Exam 1 Modules

Insertion Sort//**What is the running time of Insertion Sort when the input is an array that has already been sorted?**Θ(n)**In the worst case, the total number of comparisons for Insertion Sort is closest to:** n^2/2**In which cases are the growth rates the same for Insertion Sort?**Worst and Average only **Insertion Sort (as the code is written in this module) is a stable sorting algorithm. Recall that a stable sorting algorithm maintains the relative order of records with equal keys.**True **We know that the worst case for Insertion Sort is about n^2/2, while the average case is about n^2/4. This means that:** The growth rates are the same and the runtime in the average case is about half of that of the worst case **What is the average-case time for Insertion Sort to sort an array of n records?** Θ(n^2)**What is the best-case time for Insertion sort to sort an array of n records?**Θ(n)**When is Insertion Sort a good choice for sorting an array?**The array has only a few records out of sorted order **If I is the number of inversions in an input array of n records, then Insertion Sort will run in what time?**Θ(n+I)

Bubble Sort//**When is Bubble Sort a good choice for sorting an array?** There is no situation where Bubble Sort is the best choice over all of the others in this chapter **The order of the input records has what impact on the number of comparisons required by Bubble Sort (as presented in this module)?** None **Bubble Sort is a stable sorting algorithm. Recall that a stable sorting algorithm maintains the relative order of records with equal keys.** True **What is the running time for Bubble Sort when the input array has values that are in reverse sort order?** Θ(n^2) **What is the best-case time for Bubble Sort to sort an array of n records?** Θ(n^2) **In which cases are the time complexities the same for Bubble Sort?** Worst, Average, and Best **What is the running time of Bubble Sort when the input is an array where all record values are equal?** Θ(n^2) **In the worst case, the total number of comparisons for Bubble Sort is closest to** n^2/2 **What is the worst-case time for Bubble Sort (as the algorithm is presented in this module) to sort an array of n records?** Θ(n^2)

Selection Sort//**Selection Sort is a stable sorting algorithm. Recall that a stable sorting algorithm maintains the relative order of records with equal keys.** False **In which cases are the time complexities the same for Selection Sort?** Worst, Average and Best **The order of the input records has what impact on the number of comparisons required by Selection Sort?** None. **In which cases are the time complexities the same for Selection Sort? In the worst case, the total number of comparisons for Selection Sort is closest to:** n^2/2 **How many times does Selection Sort call the swap function on an array of n records?** n-1 **Suppose that Selection Sort is given an input of 100 records, and it has completed 46 iterations of the main loop. How many records are now guaranteed to be in their final position (never to be moved again by the sort)?** 46 **What is the running time of Selection Sort when the input is an array that has already been sorted?** Θ(n^2) **What is the best-case time for Selection Sort to sort an array of n records?** Θ(n^2) **Selection sort is simple, but less efficient than the best sorting algorithms.** True **In the worst case, the total number of swaps done by Selection Sort is closest to:** n

Exchange Sort//**If I is the number of inversions in an input array of n records, then {Insertion|Bubble} Sort will require how many swaps?** I **How many ways can n distinct values be arranged?** n! **An inversion is:** When a record with key value greater than the current record's key appears before it in the array**The Average number of inversion in an array of n records is n(-1)/4. this is:** Θ(n2) **An exchange sort is:** Any sort where only adjacent records are swapped **The total number of pairs of records among n records is:** n(n-1)/2

Exam 1 Answers

1. Which of these is NOT one of the three standard steps to follow when selecting a data structure to solve a problem? **Run Simulations to quantify the expected running time of the program**

2. Which are examples of composite types?**A bank account record…A string.**

3. An algorithm is made up of two independent time complexities f(n) and g(n). An example might be
Algo1(m, n){
if m is odd then method1(n);
else method2(n);
//assume method1() and method2() don't change m
}
What is the time complexity of Algo1() using big-oh notation?**O(f(n) X g(n))**

4. Two main measures for the efficiency of an algorithm are **Time and space**

5. Consider a singly linked list of n nodes. It has a head pointer and a cur pointer. Cur points to a node that is under consideration. What is the time taken to delete the value of the list in the node pointed by cur? **O(1)**

