Resolving collisions w/ open addressing assuming $n \leq m$
Resolving Collisions w/ Open Addressing assuming \( n \leq m \)

The point is to avoid auxiliary linked lists. Use that space for table.
Resolving Collisions w/ Open Addressing assuming \( n \leq m \)

The point is to avoid auxiliary linked lists. Use that space for table.
Instead, create a probe sequence as a function of key value.

\[ \rightarrow \text{permutation of slots to try.} \]
Resolving collisions w/ open addressing assuming $n \leq m$

The point is to avoid auxiliary linked lists. Use that space for table. Instead, create a probe sequence as a function of key value. → permutation of slots to try.

ex: $h(64) \rightarrow 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6.$
Resolving collisions with open addressing assuming \( n \leq m \)

The point is to avoid auxilliary linked lists. Use that space for table. Instead, create a probe sequence as a function of key value.

\[ \text{permuation of slots to try.} \]

\[
\begin{array}{c}
1 \quad 36 \\
2 \quad 43 \\
3 \quad 78 \\
4 \quad 5 \\
5 \quad 103 \\
6 \quad 2014 \\
7 \\
8 \\
9 \\
10 \\
11 \\
\end{array}
\]

ex: \( h(64) \rightarrow 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6. \)

Insert(64): Try \( T[9] \): full
Resolving collisions w/ open addressing assuming $n \leq m$

The point is to avoid auxiliary linked lists. Use that space for table. Instead, create a probe sequence as a function of key value. 

$\rightarrow$ permutation of slots to try.

ex: $h(64) \rightarrow 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6.$

Insert(64): 
- Try $T[9]$: full
Resolving collisions w/ open addressing assuming \( n < m \)

The point is to avoid auxiliary linked lists. Use that space for table. Instead, create a probe sequence as a function of key value. A permutation of slots to try.

Example: \( h(64) \rightarrow 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6 \).

Insert(64):
- Try \( T[9] \): full
- Try \( T[2] \): full
- Try \( T[4] \): full
Resolving collisions w/ open addressing assuming \( n \leq m \)

The point is to avoid auxiliary linked lists. Use that space for table. Instead, create a probe sequence as a function of key value. \( \rightarrow \) permutation of slots to try.

Ex: \( h(64) \rightarrow 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6 \).

Insert(64):
- Try \( T[9] \): full
- Try \( T[2] \): full
- Try \( T[4] \): full
- Try \( T[8] \): ok
Resolving collisions w/ open addressing assuming $n \leq m$

The point is to avoid auxiliary linked lists. Use that space for table. Instead, create a probe sequence as a function of key value. \( \rightarrow \) permutation of slots to try.

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
36 & 43 & 78 & 5 & 103 & & & \\
\end{array}
\]

\[
\text{ex: } h(64) \rightarrow 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6.
\]

Insert(64):

- Try T[9]: full
- Try T[2]: full
- Try T[4]: full
- Try T[8]: ok

Search(64) follows same sequence. Would return "not found" after 4 attempts.
ex: $h(64) \rightarrow 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6$.

Really this is $h(k, i)$.
ex: $h(64) \rightarrow 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6.$

Really this is $h(k, i)$  

$h(64, 1) = 9$
ex: $h(64) \rightarrow 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6.$

Really this is $h(k, i)$

$h(64, 1) = 9$ / $h(64, 2) = 2$
ex: $h(64) \rightarrow 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6.$

Really this is $h(k, i)$

$h(64, 1) = 9$ / $h(64, 2) = 2$ / $h(64, 3) = 4$ / etc
Example: $h(64) \rightarrow 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6.$

Really this is $h(k, i)$:

- $h(64, 1) = 9$
- $h(64, 2) = 2$
- $h(64, 3) = 4$

```
1 36
  43
2
3
4 78
5
6
7 103
8 64
  2014
9
10
11
```

Delete(64): ?
ex: $h(64) \rightarrow 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6.$

Really this is $h(k, i)$

$h(64, 1) = 9 / h(64, 2) = 2 / h(64, 3) = 4 / etc$

Delete(64) : $h(64, 1) = 9$, occupied by 2014
$h(64) \rightarrow 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6.$

Really this is $h(k, i)$:

$h(64, 1) = 9$  
$h(64, 2) = 2$  
$h(64, 3) = 4$  

Delete(64):  
$h(64, 1) = 9$, occupied by 2014
$h(64, 2) = 2$, occupied by 43
ex: $h(64) \rightarrow 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6.$

Really this is $h(k, i)$

$h(64, 1) = 9$ / $h(64, 2) = 2$ / $h(64, 3) = 4$ / etc.

