Analysis of Algorithms

Final Exam

All questions are weighted equally. Assume zero-based indexing for arrays. Null child pointers in red-black trees are not considered nodes.

Master recurrence theorem

1. Using the master recurrence theorem, what is the solution of $T(n) = 4T(\frac{n}{3}) + \frac{n}{\log n}$? The log (base 4) of 3 is ~ 0.792. The log (base 3) of 4 is ~ 1.262.

   (A) $\Theta(\frac{n}{\log n})$
   (B) $\Theta(n^{log_3 4})$
   (C) $\Theta(n \log n)$
   (D) $\Theta(n)$
   (E) the master theorem cannot be used
   (F) $\Theta(n^{log_4 3})$

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

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

3. Consider using the master recurrence theorem to solve the equation $T(n) = 5T(\frac{n}{7}) + n$. What is the largest listed value of ϵ that can be used in the solution? The log (base 5) of 7 is ~ 1.209. The log (base 7) of 5 is ~ 0.827.

   (A) 0.25
   (B) 0.5
   (C) no legal value is listed
   (D) ϵ is not used in the solution
   (E) the master theorem cannot be used
   (F) 0.1

4. Consider using the master recurrence theorem and the regularity condition for case 3 to solve the equation $T(n) = 3T(\frac{n}{2}) + n^2$. What is the smallest legal value of the constant c listed for the solution? The log (base 3) of 2 is ~ 0.631. The log (base 2) of 3 is ~ 1.585.

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

Linear Selection

5. Consider running the linear selection algorithm on an array with n unique elements. What are the *most* number of elements greater than the median of medians? Assume the median of medians is found with groups of three, that there are an odd number of groups, and that every group has three elements. Note: you are not trying to find the number of elements guaranteed to be greater than the median of medians, but the the number of elements that are or could possibly be greater. Choose the simplest answer.

   (A) $\frac{2n}{3} + 1$
   (B) $\frac{2n}{3}$
   (C) $\frac{n}{3}$
   (D) none of the other answers are correct
   (E) $\frac{n}{3} - 1$
   (F) $\frac{n}{3} + 1$
   (G) $\frac{2n}{3} - 1$

Decision Trees

6. In an efficient decision tree (no redundant comparisons) for the comparison sorting of 3 numbers, what is the largest possible depth of a leaf? Assume the root is at depth 0, a child of the root is at depth 1, and so on. A leaf in a decision tree is the final ordering.

   (A) 3
   (B) 1
   (C) 2
   (D) 6
   (E) 4
   (F) 5

Concept: *linear sorting*

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) $\Theta(\frac{n}{\log n} \times \log n)$
   (C) $\Theta(n \log n)$
   (D) $\Theta(\frac{n}{\log n} \times (\log n - \log\log n))$
   (E) $\Theta(\log n \times \frac{n}{\log n} \times \log n)$
   (F) $\Theta(\frac{n}{\log n} \times \log \frac{n}{\log n})$
   (G) $\Theta(\log n \times \frac{n}{\log n} \times (\log n - \log\log n))$
   (H) $\Theta(\log n \times \frac{n}{\log n} \times \log \frac{n}{\log n})$

8. Consider using a linear-time bucket sort to sort n numbers evenly distributed over the range 0 to n^2. Roughly, how many buckets should you use?

   (A) none of the other answers are correct
   (B) $(\log n)^2)$
   (C) n
   (D) $\log(n^2)$
   (E) n^2
   (F) $n \log n$

self-balancing trees

9. **T** or **F**: It is possible to generate an all-black red-black tree, consisting of more than one node, with insertions only.

10. Consider all red-black trees consisting of a root node with a black child and a red child. The child nodes may or may not have descendants. What is the minimum number of nodes in such a tree?

    (A) 5
    (B) 9
    (C) 8
    (D) the correct answer is not listed
    (E) 4
    (F) 6
    (G) 3
    (H) 7

11. Consider a red-black tree whose in-order traversal yields the sequence:

    ```
    1 2 3 4 5 6 7 8 9
    ```

    What is the smallest value that could possibly reside at the root?