6. Assume that an array consists of at least 10 elements. The worst case time (in the big-oh sense) occurs in linear search algorithm when **Item is not in the array at all…Item is the last element in the array…Item is somewhere in the middle of the array**

7. The recurrence relation

T(n) = 2 T(n/2) + c, if n >=2
T(n) = c1, otherwise
has the solution
O(n)

8. Which of the following topics have been covered in detail in the class since the start of the semester?**Binary search…Linear search…Analyzing recursive and iterative algorithms…Sequential model of computation (RAM)…Solving a one-variable recurrence relation**

9. Consider the following recurrence relation:
T(n) = 3 if n =1
T(n) = T(n-1) + 3 if n>=2
Using the expansion (aka iteration) method as discussed in class to solve and find close bound for the recurrence relations, What is the expansion of T(n) at step1 or the first step of recursion?**T(n-1) + 3**

10. Consider the following recurrence relation:
T(n) = 2 if n =1
T(n) = T(n-1) + 2 if n>=2
Using the expansion (aka iteration) method as discussed in class to solve and find close bound for the recurrence relations, What is the expansion of T(n) at step2? **T(n-2) +4**

11. Consider the following recurrence relation:
T(n) = 3 if n =1
T(n) = T(n-1) + 3 if n>=2
Using the expansion (aka iteration) method as discussed in class to solve and find close bound for the recurrence relations, what is the solution or closed-form for T(n)? **3n**

12. Given an array-based list implementation, deleting the current element takes how long in the average case?**Θ(n) time**

13. Given an array-based list implementation, inserting a new element to arbitrary position i takes how long in the average case?**Θ(n) time**

14. In the linked list, where does the insert method place the entry?**At the current position**

15. The physical order in memory for the nodes of a linked list is the same as the order in which the nodes appear in the list. **False**

16. Given a linked list implementation, inserting a new element to the current position takes how long?**Θ(1) time**

17. An advantage of linked lists over the array-based list implementation is: **Expandability**

18. Given a doubly linked list implementation, removing the current element takes how long in the average case?**Θ(1) time**

19. It is an error to pop data from a(n) _____ stack **Empty**

20. The following sequence of operations is performed on a stack: push(1), push(2), pop, push(1), push(2), pop, pop, pop, push(2), pop. The values will be output in this order:**22112**

21. Which data structure allows insertion only at the back, and removing only from the front?**Queue**

22. Which of the following is not a linear data structure?**None of the given choices, all are linear data structures.**

23. The term "FIFO" is associated with which data structure?**Queue**

24. A program is an instance of an algorithm implemented in a specific programming language. **True**

25. An algorithm can only work if it is written in the right type of programming language **False**

26. There are some algorithms that do not terminate for certain inputs **False**

27. A problem instance is a specific selection of values for the problem input.**True**

28. Which is true about the relationship between algorithms and problems?**A problem is mapping from inputs to outputs, and there might be many algorithms that can accomplish this mapping**

29. Suppose that a particular algorithm has time complexity T(n)=8n and that executing an implementation of it on a particular machine takes t seconds for n inputs. Now suppose that we are presented with a machine that is 64 times as fast. How many inputs could we process on the new machine in t seconds? **64n**

30. Suppose that a particular algorithm has time complexity T(n)=n^2 and that executing an implementation of it on a particular machine takes t seconds for n inputs. Now suppose that we are presented with a machine that is 64 times as fast. How many inputs could we process on the new machine in t seconds?**8n**

31. Hardware vendor XYZ Corp. claims that their latest computer will run 100 times faster than that of their competitor, Prunes, Inc. If the Prunes, Inc. computer can execute a program on input of size n in one hour, what size input can XYZ's computer execute in one hour for an algorithm whose growth rate is n^2?**10m**

32. For sequential search, the best case occurs when: **The search target is near the front of the array**

33. Which of these is the best upper bound for a growth rate of 5n+3? **O(nlogn)**

34. The Sequential Search algorithm is in O(n^2).**True**

35. Determine Θ for the following code fragment in the average case. Assume that all variables are of type "int".

```
sum = 0;
if (EVEN(n))
  for (i = 0; i < n; i++)
    sum++;
else
  sum = sum + n;
```

