Review Questions

1. Why did we introduce tree data structures as an alternative to linear data structures?
   a. Some complex data can only be represented with a tree data structure.
   b. Some simple data can only be represented with a tree data structure.
   c. Linear data structures cannot implement sets and maps.
   d. Tree data structures are typically more effective at implementing sets and maps.
2. How does the graph abstract data type differ from other abstract data types?
   a. It can model the relationships between elements.
   b. It is more efficient than other abstract data types.
   c. It can solve problems that other abstract data types cannot solve.
   d. It does not specify a particular data structure implementation.
3. What abstract data type do binary heaps most commonly implement?
   a. lists
   b. sets
   c. maps
   d. priority queues
4. What is one way to describe the relationship between algorithms, problems, and modeling?
   a. Algorithms are the foundation for problem models.
   b. Algorithms solve a model of a problem.
   c. Each algorithm can only be used to solve a single problem.
   d. Each problem can only have a single model.
5. At what point do computer scientists apply algorithm design patterns?
   a. when learning canonical algorithms
   b. when modeling a problem
   c. when solving new problems
   d. when analyzing an algorithm

6. How does the model of computation relate to the problem model?
    a. The model of computation is synonymous with problem model.
    b. Problem models constrain the model of computation.
    c. The model of computation constrains the problem modeling process.
    d. The model of computation describes a single algorithm for each problem model.
7. Why is case analysis important?
    a. Case analysis provides an alternative to asymptotic analysis.
    b. Case analysis focuses on small inputs.
    c. Case analysis simplifies the step-counting by introducing a cost model.
    d. Case analysis considers factors other than the size of the problem.
8. What is true about the order of growth of binary search with respect to the size of the sorted list?
    a. In the best case, the order of growth of binary search is constant.
    b. In the best case, the order of growth of binary search is logarithmic.
    c. In the worst case, the order of growth of binary search is constant.
    d. In the worst case, the order of growth of binary search is linear.
9. How does time complexity relate to space complexity?
    a. Time complexity measures efficiency according to the size of the problem, while space complexity does not.
    b. Both time and space complexity measure the efficiency of algorithms as they relate to the nature of the problem.
    c. Time complexity focuses on asymptotic analysis while space complexity focuses on experimental analysis.
    d. Both time and space complexity can apply methods from asymptotic analysis and experimental analysis.
10. What are the three steps in divide and conquer algorithms?
11. Why do many greedy algorithms fail to compute the best solution to a combinatorial problem?
12. What are the three steps in reduction algorithms?

13. What is quicksort's algorithm design pattern and algorithmic paradigm?

    a. Quicksort is an application of the binary search tree algorithm design pattern and an example of the divide and conquer algorithmic paradigm.

    b. Quicksort is an application of the binary search tree algorithm design pattern and an example of the brute-force algorithmic paradigm.

    c. Quicksort is an application of the binary search tree algorithm design pattern and an example of the greedy algorithmic paradigm.

    d. Quicksort is an application of the binary search tree algorithm design pattern and an example of the randomized incremental construction algorithmic paradigm.

14. What graph problems can breadth-first search solve?

    a. exponential node tree

    b. minimum spanning trees

    c. unweighted shortest paths

    d. weighted shortest paths

15. What is a primary drawback of hashing?

    a. There can be collisions as the same element can hash to multiple values.

    b. There can be collisions between multiple elements that hash to the same value.

    c. Hashing is slower than binary search for search problems.

    d. Hashing is faster than binary search for search problems.

16. What is the relationship between Turing machines and models of computation?

17. What is one of the three key ideas of the Turing machine?

    a. infinite memory by virtualization

    b. a memory bank for storing data

    c. using divide and conquer to always reduce an algorithm to O(n) runtime

    d. using divide and conquer to always reduce an algorithm to O(1) runtime

18. What is P versus NP?

    a. P refers to the polynomial time complexity class, whereas NP refers to the nondeterministic polynomial time complexity class.

b. P refers to any Big O notation past $O(N^3)$, whereas NP refers to any Big O notation less than $O(N^3)$.

c. NP is a constant runtime, whereas P is polynomial runtime.

d. P refers to constant runtime, whereas NP is linear runtime.

