EECS 281 F19 Additional Midterm Practice Questions

Big-O Notation

1. Consider the following four statements regarding algorithm complexities:

   i. an algorithm with a $\Theta(n^2)$ time complexity will always run faster than an algorithm with a $\Theta(n \log n)$ time complexity
   ii. an algorithm with a $\Theta(n \log n)$ time complexity will always run faster than an algorithm with a $\Theta(n^2)$ time complexity
   iii. an algorithm with a $\Theta(n^2)$ time complexity will always run faster than an algorithm with a $\Theta(n!)$ time complexity
   iv. an algorithm with a $\Theta(n!)$ time complexity will always run faster than an algorithm with a $\Theta(n^2)$ time complexity

   How many of these statements are true?

   A) 0
   B) 1
   C) 2
   D) 3
   E) 4

5. You are given five different algorithms, each with a different time complexity:

   i. algorithm A has a time complexity of $\Theta(n!)$
   ii. algorithm B has a time complexity of $\Theta(2^n)$
   iii. algorithm C has a time complexity of $\Theta(n^3)$
   iv. algorithm D has a time complexity of $\Theta(n^3 \log n)$
   v. algorithm E has a time complexity of $\Theta(n^n)$

   Which of the following correctly ranks these algorithms in order of increasing time complexity?

   A) D, C, B, A, E
   B) B, C, D, A, E
   C) B, D, C, A, E
   D) C, D, B, A, E
   E) C, D, B, E, A

Complexity Analysis

7. What is the time complexity of the following function?

```
1   int question_7(int n) {
2       int count = 0;
3       for (int i = 0; i < n; i++) {
4           for (int j = i; j > 0; j--) {
5               count += 1;
6           }
7       }
8       return count;
9   }
```

   A) $\Theta(\log n)$
   B) $\Theta(n)$
   C) $\Theta(n \log n)$
   D) $\Theta(n^2)$
   E) none of the above

8. What is the time complexity of the following function?

```
1   int question_8(int n) {
2      int r = 0;
3      while (n > 1) {
4          n /= 2;
5          ++r;
6      }
7      return r;
8   }
```

A) $\Theta(\log n)$
B) $\Theta(n)$
C) $\Theta(n \log n)$
D) $\Theta(n^2)$
E) none of the above

9. What is the time complexity of the following function?

```
1   int question_9(int n) {
2      int count = 0;
3      int m = static_cast<int>(floor(sqrt(n)));
4      for (int i = n/2; i < n; i++) {
5          for (int j = 1; j < n; j = 2 * j) {
6              for (int k = 0; k < n; k += m) {
7                  ++count;
8                  std::cout << "hello world" << std::endl;
9              }
10         }
11     }
12     return count;
13  }
```

A) $\Theta(n^{1/2} \log n)$
B) $\Theta(n \log n)$
C) $\Theta(n^{3/2} \log n)$
D) $\Theta(n^2 \log n)$
E) $\Theta(n^{5/2} \log n)$

Measuring Runtime and Optimization

12. You just completed a project after a long day of work, and good news: the code appears to run correctly with the provided test files! However, when you plot out runtimes with your own test files, you notice that there exists a $\Theta(n)$ relationship even though you had implemented a $\Theta(\log n)$ algorithm. Which of the following could be a reason for this discrepancy?
    A) there were too many programs running in parallel with your program
    B) $\Theta(n)$ algorithms can also have a complexity of $\Theta(\log n)$
    C) the tests you used were too large for the actual relationship to be revealed
    D) the tests you used were too small for the actual relationship to be revealed
    E) none of the above

13. Your friend is also working on a project, which they implemented using a $\Theta(n^2)$ algorithm. However, when they ran a test file on their program, they noticed that the program instead ran in $\Theta(n)$ time. Which of the following is not a possible reason for this discrepancy?
    A) your friend accidentally ran the test case with a different algorithm
    B) your friend incorrectly analyzed the time complexity of their algorithm
    C) your friend's test inputs exposed worst-case behavior
    D) your friend's test case input size was too small
    E) none of the above

Algorithm Design

16. Suppose you are given an unsorted array of n integers, and you are told that all but one of the values inside this array have a negative counterpart. What is the worst-case time complexity of finding the value without a negative counterpart if you use the most efficient algorithm? E.g. given the array

```
{4, 8, -3, -2, -8, 5, -5, -4, 3}
```

you would return the value -2.

