1. **Graph Shortest Paths**: Given a directed graph \( G = (V, E) \) on which each edge \((u, v) \in E\) has an associated value \(r(u, v)\), which is a real number in the range \(0 \leq r(u, v) \leq 1\) that represents the reliability of a communication channel from vertex \(u\) to vertex \(v\). We interpret \(r(u, v)\) as the probability that the channel from \(u\) to \(v\) will not fail, and we assume that these probabilities are independent. This assumption allows us to estimate the total probability of an entire path as the product of its edge probabilities. So for example if we have a path as follows:

\[
\begin{array}{c}
s \quad 0.5 \\
\quad \quad a \\
\quad 0.8 \\
\quad b \quad 0.8 \\
\quad \quad t
\end{array}
\]

Then its total probability is \(0.5 \times 0.8 \times 0.8 = 0.32\)

Give an efficient algorithm to find the most reliable path between two given vertices \(s\) and \(t\). In other words, the path whose probability product is maximum. **(Hint: You should eventually use Dijkstra’s algorithm, but not before you convert the graph from a "maximum product" to a "minimum sum". We already learned how to do it...).**

2. **Greedy algorithms**

(a) Show that the greedy cashier (change making) algorithm works for the following set of coins: \(\{2.5, 5, 10, 25, 45\}\).

**Hint:** Cap every combination of smaller coins by a combination of larger coins that has the same amount at most the same number of coins, and show that this is what the greedy algorithm would select.

(b) Show that when you add a 35 coins, so that now you have the following coins: \(\{2.5, 5, 10, 25, 35, 45\}\), the greedy method does not give you the minimum number of coins for every combination. **Hint:** Don’t try integer multiples of that 35, but a combination of it + a smaller one.
3. **Dynamic Programming:** In class, we discussed the Independent Set problem being NP complete. As a reminder, an maximum independent set is a maximal set of mutually disconnected nodes in a graph. In other words, we should find a maximum set of nodes such that no two nodes in the set have an edge between them. When the graph is a binary tree, the maximum size independent set can be found efficiently using a dynamic programming algorithm.

(a) First, notice that every node $v$ in the tree can either be picked or not picked for the independent set: If $v$ is picked, its children obviously cannot be picked but its grandchildren can be considered. If $v$ is not picked, its children can be considered. Whether we should pick $v$ or not, depends on which one of the choices (pick or not pick) will result in the largest independent set. Define the recursive formulation to the problem. Don’t forget the boundary conditions. Don’t assume the binary tree is complete.

(b) The algorithm you described above have many overlapping subproblems. Show a dynamic programming formulation for the problem. What is the runtime?

(c) Show the maximum independent set for the tree in the figure below. Mark by * the nodes that are picked for the maximum independent set.
4. Character coding:

a. Which of the following sets of codes are prefix codes for a file containing only the characters, a, b, c and d?
   a. a = 1, b = 0, c = 01, d = 10
   b. a = 10, b = 01, c = 00, d = 11
   c. a = 1, b = 01, c = 101, d = 0001
   d. All of the above are valid
   e. None of the above is valid

b. Draw the trie that results from the following character distribution: No need to show the stages of the algorithm. You can assign left and right as you wish, as long as the trie is algorithmically correct.

<table>
<thead>
<tr>
<th>char</th>
<th>z</th>
<th>x</th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>freq</td>
<td>10</td>
<td>7</td>
<td>14</td>
<td>8</td>
<td>4</td>
</tr>
</tbody>
</table>

c. Fill the code for each character in the following table:

<table>
<thead>
<tr>
<th>char</th>
<th>binary code</th>
</tr>
</thead>
<tbody>
<tr>
<td>'z'</td>
<td></td>
</tr>
<tr>
<td>'x'</td>
<td></td>
</tr>
<tr>
<td>'a'</td>
<td></td>
</tr>
<tr>
<td>'b'</td>
<td></td>
</tr>
<tr>
<td>'c'</td>
<td></td>
</tr>
</tbody>
</table>

5. Network Flow: What is the maximum flow/minimum capacity and of an s-t cut in the flow network in the figure? The capacity of each edge appears as a label next to the edge. Explain your answer by running the max-flow algorithm and show the cut.

6. Reducibility and NP: Hamiltonian path

(a) In class we saw the directed Hamiltonian cycle problem, which is the question: Given a directed graph \( G = (V, E) \), is there a simple directed cycle that contains every node in \( G \)? Similarly, a directed Hamiltonian path is a simple path that contains every node in the graph. Show that Hamiltonian path is NP.

(b) We can show that the directed Hamiltonian Path problem is NP complete by reducing the Directed Hamilton Cycle: Given a graph \( G = (V, E) \) create a graph \( G' = (V', E') \): Pick an arbitrary vertex \( v \) in the graph, and split it into two vertices, \( v_{in} \) and \( v_{out} \). All the edges \( (u, v) \) in \( G \) become \( (u, v_{out}) \) in \( G' \) and all the edges \( (v, u) \) in \( G \) become \( (v_{in}, u) \) in \( G' \). Show this is a polynomial time construction.

(c) Show that \( G \) has a Hamiltonian cycle if and only if \( G' \) has a Hamiltonian path.

(d) (Bonus) Show that in a DAG the HAM-Path problem can be solved in polynomial time.