Conceptual Questions

1. Explain the difference between algorithms and programs.
2. Explain the difference between data structures and abstract data types.
3. Explain the relationship between data representation, data structures, and algorithms.
4. What is the relationship between search algorithms and the searching problem?
5. What is the relationship between search algorithms and the autocomplete feature?
6. Why is algorithmic correctness difficult to determine?
7. What are some limitations of experimental analysis?
8. What are some benefits of experimental analysis over asymptotic analysis?
9. Why is a 1-element list the best-case situation for sequential search?
10. If phone numbers are ten digits long and can contain digits from zero through nine, what is the total number of potential phone numbers?
11. Why might we prefer a sub-optimal greedy algorithm over a correct brute-force algorithm?
12. What’s problematic about the statement, "municipal broadband planning reduces to Kruskal’s algorithm"?
13. Describe the relationship between the pivot element and the left and right sublists after the first partition in quicksort.
14. The runtime of Kruskal’s algorithm is in $O(|E| \log |E|)$ with respect to $|E|$, the number of edges. What primarily contributes to this linearithmic order of growth?
15. Both Prim’s algorithm and Dijkstra’s algorithm are greedy algorithms that organize vertices in a priority queue data structure. What is the difference between the ordering of vertices in the priority queue for Prim’s algorithm and Dijkstra’s algorithm?
16. What is the relationship between models of computation and algorithms?

17. What is the common difficulty preventing us from designing an efficient algorithm for solving NPcomplete problems?

18. What are the consequences of P = NP?

Practice Exercises

1. Binary search trees organize elements in ascending sorted order within the tree. However, binary search trees can become unbalanced. In the worst-case, a binary search tree can look exactly like a linked list. Give an order for inserting the following numbers into a binary search tree such that the resulting tree appears like a linked list: 7, 3, 8, 1, 2, 5, 6, 4.

2. Consider these two different approaches for implementing the priority queue abstract data type using a linked list data structure: (1) organize the elements by decreasing priority value, and (2) organize the elements arbitrarily. Describe algorithms for inserting an element as well as retrieving and removing the highest-priority element from these two data structures.

3. In our definition of a priority queue, we emphasized retrieval and removal of the highest-priority elements—a maximum priority queue. What if we wanted to instead prioritize retrieval and removal of the lowest-priority elements—a minimum priority queue? Describe a simple change that we could make to make any maximum priority queue function as a minimum priority queue.

4. Formally describe the problem model for a drug administration medical system in terms of input data and output data represented as lists, sets, maps, priority queues, and graphs.

5. Formally describe the problem model for a music recommendation system in terms of input data and output data represented as lists, sets, maps, priority queues, and graphs.

6. There can sometimes be thousands, if not millions, of results that match a Web search query. To make this information more helpful to humans, we might want to order the results according to a relevance score such that more-relevant results appear before less-relevant results. Describe how we can solve this problem of retrieving the N-largest elements using the following algorithm design patterns: (1) a sorting algorithm, and (2) a priority queue abstract data type.

7. What is the best-case and worst-case Big O order of growth of sequential search with respect to N, the size of the list?

8. What is the best-case and worst-case Big O order of growth of binary search with respect to N, the size of the sorted list?

9. What is the worst-case Big O order of growth of sequential search followed by binary search with respect to N, the size of the sorted list?

10. What two sublists are combined in the final step of merge sort on the list [9, 8, 2, 5, 4, 1, 3, 6]?

11. If a connected graph has unique edge weights, will Kruskal's algorithm find the same minimum spanning tree as Prim's algorithm? How about a connected graph with duplicate edge weights?

12. What is a reduction algorithm for the problem of finding the median element in a list?

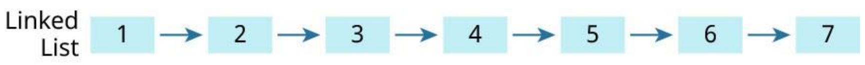
13. The heapsort algorithm applies a binary heap priority queue ordered by the comparison operation to sort elements. What can we say about the first element removed from the binary heap if it implements a minimum priority queue? What about the last element removed? Is the binary heap data structure sorted?

14. Hashing algorithms can provide a constant-time solution to the search problem under certain conditions. What are the conditions necessary to ensure the runtime of a hashing search algorithm is constant?

