CSE 331: Algorithms and Data Structures

Recitation session
10/18/2016
Recitation session

• Review, Q & A
  • Hashing
  • Graph
  • Breath First Search
  • Depth First Search
  • Topological sort
Review

• Chaining: each item in the hash table is a list for the elements having the same hash value. The expected search time is $O(1 + \alpha)$, $\alpha = \frac{n}{m}$

• Open address: $m > n$, collision: update $h(x,i)$, Linear Probing $h(x, i) = (h'(x) + i)\mod m$ and Double Hashing $h(x, i) = (h'_1(x) + ih'_2(x))\mod m$. 
Hashing

• Q1

Consider inserting the keys 10, 22, 31, 4, 15, 28, 17 into a hash table of length $m = 11$ using open addressing. Illustrate the result of inserting these keys using linear probing ($h_1(x) = x$) and using double hashing with $h_1(x) = x$ and $h_2(x) = 1 + (x \mod (m - 1))$. 
Hashing

- A1 (Linear probing)

\[ h(x, i) = (h_1(x) + i) \mod m = (x + i) \mod 11, \quad i = 0, \ldots, 10 \]

\[ x \]
\[ 10, 22, 31, 4, 15, 28, 17 \]
Hashing

- A1 (Linear probing)

\[ h(x, i) = (x + i) \mod 11 \]

Hash table \( T \)

\[ h(10, 0) = (10 + 0) \mod 11 = 10 \Rightarrow T[10] = 10 \]
### Hashing

**A1 (Linear probing)**

The hash table $T$ is defined as:

$$ h(x, i) = (x + i) \mod 11 $$

Given $x = 22$, the hash function $h(22, 0) = (22 + 0) \mod 11 = 0 \Rightarrow T[0] = 22$.
Hashing

- **A1 (Linear probing)**

  \[ h(x, i) = (x + i) \mod 11 \]

  Hash table \( T \)

  \[
  \begin{array}{c}
  22 \\
  \vdots \\
  31 \\
  10 \\
  \end{array}
  \]

  \[ h(31, 0) = (31 + 0) \mod 11 = 9 \Rightarrow T[9] = 31 \]
### Hashing

- **A1 (Linear probing)**

  \[ h(x, i) = (x + i) \mod 11 \]

  Hash table \( T \)

<table>
<thead>
<tr>
<th>( x )</th>
<th>( h(x, 0) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>22</td>
<td>3</td>
</tr>
<tr>
<td>31</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>15</td>
<td>8</td>
</tr>
<tr>
<td>28</td>
<td>7</td>
</tr>
<tr>
<td>17</td>
<td>6</td>
</tr>
</tbody>
</table>

*Example:*

\[ h(4, 0) = (4 + 0) \mod 11 = 4 \Rightarrow T[4] = 4 \]
Hashing

• A1 (Linear probing)

\[ h(x, i) = (x + i) \mod 11 \]

Hash table \( T \)

\[
\begin{array}{c}
\hline
22 \\
\hline
\hline
4 \\
\hline
\hline
31 \\
\hline
10 \\
\hline
\end{array}
\]

\[ h(15, 0) = (15 + 0) \mod 11 = 4 \]
Hashing

- A1 (Linear probing)

Hash function: \( h(x, i) = (x + i) \mod 11 \)

Hash table \( T \):

\[
\begin{array}{c}
22 \\
4 \\
15 \\
31 \\
10 \\
\end{array}
\]

\( h(15,1) = (15 + 1) \mod 11 = 5 \Rightarrow T[5] = 15 \)
Hashing

- A1 (Linear probing)

\[ h(x, i) = (x + i) \mod 11 \]

10, 22, 31, 4, 15, 28, 17

\[ h(28, 0) = (28 + 0) \mod 11 = 6 \Rightarrow T[6] = 28 \]
Hashing

- A1 (Linear probing)

**Hash table T**

\[ h(x, i) = (x + i) \mod 11 \]

\[ 17, 0 = 17 + 0 \mod 11 = 6 \]

\[ \begin{array}{c}
22 \\
4 \\
15 \\
28 \\
31 \\
10 \\
\end{array} \]

\[ 10, 22, 31, 4, 15, 28, 17 \]
Hashing

• A1 (Linear probing)