Θ(n)

36. Assuming "sort" takes nlogn time and "random" takes O(1)-time, what is the time complexity of the following code:

```
sum = 0;
for (i = 0; i < n; i++) {
  for (j = 0; j < n; j++)
    A[j] = random(n);
  sort(A);
}
```

Θ(n^2 * log n)

37. The lower bound for a problem is defined as the least cost that any algorithm could reach. **True**

38. The upper bound for a problem is defined as the upper bound cost for the best algorithm that we know. **True**

39. The worst case for sequential search occurs when the last element of the array is the value being searched for.
False

40. For all the algorithms for which we properly understand the running time analysis, the upper bound and lower bound will always be the same.
False

41. Determine Θ for the following code fragment in the average case. Assume that all variables are of type "int".

```
a = b + c;
d = a + e;
```

Θ(1)

42. Count the frequency of the method foo() in the fllowing code segment:

```
foo();
int i = 1;
while (i <= n) do
    for (int j=1; j<=i; j++)
        foo();
        i*=2;
   endfor
endwhile
```

First count the frequency, then write a closed form, then solve it if it contains summations or recurrence relation, finally express your answer in big-oh, big-theta or big-omega notation whichever fits best. **Infinity**

Exam 2 Modules

Shellsort//**If we Shellsort an array of length 9, and we start with an increment size of 5, which element (if any) will be skipped on the first pass due to being in a sublist of only one element?** 4 **In Shellsort, how are the sublists sorted?** Using InsertionSort because Shellsort generally makes each subarray close to sorted//////Radix Sort Bin Sort//**Radix Sort as implemented in this module would be a good solution for alphabetizing words.** False **Radix sort processes one digit at a time, and it uses a Binsort to sort the n records on that digit. However, any stable sort could have been used in place of the Binsort.** True **The worst-case cost for Radix Sort when sorting n keys with keys in the range 0 to r3-1 is:** Θ(n) **Assuming that the number of digits used is not excessive, the worst-case cost for Radix Sort when sorting n keys with distinct key values is:** Θ(nlogn) **The {best|average|worst} case time complexity of Binsort is:** Θ(n+MaxKeyValue)
A disadvantage of Radix Sort is: It needs auxilliary storage beyond the input array **The order of the input records has what impact on the number of comparisons required by Bin Sort (as presented in this module)?** None **Binsort can be viewed as a special case of Radix Sort where the base is so big that all keys are one digit long.** True **An important disadvantage of the first Binsort algorithm shown in this module is:** It works only for a permutation of the numbers from 0 to n−1 **Which statement is true about the second Binsort pseudocode algorithm presented in this module? Recall that a stable sorting algorithm maintains the relative order of records with equal keys.** It is a stable algorithm

Sorting Algorithms Comparison Review//**Which of the following sorts is not stable?** Quicksort **Which sorting algorithm makes use of Insertion Sort's best case behavior?** Shellsort **Which of the following sorting algorithms has a worst case complexity of Θ(nlogn)?** Heap Sort, Merge Sort **Which of the following sorting methods will be best if the number of swaps done is the only measure of efficiency?** Selection Sort **A person sorting a hand of cards might reasonably use which sorting algorithm?** Insertion Sort **When Mergesort merges two sorted lists of sizes m and n into a sorted list of size m+n, it requires how many comparisons?** M+n−1 **How are Selection Sort and Quicksort similar?** They can both swap non-adjacent records Consider what happens if someone accidentally calls sort on a file that is already sorted. **Which of the following sorting methods will be the most efficient if the input is already in sorted order.** Insertion Sort **Selection Sort is generally faster than the Bubble Sort on the same input** True

Exam 2 Answers

Assume that an array consists of at least 10 elements. The worst case time (in the big-oh sense) occurs in linear search algorithm when
Item is somewhere in the middle of the array
Item is not in the array at all
Item is the last element in the array

