Algorithms and Data Structures for Data Science
Graph Implementations 3

CS 277
Brad Solomon

April 17, 2023

Department of Computer Science
Exam 3 Signups Available

April 24 — April 27

Very limited window for makeup exams (since end of semester is near)

Covers content from week 10 — 14
Learning Objectives

Review adjacency matrix graph implementations

Introduce adjacency list implementation

Discuss the strengths and weaknesses of each implementation
Graph Implementation: Adjacency Matrix

**Vertex Storage:**
A dictionary (or list) of vertices
Also stores the indexing of the matrix

**Edge Storage:**
A $|V| \times |V|$ matrix storing edges
$e[r][c] = 1$ if there is an edge between $r$ and $c$
Graph Implementation: Adjacency Matrix

Pros:
- Fast lookup and modification of edges
- Easily extended to weighted and directed

Cons:
- Slow to add or remove new vertices
- Getting the degree of nodes is slow
- Storage costs are relatively large
Adjacency List

Vertex Storage:

Edge Storage:
Adjacency List

Vertex Storage:

Edge Storage:
Adjacency List

getVertices():

u
v
w
z

u
v
w
z

v
w

u
w

v
u
z

z
Adjacency List

getEdges(v):

```
Adjacency List

u -> v, w
v -> u, w
w -> v, u
z
```
Adjacency List

areAdjacent(u, v):

\begin{itemize}
  \item d=2
  \item d=2
  \item d=3
  \item d=1
\end{itemize}
Adjacency List

insertVertex(v):

- u
- v
- w
- z

- d=2
- d=2
- d=3
- d=1
Adjacency List

removeVertex(v):
Adjacency List

$\text{insertEdge}(u, v)$:

- $d=2$
- $d=1$
- $d=3$

Graph:

- $u$ is connected to $v$ and $w$.
- $v$ is connected to $w$ and $z$.
- $w$ is connected to $v$ and $u$.
- $z$ is connected to $z$.

Vertices and their degrees are:

- $u$: $d=2$
- $v$: $d=2$
- $w$: $d=3$
- $z$: $d=1$
Adjacency List

removeEdge(u, v):

```
Adjacency List
```

```
\begin{itemize}
  \item \text{u} \to \text{v} \quad \text{d=2}
  \item \text{v} \to \text{u} \quad \text{d=2}
  \item \text{w} \to \text{u} \quad \text{d=3}
  \item \text{z} \to \text{z} \quad \text{d=1}
\end{itemize}
```

```
\begin{itemize}
  \item \text{v} \to \text{w}
  \item \text{u} \to \text{w}
  \item \text{v} \to \text{u}
  \item \text{u} \to \text{z}
\end{itemize}
```
Adjacency List

Pros:

Cons:
Adjacency List

How would our data structure change if...

Edges are directed:
Adjacency List

How would our data structure change if...

Edges are weighted:
Expressed as $O(f)$ | Edge List | Adjacency Matrix | Adjacency List |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Space</strong></td>
<td>$n+m$</td>
<td>$n^2$</td>
<td>$n+m$</td>
</tr>
<tr>
<td>$\text{insertVertex}(v)$</td>
<td>$1^*$</td>
<td>$n^*$</td>
<td>$1^*$</td>
</tr>
<tr>
<td>$\text{removeVertex}(v)$</td>
<td>$m^{**}$</td>
<td>$n$</td>
<td>$\text{deg}(v)^{***}$</td>
</tr>
<tr>
<td>$\text{insertEdge}(u, v)$</td>
<td>$1$</td>
<td>$1$</td>
<td>$1^*$</td>
</tr>
<tr>
<td>$\text{removeEdge}(u, v)$</td>
<td>$m$</td>
<td>$1$</td>
<td>$\text{min}( \text{deg}(u), \text{deg}(v) )$</td>
</tr>
<tr>
<td>$\text{getEdges}(v)$</td>
<td>$m$</td>
<td>$n$</td>
<td>$\text{deg}(v)$</td>
</tr>
<tr>
<td>$\text{areAdjacent}(u, v)$</td>
<td>$m$</td>
<td>$1$</td>
<td>$\text{min}( \text{deg}(u), \text{deg}(v) )$</td>
</tr>
</tbody>
</table>

$|V|= n, |E|= m$
Graph Traversals

There is no clear order in a graph (even less than a tree!)

How can we systematically go through a complex graph in the fewest steps?

Tree traversals won’t work — let’s compare:

- Rooted
- Acyclic

-
Traversal: BFS
Traversal: BFS

### Adjacent Edges

<table>
<thead>
<tr>
<th>v</th>
<th>d</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Diagram:**

- A is connected to B, C, and D.
- B is connected to C and G.
- C is connected to D and E.
- D is connected to E and F.
- E is connected to F and G.
- F is connected to G and H.
- G is connected to H.
- H is connected to E.
Traversals: BFS

<table>
<thead>
<tr>
<th>v</th>
<th>d</th>
<th>P</th>
<th>Adjacent Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>-</td>
<td>B C D</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>A</td>
<td>A C E</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>A</td>
<td>A B D E F</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>A</td>
<td>A C F H</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>B</td>
<td>B C G</td>
</tr>
<tr>
<td>F</td>
<td>2</td>
<td>C</td>
<td>C D G</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>E</td>
<td>E F H</td>
</tr>
<tr>
<td>H</td>
<td>2</td>
<td>D</td>
<td>D G</td>
</tr>
</tbody>
</table>
Traversal: BFS

<table>
<thead>
<tr>
<th>v</th>
<th>d</th>
<th>P</th>
<th>Adjacent Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>-</td>
<td>C B D</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>A</td>
<td>A C E</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>A</td>
<td>A B D E F</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>A</td>
<td>A C F H</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>C</td>
<td>B C G</td>
</tr>
<tr>
<td>F</td>
<td>2</td>
<td>C</td>
<td>C D G</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>E</td>
<td>E F H</td>
</tr>
<tr>
<td>H</td>
<td>2</td>
<td>D</td>
<td>D G</td>
</tr>
</tbody>
</table>

G H F E D B C A
Running time of BFS

<table>
<thead>
<tr>
<th>v</th>
<th>d</th>
<th>P</th>
<th>Adjacent Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>-</td>
<td>C B D</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>A</td>
<td>A C E</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>A</td>
<td>B A D E F</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>A</td>
<td>A C F H</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>C</td>
<td>B C G</td>
</tr>
<tr>
<td>F</td>
<td>2</td>
<td>C</td>
<td>C D G</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>E</td>
<td>E F H</td>
</tr>
<tr>
<td>H</td>
<td>2</td>
<td>D</td>
<td>D G</td>
</tr>
</tbody>
</table>

G H F E D B C A
BFS Observations

What is the shortest path from $A$ to $H$?

What is the shortest path from $E$ to $H$?

If my node has distance $d$, do I know anything about the nodes connected by a cross edge?
BFS Observations

BFS can be used to detect cycles

The value of $d$ in BFS is the shortest distance from source to every vertex

In BFS, the endpoints of a cross edge never differ in distance, $d$, by more than 1. In other words for vertices $u$ and $v$ connected by a cross edge: $\left( |d(u) - d(v)| \leq 1 \right)$