University of Nevada, Las Vegas Computer Science 477/677 Fall 2025

Answers to Examination October 23, 2025

Name:_____

The entire examination is 280 points.

1. Write the asymptotic time complexity, in terms of n, of each of these code fragments.

   (i) [5 points]

   ```
   for(int i = 0; i < n; i = 2*i)
     cout << "Hello world" << end;
   ```

   $\Theta(\log n)$

   (ii) [5 points]

   ```
   for(int i = 1; i < n; i++)
    for(int j = i; j < n; j = 2*j)
     cout << "Hello world" << end;
   ```

   $\Theta(n)$

   (iii) [5 points]

   ```
   for(int i = 1; i*i < n; i++)
     cout << "Hello world" << end;
   ```

   $\Theta(\sqrt{n})$

   (iv) [5 points]

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

   $\Theta(\log \log n)$

   (v) [5 points] In the following problem, sqrt means square root.

   ```
   for(int i = n; i > 2; i = sqrt{i})
    cout << "Hello world." << endl;;
   ```

   $\Theta(\log \log n)$

   (vi) [5 points]

   ```
   for(int i = n; i > 0; i = i/2)
    cout << "Hello world." << endl;;
   ```

   $\Theta(\log n)$

2. [15 points] Name three different divide-and-conquer algorithms we've discussed this semester.

   Quicksort, mergesort, binary search.

3. Solve the following recurrences. Express the answers using Θ notation.

(i) [5 points] $F(n) = 3F(2n/3) + 3F(n/3) + n^3$
$F(n) = \Theta(n^3 \log n)$

(ii) [5 points] $F(n) = F(n/2) + 2F(n/4) + 1$
$F(n) = \Theta(n)$

(iii) [5 points] $F(n) = F(n/2) + F(n/3) + n$
$(n) + \Theta(n)$

(iv) [5 points] $F(n) = F(n/3) + 1$
$F(n) = \Theta(\log n)$

(v) [5 points] $F(n) = 3F(n/3) + n$;
$F(n) = \Theta(n \log n)$

(vi) Duplicate, not graded.

(vii) [5 points] $F(n) = F(n/3) + n$;
$F(n) = \Theta(n)$

(viii) [5 points] $F(n) = 9F(n/3) + n$
$F(n) = \Theta(n^2)$ ($\log_3 9 = 2$ since $3^2 = 9$)

(ix) [5 points] $F(n) = F(n-1) + \log n$;
$F(n) = \Theta(n \log n)$

(x) [5 points] $F(n) = F(3n/5) + F(4n/5) + n^2$
$F(n) = \Theta(n^2 \log n)$

(xi) [5 points] $F(n) = F(12n/13) + F(5n/13) + n$
$F(n) = \Theta(n^2)$

(xii) [5 points] $F(n) = 2F(n/2) + n$
$F(n) = \Theta(n \log n)$

(xiii) [5 points] $F(n) = 2F(n/2) + 1$
$F(n) = \Theta(n)$

(xiv) [5 points] $F(n) = F(n/2) + 1$
$F(n) = \Theta(\log n)$

(xv) [5 points] $F(n) = F(n-1) + n$;
$F(n) = \Theta(n^2)$

(xvi) [5 points] $F(n) = F(n - \log n) + \log^2 n$
$F(n) = \Theta(n \log n)$

(xvii) [5 points] $F(n) = F(n - \sqrt{n}) + \sqrt{n}$;
$F(n) = \Theta(n)$

(xviii) [5 points] $F(n) = F(n - \sqrt{n}) + 1$
$F(n) = \Theta(\sqrt{n}) = \Theta(n^{1/2})$

(xix) [5 points] $F(n) = 2F(n-1) + 1$
$F(n) = \Theta(2^n)$

4. The following is implementation of binary search tree. Write C++ code for postorder traversal of the tree rooted at mainroot.

```
const int N = 20;

struct treenode;
typedef treenode*tree;
struct treenode
 {
  int item;
  tree left;
  tree right;
 };

tree mainroot;

void postorder(tree t)
 {
  if(t)
   {
    postorder(t->left);
    postorder(t->right);
    cout << t->item;
   }
 }
```

5. [10 points]

True/False/Open

(a) **F** Every mathematically defined problem can be solved by computation, although it might not have been done yet.

(b) **O** "Many hands make light work." Any polynomial time problem can be worked in $O(\log^k n)$ time, for some k, by polynomially many processers working in parallel.

6. [10 points] The following code computes the product of its two parameters. Write the loop invariant of the loop.

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

$n * x = m * y + z$

7. Fill in the blanks. In questions where the answer is a number, you must give an exact answer.

   (a) A planar graph with 10 edges must have at least **6** vertices.

   (b) A connected graph with 12 edges must have at least **6** vertices..

