Division Strategies - Dealing by Ones

Compiled by: Theodore M. Savich

March 31, 2025

This is a sharing division problem. With sharing division problems, the number of items in each group is unknown, while the number of groups and the total number of items are both known.

$$\boxed{\text{Number of groups}} \times \boxed{\text{Unknown Number of items in each group}} = \boxed{\text{Total number of items}}$$

Transcript

Video from Carpenter et al. (1999). Strategy descriptions and examples adapted from Hackenberg (2025).

- **Teacher:** Mr. Gomez has 12 cupcakes. He wants to put the cupcakes into four boxes, so that there's the same number in each box. How many cupcakes can go in each box?
- **Student:** Okay, 1, 2, 3, 4. I got four boxes, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,12. Now, one will go in this box, one will go in this box, one will go in this box, one will go in this box. Two will go in this box, two will go in this box, two will go in this box, two will go in this box. Three will go in this box, three will go in this box, three will go in this box, and three, will go in this box. Three cupcakes can go in each box.
- **Teacher:** Nice. Thank you, Alex.

Alex began by placing 4 unifix cubes of the same color on the table, each one standing in for a different box. He then selected 12 additional cubes to represent 12 cupcakes. One by one, he distributed a cube from this pile to each box, repeating the process until he had used all the cupcake cubes. When he finished, he observed that each box contained 3 cubes, so the answer is 3 cupcakes per box.

Dealing by Ones

Strategy Overview

Dealing by Ones is a foundational division strategy where the division is performed by incrementally removing one item at a time and counting the number of groups formed. This method is particularly useful for simple division problems and serves as the basis for more advanced strategies.

Automaton Design

We design a **Pushdown Automaton (PDA)** that systematically removes one element from the total and increments the group count until all elements have been distributed.

Automaton Tuple

The PDA is defined as the 7-tuple

$$M = (Q, \Sigma, \Gamma, \delta, q_{0/accept}, \#, F)$$

where:

- $Q = \{q_{0/accept}, q_{\text{remove}}, q_{\text{output}}\}$ is the set of states. Here, $q_{0/accept}$ is the merged start and accepting state.
- $\Sigma = \{E\}$ is the input alphabet, where E represents an element.
- $\Gamma = \{\#, G, E\}$ is the stack alphabet:
  - $\#$ is the bottom-of-stack marker.
  - G represents a group identifier.
  - E represents an element.
- $q_{0/accept}$ is the start (and accepting) state.
- $\#$ is the initial stack symbol.
- $F = \{q_{0/accept}\}$ is the set of accepting states.

Transition Function

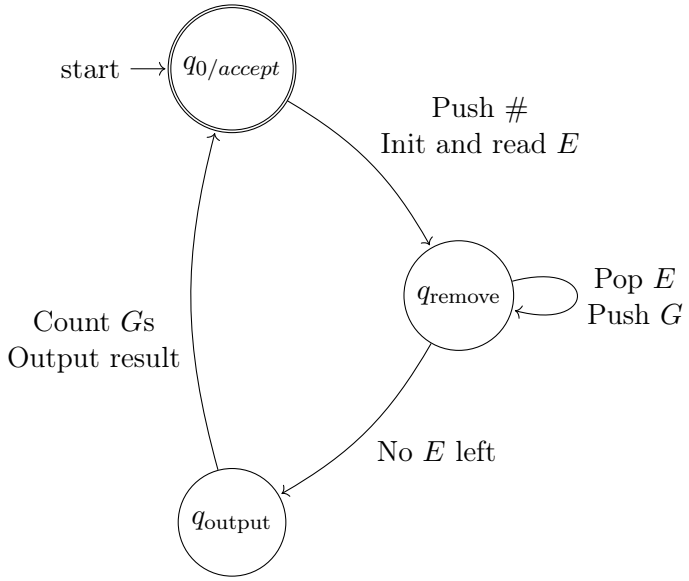
The key transitions of the PDA are as follows:

1. **Initialization:**
   $$\delta(q_{0/accept}, \varepsilon, \varepsilon) = (q_{\text{remove}}, \#)$$
   (Push the bottom marker $\#$ and move to the removal phase.)
2. **Removing Elements:**
   $$\delta(q_{\text{remove}}, \varepsilon, E) = (q_{\text{remove}}, \varepsilon \text{ (pop } E\text{) followed by pushing } G)$$
   (For each E encountered on the stack, pop it and push G to record one completed group.)
3. **Transition when no E remains:**
   $$\delta(q_{\text{remove}}, \varepsilon, \#) = (q_{\text{output}}, \#)$$
   (When no E is left (only the bottom marker remains), move to the output phase.)
4. **Outputting the Result:**
   $$\delta(q_{\text{output}}, \varepsilon, x) = (q_{0/accept}, x)$$
   (Count the number of G symbols to determine the quotient; output the final count and return to the merged start/accept state.)

State Transition Table

Current State	Input Symbol	Stack Top	Next State	Stack Operation	Description
$q_{0/accept}$	ε	—	q_{remove}	Push #	Initialization
q_{remove}	ε	E	q_{remove}	Pop E, push G	Remove one element, increment group count
q_{remove}	ε	#	q_{output}	No change	All E's removed
q_{output}	ε	(Any)	$q_{0/accept}$	Output final count	Output quotient (number of G's)

Circular PDA Diagram



Example Execution

Problem: Divide 7 items into groups of 1.

1. **Start:**
   - Initial Stack: $\# \, E \, E \, E \, E \, E \, E \, E$ (7 E's representing 7 items).
2. **Removing Elements:**
   - For each E popped, a G is pushed. After 7 removals, the stack becomes: $\# \, G \, G \, G \, G \, G \, G \, G$.
3. **Outputting the Result:**
   - The automaton counts the 7 G's and outputs the result (7 groups of 1).

