This exam has 16 questions (including question 0) worth a total of 100 points. You have 180 minutes. This exam is preprocessed by a computer when grading, so please write darkly and write your answers inside the designated spaces.

Policies. The exam is closed book, except that you are allowed to use a one-page cheatsheet (8.5-by-11 paper, two sides, in your own handwriting). Electronic devices are prohibited.

Discussing this exam. Discussing the contents of this exam before solutions have been posted is a violation of the Honor Code.

This exam. Do not remove this exam from this room. In the space provided, write your name and NetID. Also, mark your exam room and the precept in which you are officially registered. Finally, write and sign the Honor Code pledge. You may fill in this information now.

Name:

NetID:

Course: COS 126 COS 226

Exam room: McCosh 50 Other

Precept: P01 P02 P04 P05 P07 P08 P09 P10

“I pledge my honor that I will not violate the Honor Code during this examination.”

Signature
0. Initialization. (1 point)
In the space provided on the front of the exam, write your name and NetID; mark your exam
room and the precept in which you are officially registered; write and sign the Honor Code
pledge.

1. Empirical running time. (6 points)
Suppose that you observe the following running times (in seconds) for a program on graphs
with \(V\) vertices and \(E\) edges.

<table>
<thead>
<tr>
<th>(E)</th>
<th>10,000</th>
<th>20,000</th>
<th>40,000</th>
<th>80,000</th>
<th>160,000</th>
<th>320,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>10,000</td>
<td>6.25</td>
<td>8.84</td>
<td>12.50</td>
<td>17.68</td>
<td>25.00</td>
<td>35.36</td>
</tr>
<tr>
<td>20,000</td>
<td>12.50</td>
<td>17.68</td>
<td>25.00</td>
<td>35.36</td>
<td>50.00</td>
<td>70.71</td>
</tr>
<tr>
<td>(V)</td>
<td>40,000</td>
<td>25.00</td>
<td>35.36</td>
<td>50.00</td>
<td>70.71</td>
<td>100.00</td>
</tr>
<tr>
<td>80,000</td>
<td>50.00</td>
<td>70.71</td>
<td>100.00</td>
<td>141.42</td>
<td>200.00</td>
<td>282.84</td>
</tr>
<tr>
<td>160,000</td>
<td>100.00</td>
<td>141.42</td>
<td>200.00</td>
<td>282.84</td>
<td>400.00</td>
<td>565.69</td>
</tr>
<tr>
<td>320,000</td>
<td>200.00</td>
<td>282.84</td>
<td>400.00</td>
<td>565.69</td>
<td>800.00</td>
<td>1131.37</td>
</tr>
</tbody>
</table>

- **running time of the program (in seconds)**

(a) Estimate the running time of the program (in seconds) for a graph with \(V = 640,000\)
vertices and \(E = 640,000\) edges.

\[ \text{seconds} \]

(b) Estimate the *order of growth* of the running time of the program as a function of both
\(V\) and \(E\).
2. Memory. (4 points)

Suppose that you implement a symbol table (containing string keys and integer values) using an $r$-way trie with following data type:

```java
public class RwayTrie {
    private final int r;
    private Node root;

    public RwayTrie(int r) {
        this.r = r;
        root = null;
    }

    private class Node {
        private final int value;
        private Node[] next;

        private Node(int value) {
            this.value = value;
            this.next = new Node[r];
        }
    }
}
```

Using the 64-bit memory cost model from lecture and the textbook, how much memory does each Node object use? Count all memory allocated when a Node object is constructed. Write your answer as a function of $r$.

$\text{bytes}$
3. **String sorts. (5 points)**

The column on the left contains the original input of 24 strings to be sorted; the column on the right contains the strings in sorted order; the other 5 columns contain the contents at some intermediate step during one of the 3 radix-sorting algorithms listed below. Match each algorithm by writing its letter in the box under the corresponding column.

