s$ is $k$

Proof: By induction
Performance

• BFS algorithm is easily done if we use
  • an $O(|V|)$-size array to store discovered/visited information
  • a separate list for each round to store the vertices discovered in that round
• Since no vertex is discovered twice, and each edge is visited at most twice (why?)
  ➔ Total time: $O(|V|+|E|)$
  ➔ Total space: $O(|V|+|E|)$
Performance (2)

- Instead of using a separate list for each round, we can use a common queue
  - When a vertex is discovered, we put it at the end of the queue
  - To pick a vertex to visit in Step 2, we pick the one at the front of the queue
  - Done when no vertex is in the queue

⇒ No improvement in time/space ...
⇒ But algorithm is simplified

Question: Can you prove the correctness of using queue?