Improving efficiency

- Combining binary search with merge sort very efficient
  - If we search list k times, then efficiency is n*log(n) + k*log(n)
- Can we do better?
- Dictionaries use concept of hashing
  - Lookup can be done in time almost independent of size of dictionary

Hashing

- Convert key to an int
- Use int to index into a list (constant time)
- Conversion done using a **hash function**
  - Map large space of inputs to smaller space of outputs
  - Thus a many-to-one mapping
  - When two inputs go to same output – a **collision**
  - A good hash function has a uniform distribution – minimizes probability of a collision

Complexity

- If no collisions, then O(1)
- If everything hashed to same bucket, then O(n)
- But in general, can trade off space to make hash table large, and with good function get close to uniform distribution, and reduce complexity to close to O(1)