Delete(64):

$h(64, 1) = 9$, occupied by 2014
$h(64, 2) = 2$, occupied by 43
$h(64, 3) = 4$, occupied by 78
ex: $h(64) \rightarrow 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6$.

Really this is $h(k, i)$

$h(64, 1) = 9$ / $h(64, 2) = 2$ / $h(64, 3) = 4$ / etc

Delete(64):

$h(64, 1) = 9$, occupied by 2014
$h(64, 2) = 2$, occupied by 43
$h(64, 3) = 4$, occupied by 78
$h(64, 4) = 8$, found 64, DELETE IT.

OK?
ex: \( h(64) \rightarrow 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6. \)

Really this is \( h(k, i) \)

\[
\begin{align*}
h(64, 1) &= 9 & h(64, 2) &= 2 & h(64, 3) &= 4 \quad \text{etc.}
\end{align*}
\]

Delete(64): \( h(64, 1) = 9 \), occupied by 2014

\[
\begin{align*}
h(64, 2) &= 2 \quad , \text{occupied by} & 4 \quad 3
\end{align*}
\]

\[
\begin{align*}
h(64, 3) &= 4 \quad , \text{occupied by} & 7 \quad 8
\end{align*}
\]

\[
\begin{align*}
h(64, 4) &= 8 \quad , \text{found 64, DELETE IT.}
\end{align*}
\]

what if \( h(103) \rightarrow 4, 8, 2, 7, 1, 10, 11, 3, 5, 9, 6 \) ?
ex: $h(64) \rightarrow 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6$.

Really this is $h(k, i)$  

$\begin{align*}
  h(64, 1) &= 9, & h(64, 2) &= 2, & h(64, 3) &= 4, & \text{etc.}
\end{align*}$

Delete(64):

- $h(64, 1) = 9$, occupied by 2014
- $h(64, 2) = 2$, occupied by 43
- $h(64, 3) = 4$, occupied by 78
- $h(64, 4) = 8$, found 64, DELETE IT.

what if $h(103) \rightarrow 4, 8, 2, 7, 1, 10, 11, 3, 5, 9, 6$?

Search(103):

- $h(103, 1) = 4$, occupied by 78
ex: $h(64) \rightarrow 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6$.
Really this is $h(k, i)$

$h(64, 1) = 9$ / $h(64, 2) = 2$ / $h(64, 3) = 4$ / etc.

Delete(64):

- $h(64, 1) = 9$, occupied by 2014
- $h(64, 2) = 2$, occupied by 43
- $h(64, 3) = 4$, occupied by 78
- $h(64, 4) = 8$, found 64, DELETE IT.

What if $h(103) \rightarrow 4, 8, 2, 7, 1, 10, 11, 3, 5, 9, 6$?

Search(103):

- $h(103, 1) = 4$, occupied by 78
- $h(103, 2) = 8$, empty: declare 103 not in T.
ex: \( h(64) \rightarrow 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6. \)

Really this is \( h(k, i) \)

\[
\begin{align*}
h(64, 1) &= 9 / h(64, 2) = 2 / h(64, 3) = 4 / \ldots
\end{align*}
\]

Delete(64):

\[
\begin{align*}
h(64, 1) &= 9, \text{ occupied by } 2014 \\
h(64, 2) &= 2, \text{ occupied by } 43 \\
h(64, 3) &= 4, \text{ occupied by } 78 \\
h(64, 4) &= 8, \text{ found 64, DELETE IT.}
\end{align*}
\]

what if \( h(103) \rightarrow 4, 8, 2, 7, 1, 10, 11, 3, 5, 9, 6 \)?

