Analysis of Algorithms

Final Exam

Code: 2137

All questions are weighted equally. Assume zero-based indexing for arrays.

1. In terms of the master recurrence theorem, where does the equation \(T(n) = T\left(\frac{n}{2}\right) + 1\) fall?

   (A) case 1  
   (B) case 3  
   (C) between case 1 and case 2  
   (D) between case 2 and case 3  
   (E) case 2

2. Using the master recurrence theorem, what is the solution of \(T(n) = 3T\left(\frac{n}{7}\right) + \frac{n}{\log n}\)?

   (A) \(\Theta(n\log n)\)  
   (B) \(\Theta(n)\)  
   (C) the master theorem cannot be used  
   (D) \(\Theta\left(\frac{n}{\log n}\right)\)  
   (E) \(\Theta(\log n)\)

3. Consider using the master recurrence theorem and the regularity condition for case 3 to solve the equation \(T(n) = 5T\left(\frac{n}{2}\right) + n^2\). What is the smallest legal value of the constant \(c\) listed for the solution? The \(\log\) (base 5) of 2 is \(\approx 0.431\). The \(\log\) (base 2) of 5 is \(\approx 2.322\).

   (A) 0.3  
   (B) no legal value is listed  
   (C) 0.1  
   (D) 0.2  
   (E) case 3 does not apply

4. Consider running the linear selection algorithm on an array with \(n = 13^k\) unique elements. What is a reasonable recurrence equation to describe the running time of the algorithm? Assume the median of medians is found with groups of thirteen.

   (A) \(T(n) = T\left(\frac{n}{13}\right) + T\left(\frac{12n}{13}\right) + \theta(n)\)  
   (B) \(T(n) = T\left(\frac{n}{13}\right) + T\left(\frac{2n}{13}\right) + \theta(n)\)  
   (C) \(T(n) = T\left(\frac{n}{13}\right) + T\left(\frac{14n}{13}\right) + \theta(n)\)  
   (D) \(T(n) = T\left(\frac{n}{13}\right) + T\left(\frac{15n}{13}\right) + \theta(n)\)  
   (E) none are reasonable

5. Consider running the linear selection algorithm on an array with \(n = 4^k\) unique elements. What is a reasonable recurrence equation to describe the running time of the algorithm? Assume the median of medians is found with groups of four and that, after sorting a group of four, the second element (index 1) is chosen as the median of that group. Hint: you are looking for the worst case, so look at finding both the minimum number of values less than the median of medians and the minimum number of values greater than the median of medians to see which leads to the largest partition.

   (A) \(T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{3n}{4}\right) + \theta(n)\)  
   (B) \(T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{5n}{4}\right) + \theta(n)\)  
   (C) \(T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{3n}{4}\right) + \theta(n)\)  
   (D) \(T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{5n}{4}\right) + \theta(n)\)  
   (E) none are reasonable

6. In an efficient decision tree (no unnecessary comparisons) for the comparison sorting of \(n\) numbers, what does the longest path from the root to a leaf represent?

   (A) the average case behavior of the worst possible algorithm  
   (B) the best case behavior of the best possible algorithm  
   (C) the worst case behavior of the worst possible algorithm  
   (D) the average case behavior of the best possible algorithm  
   (E) the worst case behavior of the best possible algorithm  
   (F) nothing important, the shortest path is what’s important  
   (G) the best case behavior of the worst possible algorithm
7. Suppose you use \( \log n \) buckets to bucket sort \( n \) numbers, uniformly distributed. What would be the expected running time of bucket sorting those numbers if you used mergesort sort to sort the individual buckets? Make sure you choose the simplest form.

(A) the correct answer is not given
(B) \( \Theta(\log n \times \frac{n}{\log n} \times (\log n - \log \log n)) \)
(C) \( \Theta(\frac{n}{\log n} \times \log n) \)
(D) \( \Theta(\frac{n}{\log n} \times (\log n - \log \log n)) \)

8. Consider using bucket sort to sort \( n \) numbers evenly distributed over the range 0 to \( m \). Roughly, what size bucket should you use?

(A) \( \frac{m}{n} \)
(B) \( \frac{n}{m} \)
(C) \( n \)
(D) \( m \)

9. Consider a node \( n \) in a typical red-black tree and all paths from \( n \) to a leaf. Which of the following is a constraint on these trees?

(A) the number of red nodes on each path is the same
(B) the number of black nodes on each path is the same
(C) each path must start with a black node
(D) the number of nodes (red or black) on each path is the same

10. Suppose one wished to allow more red nodes in a red-black tree, but still wished this new tree to have the same asymptotic behavior as before. One could allow more red nodes on any path to a leaf as long as:

(A) no red node could have a red sibling.
(B) the number of black nodes between any two red nodes is bounded by a constant.
(C) no black node could have a red parent.
(D) the number of red nodes between any two black nodes is bounded by a constant.

11. Consider an interior node with a two children, both leaves, in a red-black tree. Must that node have a sibling? If so, what is the maximum number of descendants that sibling can have?

