Proof (completed) 

Q. How many $h$'s cause $x$ and $y$ to collide?

A. There are $m$ choices for each of $a_1, a_2, \ldots, a_r$, but once these are chosen, exactly one choice for $a_0$ causes $x$ and $y$ to collide, namely 

$$
\begin{align*}
&\sum_{i=1}^{r} a_i (x_i - y_i) \\
&= a_0 (x_0 - y_0) \\
&\equiv 0 \pmod{m}.
\end{align*}
$$

Thus, the number of $h$'s that cause $x$ and $y$ to collide is 

$$
m^r \cdot 1 = |H|/m.
$$

October 5, 2005 Copyright © 2001-5 by Erik D. Demaine and Charles E. Leiserson L7.15
Paths in graphs

Consider a digraph $G = (V, E)$ with edge-weight function $w : E \rightarrow \mathbb{R}$. The *weight* of path $p = v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_k$ is defined to be

$$w(p) = \sum_{i=1}^{k-1} w(v_i, v_{i+1}).$$
Proof (completed)

Q. How many $h$'s cause $x$ and $y$ to collide?
A. There are $m$ choices for each of $a_1, a_2, \ldots, a_r$, but once these are chosen, exactly one choice for $a_0$ causes $x$ and $y$ to collide, namely $\left( \begin{array}{c} r \\ 1 \end{array} \right)$.

Thus, the number of $h$'s that cause $x$ and $y$ to collide is $m^r \cdot 1 = m^r = |H|/m$.

October 5, 2005 Copyright © 2001-5 by Erik D. Demaine and Charles E. Leiserson

---

**Paths in graphs**

Consider a digraph $G = (V, E)$ with edge-weight function $w : E \rightarrow \mathbb{R}$. The **weight** of path $p = v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_k$ is defined to be

$$w(p) = \sum_{i=1}^{k-1} w(v_i, v_{i+1}).$$

**Example:**

\[ w(p) = -2 \]
Shortest paths

A \textit{shortest path} from \( u \) to \( v \) is a path of minimum weight from \( u \) to \( v \). The \textit{shortest-path weight} from \( u \) to \( v \) is defined as

\[
\delta(u, v) = \min \{ w(p) : p \text{ is a path from } u \text{ to } v \}.
\]

\textbf{Note:} \( \delta(u, v) = \infty \) if no path from \( u \) to \( v \) exists.
Optimal substructure

Theorem. A subpath of a shortest path is a shortest path.
Optimal substructure

**Theorem.** A subpath of a shortest path is a shortest path.

**Proof.** Cut and paste:
Optimal substructure

**Theorem.** A subpath of a shortest path is a shortest path.

**Proof.** Cut and paste:
Triangle inequality

**Theorem.** For all $u, v, x \in V$, we have
\[ \delta(u, v) \leq \delta(u, x) + \delta(x, v). \]
**Triangle inequality**

**Theorem.** For all $u, v, x \in V$, we have
\[ \delta(u, v) \leq \delta(u, x) + \delta(x, v). \]

**Proof.**
Well-definedness of shortest paths

If a graph $G$ contains a negative-weight cycle, then some shortest paths may not exist.
Well-definedness of shortest paths

If a graph $G$ contains a negative-weight cycle, then some shortest paths may not exist.

Example:
Single-source shortest paths

**Problem.** From a given source vertex \( s \in V \), find the shortest-path weights \( \delta(s, v) \) for all \( v \in V \).

If all edge weights \( w(u, v) \) are nonnegative, all shortest-path weights must exist.

