1. Fill in the blanks. [5 points each blank.]

(a) An acyclic graph of 6 vertices has no more than 5 edges. (Exact answer.)

(b) An acyclic digraph of 6 vertices has no more than 15 arcs. (Exact answer.)

(c) A planar graph with 100 vertices has no more than 294 edges. (Exact answer.)

(d) An AVL tree of height 4 has at least 12 nodes. (Exact answer.)

2. Let G be a directed graph with n vertices and m arcs. Fill in the blanks, using asymptotic notation. [5 points each blank.]

(a) The time complexity of the Floyd-Warshall algorithm is $O(n^3)$

(b) The time complexity of Dijkstra’s algorithm is $O(m \log n)$, but there can be no negative arcs.

(c) The time complexity of Johnson’s algorithm is $O(nm \log n)$.

3. What is the asymptotic time complexity, in terms of $n$, of each of the following code fragments? (10 points each)

(a) 
```java
for(int i = n; i > 0; i--)
    for(int j = i; j < n; j = 2*j)
```  
$\Theta(n)$

(b) 
```java
for(int i = n; i > 0; i=i/2)
    for(int j = 1; j < i; j++)
```  
$\Theta(n)$

4. Solve the recurrences. Give asymptotic answers in terms of $n$. Use $O$, $\Omega$, or $\Theta$, whichever is most appropriate. [10 points each.]

(a) $F(n) \leq 2F(n/2) + 1$  
$F(n) = O(n)$

(b) $G(n) \geq 9G(n/3) + n$  
$G(n) = \Omega(n^2)$

(c) $H(n) = 2H(\sqrt{n}) + \log n$  
$H(n) = \Theta(\log n \log \log n)$

(d) $K(n) = 4K(n/4) + 3K(n/2) + n$  
$K(n) = \Theta(n^2)$
5. The following C++ code computes a function f:

```cpp
int f(int n) // input condition: n >= 0
{
    if(n == 0) return 0;
    else return f(n/2)+f(n/3)+f(n/6) + n;
}
```

(a) [10 points] What is the time complexity of the recursive code given above?

Don’t confuse the time complexity of the execution of the code with the complexity of the function f itself. The code gives the recurrence for f, and the solution is f(n) = Θ(n log n). But the recurrence which defines the time is

\[ T(n) = T(n/2) + T(n/3) + T(n/6) + 1, \]

and the solution is \( T(n) = \Theta(n) \).

(b) [10 points] What is the time complexity of the dynamic programming computation of f(n)?

The value of f(n) is computed dynamically by the following code.

```cpp
cin >> n;
int f[0] = 0;
for(int i = 1; i <= n; i++)
    f[i] = f[i/2] + f[i/3] + f[i/6];
cout << f[n];
```

The time complexity of this code is \( \Theta(n) \).

(c) [20 points] Write pseudocode for a computation of f(n) using memoization. (I won’t ask you for the time complexity. The number of memos stored is \( O(\log^2 n) \). If you count the time it takes to store and fetch memos, the time complexity is \( O(\log^2 n \log \log n) \))

```cpp
int f(int n)
{
    if(n == 0) return 0;
    else if there is a memo (n,m) return m;
    else
    {
        int rslt = f(n/2)+f(n/3)+f(n/6)+n;
        store the memo (n,rslt);
        return rslt;
    }
}
```

6. [20 points] What is the Hamming distance between 01101001 and 00101100?

The Hamming distance between two binary strings of the same length is simply the number of places at which the two strings differ. These two strings differ at 3 places, so the Hamming distance is 3.
7. [20 points] What is the Levenshtein distance between \textit{abcdefg} and \textit{acdgeb}? Show the matrix.

\[
\begin{array}{cccccccc}
& a & b & c & d & e & f & g \\
0 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
a & 1 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
c & 2 & 1 & 1 & 1 & 2 & 3 & 4 & 5 \\
d & 3 & 2 & 2 & 2 & 1 & 2 & 3 & 4 \\
g & 4 & 3 & 3 & 3 & 2 & 2 & 3 & 3 \\
e & 5 & 4 & 4 & 4 & 3 & 2 & 3 & 3 \\
f & 6 & 5 & 5 & 5 & 4 & 3 & 2 & 3 \\
b & 7 & 6 & 6 & 6 & 5 & 4 & 3 & 3 \\
\end{array}
\]

The Levenshtein distance is 3.

8. [20 points] Walk through Dijkstra’s Algorithm for the following directed graph, where the start vertex is \textit{s}.