15. Why is it the case that depth-first search cannot be directly applied to compute an unweighted shortest paths tree?

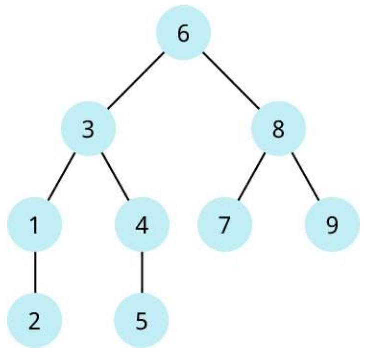
Problem Set A

1. Linked lists and binary search trees are two examples of linked data structures, in which each node in the data structure is connected to other nodes. Given a linked list of the numbers one through seven, organized in ascending sorted order, draw two different examples of binary search trees representing the same numbers.



(attribution: Copyright Rice University, OpenStax, under CC BY 4.0 license)

2. Given the following binary search tree, draw the corresponding ascending-sorted linked list containing the same numbers.



(attribution: Copyright Rice University, OpenStax, under CC BY 4.0 license)

3. We defined autocomplete as a problem that takes as an input a string of letters that might represent the start of a word, and outputs a list of words that match the input. Describe two other problem models for autocomplete and define the ramifications of the models.

4. Compare and contrast the autocomplete problem models. What are the trade-offs of each problem model?

5. Consider the problem of arranging desktop icons in a grid layout so that they fill column-by-column, starting from the left side of the screen. Suppose we are given a list of desktop icons in alphabetical order and that placing an icon in a grid position is a constant-time operation. What is the Big O order of growth of an algorithm that takes each icon in the given order and places each icon in the next open grid position with respect to N, the number of desktop icons?

6. Suppose we're given a list of desktop icons in alphabetical order, but that placing an icon in a grid position is a linear-time operation with respect to the number of icons. (Perhaps a sequential search is needed to check that the icon has not already been placed on the desktop.) What is the Big O order of growth of this icon arrangement algorithm with respect to N, the number of desktop icons?

7. Why does the greedy interval scheduling, which selects the cheapest, least time-consuming task, fail to maximize the number of completed tasks?

8. What is a simple rule for greedy interval scheduling that will maximize the number of completed tasks?

9. The runtime of most binary heap priority queue operations is in O(log N) with respect to N, the size of the priority queue. The logarithmic factor is due to the height of the binary heap data structure. But finding an element in a binary heap typically requires sequential search. How can we apply the hashing algorithm design pattern to find an element in a heap in constant time?

10. The runtime of breadth-first search is in O(|V| + |E|) because each reachable vertex and edge is processed one-by-one during the level-order graph traversal.

Why is the runtime of Prim's algorithm in O(|E| log |V| + |V| log |V|)? Explain in terms of the time it takes to process each vertex or edge in the graph.

Problem Set B

1. Each unique key in a map is paired with one (possibly not-unique) value. Sometimes, we want to associate more than one value with a given key. Describe how we can use a list or set abstract data type to associate more than one value with a unique key.

2. Each vertex in a graph can be labeled with a unique identifier, such as a unique number. Describe the relationship between adjacent vertices. How could we use abstract data types such as lists, sets, and maps to represent these relationships?

3. Describe how we might implement the graph abstract data type using other abstract data types such as lists, sets, and/or maps. Explain for graphs whose edges have associated weights as well as graphs whose edges do not have associated weights.

4. Describe two or three different problem models for a medical system designed to help doctors recommend preventive care for patients.

5. Compare and contrast the medical system problem models. What are the trade-offs of each problem model?

6. How does the choice of problem model affect potential algorithms? Describe an algorithm for each problem model.

7. Consider an autocomplete implementation that relies on a sequential search to find all matching terms. What is the worst-case Big O order of growth for computing a single autocomplete query with respect to N, the number of potential terms?

8. Consider an autocomplete implementation that sorts the list of potential terms and then performs binary search to find the matching terms. If the order of growth of the sorting algorithm is in O(Nlog N), what is the worst-case Big O order of growth for computing a single autocomplete query with respect to N, the number of potential terms?

9. Why might algorithm designers prefer to use binary search instead of sequential search for autocomplete?

10. In a 1-D space where each point is defined with xcoordinates, a common problem is to find the closest pair of points in the space: the pair of points that has the least distance among all potential pairs of points. What is a brute-force algorithm for solving this 1-D closest pair problem? What is the Big O notation order of growth of this algorithm?

