Genome 540 discussion

January 30th, 2025

Joe Min

Genome Sciences
UNIVERSITY OF WASHINGTON

Agenda

Evaluating data structure performance

Common data structures

Homework 3 questions?

Evaluating data structures

What is a data structure?

Application-level idea way to structure data in such a way that it optimizes the storage of physical data as well as the efficiency of certain data operations:

- Creating and adding new data
- Retrieving existing data
- Updating or sorting existing data
- Deleting existing data

Evaluating data structures

Storage of data and data operations are evaluated using space and time complexity

- Reminder: big O notation defines how something grows with respect to a growing input (with some coefficient)
  - O(1) stays fixed and is independent of the size of the input
  - O(n) grows linearly with the size of input
  - O(log(n)) grows at a rate bound by the log of the size of the input

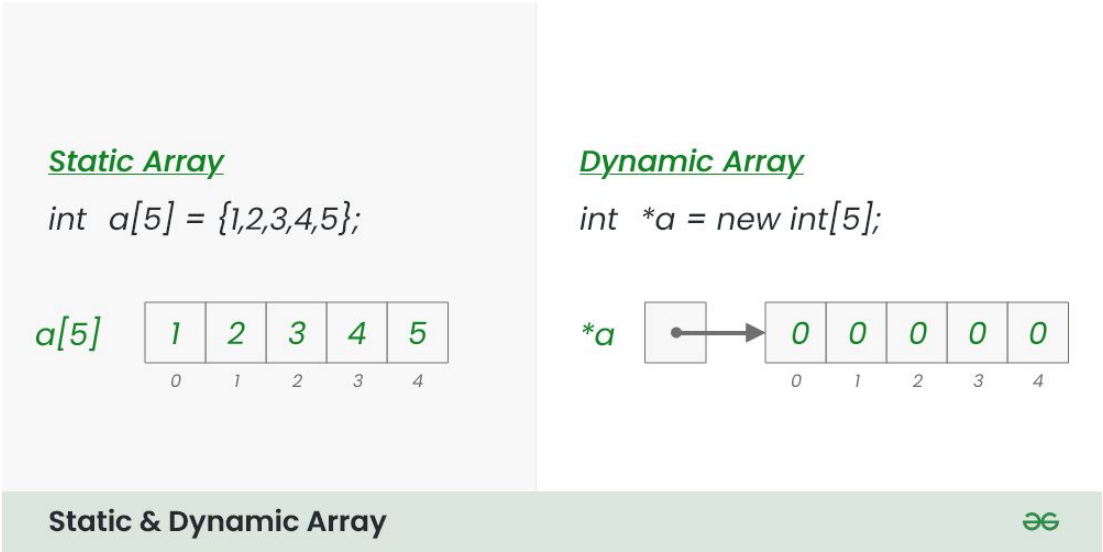
Evaluating data structures

Efficiency complexities are dependent on input, so we generally bin performance:

- Average complexity
- Worst case complexity
- Amortized complexity

Comparing static and dynamic lists

To better understand how to evaluate data structures, let's compare static and dynamic lists



https://media.geeksforgeeks.org/wp-content/uploads/20230822183342/static.png

Comparing static and dynamic lists: overhead space

Initial creation

- Statically-sized structures (e.g., static lists) incur a large overhead as they pre-allocate space
  - O(n) space complexity
- Dynamically-sized structures (e.g., a dictionary or dynamic list) instead have little space overhead
  - O(1) to make the first entry

