University of Nevada, Las Vegas Computer Science 477/677 Spring 2022

Final Examination May 9, 2022

Name:_____

No books, notes, scratch paper, or calculators. Use pen or pencil, any color. Use the rest of this page and the backs of the pages for scratch paper. If you need more scratch paper, it will be provided. If you want anything on extra pages to be graded, staple those pages to your test and write, "Please grade this page."

The entire examination is 460 points.

1. In each blank, write Θ if correct, otherwise write O or Ω, whichever is correct. [5 points each]

   (a) $n^2 = O\ (n^3)$

   (b) $\log(n^2) = \Theta\ (\log(n^3))$

   (c) $\log(n!) = \Theta\ (n \log n)$

   (d) $\log_2 n = \Theta\ (\log_4 n)$

   (e) $n^{0.000000000001} = \Omega\ (\log n)$

2. True or False. Write "T" or "F." If the answer is not known to science at this time, write "O" for "Open." [5 points each]

   (a) **T** There is a mathematical statement which is true, yet cannot be proven.

   (b) **T** The subproblems of a dynamic program form a directed acyclic graph.

   (c) **F** Kruskal's algorithm uses dynamic programming.

   (d) **T** A hash function should appear to be random, but cannot actually be random.

   (e) **F** Open hashing uses open addressing.

   Solve each recurrence, expressing the answer as an asymptotic function of n. Use O, Ω, or Θ, whichever is most appropriate.[1] [10 points each]

   (a) $F(n) \leq 2F(n/2) + n^2$

   $F(n) = O(n^2)$

   (b) $F(n) \geq 3F(n/9) + 1$

   $F(n) = \Omega(\sqrt{n})$

[1] Although I haven't in previous tests, this time I will take at least half off for expressing the answer without using asymptotic notation, or for using the wrong one.

(c) $F(n) = F(3n/5) + 4F(2n/5) + n^2$

$F(n) = \Theta(n^2 \log n)$

(d) $F(n) = F(n/5) + F(7n/10) + n$

$F(n) = \Theta(n)$

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

(a)
```
for(int i = 0; i < n; i++)
  for(int j = n; j > i; j = j/2)
```

$\Theta(n)$

(b)
```
for(int i = 0; i < n; i++)
  for(int j = i; j > 0; j = j/2)
```

$\Theta(n \log n)$

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

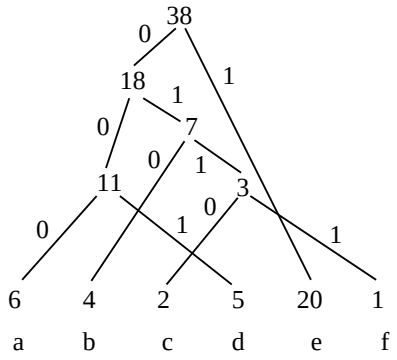
$\Theta(\log \log n)$

(d)
```
for(int i = 1; i*i < n; i++)
```

$\Theta(\sqrt{n})$

4. Find an optimal prefix code for the alphabet $\{a, b, c, d, e, f\}$ where the frequencies are given in the following array.

a	6	000
b	4	010
c	2	0110
d	5	001
e	20	1
f	1	0111



5. Fill in the blanks. [10 points each]

In problems (a) and (b), let n be the number of vertices, m the number of arcs, and p the maximum number of arcs in the shortest path between any two vertices.

(a) The asymptotic complexity of the Floyd/Warshall algorithm is $\Theta(n^3)$.

(b) The asymptotic complexity of Dijkstra's algorithm algorithm is $O(m \log n)$.

(c) A **perfect** hash function fills the hash table exactly with no collisions.

(d) **Huffman's** algorithm finds a binary code so that the code for one symbol is never a prefix of the code for another symbol.

(e) **Huffman's** algorithm and **Kruskal's** algorithm are greedy algorithms we've studied this semester.

(f) **Mergesort**, **quicksort** and **binary search** are divide-andconquer algorithms that we've studied this semester.

(g) An acyclic directed graph with 9 vertices must have at least **9** strong components. (Must be exact answer.)

(h) In **open hashing** there can be any number of items at a given index of the hash table.