A) $\Theta(1)$
B) $\Theta(\log n)$
C) $\Theta(n)$
D) $\Theta(n \log n)$
E) $\Theta(n^2)$

17. Consider an unsorted array of size n containing unique numbers in the range 0 to n. One number from 0 to n is missing from the array. What is the worst-case time and memory complexity of finding the missing number if you use the most efficient algorithm? E.g. given the array

```
{5, 4, 1, 8, 0, 9, 2, 3, 6}
```

you would return the value 7.

A) $\Theta(\log n)$ time, $\Theta(1)$ additional memory
B) $\Theta(n)$ time, $\Theta(1)$ additional memory
C) $\Theta(n)$ time, $\Theta(n)$ additional memory
D) $\Theta(n \log n)$ time, $\Theta(1)$ additional memory
E) $\Theta(n^2)$ time, $\Theta(n)$ additional memory

Vectors

20. Suppose you are given a program that stores its primary data in an instance of `std::vector`. The program executes correctly for small test datasets, but it runs out of memory while reading large datasets. What is a possible solution to overcome this problem?

i. Use `.resize()` with the exact number of elements, then use `.push_back()`.
ii. Use `.resize()` with the exact number of elements, then use `.operator[]()`.
iii. Use `.reserve()` with the exact number of elements, then use `.push_back()`.
iv. Use `.reserve()` with the exact number of elements, then use `.operator[]()`.

A) only i
B) only ii
C) either i or iv
D) either ii or iii
E) i, ii, iii, and iv

Recurrence Relations

24. What is the complexity of the following recurrence relation?

$$T(n) = \begin{cases} c_0, & n = 1 \\ 3T(n-1) + c, & n > 1 \end{cases}$$

A) $\Theta(n)$
B) $\Theta(n^2)$
C) $\Theta(n^3)$
D) $\Theta(2^n)$
E) $\Theta(3^n)$

25. What is the complexity of the following recurrence relation?

$$T(n) = \begin{cases} c_0, & n = 1 \\ 4T\left(\frac{n}{2}\right) + 16n + n^2 + c, & n > 1 \end{cases}$$

A) $\Theta(n)$
B) $\Theta(n \log n)$
C) $\Theta(n^2)$
D) $\Theta(n^2 \log n)$
E) $\Theta(n^4)$

26. What is the complexity of the following recurrence relation?

$$T(n) = \begin{cases} c_0, & n = 1 \\ 5T\left(\frac{n}{25}\right) + \sqrt{n} + c, & n > 1 \end{cases}$$

A) $\Theta(\sqrt{n})$
B) $\Theta(\sqrt{n} \log n)$
C) $\Theta(n)$
D) $\Theta(n^5 \log n)$
E) $\Theta(n^5)$

27. What is the complexity of the following recurrence relation?

$$T(n) = \begin{cases} c_0, & n = 1 \\ 729T\left(\frac{n}{9}\right) + 3n\sqrt[3]{n} + 81n + c, & n > 1 \end{cases}$$

A) $\Theta(\sqrt[3]{n} \log n)$
B) $\Theta(\sqrt[3]{n})$
C) $\Theta(n)$
D) $\Theta(n \sqrt[3]{n})$
E) $\Theta(n^3)$

29. Given the function below, calculate the recurrence relation. Assume that `cake(n)` runs in $\log n$ time.

```
1   void pie(int n) {
2     if (n == 1) {
3       return;
4     }
5
6     pie(n / 7);
7
8     int cookie = n * n;
9
10    for (int i = 0; i < cookie; ++i) {
11      for (int j = 0; j < n; ++j) {
12        cake(n);
13      }
14    }
15
16    for (int k = 0; k < n; ++k) {
17      pie(n / 3);
18    }
19
20    cake(cookie * cookie);
21  } // pie()
```

A) $T(n) = T\left(\frac{n}{7}\right) + n^2 \log n + nT\left(\frac{n}{3}\right) + \log n$

