Resizing a Hash Table

**Idea:** Rehash into a larger table when the current table "fills" and amortize that cost against the work used to fill up the current table.

**Resizing algorithm:** Start with a table of size of $m=1$ (in practice, $m$ could be an estimate of the number of elements that are to be inserted).

1. Initialize a hash table of size $m=1$.
2. $n=0$
3. **for** each new element that has to be added to the table **do begin**
   1. $n := n+1$
   2. **if** $n > m$ **then begin**
      1. $m := 2m$
      2. Allocate a new hash table of size $m$.
      3. Rehash all the existing elements into the new table.
      4. Reclaim the memory for the old table.
   **end**
4. Hash the new element into the table.
5. **end**
(resizing a hash table continued)

**Time:**

- After all $n$ elements have been inserted, final hash table has size $n \leq m < 2n$.

- Hence, if $N$ denotes the largest power of 2 that is less than $n$, then the total number hash operations is $n$ (the number of hashes to put items in the table for the first time) plus the number of rehashes, which is given by:
  
  $$n + (1 + 2 + 4 + \cdots + N) = n + 2N - 1 \leq n+2(n-1)-1 \leq 3n - 3$$

- Hence the expected time is $O(n)$.

**Space:**

- Since the space for smaller hash tables is reclaimed, the number of buckets used is bounded by the number of buckets in the final table, which is $2N < 2n$.

- Space for the $n$ linked list vertices used to store the $n$ items in the buckets is $O(n)$, and hence the total space used is $O(n)$.