Lemma. Let $u \in \mathbb{R}_{\geq 0}^n$ with $C = \|u\|_1 / n$. We can find $v \in [0, C]^n$ and $\pi \in \{0, 1 \ldots n\}^n$ such that $\pi_i = 0$ iff $v_i = C$ and

$$u_i = v_i + \sum_{j=1}^{n} [[\pi_j = i]] (C - v_j) \quad (1)$$

Proof. If $n = 1$, setting $v_1 = u_1 = C$ and $\pi_1 = 0$ satisfies (1). If $n > 1$,

1. Find $k \in \{1 \ldots n\}$ with $u_k \leq C$ (which must exist): without loss of generality assume $k = n$.

2. Find $l \neq k$ with $u_l \geq C$ (which must exist): without loss of generality assume $l = n - 1$.

Define $\bar{u} \in \mathbb{R}_{\geq 0}^{n-1}$ by

$$\bar{u}_i = \begin{cases} u_{n-1} - (C - u_n) & \text{if } i < n - 1 \\ u_n & \text{if } i = n - 1 \end{cases}$$

Note that $\bar{u}_{n-1} \geq 0$ since $u_{n-1} \geq C$ and $C - u_n \leq C$. Also, $C = \|\bar{u}\|_1 / (n - 1)$ since $\|\bar{u}\|_1 = \|u\|_1 - u_n - (C - u_n) = C(n - 1)$. By an inductive step, we can find $\bar{v} \in [0, C]^{n-1}$ and $\bar{\pi} \in \{0, 1 \ldots n - 1\}^{n-1}$ such that

$$\bar{v}_i = \bar{u}_i - \sum_{j=1}^{n-1} [[\bar{\pi}_j = i]] (C - \bar{v}_j)$$

Define $v \in [0, C]^n$ and $\pi \in \{0, 1 \ldots n\}^n$ by

$$v_i = \begin{cases} \bar{v}_i & \text{if } i < n \\ u_n & \text{if } i = n \end{cases} \quad \pi_i = \begin{cases} \bar{\pi}_i & \text{if } i < n \\ n - 1 & \text{if } i = n \end{cases}$$

We verify that this construction satisfies (1) for each index.

• ($i = n$): $v_n = u_n$ and $\pi_l \neq n$ for all $l \in \{1 \ldots n\}$.

• ($i = n - 1$):

$$v_{n-1} = \bar{v}_{n-1} = \bar{u}_{n-1} - \sum_{j=1}^{n-1} [[\bar{\pi}_j = n - 1]] (C - \bar{v}_j) = u_{n-1} - (C - u_n) - \sum_{j=1}^{n-1} [[\bar{\pi}_j = n - 1]] (C - \bar{v}_j) = u_{n-1} - n \sum_{j=1}^{n} [[\pi_j = n - 1]] (C - v_j)$$

* A formalization of the write-up by Schwarz (2020).
• \((i < n - 1)\):

\[
v_i = \bar{v}_i \\
= \bar{u}_i - \sum_{j=1}^{n-1} \left[ \begin{array}{c}
\pi_j = i
\end{array} \right] \left( C - \bar{v}_j \right)
\]

\[
u_i = u_i - \sum_{j=1}^{n-1} \left[ \begin{array}{c}
\pi_j = i
\end{array} \right] \left( C - v_j \right)
\]

The alias method. Let \(p \in \Delta^{n-1} \). By the lemma using \(u = np \) (so \(C = 1\)), we can construct \(v \in [0,1]^n\) and \(\pi \in \{0,1\ldots n\}^n\) (“alias table”) such that

\[
p_i = \frac{1}{n} \left( v_i \left. + \sum_{j=1}^{n} \left[ \begin{array}{c}
\pi_j = i
\end{array} \right] \left( 1 - v_j \right) \right) \right)
\]

\[
= \frac{1}{n} \sum_{j=1}^{n} v_j \left[ j > i \right] + (1 - v_j) \left[ \pi_j = i \right]
\]

\[
= \Pr_{j \sim \text{Unif}(\{1\ldots n\}), x \sim \text{Ber}(v_j)} \left( (x = 1 \land j = i) \lor (x = 0 \land \pi_j = i) \right)
\]

