1. Fill in the blanks. No shortest path algorithm works for a weighted directed graph which has a negative cycle.

2. Use Huffman’s algorithm to find an optimal prefix code for the alphabet \{a, b, c, d, e, f\} where frequency of each symbol is given in the following array.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>6</td>
</tr>
<tr>
<td>b</td>
<td>4</td>
</tr>
<tr>
<td>c</td>
<td>2</td>
</tr>
<tr>
<td>d</td>
<td>5</td>
</tr>
<tr>
<td>e</td>
<td>2</td>
</tr>
<tr>
<td>f</td>
<td>1</td>
</tr>
</tbody>
</table>

It is not necessary to sort the symbols by frequency, but you must maintain a data structure that enables finding the two packages of smallest weight quickly. It is not necessary to sort the symbols by frequency, but you must maintain a data structure that enables finding the two packages of smallest weight quickly. (A package is a subtree that does not yet have a parent.) Initially sorting the symbols, then placing the packages in an array in the order they are formed, enables the remainder of the algorithm to run in linear (that is, \(O(n)\)) time.

3. What is the asymptotic time complexity of the function \(\text{george}(n)\) given below? Hint: The answer is one of the following: \(\Theta(\log n), \Theta(n), \Theta(n \log n), \Theta(n^2)\).

```cpp
void george(int n)
{
    for(int i = 0; i < n; i++)
        cout << "Hello world" << end;
    if(n > 1)
    {
        george(n/2);
        george(n/2);
    }
}
```

Answer: \(\Theta(n \log n)\), just like mergesort.

4. Write pseudocode for the Floyd-Warshall algorithm on a weighted directed graph with \(n\) vertices. Let the vertices be \(x[1], \ldots, x[n]\). Let \(w[i, j]\) be the given weight of the arc from \(x[i]\) to \(x[j]\), which is \(\infty\) if there is no such arc. The algorithm should output arrays \(v\) and \(b\), where \(v[i, j]\) is the minimum weight of any path from \(x[i]\) to \(x[j]\), and \(b[i, j]\) is the backpointer of that path.

```cpp
for all i and j from 1 to n
    v[i, j] = w[i, j]
    b[i, j] = i
```

for all i from 1 to n
  v[i,i] = 0;
for all j from 1 to n
  for all i from 1 to n
    for all k from 1 to n
      if(v[i,j] + v[j,k] < v[i,k])
        v[i,k] = v[i,j] + v[j,k]
        back[i,k] = back[j,k]

The time complexity of the Floyd–Warshall algorithm is $\Theta(n^3)$.

5. Write pseudocode for the Bellman-Ford algorithm on a weighted directed graph with $n$ vertices and $m$ edges. Let the vertices be $x[1], \ldots, x[n]$, let $x[1]$ be the source vertex, and let the arcs (directed edges) be $e[1], \ldots, e[m]$, where $e[k]$ is an arc from a vertex $s[k]$ to a vertex $t[k]$, and $e[k]$ has weight $w[k]$. The output consists of values $v[1], \ldots, v[n]$, where $v[i]$ is the least weight of any path from $x[1]$ to $x[i]$, as well as backpointers $b[2], \ldots, b[n]$. Your code should include the shortcut which terminates the execution when the correct values of the $v[i]$ have been computed.

for all i from 2 to n
  v[i] = infinity
v[1] = 0;
finished = false;
while(not finished)
  finished = true;
  for all k from 1 to m // iterate through all arcs
    s = s[k]
    t = t[k]
    w = w[k]
    if(v[s] + w[s,t] < v[t])
      v[t] = v[s] + w[s,t] // update the shortest path from 1 to t by
        // concatenating the shortest path from 1 to s
        // with the arc from s to t
      finished = false; // a change has been made, so we may not be done.
      // if finished is still true, we’re done.

6. Find a dynamic programming algorithm for the following coin-row problem. There is a row of $n$ coins, each of which has some value, and the algorithm needs to find a set of coins of maximum total value, such that there are no three consecutive coins in the set.

Let $X[i]$ be the value of the $i$th coin, and let $M[i]$ be the maximum value if only the first $i$ coins are available. The following program computes $M[i]$ for $i = 0, \ldots, n$.

$M[0] = 0$
$M[1] = X[1]$
$M[i] = \max \{ M[i-1], X[i] + M[i-2], X[i] + X[i-1] + M[i-3] \}$ for $i \geq 3$

The best set of coins has value $M[n]$
7. Work problems 1 and 5 of the complex number assignment.

1. Perform the indicated operations. All answers must be in the \(a + bi\) form and reduced to the lowest terms.

(a) \((2 + 3i) - (1 - 2i)\)
Ans: \(1 + i\)

(b) \((2i) - (4 + 3i)\)
Ans: \(-4 - i\)

(c) \(\frac{4 + 3i}{1 + 2i}\)
Ans: Multiply both numerator and denominator by the conjugate of the denominator.

\[
\frac{4 + 3i}{1 + 2i} = \left(\frac{4 + 3i}{1 + 2i}\right)(\frac{1 - 2i}{1 - 2i}) = \frac{10 - 5i}{5} = 2 - i
\]

5. Find both square roots of \(1 + i\).

\(a = 1\) and \(b = 1\). By the formula given in the handout

\[
\sqrt{1+i} = \pm \left(\sqrt{\frac{\sqrt{a^2 + b^2} + a}{2}} + \sqrt{\frac{\sqrt{a^2 + b^2} - a}{2}}i\right)
\]

\[
= \pm \left(\sqrt{\frac{\sqrt{2} + 1}{2}} + \sqrt{\frac{\sqrt{2} - 1}{2}}i\right)
\]

---

*1Strictly speaking, \(\frac{1 + i}{2}\) is not in the \(a + bi\) form, but the equivalent expression \(\frac{1}{2} + \frac{1}{2}i\) is. However, I will allow a complex number to be written as a fraction where the numerator is complex and the denominator is positive real. Thus, you can write \(\frac{1 + i}{2}\), but not \(\frac{1 + i}{2}\) or \(\frac{1 + i}{2i}\). Those should be written \(\frac{-1 + i}{2}\) and \(\frac{-1 - i}{2}\), respectively.*