2019-09-19 Linked Lists

Thursday, September 19, 2019 9:11 AM

What do we know?

- Lots of pointers (at least one pointer per item in LL)
  - Linear relationship between size and # pointers
  - In Vector, there's only 1 pointer (pointer to dynamic array)
- Advantages of LL:
  - Good for pedagogical (teaching) about pointers
  - Dynamic resize (question: is this better or worse than vector?)
  - Can be templated
  - If you happen to have a pointer to where you want to insert or delete, very easy to do so
- Disadvantages of LL:
  - No random access
    - Can't use [], can't jump between element 0 and 10 easily
  - Takes up more memory relative to vector
    - In each LL box, there minimally must be
      - A box for storing data (INT: 4 bytes)
      - A pointer to the next box (min: 4 bytes)
    - A LL is more memory efficient than a vector that is less than half full
      - A vector should never be less than half full
  - It is common to use LLs as a secondary data structure within an more complex data structure
    - E.g. hash tables / dictionaries
  - Requires management of dynamic memory

vector fullness

$\frac{2x}{2} \leq \frac{?x}{2}$

Time Complexities of LL operations

- Insert (push_back)
  - O(1) if you have a pointer to back of LL (typical)
  - O(N) otherwise
- Random insert
  - O(N) if we don't have a pointer to the Nth slot (typical)
  - O(1) if somehow we magically have a pointer to the Nth slot
    - A skip list attempts to do this at O(LogN)
- Removals
  - O(N) random remove
  - O(1) remove first or remove last