$h(x, i) = (x + i) \mod 11$

10, 22, 31, 4, 15, 28, 17

Hash table $T$

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>15</td>
</tr>
<tr>
<td>28</td>
</tr>
<tr>
<td>17</td>
</tr>
<tr>
<td>31</td>
</tr>
<tr>
<td>10</td>
</tr>
</tbody>
</table>
Hashing

• A1 (Double hashing)

\[ h(x, i) = (h_1(x) + ih_2(x)) \mod m = (x + i(1 + x \mod 10)) \mod 11, \quad i = 0, \ldots, 10 \]
Hashing

- A1 (Double hashing)  
  \[ h(x, i) = (x + i(1 + x \mod 10)) \mod 11 \]

<table>
<thead>
<tr>
<th>( x )</th>
</tr>
</thead>
<tbody>
<tr>
<td>10, 22, 31, 4, 15, 28, 17</td>
</tr>
</tbody>
</table>

\[ h(10, 0) = (10 + 0 \times \ldots) \mod 11 = 10 \Rightarrow T[10] = 10 \]
Hashing

A1 (Double hashing)

\[ h(x, i) = (x + i(1 + x \mod 10)) \mod 11 \]

Hash table \( T \)

\( h(22, 0) = (10 + 0 \times \cdots) \mod 11 = 0 \Rightarrow T[10] = 10 \)
# Hashing

## A1 (Double hashing)

### Hash table $T$

$$h(x, i) = (x + i(1 + x \mod 10)) \mod 11$$

$\{10, 22, 31, 4, 15, 28, 17\}$

<table>
<thead>
<tr>
<th>$x$</th>
<th>$h(x, 0)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>$= (31 + 0 \times \cdots) \mod 11 = 0 \Rightarrow T[9] = 31$</td>
</tr>
</tbody>
</table>
Hashing

- **A1 (Double hashing)**

  \[
  h(x, i) = (x + i(1 + x \mod 10)) \mod 11
  \]

  \[
  10, 22, 31, 4, 15, 28, 17
  \]

### Hash table \( T \)

<table>
<thead>
<tr>
<th>Index</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>( h(4, 0) = (4 + 0 \times \ldots) \mod 11 = 4 \Rightarrow T[4] = 4 )</td>
</tr>
<tr>
<td>31</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>
Hashing

- A1 (Double hashing)  
  \[ h(x, i) = (x + i(1 + x \mod 10)) \mod 11 \]

Hash table \( T \):

\[
\begin{array}{c}
22 \\
\hline
4 \\
\hline
31 \\
\hline
10
\end{array}
\]

\[ h(15,0) = (15 + 0 \times \cdots) \mod 11 = 4 \]
Hashing

- A1 (Double hashing)  \( h(x, i) = (x + i(1 + x \mod 10)) \mod 11 \)

\[ \begin{array}{|c|}
\hline
22 \\
\hline
4 \\
\hline
31 \\
\hline
10 \\
\hline
\end{array} \]

\( h(15, 1) = (15 + 1 \times (1 + 15 \mod 10)) \mod 11 = 10 \)

10, 22, 31, 4, 15, 28, 17
Hashing

\[ h(x, i) = (x + i(1 + x \mod 10)) \mod 11 \]

A1 (Double hashing)  

**Hash table** \( T \)

\[
\begin{align*}
15 \\
15 \times 2 \\
15 \times 3 \\
15 \times 4 \\
15 \times 5 \\
15 \times 6 \\
15 \times 7 \\
15 \times 8 \\
15 \times 9 \\
15 \times 10
\end{align*}
\]

\[
\rightarrow T[5] = 15
\]

\[
10, 22, 31, 4, 15, 28, 17
\]
Hashing

\[ h(x, i) = (x + i(1 + x \mod 10)) \mod 11 \]

- A1 (Double hashing)  
  \[ h(28, 0) = (28 + 0 \times \cdots) \mod 11 = 6 \Rightarrow T[6] = 28 \]

