Computer Science 477/677 Spring 2019

University of Nevada, Las Vegas Computer Science 477/677 Spring 2019

Practice for Final Examination May 15, 2019

The entire practice test is 655 points.

1. True or False. Write "O" if the answer is not known to science at this time. [5 points each]

   (a) ________ Computers are so fast today that complexity theory is only of theoretical, but not practical, interest.

   (b) ________ The inverse Ackermann function, $\alpha(n)$, grows so slowly that, from a practical (as opposed to theoretical) point of view, it might as well be constant.

   (c) ________ If a problem is $\mathcal{NP}$-complete, there is no polynomial time algorithm which solves it.

   (d) ________ Quicksort takes $\Theta(n \log n)$ time on an array of size n.

   (e) ________ Planar graphs are sparse.

   (f) ________ Acyclic graphs are sparse.

   (g) ________ Acyclic directed graphs are sparse.

2. Fill in the blanks. [5 points each blank.]

   (a) If a planar graph $\mathcal{G}$ has 20 edges, then the number of vertices of $\mathcal{G}$ cannot be less than ________________. (You must give the best possible answer, exactly. No partial credit.)

   (b) A directed acyclic graph with 5 vertices cannot have more than 10 arcs, and a directed acyclic graph with 6 vertices cannot have more than 15 arcs. A directed acyclic graph with 10 vertices cannot have more than ________________ arcs. (You must give the best possible answer, exactly. No partial credit.)

   (c) A directed acyclic graph with 20 arcs cannot have fewer than ________________ vertices.

   (d) The height of a binary tree with 50 nodes is at least ________________. (You must give the best possible answer, exactly. No partial credit.)

   (e) In ________________________________ hashing, there are no collisions.

   (f) If separate chaining is used to resolve collisions in a hash table with n items and n places in the array and if the hash function is pseudo-random, then approximately ________________% of the places will have more than two items. Pick the best answer from among these choices: (0%, 1%, 2%, 4%, 8%, 16%, 32%)

   (g) The time complexity of every comparison-based sorting algorithm is ________________ . (Your answer should use Ω notation.)

   (h) ________________________________ sorting is not comparison-based.

(i) The infix expression $(x + y) * z$ is equivalent to the prefix expression ______________________________ and the postfix expression ______________________________ .

(j) What is the **only** difference between the abstract data types *queue* and *stack*?

____

(k) The items stored in a priority queue (that includes stacks, queues, and heaps) represent ______________________ ______________________.

(l) Name a divide-and-conquer searching algorithm.

____

(m) Name two divide-and-conquer sorting algorithms.

____

____

(n) The following is pseudo-code for which sorting algorithm we've discussed?

____

```
int x[n];
obtain values of x;
for(int i = n-1; i > 0; i--)
 Find the largest element of x[0], ... x[i] and swap it with x[i]
```

(o) The following is pseudo-code for which sorting algorithm we've discussed?

____

```
int x[n];
obtain values of x;
bool finished = false;
for(int i = n-1; i > 0 and not finished; i--)
 {
  finished = true;
  for(int j = 0; j < i; j++)
    if(x[j] > x[j+1])
     {
      swap(x[j],x[j+1]);
      finished = false;
     }
 }
```

3. Give the asymptotic complexity, in terms of `n`, of each of the following code fragments. [10 points each]

(a)
```
for(int i = n; i > 1; i = i/2)
  cout << "hello world" << endl;
```

(b)
```
for(int i = 1; i < n; i++)
  for(int j = 1; j < i; j = 2*j)
    cout << "hello world" << endl;
```

(c)
```
for(int i = 1; i < n; i++)
  for(int j = i; j < n; j = 2*j)
    cout << "hello world" << endl;
```

(d)
```
for(int i = 2; i < n; i = i*i)
  cout << "hello world" << endl;
```

4. [30 points] Name two problems which are known to be $\mathcal{NP}$-complete, and one problem that is known to be undecidable.

