University of Nevada, Las Vegas Computer Science 477/677 Spring 2020

Examination April 30, 2020

Name:____

The exam is take-home, open book, open notes, open internet. You must finish by midnight of April 30. Scan and email the completed examination paper to your TA, Shekhar Singh. The email must have an April 30 time stamp.

The entire examination is 285 points.

1. Solve the recurrences. Give asymptotic answers in terms of n, using either O, Ω, or Θ, whichever is most appropriate. Use whichever technique is appropriate for each problem. [10 points each]

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

   (b) $F(n) \leq F(n-3) + 3\log n$

   (c) $F(n) = F(\sqrt{n}) + 1$; (Hint: use a substitution. Introduce the variable m and let $m = \log_2 n$, and introduce the function G such that $G(m) = F(2^m) = F(n)$. Then find a new recurrence using the function G. Solve that recurrence, and then substitute back.. to solve the original recurrence.)

   (d) $F(n) \leq F(n/2) + F(n/3) + n$;

   (e) $F(n) \leq 2F(n-1) + 1$

   (f) $F(n) \geq 4F(n/2) + n$

   (g) $F(n) = F(n - \sqrt{n}) + \sqrt{n}$

Name:____

2. [10 points] Find an integer K such that the solution to the recurrence below is $F(n) = \Theta(n^2 \log n)$.

   $F(n) = F(3n/5) + K\,F(n/5) + n^2$

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

   (a)
   ```
   for(int i = n; i > 1; i = log2(i)))
   ```

   (b)
   ```
   for(int i = n; i > 1; i = i/2)
     for(int j = 1; j<i; j++)
   ```

   (c)
   ```
   for(int i = n; i > 1; i = i/2)
     for(int j = i; j<n; j++)
   ```

   (d)
   ```
   for(int i = 0; i < n; i++)
     for(int j = 0; j < i*i; j++)
   ```

   (e)
   ```
   for(int i = 2; i < n; i=i*i)
   ```

   (f)
   ```
   for(int i = n; i > 1; i = sqrt(i))
   ```

   (g)
   ```
   for(int i = n; i > 1; i = sqrt(i))
     for(int j = 0; j < i; j++)
   ```

Name:____

4. Arrays. There is a built-in implementation of the abstract data type Array in C++. However, sometimes it is best to use a different implemntation.

   (a) [10 points] What space saving data structure would you use to implement a 1000×1000 2-dimensional array where there are 2000 non-zero entries, the rest zero?

   (b) [10 points] What space saving data structure would you use to implement a 10000×10000 2-dimensional array which has 100 non-zero entries, the rest zero?

5. Graphs.

   (a) [10 points] What does it mean to say that a graph is "sparse"?

   (b) [10 points] What data structure would you use to implement a graph with 1000 vertices and 10000 edges?

   (c) [5 points] What is the maximum number of edges a planar graph with 5 vertices can have?

6. [20 points] Construct an optimal prefix-free code for the alphabet a,b,c,d,e,f with the frequencies given by the table below.

a	12
b	6
c	7
d	15
e	22
f	4
g	5

Name:____

7. [20 points] In my video at https://www.youtube.com/watch?v=iKA4URlAtKo I show how to implement a min-heap as an array. For example, a min-heap of size 6 could be implemented as the following array:

D	F	H	M	L	R

If *deletemin* is executed, the array changes in a sequence of steps. Show those steps.

Subsequently, E is inserted. Show the array after each step.

8. [20 points] Explain, using diagrams and text, but not pseudo-code, the linked list implementation of a stack. Represent the items on the stack by capital letters.

   (a) Illustrate the stack with item K, F, L, A, in that order, where K is the top item.

   (b) Starting with the previous stack, illustrate **pop.**

   (c) Starting with the previous stack, illustrate **push,** where the item D is pushed onto the stack.

Name:____

9. Let $G = (V, E)$ be a weighted directed graph with n vertices and m edges. The vertices are $v_1, v_2, \ldots v_n$ and the edges are $e_1, e_2, \ldots e_m$.

   Each e_j is a directed edge (x_j, y_j), where $x_j, y_j \in V$. $W[e_j] = W[x_j, y_j]$ is the given weight of the edge e_j. Note that a vertex could have several names. For example, If e_3 is a directed edge from v_2 to v_5, v_2 has the alternative name x_3 and v_5 has the alternative name y_3, and $W[e_3] = W[x_3, y_3] = W[v_2, v_5]$.

   Here is pseudo-code for the Belman-Ford algorithm for the single source shortest path problem for G, where v_1 is the source. We will write $F[v_i]$ for the smallest weight of any path from v_1 to v_i that we have found so far, and we write $B[v_i]$ for the backpointer of that path. The arrays V and B are the outputs of the program.

   (a) [10 points] Fill in the missing line which assigns a backpointer.

   (b) [20 points] The main outer loop iterates $n - 1$ times. However, in most practical situations, The values of V are updated only during the first few iterations. Insert code (4 lines) to end the outer loop if there are no further changes to V. Assume that G has no negative cycle.

```
F[v_1] = 0;
for all i from 2 to n
  F[v_i] = ∞;

for all t from 1 to n-1

    {

      for all j from 1 to m
        if (V[x_j] + W[e_j] < V[y_j])
          {
            V[y_j] = V[x_j] + W[e_j]
                                  //assign a backpointer


          }
    }
```