Hash table \( T \):

- 22
- 4
- 15
- 28
- 31
- 10

10, 22, 31, 4, 15, 28, 17
Hashing

- A1 (Double hashing)

\[
h(x, i) = (x + i(1 + x \mod 10)) \mod 11 \hspace{1cm} 10, 22, 31, 4, 15, 28, 17
\]

\[
\begin{array}{c}
22 \\
4 \\
15 \\
28 \\
31 \\
10
\end{array}
\]

\[
h(17, 0) = (17 + 0 \times \cdots) \mod 11 = 6
\]
Hashing

\[ h(x, i) = (x + i(1 + x \mod 10)) \mod 11 \]

\[ 10, 22, 31, 4, 15, 28, 17 \]

• A1 (Double hashing)  Hash table \( T \)

\[
\begin{array}{c}
22 \\
\hline
17 \\
\hline
4 \\
\hline
15 \\
\hline
28 \\
\hline
31 \\
\hline
10
\end{array}
\]

\[ h(17, 1) = (17 + 1 \times (1 + 17 \mod 10)) \mod 11 \]
\[ = 3 \Rightarrow T[3] = 17 \]
Hashing

- **Q2**

*Quadratic probing:* \( h(x, i) = (h_1(x) + c_1 i + c_2 i^2) \mod m, i = 0, \ldots, m - 1 \)

\[
\begin{align*}
    c_1 &= c_2 = \frac{1}{2} \\
    h_1(x) &= x \mod 8
\end{align*}
\]

**Hash table** \( T \)

<table>
<thead>
<tr>
<th>Slot 0:</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Entries: \( 29, 13, 4, 12, 61 \)
Hashing

• A2

\[ h(x, i) = (x \mod 8 + \frac{1}{2} (i + i^2)) \mod 8 \]

<table>
<thead>
<tr>
<th>Slot 0:</th>
<th>1:</th>
<th>2:</th>
<th>3:</th>
<th>4:</th>
<th>5:</th>
<th>6:</th>
<th>7:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>29</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ h(29,0) = \left( 29 \mod 8 + \frac{1}{2} (0 + 0^2) \right) \mod 8 = 5 \Rightarrow T[5] = 29 \]
Hashing

- A2

\[ h(x, i) = (x \mod 8 + \frac{1}{2}(i + i^2)) \mod 8 \]

Hash table \( T \)

<table>
<thead>
<tr>
<th>Slot 0:</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5:</td>
<td>29</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( h(13,0) = \left( 13 \mod 8 + \frac{1}{2}(0 + 0^2) \right) \mod 8 = 5 \)
Hashing

• A2

\[ h(x, i) = (x \mod 8 + \frac{1}{2}(i + i^2)) \mod 8 \]

\[
\begin{array}{c|c}
\text{Slot 0:} & \\
1: & \\
2: & \\
3: & \\
4: & \\
5: & 29 \\
6: & 13 \\
7: & \\
\end{array}
\]

\[ h(13, 1) = \left(13 \mod 8 + \frac{1}{2}(1 + 1^2)\right) \mod 8 = 6 \]
Hashing

- A2

\[ h(x, i) = (x \mod 8 + \frac{1}{2}(i + i^2)) \mod 8 \]

Hash table T

<table>
<thead>
<tr>
<th>Slot 0:</th>
<th>1:</th>
<th>2:</th>
<th>3:</th>
<th>4:</th>
<th>5:</th>
<th>6:</th>
<th>7:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td>29</td>
<td></td>
</tr>
</tbody>
</table>

\[ h(4, 0) = \left(4 \mod 8 + \frac{1}{2}(0 + 0^2)\right) \mod 8 = 4 \]
Hashing

• A2

\[ h(x, i) = (x \mod 8 + \frac{1}{2}(i + i^2)) \mod 8 \]

\[
\begin{array}{c|c|c|c|c|c|c|c}
\hline
\end{array}
\]

\[ h(12, 0) = \left(12 \mod 8 + \frac{1}{2}(0 + 0^2)\right) \mod 8 = 4 \]
Hashing

