Recitation 10

Prelim Review

Big O

See the Study Habits Note @282 on the course Piazza. There is a 2-page pdf file that says how to learn what you need to know for O-notation.

Big O definition

\[ f(n) = O(g(n)) \]

If there is a positive constant \( c \) and a real number \( N \) such that:
\[ f(n) \leq c \cdot g(n) \text{ for } n \geq N \]

Is merge sort \( O(n^2) \)?

Yes, but not tightest upper bound

Review: Informal Big O rules

1. Usually: \( O(f(n)) \cdot O(g(n)) = O(f(n) \cdot g(n)) \)
   - Such as if something that takes \( g(n) \) time for each of \( f(n) \) repetitions . . . (loop within a loop)
2. Usually: \( O(f(n)) + O(g(n)) = O(\max(f(n), g(n))) \)
   - \( \text{“max” is whatever’s dominant as } n \text{ approaches infinity} \)
   - Example: \( O((n^4-n)/2) = O(1/2n^4 + (-1/2)n) = O(1/2n^4) = O(n^4) \)
3. Why don’t logarithm bases matter?
   - For constants \( x, y \): \( O(\log(n)) = O(\log_x(n)) \)
   - Since \( \log_x(x) = 1 \), \( O(\log_x(n)) = O(\log(n)) \)

Test will not require understanding such rules for logarithms

Review: Big O

Is used to classify algorithms by how they respond to changes in input size \( n \).

Important vocabulary:
- Constant time: \( O(1) \)
- Logarithmic time: \( O(\log n) \)
- Linear time: \( O(n) \)
- Quadratic time: \( O(n^2) \)

Let \( f(n) \) and \( g(n) \) be two functions that tell how many statements two algorithms execute when running on input of size \( n \). \( f(n) \gg 0 \) and \( g(n) \gg 0 \).

1. \( \log(n) + 20 \) is \( O(\log(n)) \) (logarithmic)
2. \( n + \log(n) \) is \( O(n) \) (linear)
3. \( n^2 \text{ and } 3n \) are \( O(n) \)
4. \( n \cdot \log(n) + n \) is \( O(n \cdot \log(n)) \)
5. \( n^2 + 2^n + 6 \) is \( O(2^n) \) (cubic)
6. \( n^2 + n^3 \) is \( O(n^3) \) (cubic)
7. \( 2^n + n^5 \) is \( O(2^n) \) (exponential)
Review: Big O examples

1. What is the runtime of an algorithm that runs insertion sort on an array \(O(n^2)\) and then runs binary search \(O(\log n)\) on that now sorted array?

2. What is the runtime of finding and removing the fifth element from a linked list? What if in the middle of that remove operation we swapped two integers exactly 100000 times, what is the runtime now?

3. What is the runtime of running merge sort 4 times? \(n\) times?

Binary Search Trees

Heaps

Array Representation of Binary Heap

Review: Binary heap

PriorityQueue
- Maintains max or min of collection (no duplicates)
- Follows heap order invariant at every level
- Always balanced!
- worst case:
  - \(O(\log n)\) insert
  - \(O(\log n)\) update
  - \(O(1)\) peek
  - \(O(\log n)\) removal
Review: Binary heap

How do we insert element 0 into the min heap?
After we remove the root node, what is the resulting heap?

How are heaps usually represented? If we want the right child of index i, how do we access it?

Review: Hashing

<table>
<thead>
<tr>
<th>Method</th>
<th>Expected Runtime</th>
<th>Worst Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>O(1)</td>
<td>O(n)</td>
</tr>
<tr>
<td>contains</td>
<td>O(1)</td>
<td>O(n)</td>
</tr>
<tr>
<td>remove</td>
<td>O(1)</td>
<td>O(n)</td>
</tr>
</tbody>
</table>

Collision resolution

Two ways of handling collisions:
1. Chaining
2. Open Addressing
Load factor: $b$’s saturation

Load factor: $\lambda = \frac{\text{# of entries}}{\text{length of array}} = \frac{3}{6}$

Using linear probing to resolve collisions,

1. Add element SC (hashes to 9).
2. Remove VA (hashes to 3).
3. Check to see if MA (hashes to 21) is in the set.
4. What should we do if we override equals()?

Graphs

1. Starting from node A, run BFS and DFS to find node Z. What is the order that the nodes were processed in? Visit neighbors in alphabetical order.
2. What is the difference between DFS and BFS?
3. What algorithm would be better to use if our graph were near infinite and a node was nearby?
4. Is Dijkstra’s more like DFS or BFS? Why?
5. Can you run topological sort on this graph?

Topological ordering

All edges go from a smaller-numbered node to a larger-numbered node. How can this be useful?

Dijkstra’s Algorithm

The nodes are numbered in the order they are visited if we start at 1. Why are they visited in this order?