1. In an union-find problem, we start with \( n \) disjoint sets each containing a single element. If all the union operations were to happen before all the \( m \) find operations, can you describe an algorithm that works in \( O(m + n) \) time? Note that the number of union operations may be less than \( n - 1 \), i.e., you may have multiple disjoint sets. You can use path compression heuristic but don’t make any references to the \( O((n + m) \log^* n) \) algorithm. (10 marks)

Unions cost \( O(1) \) per operation, so the total cost of unions is at most \( O(n) \). We will use the path compression heuristic. Any element that is visited once during the traversal, becomes a direct descendent of the root. Since there are no further unions, the element continues to be a child of the root. The find operation can be charged to the elements that are visited. Each element can be associated with a counter. This counter is incremented either when it is visited because of path compression or because it is actually the element on which find is applied. The maximum number of charges of the first kind is 1, so the total charges for all elements is \( \sum_x (1 + F(x)) = n + \sum_x F(x) = n + m \) where \( F(x) \) is the number of find operations for element \( x \).
2. In an e-auction there are a set of $m$ articles for which $n$ buyers have made some bids (a positive number). An article can be bought by at most one buyer (among the ones who have bid for that article). The company wants to assign the articles to buyers in such a way that the total sum of bids is maximized. A buyer can buy multiple articles.

For example consider articles A and B and buyers 1, 2, 3. The bids are as follows

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>5</td>
<td>no bid</td>
<td>7</td>
</tr>
<tr>
<td>B</td>
<td>8</td>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>

Then assigning A to 3 and B to 3 will maximise the sum of bids. (3+2+5+5 marks)

(i) Describe a graph based representation for this problem.

Consider a directed graph with two sets of vertices A and B where $|A| = m$ and $|B| = n$. Draw a directed edge from a buyer $x \in B$ to an article $y \in A$ having weight $w$ equal to the bid of $x$ for $y$ (no edge if there is no bid). The problem is to find a set of edges having maximum weight such that at most one directed edge is incident on any article.

(ii) Define independent subsets and show that it is a subset system.

A set of directed edges is independent if no two edges are incident on a vertex in $A$. Note that no such restriction holds for $B$. Clearly any subset of such a set of edges is also independent.

(iii) Is this a matroid? If so, prove it, else, provide a counter-example.

We will show that the exchange property holds. Consider two sets of edges $E_1$ and $E_2$ such that $|E_1| = |E_2| + 1$. Since there can be only one edge incident on a vertex of $A$, the number of induced vertices in $A$ must be $|E_1|$ and $|E_2|$ respectively. So there is clearly a vertex $x' \in E_1 - E_2$ and the corresponding edge can be added to $E_2$ without affecting independence.

(iv) Can you get an optimal solution by assigning the article to the highest bidder? If so, is it better than the generic-greedy algorithm?

Yes, it can be done. First this solution, say $S$ preserves independence. Suppose there is a better optimal solution $O$, which must be different from the solution $S$ that we have obtained by picking the best bid for every article. Then there exists at least one edge is $(a, b) \in O - S$. Then we can exchange $(a, b)$ with the the best bid for $a$ without affecting independene and improve the solution $S$ which is a contradiction.
3. A taxi-driver has to decide about a schedule to maximize his profit based on an estimated profit for each day. Due to some constraint, he cannot go out on consecutive days. For example, over a period of 5 days, if his estimated profits are 30, 50, 40, 20, 60, then by going out on 1st, 3rd and 5th days, he can make a profit of 30+40+60 =130. Alternately, by going out on 2nd and 5th days, he can earn 110. \(3+12\) marks

(a) Give an example (estimated profit sequence) to show that by choosing alternate days (there are two such schedules) he won’t maximize his profit.

Consider 50, 20, 10, 50
Choosing 1st and 3rd yields 60 and choosing 2nd and 4th yields 70. Choosing 1st and 4th gives 100 which is better than the two alternations.

(b) Design an efficient algorithm based on dynamic programming to solve this problem for an estimated profit sequence over \(n\) days.

(c) Design an efficient algorithm based on dynamic programming to solve this problem for an estimated profit sequence over \(n\) days.

Recurrence and Justification

Let \(p_i\) denote the profits on day \(i\) \(1 \leq i \leq n\) and \(\Pi(i)\) denote the maximum profit from days 1 to \(i\). We want to compute \(\Pi(n)\). Clearly \(\Pi(i + 1) \geq \Pi(i)\). Then

\[
\Pi(i) = \max\{\Pi(i - 2) + p_i, \Pi(i - 1)\} \quad i > 2
\]

The base cases are \(\Pi(1) = p_1\) and \(\Pi(2) = \max\{p_1, p_2\}\).

The two terms correspond to whether the tax-driver wants to go out on day \(i\) or not. If he does, then the best that he can do is add it to his profits in days 1 to \(i - 2\) since he can’t go out on day \(i - 1\) (note that this doesn’t imply that he goes out on day \(i - 2\). Otherwise, his maximum profit remains the same as \(\Pi(i - 1)\) - again it doesn’t imply that he goes out on day \(i - 1\).

Time and Space complexity

There are \(n\) entries and filling up an entry takes \(O(1)\) steps and needs only the previous two entries, i.e. time is \(O(n)\) and space is \(O(1)\). However, if you want to compute the actual schedule (not just the profit), then you need to keep track of the days and this will be \(O(n)\).