11. In a 2-D space, each point is defined with (x, y) coordinates. What is a brute-force algorithm for solving the 2-D closest pair problem?

12. What are the recursive subproblems in a divide and conquer algorithm for solving for the closest pair problem?

13. Digital images are represented in computers as a 2-D grid of colored pixels. In image editing, the flood fill problem takes a given starting pixel and replaces all the pixels in a contiguous region that share the same color with a different color. How can we represent the colored pixels in a digital image as a graph with vertices and edges for the flood fill problem?

14. How should we modify a graph traversal algorithm to solve the flood fill problem using your graph representation?

15. What is a reduction algorithm for reducing from the flood fill problem to the graph traversal problem? In this case, the graph traversal algorithm cannot be modified. Instead, define a preprocessing step to create a graph representation that encodes the flood fill same-color rule.

Thought Provokers

1. Maps can be defined in terms of sets: every map is a set whose elements represent key-value pairs, where the key must be unique, but the value might not be unique. Consider other relationships between abstract data types. Can sets and maps be defined in terms of graphs? Can lists be defined in terms of maps? Can priority queues be defined in terms of maps? Why might it be useful to define abstract data types in terms of other data types?

2. Graph theory refers to the mathematical study of graphs. How might a graph theorist describe linked lists and tree data structures? How does this differ from our use of abstract data types?

3. Sorting and searching are two examples of data structure problems related to the storage and retrieval of elements. Where do sorting and searching appear in linear data structures, tree data structures, and/or graph data structures?

4. What are some benefits and drawbacks of simpler problem models, as they compare to more complicated problem models?

5. The formal definition of Big O notation does not exactly match our working definition for orders of growth. Do some additional research to explain why binary search is also in O(N).

6. Since binary search is in O(N), it is also true that binary search is in $O(N^2)$. Explain why computer scientists might find $O(N^2)$ to be a less useful description of the runtime of binary search compared to O(log N).

7. We can show that the worst-case order of growth for any comparison sorting algorithm must be at least linearithmic using an argument from combinatorial explosion in the number of unique permutations of elements in a list. What are the number of unique permutations of a list with Nelements? How many comparison questions need to be asked to identify a particular permutation from among all the permutations? How do these questions relate to comparison sorting?

8. Breadth-first search is a fundamental algorithm design pattern for graph problems. How is breadth-first search applied as a foundation for designing greedy algorithms such as Prim's algorithm and Dijkstra's algorithm? How does Kruskal's algorithm fit into these algorithm design patterns and paradigms? If Prim's algorithm is analogous to sorting in Kruskal's algorithm, why is there no analogue to the Dijkstra's algorithm in sorting as well?

9. Suppose we want to find the longest path from a starting vertex to an ending vertex in a graph. How might a nondeterministic algorithm solve this problem in polynomial time?

10. Suppose we want to find the longest path from a starting vertex to an ending vertex in a graph (solving the function problem) without using a nondeterministic algorithm. Let's say that P = NP and we have a deterministic polynomial-time algorithm that returns whether there is a path with exactly cost k(solving the decision problem). We also know the cost of the actual longest path. How can we repeatedly apply this

decision algorithmto design a polynomial-time longest paths function algorithm?

Labs

1. Simulate patients entering and exiting a hospital emergency room with a priority queue using patients' time of arrival, basic assessment of severity, and availability of doctors specializing in the appropriate type of care. Decide how to prioritize patients based on a property of each patient, such as their arrival time, numeric severity rating, numeric urgency rating, and availability of care providers. Then, consider how your decision might result in unfair allocation of medical care to patients.

2. Choose a lab from the Ethical Reflections Modules (https://openstax.org/r/76Ethics) for CS1. Follow the instructions to complete the assignment. Once you have finished, answer this additional reflection question connecting back to algorithm design: How did you utilize algorithm design patterns? How did your choice of algorithm designs affect the end outcomes in your program?

3. Experimental analysis: Use a software-based “stopwatch” to compare the time (in microseconds) it takes to run a sequential search versus a binary search for successively larger and larger inputs. Relate experimental analysis to asymptotic analysis. What happens to small arrays? What happens to large-size array inputs? What happens when the target is near the front of the array? What happens when the target is near the end of the array?