

Division Strategies - Dealing by Ones

Compiled by: Theodore M. Savich

April 1, 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:

- $Q = \{q_{0/accept}, q_{remove}, q_{output}\}$ 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:

$$\delta(q_{0/accept}, \varepsilon, \varepsilon) = (q_{remove}, \#)$$

(Push the bottom marker $\#$ and move to the removal phase.)

2. Removing Elements:

$$\delta(q_{remove}, \varepsilon, E) = (q_{remove}, \varepsilon \text{ (pop } E) \text{ followed by pushing } G)$$

(For each E encountered on the stack, pop it and push G to record one completed group.)

3. Transition when no E remains:

$$\delta(q_{remove}, \varepsilon, \#) = (q_{output}, \#)$$

(When no E is left (only the bottom marker remains), move to the output phase.)

4. Outputting the Result:

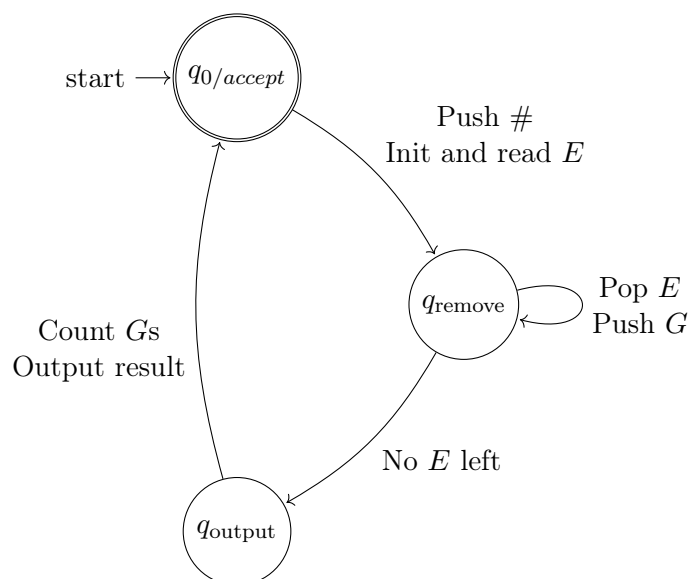
$$\delta(q_{output}, \varepsilon, x) = (q_{0/accept}, x)$$

(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

```

1 <!DOCTYPE html>
2 <html>
3 <head>

```

```

4 <title>Division: Dealing by Ones</title>
5 <style>
6   body { font-family: sans-serif; line-height: 1.6; }
7   .container { max-width: 800px; margin: 10px auto; padding: 10px;}
8   .control-section, .pile-section, .groups-section, .result-section {
9     margin-bottom: 20px; padding: 10px; border: 1px solid #eee;
10    background-color: #f9f9f9; border-radius: 5px;
11  }
12  label { margin-right: 5px;}
13  input[type=number] { width: 60px; margin-right: 15px;}
14  button { padding: 5px 10px; font-size: 1em; margin-right: 5px; }
15  #statusMessage { color: #e65c00; font-weight: bold; margin-left: 15px;}
16  .pile-container, .groups-container {
17    min-height: 40px; padding: 5px; background-color: #fff; border: 1px dashed #
18    ccc;
19    margin-top: 5px;
20  }
21  .group-box {
22    display: inline-block; /* Changed from flex */
23    vertical-align: top; /* Align boxes at the top */
24    width: 100px; /* Fixed width for each group box */
25    min-height: 80px;
26    border: 1px solid #999;
27    padding: 5px;
28    margin: 5px;
29    background-color: #e8f4ff;
30    text-align: center;
31  }
32  .group-box-label {
33    font-size: 0.9em;
34    color: #555;
35    margin-bottom: 5px;
36    display: block;
37    min-height: 1.2em; /* Ensure space even if empty */
38  }
39  .item-block { /* Renamed from .box for clarity */
40    display: inline-block;
41    width: 12px; height: 12px; margin: 1px;
42    background-color: dodgerblue; border: 1px solid #666;
43    vertical-align: middle;
44  }
45  #resultValue { font-size: 1.5em; font-weight: bold; color: darkgreen; }
46 </style>
47 </head>
48 <body>
49 <div class="container">
50
51   <h1>Division Strategies - Dealing by Ones</h1>
52
53   <div class="control-section">
54     <label for="dealTotalInput">Total Items:</label>
55     <input type="number" id="dealTotalInput" value="12" min="0">
56     <label for="dealGroupsInput">Number of Groups:</label>