    (A) 4
    (B) 3
    (C) 5
    (D) 6
    (E) 1
    (F) 2

12. The most number of rotations that can occur after an insertion into an AVL tree is:

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

Concept: *amortized analysis*

13. Consider a dynamic fillable array whose capacity C grows to $\frac{4C}{3}$ when filled. Let S represent the number of filled slots and C, the total number of slots. Which potential function works for proving the amortized cost of an insertion into a full array is a constant? Remember, the potential always has to be equal to or greater than zero. Assume that things always divide evenly. Hint: The actual cost of the insert is $S_B + 1$, where S_B is the size immediately before the insert.

    (A) $3S - C$
    (B) $4S - 3C$
    (C) $2S - C$
    (D) the correct answer is not listed
    (E) $3S - 4C$
    (F) $4S - C$

Consider the following recursive function which performs redundant computations:

```
function change(amount,coins,i)
    {
```

```
if (amount == 0) return 1;
if (amount < 0 || i < 0) return 0;

var with = change(amount-coins[i],coins,i);
var without = change(amount,coins,i-1);
return with + without;
}
```

Answer the following questions about a dynamic programming solution which performs no redundant computations.

14. Suppose the initial call to the function is `change(total,coins,size-1)`. where *size* is the length of the *coins* array. What is the minimum sized table needed for a dynamic programming solution? Assume a straightforward transformation of the original function.

    (A) *total* rows $\times$ *size* + 1 columns
    (B) *total* rows $\times$ *size* - 1 columns
    (C) *total* + 1 rows $\times$ *size* columns
    (D) *total* + 1 rows $\times$ *size* - 1 columns
    (E) *total* - 1 rows $\times$ *size* columns
    (F) *total* rows $\times$ *size* columns
    (G) the correct answer is not listed
    (H) *total* + 1 rows $\times$ *size* + 1 columns

15. 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 correct answer is not listed
    (B) from low rows to high rows, from low columns to high columns
    (C) from high rows to low rows, from high columns to low columns
    (D) from high rows to low rows, from low columns to high columns
    (E) from low rows to high rows, from high columns to low columns

16. Suppose instead of a dynamic programming solution, the results of the recursive calls were memoized in an AVL tree. What would be the asymptotic runtime of the memoized function, where t is *total* and s is *size*?

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

17. Consider 6 unconnected vertices, numbered 0 through 5. Suppose we add as many undirected edges with unique weights as possible, yielding the complete graph K_6. If we use the following strategy to add edges:

```
function addEdges(g,v) //g is a graph; v is the number of vertices
    {
    var i,j,weight = 1;
    for (i = 0; i < v; ++i)
        for (j = i+1; j < v; ++j)
            {
            addAnEdge(g,i,j,weight) //add edge between i and j with weight
            weight = weight + 1;
            }
    }
```

what is the weight of the minimum spanning tree?

    (A) 55
    (B) 5
    (C) 50
    (D) 35
    (E) 45
    (F) none of the other answers are correct
    (G) 20
    (H) 15

18. Suppose Prim's algorithm for minimum spanning trees was implemented using a binomial heap 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.

(A) $c = V \log V$, $m = V$, $d = EV$
(B) $c = V^2$, $m = V \log V$, $d = E$
(C) $c = V^2$, $m = V \log V$, $d = EV$
(D) $c = V$, $m = V \log V$, $m = E \log V$
(E) the correct answer is not listed
(F) $c = V$, $m = V$, $d = E$
(G) $c = V^2$, $m = V \log V$, $d = E \log V$
(H) $c = V \log V$, $m = V$, $d = E \log V$

19. Some one shows you a correct algorithm for an $\mathcal{NP}$ problem X that finds a solution in polynomial time. This person then shows you a polynomial time reduction from X to an $\mathcal{NP}$-complete problem. 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, $\mathcal{P} \neq \mathcal{NP}$
(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. Suppose you wish to prove the Hamiltonian Cycle Problem (HCP) is $\mathcal{NP}$-complete using the fact that the Hamiltonian Path Problem (HPP) is $\mathcal{NP}$-complete. Starting with graph G, what strategy should you use?

(A) add a vertex to G with edges to the original vertices in G and solve with HCP.
(B) remove two edges from G and solve with HPP.
(C) remove a vertex from G and solve with HPP.
(D) add a vertex to G with edges to the original vertices in G and solve with HPP.
(E) remove a vertex from G and solve with HCP.
(F) remove two edges from G and solve with HCP.