(i) The asymptotic expected time to find the median item in an unordered array of size n, using a randomized selection algorithm, is $O(n)$.

(j) If $h(x)$ is already occupied for some data item x, a **probe sequence** is used to find an unoccupied position in the hash table.

(k) If a directed acyclic graph has n vertices, it must have n strong components.

(l) If a planar graph has 10 edges, it must have at least **6** vertices.

(m) If G is a weighted graph, then it is impossible to solve the all pairs shortest path problem for G if G has a **negative weight cycle**. (Or, simply a negative cycle).

(n) Fill in the blank with one letter. If all arc weights are equal, then Dijkstra's algorithm visits the vertices in same order as **BFS**.

6. [20 points] What is the loop invariant of the loop in the following function?

```
float product(float x, int n)
 {
  // assert(n >= 0);
  float z = 0.0;
  float y = x;
  int m = n;
  while(m > 0)
   {
    if(m%2) z = z+y;
    m = m/2;
```

```
    y = y+y;
   }
  return z;
 }
```

$x * n = z + y * m$

8. [20 points] Compute the Levenstein distance between *abcdafg* and *agbccdfc*. Show the matrix.

		a	g	b	c	c	d	f	c
	0	1	2	3	4	5	6	7	8
a	1	0	1	2	3	4	5	6	7
b	2	1	1	1	2	3	4	5	6
c	3	2	2	2	1	2	3	4	5
d	4	3	3	3	2	2	2	3	4
a	5	4	4	4	3	3	3	3	4
f	6	5	5	5	4	4	4	3	4
g	7	6	5	6	5	5	5	4	4

I gave you an incorrect formula for this problem. In case the symbols of the two strings do not match, such as at several places in the matrix and in the example below, the lower right corner is computed by the formula $w = \min(x + 1, y + 1, z + 1)$.

		a
	x	y
b	z	w

For example:

		a
	3	4
b	4	4

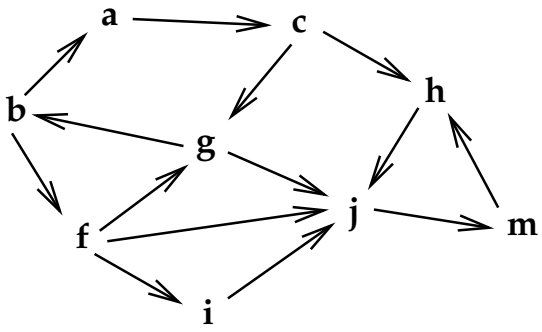
The Levenshtein edit distance between *abcdafg* and *agbccdfc* is 4, the number in the lower right corner of the matrix.

9. [20 points] You need to store Pascal's triangle in row-major order into a 1-dimensional array P whose indices start at 0. The triangle is infinite, but you will only store $\binom{n}{k}$ for $n < N$. Write a function I such that $P[I(n,k)] = \binom{n}{k}$ for $0 \le k \le n < N$. For example, $I(3,2) = 8$.

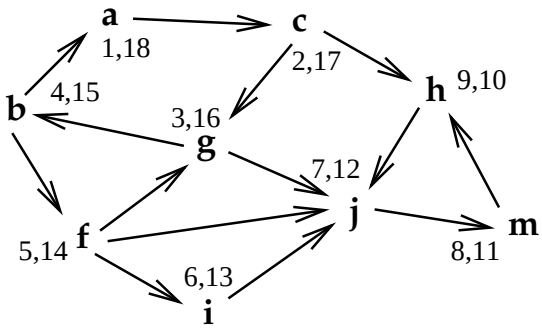
```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
```

```
int I(int n, int k)
 {
  // the position of n choose k in the linear array
  assert(k >= 0 and n >= k and n < N);
  int indx = n*(n+1)/2+k;
  return indx;
 }
```

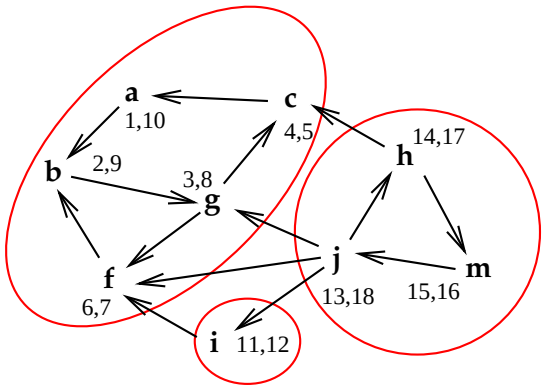
10. [20 points] Use the DFS method to find the strong components of the digraph shown in Figure (a) below, and illustrate those components in Figure (d). Use the other two figures for your work.