----

5. Solve the recurrences. Give asymptotic answers in terms of n, using either O, Ω, or Θ, whichever is most appropriate.

   (a) [10 points] $F(n) = 2F\left(\frac{n}{2}\right) + n$

   (b) [10 points] $F(n) \geq 4F\left(\frac{n}{2}\right) + n^2$

   (c) [10 points] $F(n) = F(n-1) + \frac{n}{4}$

   (d) [10 points] $F(n) \leq F\left(\frac{n}{2}\right) + F\left(\frac{n}{4}\right) + F\left(\frac{n}{5}\right) + n$

   (e) [10 points] $F(n) = F\left(n - \sqrt{n}\right) + n$

   (f) [10 points] $F(n) = F(\log n) + 1$

6. [30 points] Use dynamic programming to compute the length of the longest common subsequence of the strings "011011001" and "1010011001."

7. Solve each of the following recurrences, giving the answer in terms of O, Θ, or Ω, whichever is most appropriate [10 points each].

   (a) $T(n) < T(n-2) + n^2$

   (b) $F(n) \geq F(\sqrt{n}) + \lg n$

   (c) $G(n) \geq G(n-1) + n$

   (d) $F(n) = 4F(n/2) + n^2$.

   (e) $H(n) \leq 2H(\sqrt{n}) + O(\log n)$.

   (f) $K(n) = K(n - \sqrt{n}) + 1$.

   (g) $F(n) = 4F\left(\frac{3n}{4}\right) + n^5$ (No, you don't need a calculator.)

8. Find the asymptotic complexity, in terms of n, for each of these fragments, expressing the answers using O, Θ, or Ω, whichever is most appropriate.

   (a)
   ```
   for(i = 0; i < n; i = i+1);
   cout << "Hi!" << endl;
   ```

   (b)
   ```
   for(i = 1; i < n; i = 2*i);
   cout << "Hi!" << endl;
   ```

   (c)
   ```
   for(i = 2; i < n; i = i*i);
   cout << "Hi!" << endl;
   ```

   (d) The following code models the first phase of heapsort.

   ```
   for(int i = n; i > 0; i--)
    for(int j = i; 2*j <= n; j = 2*j)
     cout << "swap" << endl;
   ```

(e) The following code models the second phase of heapsort.

```
for(int i = n; i > 0; i--}
 {
  cout << "swap" << endl;
  for(int j = 1; 2*j <= i; j = 2*j)
   cout << "swap" << endl;
 }
```

(f) The following code models insertion of n items into an AVL tree.

```
for(int i = 1; i < n; i++)
 for(int j = n; j > 0; j = j/2)
   cout << "check AVL property and possibly rotate" << endl;
```

9. Solve each of the following recurrences, expressing the answers using O, Θ, or Ω, whichever is most appropriate. [10 points each]

(a) $F(n) = F(n/2) + 1$

(b) $F(n) = F(n-1) + O(\log n)$

(c) $F(n) = F\left(\frac{n}{2}\right) + 2F\left(\frac{n}{4}\right) + n$

(d) $F(n) = F\left(\frac{3n}{5}\right) + F\left(\frac{4n}{5}\right) + n^2$
Use the same method you used for the previous problem. Hint: $3^2 + 4^2 = 5^2$.

(e) $F(n) = F(n-2) + n$

10. (a) Use Huffman's algorithm to construct an optimal prefix code for the alphabet $\{A, B, C, D, E, F\}$ where the frequencies of the symbols are given by the following table.

A	2
B	8
C	9
D	3
E	7
F	5

(b) Use the Hu-Tucker algorithm to find an optimal alphabetic prefix-free code on those same letters using the same distribution.

11. [10 points]

Write pseudo-code for binary search.