Search(103):

\[
\begin{align*}
h(103, 1) &= 4, \text{ occupied by } 78 \\
h(103, 2) &= 8, \text{ empty: declare 103 not in T.}
\end{align*}
\]

Could use special "deleted" markers, but search time increases.
Typical probing sequences
Typical probing sequences

Linear probing : \( h(k, i) = (h(k, 0) + i) \mod m \)
Typical probing sequences

Linear probing: \( h(k, i) = (h(k, 0) + i) \mod m \sim h(k) \) and wrap around.
**Typical probing sequences**

Linear probing: \( h(k,i) = (h(k,0) + i) \mod m \Rightarrow h(k) \) and wrap around.

...tends to generate clusters.

\[
\text{probability of extending a cluster} = \frac{|\text{cluster}|}{m}
\]

slows down search
Typical probing sequences

Linear probing: \( h(k,i) = (h(k,0) + i) \mod m \) tends to generate clusters.

\( \sim h(k) \) and wrap around.
Typical probing sequences

Linear probing: \( h(k, i) = (h(k, 0) + i) \mod m \) \( \sim h(k) \) and wrap around.

...tends to generate clusters.

Quadratic probing: \( h(k, i) = (h(k, 0) + c \cdot i + d \cdot i^2) \mod m \)
Typical probing sequences

Linear probing: \( h(k, i) = (h(k, 0) + i) \mod m \) \( \sim h(k) \) and wrap around.

...tends to generate clusters.

Quadratic probing: \( h(k, i) = (h(k, 0) + c \cdot i + d \cdot i^2) \mod m \)

linear \( \sim \) make it look more random
Typical probing sequences

Linear probing: $h(k,i) = (h(k,0) + i) \mod m$ \quad \sim \quad h(k)$ and wrap around.

... tends to generate clusters.

Quadratic probing: $h(k,i) = (h(k,0) + c \cdot i + d \cdot i^2) \mod m$

Less clustering, need to make sure sequence hits all slots.
Typical probing sequences

Linear probing: \[ h(k,i) = (h(k,0) + i) \mod m \] \( \sim h(k) \) and wrap around.

... tends to generate clusters.

Quadratic probing: \[ h(k,i) = (h(k,0) + c \cdot i + d \cdot i^2) \mod m \] 

Less clustering, need to make sure sequence hits all slots

Both generate \( m \) probe sequences in total
Typical probing sequences

Linear probing: \( h(k, i) = (h(k, 0) + i) \mod m \) \( \sim h(k) \) and wrap around.

... tends to generate clusters.

Quadratic probing: \( h(k, i) = (h(k, 0) + c \cdot i + d \cdot i^2) \mod m \)

Make it look more random

Less clustering, need to make sure sequence hits all slots

Both generate \( m \) probe sequences in total

Double hashing: \( h(k, i) = (h_1(k) + i \cdot h_2(k)) \mod m \)
Typical probing sequences

Linear probing: \( h(k, i) = (h(k, 0) + i) \mod m \sim h(k) \) and wrap around.

... tends to generate clusters.

Quadratic probing: \( h(k, i) = (h(k, 0) + c \cdot i + d \cdot i^2) \mod m \)

Less clustering, need to make sure sequence hits all slots

Both generate \( m \) probe sequences in total

Double hashing: \( h(k, i) = (h_1(k) + i \cdot h_2(k)) \mod m \)

Each \( k \) has "random" offset
Typical probing sequences

- **Linear probing**: \( h(k,i) = (h(k,0) + i) \mod m \) ~ \( h(k) \) and wrap around.
  ... tends to generate clusters.

- **Quadratic probing**: \( h(k,i) = (h(k,0) + c \cdot i + d \cdot i^2) \mod m \)
  Less clustering, need to make sure sequence hits all slots
  Linear make it look more random

\[ \Rightarrow \text{Both generate m probe sequences in total} \]

- **Double hashing**: \( h(k,i) = (h_1(k) + i \cdot h_2(k)) \mod m \)
  Each \( k \) has "random" offset

Generates \( O(m^2) \) probe sequences: better
Typical probing sequences

**Linear probing:** \( h(k,i) = (h(k,0) + i) \mod m \) \( \sim h(k) \) and wrap around.

... tends to generate clusters.

