

CS 14: Data Structures and Algorithms

Friday, October 31, 2003

Midterm Exam, Form: A



This exam is worth **60** points and lasts **40 minutes**. There are **30** multiple choice questions, which means you should not spend more than **1 minute and 20 seconds** per question, on average. Note: use of witchcraft practices during this test will be considered cheating. Good luck!

1. [2 points] Finding the **minimum** value in a linked list of integers sorted in **ascending** order is an operation with a runtime complexity
  - (a)  equal to  $\mathcal{O}[1]$ .
  - (b)  that depends on whether the list is singly linked or doubly linked.
  - (c)  that depends on whether or not the list has a tail pointer.
  - (d)  that depends on whether or not the list has a head pointer.
  - (e)  None of the above.
  
2. [2 points] Finding the **minimum** value in a SquareList of integers is an operation with a runtime complexity of
  - (a)   $\mathcal{O}[1]$
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$

3. [2 points] A particular function takes a positive integer parameter  $n$  and performs exactly  $3n + 2n \log(n) + 50 \log(n) + 1000$  steps before returning. What is its runtime complexity?
- (a)   $\mathcal{O}[1]$
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$
4. [2 points] Which of the sorting algorithms mentioned below is a **recursive** algorithm?
- (a)  naïve BubbleSort
  - (b)  smart BubbleSort
  - (c)  SelectionSort
  - (d)  InsertionSort
  - (e)  MergeSort
5. [2 points] What is the **best**-case runtime complexity of searching for an item in a **sorted** array using **linear** search?
- (a)   $\mathcal{O}[1]$
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$
6. [2 points] Which of the following is a **dis**advantage of using an array versus using a linked list?
- (a)  inserting into an array may require shifting many other elements.
  - (b)  arrays cannot be sorted.
  - (c)  not all elements in an array can be accessed.
  - (d)  declaring an array is a cumbersome programming task.
  - (e)  arrays can only be used with built-in data types.

7. [2 points] If you have a **sorted** array of 1000 integers, what is the **maximum** number of comparisons against the target item that might be performed by **binary** search?
- (a)  1
  - (b)  2
  - (c)  10
  - (d)  500
  - (e)  1000
8. [2 points] Removing the **last** element of a **doubly**-linked list **without** a tail pointer is an operation with a runtime complexity of
- (a)   $\mathcal{O}[1]$
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$
9. [2 points] Which of the sorting algorithms mentioned below is the **fastest**, in the **worst**-case?
- (a)  naïve BubbleSort
  - (b)  smart BubbleSort
  - (c)  SelectionSort
  - (d)  InsertionSort
  - (e)  MergeSort
10. [2 points] In order to **safely delete** an element in a list, how many iterators should be positioned at the element to be deleted?
- (a)  0.
  - (b)  1.
  - (c)  2.
  - (d)  The question doesn't make sense because lists do not support iterators; only arrays do.
  - (e)  None of the above.

11. [2 points] Removing the **last** element of a **doubly**-linked list **with** a tail pointer is an operation with a runtime complexity of
- (a)   $\mathcal{O}[1]$
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$
12. [2 points] Finding the **maximum** value in a linked list of integers sorted in **ascending** order is an operation with a runtime complexity
- (a)  equal to  $\mathcal{O}[1]$ .
  - (b)  that depends on whether the list is singly linked or doubly linked.
  - (c)  that depends on whether or not the list has a tail pointer.
  - (d)  that depends on whether or not the list has a head pointer.
  - (e)  None of the above.
13. [2 points] What is the **best**-case runtime complexity of searching for an item in an **unsorted** array using **linear** search?
- (a)   $\mathcal{O}[1]$
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$
14. [2 points] If you have a **sorted** array of 1000 integers, what is the **minimum** number of comparisons against the target item that might be performed by **binary** search?
- (a)  1
  - (b)  2
  - (c)  10
  - (d)  500
  - (e)  1000