```

```

57     <input type="number" id="dealGroupsInput" value="4" min="1">
58     <!-- Ensure onclick calls the globally exposed functions -->
59     <button onclick="setupSimulation()">Set Up / Reset</button>
60     <button onclick="dealOneItem()" id="dealBtn" disabled>Deal One Item</button>
61     <span id="statusMessage"></span>
62 </div>
63
64 <div class="pile-section">
65     <strong>Items Remaining in Pile:</strong> <span id="pileCount">0</span>
66     <div id="pileDisplay" class="pile-container"></div>
67 </div>
68
69 <div class="groups-section">
70     <strong>Groups (Dealing items into these):</strong>
71     <div id="groupsDisplay" class="groups-container">
72         <!-- Group boxes will be added here -->
73     </div>
74 </div>
75
76 <div class="result-section">
77     <strong>Result (Items per group):</strong> <span id="resultValue">?</span>
78 </div>
79
80
81 <script>
82     // --- Simulation State Variables (Global in this simple example) ---
83     let initialTotalItems = 0;
84     let numGroups = 0;
85     let itemsRemaining = 0;
86     let groupsData = []; // Stores item count for each group: [3, 3, 3, 3]
87     let nextGroupIndex = 0;
88     let isDealingComplete = true;
89
90     // --- DOM Element References (Get them once DOM is loaded) ---
91     let numericValueSpan, resultValueSpan, pileDisplay, pileCountSpan, groupsDisplay,
92         dealBtn, statusMessage, totalInput, groupsInput;
93
94     // --- Simulation Functions ---
95     // Note: These are defined globally OR attached to window inside DOMContentLoaded
96
97     function updatePileDisplay() {
98         if (!pileDisplay || !pileCountSpan) return; // Check if elements exist
99         pileCountSpan.textContent = itemsRemaining;
100         pileDisplay.innerHTML = ""; // Clear previous
101         for (let i = 0; i < itemsRemaining; i++) {
102             const item = document.createElement("div");
103             item.className = "item-block";
104             pileDisplay.appendChild(item);
105         }
106
107     function drawGroupContainers() {
108         if (!groupsDisplay) return; // Check if element exists
109         groupsDisplay.innerHTML = ""; // Clear previous

```

```

110     for (let i = 0; i < numGroups; i++) {
111         const groupBox = document.createElement("div");
112         groupBox.className = "group-box";
113         groupBox.id = `group-${i}`;
114
115         const label = document.createElement("div");
116         label.className = "group-box-label";
117         label.textContent = `Group ${i + 1}`;
118         groupBox.appendChild(label);
119
120         const itemContainer = document.createElement("div");
121         itemContainer.id = `group-items-${i}`;
122         groupBox.appendChild(itemContainer);
123
124         groupsDisplay.appendChild(groupBox);
125     }
126 }
127
128 function updateSpecificGroupBox(groupIndex) {
129     const itemContainer = document.getElementById(`group-items-${groupIndex}`);
130     if(itemContainer) {
131         const item = document.createElement("div");
132         item.className = "item-block";
133         itemContainer.appendChild(item);
134     }
135 }
136
137 function setupSimulation() {
138     // Get elements again in case they weren't ready before DOM load
139     totalInput = totalInput || document.getElementById("dealTotalInput");
140     groupsInput = groupsInput || document.getElementById("dealGroupsInput");
141     resultValueSpan = resultValueSpan || document.getElementById("resultValue");
142     pileCountSpan = pileCountSpan || document.getElementById("pileCount");
143     pileDisplay = pileDisplay || document.getElementById("pileDisplay");
144     groupsDisplay = groupsDisplay || document.getElementById("groupsDisplay");
145     dealBtn = dealBtn || document.getElementById("dealBtn");
146     statusMessage = statusMessage || document.getElementById("statusMessage");
147
148     if (!totalInput || !groupsInput || !resultValueSpan || !pileCountSpan || !
149         pileDisplay || !groupsDisplay || !dealBtn || !statusMessage) {
150         console.error("One or more required elements not found during setup!");
151         return;
152     }
153
154     initialTotalItems = parseInt(totalInput.value);
155     numGroups = parseInt(groupsInput.value);
156
157     if (isNaN(initialTotalItems) || isNaN(numGroups) || numGroups <= 0 ||
158         initialTotalItems < 0) {
159         statusMessage.textContent = "Please enter valid positive numbers (Groups >
160             0).";
161         dealBtn.disabled = true;
162         isDealingComplete = true;