*You may use each letter once, more than once, or not at all.*

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4170</td>
<td>1233</td>
<td>3963</td>
<td>4170</td>
<td>9601</td>
</tr>
<tr>
<td>1</td>
<td>9601</td>
<td>1866</td>
<td>2145</td>
<td>9601</td>
<td>5601</td>
</tr>
<tr>
<td>2</td>
<td>8287</td>
<td>1018</td>
<td>1018</td>
<td>5601</td>
<td>4514</td>
</tr>
<tr>
<td>3</td>
<td>6853</td>
<td>2119</td>
<td>2119</td>
<td>3052</td>
<td>1018</td>
</tr>
<tr>
<td>4</td>
<td>5185</td>
<td>2145</td>
<td>3923</td>
<td>9152</td>
<td>2119</td>
</tr>
<tr>
<td>5</td>
<td>5052</td>
<td>3923</td>
<td>1866</td>
<td>7722</td>
<td>7722</td>
</tr>
<tr>
<td>6</td>
<td>9152</td>
<td>3963</td>
<td>1233</td>
<td>6853</td>
<td>3923</td>
</tr>
<tr>
<td>7</td>
<td>3923</td>
<td>4170</td>
<td>4514</td>
<td>3923</td>
<td>4528</td>
</tr>
<tr>
<td>8</td>
<td>9388</td>
<td>4528</td>
<td>4170</td>
<td>1233</td>
<td>6728</td>
</tr>
<tr>
<td>9</td>
<td>1233</td>
<td>4514</td>
<td>4435</td>
<td>7453</td>
<td>1233</td>
</tr>
<tr>
<td>10</td>
<td>4528</td>
<td>4435</td>
<td>4528</td>
<td>3963</td>
<td>4435</td>
</tr>
<tr>
<td>11</td>
<td>8587</td>
<td>5185</td>
<td>7453</td>
<td>4514</td>
<td>2145</td>
</tr>
<tr>
<td>12</td>
<td>7453</td>
<td>5052</td>
<td>6728</td>
<td>5185</td>
<td>5052</td>
</tr>
<tr>
<td>13</td>
<td>6728</td>
<td>5601</td>
<td>8587</td>
<td>4435</td>
<td>9152</td>
</tr>
<tr>
<td>14</td>
<td>1866</td>
<td>6853</td>
<td>9388</td>
<td>2145</td>
<td>6853</td>
</tr>
<tr>
<td>15</td>
<td>2119</td>
<td>6728</td>
<td>9152</td>
<td>1866</td>
<td>7453</td>
</tr>
<tr>
<td>16</td>
<td>1018</td>
<td>7453</td>
<td>5052</td>
<td>7056</td>
<td>7056</td>
</tr>
<tr>
<td>17</td>
<td>4514</td>
<td>7056</td>
<td>5185</td>
<td>8287</td>
<td>3963</td>
</tr>
<tr>
<td>18</td>
<td>4435</td>
<td>7722</td>
<td>6853</td>
<td>8887</td>
<td>1866</td>
</tr>
<tr>
<td>19</td>
<td>2145</td>
<td>8287</td>
<td>8287</td>
<td>9388</td>
<td>4170</td>
</tr>
<tr>
<td>20</td>
<td>3963</td>
<td>8587</td>
<td>7056</td>
<td>4528</td>
<td>5185</td>
</tr>
<tr>
<td>21</td>
<td>7056</td>
<td>9601</td>
<td>5601</td>
<td>6723</td>
<td>8287</td>
</tr>
<tr>
<td>22</td>
<td>5601</td>
<td>9152</td>
<td>7722</td>
<td>1018</td>
<td>8587</td>
</tr>
<tr>
<td>23</td>
<td>7722</td>
<td>9388</td>
<td>9601</td>
<td>2119</td>
<td>9388</td>
</tr>
</tbody>
</table>

A. Original input  
B. LSD radix sort  
C. MSD radix sort  
D. 3-way radix quicksort (*no shuffle*)  
E. Sorted
4. Depth-first search. (6 points)

Run depth-first search on the following digraph, starting from vertex 0. Assume the adjacency lists are in sorted order: for example, when iterating over the edges leaving vertex 3, consider the edge $3 \rightarrow 2$ before either $3 \rightarrow 4$ or $3 \rightarrow 8$.