**Idea:** Greedy.

1. Maintain a set \( S \) of vertices whose shortest-path distances from \( s \) are known.
2. At each step add to \( S \) the vertex \( v \in V - S \) whose distance estimate from \( s \) is minimal.
3. Update the distance estimates of vertices adjacent to \( v \).
Dijkstra’s algorithm

\[
d[s] \leftarrow 0
\]

for each \( v \in V - \{s\} \)
\[
\text{do } d[v] \leftarrow \infty
\]

\( S \leftarrow \emptyset \)

\( Q \leftarrow V \) \hspace{1cm} \triangleright \text{ } Q \text{ is a priority queue maintaining } V - S
Dijkstra’s algorithm

\[ d[s] \leftarrow 0 \]
\[ \text{for each } v \in V - \{s\} \]
\[ \quad \text{do } d[v] \leftarrow \infty \]
\[ S \leftarrow \emptyset \]
\[ Q \leftarrow V \quad \triangleright Q \text{ is a priority queue maintaining } V - S \]
\[ \text{while } Q \neq \emptyset \]
\[ \quad \text{do } u \leftarrow \text{Extract-Min}(Q) \]
\[ S \leftarrow S \cup \{u\} \]
\[ \text{for each } v \in Adj[u] \]
\[ \quad \text{do if } d[v] > d[u] + w(u, v) \]
\[ \quad \text{then } d[v] \leftarrow d[u] + w(u, v) \]
Dijkstra’s algorithm

\[
\begin{align*}
d[s] & \leftarrow 0 \\
\text{for each } v \in V - \{s\} & \text{ do } d[v] \leftarrow \infty \\
S & \leftarrow \emptyset \\
Q & \leftarrow V \quad \triangleright Q \text{ is a priority queue maintaining } V - S \\
\text{while } Q \neq \emptyset & \text{ do } u \leftarrow \text{EXTRACT-MIN}(Q) \\
& \quad S \leftarrow S \cup \{u\} \\
& \quad \text{for each } v \in \text{Adj}[u] \\
& \quad \text{do if } d[v] > d[u] + w(u, v) \\
& \quad \quad \text{then } d[v] \leftarrow d[u] + w(u, v) \\
\end{align*}
\]

relaxation step

Implicit DECREASE-KEY
Q. How many $h$'s cause $x$ and $y$ to collide?

A. There are $m$ choices for each of $a_1, a_2, \ldots, a_r$, but once these are chosen, exactly one choice for $a_0$ causes $x$ and $y$ to collide, namely

$$\begin{vmatrix}
\hat{a}_1 - \hat{a}_2 \\
\hat{a}_2 - \hat{a}_3 \\
\vdots \\
\hat{a}_r - \hat{a}_0
\end{vmatrix} = -a_i (x_i - y_i) \mod m.$$ 

Thus, the number of $h$'s that cause $x$ and $y$ to collide is $m r = |H|/m$. 

October 5, 2005
Copyright © 2001-5 by Erik D. Demaine and Charles E. Leiserson

**Example of Dijkstra’s algorithm**

Graph with nonnegative edge weights:
Proof (completed)

Q. How many h's cause x and y to collide?
A. There are choices for each of a_1, a_2, …, a_r, but once these are chosen, exactly one choice for a_0 causes x and y to collide, namely

Thus, the number of h's that cause x and y to collide is

October 5, 2005
Copyright © 2001-5 by Erik D. Demaine and Charles E. Leiserson

Example of Dijkstra’s algorithm

Initialize:

<table>
<thead>
<tr>
<th>Q:</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
</tbody>
</table>

S: {}
Proof (completed)

Q. How many h's cause x and y to collide?
A. There are m choices for each of a_1, a_2, …, a_r, but once these are chosen, exactly one choice for a_0 causes x and y to collide, namely

\[
\begin{pmatrix}
\sum _{i=1}^{r} a_i (x_i - y_i) \\
- a_0
\end{pmatrix}
\mod m.
\]

Thus, the number of h's that cause x and y to collide is m = |H| / m.

October 5, 2005
Copyright © 2001-5 by Erik D. Demaine and Charles E. Leiserson

Example of Dijkstra’s algorithm

“A” ← EXTRACT-MIN(Q):

Q: A B C D E
0 ∞ ∞ ∞ ∞ ∞

S: \{ A \}
**Example of Dijkstra's algorithm**

**Q:** How many \( h \)'s cause \( x \) and \( y \) to collide?

**A:** There are \( m \) choices for each of \( a_1, a_2, \ldots, a_r \), but once these are chosen, exactly one choice for \( a_0 \) causes \( x \) and \( y \) to collide, namely:

\[
\begin{align*}
\begin{bmatrix}
 r \\
 \sum_{i=1}^{r} \\
 \end{bmatrix} \\
\cdot \\
\begin{bmatrix}
 a_i \\
 (x_i - y_i) \\
\end{bmatrix}
\end{align*}
\]

\( \mod m \).