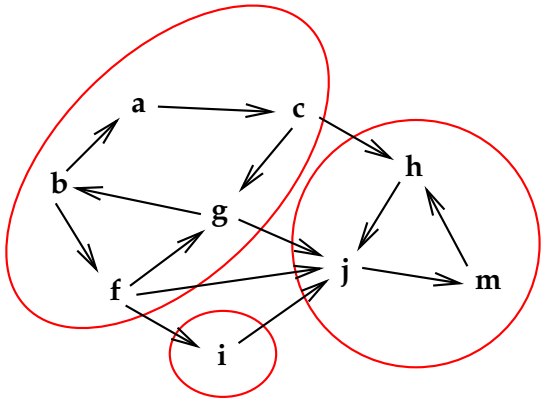
(a)



(b)

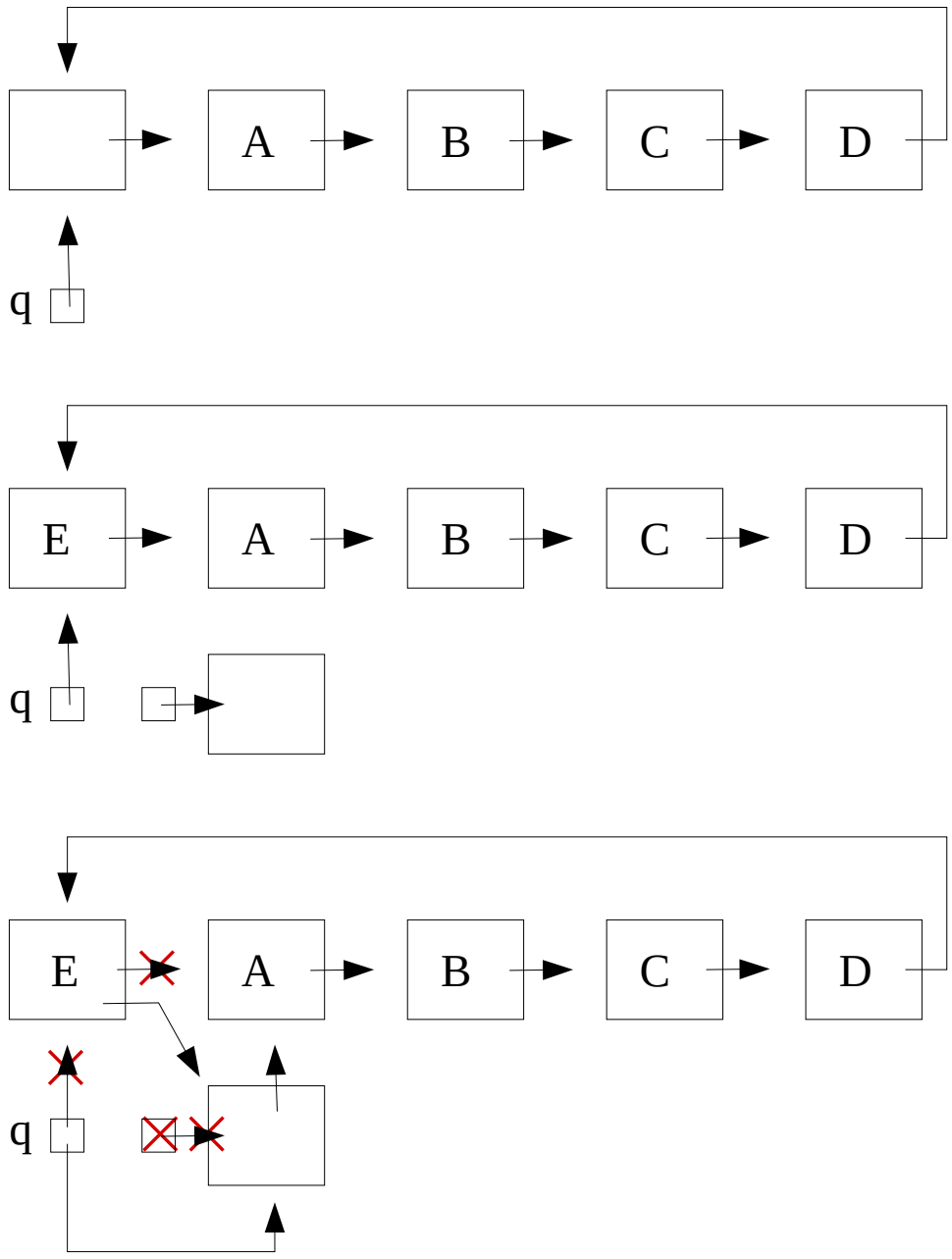


(c)



(d)

11. [20 points]Sketch a circular linked list with dummy node which implements a queue. The queue has four items. From front to rear, these are A, B, C, D, and show the insertion of E into the queue. Show the steps. Don't erase deleted objects; instead, simply cross them out.



12. In class, we implemented a minqueue as an almost complete binary tree implemented as an array.

    (a) [15 points] Suppose the minheap is initialized as shown in the first line of the array shown below. Show the evolution of the structure when deletemin is executed.

A	C	F	D	Q	H	L	R	Z
Z	C	F	D	Q	H	L	R	
C	Z	F	D	Q	H	L	R	
C	D	F	Z	Q	H	L	R	
C	D	F	R	Q	H	L	Z	

(b) [15 points] Starting from the final configuration above, show the evolution of the structure when B is inserted.

C	D	F	R	Q	H	L	Z	
C	D	F	R	Q	H	L	Z	B
C	D	F	B	Q	H	L	Z	R
C	B	F	D	Q	H	L	Z	R
B	C	F	D	Q	H	L	Z	R

13. [30 points] You are given an acyclic directed graph $G = (V, E)$ where each arc is weighted. If (x, y) is an arc, we write $w(x, y)$ for the weight of that arc. Describe a dynamic programming algorithm which calculates the directed path through G of maximum weight.

There are two ways to set this problem up. I want you to use the right-to-left method, not the left-to-right. There is one subproblem for each vertex v, namely to compute $M[v]$, the maximum weight of any directed path starting at v. Compute the forward pointer `forw`$[v]$ for each vertex. Explain how those pointers are used to find the path.