(a) List the 10 vertices in preorder.

(b) List the 10 vertices in postorder.

(c) The above digraph does not have a topological order. If, however, you delete one edge, it will have a topological order. Which edge?
5. Breadth-first search. (7 points)

Consider the following buggy implementation of breadth-first search in a digraph.

```java
private void bfs(Digraph G, int s) {
    marked = new boolean[G.V()];
    distTo = new int[G.V()];
    Queue<Integer> queue = new Queue<Integer>();
    queue.enqueue(s);
    while (!queue.isEmpty()) {
        int v = queue.dequeue();
        for (int w : G.adj(v)) {
            if (!marked[w]) {
                distTo[w] = distTo[v] + 1;
                queue.enqueue(w);
            }
        }
    }
}
```

(a) Suppose that you run the code fragment on the following DAG, starting from \( s = 0 \). Mark all statements below that are true.

- [ ] It terminates.
- [ ] Some vertices are added to the queue more than once.
- [ ] At some point, the queue contains multiple copies of the same vertex.
- [ ] Upon termination, \( \text{distTo}[1] \) is 1 (the length of the shortest path from 0 to 1).
- [ ] Upon termination, \( \text{distTo}[9] \) is 9 (the length of the longest path from 0 to 9).

(b) Annotate the code above to correct it.
6. Minimum spanning tree. (6 points)

Consider the following edge-weighted graph.

(a) List the weights of the MST edges in the order that Kruskal’s algorithm adds them to the MST.

____  ____  ____  ____  ____

(b) List the weights of the MST edges in the order that Prim’s algorithm adds them to the MST. Start Prim’s algorithm from vertex $s$.

____  ____  ____  ____  ____  ____
7. Knuth–Morris–Pratt substring search. (6 points)

Consider the Knuth–Morris–Pratt DFA for the following string of length 8 over the alphabet \{ A, B, C \}:

B B C B B B C A

(a) Complete the last three columns of this partially-completed DFA table.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) In which state is the DFA after consuming the following sequence of characters?
Mark the correct answer.

C B B B B C B C B B C B B C B B C B B C B B C B B C B
8. Java String library performance. (8 points)

For each of the String expressions at left, write the letter of the best-matching worst-case running time (as a function of $m$ and $n$) at right, where

- $s$ is a string of length $n$
- $t$ and regexp are strings of length $m$
- $m \leq n$

Assume the standard (Oracle or OpenJDK) Java 8 representation and implementation for the String data type. You may use each letter once, more than once, or not at all.

- $s.length() + t.length()$  
  A. $1$

- $s.charAt(n/2)$  
  B. $\log m$

- $s.substring(n/2, n)$  
  C. $\log n$

- $s.equals(t)$  
  D. $m$

- $s.indexOf(t)$  
  E. $n$

- $s.matches(regexp)$  
  F. $m^2$

- $s += t$  
  G. $mn$

- for (int $i = 0; i < s.length(); i++)$
  
  $t += s.charAt(i);$  
  
  H. $n^2$

  I. $2^n$

Consider compressing strings of length $6n$ that contains $n$ copies of $X X Y Y Z Z$ concatenated together. For example, here is the string corresponding to $n = 5$.

\[
X X Y Y Z Z X X Y Y Z Z X X Y Y Z Z X X Y Y Z Z X X Y Y Z Z
\]

For each transformation at left, determine the compression ratio (as a function of $n$) and write the letter of the best-matching term at right. As usual, assume the alphabet size $R = 256$.

*You may use each letter once, more than once, or not at all.*

A. \( \sim \frac{1}{2} \)

B. \( \sim \frac{7}{8} \)

C. \( \sim \frac{1}{2} \)

D. \( \sim \frac{3}{8} \)

E. \( \sim \frac{5}{24} \)

F. \( \sim \frac{1}{4} \)

G. \( \sim \frac{3}{16} \)

H. \( \sim \frac{1}{8} \)

I. \( \sim \frac{1}{16} \)

J. \( \sim \frac{5}{768} \)

K. \( \sim \frac{1}{256} \)
10. Why did we do that? (8 points)

For each pair of algorithms or data structures, identify a critical reason why we prefer the first to the second. Write the letter of the best-matching answer.