**Quadratic probing:** \( h(k,i) = (h(k,0) + c \cdot i + d \cdot i^2) \mod m \)

Less clustering, need to make sure sequence hits all slots

→ Both generate m probe sequences in total

**Double hashing:** \( h(k,i) = (h_1(k) + i \cdot h_2(k)) \mod m \)

Each k has "random" offset

Generates \( O(m^2) \) probe sequences: better

Heuristic: choose \( m = 2^r \) & \( h_2(k) \) : odd.
Analysis of open addressing
Analysis of Open Addressing

Assuming Uniform Hashing: each key is equally likely to have any of the $m!$ permutations as probe sequence (independent of other keys)

For a random $h$, every slot is equally likely
Analysis of open addressing

Assuming uniform hashing: each key is equally likely to have any of the \( m! \) permutations as probe sequence (independent of other keys)

Even though all we have so far is \( O(m^2) \)

Simple uniform hashing

For a random \( h \), every slot is equally likely
Analysis of open addressing

Assuming uniform hashing: each key is equally likely to have any of the \( m! \) permutations as probe sequence (independent of other keys)

Recall \( n < m \), so \( \alpha < 1 \). Claim: \( E[\text{#probes}] \leq \frac{1}{1-\alpha} \left( \frac{m}{m-n} \right) \) (search)
Analysis of open addressing

Assuming uniform hashing: each key is equally likely to have any of the \( m! \) permutations as probe sequence (independent of other keys)

Recall \( n < m \), so \( \alpha < 1 \). Claim: \( E[\#\text{probes}] \leq \frac{1}{1-\alpha} \left( \frac{m}{m-n} \right) \) (search)

If true, then for \( n \ll m \) \( E[\#\text{probes}] = O(1) \)
Analysis of open addressing

Assuming uniform hashing: Each key is equally likely to have any of the $m!$ permutations as probe sequence (independent of other keys).

Recall $n < m$, so $\alpha < 1$. Claim: $E[\# \text{probes}] \leq \frac{1}{1-\alpha} \left( \frac{m}{m-n} \right)$ (search)

If true, then for $n \ll m \quad E[\# \text{probes}] = O(1)$

$\Rightarrow n = \frac{1}{2} m \rightarrow 2 \text{ probes}$

$\Rightarrow 90\% \text{ full table} \rightarrow 10 \text{ probes}$
ANALYSIS OF OPEN ADDRESSING

ASSUMING UNIFORM HASHING: each key is equally likely to have any of the m! permutations as probe sequence (independent of other keys).

Recall n < m, so \( \alpha < 1 \).

Claim: \( E[\# \text{probes}] \leq \frac{1}{1-\alpha} \left( \frac{m}{m-n} \right) \) (search)

If true, then for \( n \ll m \) \( E[\#\text{probes}] = O(1) \)

\( \therefore n = \frac{1}{2} m \rightarrow 2 \) probes

\( \therefore 90\% \text{ full table} \rightarrow 10 \) probes

Works well if you can afford a table \( \sim \text{data} \times 2 \)
Claim: \( E[\#\text{probes}] \leq \frac{1}{1-\alpha} \)  

Look at unsuccessful search
Claim: $E[\#\text{probes}] \leq \frac{1}{1-\alpha}$

Look at unsuccessful search

Remember, probe sequence is a permutation.
Never check one slot twice.
Claim: \( E[\# \text{probes}] \leq \frac{1}{1 - \alpha} \)

Look at unsuccessful search

\[ P[\text{1st probe collides}] = \frac{n}{m} \]

Remember, probe sequence is a permutation. Never check one slot twice.
Claim: $E[\#\text{probes}] \leq \frac{1}{1 - \alpha}$

$P[\text{1st probe collides}] = \frac{n}{m} \rightarrow \text{need 2nd probe}$

Look at unsuccessful search

Remember, probe sequence is a permutation. Never check one slot twice.
Claim: $E[\#\text{probes}] \leq \frac{1}{1-\alpha}$

Look at unsuccessful search

$P[1\text{st probe collides}] = \frac{n}{m} \rightarrow \text{need 2nd probe}$

$P[2\text{nd probe collides}] = \frac{n-1}{m-1} \rightarrow \text{need 3rd probe}$

Remember, probe sequence is a permutation.
Never check one slot twice.
Claim: \( E[\text{#probes}] \leq \frac{1}{1-\alpha} \)