B) $T(n) = T\left(\frac{n}{7}\right) + n^2 \log n + nT\left(\frac{n}{3}\right) + 2\log n$

C) $T(n) = T\left(\frac{n}{7}\right) + n^3 \log n + nT\left(\frac{n}{3}\right) + \log n$

D) $T(n) = T\left(\frac{n}{7}\right) + n^3 \log n + nT\left(\frac{n}{3}\right) + 2\log n$

E) $T(n) = T\left(\frac{n}{7}\right) + n^3 \log n + nT\left(\frac{n}{3}\right) + 4\log n$

30. Suppose you are given an incomplete recurrence relation for a recursive function:

$$T(n) = T(?) + 14\sqrt{n} + 26n + 51$$

You are told that this recurrence relation is directly solvable by the Master Theorem. Knowing this information, which of the following must be true?

A) the complexity of $T(n)$ is $\Theta(1)$

B) the complexity of $T(n)$ is $\Theta(\sqrt{n})$

C) the complexity of $T(n)$ is $\Theta(n)$

D) the complexity of $T(n)$ is $\Theta(n^2)$

E) any of the above may be true

Stacks and Queues and Priority Queues

60. Which of the following is not a disadvantage of using a linked list over an array to implement a stack?

A) the time complexity of pushing an element into an array is $\Theta(1)$, while the time complexity of pushing an element into a linked list is $\Theta(n)$

B) if you do not track the size of a linked list, the time complexity of finding the size of your stack becomes $\Theta(n)$

C) using a linked list is less efficient than using an array, as a linked list must allocate memory for each node individually

D) using a linked list results in high memory overhead

E) none of the above

Consider the following snippet of code. The variable `eecs` is instantiated to be an instance of an unspecified standard library container type.

```
1    eecs.push(203);
2    eecs.push(370);
3    eecs.push(281);
4    std::cout << eecs.top() << ", ";
5    eecs.pop();
6    eecs.push(280);
7    eecs.push(376);
8    eecs.push(183);
9    std::cout << eecs.top() << ", ";
10   eecs.pop();
11   std::cout << eecs.top();
12   eecs.pop();
```

65. If the output of this code is "`370, 376, 281`", what is a valid type for `eecs`?
    - **A)** `std::deque<int>`
    - **B)** `std::stack<int>`
    - **C)** `std::queue<int>`
    - **D)** `std::priority_queue<int>`
    - **E)** none of the above

66. If the output of this code is "`281, 183, 376`", what is a valid type for `eecs`?
    - **A)** `std::deque<int>`
    - **B)** `std::stack<int>`
    - **C)** `std::queue<int>`
    - **D)** `std::priority_queue<int>`
    - **E)** none of the above

68. Suppose that you are using a circular buffer with an array to implement a queue. You utilize two iterators to keep track of your circular buffer: a front iterator that points to the first element in the queue and a back iterator that points one past the last element in the queue. These iterators wrap around the array as needed. Which of the following would indicate that your circular buffer is full, and that you should reallocate your data to a larger array?

A) incrementing the front iterator causes it to point one past the end of the array
B) incrementing the back iterator causes it to point one past the end of the array
C) incrementing the front iterator causes it to point to the same location as the back iterator
D) incrementing the back iterator causes it to point to the same location as the front iterator
E) more than one of the above

69. Two retail companies, Darget and Paolmart, keep track of their inventory in different ways. Darget tracks its inventory using the FIFO (first in, first out) method, assuming that the goods added to inventory first are also the first removed from inventory for sale. Paolmart tracks its inventory using the LIFO (last in, first out) method, assuming that the goods added to inventory last are the first removed from inventory for sale. The two companies want to use a data structure to keep track of their inventories, and they come to you for advice on which data structure to select. What should you tell these two companies?