Thus, the number of \( h \)'s that cause \( x \) and \( y \) to collide is \( m^r \cdot 1 = m^r = |H|/m \).

October 5, 2005

Copyright © 2001-05 by Erik D. Demaine and Charles E. Leiserson

---

**Relax all edges leaving \( A \):**

**Q:**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**S:** \{ A \}
Example of Dijkstra’s algorithm

“C” ← **EXTRACT-MIN(Q):**

$$Q: \begin{array}{cccccc}
A & B & C & D & E \\
0 & \infty & \infty & \infty & \infty \\
10 & 3 & \infty & \infty & \infty \\
\end{array}$$

**S: \{ A, C \}**
Example of Dijkstra’s algorithm

Relax all edges leaving C:

Q: A  B  C  D  E
  0  ∞  ∞  ∞  ∞
  10 3  ∞  ∞
  7 11 5

S: \{ A, C \}
Example of Dijkstra’s algorithm

“E” ← EXTRACT-MIN(Q):

Q: | A   | B | C | D | E |
---|-----|---|---|---|---|
    | 0   | ∞ | ∞ | ∞ | ∞ |
0  | 10  | 3 | ∞ | ∞ | ∞ |
10 | 7   | 11| 5 |

S: { A, C, E }
Example of Dijkstra’s algorithm

Relax all edges leaving $E$:

$Q$: $\begin{array}{cccccc}
A & B & C & D & E \\
\hline
0 & \infty & \infty & \infty & \infty \\
10 & 3 & \infty & \infty & \\
7 & 11 & 5 & \\
7 & 11 & \\
\end{array}$

$S$: \{ A, \ C, \ E \}$
Example of Dijkstra’s algorithm

“B” ← \text{EXTRACT-MIN}(Q):

\begin{align*}
Q: & \quad A & B & C & D & E \\
& 0 & \infty & \infty & \infty & \infty \\
& 10 & 3 & \infty & \infty & \\
& 7 & 11 & 5 & \\
& 7 & 11 & \\
\end{align*}

\begin{align*}
S: \{ A, C, E, B \}
\end{align*}
Example of Dijkstra’s algorithm

Relax all edges leaving B:

\[
\begin{array}{cccccc}
Q: & A & B & C & D & E \\
0 & 0 & \infty & \infty & \infty & \infty \\
10 & 10 & 3 & \infty & \infty & \infty \\
7 & 7 & 11 & 11 & 5 & \\
7 & 7 & & & & \\
\end{array}
\]

\[S: \{A, C, E, B\}\]
Example of Dijkstra’s algorithm

“How many h’s cause x and y to collide?”

A. There are choices for each of ..., but once these are chosen, exactly one choice for causes x and y to collide, namely

\[
\begin{pmatrix}
\sum_i a_i \cdot (x_i - y_i) \mod m
\end{pmatrix}
\]

Thus, the number of h’s that cause x and y to collide is \( m^r \cdot 1 = m^r \cdot 1 = |H|/m. \)

October 5, 2005
Copyright © 2001-5 by Erik D. Demaine and Charles E. Leiserson

“D” ← EXTRACT-MIN(Q):

\[
Q: \begin{array}{cccccc}
A & B & C & D & E \\
0 & \infty & \infty & \infty & \infty \\
10 & 3 & \infty & \infty & \infty \\
7 & 11 & 5 & 11 & 9 \\
\end{array}
\]

\[
S: \{ A, C, E, B, D \}
\]
Correctness — Part I

**Lemma.** Initializing $d[s] \leftarrow 0$ and $d[v] \leftarrow \infty$ for all $v \in V - \{s\}$ establishes $d[v] \geq \delta(s, v)$ for all $v \in V$, and this invariant is maintained over any sequence of relaxation steps.
Correctness — Part I

Lemma. Initializing $d[s] \leftarrow 0$ and $d[v] \leftarrow \infty$ for all $v \in V - \{s\}$ establishes $d[v] \geq \delta(s, v)$ for all $v \in V$, and this invariant is maintained over any sequence of relaxation steps.