Let $V = \{v_1, v_2, \ldots v_n\}$, given in topological order, that is, if there is an arc from v_i to v_j, then $i < j$.

for all $v \in V$, $M[v] = 0$ and `forw`$[v]] = v$.
    (default to detect the end of a maximal weight path)
for all i from n to 1 (reverse order)
    for all j such that $v[j]$ is an outneighbor of $v[i]$
        temp $= M[j] + w(v[i], v[j])$
        if(temp $> M[i]$)
            $M[i]$ = temp and `forw`$[v[i]] = v[j]$

The following code writes the maximum weight path starting at $v[i]$.

write $v[i]$
while($M[v_i] > 0$)

$v[i]$ =forw$[v[i]]$
write $v[i]$

14. [30 points] Write pseudocode for the Bellman-Ford algorithm. Be sure to include the shortcut that ends the program when the final values have been found.

    Assume there are m arcs, numbered from 0 to $m-1$, and the k^{th} arc is from $s[k]$ to $t[k]$ and has weight $w[k]$. The vertices are numbers $0 \ldots n-1$ and the source vertex is 0. For each vertex i, The algorithm computes $V[i]$, the minimum weight of a path from 0 to i, and the backpointer $b[i]$ of that path. The algorithm uses a shortcut.

```
V[0] = 0;
for(int i = 1; i < n; i++)
  V[i] = infinity;
bool done = false;
while (not done)
  {
   done = true;
   for(int k = 1
   for(int k = 0; k < m; k++)
    {
     temp = V[s[k]] + w[k];
     if(temp < V[t[k]])
      {
       V[t[k]] = temp;
       done = false;
      }
    }
  }
```