Hashing (1)
The general setup:

- \( \mathcal{U} \) – (large, finite) universe of possible keys
- \( \mathcal{V} \) – (small) set of slots of size \( m \)
  
  \( \text{typically } \mathcal{V} = [0..m) \)
- \( h : \mathcal{U} \rightarrow \mathcal{V} \) – a “hash function” from \( \mathcal{U} \) to \( \mathcal{V} \)
  
  maps keys to slots
- \( T[\mathcal{V}] \) – a “hash table” for storing keys, indexed by \( \mathcal{V} \)

Implementing a dictionary:

- A key \( a \in \mathcal{U} \) is stored in the hash table \( T \) at slot \( s = h(a) \)
- As long as no two keys hash to the same slot (a “collision”), we can perform all dictionary operations (insert, search, delete) in constant time
Resolving collisions by chaining
Dictionary Operations:

- `insert(a)`: insert `a` in the linked list `T[h(a)]`
- `search(a)`: search for `a` in `T[h(a)]`
- `delete(a)`: search for and delete `a` in `T[h(a)]`

Running times:

- `insert` – \( O(1) \)
- `search`, `delete` – \( O(n) \) (worst case)

Worst case occurs when all keys hash to the same slot

Better: choose a *random* hash function — no “pile ups”
Universal Hashing [Carter & Wegman, 1975]

- $\Lambda$ – a finite, non-empty set of hash function indices
- $\mathcal{H} = \{h_\lambda\}_{\lambda \in \Lambda}$ – a family of hash functions from $\mathcal{U}$ to $\mathcal{V}$, indexed by $\lambda \in \Lambda$
- $m := |\mathcal{V}|

Def’n: $\mathcal{H}$ is called universal if for all $a, b \in \mathcal{U}$ with $a \neq b$,

$$|\{\lambda \in \Lambda : h_\lambda(a) = h_\lambda(b)\}| \leq \frac{|\Lambda|}{m}.$$

Probabilistic interpretation: if $R$ is a random variable, uniformly distributed over $\Lambda$, then

$$\Pr[h_R(a) = h_R(b)] \leq \frac{1}{m}.$$
Using Universal Hash Functions

Assume distinct keys $a_1, \ldots, a_n$ are stored in table
Let $\alpha := n/m = \text{“load factor”}$
Assume $R$ is uniformly distributed over $\Lambda$
For $i = 1, \ldots, n$, define
$$S_i := \# \text{ of keys in slot } h_R(a_i)$$
That is, $S_i$ is the number of keys in the slot occupied by $a_i$
The values $R, S_1, \ldots, S_n$ are random variables.
For each $i = 1, \ldots, n$, we wish to bound $E[S_i]$. 
**Claim:** \( \mathbb{E}[S_i] \leq \alpha + 1 \) for each \( i = 1, \ldots, n \).

**Proof:** for \( i, j = 1, \ldots, n \), define indicator variables

\[
C_{ij} := \begin{cases} 
1 & \text{if } h_R(a_i) = h_R(a_j) \\
0 & \text{otherwise}
\end{cases}
\]

For all \( i, j \):

\[
\mathbb{E}[C_{ij}] = \Pr[h_R(a_i) = h_R(a_j)] \leq 1/m \quad \text{if } i \neq j
\]

\[
= 1 \quad \text{if } i = j
\]

Write \( S_i \) as sum of indicator variables: \( S_i = \sum_{j=1}^{n} C_{ij} \)

By linearity of expectation:

\[
\mathbb{E}[S_i] = \sum_{j=1}^{n} \mathbb{E}[C_{ij}] = \mathbb{E}[C_{ii}] + \sum_{j \neq i} \mathbb{E}[C_{ij}]
\]

\[
\leq 1 + (n - 1)/m
\]

\[
\leq \alpha + 1 \quad \text{QED}
\]
interpretation:

- for each $i$, the expected number of keys in $a_i$'s slot (including $a_i$ itself) is $\leq \alpha + 1$

- the expected time to perform a single dictionary operation is $O(\alpha + 1)$

- by linearity of expectation, expected time to perform $k$ dictionary operations is $O(k(\alpha + 1))$

**special case:** $\alpha = O(1)$ (i.e., $n = O(m)$)

- expected time per operation is $O(1)$
Maximum Load: another performance measure

Suppose hash table contains keys $a_1, \ldots, a_n$, and that $R$ is uniform over $\Lambda$

For $s \in \mathcal{V}$, define

$$L_s := \# \text{ of } a_i \text{'s that hash to slot } s \text{ under } h_R$$

Set $M := \max \{L_s : s \in \mathcal{V}\}$

We want to bound $E[M]$, assuming universal hashing

**Jensen says:** $E[M]^2 \leq E[M^2]

**Observe:** $M^2 \leq V := \sum_{s \in \mathcal{V}} L_s^2$
**Claim:** \( E[V] \leq n^2 / m + n \)

We have \( V = \sum_s L_s^2 \)

We will first argue that \( V = \sum_{i,j} C_{ij} \)

Let us order the keys so that the first bunch keys go in the one slot, the second bunch of keys go in another slot, and so on.

Shaded area is \( \sum_s L_s^2 = \sum_{i,j} C_{ij} \)
So we have

\[ V = \sum_{i,j} C_{ij} \]

and by linearity of expectation, we have

\[
E[V] = \sum_{i,j} E[C_{ij}]
\]

\[
= \sum_{i} E[C_{ii}] + \sum_{i \neq j} E[C_{ij}]
\]

\[
\leq n + n(n - 1)/m
\]

\[
\leq n^2/m + n
\]

QED
Corollary: \( E[M] \leq \sqrt{n^2/m + n} \)

Special case: \( \alpha = O(1) \)

\[
E[M] = O(\sqrt{n})
\]

- This bound is tight
- Counter-intuitive: it may be the case that \( E[L_s] = O(1) \) for each slot \( s \)

\textit{Expected value of max may be much larger than max of expected values}