Proof. Suppose not. Let $v$ be the first vertex for which $d[v] < \delta(s, v)$, and let $u$ be the vertex that caused $d[v]$ to change: $d[v] = d[u] + w(u, v)$. Then,

\[
\begin{align*}
  d[v] &< \delta(s, v) & \text{supposition} \\
  \leq \delta(s, u) + \delta(u, v) & \text{triangle inequality} \\
  \leq \delta(s, u) + w(u, v) & \text{sh. path} \leq \text{specific path} \\
  \leq d[u] + w(u, v) & v \text{ is first violation}
\end{align*}
\]

Contradiction.
Correctness — Part II

Lemma. Let $u$ be $v$’s predecessor on a shortest path from $s$ to $v$. Then, if $d[u] = \delta(s, u)$ and edge $(u, v)$ is relaxed, we have $d[v] = \delta(s, v)$ after the relaxation.
Correctness — Part II

Lemma. Let $u$ be $v$’s predecessor on a shortest path from $s$ to $v$. Then, if $d[u] = \delta(s, u)$ and edge $(u, v)$ is relaxed, we have $d[v] = \delta(s, v)$ after the relaxation.

Proof. Observe that $\delta(s, v) = \delta(s, u) + w(u, v)$. Suppose that $d[v] > \delta(s, v)$ before the relaxation. (Otherwise, we’re done.) Then, the test $d[v] > d[u] + w(u, v)$ succeeds, because $d[v] > \delta(s, v) = \delta(s, u) + w(u, v) = d[u] + w(u, v)$, and the algorithm sets $d[v] = d[u] + w(u, v) = \delta(s, v)$. 

October 5, 2005
Copyright © 2001-5 by Erik D. Demaine and Charles E. Leiserson
Theorem. Dijkstra’s algorithm terminates with 
\( d[v] = \delta(s, v) \) for all \( v \in V \).
Correctness — Part III

**Theorem.** Dijkstra’s algorithm terminates with \( d[v] = \delta(s, v) \) for all \( v \in V \).

**Proof.** It suffices to show that \( d[v] = \delta(s, v) \) for every \( v \in V \) when \( v \) is added to \( S \). Suppose \( u \) is the first vertex added to \( S \) for which \( d[u] > \delta(s, u) \). Let \( y \) be the first vertex in \( V - S \) along a shortest path from \( s \) to \( u \), and let \( x \) be its predecessor:

\[
S, \text{ just before adding } u.
\]
Correctness — Part III (continued)

Since \( u \) is the first vertex violating the claimed invariant, we have \( d[x] = \delta(s, x) \). When \( x \) was added to \( S \), the edge \( (x, y) \) was relaxed, which implies that \( d[y] = \delta(s, y) \leq \delta(s, u) < d[u] \). But, \( d[u] \leq d[y] \) by our choice of \( u \) as \( \min \) element in \( Q \) in Dijkstra’s algorithm. Contradiction.
Analysis of Dijkstra

while \( Q \neq \emptyset \)
  do \( u \leftarrow \text{Extract-Min}(Q) \)
  \( S \leftarrow S \cup \{u\} \)
  for each \( v \in \text{Adj}[u] \)
    do if \( d[v] > d[u] + w(u, v) \)
      then \( d[v] \leftarrow d[u] + w(u, v) \)
Analysis of Dijkstra's Algorithm

while $Q \neq \emptyset$
  do $u \leftarrow \text{Extract-Min}(Q)$
  $S \leftarrow S \cup \{u\}$
  for each $v \in \text{Adj}[u]$
    do if $d[v] > d[u] + w(u, v)$
      then $d[v] \leftarrow d[u] + w(u, v)$