You may use each letter once, more than once, or not at all.

Use adjacency lists instead of an adjacency-matrix to represent a sparse undirected graph.  

A. Guarantees correctness.

B. Improves performance in practice.

C. None of the above.

Use union–find instead of depth-first search for cycle detection in Kruskal’s algorithm.

Relax the vertices in increasing order of distance from the source in Dijkstra’s algorithm instead of in reverse DFS postorder to compute shortest paths in digraphs with positive edge weights.

Use 3-way radix quicksort instead of mergesort to sort an array of strings.

Use Boyer–Moore instead of Knuth–Morris–Pratt for substring search.

Use depth-first search instead of breadth-first search to compute a topological order in a directed acyclic graph.

Use depth-first search instead of breadth-first search to determine all vertices reachable from a set of vertices in NFA simulation.

Use breadth-first search instead of depth-first search to find a shortest ancestral path in the WordNet assignment.
11. Shortest paths. (7 points)

Given a digraph $G$ with positive edge weights, complete the constructor below to compute the length of the shortest path from $s$ to each vertex. To do so, write the letter of one of the following code fragments in each provided space.

A. (int $i = 1; i < G.V(); i++$)  
B. (int $i = 1; i < G.E(); i++$)  
C. (int $v = 0; v < G.V(); v++$)  
D. (int $v = 0; v < G.E(); v++$)  
E. (DirectedEdge $e : G.adj(v)$)

You may use each letter once, more than once, or not at all. No other code is allowed.

```java
public BellmanFordSP(EdgeWeightedDigraph G, int s) {
    distTo = new double[G.V()];
    for (int v = 0; v < G.V(); v++)
        distTo[v] = _________ ;
    distTo[s] = _________ ;

    for _________ {
        for _________ {
            for _________ {
                int w = e.to();
                if ( _________ > _________ )
                    _________ = _________ ;
            }
        }
    }
}
```
12. **Ternary search tries. (5 points)**

Consider the following TST, where the integer values are shown next to the nodes of the corresponding string keys. Each node labeled with a ? contains some uppercase letter (possibly different for each node).

Which of the following string keys are (or could possibly be) in the TST? Mark all that apply.

<table>
<thead>
<tr>
<th>DATA</th>
<th>BRIE</th>
<th>DAD</th>
<th>DARK</th>
<th>DO</th>
<th>FUN</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ ]</td>
<td>[ ]</td>
<td>[ ]</td>
<td>[ ]</td>
<td>[ ]</td>
<td>[ ]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HUE</th>
<th>PRO</th>
<th>QUEUE</th>
<th>TRIE</th>
<th>TRUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ ]</td>
<td>[ ]</td>
<td>[ ]</td>
<td>[ ]</td>
<td>[ ]</td>
</tr>
</tbody>
</table>
13. Regular expressions. (6 points)

Consider the NFA that results from applying the RE-to-NFA construction algorithm from lecture and the textbook to the regular expression

\(( ( A \mid B \ C ) \ast B \ast )\)

The states and match transitions (solid lines) are shown below, but some of the \(\epsilon\)-transitions (dotted lines) are suppressed.

(a) Mark all edges in the \(\epsilon\)-transition digraph.

(b) Suppose that you want to construct an NFA for the regular expression

\(( ( A \mid B \ C ) ? B \ast )\)

where the operator ? means zero or one copy of the expression that precedes it. What minimal change(s) would you make (e.g., adding or removing \(\epsilon\)-transitions) to the NFA you defined in part (a)?
14. Prefix-free codes. (10 points)

(a) For a final exam question, an absentminded professor created a Huffman code for a set of 7 symbols. Unfortunately, she forgot to write down the codeword for one of the symbols.