The recurrence relation
T(n) = 2 T(n/2) + c, if n >=2
T(n) = c1, otherwise
has the solution
O(n)

Given an array-based list implementation, deleting the current element takes how long in the average case?
Θ(n) time

There are some algorithms that do not terminate for certain inputs.
False

Suppose that a particular algorithm has time complexity T(n)=n^2 and that executing an implementation of it on a particular machine takes t seconds forn inputs. Now suppose that we are presented with a machine that is 64 times as fast. How many inputs could we process on the new machine in tseconds?
8n

Hardware vendor XYZ Corp. claims that their latest computer will run 100 times faster than that of their competitor, Prunes, Inc. If the Prunes, Inc. computer can execute a program on input of size n in one hour, what size input can XYZ's computer execute in one hour for an algorithm whose growth rate is n^2?
10n

Which of these is the best upper bound for a growth rate of 5n+3?
O(nlogn)

The Sequential Search algorithm is in O(n^2).
True

For all the algorithms for which we properly understand the running time analysis, the upper bound and lower bound will always be the same.
True

Consider the following recurrence relation:
T(n) = 3 if n =1
T(n) = T(n-1) + 3 if n>=2
Using the expansion (*aka* iteration) method as discussed in class to solve and find close bound for the recurrence relations, w*hat is the solution or closed-form for T(n)?*
3n

What is the smallest integer k such that nlogn is in O(n^k)?
2

The lower bound for the cost of sequential search is Ω(1) since this is the running time of the algorithm in the best case.
False

The worst case for the sequencial search algorithm occurs when the array size tends to infinity.
False

Which of these is the best definition for a stable sorting algorithm?
An algorithm that does not change the relative ordering of records with identical keys

Which of these will NOT affect the RELATIVE running times for two sorting algorithms?
The CPU speed

The upper bound and lower bounds of the sequential search algorithm is in O(n) and Ω(n)respectively.
False

In which case might the number of comparisons NOT be a good representation of the cost for a sorting algorithm?
When we are comparing strings of widely varying length

Determine Θ for the following code fragment in the average case. Assume that all variables are of type "int".

```
a = b + c;
d = a + e;
```

Θ(1)

Insertion Sort (as discussed in class) is a stable sorting algorithm. Recall that a stable sorting algorithm maintains the relative order of records with equal keys.
True

What is the average-case time for Insertion Sort to sort an array of n records?
Θ(n^2)

When implementing Insertion Sort, a binary search could be used to locate the position within the first i−1 records of the array into which record i should be inserted. Using binary search will:
Not speed up the asymptotic running time because shifting the records to make room for the insert will require Θ(i) time

In which cases are the growth rates the same for Insertion Sort?
Worst and Average only

The order of the input records has what impact on the number of comparisons required by Insertion Sort (as presented in this module)?
There is a big difference, the asymptotic running time can change

When is Insertion Sort a good choice for sorting an array?

The array has only a few records out of sorted order

If I is the number of inversions in an input array of n records, then Insertion Sort will run in what time?
Θ(n+I)

The order of the input records has what impact on the number of comparisons required by Bubble Sort (as presented in class)?
None

What is the running time of Bubble Sort (as the algorithm is presented in this module) when the input is an array that has already been sorted?
Θ(n^2)

In which cases are the time complexities the same for Selection Sort?
Worst, Average and Best

The average number of inversions in an array of n records is n(n−1)/4. This is:
Θ(n^2)

An inversion is:
When a record with key value greater than the current record's key appears before it in the array

If I is the number of inversions in an input array of n records, then {Insertion|Bubble} Sort will require how many swaps?
I

Consider an array A of n records each with a unique key value, and AR the same array in reverse order. There are a certain number of pairs, where an arbitrary position i and position jis a pair. Between these two arrays, exactly half of these pairs must be inverted.
False

How many ways can n distinct values be arranged?
n!

Which is the divide-by-twos increment series for an array of 23 elements?
16, 8, 4, 2, 1

You must merge 2 sorted lists of size m and n, respectively. The number of comparisons needed in the worst case by the merge algorithm will be:
m+n-1

In the worst case, the total number of comparisons for Mergesort is closest to:
n log n

When is Mergesort a good choice for sorting an array?
We need a reasonably fast algorithm with a good worst case cost

What is the average-case time for Mergesort to sort an array of n records?