$|V|$ times
Analysis of Dijkstra

\[ Q \neq \emptyset \]

\[ u \leftarrow \text{Extract-Min}(Q) \]

\[ S \leftarrow S \cup \{u\} \]

for each \( v \in \text{Adj}[u] \)

\[ \text{do if } d[v] > d[u] + w(u, v) \]

\[ \text{then } d[v] \leftarrow d[u] + w(u, v) \]
Analysis of Dijkstra's Algorithm

\begin{align*}
&\text{while } Q \neq \emptyset \\
&\quad \text{do } u \leftarrow \text{EXTRACT-MIN}(Q) \\
&\quad S \leftarrow S \cup \{u\} \\
&\quad \text{for each } v \in \text{Adj}[u] \\
&\quad \quad \text{do if } d[v] > d[u] + w(u, v) \\
&\quad \quad \quad \text{then } d[v] \leftarrow d[u] + w(u, v)
\end{align*}

Handshaking Lemma $\Rightarrow \Theta(E)$ implicit DECREASE-KEY’s.
Analysis of Dijkstra

while $Q \neq \emptyset$
do $u \leftarrow \text{Extract-Min}(Q)$
$S \leftarrow S \cup \{u\}$
for each $v \in \text{Adj}[u]$
do if $d[v] > d[u] + w(u, v)$
then $d[v] \leftarrow d[u] + w(u, v)$

Handshaking Lemma $\Rightarrow \Theta(E)$ implicit $\text{Decrease-Key}$’s.

Time $= \Theta(V \cdot T_{\text{Extract-Min}} + E \cdot T_{\text{Decrease-Key}})$

Note: Same formula as in the analysis of Prim’s minimum spanning tree algorithm.
### Analysis of Dijkstra (continued)

\[
\text{Time} = \Theta(V) \cdot T_{\text{EXTRACT-MIN}} + \Theta(E) \cdot T_{\text{DECREASE-KEY}}
\]

<table>
<thead>
<tr>
<th>Q</th>
<th>( T_{\text{EXTRACT-MIN}} )</th>
<th>( T_{\text{DECREASE-KEY}} )</th>
<th>Total</th>
</tr>
</thead>
</table>

October 5, 2005

Copyright © 2001-5 by Erik D. Demaine and Charles E. Leiserson
Analysis of Dijkstra (continued)

\[
\text{Time} = \Theta(V) \cdot T_{\text{EXTRACT-MIN}} + \Theta(E) \cdot T_{\text{DECREASE-KEY}}
\]

<table>
<thead>
<tr>
<th>Q</th>
<th>(T_{\text{EXTRACT-MIN}})</th>
<th>(T_{\text{DECREASE-KEY}})</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>(O(V))</td>
<td>(O(1))</td>
<td>(O(V^2))</td>
</tr>
</tbody>
</table>
**Analysis of Dijkstra (continued)**

\[
\text{Time} = \Theta(V) \cdot T_{\text{EXTRACT-MIN}} + \Theta(E) \cdot T_{\text{DECREASE-KEY}}
\]

<table>
<thead>
<tr>
<th></th>
<th>( Q )</th>
<th>( T_{\text{EXTRACT-MIN}} )</th>
<th>( T_{\text{DECREASE-KEY}} )</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>( O(V) )</td>
<td>( O(1) )</td>
<td>( O(V^2) )</td>
<td></td>
</tr>
<tr>
<td>binary heap</td>
<td>( O(\log V) )</td>
<td>( O(\log V) )</td>
<td>( O(E \log V) )</td>
<td></td>
</tr>
</tbody>
</table>
### Analysis of Dijkstra's Algorithm (continued)

The time complexity of Dijkstra's algorithm can be expressed as:

$$\text{Time} = \Theta(V) \cdot T_{\text{EXTRACT-MIN}} + \Theta(E) \cdot T_{\text{DECREASE-KEY}}$$

<table>
<thead>
<tr>
<th>Question (Q)</th>
<th>$T_{\text{EXTRACT-MIN}}$</th>
<th>$T_{\text{DECREASE-KEY}}$</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>$O(V)$</td>
<td>$O(1)$</td>
<td>$O(V^2)$</td>
</tr>
<tr>
<td>binary heap</td>
<td>$O(\lg V)$</td>
<td>$O(\lg V)$</td>
<td>$O(E \lg V)$</td>
</tr>
<tr>
<td>Fibonacci heap</td>
<td>$O(\lg V)$</td>
<td>$O(1)$</td>
<td>$O(E + V \lg V)$</td>
</tr>
</tbody>
</table>