12. Find the asymptotic complexity, in terms of n, for each of these fragments, expressing the answers using O, Θ, or Ω, whichever is most appropriate. [10 points each]

(a)
```
for(int i = 1; i*i < n; i++)
  cout << "Hi!" << endl;
```

(b)
```
for(int i = n; i > 1; i = sqrt(i));
  cout << "Hi!" << endl;
```

Find the asymptotic time complexity, in terms of n, for each of these code fragments, expressing the answers using O, Θ, or Ω, whichever is most appropriate. [10 points each]

(a)
```
int f(int n)
  {
   if (n < 2) return 1;
   else return f(n-1)+f(n-1);
  }
```

(b)
```
void hello(int n)
   {
    if(n >= 1)
     {
      for(int i = 1; i < n; i++)
       cout << "Hello!" << endl;
      hello(n/2);
      hello(n/2);
     }
   }
```

13. [20 points] Define the Collatz function as follows:

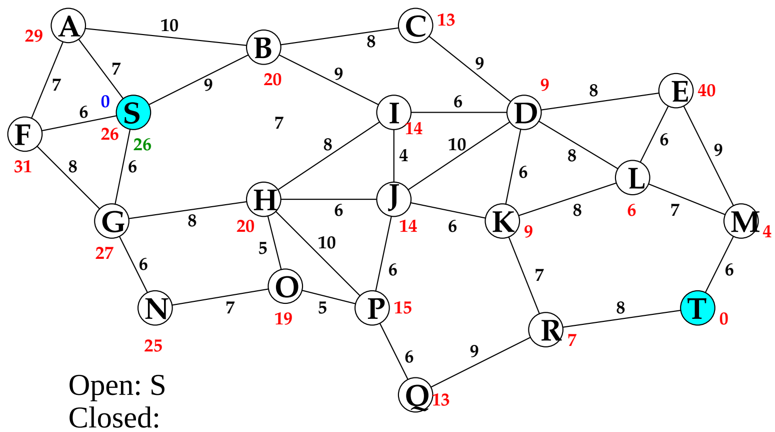
```
int collatz(int n)
  {
    assert(n > 0);
    if(n == 1) return 0;
    else if (n%2) return collatz(3*n+1); // n is odd, greater than 1
    else return collatz(n/2); // n is even
   }
```

Write pseudo-code for a memoization algorithm which prints collatz(n) for all n from 1 to 1000.

14. [20 points] Give pseudocode for a recursive algorithm which computes the median of the union of two sorted lists in logarithmic time.

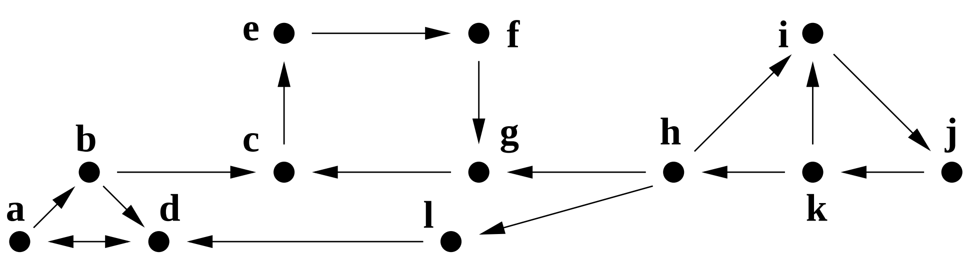
15. [20 points] Describe a randomized algorithm which finds the k^{th} smallest element of an unsorted list of n distinct numbers, for a given $k \leq n$, in $O(n)$ expected time. (By "distinct," I mean that no two numbers in the list are equal.)

16. [20 points] Walk through the A^* algorithm for the following weighted graph to find the shortest path from S to T. Edge weights are shown in black, and the values of the heuristic are shown in red.



Open: S
Closed:

17. [20 points] Walk through Kosaraju's algorithm to find the strong components of the directed graph.



18. [20 points] Give pseudocode for the Bellman-Ford algorithm.

19. [20 points] Give pseudocode for the Floyd-Warshall algorithm.