15. [2 points] What does it mean to say that an operation takes constant time?
- (a)  the operation takes the same time in every computer.
  - (b)  the operation takes an amount of time that is **in**dependent of the size of its input.
  - (c)  the operation takes an amount of time equal to the **size** of its input **plus** some constant.
  - (d)  the operation takes an amount of time equal to the **size** of its input **times** some constant.
  - (e)  the operation takes an amount of time that is **in**dependent of how long your algorithm has been running.
16. [2 points] Which of the following is an advantage of using an array versus using a linked list?
- (a)  arrays are always sorted.
  - (b)  arrays support more data types.
  - (c)  array indices start at zero.
  - (d)  arrays don't require any memory.
  - (e)  accessing an arbitrary element in an array is faster.
17. [2 points] The mummy in the first page of this test
- (a)  helps you to concentrate.
  - (b)  scares you.
  - (c)  looks like you.
  - (d)  looks like Wagner.
  - (e)  is there because today is Halloween.

18. [2 points] What does the function `f()` below do?

```
bool f(const int a[ ], const unsigned int SIZE)
{
    for (unsigned int i = 0; i < SIZE-1; ++i)
    {
        if (a[i+1] < a[i])
            return false;
    }

    return true;
}
```

- (a)  it returns whether the last element of the array is the **max** of all the elements.
  - (b)  it returns whether the last element of the array is the **min** of all the elements.
  - (c)  it returns whether the array is sorted in **ascending** order.
  - (d)  it returns whether the array is sorted in **descending** order.
  - (e)  None of the above.
19. [2 points] MergeSort is a sorting algorithm with a runtime complexity, in the **worst**-case, of
- (a)   $\mathcal{O}[1]$
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$
20. [2 points] Which of the sorting algorithms mentioned below is the **fastest**, in the **best**-case?
- (a)  naïve BubbleSort
  - (b)  smart BubbleSort
  - (c)  SelectionSort
  - (d)  InsertionSort
  - (e)  MergeSort

21. [2 points] SelectionSort is a sorting algorithm with a runtime complexity, in the **worst**-case, of
- (a)   $\mathcal{O}[1]$
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$
22. [2 points] A queue is a data structure where elements are
- (a)  inserted at the front and removed from the back.
  - (b)  inserted **and** removed from the top.
  - (c)  inserted at the back and removed from the front.
  - (d)  inserted **and** removed from both ends.
  - (e)  None of the above.
23. [2 points] push and pop are operations affecting which end of a stack?
- (a)  The bottom.
  - (b)  The front.
  - (c)  The top.
  - (d)  The back.
  - (e)  push and pop are **not** stack operations.
24. [2 points] What is the **worst**-case runtime complexity of searching for an item in an **unsorted** array using **linear** search?
- (a)   $\mathcal{O}[1]$
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$

25. [2 points] **Black-box** testing of the implementation of an ADT is a testing procedure in which you do **not** have access to the ADT's
- (a)  author.
  - (b)  source code.
  - (c)  interface.
  - (d)  axioms.
  - (e)  semantics.
26. [2 points] Removing the **last** element of a **singly**-linked list **with** a tail pointer is an operation with a runtime complexity of
- (a)   $\mathcal{O}[1]$
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$
27. [2 points] Removing the **last** element of a **singly**-linked list **without** a tail pointer is an operation with a runtime complexity of
- (a)   $\mathcal{O}[1]$
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$

28. [2 points] What is the **worst**-case runtime complexity of searching for an item in a **sorted** array using **binary** search?
- (a)   $\mathcal{O}[1]$
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$
29. [2 points] What is the **best**-case runtime complexity of searching for an item in a **sorted** array using **binary** search?
- (a)   $\mathcal{O}[1]$
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$
30. [2 points] What is the **worst**-case runtime complexity of searching for an item in a **sorted** array using **linear** search?
- (a)   $\mathcal{O}[1]$
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$

# Answer Key for Exam A

This exam is worth **60** points and lasts **40 minutes**. There are **30** multiple choice questions, which means you should not spend more than **1 minute and 20 seconds** per question, on average. Note: use of witchcraft practices during this test will be considered cheating. Good luck!