Θ(nlogn)

When is Quicksort a good choice for sorting an array?
In most standard situations where you want to sort many records

Quicksort (as discussed in class along with pseudo-code) is a stable sorting algorithm. Recall that a stable sorting algorithm maintains the relative order of records with equal keys.
False

In which cases are the time complexities the same for Quicksort?
Best and Average only

What is the worst-case cost for Quicksort to sort an array of n elements?
Θ(n^2)

The lower bound for a problem is defined to be:
The best possible cost for any algorithm that solves the problem

What is the worst-case cost for Quicksort's partition step?
Θ(n)

Exam 3 Modules

A range query is: **Where a record is returned if its relevant key value falls between a pair of values**
An exact-match query is: **Where a record is returned if its unique identifier matches the search value**
A tool for measuring the efficiency of an algorithm or problem is: **Algorithm analysis**
Which is NOT a topic that OpenDSA focuses on? **How to design and maintain large programs**
An *ADT* is: **the specification of a data type within some language, independent of an implementat**
A *type* is: **A collection of values**
A *data item* is: **A member of a type**
A *composite type* is: **A type comprised of other types**
n ADT is a form of: **Abstraction**
A *data structure* is: **The implementation for an ADT**
A *simple type* is: **A type that cannot be broken down into smaller types**
An ADT is most like which of the following? **INTERFACE**
In computer science a *metaphor* is: **A label applied to a group of concepts**
For the string hash functions, the size of the hash table limits the length of the string that can be hashed. **False**
For the simple string hash function that sums the ASCII values for the letters, does the order of the letters in the string affect the result? **NO**
The mid-squares method hash function makes use of: **All of the digits or bits of the key**
A hash function must: **Return a value between 0 and M−1M-1M−1**
For the 'improved' string hash function that sums up 32-bit values each generated by 4 characters at a time, does the output change when the order of the letters changes? **Usually**
If you double the size of a hash table, can you keep using the same hash function? **Yes, but you will not hash to the new slots**
The number of possible total orderings that can be defined on a set of nnn elements is: **n!**
Disk drive access time is normally measured in: **Milliseconds (thousandths of a second)**
Ram access time is in **NanoSeconds**
A program is an instance of an algorithm implemented in a specific programming language. **True**
An algorithm maps inputs to outputs. **False**
A problem instance is a series of steps that act as a recipe to solve a particular problem. **False**
There are some algorithms that do not terminate for certain inputs. **False**
A problem instance is a specific selection of values for the problem input. **True**
Hardware vendor XYZ Corp. claims that their latest computer will run 100 times faster than that of their competitor, Prunes, Inc. If the Prunes, Inc. computer can execute a program on input of size nnn in one hour, what size input can XYZ's computer execute in one hour for an algorithm whose growth rate is n2n^2n2? **10n**
Suppose that a particular algorithm has time complexity T(n)=3×2n T(n) = 3 \times 2^n T(n)=3×2n and that executing an implementation of it on a particular machine takes tt t seconds for nnn inputs. Now suppose that we are presented with a machine that is 64 times as fast. How many inputs could we process on the new machine in ttt seconds? **N + 6**

Suppose that a particular algorithm has time complexity T(n)=n^2 T(n) = n^2T(n)=n2 and that executing an implementation of it on a particular machine takes tt t seconds for nnn inputs. Now suppose that we are presented with a machine that is 64 times as fast. How many inputs could we process on the new machine in ttt seconds? **8n**
Suppose that a particular algorithm has time complexity T(n)=8nT(n) = 8nT(n)=8n and that executing an implementation of it on a particular machine takes tt t seconds for nnn inputs. Now suppose that we are presented with a machine that is 64 times as fast. How many inputs could we process on the new machine in ttt seconds? **64n**

The Sequential Search algorithm is Θ(n2. **False**

Big-Theta notation (Θ\ThetaΘ) defines an equivalence relation on the set of functions. **True**