A) Darget should use a queue, and Paolmart should use a stack
B) Darget should use a stack, and Paolmart should use a queue
C) both Darget and Paolmart should use queues
D) both Darget and Paolmart should use stacks
E) both Darget and Paolmart should use priority queues

73. Consider the following snippet of code. The same elements are pushed into the priority queue as in the previous question, but a custom comparator is now used to determine an element's priority:

```
1    struct CustomComparator {
2      bool operator() (const int lhs, const int rhs) const {
3        // abs(x) returns the absolute value of x
4        return (abs(lhs - 25) > abs(rhs - 25));
5      }
6    };
7
8    int main() {
9      std::priority_queue<int, std::vector<int>, CustomComparator> myPQ;
10     myPQ.push(22);
11     myPQ.push(34);
12     myPQ.push(15);
13     myPQ.push(41);
14     myPQ.push(26);
15
16     while (!myPQ.empty()) {
17       std::cout << myPQ.top() << ' ';
18       myPQ.pop();
19     }
20   }
```

What does the above code output?

A) `41 34 26 22 15`
B) `41 15 34 22 26`
C) `15 22 26 34 41`
D) `26 22 34 15 41`
E) none of the above

74. Suppose you have an array of n elements. What is the worst-case time complexity of pushing these n elements into a `std::priority_queue`?

A) $\Theta(\log n)$
B) $\Theta(n)$
C) $\Theta(n \log n)$
D) $\Theta(n^2)$
E) $\Theta(n^2 \log n)$

Heaps

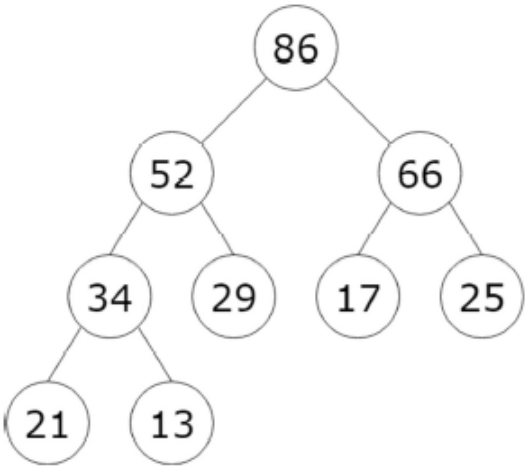
82. Which of the following represents a valid binary min-heap? The "top" of the heap is the element at the first index of the array.

A) `{2, 13, 8, 16, 13, 10, 40, 25, 17}`
B) `{47, 9, 12, 9, 2, 10, 10, 4, 3, 1}`
C) `{3, 5, 6, 7, 12, 15, 14, 9, 10, 11}`
D) `{59, 58, 60, 57, 85, 49, 32, 21, 5}`
E) none of the above

83. Which of the following represents a valid binary max-heap? The "top" of the heap is the element at the first index of the array.

A) `{14, 25, 27, 26, 28, 24, 30, 32, 34}`
B) `{52, 11, 23, 9, 10, 12, 11, 10, 9, 8}`
C) `{33, 24, 24, 7, 9, 24, 18, 7, 5, 9, 8}`
D) `{76, 54, 33, 56, 32, 55, 33, 12, 14}`
E) none of the above

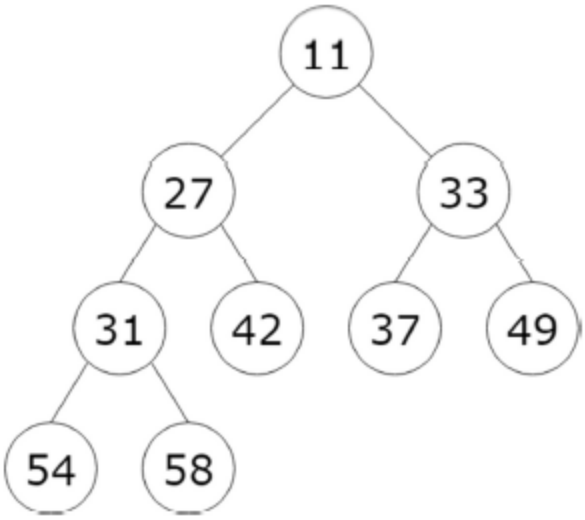
86. Consider the following max-heap:



You insert the value of 93 into this max-heap. After the invariant is fixed, what is the final array representation of this max-heap?

