Overflow Handling

- An overflow occurs when the home bucket for a new pair \((\text{key}, \text{element})\) is full.
- We may handle overflows by:
  - Search the hash table in some systematic fashion for a bucket that is not full.
    - Linear probing (linear open addressing).
    - Quadratic probing.
    - Random probing.
  - Eliminate overflows by permitting each bucket to keep a list of all pairs for which it is the home bucket.
    - Array linear list.
    - Chain.
Linear Probing – Get And Put

- divisor = b (number of buckets) = 17.
- Home bucket = key % 17.

- Put in pairs whose keys are 6, 12, 34, 29, 28, 11, 23, 7, 0, 33, 30, 45
Linear Probing – Erase

- **erase(0)**

- Search cluster for pair (if any) to fill vacated bucket.
Linear Probing – erase(34)

- Search cluster for pair (if any) to fill vacated bucket.

0 4 8 12 16

34 0 45 6 23 7 28 12 29 11 30 33

0 4 8 12 16

0 45 6 23 7 28 12 29 11 30 33
Linear Probing – erase(29)

Search cluster for pair (if any) to fill vacated bucket.
Performance Of Linear Probing

- Worst-case find/insert/erase time is $\Theta(n)$, where $n$ is the number of pairs in the table.
- This happens when all pairs are in the same cluster.
Expected Performance

- $\alpha = \text{loading density} = \frac{\text{(number of pairs)}}{b}$.
  - $\alpha = \frac{12}{17}$.
- $S_n = \text{expected number of buckets examined in a successful search when } n \text{ is large}$
- $U_n = \text{expected number of buckets examined in a unsuccessful search when } n \text{ is large}$
- Time to put and remove governed by $U_n$. 
Expected Performance

- $S_n \sim \frac{1}{2}(1 + \frac{1}{1 - \alpha})$
- $U_n \sim \frac{1}{2}(1 + \frac{1}{1 - \alpha^2})$
- Note that $0 \leq \alpha \leq 1$.

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$S_n$</th>
<th>$U_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.50</td>
<td>1.5</td>
<td>2.5</td>
</tr>
<tr>
<td>0.75</td>
<td>2.5</td>
<td>8.5</td>
</tr>
<tr>
<td>0.90</td>
<td>5.5</td>
<td>50.5</td>
</tr>
</tbody>
</table>

$\alpha \leq 0.75$ is recommended.
Hash Table Design

- Performance requirements are given, determine maximum permissible loading density.
- We want a successful search to make no more than 10 compares (expected).
  - $S_n \sim \frac{1}{2}(1 + 1/(1 - \alpha))$
  - $\alpha \leq 18/19$
- We want an unsuccessful search to make no more than 13 compares (expected).
  - $U_n \sim \frac{1}{2}(1 + 1/(1 - \alpha)^2)$
  - $\alpha \leq 4/5$
- So $\alpha \leq \min\{18/19, 4/5\} = 4/5$. 
Hash Table Design

• Dynamic resizing of table.
  ▪ Whenever loading density exceeds threshold (4/5 in our example), rehash into a table of approximately twice the current size.

• Fixed table size.
  ▪ Know maximum number of pairs.
  ▪ No more than 1000 pairs.
  ▪ Loading density $\leq 4/5 \Rightarrow b \geq 5/4*1000 = 1250$.
  ▪ Pick $b$ (equal to divisor) to be a prime number or an odd number with no prime divisors smaller than 20.
Linear List Of Synonyms

- Each bucket keeps a linear list of all pairs for which it is the home bucket.
- The linear list may or may not be sorted by key.
- The linear list may be an array linear list or a chain.
Sorted Chains

- Put in pairs whose keys are 6, 12, 34, 29, 28, 11, 23, 7, 0, 33, 30, 45
- Home bucket = key % 17.
Expected Performance

- Note that $\alpha \geq 0$.
- Expected chain length is $\alpha$.
- $S_n \sim 1 + \alpha/2$.
- $U_n \leq \alpha$, when $\alpha < 1$.
- $U_n \sim 1 + \alpha/2$, when $\alpha \geq 1$. 