1. [2 points] Finding the **minimum** value in a linked list of integers sorted in **ascending** order is an operation with a runtime complexity
  - (a)  equal to  $\mathcal{O}[1]$ .
  - (b)  that depends on whether the list is singly linked or doubly linked.
  - (c)  that depends on whether or not the list has a tail pointer.
  - (d)  that depends on whether or not the list has a head pointer.
  - (e)  None of the above.
  
2. [2 points] Finding the **minimum** value in a SquareList of integers is an operation with a runtime complexity of
  - (a)   $\mathcal{O}[1]$
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$
  
3. [2 points] A particular function takes a positive integer parameter  $n$  and performs exactly  $3n + 2n \log(n) + 50 \log(n) + 1000$  steps before returning. What is its runtime complexity?
  - (a)   $\mathcal{O}[1]$
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$

4. [2 points] Which of the sorting algorithms mentioned below is a **recursive** algorithm?
- (a)  naïve BubbleSort
  - (b)  smart BubbleSort
  - (c)  SelectionSort
  - (d)  InsertionSort
  - (e)  MergeSort**
5. [2 points] What is the **best**-case runtime complexity of searching for an item in a **sorted** array using **linear** search?
- (a)   $\mathcal{O}[1]$**
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$
6. [2 points] Which of the following is a **dis**advantage of using an array versus using a linked list?
- (a)  inserting into an array may require shifting many other elements.**
  - (b)  arrays cannot be sorted.
  - (c)  not all elements in an array can be accessed.
  - (d)  declaring an array is a cumbersome programming task.
  - (e)  arrays can only be used with built-in data types.
7. [2 points] If you have a **sorted** array of 1000 integers, what is the **maximum** number of comparisons against the target item that might be performed by **binary** search?
- (a)  1
  - (b)  2
  - (c)  10**
  - (d)  500
  - (e)  1000

8. [2 points] Removing the **last** element of a **doubly**-linked list **without** a tail pointer is an operation with a runtime complexity of
- (a)   $\mathcal{O}[1]$
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)**   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$
9. [2 points] Which of the sorting algorithms mentioned below is the **fastest**, in the **worst**-case?
- (a)  naïve BubbleSort
  - (b)  smart BubbleSort
  - (c)  SelectionSort
  - (d)  InsertionSort
  - (e)**  **MergeSort**
10. [2 points] In order to **safely delete** an element in a list, how many iterators should be positioned at the element to be deleted?
- (a)  0.
  - (b)**  **1.**
  - (c)  2.
  - (d)  The question doesn't make sense because lists do not support iterators; only arrays do.
  - (e)  None of the above.
11. [2 points] Removing the **last** element of a **doubly**-linked list **with** a tail pointer is an operation with a runtime complexity of
- (a)**   $\mathcal{O}[1]$
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$

12. [2 points] Finding the **maximum** value in a linked list of integers sorted in **ascending** order is an operation with a runtime complexity
- (a)  equal to  $\mathcal{O}[1]$ .
  - (b)  that depends on whether the list is singly linked or doubly linked.
  - (c)  that depends on whether or not the list has a tail pointer.**
  - (d)  that depends on whether or not the list has a head pointer.
  - (e)  None of the above.
13. [2 points] What is the **best**-case runtime complexity of searching for an item in an **unsorted** array using **linear** search?
- (a)   $\mathcal{O}[1]$**
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$
14. [2 points] If you have a **sorted** array of 1000 integers, what is the **minimum** number of comparisons against the target item that might be performed by **binary** search?
- (a)  1**
  - (b)  2
  - (c)  10
  - (d)  500
  - (e)  1000