- $E$ is the number of edges in the graph.
- $V$ is the number of vertices in the graph.
Unweighted graphs

Suppose that $w(u, v) = 1$ for all $(u, v) \in E$. Can Dijkstra’s algorithm be improved?
Unweighted graphs

Suppose that $w(u, v) = 1$ for all $(u, v) \in E$. Can Dijkstra’s algorithm be improved?

- Use a simple FIFO queue instead of a priority queue.
Unweighted graphs

Suppose that $w(u, v) = 1$ for all $(u, v) \in E$. Can Dijkstra’s algorithm be improved?
- Use a simple FIFO queue instead of a priority queue.

**Breadth-first search**

```plaintext
while $Q \neq \emptyset$
do $u \leftarrow \text{DEQUEUE}(Q)$
for each $v \in \text{Adj}[u]$
do if $d[v] = \infty$
then $d[v] \leftarrow d[u] + 1$
$\text{ENQUEUE}(Q, v)$
```
Unweighted graphs

Suppose that $w(u, v) = 1$ for all $(u, v) \in E$. Can Dijkstra’s algorithm be improved?
• Use a simple FIFO queue instead of a priority queue.

*Breadth-first search*

```plaintext
while $Q \neq \emptyset$
do $u \leftarrow \text{DEQUEUE}(Q)$
for each $v \in \text{Adj}[u]$
do if $d[v] = \infty$
then $d[v] \leftarrow d[u] + 1$
\text{ENQUEUE}(Q, v)
```

**Analysis:** Time $= O(V + E)$. 

October 5, 2005

Copyright © 2001-5 by Erik D. Demaine and Charles E. Leiserson

L7.15
Q. How many h's cause x and y to collide?

A. There are m choices for each of a_1, a_2, …, a_r, but once these are chosen, exactly one choice for a_0 causes x and y to collide, namely:

\[
\begin{pmatrix}
\vdots \\
\end{pmatrix}
\]

Thus, the number of h's that cause x and y to collide is $m^r = |H|/m$.

October 5, 2005
Copyright © 2001-5 by Erik D. Demaine and Charles E. Leiserson
Q. How many $h$'s cause $x$ and $y$ to collide?

A. There are $m$ choices for each of $a_1, a_2, \ldots, a_r$, but once these are chosen, exactly one choice for $a_0$ causes $x$ and $y$ to collide, namely

$$
\begin{align*}
\sum_{i=1}^{r} a_i (x_i - y_i) &\mod m.
\end{align*}
$$

Thus, the number of $h$'s that cause $x$ and $y$ to collide is $m^r = |H|/m$. October 5, 2005 Copyright © 2001-5 by Erik D. Demaine and Charles E. Leiserson L7.15
Example of breadth-first search

Q: a b d

October 5, 2005
Copyright © 2001-5 by Erik D. Demaine and Charles E. Leiserson
Example of breadth-first search

Q: a b d c e
Proof (completed)

Q. How many $h$'s cause $x$ and $y$ to collide?

A. There are $m$ choices for each of $a_1, a_2, \ldots, a_r$, but once these are chosen, exactly one choice for $a_0$ causes $x$ and $y$ to collide, namely

$$
\begin{bmatrix}
\vdots \\
-1 \\
\vdots
\end{bmatrix}
= -a_i (x_i - y_i) \mod m.
$$

Thus, the number of $h$'s that cause $x$ and $y$ to collide is $m^r \cdot 1 = m^r = |H|/m$.