(A) the node must have a sibling / 0
(B) the node must have a sibling / 30
(C) the correct answer is not listed
(D) the node must have a sibling / 6

12. Consider a dynamic fillable array whose capacity \( C \) grows to \( \frac{3C}{2} \) (integer division) when filled. Let \( S \) represent the number of filled slots and \( E \), the number of empty slots. Which potential function works for proving the amortized cost of an insertion into a full array is a constant? If it is not possible to show an amortized constant cost, select “not possible”. Hint: The actual cost of the insert is \( S + 1 \) and, after the insert, \( C = \frac{3S}{2} \) and \( E = \frac{S}{2} - 1 \), where \( S \) is the size immediately before the insert.

(A) \( 3S - \frac{E}{2} \)
(B) not possible
(C) the correct answer is not listed
(D) \( 3S - 2E \)

Consider the following recursive function which performs redundant computations:

```plaintext
function knapsack(w, wts, vals, i)
{
    if (i == 0) return 0;
    if (w == 0) return 0;
    if (wts[i-1] > w)
        return knapsack(w, wts, vals, i-1);
    else
        {
        var a = vals[i-1] + knapsack(w-wts[i-1], wts, vals, i-1);
        var b = knapsack(w, wts, vals, i-1);
        }
Answer the following questions about a dynamic programming solution which performs no redundant computations.

13. Suppose the initial call to the function is `knapsack(100,weights,values,20)` where the length of the weights and values arrays is 20. What are the row × column dimensions of the table needed for a dynamic programming solution? Choose the most informative answer.

   (A) the correct answer is not listed
   (B) 100 × 21
   (C) 100 × 20
   (D) the correct table is not two-dimensional
   (E) 101 × 20
   (F) 101 × 21

14. Which direction is the table populated, where low means a low index and high means a high index? Choose the most informative answer.

   (A) the table is not two-dimensional
   (B) the correct answer is not listed
   (C) from high rows to low rows, from low columns to high columns
   (D) from low rows to high rows, from low columns to high columns
   (E) from low rows to high rows, from high columns to low columns
   (F) from high rows to low rows, from high columns to low columns

15. Suppose instead of a dynamic programming solution, the results of the recursive calls were memoized in a red-black tree. What would be the asymptotic runtime of the memoized function?

   (A) \( \Theta(i \log i) \)
   (B) \( \Theta(w \log w) \)
   (C) \( \Theta(i) \)
   (D) \( \Theta(w \times i \times (\log w + \log i)) \)
   (E) \( \Theta(w) \)
   (F) the correct answer is not listed
   (G) \( \Theta(w \times i) \)

16. What is the best description of a cycle?

   (A) a path that visits all vertices
   (B) a set of edges from which the removal of any edge leaves a single path
   (C) a path plus one edge connecting the starting and ending vertices
   (D) a trail that ends up back at the starting vertex

17. Consider running Kruskal’s algorithm on the undirected, complete graph \( K_N \). What is the largest number of edges that can be processed before all remaining edges cause a cycle? Be sure to include the last edge that doesn’t cause a cycle in your count.

   (A) \( \frac{(N-1)(N-2)}{2} + 1 \)
   (B) \( \frac{N(N-1)}{2} - 1 \)
   (C) \( N(N-1) - 1 \)
   (D) \( (N-1)(N-2) + 1 \)
   (E) \( \frac{N(N-1)}{2} + 1 \)
   (F) \( \frac{(N-1)(N-2)}{2} - 1 \)
   (G) \( N(N-1) + 1 \)
   (H) the correct answer is not listed

18. Suppose Dijkstra’s algorithm for shortest paths was implemented using an ordered linked list as a priority queue. What would be the runtime contributions of \( c \), the creation of the priority queue, \( m \), the extract min operations (in total), and \( d \), the decrease key operations (in total), respectively. Assume the ordered linked list is created by merge-sorting an array of vertices, then generating the linked list from the sorted array.

   (A) \( c = V \log V, \ d = V, \ m = E \log V \)
   (B) \( c = V \log V, \ d = V, \ m = EV \)
   (C) \( c = V \log V, \ d = V \log V, \ m = E \)
   (D) \( c = V \log V, \ d = V, \ m = E \)
   (E) \( c = V^2, \ d = V \log V, \ m = EV \)
   (F) \( c = V^2, \ d = V \log V, \ m = E \log V \)
   (G) \( c = V^2, \ d = V \log V, \ m = E \)
   (H) the correct answer is not listed
19. Some one shows you a correct algorithm for an $\mathcal{NP}$-complete program $X$ that can be verified in polynomial time. Is this a new result? If so, what is that new result? Choose the best answer.

(A) Yes, problem $X$ is in $\mathcal{P}$
(B) Yes, $\mathcal{P}! = \mathcal{NP}$
(C) Yes, problem $X$ is not in $\mathcal{P}$
(D) Yes, problem $X$ is in $\mathcal{NP}$
(E) No
(F) Yes, problem $X$ is not in $\mathcal{NP}$
(G) Yes, $\mathcal{P} = \mathcal{NP}$

20. What is the best definition of an $\mathcal{NP}$-complete problem? Assume we already know the problem is in $\mathcal{NP}$. Also assume reductions can be accomplished in polynomial time and space.

(A) it can be reduced to any problem in $\mathcal{NP}$
(B) any $\mathcal{NP}$-complete problem can be reduced to it
(C) it can be reduced to any problem in $\mathcal{P}$
(D) it can be reduced to any $\mathcal{NP}$-complete problem
(E) any problem in $\mathcal{NP}$ can be reduced to it
(F) any problem in $\mathcal{P}$ can be reduced to it
Analysis of Algorithms

Exam 2

Name: ______________________

Email: ______________________

Code: 3373

All questions are weighted equally. If more than one question appears correct, choose the more specific answer, unless otherwise instructed. Assume zero-based indexing, unless otherwise instructed.

1. Consider memoizing this function:

function f(x)
{
    if (x == 0) return 0;
    if (x == 1) return 1;
    return f(x-1) + f(x-2);
}

Suppose the memoized version is called with an initial value of 7. What is the first recursive call that performs a table look up and finds a previously computed value? Do not consider calls which involve the base cases. Assume the f(x-1) call is always performed before the f(x-2) call. Hint: draw a tree of the function calls, as in f(7) calls f(6), then f(5).

(A) f(2)  (B) f(6)  (C) f(5)  (D) f(3)  (E) a previously computed value is never found  (F) f(4)

2. Consider memoizing this function:

function f(x)
{
    if (x == 0) return 1;
    return x + f(x-1);
}

Suppose the memoized version is called with an initial value of 7. What is the first recursive call that performs a table look up and finds a previously computed value? Do not consider calls which involve the base case. Hint: draw a tree of the function calls, as in f(7) calls f(6).

(A) f(3)  (B) f(2)  (C) f(5)  (D) f(6)  (E) f(4)  (F) a previously computed value is never found

3. Consider memoizing this function:

function f(x)
{
    if (x == 0) return 0;
    if (x == 1) return 1;
    if (x == 2) return 2;
    return f(x-2) * f(x-1) + f(x-3);
}

Is memoization useful (i.e. redundant computations are eliminated) and, if so, what is the dimensionality of the memoization table?

(A) no  (B) yes, 2  (C) yes, 3  (D) yes, 1
4. Consider using dynamic programming to rewrite this function:

```javascript
function g(s,i,t,j)
{
    if (i == s.length) return 0;
    if (j == t.length) return 0;
    if (s[i] == t[j])
        return maximum(g(s,i+1,t,j),g(s,i,t,j+1));
    else
        return 1 + g(s,i+i,t,j+1);
}
```

Is dynamic programming useful (i.e. redundant computations are eliminated) and, if so, what is the dimensionality of the memoization table?

(A) yes, 1
(B) yes, 3
(C) no
(D) yes, 2

5. Consider using dynamic programming to improve the efficiency of this function:

```javascript
function h(n,v,s,t)
{
    if (n == 0) return 0;
    if (s[n] > t) return h(n-1,v,s,t);
    return maximum(v[n] + h(n-1,v,s,t-v[n]),h(n-1,v,s,t));
}
```

How would the dynamic programming table be filled, using \( n \) as an index?

(A) from high \( n \) to low \( n \)
(B) \( n \) is not used as an index
(C) from low \( n \) to high \( n \)

6. Consider using dynamic programming to improve the efficiency of this function:

```javascript
function h(n,v,s,t)
{
    if (n == 0) return 0;
    if (s[n] > t) return h(n-1,v,s,t);
    return maximum(v[n] + h(n-1,v,s,t-v[n]),h(n-1,v,s,t));
}
```

How would the dynamic programming table be filled, using \( t \) as an index?

(A) from low \( t \) to high \( t \)
(B) from high \( t \) to low \( t \)
(C) \( t \) is not used as an index

For counting sort, assume array \( A \) holds the data to be sorted, array \( B \) holds the sorted data, and array \( C \) holds the counts. The index \( i \) is used to sweep the array \( A \) for counting (phase I), index \( j \) is used to sweep array \( C \) for summing the counts (phase II), and index \( k \) is used to sweep array \( A \) when placing elements from \( A \) into \( B \) (phase III). Assume \( A[k] \) is placed at \( B[C[A[k]]-1] \) and that the smallest element is placed at \( B[0] \).

7. Counting sort is:

(A) stable if \( i \) and \( j \) sweep from high to low, \( k \) low to high
(B) stable if \( i \) and \( k \) sweep from high to low, \( j \) low to high
(C) stable if \( i, j, \) and \( k \) sweep from high to low
(D) stable if \( j \) and \( k \) sweep from low to high, \( i \) high to low
(E) stable if \( i, j, \) and \( k \) sweep from low to high
(F) stable under all conditions
(G) unstable under all conditions
(H) none of the other answers are correct

8. Suppose you are to sort \( n \) numbers using counting sort. What size should the \( C \) array be if the range of numbers is greater than \( n \)?

(A) less than \( n \)
(B) there’s not enough information
(C) greater than \( n \)
(D) equal to \( n \)
9. Immediately after phase II, the value in the rightmost slot of the $C$ array is:
   (A) an $A$ array index, yielding the largest number in $A$  
   (B) an $A$ array index, yielding the smallest number in $A$  
   (C) the largest number in the $A$ array  
   (D) the number of elements in the $A$ array

10. Consider using a counting sort to sort the input array $[4,5,4,3,2,2,0]$. Immediately after phase I, the $C$ array looks like:
   (A) $[1,2,3,4,5,6]$  
   (B) $[1,0,3,1,2,1]$  
   (C) $[1,3,2,1]$  
   (D) the correct answer is not given  
   (E) $[0,1,2,3,4,5]$  
   (F) $[1,1,4,5,7,8]$

11. T or F: Consider sorting two arrays with counting sort. If the $C$ arrays look exactly the same for both inputs just after phase II, then the $C$ arrays must have looked exactly the same just before phase II.

12. Consider using a stable counting sort to sort the input array $[2,5,4,3,1,2,3,1]$ with destination array $B$. At start of phase III, the first element to be placed in $B$ is:
   (A) 1, at index 1  
   (B) 2, at index 4  
   (C) none of the other answers is correct  
   (D) 5, at index 7

13. Let $n$ be the count of numbers in a collection of base_{10} numbers. Suppose zero is the minimum number and $k$ is the maximum number in the collection. If $k = \Theta(\log n)$, then the time complexity of counting sort is:
   (A) $\Theta(n \log k)$  
   (B) $\Theta(n^2)$  
   (C) $\Theta(k)$  
   (D) $\Theta(k^2)$  
   (E) $\Theta(k \log n)$  
   (F) $\Theta(n)$

14. Let $n$ be the count of numbers in a collection of base_{10} numbers. Suppose zero is the minimum number and $k$ is the maximum number in the collection. If $k = \omega(n)$, then the time complexity of counting sort is:
   (A) $\Theta(n \log k)$  
   (B) $\Theta(n^2)$  
   (C) $\Theta(k \log n)$  
   (D) $\Theta(n)$  
   (E) $\Theta(k^2)$  
   (F) $\Theta(k)$

15. Consider using counting sort to sort $n$ numbers uniformly distributed over the range of zero to $n^k$. The asymptotic complexity of the sort, in the simplest form, will be
   (A) $\theta(k)$  
   (B) $\theta(n^k)$  
   (C) $\theta(n)$  
   (D) $\theta(n + n^k)$  
   (E) $\theta(k \log n)$  
   (F) $\theta(n + k)$

16. Consider using radix sort for sorting the following numbers:

   477, 744, 447, 774

   After the first pass, the order of the numbers is:
   (A) 774, 744, 447, 477  
   (B) 477, 447, 774, 744  
   (C) 447, 477, 744, 774  
   (D) 744, 774, 477, 447  
   (E) 447, 477, 774, 744  
   (F) the correct order is not listed
17. Consider using radix sort for sorting the following numbers:

477, 744, 447, 774

After the second pass, the order of the numbers is:

(A) 774, 477, 744, 447  
(B) 447, 747, 774, 447  
(C) 447, 747, 744, 774  
(D) 774, 744, 447, 477  
(E) 744, 447, 774, 477  
(F) the correct order is not listed

18. Let \( n \) be the count of numbers in a collection of positive, base_{10} numbers. Let \( m \) be the number of digits in the largest number in the collection. Suppose the auxiliary sort works in \( \Theta(n) \) time. Then radix sorting takes time:

(A) \( \Theta(n + m) \)  
(B) \( \Theta(n \log n) \)  
(C) \( \Theta(n \log m) \)  
(D) \( \Theta(m^n) \)  
(E) \( \Theta(m \log n) \)  
(F) \( \Theta(nm) \)  
(G) \( \frac{n^2}{\log n} \)  
(H) \( (\log n)^2 \)

19. **T** or **F**: Suppose during one pass of radix sort, there is a tie between two numbers. Since they are considered the same number, it does not matter if those two numbers swap positions.

20. Suppose you use \( \log n \) buckets to bucket sort \( n \) numbers, uniformly distributed. What would be the expected running time of bucket sort if you used merge sort to sort the individual buckets?

(A) \( (\log n)(\log (\log n)) \)  
(B) \( (\log (\log n))(\log n)^2 \)  
(C) \( (\log (\log n))^2 \)  
(D) \( n(\log n)^2 \)  
(E) \( n \log n \)  
(F) the correct answer is not given  
(G) \( \frac{n^2}{\log n} \)  
(H) \( (\log n)^2 \)

21. Consider using bucket sort to sort \( n \) numbers evenly distributed over the range 0 to \( m \). Roughly, how many buckets should you use if you wish to have sort in linear time?

(A) \( n \)  
(B) \( \frac{n}{m} \)  
(C) \( m \)  
(D) \( \frac{m}{n} \)

22. Consider using bucket sort to sort \( n \) numbers evenly distributed over the range 0 to \( n^3 \). The expected bucket count and the overall expected running time of the sort, respectively, is:

(A) \( \Theta(n) \) and linear  
(B) \( \Theta(n) \) and cubic  
(C) \( \Theta(1) \) and quadratic  
(D) \( \Theta(1) \) and linear  
(E) \( \Theta(1) \) and linear  
(F) \( \Theta(n) \) and quadratic

23. Suppose a dynamic array was implemented so that growing the array increased the capacity by 10000 elements. What is the amortized cost of the append operation? The append operation extends the filled portion by one slot (at the back).

(A) constant  
(B) log linear  
(C) quadratic  
(D) linear

24. Consider a dynamic fillable array which grows by tripling in size. Let \( S \) represent the number of filled slots and \( E \), the number of empty slots, and \( C \), the capacity of the array. Which is a valid potential function for proving the amortized cost of an insert is a constant 3? Assume the actual cost of an insert when there is room is 1 and the actual cost of an insert when there is no room is \( S+1 \).

(A) \( 2S - \frac{C}{3} \)  
(B) \( 2S - C \)  
(C) \( 2S - \frac{C}{3} \)
25. Consider a dynamic fillable array which grows by tripling in size. Which is a valid equation for calculating the total cost incurred when insertion $3^i + 1$ is made (e.g. insertions 1, 4, 10, etc)? Assume the individual cost of an insert when there is room is 1 and the individual cost of an insert when there is no room is $3^i + 1$. For example, the cost of inserting the $9^{th}$ item is 1, while the cost of inserting the $10^{th}$ item is 10. Assume the initial capacity of the array is 1.

