1. Derive a greedy strategy solving the following problem: We have a company with $n$ workers. Worker $w_i$ works a shift $(s_i, f_i)$, where $s_i$ is that worker’s start time and $f_i$ the finish time. (We will also refer to a shift as an interval, and you can assume $0 \leq s_i < f_i$ are integers.)

We want to form a small committee $C \subseteq \{w_1, \ldots, w_n\}$ which is complete, in other words for every worker not on the committee, that worker’s shift overlaps (at least partially) with the shift of some other worker who is on the committee. Or formally, for every worker $w_i$ there exists a worker $w_c \in C$ such that the shift of $w_i$ overlaps with the shift of $w_c$ (the intervals $(s_i, f_i)$ and $(s_c, f_c)$ must non-trivially intersect).

The problem here is to find the smallest possible complete set $C$ of workers. A greedy strategy will work. Here are two possible (different) strategies. One works and one does not.

- choose the worker whose interval intersects the the largest number other intervals, select that worker to add to $C$, and eliminate all intersecting intervals
- look at the earliest finishing uncovered interval $w$, and from among all intervals that intersect $w$ add to the committee $C$ the worker whose interval finishes the latest

(a) Choose one of the greedy choices above and prove that it yields the smallest possible $C$. (write your choice and justification on the next page)

(b) In a few sentences, describe an algorithm that implements your greedy choice and give its run-time bound
(place the answer to part (a) here)
2. The *longest common subsequence* problem is to find the longest (non-contiguous) sequence of characters shared by two strings $X = x_1x_2\ldots x_n$ and $Y = y_1y_2\ldots y_m$. For example, if $X = CAB$ and $Y = BACB$, the answer is 2 (they both share length 2 subsequences AB and CB). Here, we define a sub-problem $C(i, j)$ to be the longest subsequence of $x_1x_2\ldots x_i$ and $y_1y_2\ldots y_j$. The recurrence we have seen for $C$ is

$$
C(i, j) = \begin{cases} 
0 & \text{if } i = 0 \text{ or } j = 0 \\
1 + C(i - 1, j - 1) & \text{if } i > 0 \text{ and } j > 0 \text{ and } x_i = y_j \\
\max[C(i - 1, j), C(i, j - 1)] & \text{if } i > 0 \text{ and } j > 0 \text{ and } x_i \neq y_j 
\end{cases}
$$

Use the recurrence to fill in a table for $C$ for the strings $X = poppo$ and $Y = ppoo$. You may use (or not) the sketch below.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>p</th>
<th>o</th>
<th>o</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>o</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>o</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3. The following flow graph $G$ has some flow assignments already made. Use the Ford-Fulkerson method to continue the process of deriving the maximum flow from $s$ to $t$. Show the residual graph $G_f$, find an augmenting path in $G_f$, and then update the flow in $G$. One step should be enough to find the max-flow.
4. In class we started the process of devising a dynamic programming solution to the following problem from the DPV text, here slightly modified: Yuckdonalds is considering opening a series of restaurants along Quaint Valley Highway (QVH). The \( n \) possible locations are along a straight line, and the distances of these locations from the start of QVH are, in miles and in increasing order, \( m_1, m_2, \ldots, m_n \). The constraints are as follows:

- At each location, Yuckdonalds may open at most one restaurant. The expected profit from opening a restaurant at location \( i \) is \( p_i \), where \( p_i > 0 \) and \( i = 1, 2, \ldots, n \).
- Any two restaurants should be at least \( k \) miles apart, where \( k \) is a positive integer.
- We must place a restaurant at the last location \( m_n \) (this is the small modification)

Give an efficient algorithm to compute the maximum expected total profit subject to the given constraints.

*comment:* We add a restaurant location at milepost \( m_0 = -\infty \) with profit \( p_0 = 0 \).

*subproblem:* \( RP(i) \) is the maximum profit available by placing restaurants at locations chosen from mileposts \( m_0, m_1, \ldots, m_i \) subject to the restrictions

- each restaurant is at least \( k \) miles from the nearest neighbor
- a restaurant is placed at location \( i \)

*recurrence:*

\[
RP(i) = \begin{cases} 
0 & \text{if } i = 0 \\
p_i + \max \{ RP(j) \mid 0 \leq j < i \text{ and } m_i - m_j \geq k \} & \text{otherwise.}
\end{cases}
\]

You are to write code that will initialize and fill out an array \( RP \) using the recurrence. You may do so in either a bottom-up (iterative) or top-down (memoized) manner.

**PART I:** The pseudo-code is **ITERATIVE** or **MEMOIZED** (circle one)

**PART II:** The time bound of the pseudo-code is:

*continue on next page*
PART III: Write pseudo-code here.
5. Use Huffman’s algorithm to derive a prefix (-free) code for the given characters and frequencies.

<table>
<thead>
<tr>
<th>char</th>
<th>m</th>
<th>n</th>
<th>o</th>
<th>p</th>
<th>q</th>
<th>r</th>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>freq</td>
<td>2</td>
<td>3</td>
<td>8</td>
<td>21</td>
<td>55</td>
<td>144</td>
<td>377</td>
</tr>
</tbody>
</table>

Show your code tree and write out the derived code for each character.
6. (exercise 6.6, DPV) We define a multiplication operation on three symbols $a, b, c$, where the operation is described by a given table such as the one below: here $ab = b$, $ca = a$, and so on. The goal is to determine whether a string of symbols, $s_1 s_2 \cdots s_n$ ($s_i \in \{a, b, c\}$) can be parenthesized in such a way so that their multiplication together equals $a$. Note that the operation is neither associative nor commutative (that is, it is possible that $ab \neq ba$ and $(ab)c \neq a(bc)$).

Table 1: multiplication table for $a, b, c$

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>b</td>
<td>c</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>c</td>
<td>a</td>
<td>c</td>
<td>c</td>
</tr>
</tbody>
</table>

For example, if the input string is $bbbac$ then the answer should be true since $((bb)(ba))c = a$.

Define a subproblem $P(i, j, t)$, where $1 \leq i \leq j \leq n$ and $t \in \{a, b, c\}$. $P(i, j, t)$ will be true if it is possible to parenthesize $s_is_{i+1} \cdots s_j$ in such a way that the multiplication in that order equals the symbol $t$, and false otherwise. Give a recurrence relation for $P(i, j, t)$ (no code!).

7. Which of the following does NP stand for?

(a) Non-deterministically Possible
(b) Not Possible
(c) Not Provable
(d) Not Political
(e) Non-exponential Polynomial
(f) Now Polynomial
(g) Non-deterministic Polynomial
(h) Not Polynomial
(i) Not enough Provolone
(j) Never Polynomial
(k) Non-definable Polynomial
(l) Non-deterministic Provable

8. Answer the following questions about $P$ and $NP$:

(a) Claim: Every problem that can be solved in polynomial time non-deterministically can be solved by a polynomial time deterministic algorithm. True or False or Open?

(b) Claim: If a single NP-complete problem is shown to have a polynomial time (deterministic) algorithm, then it can be concluded that $P = NP$. True or False or Open?

points [11, 8, 10, 11, 8, 11, 2, 4] = 65 total