• A2

$h(x, i) = (x \mod 8 + \frac{1}{2}(i + i^2)) \mod 8$

|x| 29, 13, 4, 12, 61

<table>
<thead>
<tr>
<th>Slot 0:</th>
<th>1:</th>
<th>2:</th>
<th>3:</th>
<th>4:</th>
<th>5:</th>
<th>6:</th>
<th>7:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td>29</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>13</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$h(12, 1) = \left(12 \mod 8 + \frac{1}{2}(1 + 1^2)\right) \mod 8 = 5$
Hashing

• A2

\[ h(x, i) = (x \mod 8 + \frac{1}{2} (i + i^2)) \mod 8 \]

**Hash table T**

<table>
<thead>
<tr>
<th>Slot</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>29</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
</tr>
<tr>
<td>7</td>
<td>12</td>
</tr>
</tbody>
</table>

\[ h(12, 2) = \left(12 \mod 8 + \frac{1}{2} (2 + 2^2)\right) \mod 8 = 7 \]
Hashing

- A2

\[ h(x, i) = (x \mod 8 + \frac{1}{2}(i + i^2)) \mod 8 \]

<table>
<thead>
<tr>
<th>Slot 0:</th>
<th>1:</th>
<th>2:</th>
<th>3:</th>
<th>4:</th>
<th>5:</th>
<th>6:</th>
<th>7:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>29</td>
<td>13</td>
<td>12</td>
</tr>
</tbody>
</table>

\[ h(61,0) = \left( 61 \mod 8 + \frac{1}{2}(0 + 0^2) \right) \mod 8 = 5 \]
Hashing

- A2

Hashing function:

\[ h(x, i) = (x \mod 8 + \frac{1}{2}(i + i^2)) \mod 8 \]

**Hash table**:

<table>
<thead>
<tr>
<th>Slot 0:</th>
<th>1:</th>
<th>2:</th>
<th>3:</th>
<th>4:</th>
<th>5:</th>
<th>6:</th>
<th>7:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>29</td>
<td>13</td>
<td>12</td>
</tr>
</tbody>
</table>

**Example**:

\[ h(61,1) = \left( 61 \mod 8 + \frac{1}{2}(1 + 1^2) \right) \mod 8 = 6 \]
Hashing

• A2

\[ h(x, i) = (x \mod 8 + \frac{1}{2}(i + i^2)) \mod 8 \]

Hash table \( T \)

<table>
<thead>
<tr>
<th>Slot 0:</th>
<th>61</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td></td>
</tr>
<tr>
<td>2:</td>
<td></td>
</tr>
<tr>
<td>3:</td>
<td></td>
</tr>
<tr>
<td>4:</td>
<td>4</td>
</tr>
<tr>
<td>5:</td>
<td>29</td>
</tr>
<tr>
<td>6:</td>
<td>13</td>
</tr>
<tr>
<td>7:</td>
<td>12</td>
</tr>
</tbody>
</table>

\( h(61,2) = \left( 61 \mod 8 + \frac{1}{2}(2 + 2^2) \right) \mod 8 = 0 \)
A graph $G = (V, E)$ consists of a vertex set $V$ and an edge set $E$. $|V| = n$ and $|E| = m$.

Each edge $e = (u, v) \in E$ is an unordered pair of vertices.

If $(u, v) \in E$, we say $v$ is a neighbor of $u$.

The degree $\text{deg}(u)$ of a vertex $u$ is the number of edges incident to $u$. 
Review

Definition

- If the two end vertices of $e$ are ordered, the edge is directed, and we write $e = u \rightarrow v$.
- If all edges are directed, then $G$ is a directed graph.
- The in-degree $\text{deg}_\text{in}(u)$ of a vertex $u$ is the number of edges that are directed into $u$.
- The out-degree $\text{deg}_\text{out}(u)$ of a vertex $u$ is the number of edges that are directed from $u$. 
Review

**Adjacency List Representation**

- For each vertex \( v \in V \), there's a linked list \( Adj[v] \). Each entry of \( Adj[v] \) is a vertex \( w \) such that \( (v, w) \in E \).