A) `{93, 86, 66, 34, 52, 17, 25, 21, 13, 29}`
B) `{93, 86, 66, 52, 34, 17, 25, 21, 13, 29}`
C) `{93, 52, 86, 34, 29, 66, 25, 21, 13, 17}`
D) `{93, 86, 66, 52, 29, 17, 25, 34, 13, 21}`
E) none of the above

87. Consider the following min-heap:

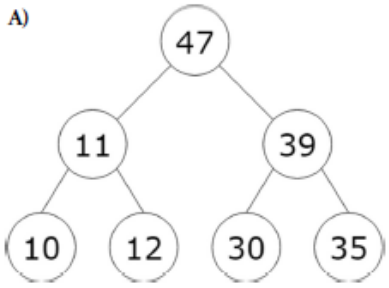


You delete the value of 11 from this min-heap. After the invariant is fixed, what is the final array representation of this min-heap?

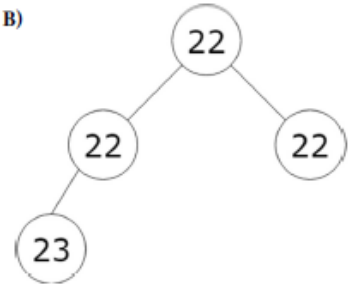
A) `{33, 27, 37, 31, 42, 58, 49, 54}`
B) `{27, 42, 33, 31, 58, 37, 49, 54}`
C) `{58, 27, 33, 31, 42, 37, 49, 54}`
D) `{27, 31, 33, 54, 42, 37, 49, 58}`
E) none of the above

88. Which of the following are valid binary heaps? Select all that apply.

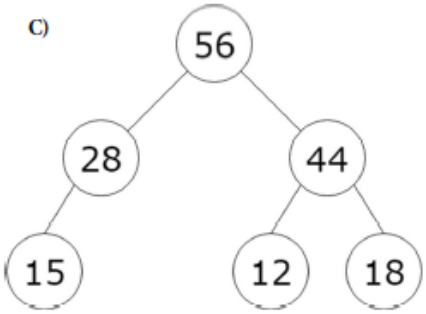
A)



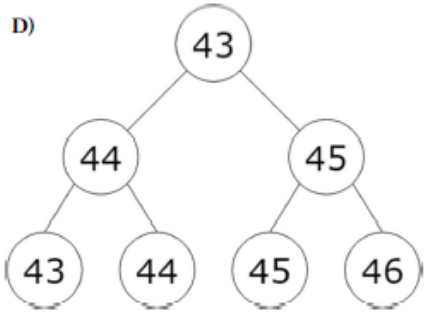
B)



C)



D)



E) none of the above are valid binary heaps

89. What is the worst-case *memory* complexity of running heapify on an array of n integers if you use the most efficient algorithm?

A) $\Theta(1)$
B) $\Theta(\log n)$
C) $\Theta(n)$
D) $\Theta(n \log n)$
E) $\Theta(n^2)$

92. Consider the following unsorted array:

```
{13, 24, 1, 58, 69, 10, 32, 27, 71}
```

Perform a $\Theta(n)$ heapify operation to turn this data into a valid max-heap. What are the contents of the array after the heapify operation?

A) `{71, 13, 32, 24, 69, 10, 1, 27, 58}`
B) `{71, 69, 32, 27, 13, 10, 1, 24, 58}`
C) `{71, 69, 58, 32, 27, 13, 10, 24, 1}`
D) `{71, 69, 32, 58, 13, 10, 1, 27, 24}`
E) none of the above

93. Given an empty min-heap priority queue, you push the following values in this order:

34, 29, 22, 25, 19, 12, 15, 37

What would the underlying array structure look like if one value were popped off the heap?

A) `{12, 22, 15, 34, 25, 29, 19}`
B) `{15, 22, 19, 34, 25, 29, 37}`
C) `{15, 22, 37, 34, 25, 29, 19}`
D) `{15, 19, 22, 34, 25, 29, 37}`
E) none of the above

94. Consider a min-heap represented by the following array:

```
{63, 74, 67, 85, 91, 94, 72, 88}
```

Perform the following operations using the algorithms for binary heaps discussed in lecture. Ensure that the heap property is restored at the end of every individual operation:

i. push the value of 60 into this min-heap
ii. push the value of 79 into this min-heap
iii. update element 85 to have a value of 58
iv. update element 67 to have a value of 96
v. remove the min element from the heap

What does the array representation look like after all five operations are completed?

