# EECS 281, Week 4: Wednesday

## Asymptotics

1. Fill in the following table by selecting all relations $\rho$ such that $f(x)$ is $\rho(g(x))$.

| Question | $f(x)$ | $g(x)$ | $f(x)$ is … of $g(x)$ | | |
|---|---|---|---|---|---|
| | | | O | Ω | Θ |
| Example | $203x^2$ | $203x$ | | ✔ | |
| 1.1 | $x$ | $x + 183$ | | | |
| 1.2 | $280^x$ | $281^x$ | | | |
| 1.3 | $\log x$ | $\sqrt{x}$ | | | |

2. Order from most to least efficient:

   $O(n)$ $O(n!)$ $O(n^n)$ $O(n^2)$ $O(2^n)$ $O(\log n)$ $O(1)$
   $O\left(\sqrt{n}\right)$ $O(n^3)$ $O(n^2 \log n)$ $O(3^n)$ $O(n \log n)$

   _____ ⊂ _____ ⊂ _____ ⊂ _____ ⊂ _____ ⊂ _____ ⊂ _____ ⊂

   _____ ⊂ _____ ⊂ _____ ⊂ _____ ⊂ _____.

3. (T/F) If $f_1(n) = O(g_1(n))$ and $f_2(m) = O(g_2(m))$, then $f_1(n) \cdot f_2(m) = O(g_1(n) \cdot g_2(m))$.

4. (T/F) It is possible for an $\Omega(n^3)$ algorithm to grow at a rate slower than an $\Omega(n)$ algorithm.

5. If the running time of an (improved search) algorithm is $f(n) = \log n$, so that the time it takes for an input of size $S$ is $T = \log S$, then in time $T + 1$ you can solve a problem of size _____. *(Express your answer in terms of $S$.)*

6. If the running time of an algorithm is $f(n) = n^2$, so that the time it takes for an input of size $S$ is $T = S^2$, then in time $2T$ you can solve a problem of size _____. (Express your answer in terms of $S$.)

7. Several sorting algorithms have a running time of $f(n) = n \log n$, so that the time it takes for an input of size $S$ is $T = S \log S$. If you instead need to sort <u>two</u> lists, each of size $S$, you will need time _____ to solve your new problem. (Express your answer in terms of T.)

8. Suppose we find that an algorithm has a runtime $R(N)$ that is quadratic in the worst case, and linear in the best case. Which of the following statements are true?
   A. $R(N) \in O(N)$
   B. $R(N) \in O(N^2)$
   C. $R(N) \in O(N^3)$
   D. $R(N) \in \Theta(N)$
   E. $R(N) \in \Theta(N^2)$
   F. $R(N) \in \Theta(N^3)$
   G. In the worst case, $R(N) \in O(N)$
   H. In the worst case, $R(N) \in O(N^2)$
   I. In the worst case, $R(N) \in O(N^3)$
   J. In the worst case, $R(N) \in \Theta(N)$
   K. In the worst case, $R(N) \in \Theta(N^2)$
   L. In the worst case, $R(N) \in \Theta(N^3)$
   M. In the best case, $R(N) \in O(N)$
   N. In the best case, $R(N) \in O(N^2)$
   O. In the best case, $R(N) \in O(N^3)$
   P. In the best case, $R(N) \in \Theta(N)$
   Q. In the best case, $R(N) \in \Theta(N^2)$
   R. In the best case, $R(N) \in \Theta(N^3)$
9. $1 + 2 + 3 + 4 + \ldots + N =$

10. $1 + 2 + 4 + 8 + 16 + \ldots + N =$

## Analysis of Algorithms

11. Find a simple $f(n)$ such that the runtime $R(n) \in \Theta(f(n))$.

```
int fn(int n) {
    if (n <= 1) {
        return 1;
    }
    return fn(n / 2) + fn(n / 2);
}
```

# EECS 281, Week 4: Wednesday

12. Find a simple $f(n)$ such that the runtime $R(n) \in \Theta(f(n))$.

```
int recursive(int n) {
    if (n <= 1) {
        return 1;
    }
    return recursive(n - 1) + recursive(n - 1);
}
```