F(n)/g(n), O = 0, horseshoe = infinity, Theta = constant

If we know that algorithm X is Θ(n)\Theta(n)Θ(n) in the average case, then what can we say about its TIGHTEST upper bound? **It is O(n) in the average case.**

Which of these is the best lower bound for a growth rate of 5n+3? **O(n)**

f(n)=2n

g(n)=nng(n) = n^n

O

f(n)=logn2

g(n)=logn+5g(n) = \log n + 5 g(n)=logn+5

Theta

```
for (i = 0; i < n; i++) {
for (j = 0; j < n; j++)
A[j] = random(n);
sort(A);
}
```

n^2 log n

```
for (i = 0; i < n; i++) {
for (j = 0; A[j] != i; j++)
sum++;
}
```

n^2

```
for (i = 1; i ≤ n; i++)
for (j = 1; j ≤ n; j *= 2)
sum++;
```

n log n

```
for (i = 0; i < n - 1; i++)
for (j = i + 1; j < n; j++) {
tmp = AA[i][j];
AA[i][j] = AA[j][i];
AA[j][i] = tmp;
}
```

n^2

```
  sum = 0;
for (i = 0; i < 3; i++)
for (j = 0; j < n; j++)
sum++;
```

n

```
for (i = 0; i < n * n; i++)
sum++;
```

n^2

No algorithm for searching in an unsorted array can be worse than O(n)O(n)O(n) since any algorithm must look at every value in the array in the worst case.

False

The lower bound for a problem is defined as the cost of the best algorithm that we know. **False**

The worst case lower bound for sorting an array is O(nlogn)O(n \log n)O(nlogn) since this is the cost of the best algorithm (in the worst case) that we know about. **False**
The lower bound for a problem is defined as the cost of the best algorithm that we know **False**
The upper bound for a problem is defined as the upper bound cost for the worst algorithm that we know. **False**
The upper bound for a problem is defined as the upper bound cost for the best algorithm that we know. **True**
The worst case upper bound for sorting an array is O(nlogn) since this is the cost of the best algorithm (in the worst case) that we know about. **True**
The lower bound in the worst case for the problem of searching an unsorted array is Ω(n) because this is the worst case cost of the sequential search algorithm. **False**
The best case for the sequential search algorithm occurs when the array has only a single element. **False**
The lower bound for the cost of sequential search is Ω(1)\Omega(1)Ω(1) since this is the running time of the algorithm in the best case. **False**
The worst case for the sequencial search algorithm occurs when the array size tends to infinity.
False
For all algorithms that we have properly understand the running time analysis, the upper bound and lower bound will be always the same. **True**
The worst case for sequential search occurs when the last element of the array is the value being searched for. **True**
The lower bound for the cost of sequential search is Ω(1)\Omega(1)Ω(1) since this is the running time of the algorithm in the best case. **False**
The upper bound and lower bounds of the sequential search algorithm is in O(n)O(n)O(n) and Ω(n)\Omega(n)Ω(n) respectively. **False**
Which of these is the best upper bound for a growth rate of 5n+3? **Nlogn**
Given an array-based list implementation, inserting a new element to arbitrary position iii takes how long in the average case? **Θ(n) time**
In an array-based list, the successive elements in the list: **Must occupy contiguous space in memory**
Given an array-based list implementation, inserting a new element to the current position takes how long in the average case? **Θ(n) time**
Given an array-based list implementation, deleting the element at arbitrary position iii takes how long in the average case? **Θ(n) time**
In an array-based list implementation, to access the predecessor of the current node we must start at the first node in the list. **False**
An advantage of linked lists over the array-based list implementation is: **Expandability**
In a singly linked list implementation, to access the predecessor of the current node we must start at the first node in the list. **True**
In our linked list implmentation, an empty list contains how many nodes? **Two**
Given a linked list implementation, inserting a new element to arbitrary position iii takes how long in the average case? **Θ(i) time**
Suppose cursor points to a node in a linked list. Which statement changes cursor so that it points to the next node? **cursor = cursor.next()**
Which boolean expression indicates whether the values for the nodes pointed to by p and q are the same? Assume that neither p nor q is null. **p.element() == q.element()**
The physical order in memory for the nodes of a linked list is the same as the order in which the nodes appear in the list. **False**
Given a linked list implementation, deleting the current element takes how long in the average case? **Θ(1) time**
Given a linked list implementation, deleting the element at arbitrary position iii takes how long in the average case? **Θ(i) time**
To access the node at position iii in a singly-linked list **requires that all its predecessors be visited**
In the linked list, where does the insert method place the new entry? **At the current position**