A) `{60, 72, 63, 94, 96, 74, 79, 88, 91}`
B) `{60, 63, 72, 74, 79, 94, 96, 88, 91}`
C) `{60, 74, 79, 63, 91, 94, 72, 88, 96}`
D) `{60, 63, 72, 74, 79, 88, 91, 94, 96}`
E) none of the above

Union-Find

For questions 105-107, consider the following union-find container:

Item	1	2	3	4	5	6	7	8	9	10
Representative	6	7	8	3	2	4	7	10	9	9

105. Among these 10 elements, how many disjoint sets are there?
   - **A)** 1
   - **B)** 2
   - **C)** 3
   - **D)** 4
   - **E)** none of the above

106. Suppose you used the path compression approach when you implemented union-find. If you were to call the find function on item 1, which of the following statements would be true?
   - **A)** 1's representative would become 4
   - **B)** 6's representative would become 8
   - **C)** 5's representative would become 7
   - **D)** 3's representative would become 9
   - **E)** more than one of the above

107. Suppose you used the path compression approach when you implemented union-find. Which of the following operations would not change any of the representatives given in the above table?
   - **A)** calling find on item 8
   - **B)** calling find on item 5
   - **C)** calling find on item 4
   - **D)** calling find on item 2
   - **E)** none of the above

Sorting Algorithms

109. You decided to run a sorting algorithm on two different arrays:

```
arr1[] = {4, 16, 8, 19, 11, 25, 1, 3};
arr2[] = {5, 7, 9, 12, 16, 14, 15, 18};
```

After running these sorts, you notice that sorting `arr2[]` took less time than sorting `arr1[]`. Which of the following could be the sorting algorithm that you used?

   - **A)** insertion sort
   - **B)** selection sort
   - **C)** heapsort
   - **D)** mergesort
   - **E)** quicksort

111. The university registrar has developed a new method for selecting people off the waitlist. Students with more credits are given priority over students with fewer credits. Ties in credit hours are settled with arrival order — that is, students who signed up for the waitlist earlier are selected over students who signed up later. If you could only look at the number of credits, which of the following sorts would not allow the registrar to accomplish these goals?

A) bubble sort
B) selection sort
C) insertion sort
D) mergesort
E) all of the above sorts would satisfy the registrar's requirements

112. Which of the following statements is true?

A) bubble sort can be implemented to be either adaptive or stable, but not both
B) insertion sort is fast on nearly sorted arrays but requires $\Theta(n)$ auxiliary space
C) using binary search within an insertion sort will increase the speed, but the sort will no longer be stable
D) selection sort performs only $\Theta(n)$ comparisons on an already sorted array
E) the best-case time complexity of selection sort is $\Theta(n^2)$

116. One way to classify sorting algorithms is by their adaptability. What is an advantage of a non-adaptive sorting algorithm?

A) non-adaptive sorting algorithms may perform less work depending on the order of the data
B) non-adaptive sorting algorithms are always faster than adaptive algorithms
C) non-adaptive sorting algorithms may change its sequence of operations based on the input
D) non-adaptive sorting algorithms may be simpler to implement than adaptive algorithms
E) non-adaptive sorting algorithms are always guaranteed to be stable

117. For which of the following tasks would bucket sort not be useful?

A) sorting students at the university by their birthday
B) sorting students at the university by the number of credits they are currently taking
C) sorting students at the university by their city or town of birth
D) sorting students at the university by the number of siblings they have
E) none of the above