13. Find a simple $f(n)$ such that the runtime $R(n) \in \Theta(f(n))$.

```
int fn(int n) {
    int x = 0;
    for (int i = 0; i < n; i++) {
        x++;
    }
    return x;
}
```

14. Find a simple $f(n)$ such that the runtime $R(n) \in \Theta(f(n))$.

```
int fn(int n) {
    int x = 0;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < i * i; j++) {
            x++;
        }
    }
    return x;
}
```

15. Find a simple $f(n)$ such that the runtime $R(n) \in \Theta(f(n))$.

```
int fn(int n) {
    if (n <= 1) {
        return 1;
    }
    return fn(n - 1) + fn(n - 1);
}
```

16. Find a simple $f(n)$ such that the runtime $R(n) \in \Theta(f(n))$.

```
int fn(int n) {
    if (n <= 1) {
        return 1;
    }
    return fn(n / 2) + fn(n / 2);
}
```

17. Find a simple $f(n)$ such that the runtime $R(n) \in \Theta(f(n))$.

```
int fn(int n) {
    if (n <= 1) {
        return 1;
    }
    return fn(n) + fn(n / 2) + fn(n / 2);
}
```

18. Describe the difference between best-case, worst-case, average-case and amortized complexity.

### Recurrence Relations

*If possible, use the Master Theorem to find the complexity class for the following recurrence relations. Express your answers using Big Θ notation. If the Master Theorem does not apply, enter "N/A" and explain why not.*

19. $T(n) = 3T\left(\frac{n}{4}\right) + n^2$
20. $T(n) = nT\left(\frac{n}{2}\right)$
21. $T(n) = 8T\left(\frac{n}{2}\right) + n^3$
22. $T(n) = 2T(n-1)$
23. $T(n) = 4T\left(\frac{n}{2}\right)$

# EECS 281, Week 4: Wednesday

## STL

24. If there are $A$ algorithms and $C$ containers, STL must provide only _____ total number of implementation.

25. Why are there several kinds of iterators?

26. What is the difference between iterators and output iterators?

27. (T/F) STL list only provides forward iterators.

28. (T/F) Bidirectional iterators can be compared with operators < and >.

29. (T/F) For a valid STL container `c`, when the expression `c.end() - c.begin()` is well-defined, it returns the same value as `c.size()`.

30. Given an STL container called `aContainer`, what is the difference between `aContainer.being()`, `aContainer.end()`, `aContainer.cbeing()`, `aContainer.cend()`, `aContainer.rbeing()`, `aContainer.rend()`, `aContainer.crbeing()` and `aContainer.crend()`?

31. (T/F) An iterator always holds an address (pointer), and operator++ applied to the iterator always increases that address.

32. Which STL containers among deque, vector and list can invalidate iterators, pointers and references, and when?

33. (T/F) For a valid STL container `aContainer.rend`, the iterator returned by `aContainer.rend()` refers to the first valid element in `c`.

34. What are the drawbacks of using an STL vector?

35. How to minimize reallocations of a vector?

36. How is an STL sequel different from an STL vector?

37. Describe the situations when a deque is preferred over a vector and when a vector is preferred over a deque.

## Containers, Linked Lists, Arrays

38. What is the difference between an ordered container and a sorted container?

39. (T/F) A sorted container must be ordered.

40. (T/F) Assuming the list has no tail pointer, appending to the end of the list is O(n).

41. A linked list is normally specified by giving a pointer pointing to the first node of a list. An empty list has no nodes at all. What is the value of a head pointer for an empty list?
42. (T/F) If the data that a link list contains is small, it generally take up less memory than an STL vector of the same length.

43. (T/F) For both linked lists and arrays, it takes linear time to insert something if you want to maintain it in sorted order.

44. Show that doing a binary search on a sorted linked list takes $\Theta(n)$ time.

45. (T/F) Inserting an element at a random index in a vector has tightest upper bound $O(n)$.