<table>
<thead>
<tr>
<th>symbol</th>
<th>codeword</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>00</td>
</tr>
<tr>
<td>B</td>
<td>01100</td>
</tr>
<tr>
<td>E</td>
<td>10</td>
</tr>
<tr>
<td>P</td>
<td>0111</td>
</tr>
<tr>
<td>R</td>
<td>01101</td>
</tr>
<tr>
<td>S</td>
<td>?</td>
</tr>
<tr>
<td>T</td>
<td>11</td>
</tr>
</tbody>
</table>

Deduce the codeword associated with the symbol S.

(b) Given a Huffman code (or optimal prefix-free code) for a set of \( n \geq 3 \) symbols, with one codeword missing, design an algorithm to deduce the missing codeword. The input to the problem is an array of the \( n-1 \) known codewords.

Write your answer in the spaces provided on the next page.

Your answers to (b) and (c) will be graded for correctness, efficiency, and clarity. For full credit, your algorithm must take time linear in the input size (the total number of bits to represent the codewords) in the worst case.
Briefly describe your algorithm in the space below.

Draw a diagram of your data structure(s) for deducing the missing codeword when the known codewords are 00, 01100, 10, 0111, 01101, and 11, as in part (a).
(c) Now, suppose that there are two codewords missing. Design an algorithm to deduce the two missing codewords. Do not repeat details from part (b) if they are identical.
15. **Writing seminar assignment problem.** (10 points)

A prominent northeastern university assigns $n$ students to $m$ writing seminars. Each student ranks the writing seminars in order of preference (from favorite to least favorite). Each writing seminar has space for as many as $p$ students. Design an algorithm to determine whether it is possible to assign the students to the writing seminars so that each student gets one of their top two choices. To do so, model the problem as a *maximum flow problem*.

**An example.** Here is an example input with $n = 6$ students (Abigail, Bjarne, Čazir, DeAndre, Eun-jung, and Flor) and $m = 3$ writing seminars (*X-Ray Crystallography*, *Your Life in Numbers*, and *Zoom!*), where each seminar has space for $p = 2$ students.

<table>
<thead>
<tr>
<th></th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>B</td>
<td>Z</td>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>C</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>D</td>
<td>Y</td>
<td>Z</td>
<td>X</td>
</tr>
<tr>
<td>E</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>F</td>
<td>Y</td>
<td>X</td>
<td>Z</td>
</tr>
</tbody>
</table>

In this example, there is no assignment in which each student gets their first choice (because three students rank $X$ as their first choice). However, there is an assignment in which each student gets one of their top two choices:

<table>
<thead>
<tr>
<th>assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>A—X</td>
</tr>
<tr>
<td>B—Z</td>
</tr>
<tr>
<td>C—X</td>
</tr>
<tr>
<td>D—Z</td>
</tr>
<tr>
<td>E—Y</td>
</tr>
<tr>
<td>F—Y</td>
</tr>
</tbody>
</table>
(a) Draw the flow network that you would construct in order to solve the writing seminar assignment problem on the facing page (with 6 students and 3 writing seminars). Be sure to label the source and destination vertices and specify the edge capacities.

(b) After solving such a maximum flow problem, how would you determine whether there exists an assignment in which each student gets one of their top two choices?

(c) In the worst case, how many augmenting paths will the Ford–Fulkerson algorithm find (as a function of $m$ and $n$)? Assume $n \geq m$. Mark the best answer.

\[ m \quad n \quad m + n \quad n \log n \quad m \log m \quad mn \quad mn^2 \quad m^2 n \quad m^2 n^2 \]
(d) Let $k$ be an integer between 1 and $m$. Suppose that you want to know whether it is possible to assign the students to writing seminars so that each student gets one of their top $k$ choices (instead of top 2 choices). Briefly describe how you would modify your solution to (a).

(e) Design an efficient algorithm to find the smallest integer $k$ for which it is possible to assign the students to writing seminars so that each student gets one of their top $k$ choices. Your algorithm should be substantially faster in the worst case than repeatedly applying (d) to solve $m$ maximum flow problems (one for each possible value of $k$).