# Open Address Hashing (continued)

Adapted from the CLRS book slides

# ANALYSIS OF OPEN-ADDRESS HASHING

***Assumptions***

- Analysis is in terms of load factor $\alpha$. Assume that the table never completely fills, so always have $0 \leq n < m \Rightarrow 0 \leq \alpha < 1$.
- Assume independent uniform permutation hashing.
- No deletion.
- In a successful search, each key is equally likely to be searched for.

***Theorem***

The expected number of probes in an unsuccessful search is at most $1/(1-\alpha)$.

# ANALYSIS OF OPEN-ADDRESS HASHING (continued)

***Intuition behind the proof:*** The first probe always occurs. The first probe finds an occupied slot with probability (approximately) $\alpha$, so that a second probe happens. With probability (approximately) $\alpha^2$, the first two slots are occupied, so that a third probe occurs. Get the geometric series $1 + \alpha + \alpha^2 + \alpha^3 + \cdots = 1/(1 - \alpha)$ (since $\alpha < 1$).

# ANALYSIS OF OPEN-ADDRESS HASHING (continued)

***Interpretation***

If $\alpha$ is constant, an unsuccessful search takes $O(1)$ time.

- If $\alpha = 0.5$, then an unsuccessful search takes an average of $1/(1 - 0.5) = 2$ probes.
- If $\alpha = 0.9$, takes an average of $1/(1 - 0.9) = 10$ probes.

***Corollary***

The expected number of probes to insert is at most $1/(1 - \alpha)$.

***Proof*** Since there is no deletion, insertion uses the same probe sequence as an unsuccessful search. ■

# ANALYSIS OF OPEN-ADDRESS HASHING (continued)

***Theorem***

The expected number of probes in a successful search is at most $\frac{1}{\alpha} \ln \frac{1}{1-\alpha}$.

***Proof*** A successful search for key $k$ follows the same probe sequence as when key $k$ was inserted.

By the previous corollary, if $k$ was the $(i+1)$st key inserted, then $\alpha$ equaled $i/m$ at the time. Thus, the expected number of probes made in a search for $k$ is at most $1/(1-i/m) = m/(m-i)$.

# ANALYSIS OF OPEN-ADDRESS HASHING (continued)

That was assuming that $k$ was the $(i + 1)$st key inserted. We need to average over all $n$ keys:

$$
\begin{aligned}
\frac{1}{n}\sum_{i=0}^{n-1}\frac{m}{m-i} &= \frac{m}{n}\sum_{i=0}^{n-1}\frac{1}{m-i} \\
&= \frac{1}{\alpha}\sum_{k=m-n+1}^{m}\frac{1}{k} \\
&\le \frac{1}{\alpha}\int_{m-n}^{m}(1/x)\,dx \qquad \text{(by inequality (A.19))} \\
&= \frac{1}{\alpha}\ln\frac{m}{m-n} \\
&= \frac{1}{\alpha}\ln\frac{1}{1-\alpha}
\end{aligned}
$$

# PRACTICAL CONSIDERATIONS

Modern CPUs have features that affect hashing:

**Memory hierarchies:** ***Caches*** are small, fast memory units closer to where instructions execute. They are organized in ***cache lines*** of a specific size (e.g., 64 bytes) of contiguous bytes from main memory. It's much faster to reuse a cache line than to fetch from main memory. Iterating through a cache line is relatively fast.

**Advanced instruction sets:** Advanced primitives for encryption and cryptography can also be used to compute hash functions.

# ANALYSIS OF LINEAR PROBING

Linear probing exhibits ***primary clustering***: long runs of occupied sequences build up. And long runs tend to get longer, since an empty slot preceded by $i$ full slots gets filled next with probability $(i + 1)/m$. Result is that the average search and insertion times increase.

Primary clustering slows things down in the RAM model, but it helps in hierarchical memory because searching stays in the same cache line for as long as possible.