46. (T/F) Inserting an element at the end of a vector has tightest upper bound $O(n)$.

47. In a _____ array with $n$ elements, $k$th element can be removed in worst-case $O(1)$ time, but in a _____ array with $n$ elements, this may take $O(n)$.
48. Describe at least three situations when linked lists are preferred over arrays.

49. Describe at least three situations when arrays are preferred over linked lists.

## Queues and Stacks

50. (T/F) In a queue, items are added and removed according to the LIFO principle.

51. In a stack, items are added and removed according to the FIFO principle.

52. How can we implement a queue that can hold at most $N$ elements using an array of size $N$?

53. Describe how to implement a queue with two stacks.

# EECS 281, Week 4: Wednesday

## Priority Queues and Heaps

54. (T/F) The height of a binary heap is O(n).

55. If we represent a binary heap with a zero-indexed array, what is the index of the parent of the element at index $i$?

56. What is the index of the parent of the left child of the element at index $i$?

57. What is the index of the parent of the right child of the element at index $i$?

58. What is the height of the tree with $n$ elements?

59. (T/F) buidMaxHeap is an algorithm that can turn any array into a heap in $\Theta(n)$ time, where $n$ is the number of elements in the array.

60. Which of the following represents a max-heap?
    A. [47, 8, 12, 9, 2, 10, 10, 4, 3, 1]
    B. [2, 13, 8, 16, 13, 10, 40, 25, 17]
    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.

61. Suppose that seven values are pushed into an empty maximum priority queue binary heap in the following order:

    9, 7, 6, 17, 21, 1, 14

    What would be the array representation of the heap after all seven values are inserted? What would be the tree representation?

62. What would the array representation of the heap from the previous question look like after two deletions? Tree representation?

63. Suppose that seven values are pushed into an empty minimum priority queue binary heap in the following order:

    12, 4, 9, 27, 13, 2, 6

    What would be the array representation of the heap after all seven values are inserted? What would be the tree representation?

64. What would the array representation of the heap from the previous question look like after two deletions? Tree representation?

65. Suppose we wish to create a binary heap containing the keys `H E L L O W O R L D`. (All comparisons use alphabetical order.) Show the resulting min-heap if we build it using successive `insert()` operations (starting from `H`).

66. Suppose we wish to create a binary heap containing the keys `H E L L O W O R L D`. (All comparisons use alphabetical order.) Show the resulting min-heap if we build it using `buildMinHeap()`.

# EECS 281, Week 4: Wednesday

## Disjoint sets

67. (T/F) Representing the items in a disjoint set as a tree and storing the true identity of the set at the root provides a fast find operation and a slower union operation.

68. If we restrict the items in the universe to $N$ integers from $0$ to $1$, how can we represent tree-based disjoint sets as an array?

69. What optimization can we make to the union operation on tree-based disjoint sets?

70. What optimization can we make to the find operation on tree-based disjoint sets?

71. Write the find function for a tree-based disjoint set represented as an array that leverages the optimization from the previous question.

## Sorting

72. What is an inversion?

73. What is a stable sort?

74. What is an in-place sort?

75. What are the properties of sorting algorithms that we care about?

76. (T/F) All in-place sorting algorithms are stable.

77. (T/F) Running time of insertion sort is proportional to the number of inversions.

78. (T/F) Selection sort takes $\Theta(n)$ time in the best case and $\Theta(n^2)$ time in the worst case.

79. (T/F) Merge sort is a divide-and-conquer algorithm and most of the work is done in the "dividing" part.

80. Write a recurrence relation for merge sort.

81. (T/F) Merge sort is suited well for linked lists.

82. (T/F) In-place versions of merge sort exist and outperform quicksort.

83. (T/F) Quicksort is a divide-and-conquer algorithm and most of the work is done in the "conquering" part.

84. (T/F) Worst case time complexity of quicksort is $\Theta(n \log n)$.

85. (T/F) Worst case time complexity of a randomized quicksort algorithm is $\Theta(n \log n)$.