- \( P[\text{1st probe collides}] = \frac{n}{m} \rightarrow \text{need 2nd probe} \)
- \( P[\text{2nd probe collides}] = \frac{n-i}{m-1} \rightarrow \text{need 3rd probe} \)

\[ \vdots \]
- \( P[\text{ith probe collides}] = \frac{n-i}{m-i} \)

Look at unsuccessful search

Remember, probe sequence is a permutation.
Never check one slot twice.
Claim: \( E[\#\text{probes}] \leq \frac{1}{1-\alpha} \)

Look at unsuccessful search

\[
P[1\text{st probe collides}] = \frac{n}{m} \quad \rightarrow \text{need 2nd probe}
\]

\[
P[2\text{nd probe collides}] = \frac{n-i}{m-1} \quad \rightarrow \text{need 3rd probe}
\]

\[
\vdots
\]

\[
\frac{n-i}{m-i} \quad < \quad \frac{n}{m} = \alpha
\]

Remember, probe sequence is a permutation.
Never check one slot twice.
Claim: $E[\#\text{probes}] \leq \frac{1}{1-\alpha}$

Look at unsuccessful search

* $P[\text{1st probe collides}] = \frac{n}{m} \rightarrow \text{need 2nd probe}$

* $P[\text{2nd probe collides}] = \frac{n-i}{m-1} \rightarrow \text{need 3rd probe}$

\[ \vdots \]

* $\frac{n-i}{m-i} < \frac{n}{m} = \alpha$

$E[\#\text{probes}] = 1 + \frac{n}{m} \left( \text{need at least a 2nd probe} \right)$

↑

Remember, probe sequence is a permutation.
Never check one slot twice.
Claim: \( E[\text{#probes}] \leq \frac{1}{1-\alpha} \)

Look at unsuccessful search

\[
P[\text{1st probe collides}] = \frac{n}{m} \quad \rightarrow \quad \text{need 2nd probe}
\]

\[
P[\text{2nd probe collides}] = \frac{n-1}{m-1} \quad \rightarrow \quad \text{need 3rd probe}
\]

\[
\vdots
\]

\[
\frac{n-i}{m-i} < \frac{n}{m} = \alpha
\]

\[
E[\text{#probes}] = 1 + \frac{n}{m} \left( 1 + \frac{n-1}{m-1} \right) \quad \text{(need a 3rd probe)}
\]
Claim: \( E[\#\text{probes}] \leq \frac{1}{1-\alpha} \)

Look at unsuccessful search

\[
P[1\text{st probe collides}] = \frac{n}{m} \quad \rightarrow \quad \text{need 2nd probe}
\]

\[
P[2\text{nd probe collides}] = \frac{n-1}{m-1} \quad \rightarrow \quad \text{need 3rd probe}
\]

\[
\vdots
\]

\[
\frac{n-i}{m-i} < \frac{n}{m} = \alpha
\]

\[
E[\#\text{probes}] = 1 + \frac{n}{m} \left( 1 + \frac{n-1}{m-1} \left( 1 + \frac{n-2}{m-2} \left( \ldots \right) \right) \right)
\]

Remember, probe sequence is a permutation. Never check one slot twice.
Claim: $E[\# \text{probes}] \leq \frac{1}{1-\alpha}$  

Look at unsuccessful search

$P[1\text{st\ probe\ collides}] = \frac{n}{m} \rightarrow \text{need 2nd probe}$

$P[2\text{nd\ probe\ collides}] = \frac{n-1}{m-1} \rightarrow \text{need 3rd probe}$

$\vdots$

$\frac{n-i}{m-i} < \frac{n}{m} = \alpha$

Remember, probe sequence is a permutation. Never check one slot twice.