HTML Implementation

```
<!DOCTYPE html>
<html>
<head>
```

```
    <title>Division: Dealing by Ones</title>
    <style>
        body { font-family: sans-serif; line-height: 1.6; }
        .container { max-width: 800px; margin: 10px auto; padding: 10px;}
        .control-section, .pile-section, .groups-section, .result-section {
            margin-bottom: 20px; padding: 10px; border: 1px solid #eee;
            background-color: #f9f9f9; border-radius: 5px;
        }
        label { margin-right: 5px;}
        input[type=number] { width: 60px; margin-right: 15px;}
        button { padding: 5px 10px; font-size: 1em; margin-right: 5px; }
        #statusMessage { color: #e65c00; font-weight: bold; margin-left: 15px;}
        .pile-container, .groups-container {
            min-height: 40px; padding: 5px; background-color: #fff; border: 1px dashed #
                ccc;
            margin-top: 5px;
         }
        .group-box {
            display: inline-block; /* Changed from flex */
            vertical-align: top; /* Align boxes at the top */
            width: 100px; /* Fixed width for each group box */
            min-height: 80px;
            border: 1px solid #999;
            padding: 5px;
            margin: 5px;
            background-color: #e8f4ff;
            text-align: center;
         }
         .group-box-label {
           font-size: 0.9em;
           color: #555;
           margin-bottom: 5px;
           display: block;
           min-height: 1.2em; /* Ensure space even if empty */
         }
        .item-block { /* Renamed from .box for clarity */
           display: inline-block;
           width: 12px; height: 12px; margin: 1px;
           background-color: dodgerblue; border: 1px solid #666;
           vertical-align: middle;
        }
         #resultValue { font-size: 1.5em; font-weight: bold; color: darkgreen; }

    </style>
</head>
<body>
<div class="container">

    <h1>Division Strategies - Dealing by Ones</h1>

    <div class="control-section">
        <label for="dealTotalInput">Total Items:</label>
        <input type="number" id="dealTotalInput" value="12" min="0">
        <label for="dealGroupsInput">Number of Groups:</label>
```

```
        <input type="number" id="dealGroupsInput" value="4" min="1">
        <!-- Ensure onclick calls the globally exposed functions -->
        <button onclick="setupSimulation()">Set Up / Reset</button>
        <button onclick="dealOneItem()" id="dealBtn" disabled>Deal One Item</button>
         <span id="statusMessage"></span>
    </div>

    <div class="pile-section">
        <strong>Items Remaining in Pile:</strong> <span id="pileCount">0</span>
        <div id="pileDisplay" class="pile-container"></div>
    </div>

    <div class="groups-section">
         <strong>Groups (Dealing items into these):</strong>
         <div id="groupsDisplay" class="groups-container">
             <!-- Group boxes will be added here -->
         </div>
    </div>

     <div class="result-section">
        <strong>Result (Items per group):</strong> <span id="resultValue">?</span>
    </div>


    <script>
        // --- Simulation State Variables (Global in this simple example) ---
        let initialTotalItems = 0;
        let numGroups = 0;
        let itemsRemaining = 0;
        let groupsData = []; // Stores item count for each group: [3, 3, 3, 3]
        let nextGroupIndex = 0;
        let isDealingComplete = true;

        // --- DOM Element References (Get them once DOM is loaded) ---
        let numericValueSpan, resultValueSpan, pileDisplay, pileCountSpan, groupsDisplay,
            dealBtn, statusMessage, totalInput, groupsInput;

        // --- Simulation Functions ---
        // Note: These are defined globally OR attached to window inside DOMContentLoaded

        function updatePileDisplay() {
            if (!pileDisplay || !pileCountSpan) return; // Check if elements exist
            pileCountSpan.textContent = itemsRemaining;
            pileDisplay.innerHTML = ""; // Clear previous
            for (let i = 0; i < itemsRemaining; i++) {
                const item = document.createElement("div");
                item.className = "item-block";
                pileDisplay.appendChild(item);
            }
        }

        function drawGroupContainers() {
            if (!groupsDisplay) return; // Check if element exists
            groupsDisplay.innerHTML = ""; // Clear previous
```

```
            for (let i = 0; i < numGroups; i++) {
                const groupBox = document.createElement("div");
                groupBox.className = "group-box";
                groupBox.id = `group-${i}`;

                const label = document.createElement("div");
                label.className = "group-box-label";
                label.textContent = `Group ${i + 1}`;
                groupBox.appendChild(label);

                const itemContainer = document.createElement("div");
                itemContainer.id = `group-items-${i}`;
                groupBox.appendChild(itemContainer);

                groupsDisplay.appendChild(groupBox);
            }
        }

        function updateSpecificGroupBox(groupIndex) {
            const itemContainer = document.getElementById(`group-items-${groupIndex}`);
            if(itemContainer) {
                const item = document.createElement("div");