October 5, 2005

Copyright © 2001-5 by Erik D. Demaine and Charles E. Leiserson

L7.15
Example of breadth-first search

Q: a b d c e

0
a

1
b

c

1
d

2
e

2
g

2
h

i

f

October 5, 2005
Copyright © 2001-5 by Erik D. Demaine and Charles E. Leiserson
Proof (completed)

Q. How many h's cause x and y to collide?
A. There are \( m \) choices for each of \( a_1, a_2, \ldots, a_r \), but once these are chosen, exactly one choice for \( a_0 \) causes x and y to collide, namely

\[
\begin{pmatrix}
\sum_{i=1}^{r} \cdot \left( x_i - y_i \right)
\end{pmatrix}
\]

Thus, the number of h's that cause x and y to collide is

\[
m^r \cdot 1 = m^r = \frac{|H|}{m}.
\]

October 5, 2005
Copyright © 2001-5 by Erik D. Demaine and Charles E. Leiserson

Example of breadth-first search

Q: a b d c e g i
Example of breadth-first search

Q: a b d c e g i f
Q. How many $h$'s cause $x$ and $y$ to collide?

A. There are $m$ choices for each of $a_1, a_2, \ldots, a_r$, but once these are chosen, exactly one choice for $a_0$ causes $x$ and $y$ to collide, namely $\sum_i (x_0 - y_0 - 1) \cdot a_i \mod m$.

Thus, the number of $h$'s that cause $x$ and $y$ to collide is $m^r = |H| / m$.
Proof (completed)

Q. How many $h$'s cause $x$ and $y$ to collide?

A. There are $m$ choices for each of $a_1, a_2, \ldots, a_r$, but once these are chosen, exactly one choice for $a_0$ causes $x$ and $y$ to collide, namely

$$
\begin{align*}
\left( x_0 - y_0 \right) & \equiv a_i (x_i - y_i) \pmod{m}.
\end{align*}
$$

Thus, the number of $h$'s that cause $x$ and $y$ to collide is $m r \cdot 1 = m r = |H|/m$. 

October 5, 2005  
Copyright © 2001-2005 by Erik D. Demaine and Charles E. Leiserson
Q. How many $h$'s cause $x$ and $y$ to collide?

A. There are $m$ choices for each of $a_1, a_2, \ldots, a_r$, but once these are chosen, exactly one choice for $a_0$ causes $x$ and $y$ to collide, namely

\[
\begin{bmatrix}
1 \\
4 \\
3 \\
2 \\
1 \\
0
\end{bmatrix}
\begin{bmatrix}
x_0 \\
y_0 \\
x_i \\
y_i \\
x_j \\
x_k
\end{bmatrix}
\equiv
\begin{bmatrix}
a_0 \\
a_i \\
\vdots \\
\vdots \\
\vdots \\
\vdots
\end{bmatrix}
\]

Thus, the number of $h$'s that cause $x$ and $y$ to collide is $m^r \cdot 1 = m^r = |H|/m$. 

October 5, 2005
Copyright © 2001-5 by Erik D. Demaine and Charles E. Leiserson

Example of breadth-first search

\[Q: \quad a \ b \ d \ c \ e \ g \ i \ f \ h\]
Q. How many \( h \)’s cause \( x \) and \( y \) to collide?

A. There are \( m \) choices for each of \( a_1, a_2, \ldots, a_r \), but once these are chosen, exactly one choice for \( a_0 \) causes \( x \) and \( y \) to collide, namely

\[
\begin{bmatrix}
  r \\
  \sum_{i=1}^{n} \end{bmatrix}
\]

Thus, the number of \( h \)’s that cause \( x \) and \( y \) to collide is \( m^r = |H|/m \).

October 5, 2005
Copyright © 2001-5 by Erik D. Demaine and Charles E. Leiserson

Example of breadth-first search

\[ Q: \ a \ b \ d \ c \ e \ g \ i \ f \ h \]
Correctness of BFS

while $Q \neq \emptyset$
    do $u \leftarrow \text{DEQUEUE}(Q)$
    for each $v \in \text{Adj}[u]$
        do if $d[v] = \infty$
            then $d[v] \leftarrow d[u] + 1$
            $\text{ENQUEUE}(Q, v)$

Key idea:

The FIFO $Q$ in breadth-first search mimics the priority queue $Q$ in Dijkstra.

- **Invariant:** $v$ comes after $u$ in $Q$ implies that $d[v] = d[u]$ or $d[v] = d[u] + 1$. 