$E[\# \text{probes}] = 1 + \frac{n}{m} \left( 1 + \frac{n-1}{m-1} \left( 1 + \frac{n-2}{m-2} \left( \ldots (1 + \frac{1}{m-n}) \right) \right) \right)$
Claim: $E[\#\text{probes}] \leq \frac{1}{1-\alpha}$  

Look at unsuccessful search

$P[\text{1st probe collides}] = \frac{n}{m} \quad \rightarrow \quad \text{need 2nd probe}$

$P[\text{2nd probe collides}] = \frac{n-1}{m-1} \quad \rightarrow \quad \text{need 3rd probe}$

\[ \vdots \]

$\frac{n-i}{m-i} \quad \leq \quad \frac{n}{m} = \alpha$

$E[\#\text{probes}] = 1 + \frac{n}{m} \left( 1 + \frac{n-1}{m-1} \left( 1 + \frac{n-2}{m-2} \left( \cdots \left( 1 + \frac{1}{m-n} \right) \right) \right) \right)$

\[ \leq 1 + \alpha \left( 1 + \alpha \left( 1 + \alpha \left( \cdots \left( 1 + \alpha \right) \right) \right) \right) \quad \ldots \quad n \text{ terms} \]
Claim: $E[\#\text{probes}] \leq \frac{1}{1-\alpha}$

Look at unsuccessful search

$P[1\text{st probe collides}] = \frac{n}{m} \rightarrow \text{need 2nd probe}$

$P[2\text{nd probe collides}] = \frac{n-1}{m-1} \rightarrow \text{need 3rd probe}$

\[ \vdots \]

$\frac{n-i}{m-i} < \frac{n}{m} = \alpha$

Remember, probe sequence is a permutation.
Never check one slot twice.

$E[\#\text{probes}] = 1 + \frac{n}{m} \left(1 + \frac{n-1}{m-1} \left(1 + \frac{n-2}{m-2} \left( \cdots \left(1 + \frac{1}{m-n}\right)\right)\right)\right)$

$\leq 1 + \alpha \left(1 + \alpha \left(1 + \alpha \left( \cdots \left(1 + \alpha \right)\right)\right)\right)$ \hspace{1cm} \ldots \text{n terms}$

$\leq 1 + \alpha + \alpha^2 + \alpha^3 + \cdots \hspace{1cm} \ldots \ldots \alpha \text{ terms}$
Claim: $E[\#\text{probes}] \leq \frac{1}{1-\alpha}$

Look at unsuccessful search

$P[1\text{st probe collides}] = \frac{n}{m} \rightarrow \text{need 2nd probe}$

$P[2\text{nd probe collides}] = \frac{n-1}{m-1} \rightarrow \text{need 3rd probe}$

$\vdots$

$\frac{n-i}{m-i} < \frac{n}{m} = \alpha$

$E[\#\text{probes}] = 1 + \frac{n}{m} \left( 1 + \frac{n-1}{m-1} \left( 1 + \frac{n-2}{m-2} \left( \ldots \left( 1 + \frac{1}{m-n} \right) \right) \right) \right)$

$\leq 1 + \alpha \left( 1 + \alpha \left( 1 + \alpha \left( \ldots \left( 1 + \alpha \right) \right) \right) \right) \ldots \text{n terms}$

$\leq 1 + \alpha + \alpha^2 + \alpha^3 + \ldots \ldots \text{\infty terms}$

$= \sum_{i=0}^{\infty} \alpha^i$
Claim: $\mathbb{E}[\#\text{probes}] \leq \frac{1}{1-\alpha}$  

Look at unsuccessful search

\[
P[1\text{st probe collides}] = \frac{n}{m} \quad \rightarrow \text{need 2nd probe}
\]

\[
P[2\text{nd probe collides}] = \frac{n-1}{m-1} \quad \rightarrow \text{need 3rd probe}
\]

\[\vdots\]

\[
\frac{n-i}{m-i} \leq \frac{n}{m} = \alpha
\]

\[
\mathbb{E}[\#\text{probes}] = 1 + \frac{n}{m} \left( 1 + \frac{n-1}{m-1} \left( 1 + \frac{n-2}{m-2} \left( \cdots (1 + \frac{1}{m-n}) \right) \right) \right)
\]

\[\leq 1 + \alpha \left( 1 + \alpha \left( 1 + \alpha \left( \cdots (1 + \alpha) \right) \right) \right) \quad \cdots \text{\(n\) terms}
\]

\[\leq 1 + \alpha + \alpha^2 + \alpha^3 + \cdots \quad \cdots \infty \text{ terms}
\]

\[
= \sum_{i=0}^{\infty} \alpha^i = \frac{1}{1-\alpha}
\]

Remember, probe sequence is a permutation. Never check one slot twice.

See CLRS for alternate analysis incl. successful search.