This test has 12 questions worth a total of 100 points. You have 180 minutes. The exam is closed book, except that you are allowed to use a one page cheatsheet (8.5-by-11, in your own handwriting). No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided. Write out and sign the Honor Code pledge before turning in the test.

“I pledge my honor that I have not violated the Honor Code during this examination.”
1. **Graph search. (10 points)**

Consider the following directed graph.

(a) Run *depth-first search*, starting at vertex A. Assume the adjacency lists are in lexicographic order, e.g., when exploring vertex E, consider E-D before E-G or E-H. Complete the list of vertices in *preorder* (the order they are first discovered by DFS).

A B C ___ ___ ___ ___ ___ ___

(b) Run *breadth-first search*, starting at vertex A. Assume the adjacency lists are in lexicographic order. Complete the list of vertices in the order in which they are enqueued.

A B D E ___ ___ ___ ___ ___ ___

(c) Identify one situation where you would need to use BFS instead of DFS.

(d) Identify one situation where you would need to use DFS instead of BFS.
2. Minimum spanning tree. (8 points)

Consider the following weighted graph.

(a) Complete the list of edges in the MST in the order that Kruskal’s algorithm includes them. For reference, the edge weights in ascending order are:

\[
\begin{align*}
4 & \quad 16 & \quad 18 & \quad 19 & \quad 20 & \quad 21 & \quad 22 & \quad 23 & \quad 25 & \quad 30 & \quad 33 & \quad 34 & \quad 35 & \quad 36 & \quad 42 & \quad 65 \\
B-D & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots
\end{align*}
\]

(b) Complete the list of edges in the MST in the order that Prim’s algorithm includes them. Start Prim’s algorithm from vertex A.

\[
\begin{align*}
A-H & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots
\end{align*}
\]
3. Minimum spanning tree. (8 points)
Suppose you know the MST of a weighted graph $G$. Now, a new edge $v-w$ of weight $c$ is inserted into $G$ to form a weighted graph $G'$. Design an $O(V)$ time algorithm to determine if the MST in $G$ is also an MST in $G'$. You may assume all edge weights are distinct.
Your answer will be graded for correctness, clarity, and conciseness.

(a) State the algorithm.

(b) Explain briefly why it takes $O(V)$ time.
4. Data compression. (4 points)

How many bits are in the Huffman encoding of the following message?
(Do not count the bits to encode the table.)

a b a a b a c a b a a b a c d a b a a b a c a b a a b a c d e

For reference, the frequency of each symbol is given in the table below.

<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></td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

5. Ternary search tries. (6 points)

Consider the following TST, where the black nodes correspond to strings in the TST.

(a) Which 7 strings (in alphabetical order) are in the TST?

(b) Draw the results of adding the following strings into the TST above:

`cgt aaca tt`
6. String searching. (8 points)

Complete the following DFA to match precisely those strings (over the two letter alphabet) that contain \texttt{bababba} as a substring. State 0 is the start state and state 7 is the accept state.

<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>2</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7. Algorithm matching. (10 points)

Match up each application with an algorithm or data structure that we used to solve it in this course. Use each answer exactly once.

\[\begin{align*}
&\underline{\text{T9 texting in a cell phone}} & \text{A. Trie} \\
&\underline{\text{1D range search}} & \text{B. Hashing} \\
&\underline{\text{2D range search}} & \text{C. 3-way radix quicksort} \\
&\underline{\text{Document similarity}} & \text{D. Binary search tree} \\
&\underline{\text{Traveling salesperson problem}} & \text{E. Kd tree} \\
&\underline{\text{Sudoku solver}} & \text{F. Depth-first search} \\
&\underline{\text{Arbitrage detection in currency exchange rates}} & \text{G. Breadth-first search} \\
&\underline{\text{Mark-sweep garbage collector}} & \text{H. Dijkstra’s algorithm} \\
&\underline{\text{Web crawler}} & \text{I. Topological sort} \\
&\underline{\text{Google maps}} & \text{J. Bellman-Ford} \\
&\underline{\text{PERT/CPM (Program Evaluation and Review Technique / Critical Path Method).}} & \text{K. Enumerate permutations} \\
&\underline{\text{Longest repeated substring}} & \text{L. Enumerate base-R integers} \\
\end{align*}\]
8. Regular expressions. (8 points)

Draw the NFA that results from the RE-to-NFA conversion algorithm described in lecture when applied to the regular expression \((a \mid (bc)^*)d^*\). Label the start state 0, the accept state 1, and remaining states in the order they are created by the RE-to-NFA algorithm.
9. Convex hull. (8 points)

Run the Graham scan algorithm to compute the convex hull of the 9 points below, using F as the base point, and continuing counterclockwise starting at G.

(a) List the points in the order that they are considered for insertion into the convex hull.

F G H I ___ ___ ___ ___ ___

(b) Give the points that appear on the trial hull (after each of the remaining iterations).

1. F -> G -> H

2. F -> G -> H -> I

3. 

4. 

5. 

6. 

7.
10. **4-sum. (12 points)**

Consider the 4-SUM problem: *Given N integers, do any 4 of them sum up to exactly 0?*

(a) Consider the following brute-force solution (we ignore integer overflow).

```java
public static fourSum(int[] a) {
    int N = a.length;
    for (int i = 0; i < N; i++)
        for (int j = i+1; j < N; j++)
            for (int k = j+1; k < N; k++)
                for (int l = k+1; l < N; l++)
                    if (a[i] + a[j] + a[k] + a[l] == 0) return true;
    return false;
}
```

What is the order of growth of the worst-case running time? Circle the best answer.


(b) Design an algorithm for 4-SUM that runs in $O(N^2)$ time and uses $O(N^2)$ memory.

Assume that you have access to a hashing-based symbol table that can `put()` and `get()` integer keys in constant time per operation.
11. **Reductions and shortest paths. (10 points)**

Given a digraph $G$, a distinguished vertex $s$, and nonnegative vertex weights, the *single-source vertex-weighted shortest path problem* is to find the shortest path from $s$ to each vertex, where the length of the path is the sum of the weights of the vertices on the path.

![Diagram](image)

**Figure 1:** The shortest path from $s$ to $5$ is $s \rightarrow 1 \rightarrow 4 \rightarrow 5$ and has length 22.

Give a linear-time reduction from the single-source *vertex-weighted* shortest path problem to the classic single-source *edge-weighted* shortest path problem. Demonstrate your reduction by drawing the corresponding digraph $G'$ along with its edge weights.
12. **Suffix sorting. (8 points)**

Your Harvard friend is trying to sort the suffixes of a string \( s \) consisting of \( N \) ASCII characters, none of which is ‘\0’. The code calls `RadixQuicksort3way.sort()`, which sorts an array of ‘\0’-terminated strings.

```java
// form the N suffixes, appending ‘\0’ to the end of each string
String[] suffixes = new String[N];
for (int i = 0; i < N; i++) {
    suffixes[i] = s.substring(i, N) + "\0"
}

// sort the N strings
RadixQuicksort3way.sort(suffixes);
```

Unfortunately, when your friend uses this code for large \( N \), it fails spectacularly, even for non-pathological inputs.

(a) Briefly explain the problem.

(b) Fix it so that it runs efficiently.