Probing Strategies

- Linear Probing
  - primary clustering
  - long runs of filled slots

$$h(k, i) = \left(h(k, 0) + i\right) \bmod m$$

- Quadratic Probing

$$h(k, i) = \left(h(k, 0) + i^2\right) \bmod m$$

- Double hashing

$$h(k, i) = \left(h_1(k) + i\, h_2(k)\right) \bmod m$$

↳ excellent method by statistical studies

Usually pick $m = 2^r$ and $h_2(k)$ odd

Analysis of open addressing

* **Assumption of uniform hashing**

Each key is equally likely to have any one of the $m!$ permutations as its probe sequence independent of other keys

Theorem: $E[\#\text{ probes}] \leq \frac{1}{1-\alpha}$ if $\alpha < 1$ ie $n < m$

Pf: (unsuccessful search)

1 probe is always necessary

with prob n/m, collision $\Rightarrow$ 2nd probe necessary

" " $n-1/m-1$, " $\Rightarrow$ 3rd probe necessary

" " $n-2/m-2$, " $\Rightarrow$ 4th probe necessary

Check $\frac{n-i}{m-i} \leq \frac{n}{m} = \alpha$ for $i = 1, 2, \ldots, m-1$

$\therefore$ Expected # of probes

$$= 1 + \frac{n}{m}\left(1 + \frac{n-1}{m-1}\left(1 + \frac{n-2}{m-2}\left(1 + \ldots 1 + \frac{1}{m-n}\right)\ldots\right)\right)$$

$$\leq 1 + \alpha(1 + \alpha(1 + \alpha(\ldots\ldots(1+\alpha)\ldots)))$$

$$\leq 1 + \alpha + \alpha^2 + \alpha^3 + \ldots\ldots = \frac{1}{1-\alpha}$$

if $\alpha < 1$ is a constant,

$\Rightarrow O(1)$ probes

Ex: Table 50% full $\Rightarrow \alpha = \frac{1}{2} \Rightarrow 2$ probes

90% full $\Rightarrow \alpha = 9/10 \Rightarrow 10$ probes

→ Some Background on Probability Theory (1.3.4)

→ Sample space S

* outcomes of an experiment

* can be finite or infinite

→ probability space

= Sample space + prob fn

→ event E is a subset of S, $E \subseteq S$

* $Pr(\text{event } E) = \sum_{o \in E} Pr(o)$

→ Two events A & B are independent if

$$Pr(A \mid B) = Pr(A)$$

$$\approx \quad Pr(A \cap B) = Pr(A)\, Pr(B)$$

Random Variables

$$X : S \to \mathbb{R}$$

$$E(X) = \sum_x x \Pr(X = x)$$

$$= \sum_x \Pr(X \geq x)$$

→ Linearity of Expectation

$$Z = X + Y$$

$$\to E(Z) = E(X) + E(Y)$$

If X & Y are independent,

$$E(XY) = E(X)E(Y)$$

Remark

Hash functions

We have seen ~~compression maps~~ till now where we assume that the keys come from an underlying $U = \{0, 1, 2, \ldots, m-1\}$

How do we encode general keys to numbers?

$\rightarrow$ hash code