15. [2 points] What does it mean to say that an operation takes constant time?
- (a)  the operation takes the same time in every computer.
  - (b)  the operation takes an amount of time that is independent of the size of its input.**
  - (c)  the operation takes an amount of time equal to the **size** of its input **plus** some constant.
  - (d)  the operation takes an amount of time equal to the **size** of its input **times** some constant.
  - (e)  the operation takes an amount of time that is **in**dependent of how long your algorithm has been running.
16. [2 points] Which of the following is an advantage of using an array versus using a linked list?
- (a)  arrays are always sorted.
  - (b)  arrays support more data types.
  - (c)  array indices start at zero.
  - (d)  arrays don't require any memory.
  - (e)  accessing an arbitrary element in an array is faster.**
17. [2 points] The mummy in the first page of this test
- (a)  helps you to concentrate.
  - (b)  scares you.
  - (c)  looks like you.
  - (d)  looks like Wagner.
  - (e)  is there because today is Halloween.**

18. [2 points] What does the function `f()` below do?

```
bool f(const int a[ ], const unsigned int SIZE)
{
    for (unsigned int i = 0; i < SIZE-1; ++i)
    {
        if (a[i+1] < a[i])
            return false;
    }

    return true;
}
```

- (a)  it returns whether the last element of the array is the **max** of all the elements.
  - (b)  it returns whether the last element of the array is the **min** of all the elements.
  - (c)  it returns whether the array is sorted in ascending order.
  - (d)  it returns whether the array is sorted in **descending** order.
  - (e)  None of the above.
19. [2 points] MergeSort is a sorting algorithm with a runtime complexity, in the **worst**-case, of
- (a)   $\mathcal{O}[1]$
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$
20. [2 points] Which of the sorting algorithms mentioned below is the **fastest**, in the **best**-case?
- (a)  naïve BubbleSort
  - (b)  smart BubbleSort
  - (c)  SelectionSort
  - (d)  InsertionSort
  - (e)  MergeSort

21. [2 points] SelectionSort is a sorting algorithm with a runtime complexity, in the **worst**-case, of
- (a)   $\mathcal{O}[1]$
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$**
22. [2 points] A queue is a data structure where elements are
- (a)  inserted at the front and removed from the back.
  - (b)  inserted **and** removed from the top.
  - (c)  inserted at the back and removed from the front.**
  - (d)  inserted **and** removed from both ends.
  - (e)  None of the above.
23. [2 points] push and pop are operations affecting which end of a stack?
- (a)  The bottom.
  - (b)  The front.
  - (c)  The top.**
  - (d)  The back.
  - (e)  push and pop are **not** stack operations.
24. [2 points] What is the **worst**-case runtime complexity of searching for an item in an **unsorted** array using **linear** search?
- (a)   $\mathcal{O}[1]$
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)   $\mathcal{O}[n]$**
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$

25. [2 points] **Black-box** testing of the implementation of an ADT is a testing procedure in which you do **not** have access to the ADT's
- (a)  author.
  - (b)**  **source code.**
  - (c)  interface.
  - (d)  axioms.
  - (e)  semantics.
26. [2 points] Removing the **last** element of a **singly**-linked list **with** a tail pointer is an operation with a runtime complexity of
- (a)   $\mathcal{O}[1]$
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)**   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$
27. [2 points] Removing the **last** element of a **singly**-linked list **without** a tail pointer is an operation with a runtime complexity of
- (a)   $\mathcal{O}[1]$
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)**   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$

28. [2 points] What is the **worst**-case runtime complexity of searching for an item in a **sorted** array using **binary** search?
- (a)   $\mathcal{O}[1]$
  - (b)**   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$
29. [2 points] What is the **best**-case runtime complexity of searching for an item in a **sorted** array using **binary** search?
- (a)**   $\mathcal{O}[1]$
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$
30. [2 points] What is the **worst**-case runtime complexity of searching for an item in a **sorted** array using **linear** search?
- (a)   $\mathcal{O}[1]$
  - (b)   $\mathcal{O}[\log(n)]$
  - (c)   $\mathcal{O}[n \log(n)]$
  - (d)**   $\mathcal{O}[n]$
  - (e)   $\mathcal{O}[\sqrt{n}]$
  - (f)   $\mathcal{O}[n^2]$