Comparing static and dynamic lists: adding new data

Static lists

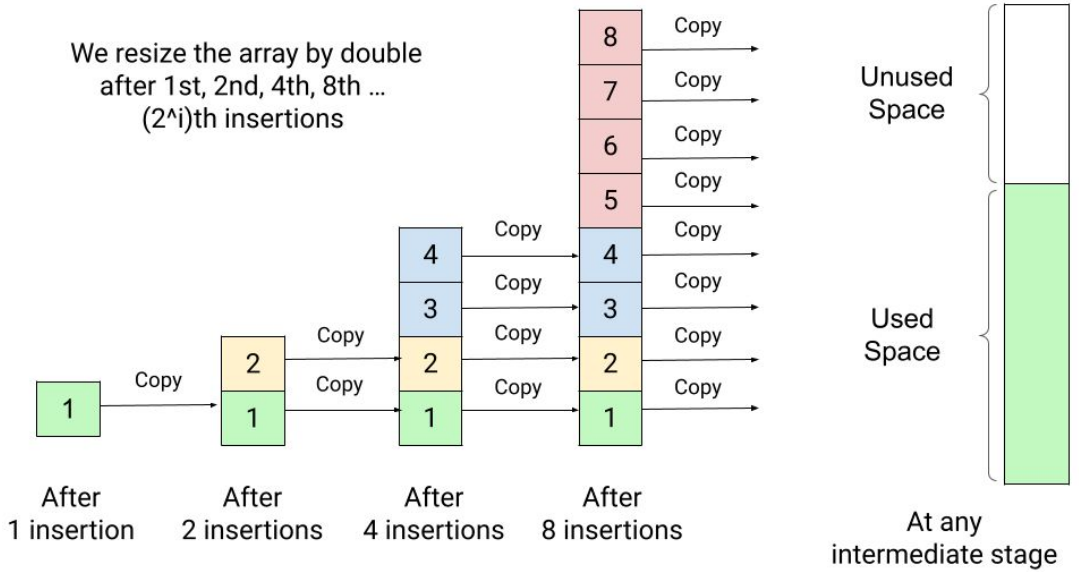
- Space is pre-allocated, so it doesn't matter how many spots are full, we can just put it in the next open spot
- O(1) time complexity

Dynamic lists

- If there are open spots, O(1); if not, it's O(n)
- Implementation can make the amortized cost O(1)

Dynamic list data addition algorithm

By doubling the allocated space whenever the list is full, we can reduce the number of allocate+copy events that are O(n) each

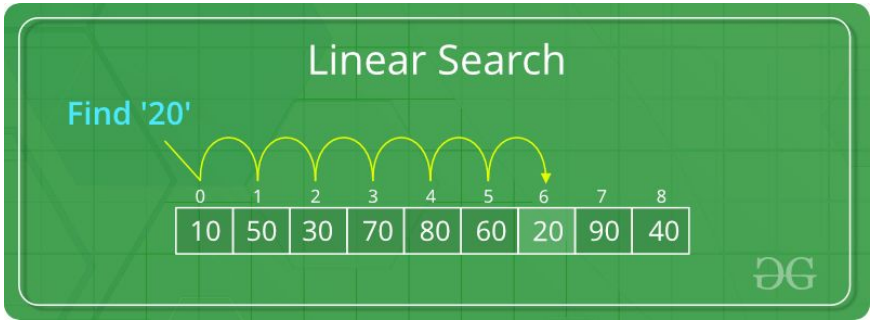


https://cdn-images-1.medium.com/max/960/1*9s7_mGUIzA_JcOOw-zQh9Q.png

Comparing static and dynamic lists: retrieving data

With a known index, data is retrieved from both lists in O(1) time by going to 1 physical spot in memory

When searching for an index with data of a specific value, both lists can do so in O(n) time



https://media.geeksforgeeks.org/wp-content/cdn-uploads/Linear-Search.png

Comparing static and dynamic lists: modifying data

To update the values of existing entries, time complexity is the same for data modification as it was for data retrieval

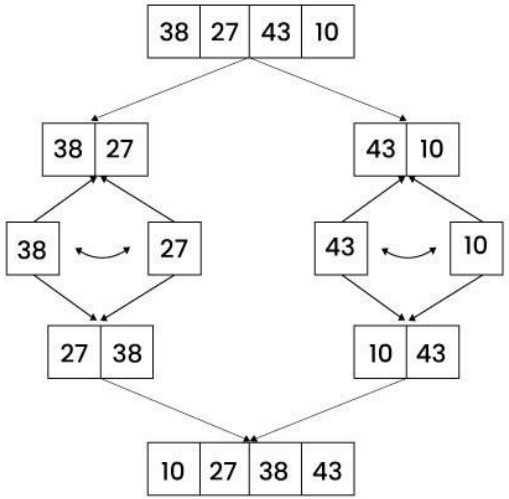
- Index known: O(1)
- Search required: O(n)

