1. A server has n customers waiting to be served. The service time required by each customer is known in advance: it is $t_i$ minutes for customer $i$. So if, for example, the customers are served in order of increasing $i$, then the $i$th customer has to wait $\sum_{j=1}^{i} t_j$ minutes. We wish to minimize the total waiting time

$$T = \sum_{i=1}^{n} \text{(time spent waiting by customer } i).$$

(a) Give an efficient (greedy!) algorithm for computing the optimal order in which to process the customers.

(b) Describe the greedy choice your algorithm makes and argue that it is correct.
2. Illustrate the following sequence of disjoint set operations, using union-by-rank and path compression. First perform a **MAKESET** on each of 1, 2, \ldots, 12. Then,

- **UNION**(1, 2), **UNION**(3, 4), **UNION**(5, 6), **UNION**(7, 8), **UNION**(9, 10), **UNION**(11, 12)
- **UNION**(2, 4), **UNION**(6, 8), **UNION**(10, 12)
- **UNION**(8, 12)
- **UNION**(1, 9)
- **FINDSET**(5)
3. Imagine we are on a tour of some popular tourist spots and we wish to visit each one of them. They are all along a single road at mileposts $m_0, m_1, m_2, \ldots, m_n$, and we will stop at each one in that order. The choice we have to make is which taxi company to use to take us from one location to the next.

The choices are taxi company S (slow) and company T (tedious). Company S charges $s_i$ dollars per mile to travel from location $i$ to $i + 1$, so the charge for that segment is $(m_{i+1} - m_i) \cdot s_i$ dollars. Company T charges a flat rate of $t$ dollars per segment but if chosen they must be used for three consecutive segments. For example, we could start with company T at location $i$, have them take us to $i + 1$, $i + 2$, and finally finish with them at location $i + 3$ for a total of $3t$ dollars (of course, we could decide to use them for the next three segments). Our goal is to find the cheapest cost for all segments taking us from location 0 to location $n$.

For example, suppose the mileposts are at locations (0, 25, 50, 100, 150, 200), the costs for company S are (2, 3, 2, 1, 2), and the flat rate for company T is 40. If the travel plan is (S, T, T, T, S), the travel cost is $25 \cdot 2 + 40 + 40 + 40 + 50 \cdot 2 = 270$ dollars. However the plan (S, S, T, T, T) incurs a cost of $25 \cdot 2 + 25 \cdot 3 + 40 + 40 + 40 = 245$ dollars.

Define the subproblem $C(i)$ to be the minimum cost of a plan that starts at location 0 and ends at location $i$, and so that either (a) company S brought us from $m_{i-1}$ to $m_i$ (at a cost of $s_{i-1}$ per mile, or (b) company T was used on the last three segments (from $m_{i-3}$ to $m_{i-2}$ to $m_{i-1}$ to $m_i$). A recurrence for $C$ is given by

$$C(i) = \begin{cases} 
0 & \text{if } i = 0 \\
(m_i - m_{i-1}) \cdot s_{i-1} + C(i-1) & \text{if } 0 < i \leq 2 \ (\text{can only use company S}) \\
\min[(m_i - m_{i-1}) \cdot s_{i-1} + C(i-1), 3 \cdot t + C(i-3)] & \text{if } i \geq 3 \ (\text{min of choices (a) and (b) above})
\end{cases}$$

The minimum cost to go from location 0 (at $m_0$) to location $n$ (at $m_n$) You are to write pseudo-code which will fill up an array $C$ in either a bottom-up (iterative) manner or in a top-down (memoized) manner (you may use a hash table in the memoized case if you wish).

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

**PART II**: The time bound of my pseudo-code is:
PART III: Write your pseudo-code here.
4. Use Huffman’s algorithm to derive a code for the given characters and frequencies.

<table>
<thead>
<tr>
<th>char</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>freq</td>
<td>12</td>
<td>6</td>
<td>12</td>
<td>3</td>
<td>24</td>
<td>3</td>
<td>18</td>
<td>6</td>
</tr>
</tbody>
</table>

Show your code tree and write out the derived code for each character.
5. Here we are given an \( n \times n \) grid \( T \) and will follow a path from the upper-left corner (entry \((1,1)\)) to the bottom-right corner (entry \((n,n)\)) by some sequence of R ("right") and D ("down") moves. Each time we pass through a location, we collect the number of dollars given in that entry. We want to \textbf{maximize} the amount of money we collect. For example, let \( T \) be

\[
\begin{array}{cccc}
3 & 7 & 4 & 11 \\
13 & 2 & 5 & 4 \\
7 & 8 & 9 & 10 \\
17 & 4 & 13 & 2 \\
\end{array}
\]

For example, if the path from the upper left to lower right is (D, R, R, D, D, R), then we collect, in order, \( T(1,1) = 3 \), \( T(2,1) = 13 \), \( T(2,2) = 2 \), \( T(2,3) = 5 \), \( T(3,3) = 9 \), \( T(4,3) = 13 \), and \( T(4,4) = 2 \), for a total of \( 3 + 13 + 2 + 5 + 9 + 13 + 2 = 47 \) dollars. On the other hand, the path (D, D, D, R, R, R) has a payoff of \( 3 + 13 + 7 + 17 + 4 + 13 + 2 = 59 \) dollars.

Define \( MTP[i,j] \) ("max T-path") as the maximum sum of the entries of a given \( n \times n \) grid \( T \) considered over all paths that start at \((1,1)\) and end at \((i,j)\) through a series of R and D moves (in any order, as long as the moves stay in the grid \( T \)). Here \( 1 \leq i \leq n \) and \( 1 \leq j \leq n \), and the desired maximum value is \( T(n,n) \).

Give a recurrence for the function \( MTP \). Be sure to include base and boundary cases.
6. Use the Ford-Fulkerson method to find the maximum flow from $S$ to $T$ in the following graph:
7. Which of the following does NP stand for? (This NP being an acronym for the computer-science related complexity class.)

(a) Need more Potato
(b) Non-deterministically Possible
(c) Not Possible
(d) Not Provable
(e) Non-exponential Polynomial
(f) Now Polynomial
(g) Non-deterministic Polynomial
(h) Not Polynomial
(i) No Provolone
(j) Never Polynomial
(k) pass No Pass
(l) Non-deterministic Provable
(m) None of the Previous

8. Choose True or False or Open for each of the claims about NP:

(a) 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?

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

points $[10, 8, 14, 8, 10, 10, 2, 4] = 66$ total