- If there are other information associated with the edges (such as edge weight), they can be stored in the entries of the adjacency list.

- For undirected graphs, each edge \( e = (u, v) \) has two entries in this representation, one in \( Adj[u] \) and one in \( Adj[v] \).

- The Adjacency List Representation for directed graphs is similar. For each edge \( e = \vec{u}v \), there is an entry in \( Adj[u] \).

- For directed graphs, each edge has only one entry in the representation.
Review

\[ \text{BFS}(G, s) \]

1. \( Q \leftarrow \emptyset \)
2. \textbf{for each} \( u \in V - \{s\} \) \textbf{do}
3. \hspace{1em} \( \pi[u] = \text{NIL}; \quad d[u] = \infty; \quad \text{color}[u] = \text{white} \)
4. \hspace{1em} \( d[s] = 0; \quad \text{color}[s] = \text{grey}; \quad \pi[s] = \text{NIL} \)
5. \( \text{Enqueue}(Q, s) \)
6. \textbf{while} \( Q \neq \emptyset \) \textbf{do}
7. \hspace{1em} \( u \leftarrow \text{Dequeue}(Q) \)
8. \hspace{1em} \textbf{for each} \( v \in \text{Adj}[u] \) \textbf{do}
9. \hspace{2em} \textbf{if} \( \text{color}[v] = \text{white} \)
10. \hspace{3em} \textbf{then} \( \text{color}[v] = \text{grey}; \quad d[v] \leftarrow d[u] + 1; \quad \pi[v] \leftarrow u; \quad \text{Enqueue}(Q, v) \)
11. \hspace{1em} \text{color}[u] = \text{black} \)
Breadth First Search

• Q1:

Adjacency list

Breadth first search

Spanning tree
Breadth First Search

• A1:

Adjacency list:
Breadth First Search

• A1:

Queue:

Result:
Breadth First Search

• A1:

Queue:

Result: A
Breadth First Search

- A1
Breadth First Search

• A1:

Queue: 

D G E F

Result: A B
Breadth First Search

• A1:

Queue:

D G E F

Result: A B
Breadth First Search

• A1:

Queue:

G E F

Result: A B D
Breadth First Search

- A1:

Queue:

Result: A B D G E
Breadth First Search

• A1:

Queue:

Result: A B D G E F
Breadth First Search

• A1:

Queue:

Result: A B D G E F
Breadth First Search

• A1:

Queue:

Result: A B D G E F C
Breadth First Search

• A1:

Queue:

Result: A B D G E F C H
Breadth First Search

• A1:

Result: A B D G E F C H
Breadth First Search

• Q2:

**T  F** Let $T$ be a complete binary tree with $n$ nodes. Finding a path from the root of $T$ to a given vertex $v \in T$ using breadth-first search takes $O(\lg n)$ time.
Breadth First Search

• A2:

T  F  Let $T$ be a complete binary tree with $n$ nodes. Finding a path from the root of $T$ to a given vertex $v \in T$ using breadth-first search takes $O(\lg n)$ time.

Solution: False. Breadth-first search requires $\Omega(n)$ time. Breadth-first search examines each node in the tree in breadth-first order. The vertex $v$ could well be the last vertex explored. (Also, notice that $T$ is not necessarily sorted.)
Review

- DFS: Recursive algorithm

DFS(G)

1. for each vertex $u \in V$ do
2. \hspace{1em} color[u] \leftarrow \text{white}; \quad \pi[u] = \text{NIL}
3. \hspace{1em} time \leftarrow 0
4. for each vertex $u \in V$ do
5. \hspace{2em} if color[u] = \text{white} then DFS-Visit(u)

DFS-Visit(u)

1. color[u] \leftarrow \text{grey}; \quad time \leftarrow time + 1; \quad d[u] \leftarrow time
2. for each vertex $v \in Adj[u]$ do
3. \hspace{2em} if color[v] = \text{white}
4. \hspace{3em} then $\pi[v] \leftarrow u; \quad$ DFS-Visit(v)
5. color[u] \leftarrow \text{black}
6. $f[u] \leftarrow time \leftarrow time + 1$
Depth First Search