Comparing static and dynamic lists: sorting data

Both static and dynamic lists can be sorted in O(n*log(n)) time using optimized search algorithms

- E.g., merge sort or quick sort

Merge sort gets its log(n) by continuously halving the input into O(1) sortable chunks (i.e., a single comparison)



https://media.geeksforgeeks.org/wp-content/uploads/20240221173657/Merge-Sort.jpg

Comparing static and dynamic lists: deleting data

To delete the values of existing entries, time complexity is the same for data modification as it was for data retrieval

- Index known: O(1)
- Search required: O(n)

This is only possible because lists allow for empty indices; if we had to dynamically shrink to the size of the data every time, this would be much more expensive

Common data structures

Linked list

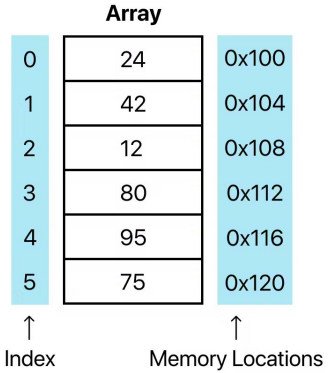
Creation overhead: O(1)

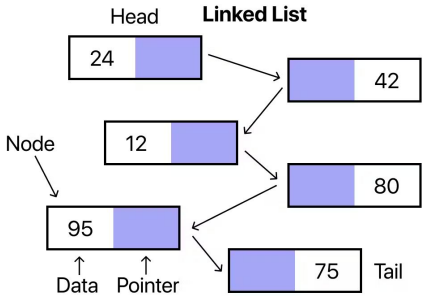
Adding data: O(n)

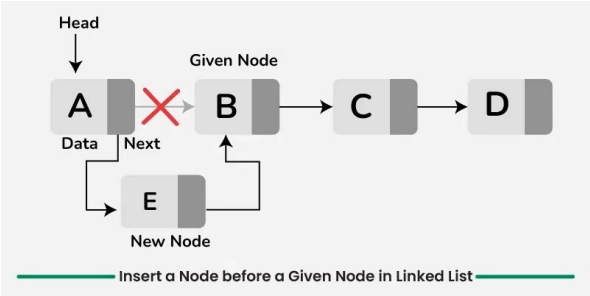
Finding data: O(n)

Sorting: n/a or free if handled during insertions

Deleting data: same as finding







https://cdn.hashnode.com/res/hashnode/image/upload/v1637603328903/E1PjE0gz9.jpeg
https://www.geeksforgeeks.org/insertion-in-linked-list/

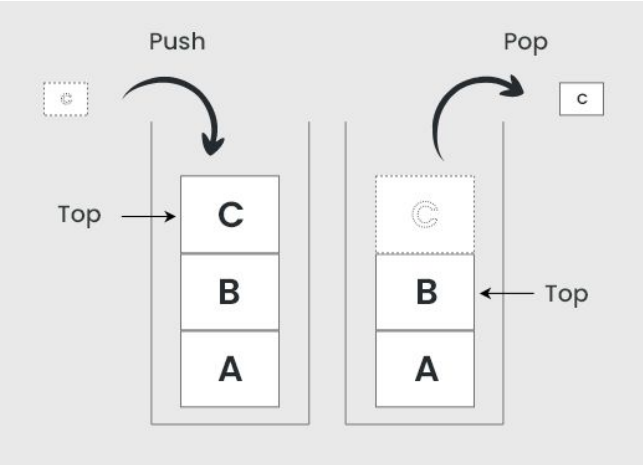
Stack

Stacks are designed to keep a specified ordering of data called Last-In-First-Out. You probably won't ever sort or change the value of an element in a stack

Creation overhead: O(1)

Adding data: O(1)

Deleting data: O(1)



https://media.geeksforgeeks.org/wp-content/cdn-uploads/20230726165552/Stack-Data-Structure.png

Dictionaries

Creation overhead: O(1)