(A) \[ \frac{5n-4}{3} \]
(B) \[ \frac{5n-1}{3} \]
(C) \[ \frac{5n-4}{2} \]
(D) \[ \frac{2n-2}{3} \]
(E) \[ \frac{2n-4}{3} \]
(F) \[ \frac{5n-3}{2} \]

26. Suppose a data structure has operation $A$ with a real cost of $2n + 1$ and operation $B$ with a real cost of $3n + 1$. After an $A$ operation, $n$ decreases to $\frac{n}{2}$ while after a $B$ operation, $n$ decreases to $\frac{n}{4}$.

Which of the following potential functions can be used to show an amortized bound of $\Theta(1)$ for operations $A$ and $B$ on this data structure?

(A) $\Phi = 3n$
(B) $\Phi = \frac{n}{2}$
(C) none of the other answers work
(D) $\Phi = 4n$

27. Suppose a data structure has operation $A$ with a real cost of 1, operation $B$ with a real cost of 1, and operation $C$ with a real cost of $n$. After an $A$ operation, $n$ increases by 1, while after a $B$ operation, $n$ stays the same. After a $C$ operation, $n$ goes to zero.

Which of the following potential functions can be used to show an amortized bound of $\Theta(1)$ for $A$, $B$, and $C$ operations?

(A) none of the other answers work
(B) $\Phi = n$
(C) $\Phi = 3n$
(D) $\Phi = 2n$

Assume min-heaps unless otherwise directed. For binomial heaps, assume that root lists (and child lists) are ordered left-to-right from low degree to high degree. Also assume when a child list is merged with a root list, the child list is placed to the left of the root list (i.e. the child list is scanned before the root list during the consolidation phase).

28. Which operations of a binomial heap are asymptotically faster than a binary heap, in the worst case?

(A) none of the other answers are correct
(B) insertion, extractMin, deletion
(C) insertion, deletion
(D) extractMin, deletion
(E) extractMin
(F) deletion
(G) insertion, extractMin
(H) insertion

29. Suppose you wish to show that the amortized cost of an insertion into a binomial heap is constant? Which amortization example in the textbook would best serve as an analogy?

(A) multipop stack
(B) dynamic array (size doubles when full)
(C) dynamic array (size increases by a constant when full)
(D) binary counter

30. Merging two binary heaps takes how much time compared to merging two binomial heaps?

(A) log linear versus linear
(B) log linear versus logarithmic
(C) logarithmic versus logarithmic
(D) log linear versus log linear
(E) linear versus linear
(F) linear versus logarithmic

31. After 15 consecutive inserts into an empty binomial heap, how many subheaps are in the root list?

(A) none of the other answers are correct
(B) 4
(C) 1
(D) 2
(E) 6
(F) 7
(G) 5
(H) 3
32. After 18 insertions into an empty binomial heap, how many subheaps are in the root list?

(A) 7  (E) 6
(B) 8  (F) 2
(C) 3  (G) 5
(D) 4  (H) none of the other answers are correct

33. T or F: Consider a sequence of \( n \) insertions into an empty binomial heap, followed by a single \textit{extractMin} operation. At this point, the number of subheaps in the root list can be calculated from \( n \).

34. Consider inserting the following values, in the order given:

\[ 3 \ 2 \ 9 \ 5 \ 6 \ 4 \ 1 \ 7 \]

into an empty binomial heap. The value 9 is found in a subheap whose root has which value?

(A) 7  (E) 1
(B) 6  (F) 5
(C) 3  (G) 2
(D) 4  (H) none of the other answers are correct

35. Consider inserting the consecutive integers from 0 to 12, inclusive and in increasing order, into an empty binomial heap. After deleting the value 5, the value 12 can be found in the subheap whose root has value:

(A) 6  (E) 0
(B) 3  (F) none of the other answers are correct
(C) 5  (G) 4
(D) 2  (H) 1

36. One expects to find the minimum value in a binomial heap in time that is:

(A) constant  (D) linear
(B) log  (E) log linear
(C) log log

37. Consider inserting the consecutive integers from 1 to 12, inclusive and not necessarily in order, into an empty binomial heap. What are the largest root value possible, after all values have been inserted?

(A) 8  (E) 10
(B) 7  (F) none of the other answers are correct
(C) 11  (G) 9
(D) 12  (H) 6
1. Consider memoizing this function:

```java
function f(x) {
    if (x == 0) return 0;
    if (x == 1) return 1;
    return f(x-1) + f(x-2);
}
```

Suppose the memoized version is called with an initial value of 7. What is the first recursive call that performs a table look up and finds a previously computed value? Do not consider calls which involve the base cases. Assume the \( f(x-1) \) call is always performed before the \( f(x-2) \) call. Hint: draw a tree of the function calls, as in \( f(7) \) calls \( f(6) \), then \( f(5) \).

(A) \( f(5) \)  (B) \( f(6) \)  (C) \( f(4) \)  (D) a previously computed value is never found  (E) \( f(3) \)  (F) \( f(2) \)

2. Consider memoizing this function:

```java
function f(x) {
    if (x == 0) return 1;
    return x + f(x-1);
}
```

Suppose the memoized version is called with an initial value of 7. What is the first recursive call that performs a table look up and finds a previously computed value? Do not consider calls which involve the base case. Hint: draw a tree of the function calls, as in \( f(7) \) calls \( f(6) \).

(A) \( f(6) \)  (B) \( f(2) \)  (C) \( f(5) \)  (D) \( f(3) \)  (E) \( f(4) \)  (F) a previously computed value is never found

3. Consider memoizing this function:

```java
function f(x) {
    if (x == 0) return 0;
    if (x == 1) return 1;
    if (x == 2) return 2;
    return f(x-2) * f(x-1) + f(x-3);
}
```

Is memoization useful (i.e. redundant computations are eliminated) and, if so, what is the dimensionality of the memoization table?

(A) yes, 2  (C) no  (B) yes, 3  (D) yes, 1
4. Consider using dynamic programming to rewrite this function:

```cpp
function g(s,i,t,j)
{
    if (i == s.length) return 0;
    if (j == t.length) return 0;
    if (s[i] == t[j])
        return maximum(g(s,i+1,t,j),g(s,i,t,j+1));
    else
        return 1 + g(s,i+i,t,j+1);
}
```

Is dynamic programming useful (i.e. redundant computations are eliminated) and, if so, what is the dimensionality of the memoization table?

(A) no  
(B) yes, 2  
(C) yes, 3  
(D) yes, 1

5. Consider using dynamic programming to improve the efficiency of this function:

```cpp
function h(n,v,s,t)
{
    if (n == 0) return 0;
    if (s[n] > t) return h(n-1,v,s,t);
    return maximum(v[n] + h(n-1,v,s,t-v[n]),h(n-1,v,s,t));
}
```

How would the dynamic programming table be filled, using n as an index?

(A) from high n to low n  
(B) from low n to high n  
(C) n is not used as an index

6. Consider using dynamic programming to improve the efficiency of this function:

```cpp
function h(n,v,s,t)
{
    if (n == 0) return 0;
    if (s[n] > t) return h(n-1,v,s,t);
    return maximum(v[n] + h(n-1,v,s,t-v[n]),h(n-1,v,s,t));
}
```

How would the dynamic programming table be filled, using t as an index?

(A) t is not used as an index  
(B) from high t to low t  
(C) from low t to high t

For counting sort, assume array A holds the data to be sorted, array B holds the sorted data, and array C holds the counts. The index i is used to sweep the array A for counting (phase I), index j is used to sweep array C for summing the counts (phase II), and index k is used to sweep array A when placing elements from A into B (phase III). Assume A[k] is placed at B[C[A[k]]-1] and that the smallest element is placed at B[0].

7. Counting sort is:

(A) stable under all conditions  
(B) stable if j and k sweep from low to high, i high to low  
(C) stable if i, j, and k sweep from high to low  
(D) stable if i and j sweep from high to low, k low to high  
(E) stable if i and k sweep from high to low, j low to high  
(F) stable if i, j, and k sweep from low to high  
(G) unstable under all conditions  
(H) none of the other answers are correct

8. Suppose you are to sort n numbers using counting sort. What size should the C array be if the range of numbers is greater than n?

(A) equal to n  
(B) greater than n  
(C) less than n  
(D) there’s not enough information
9. Immediately after phase II, the value in the rightmost slot of the $C$ array is:
   (A) an $A$ array index, yielding the smallest number in $A$  
   (B) an $A$ array index, yielding the largest number in $A$  
   (C) the largest number in the $A$ array  
   (D) the number of elements in the $A$ array

10. Consider using a counting sort to sort the input array $[4,5,4,3,2,2,0]$. Immediately after phase I, the $C$ array looks like:
   (A) $[1,2,3,4,5,6]$  
   (B) $[1,0,3,1,2,1]$  
   (C) $[1,3,2,1]$  
   (D) $[1,1,4,5,7,8]$  
   (E) the correct answer is not given  
   (F) $[0,1,2,3,4,5]$  

11. **T** or **F**: Consider sorting two arrays with counting sort. If the $C$ arrays look exactly the same for both inputs just after phase II, then the $C$ arrays must have looked exactly the same just before phase II.

12. Consider using a stable counting sort to sort the input array $[2,5,4,3,1,2,3,1]$ with destination array $B$. At start of phase III, the first element to be placed in $B$ is:
   (A) 5, at index 7  
   (B) 2, at index 2  
   (C) none of the other answers is correct  
   (D) 1, at index 0

13. Let $n$ be the count of numbers in a collection of base_{10} numbers. Suppose zero is the minimum number and $k$ is the maximum number in the collection. If $k = \Theta(\log n)$, then the time complexity of counting sort is:
   (A) $\Theta(n \log k)$  
   (B) $\Theta(n)$  
   (C) $\Theta(n^2)$  
   (D) $\Theta(k^2)$  
   (E) $\Theta(k)$  
   (F) $\Theta(k \log n)$

14. Let $n$ be the count of numbers in a collection of base_{10} numbers. Suppose zero is the minimum number and $k$ is the maximum number in the collection. If $k = \omega(n)$, then the time complexity of counting sort is:
   (A) $\Theta(n \log k)$  
   (B) $\Theta(k^2)$  
   (C) $\Theta(k)$  
   (D) $\Theta(n^2)$  
   (E) $\Theta(k \log n)$  
   (F) $\Theta(n)$

15. Consider using counting sort to sort $n$ numbers uniformly distributed over the range of zero to $n^k$. The asymptotic complexity of the sort, in the simplest form, will be:
   (A) $\theta(k \log n)$  
   (B) $\theta(n^k)$  
   (C) $\theta(n + n^k)$  
   (D) $\theta(k)$  
   (E) $\theta(n + k)$  
   (F) $\theta(n)$

16. Consider using radix sort for sorting the following numbers:
    $$477, 744, 447, 774$$
    After the first pass, the order of the numbers is:
    (A) 447, 477, 774, 744  
    (B) 774, 447, 477, 744  
    (C) 447, 477, 744, 774  
    (D) 477, 447, 774, 744  
    (E) the correct order is not listed  
    (F) 744, 774, 477, 447
17. Consider using radix sort for sorting the following numbers:

\[ 477, 744, 447, 774 \]

After the second pass, the order of the numbers is:

(A) 447, 477, 744, 774  
(D) 447, 477, 774, 744  
(B) 744, 447, 774, 477  
(E) the correct order is not listed  
(C) 774, 447, 744, 477  
(F) 774, 744, 447, 477

18. Let \( n \) be the count of numbers in a collection of positive, base-\( 10 \) numbers. Let \( m \) be the number of digits in the largest number in the collection. Suppose the auxiliary sort works in \( \Theta(n) \) time. Then radix sorting takes time:

(A) \( \Theta(n + m) \)  
(B) \( \Theta(n \log n) \)  
(C) \( \Theta(n \log m) \)  
(D) \( \Theta(m \log n) \)  
(E) \( \Theta(nm) \)  
(F) \( \Theta(nm) \)

19. T or F: Suppose during one pass of radix sort, there is a tie between two numbers. Since they are considered the same number, it does not matter if those two numbers swap positions.

20. Suppose you use \( \log n \) buckets to bucket sort \( n \) numbers, uniformly distributed. What would be the expected running time of bucket sort if you used mergesort to sort the individual buckets?

(A) \( n (\log n)^2 \)  
(B) \( (\log n)^2 \)  
(C) \( n \log n \)  
(D) \( (\log n)^2 \)  
(E) \( (\log (\log n))(\log n)^2 \)  
(F) \( \frac{n^2}{\log n} \)  
(G) the correct answer is not given  
(H) \( (\log n)(\log (\log n)) \)

21. Consider using bucket sort to sort \( n \) numbers evenly distributed over the range 0 to \( m \). Roughly, how many buckets should you use if you wish to have sort in linear time?

(A) \( \frac{n}{m} \)  
(B) \( m \)  
(C) \( \frac{n}{m} \)  
(D) \( n \)

22. Consider using bucket sort to sort \( n \) numbers evenly distributed over the range 0 to \( n^3 \). The expected bucket count and the overall expected running time of the sort, respectively, is:

(A) \( \Theta(n) \) and quadratic  
(B) \( \Theta(1) \) and linear  
(C) \( \Theta(1) \) and quadratic  
(D) \( \Theta(1) \) and cubic  
(E) \( \Theta(n) \) and linear  
(F) \( \Theta(n) \) and cubic

23. Suppose a dynamic array was implemented so that growing the array increased the capacity by 10000 elements. What is the amortized cost of the \emph{append} operation? The \emph{append} operation extends the filled portion by one slot (at the back).

(A) quadratic  
(B) linear  
(C) log linear  
(D) constant

24. Consider a dynamic fillable array which grows by tripling in size. Let \( S \) represent the number of filled slots and \( E \), the number of empty slots, and \( C \), the capacity of the array. Which is a valid potential function for proving the amortized cost of an insert is a constant 3? Assume the actual cost of an insert when there is room is 1 and the actual cost of an insert when there is no room is \( S+1 \).

(A) \( 2S - \frac{C}{3} \)  
(B) \( 2S - C \)  
(C) \( 2S - \frac{C}{2} \)
25. Consider a dynamic fillable array which grows by tripling in size. Which is a valid equation for calculating the total cost incurred when insertion $3^i + 1$ is made (e.g. insertions 1, 4, 10, etc)? Assume the individual cost of an insert when there is room is 1 and the individual cost of an insert when there is no room is $3^i + 1$. For example, the cost of inserting the $9^{th}$ item is 1, while the cost of inserting the $10^{th}$ item is 10. Assume the initial capacity of the array is 1.

(A) $\frac{5n-2}{3}$
(B) $\frac{5n-4}{3}$
(C) $\frac{5n-3}{2}$
(D) $\frac{5n-1}{3}$
(E) $\frac{5n-4}{2}$
(F) $\frac{5n-2}{2}$

26. Suppose a data structure has operation $A$ with a real cost of $2n + 1$ and operation $B$ with a real cost of $3n + 1$. After an $A$ operation, $n$ decreases to $\frac{n}{2}$ while after a $B$ operation, $n$ decreases to $\frac{n}{4}$.

Which of the following potential functions can be used to show an amortized bound of $\Theta(1)$ for operations $A$ and $B$ on this data structure?

(A) none of the other answers work
(B) $\Phi = \frac{n}{2}$
(C) $\Phi = 3n$
(D) $\Phi = 4n$

27. Suppose a data structure has operation $A$ with a real cost of 1, operation $B$ with a real cost of 1, and operation $C$ with a real cost of $n$. After an $A$ operation, $n$ increases by 1, while after a $B$ operation, $n$ stays the same. After a $C$ operation, $n$ goes to zero.

Which of the following potential functions can be used to show an amortized bound of $\Theta(1)$ for $A$, $B$, and $C$ operations?

(A) $\Phi = n$
(B) $\Phi = 3n$
(C) none of the other answers work
(D) $\Phi = 2n$

Assume min-heaps unless otherwise directed. For binomial heaps, assume that root lists (and child lists) are ordered left-to-right from low degree to high degree. Also assume when a child list is merged with a root list, the child list is placed to the left of the root list (i.e. the child list is scanned before the root list during the consolidation phase).

28. Which operations of a binomial heap are asymptotically faster than a binary heap, in the worst case?

(A) insertion, deletion
(B) none of the other answers are correct
(C) insertion
(D) extractMin, deletion
(E) insertion, extractMin, deletion
(F) insertion, extractMin
(G) extractMin
(H) deletion

29. Suppose you wish to show that the amortized cost of an insertion into a binomial heap is constant? Which amortization example in the textbook would best serve as an analogy?

(A) dynamic array (size increases by a constant when full)
(B) multipop stack
(C) dynamic array (size doubles when full)
(D) binary counter

30. Merging two binary heaps takes how much time compared to merging two binomial heaps?

(A) linear versus linear
(B) linear versus logarithmic
(C) log linear versus linear
(D) log linear versus logarithmic
(E) logarithmic versus logarithmic
(F) log linear versus log linear

31. After 15 consecutive inserts into an empty binomial heap, how many subheaps are in the root list?

(A) 6
(B) none of the other answers are correct
(C) 1
(D) 5
(E) 3
(F) 7
(G) 2
(H) 4
32. After 18 insertions into an empty binomial heap, how many subheaps are in the root list?

(A) 5  (E) none of the other answers are correct  
(B) 3  (F) 7  
(C) 6  (G) 8  
(D) 2  (H) 4  

33. T or F: Consider a sequence of \( n \) insertions into an empty binomial heap, followed by a single \( \text{extractMin} \) operation. At this point, the number of subheaps in the root list can be calculated from \( n \).

34. Consider inserting the following values, in the order given:

\[ 3 \ 2 \ 9 \ 5 \ 6 \ 4 \ 1 \ 7 \]

into an empty binomial heap. The value 9 is found in a subheap whose root has which value?

(A) 4  (E) 7  
(B) 2  (F) 5  
(C) 1  (G) none of the other answers are correct  
(D) 6  (H) 3  

35. Consider inserting the consecutive integers from 0 to 12, inclusive and in increasing order, into an empty binomial heap. After deleting the value 5, the value 12 can be found in the subheap whose root has value:

(A) 2  (E) 1  
(B) 5  (F) 6  
(C) none of the other answers are correct  (G) 3  
(D) 0  (H) 4  

36. One expects to find the minimum value in a binomial heap in time that is:

(A) log linear  (D) constant  
(B) log  (E) log log  
(C) linear  

37. Consider inserting the consecutive integers from 1 to 12, inclusive and not necessarily in order, into an empty binomial heap. What are the largest root value possible, after all values have been inserted?

(A) 7  (E) 9  
(B) none of the other answers are correct  (F) 11  
(C) 12  (G) 6  
(D) 8  (H) 10