   (c) A strongly connected digraph with 8 vertices must have at least **8** arcs..

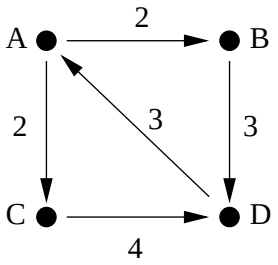
   (d) An acyclic digraph of 12 vertices must have **12** strong components.

   (e) **Dijkstra's** algorithm does not permit the weight of any arc to be negative.

8. [10 points] Write pseudocode for the Floyd-Warshall algorithm. Assume that the vertices are numbered from 1 to n, that W(i,j) is the weight of the arc from i to j, and that W(i,j) = ∞ if there is no arc from i to j. Be sure to compute back pointers.

```
for all i and all j
    V(i,j) = W(i,j)
    back(i,j) = i
for all i
    V(i,i) = 0
for all j
    for all i
        for all k
            temp = V(i,j) + V(j,k)
                if temp < V(i,k)
                    V(i,k) = temp
                    back(i,k) = back(j,k)
```

9. [20 points] (a) Finish the matrix W representing the weighted directed graph shown below. I have written the first two rows. (b) Use tropical matrix multiplication to compute W^3, which represents the graph of minimum path distances. Do not calculate backpointers.



$W =$

	A	B	C	D
A	0	2	2	∞
B	∞	0	∞	3
C	∞	∞	0	4
D	3	∞	∞	0

$W^2 =$

	A	B	C	D
A	0	2	2	∞
B	∞	0	∞	3
C	∞	∞	0	4
D	3	∞	∞	0

	A	B	C	D
A	0	2	2	∞
B	∞	0	∞	3
C	∞	∞	0	4
D	3	∞	∞	0

$=$

	A	B	C	D
A	0	2	2	5
B	6	0	∞	3
C	7	∞	0	4
D	3	5	5	0

$W^3 =$

	A	B	C	D
A	0	2	2	∞
B	∞	0	∞	3
C	∞	∞	0	4
D	3	∞	∞	0

	A	B	C	D
A	0	2	2	5
B	6	0	∞	3
C	7	∞	0	4
D	3	5	5	0

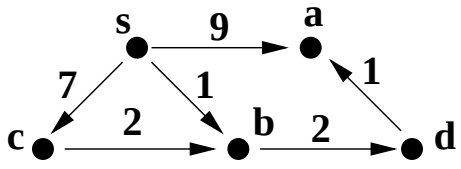
$=$

	A	B	C	D
A	0	2	2	5
B	6	0	8	3
C	7	9	0	4
D	3	5	5	0

10. [10 points] Write pseudocode for the Bellman-Ford algorithm. Be sure to include the shortcut. Assume that the vertices are numbered from 0 to $n-1$ with source vertex 0, and that the arcs are numbered from 0 to $m-1$, and that the j^{th} arc is $a_j = (s_j, t_j)$ for vertices s_j and t_j, and has weight w_j. Be sure to compute back pointers, and be sure to encode the shortcut.

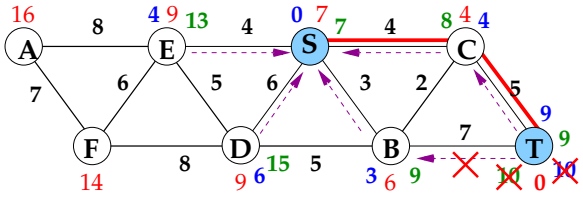
```
V(0) = 0
for all i from 1 to n − 1
    V(i) = ∞
altered = true
while(altered)
    altered = false
    For all j from 1 to m − 1
        temp = V(s_j) + w_j
        if temp < V(t_j)
            V(t_j) = temp
            back(t_j) = s_j
            altered = true
```

11. [20 points] Solve the single source minpath problem using Dijkstra's algorithm for the weighted graph shown, where S is the source vertex. Fill in the matrix at each step, as was done for the example shown in the handout shown in class. Each edge shown is actually two arcs, one in each direction.



	s	a	b	c	d
V	0	~~9~~ 4	1	7	3
back		~~s~~ d	s	s	b

12. [20 points] Walk through the A^* algorithm for the following weighted digraph, where S and T are the source and target vertices. You do not have to draw the graph after each step. Just write the values of f and g as well as the backpointers, crossing out (not erasing) incorrect temporary values and pointers, and indicate the least weight path.



13. [20 points] Compute the first phase of Johnson's algorithm using the weighted graph (a) below. Use (b) for your intermediate work, and show the adjusted arc weights in (c). Do not finish Johnson's algorithm. The weighted digraph in Figure (a) has negative weight arcs. Compute the adjusted weight for each arc as is need to execute Johnson's algorithm, but do not complete Johnson's algorithm. Write the adjusted weights in Figure (c), and use Figure (b) for your work.