• Q1:
Depth First Search

• Q1:

Result: E
Depth First Search

• Q1:

Stack: 

Result: E C
Depth First Search

• Q1:

Stack:

Result: E C D G
Depth First Search

• Q1:

Result: E C D G

Stack:

D
c
E
Depth First Search

• Q1:

Stack: 

Result: E C D G
Depth First Search

• Q1:

Result: E C D G A
Depth First Search

• Q1:

Stack:

Result: E C D G A F
Depth First Search

• Q1:

Stack:

Result: E C D G A F H
Depth First Search

• Q1:

Stack:

Result: E C D G A F H
Depth First Search

Q1:

Depth First Search:

- Stack:
  - A
  - C
  - E

- Result: E C D G A F H
Depth First Search

• Q1:

Stack: B A C E

Result: E C D G A F H B
Depth First Search

• Q1:

Stack:

A
C
E

Result: E C D G A F H B
Depth First Search

• Q1:

Stack:

Result: E C D G A F H B
Depth First Search

- Q1:

Result: E C D G A F H B
Depth First Search

• Q2:

Prove or give a counter example:

(a) In the textbook, we have two routines for graph traversal - $\text{DFS}(G)$ and $\text{BFS}(G,s)$ - where $G$ is a graph and $s$ is any node in $G$. These two procedures will create a DFS tree and a BFS tree respectively. If $G = (V, E)$ is a connected, undirected graph then the height of $\text{DFS}(G)$ tree is always larger than or equal to the height of any of the BFS trees created by $\text{BFS}(G, s)$. 
Depth First Search

- A2:
  
  1. **DFS** ($s = 1$):
  
  2. **BFS** ($s = 2$):
Depth First Search

• Q3:

Prove or give a counter example:

(b) BFS and DFS algorithms on a undirected, connected graph $G = (V, E)$, produce the same tree if and only if $G$ is a tree.
Depth First Search

• A3:

(b) BFS and DFS algorithms on a undirected, connected graph $G = (V, E)$, produce the same tree if and only if $G$ is a tree.

*Answer:* True

*Prove:* DFS tree = BFS tree = $T$ (of $G$) $\Rightarrow$ $G$ is a tree
Depth First Search

• A3:

(b) BFS and DFS algorithms on a undirected, connected graph $G = (V, E)$, produce the same tree if and only if $G$ is a tree.

Prove: DFS tree = BFS tree = $T$ (of $G$) $\Rightarrow G$ is a tree

$T = G \Rightarrow G$ is a tree

$T \neq G \Rightarrow e(u, v) \in G$ and $e(u, v) \notin T$
Depth First Search

• A3:

\[ T \neq G \Rightarrow e(u, v) \in G \text{ and } e(u, v) \notin T \]

**Theorem**

Let \( G \) be an undirected graph, and \( T \) the DFS tree of \( G \) constructed by DFS algorithm. Then there are no cross edges.

\[ e(u, v) \text{ is a back edge} \Rightarrow \text{one of } u \text{ or } v \text{, is ancestor of the other} \]
Depth First Search

• A3:

\[ T \neq G \Rightarrow e(u, v) \in G \text{ and } e(u, v) \notin T \]

**Theorem**

Let \( G = (V, E) \) be a graph. Let \( d[u] \) be the value computed by BFS algorithm. Then for any \( (u, v) \in E \), \( |d[u] - d[v]| \leq 1 \).

\( u \) and \( v \) differ in one level
Depth First Search

• A3:

One of u or v \( (e(u, v)) \), is ancestor of the other

\( u \) and \( v \) differ in one level

\( e(u, v) \in T \)

Contradict with \( e(u, v) \notin T \)

\( T = G \), hence \( G \) is a tree
Depth First Search

• A3:

Prove: $G$ is a tree $\Rightarrow$ DFS tree = BFS tree
Depth First Search

• A3:

\[ G \text{ is a tree } \Leftrightarrow \text{DFS tree } = \text{BFS tree} \]