119. Consider the following snippet of code, which adds the names of Winter 2018 EECS 281 instructors into a vector and sorts them by first name:

```
1   struct CompareFirstOnly {
2      template<typename T1, typename T2>
3      bool operator()(const pair<T1, T2>& p1, const pair<T1, T2>& p2) {
4         return p1.first < p2.first;
5      }
6   };
7
8   int main() {
9      std::vector<pair<std::string, std::string>> instructors;
10     instructors.push_back(make_pair("nathan", "moos"));
11     instructors.push_back(make_pair("will", "wendorf"));
12     instructors.push_back(make_pair("jake", "hage"));
13     instructors.push_back(make_pair("nathan", "fenner"));
14     instructors.push_back(make_pair("fee", "christoph"));
15     instructors.push_back(make_pair("bing", "schaefer"));
16
17     mysterySort(instructors.begin(), instructors.end(), CompareFirstOnly());
18
19     for (auto i : instructors) {
20        std::cout << i.first << " " << i.second << std::endl;
21     }
22  }
```

After running this code, you notice that the output is:

```
bing schaefer
fee christoph
jake hage
nathan fenner
nathan moos
will wendorf
```

Which of the following could be the sorting algorithm used to implement `mysterySort`?

A) bubble sort
B) insertion sort
C) selection sort
D) mergesort
E) any of the above

120. You are using quicksort to sort the following 9 elements in increasing order:

```
int arr[] = {43, 82, 50, 65, 24, 19, 78, 16, 37}
```

Which of the following elements would be the best choice for the pivot?

A) 43
B) 50
C) 24
D) 37
E) none of the above

123. Suppose you had the following unsorted array:

```
{22, 9, 13, 52, 66, 74, 28, 59, 71, 35, 11, 47}
```

A snapshot is taken during execution of a sorting algorithm. If the snapshot of the array is

```
{9, 13, 22, 52, 66, 74, 28, 59, 71, 11, 35, 47}
```

which of the following sorts is currently being run on this array?

A) bubble sort
B) insertion sort
C) selection sort
D) quicksort
E) mergesort

124. Suppose you had the following unsorted array:

```
{22, 9, 13, 52, 66, 74, 28, 59, 71, 35, 11, 47}
```

A snapshot is taken during execution of a sorting algorithm. If the snapshot of the array is

```
{9, 11, 13, 22, 28, 74, 66, 59, 71, 35, 52, 47}
```

which of the following sorts is currently being run on this array?

A) bubble sort
B) insertion sort
C) selection sort
D) quicksort
E) mergesort

125. Suppose you had the following unsorted array:

```
{22, 9, 13, 52, 66, 74, 28, 59, 71, 35, 11, 47}
```

A snapshot is taken during execution of a sorting algorithm. If the snapshot of the array is

```
{22, 9, 13, 11, 35, 28, 47, 59, 71, 66, 52, 74}
```

which of the following sorts is currently being run on this array?

A) bubble sort
B) insertion sort
C) selection sort
D) quicksort
E) mergesort

126. Suppose you had the following unsorted array:

```
{22, 9, 13, 52, 66, 74, 28, 59, 71, 35, 11, 47}
```

A snapshot is taken during execution of a sorting algorithm. If the snapshot of the array is

```
{9, 13, 22, 28, 52, 59, 35, 66, 11, 47, 71, 74}
```

which of the following sorts is currently being run on this array?

A) bubble sort
B) insertion sort
C) selection sort
D) quicksort
E) mergesort

127. Suppose you had the following unsorted array:

```
{22, 9, 13, 52, 66, 74, 28, 59, 71, 35, 11, 47}
```

A snapshot is taken during execution of a sorting algorithm. If the snapshot of the array is

```
{9, 13, 22, 28, 52, 59, 66, 74, 71, 35, 11, 47}
```

which of the following sorts is currently being run on this array?

A) bubble sort
B) insertion sort
C) selection sort
D) quicksort
E) mergesort

Strings

129. What is the worst-case time complexity of sorting a vector of n strings, each of length m characters, if you use the most efficient algorithm?

A) $\Theta(m \log n)$
B) $\Theta(n \log n)$
C) $\Theta(mn \log n)$
D) $\Theta(m^2 n \log n)$
E) $\Theta(mn^2)$

131. If you are searching for a string of length n within a string of length h using brute-force string searching, the average-case time complexity would be _____ and the worst-case time complexity would be _____.

A) $\Theta(n), \Theta(h)$
B) $\Theta(n), \Theta(n + h)$
C) $\Theta(n), \Theta(n^2)$
D) $\Theta(h), \Theta(h^2)$
E) $\Theta(h), \Theta(nh)$