February 7, 2025

LEC 13 - Linked List Operation Run Time

Recap

- Python lists are array based
- Each list store the ids of its elements in a contiguous block of memory
- Every insertion and deletion causes every element after the changed index to move

---

Linked List Operations

- Removing and inserting elements is much faster
  - No need to shift all elements after the index

Linked List Vs. Array Lists

At which end would it be best to insert and remove an element at?

Array Lists → End of the list
Linked Lists → Anywhere in the list

Run Time

- When analyzing running time, we use Big-Oh notation to capture the type of growth of running time as a function of input size
  - Ex. O(1) → Constant growth, O(n) → Linear growth
- Running time can vary, even for a fixed input size
  - Ex. edge cases that allow the function to terminate very quickly

---

Other Linked List Designs

Consider storing more information about the list:

- _first
- _last
- _size

1. Does the implementation of operations change
   - Yes, since when mutating the list, we will need to update _size as we add or remove elements
2. What are the performance implications
   - It will now be the slowest to traverse to the middle of the list
   - More memory is needed

February 7, 2025

LEC 13 - Linked List Operation Run Time

Other Ways to Organize Nodes

- We have so far use linked lists to organize nodes
- We can use doubly linked lists
    - Store the next and previous nodes
    - Allows us to make circular lists
- We can use a hierarchical structure (Tree)
    - Search path to each item in a tree is much shorter than in a linked list (assuming the tree is reasonably balanced)