86. (T/F) Quicksort is suited well for linked lists.

87. (T/F) Quicksort cannot be performed in-place.

88. (T/F) The most optimal partitioning policy for quicksort on an array we know nothing about would be selecting a random element in the array.

89. Write a recurrence relation for quicksort.

# EECS 281, Week 4: Wednesday

90. State at least four ways to optimize sorting algorithms.

91. (T/F) The fastest possible comparison sort has a worst case no better than O(n log n).

92. Give an example of a situation where using insertion sort is more efficient than using merge sort.

93. Match the sorting algorithms to the sequences, each of which represents several intermediate steps in the sorting of an array of integers.

    - Sort W

      ```
      12,  7,  8,  4, 10,  2,  5, 34, 14
       7,  8,  4, 10,  2,  5, 12, 34, 14
       4,  2,  5,  7,  8, 10, 12, 14, 34
      ```

    - Sort X

      ```
      23, 45, 12, 4, 65, 34, 20, 43
      12, 23, 45, 4, 65, 34, 20, 43
      ```

    - Sort Y

      ```
      45, 23,  5, 65, 34,  3, 76, 25
      23, 45,  5, 65,  3, 34, 25, 76
       5, 23, 45, 65,  3, 25, 34, 76
      ```

    - Sort Z

      ```
      23, 44, 12, 11, 54, 33, 1, 41
      54, 44, 33, 41, 23, 12, 1, 11
      44, 41, 33, 11, 23, 12, 1, 54
      ```

94. Given an array containing the elements `[6, 1, 7, 0, 7, 3, 9, 5]`, show how the order of the elements changes during each step of…
    94.1. bubble sort
    94.2. insertion sort
    94.3. selection sort
    94.4. mergesort
    94.5. quicksort
    94.6. heapsort

95. Some sorting algorithms are not naturally stable. Suggest a way to make any sorting algorithm stable by extending the keys.

96. Suppose we do the following:
    - Read 1,000,000 integers from a file into an array of length 1,000,000.
    - Use merge sort to sort these integers.
    - Randomly select one integer and change it.
    - Sort using algorithm S of your choice.

    Which sorting algorithm would be the fastest choice for S?

    A. Selection sort
    B. Heap sort
    C. Merge sort
    D. Insertion sort

97. Fill this table:

| Algorithm | Best case | Average case | Worst case | Memory | Stable | Notes |
|---|---|---|---|---|---|---|
| Bubble sort | | | | | | |
| Selection sort | | | | | | |
| Insertion sort | | | | | | |
| Heapsort | | | | | | |
| Merge sort | | | | | | |
| Quicksort | | | | | | |
| Random Quicksort | | | | | | |

## Miscellaneous

98. Define deep copy and shallow copy.

99. Define memory leak.

100. Explain the copy-swap method.

## Additional Practice Questions

101. Write a function template for a function named minimum. The function will have two parameters of the same type. It returns the smaller of these (either if they are equal.)

102. Implement merge function using iterators.

```
template <class InputIterator1, class InputIterator2, class OutputIterator>
OutputIterator merge(InputIterator1 first1, InputIterator1 last1,
                     InputIterator2 first2, InputIterator2 last2,
                     OutputIterator result) {









}
```

103. Given a function that partitions an array on a pivot in linear time using constant space (you don't need to worry about the pivot selection or the partitioning method), implement a function to find median using partitioning.

```
int partition(double a[], int left, int right);
```

```
int findMedian(int a[], int leftIndex, rightIndex) {









}
```

104. What is the time complexity of finding the median using partitioning?

105. Find if a linked list has a loop/cycle in O(n) time.

106. Reverse a singly-linked list.

107. Implement `isPermutation`, a function that takes two vectors and returns true if one is a permutation of the other and false otherwise. In other words, the function should return true if the two vectors contain the same elements (although possibly in different order) and false otherwise. Time complexity should be better than $\Theta(n)$.

```
bool isPermutation(const vector<int> &v1, const vector<int> &v2) {









}
```