We consider multi-sided markets, with traders of different categories. Each deal should follow a "recipe" specifying the participating traders' categories. Our mechanism allows multiple recipes in a tree structure. It is truthful and strongly-budget-balanced. It attains asymptotically-optimal gain-from-trade.

**Algorithm 1** Ascending prices mechanism — recipe-tree.

**Input:** A market \( N \), a set of categories \( G \) and a recipe-tree \( R \).

**Output:** Strongly-budget-balanced trade.

1. **Initialization:** Let \( M_g := N_g \) for each \( g \in G \).
   - Determine initial price-vector \( p \):
     - For each non-leaf \( g \), set \( p_g := -V \).
     - For each leaf \( g \), set \( p_g := -V \cdot (\text{MaxDepth} - \text{Depth}(g) + 1) \).

2. **Using Algorithm 2**, select a set \( G^* \subseteq G \) of categories.

3. For each \( g^* \in G^* \), ask each agent in \( i \in M_{g^*} \) whether \( v_i > p_{g^*} \).
   - If an agent \( i \in M_{g^*} \) answers "no", then —
     - remove \( i \) from \( M_{g^*} \) and go back to step 2.
   - If all agents in \( M_{g^*} \) for all \( g^* \in G^* \) answer "yes", then —
     - for all \( g^* \in G^* \), let \( p_{g^*} := p_{g^*} + 1 \).
   - If after the increase \( \sum_{g \in G} p_g \cdot r_g = 0 \) for some \( r \in R \), then go on to step 4;
     - else go back to step 3.

4. **Determine final trade using Algorithm 3.**

**Algorithm 2** Given a recipe-tree, find a set of prices to increase.

**Input:** A set of categories \( G \),
  a set of remaining traders \( M_g \) for all \( g \in G \),
  and a recipe-tree \( R \) based on a tree \( T \).

**Output:** A subset of \( G \) denoting categories whose price should be increased.

0. **Initialization:** For each category \( g \in G \), let \( m_g := |M_g| = \text{the number of agents of } N_g \text{ who are in the market}. \)

1. Let \( g_0 \) be the root category. Let \( c_{g_0} := \sum_{g'} \) is a child of \( g_0 \) \( m_{g'} \)

2. If \( m_{g_0} > c_{g_0} \) (or \( g_0 \) has no children at all),
   - then return the singleton \( \{ g_0 \} \).

3. Else (\( c_{g_0} \geq m_{g_0} \)), for each child \( g' \) of \( g_0 \):
   - Recursively run Algorithm 2 on the sub-tree rooted at \( g' \).
   - Denote the outcome by \( I_{g'} \).

   Return \( \bigcup I_{g'} \) is a child of \( g_0 \) \( I_{g'} \).

<table>
<thead>
<tr>
<th>Category</th>
<th>Agents' values</th>
<th>Price-increase stops when</th>
<th>New prices</th>
<th>Price-sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>N1: buyers</td>
<td>17, 14, 13, 9, 6, 2</td>
<td>B-producer -6 exits</td>
<td>-V, -V -6, -V, -6</td>
<td>-2V - 6</td>
</tr>
<tr>
<td>N2: sellers</td>
<td>-4, -5, -8, -10</td>
<td>B-producer -5 exits</td>
<td>-V, -11, -5, -6</td>
<td>-V 11</td>
</tr>
<tr>
<td>N3: A-producers</td>
<td>-1, -3, -5</td>
<td>seller -10 exits</td>
<td>-V, -10, -5, -5</td>
<td>-V 10</td>
</tr>
<tr>
<td>N4: B-producers</td>
<td>-1, -4, -6</td>
<td>buyer 2 exits</td>
<td>2, -10, -5, -5</td>
<td>-8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>buyer 2 exits</td>
<td>2, -9, -5, -4</td>
<td>-7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>seller -8 exits</td>
<td>2, -8, -4, -4</td>
<td>-6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>buyer 6 exits</td>
<td>6, -8, -4, -4</td>
<td>-2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A-producer -3 exits</td>
<td>6, -7, -3, -4</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>price-sum crosses zero</td>
<td>7, -7, -3, -4</td>
<td>0</td>
</tr>
</tbody>
</table>

Recipies-tree:
\[ R = \{(1, 1, 0, 0), (1, 0, 1, 1)\} \]