In a linked list, the successive elements in the list: **Need not occupy contiguous space in memory**
To find an element with value X in a linked list with nnn nodes requires how many nodes to be visited in the worst case? **n nodes**
Given a doubly linked list implementation, removing the current element takes how long in the average case? **Θ(1) time**
A doubly linked list is a pair of singly linked lists that each interconnect the same nodes. **False**
Given a doubly linked list implementation, removing the current element takes how long in the average case? **Θ(1) time**
Which list implementation is best to answer the question "What is the item at position iii?"
Array-based list
Which of the following is not a linear data structure? **Tree**
The term "FIFO" is associated with which data structure? **Queue**
Which data structure would be used to implement recursion? **Stack**
The list is an efficient data structure to use to implement a dictionary because we can make both of the key operations (insert and search) efficient. **False**
The problem with using a list to implement a dictionary is that, while search or insert can be implemented efficiently, deletion cannot. **True**
A list can be used to implement a dictionary. **True**
To be efficient, a dictionary implementation requires that the key type define a total order. **True**
Two vertices of a graph are ADJACENT if there is an edge joining them **True**
A free tree is: **a connected graph with no cycles**
The number of edges incident to that vertex called its: **Degree**
Given a subset S of the vertices in a graph, when all vertices in S connect to all other vertices in S, this is called a: **clique**
A complete graph is a clique of size: |**V**|
A graph without cycles is called a/an: **acyclic graph**
When a vertex Q is connected by an edge to a vertex K, what is the term for their relationship?
Q and K are adjacent
A simple path: **must have all vertices be unique**
A graph containing all possible edges is a _____ graph **complete**
A weighted graph must have edge weights and be directed **False**
All graphs must have edges. **False**
Two vertices can be adjacent even if they are not neighbors. **False**
Which statement is false?
Every binary tree has at least one node
Suppose T is a binary tree with 14 nodes. What is the minimum possible height of T? **4**
What is the minimum number of internal nodes in a binary tree with 8 nodes? **4**
If you are given the order of the nodes as visited by a postorder traversal and the order of the nodes as visited by an inorder traversal, do you have enough information to reconstruct the original tree? Assume that the nodes all have unique values. **True**
Consider a node RRR of a complete binary tree whose value is stored in position iii of an array representation for the tree. If RRR has a right child, where will the right child's position be in the array? **2∗i+2**
Which of these is a true statement about the worst-case time for operations on heaps?
Both insertion and removal are better than linear
BST search, insert, and delete operations typically run in time O(d). What is d? **The depth of the relevant node in the tree**
In a max-heap containing nnn elements, what is the position of the element with the greatest value? **0**

Why does function preorder2() presented in the Traversal module make only half as many recursive calls as function preorder()?**Because half of the pointers are null**

Consider a node RRR of a complete binary tree whose value is stored in position iii of an array representation for the tree. If RRR has a left sibling, where will the left sibling's position be in the array? **i−1**

In a max-heap containing n elements, what is the position of the element with the least value?

Possibly in any leaf node

Which feature of heaps allows them to be efficienty implemented using an array? **Heaps are complete binary trees**

If you are given the order of the nodes as visited by a preorder traversal and the order of the nodes as visited by a postorder traversal, do you have enough information to reconstruct the original tree? Assume that the nodes all have unique values. **False**

For the simple string hash function that sums the ASCII values for the letters, does the order of the letters in the string affect the result? **NO**

A hash function must: **M-1**

The simple mod hash function makes use of: **The low order digits or bits in the key**

If you double the size of a hash table, can you keep using the same hash function**? Yes, but you will not hash to the new slots**