From sets, toward maps, via big-Oh

- **Consider the MemberCheck weekly problem**
  - Find elements in common to two vectors
  - Alphabetize/sort resulting vector, no duplicates

- **Write/code an algorithm, use vectors/algorithms, but no sets**
  - Goal: easy to write and verify as correct
  - Goal: efficient (what does this mean?)

- **How do we measure efficiency of code and data structures?**
  How do we measure "easy to write and verify"?
Reasoning about tradeoffs

- Consider intersection implementations in `membercheck.cpp`
  - Which was easier to write?
  - Which is easier to verify as correct?
  - Which is more efficient?

- `tset<string>`, simple templated set implementation
  - Uses an unbalanced search tree map internally
  - $O(\log n)$ operations on average, but $O(n)$ worst case
    - Insert, search, delete
  - Object of study, not meant as industrial strength
Tradeoffs in listunique.cpp

- **Represent a set as an unsorted vector**
  - Complexity of insert? Complexity of search?

- **Represent a set as an unsorted linked list**
  - Complexity of insert? Complexity of search?

- **Represent a set as a vector of 256 linked lists**
  - Complexity of insert? Complexity of search?
C++ coding issues in class Unique

- Why are all methods but add-a-vector pure virtual?
  - What does pure virtual mean?

- How is helper function in ListUnique used in subclass?
  - Probably no reason to implement originally
  - Refactoring when we see duplicated code
  - Opportunity for getAll refactoring?

- Why is node class protected, why is helper class after it?
  - What does public need to know? Scope.
solving recurrence: findkth

\[ T(n) = T(n/100) + n \quad T(XXX) = T(XXX/100) + XXX \]
\[ T(1) = 1 \]

(note: \( \frac{n}{100}/100 = \frac{n}{10^4} \))

\[ T(n) = [T(n/10^4) + n/100] + n \]
\[ = [T(n/10^6) + n/10^4] + n/100 + n \]
\[ = \ldots \text{eureka!} \]
\[ = T(1) + \text{sum of geometric series} \]
\[ = n + n/100 + n/100 \times 100 + \ldots = n(1 - 1/100^m)/(1/100) \]
\[ = O(n) \]
merge sort linked lists:  

- Given a linked list, we want to sort it
  - Divide the list into two equal halves
  - Sort the halves
  - Merge the sorted halves together

- What’s complexity of dividing an n-node in half?
  - How do we do this?

- What’s complexity of merging (zipping) two sorted lists?
  - How do we do this?

- \( T(n) = \text{time to sort n-node list} = 2 \cdot T(n/2) + O(n) \)
  - why?
sidebar: solving recurrence

\[ T(n) = 2T(n/2) + n \]
\[ T(1) = 1 \]

\[ T(n) = 2\left[2T(n/4) + n/2\right] + n \]
\[ = 4 \ T(n/4) + n + n \]
\[ = 4\left[2T(n/8) + n/4\right] + 2n \]
\[ = 8T(n/8) + 3n \]
\[ = \ldots \text{ eureka!} \]
\[ = 2^k T(n/2^k) + kn \]

let \( 2^k = n \)

\[ k = \log n, \text{ this yields } 2^{\log n} T(n/2^{\log n}) + n(\log n) \]
\[ n T(1) + n(\log n) \]
\[ O(n \log n) \]
Nancy Leveson: Software Safety

Founded the field

- Mathematical and engineering aspects
  - Air traffic control
  - Microsoft word

"C++ is not state-of-the-art, it's only state-of-the-practice, which in recent years has been going backwards"

- Software and steam engines: once extremely dangerous?
  - [http://sunnyday.mit.edu/steam.pdf](http://sunnyday.mit.edu/steam.pdf)
- THERAC 25: Radiation machine that killed many people