                item.className = "item-block";
                itemContainer.appendChild(item);
            }
        }

        function setupSimulation() {
            // Get elements again in case they weren't ready before DOM load
            totalInput = totalInput || document.getElementById("dealTotalInput");
            groupsInput = groupsInput || document.getElementById("dealGroupsInput");
            resultValueSpan = resultValueSpan || document.getElementById("resultValue");
            pileCountSpan = pileCountSpan || document.getElementById("pileCount");
            pileDisplay = pileDisplay || document.getElementById("pileDisplay");
            groupsDisplay = groupsDisplay || document.getElementById("groupsDisplay");
            dealBtn = dealBtn || document.getElementById("dealBtn");
            statusMessage = statusMessage || document.getElementById("statusMessage");

            if (!totalInput || !groupsInput || !resultValueSpan || !pileCountSpan || !
                pileDisplay || !groupsDisplay || !dealBtn || !statusMessage) {
                console.error("One␣or␣more␣required␣elements␣not␣found␣during␣setup!");
                return;
            }


            initialTotalItems = parseInt(totalInput.value);
            numGroups = parseInt(groupsInput.value);

            if (isNaN(initialTotalItems) || isNaN(numGroups) || numGroups <= 0 ||
                initialTotalItems < 0) {
                statusMessage.textContent = "Please␣enter␣valid␣positive␣numbers␣(Groups␣>␣
                    0).";
                dealBtn.disabled = true;
                isDealingComplete = true;
```

```
            resultValueSpan.textContent = "?";
            pileCountSpan.textContent = "0";
            pileDisplay.innerHTML = "";
            groupsDisplay.innerHTML = "";
            return;
        }

        itemsRemaining = initialTotalItems;
        groupsData = Array(numGroups).fill(0); // Initialize group counts to 0
        nextGroupIndex = 0;
        isDealingComplete = (itemsRemaining === 0); // Complete if starting with 0 items

        statusMessage.textContent = itemsRemaining > 0 ? "Ready to deal." : "No items to deal.";
        resultValueSpan.textContent = "?";
        updatePileDisplay();
        drawGroupContainers(); // Draw the empty boxes
        dealBtn.disabled = isDealingComplete;
    }

    function dealOneItem() {
        if (!dealBtn || !statusMessage || !resultValueSpan) { // Check elements exist
            console.error("Button or status element not found during deal!");
            return;
        }

        if (isDealingComplete || itemsRemaining <= 0) {
            statusMessage.textContent = "Dealing complete!";
            dealBtn.disabled = true;
            return;
        }

        statusMessage.textContent = ""; // Clear message

        // 1. Decrement remaining items
        itemsRemaining--;

        // 2. Increment count for the target group
        groupsData[nextGroupIndex]++;

        // 3. Visually update pile and target group
        updatePileDisplay();
        updateSpecificGroupBox(nextGroupIndex);

        // 4. Move to next group index (cycle)
        nextGroupIndex = (nextGroupIndex + 1) % numGroups;

        // 5. Check for completion
        if (itemsRemaining === 0) {
            isDealingComplete = true;
            dealBtn.disabled = true;
            statusMessage.textContent = "Dealing complete!";
```

```
                resultValueSpan.textContent = groupsData[0]; // Show result (items in first group)
            } else {
                statusMessage.textContent = `Dealt 1 item to Group ${nextGroupIndex === 0 ? numGroups : nextGroupIndex}. ${itemsRemaining} left.`;
            }
        }


        // --- Initialize after DOM is loaded ---
        document.addEventListener('DOMContentLoaded', function() {
            // Assign elements to variables now that DOM is ready
            resultValueSpan = document.getElementById("resultValue");
            pileDisplay = document.getElementById("pileDisplay");
            pileCountSpan = document.getElementById("pileCount");
            groupsDisplay = document.getElementById("groupsDisplay");
            dealBtn = document.getElementById("dealBtn");
            statusMessage = document.getElementById("statusMessage");
            totalInput = document.getElementById("dealTotalInput");
            groupsInput = document.getElementById("dealGroupsInput");

            // Now that functions are defined, attach to window if needed by HTML onclick
            // Alternatively, add event listeners here instead of using onclick in HTML
            window.setupSimulation = setupSimulation;
            window.dealOneItem = dealOneItem;


            // Initialize the display on page load
            setupSimulation();

        }); // <<< --- THIS was the likely extra '}' or missing scope boundary ---

    </script>

</div> <!-- End Container -->
</body>
</html>
```

References

Carpenter, T. P., Fennema, E., Franke, M. L., Levi, L., & Empson, S. B. (1999). Children's mathematics: Cognitively guided instruction – videotape logs [supplementary material]. In *Children's mathematics: Cognitively guided instruction*. Heinemann, in association with The National Council of Teachers of Mathematics, Inc.

Hackenberg, A. (2025). *Course notes* [Unpublished course notes].