Twin Hashing

Idea:

• An issue, at least theoretically, is the size of the largest bucket.
• If we are unlucky and after many items have been placed into the table there are many member instructions to a single bucket, it would be nice to know that even in the worst case when this was the largest bucket, it cannot be too large.
• We have already seen that asymptotically the expected size of the largest bucket is $O(\log(n)/\log\log(n))$.
• Twin hashing reduces by an exponential factor the expected size of the largest bucket to only $O(\log\log(n))$. 
Twin hashing algorithm:

Select two different hash functions $h_1$ and $h_2$.

**MEMBER($d$):** Search the buckets at $A[h_1(d)]$ and $A[h_2(d)]$.

**INSERT($d$):** Do MEMBER($d$) and if $d$ is not present, add $d$ to the smaller of the buckets at $A[h_1(d)]$ and $A[h_2(d)]$ (if they are the same size, choose $A[h_1(d)]$).

**DELETE($d$):** Do MEMBER($d$) and if $d$ is present, remove $d$ from its bucket.

Complexity:

- The time analysis is the same as for a single hash function; the overhead of using two hash functions and searching two buckets only increases time by a constant factor.
- No additional space is needed; although we could store the size of each bucket (which would only increase space by a constant factor), it is not necessary for a worst case analysis since the two buckets must be searched in any case for a MEMBER operation (and their size can be counted).
(twin hashing continued)

**Size of the largest bucket:**
It can be shown that the expected size of the largest bucket is only $O(\log\log(n))$ under the assumption that $h_1$ and $h_2$ are ideal hash functions that distribute items uniformly and randomly.

(In fact, performance is improved slightly if $h_1$ and $h_2$ place elements uniformly and randomly into the left and right halves of the table respectively, so that $0 \leq h_1(d) < \lceil m/2 \rceil \leq h_2(d) < m$ for a table that goes from 0 to $m-1$).

**Practical considerations:**
- Twin hashing basically doubles the expected time for hashing. The asymptotic "insurance" it provides may not be worth this cost in many typical applications.
- In any case, however, it is a nice theoretical tool.