```

```

161         resultValueSpan.textContent = "?";
162         pileCountSpan.textContent = "0";
163         pileDisplay.innerHTML = "";
164         groupsDisplay.innerHTML = "";
165         return;
166     }
167
168     itemsRemaining = initialTotalItems;
169     groupsData = Array(numGroups).fill(0); // Initialize group counts to 0
170     nextGroupIndex = 0;
171     isDealingComplete = (itemsRemaining === 0); // Complete if starting with 0
172         items
173
174     statusMessage.textContent = itemsRemaining > 0 ? "Ready_to_deal." : "No_items_
175         to_deal.";
176     resultValueSpan.textContent = "?";
177     updatePileDisplay();
178     drawGroupContainers(); // Draw the empty boxes
179     dealBtn.disabled = isDealingComplete;
180 }
181
182 function dealOneItem() {
183     if (!dealBtn || !statusMessage || !resultValueSpan) { // Check elements exist
184         console.error("Button_or_status_element_not_found_during_deal!");
185         return;
186     }
187
188     if (isDealingComplete || itemsRemaining <= 0) {
189         statusMessage.textContent = "Dealing_complete!";
190         dealBtn.disabled = true;
191         return;
192     }
193
194     statusMessage.textContent = ""; // Clear message
195
196     // 1. Decrement remaining items
197     itemsRemaining--;
198
199     // 2. Increment count for the target group
200     groupsData[nextGroupIndex]++;
201
202     // 3. Visually update pile and target group
203     updatePileDisplay();
204     updateSpecificGroupBox(nextGroupIndex);
205
206     // 4. Move to next group index (cycle)
207     nextGroupIndex = (nextGroupIndex + 1) % numGroups;
208
209     // 5. Check for completion
210     if (itemsRemaining === 0) {
211         isDealingComplete = true;
212         dealBtn.disabled = true;
213         statusMessage.textContent = "Dealing_complete!";

```

```

212         resultValueSpan.textContent = groupsData[0]; // Show result (items in first
           group)
213     } else {
214         statusMessage.textContent = 'Dealt 1 item to Group ${nextGroupIndex === 0
           ? numGroups : nextGroupIndex}. ${itemsRemaining} left.';
215     }
216 }
217
218
219 // --- Initialize after DOM is loaded ---
220 document.addEventListener('DOMContentLoaded', function() {
221     // Assign elements to variables now that DOM is ready
222     resultValueSpan = document.getElementById("resultValue");
223     pileDisplay = document.getElementById("pileDisplay");
224     pileCountSpan = document.getElementById("pileCount");
225     groupsDisplay = document.getElementById("groupsDisplay");
226     dealBtn = document.getElementById("dealBtn");
227     statusMessage = document.getElementById("statusMessage");
228     totalInput = document.getElementById("dealTotalInput");
229     groupsInput = document.getElementById("dealGroupsInput");
230
231     // Now that functions are defined, attach to window if needed by HTML onclick
232     // Alternatively, add event listeners here instead of using onclick in HTML
233     window.setupSimulation = setupSimulation;
234     window.dealOneItem = dealOneItem;
235
236
237     // Initialize the display on page load
238     setupSimulation();
239
240     }); // <<< --- THIS was the likely extra '}' or missing scope boundary ---
241
242 </script>
243
244 <!-- New button for viewing PDF documentation -->
245 <button onclick="openPdfViewer()">Want to learn more about this strategy? Click here
    .</button>
246
247 <script>
248     function openPdfViewer() {
249         // Opens the PDF documentation for the strategy.
250         window.open('../SMR_DIV_Dealing_By_Ones.pdf', '_blank');
251     }
252 </script>
253
254 </div> <!-- End Container -->
255 </body>
256 </html>

```


References

- Carpenter, T. P., Fennema, E., Franke, M. L., Levi, L., & Empson, S. B. (1999). *Children's mathematics: Cognitively guided instruction* [Includes supplementary material: Children's mathematics: Cognitively guided instruction – videotape logs]. Heinemann; The National Council of Teachers of Mathematics, Inc.
- Hackenberg, A. (2025). *Course notes* [Unpublished course notes].