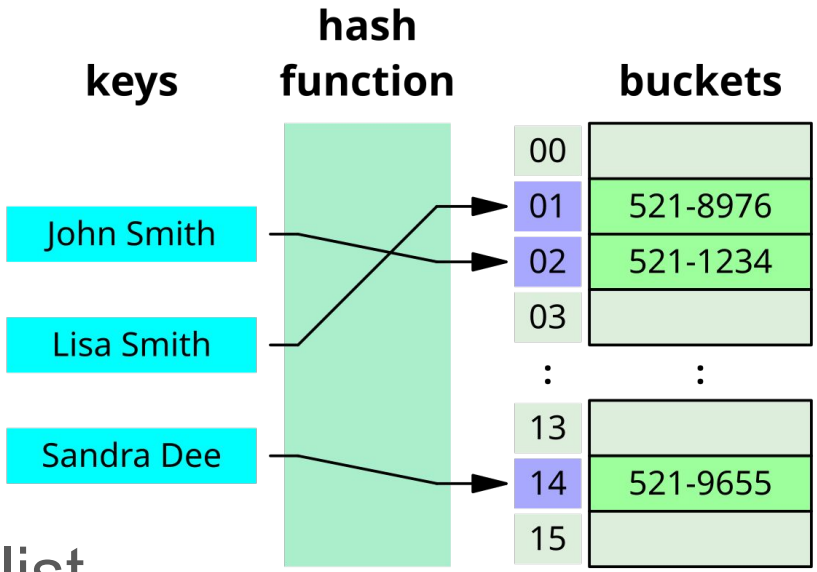
Adding data: O(1)

Updating data: O(1)

Sorting: n/a or O(nlog(n))

- Have to first get values as a list

Deleting data: O(1)



https://upload.wikimedia.org/wikipedia/commons/thumb/7/7d/Hash_table_3_1_1_0_1_0_0_SP.svg/1200px-Hash_table_3_1_1_0_1_0_0_SP.svg.png

Binary search trees

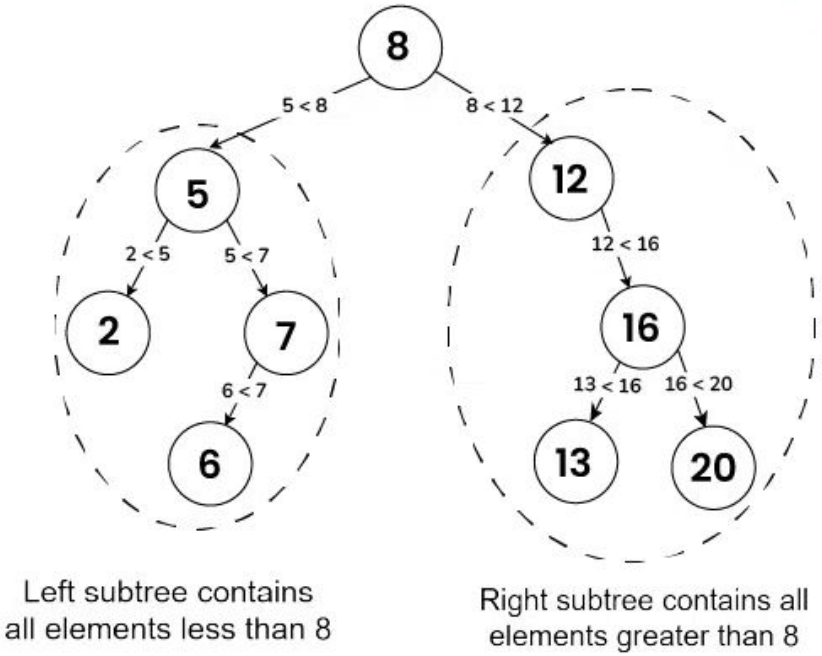
Creation overhead: O(1)

Adding data: O(h)

Finding data: O(h)

Sorting: Free!