9. [20 points] \textbf{Error was here}. Find the longest common subsequence of the sequences \textit{X} = \{1, 5, 9, 3, 6, 0, 4, 2, 7, 8\} and \textit{Y} = \{529403821\}. Show the matrix.

\[
\begin{array}{cccccccc}
& 1 & 5 & 9 & 3 & 6 & 0 & 4 & 2 & 7 & 8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
5 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
2 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 \\
9 & 0 & 0 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\
4 & 0 & 0 & 1 & 2 & 2 & 2 & 2 & 3 & 3 & 3 \\
0 & 0 & 0 & 1 & 2 & 2 & 2 & 3 & 3 & 3 & 3 \\
3 & 0 & 0 & 1 & 2 & 3 & 3 & 3 & 3 & 3 & 3 \\
8 & 0 & 0 & 1 & 2 & 3 & 3 & 3 & 3 & 3 & 4 \\
2 & 0 & 0 & 1 & 2 & 3 & 3 & 3 & 4 & 4 & 4 \\
1 & 0 & 1 & 1 & 2 & 3 & 3 & 3 & 4 & 4 & 4 \\
\end{array}
\]

There are two choices of longest common subsequence, each of length 4. They are 5942 and 5948.
10. [20 points] Walk partway through Johnson’s Algorithm for the all-pairs shortest path problem on the weighted digraph shown below.

(a) Create a new graph by adding one node and $n$ arcs.

(b) Compute $h(v)$, the solution to the Bellman-Ford algorithm on each vertex. Note that $h(v) \leq 0$ for each vertex $v$. Indicate the value of $h(v)$ for each $v$.

(c) Adjust the weights. No edge will have a negative weight. Indicate those adjusted weights.

The next step is to use Dijkstra’s algorithm to solve the single source shortest path problem $n$ times, letting each vertex in turn be the source; but we will skip this step and all remaining steps.

The red figures are the values of $h$. The blue figures are the adjusted arc weights.
11. Each island in an ocean has some amount of treasure. A pirate ship is located at one island, and needs to reach its home island, picking up the treasure from each island it visits. However, due to prevailing winds, the ship can only sail South or East. The pirate captain wants to use Dynamic programming to chart the course, so as to maximize the treasure collected.

(a) [10 points] What are the subproblems?

There is one subproblem for each island. The value of this subproblem is the maximum treasure that can be collected for any course ending at this island.

(b) [10 points] What would be the arcs of the directed graph?

If $x$ and $y$ are island, there is an arc from the subproblem for $x$ to the subproblem for $y$ if and only if $y$ is either South, East, or Southeast of $x$ at any angle.

12. [20 points] Walk through the A* Algorithm for the weighted graph shown below. The heuristic $h(x)$ is indicated by a red numeral at each vertex.

In the figures below, the blue figures are the values of $f(x)$, the current shortest distance from S to $x$, and the green figures show the values of $g(x) = h(x) + f(x)$. Vertices are callee “Open” if they are in the minqueue, closed if they are fully processed, implying that their values of $f$ are permanent. At each step, that open vertex with the smallest value of $g$ is closed, and its neighbors which have not been in the minqueue so far become open. Backpointers are purple arrows.
Eventually, T becomes the open vertex with the smallest value of \( g \). Following the backpointers from T gives us the shortest path.

13. [20 points] You are given a sequence of \( n \) numbers, \( x_1, \ldots, x_n \). A range query is a query \( Q(i, j) \) for \( i \leq j \), and must return the sum \( x_i + \cdots + x_j \). (Alternatively, it returns the maximum or minimum of the set \{\( x_i, \ldots, x_j \} \).) If the sequence is stored in an array, a query \( Q(i, j) \) takes \( O(n) \) time to answer. Suppose we anticipate \( O(n) \) queries. If we store the matrix of values of all \( Q(i, j) \), Each query can be answered in \( O(1) \) time, but it takes \( O(n^2) \) time to initialize that matrix and \( O(n^2) \) space to store it.

**There will be no question on the third test relating to this problem.**

You’ve encountered information theory before, although you may not have heard that name. The proof that no sorting algorithm on \( n \) items can use fewer than \( \log_2(n!) \) comparisons in the worst case uses information theory.

If you have two way decisions, and you have \( t \) remaining comparisons, the number of possible outcomes cannot exceed \( 2^t \). Similarly (as in the balance scale problems below) if you have \( t \) remaining comparisons, the number of possible outcomes cannot exceed \( 3^t \).

**There will be no question on the third test relating to this material.**