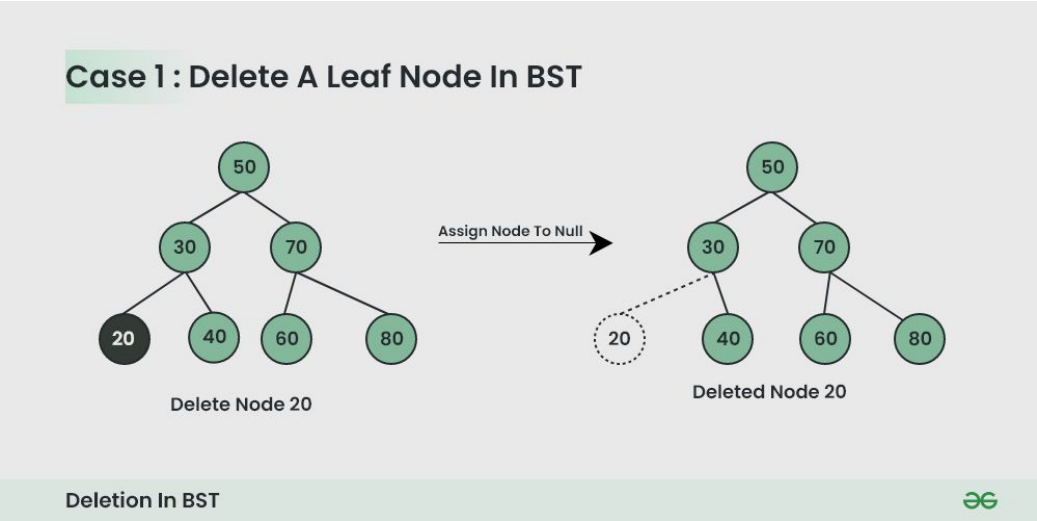
Deleting data: O(h)



https://www.geeksforgeeks.org/introduction-to-binary-search-tree/

Binary search trees

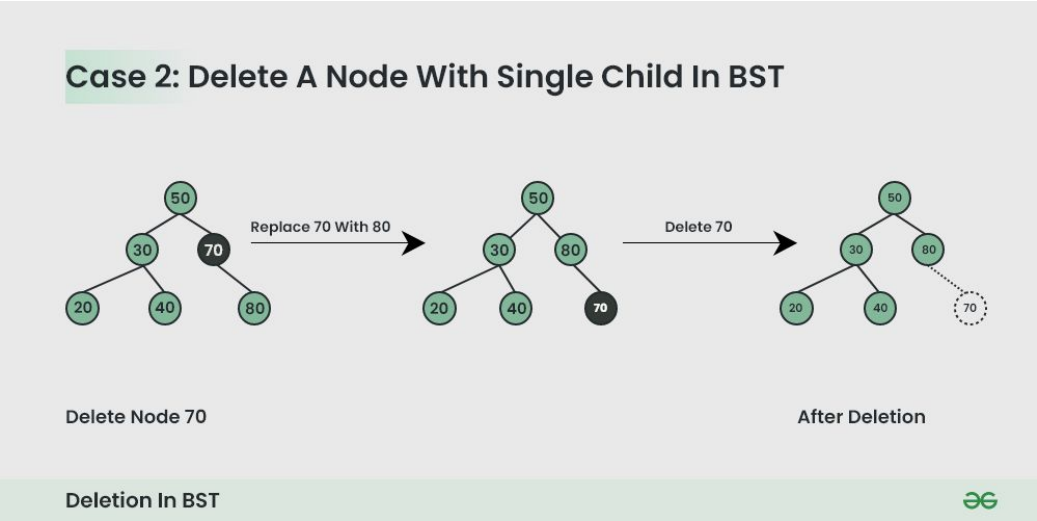
Deleting can be complicated due to the state of the tree:



https://www.geeksforgeeks.org/deletion-in-binary-search-tree/

Binary search trees

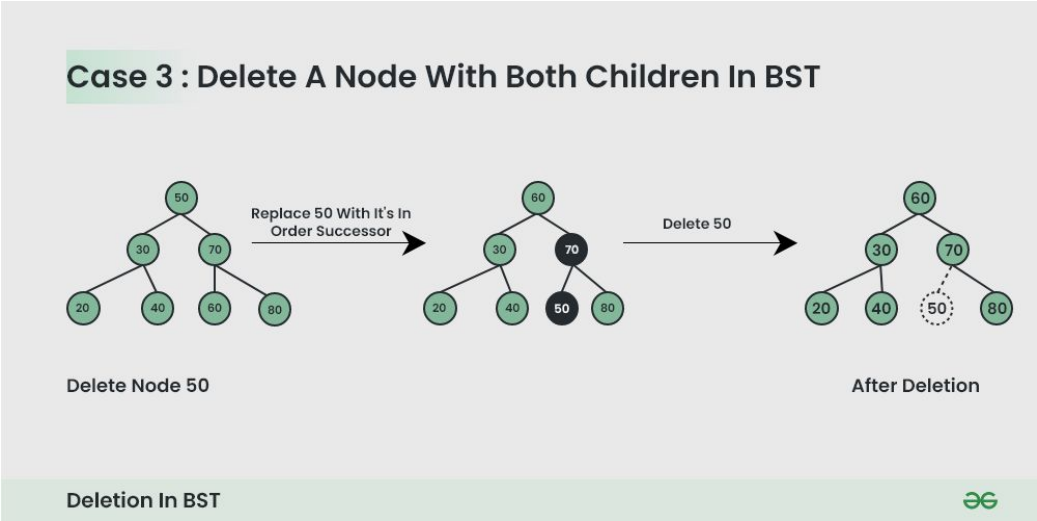
Deleting can be complicated due to the state of the tree:



https://www.geeksforgeeks.org/deletion-in-binary-search-tree/

Binary search trees

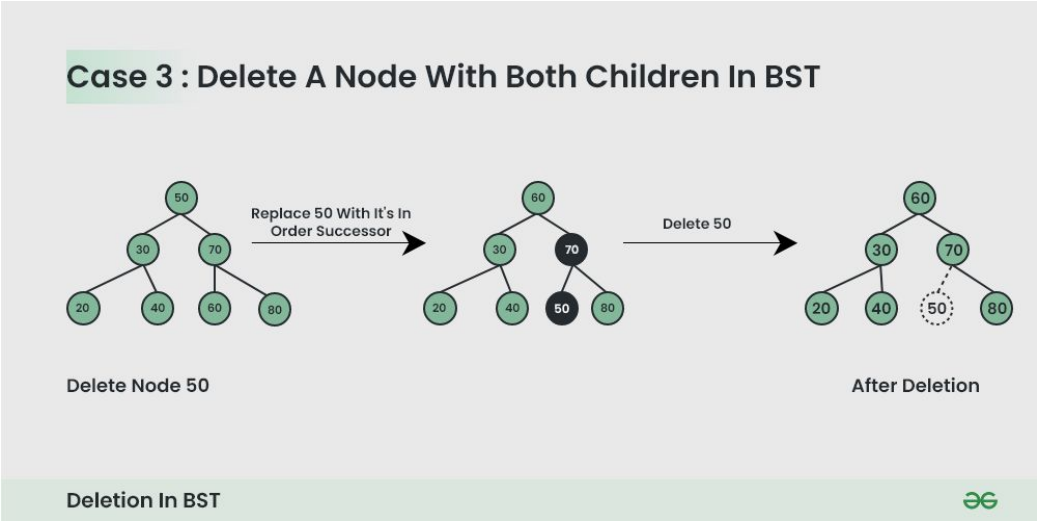
Deleting can be complicated due to the state of the tree:



https://www.geeksforgeeks.org/deletion-in-binary-search-tree/

Binary search trees

Deleting can be complicated due to the state of the tree:



…but is still O(1) by itself! So the overall is still O(h)

https://www.geeksforgeeks.org/deletion-in-binary-search-tree/

Other common data structures

Queues

Heaps

Trees

Tries

Graphs

How to pick a data structure

Built-in structures are very versatile and general-use

Specific tasks might require specific data structures

- E.g., if you want to keep track of landmarks on a walk so you can find your way back, you may want a stack to find your next backward landmark in O(1) time

You can also build your own custom types!

- More on implementations on Tuesday

Homework 3 questions?

Office hours

Reminder:

Homework 3 is due Sunday, February 2nd at 11:59pm!