Thus assuming the knowledge of such \(v, \pi\) and the ability to sample from a uniform distribution over \(n\) items and the Bernoulli distribution in \(O(1)\) time (e.g., by applications of sampling from a uniform real distribution), we can sample \(i \sim \text{Cat}(p)\) in \(O(1)\) time by sampling \(j \sim \text{Unif}(\{1\ldots n\})\), \(x \sim \text{Ber}(v_j)\), then setting \(i = j\) if \(x = 1\) and \(i = \pi_j\) if \(x = 0\) (which never happens if \(\pi_j = 0\)).

Algorithm for constructing \((v, \pi)\). The proof of the lemma is constructive and a recursive algorithm itself. Here is an in-place iterative version of the algorithm:

<table>
<thead>
<tr>
<th>FindAlias</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> (u \in \mathbb{R}^n_{\geq 0}) with (C = |u|_1/n)</td>
</tr>
</tbody>
</table>
| **Output:** \(v \in [0,C]^n\) and \(\pi \in \{0,1\ldots n\}^n\) such that \(\pi_i = 0\) iff \(v_i = C\) and \(u_i = v_i + \sum_{j=1}^{n} \left[ \begin{array}{c}
\pi_j = i
\end{array} \right] \left( C - v_j \right)\)
| **Runtime:** \(O(n^2)\) or \(O(n \log n)\) |

1. Initialize \(v, \pi \in \mathbb{R}^n\) arbitrarily and set \(I \leftarrow \{1\ldots n\}\).
2. While \(I \neq \emptyset\)
   (a) If \(I = \{i\}\) (we must have \(u_i = C\)), set \(v_i \leftarrow C\), \(\pi_i \leftarrow 0\), and \(I \leftarrow \emptyset\).
   (b) Else, search for
   \[
k \in \{ i \in I : u_i \leq C \}
\]
   \[
l \in \{ i \in I : i \neq k, u_i \geq C \}
\]
   and set \(v_k \leftarrow u_k, \pi_k \leftarrow l, u_l \leftarrow u_l - (C - u_k),\) and \(I \leftarrow I \setminus \{k\}\).

A naive implementation of **FindAlias** yields a \(O(n^2)\) runtime because of the \(O(n)\) search in (2–3).\(^1\) But we observe that we do not need to search at all if we maintain a partition of indices based on the threshold \(C\). This is first proposed by Vose (1991) and yields the \(O(n)\)-time algorithm shown below (with some numerical stability tricks):

\(^1\)This can be improved to \(O(\log n)\) by using a binary search tree.
FindAliasFast
Input: \( u \in \mathbb{R}_0^n \) with \( C = \|u\|_1/n \)
Output: \( v \in [0, C]^n \) and \( \pi \in \{0, 1, \ldots, n\}^n \) such that \( \pi_i = 0 \) iff \( v_i = C \) and \( u_i = v_i + \sum_{j=1}^n \lfloor \pi_j = i \rfloor (C - v_j) \)
Runtime: \( O(n) \)

1. Initialize \( v, \pi \in \mathbb{R}^n \) arbitrarily and set
   \[ S \leftarrow \{ i \in \{1 \ldots n\} : u_i < C \} \]
   \[ L \leftarrow \{ i \in \{1 \ldots n\} : u_i \geq C \} \]

2. While \( S \neq \emptyset \) and \( L \neq \emptyset \)
   (a) Select arbitrary \( k \in S \). Set \( v_k \leftarrow u_k \) and \( S \leftarrow S \setminus \{k\} \).
   (b) Select arbitrary \( l \in L \). Set \( \pi_k \leftarrow l \) and \( L \leftarrow L \setminus \{l\} \).
   (c) Set \( u_l \leftarrow (u_l + u_k) - C \). If \( u_l < C \), set \( S \leftarrow S \cup \{l\} \); else, set \( L \leftarrow L \cup \{l\} \).

3. For all \( l \in L \) (we must have \( u_l = C \) ), set \( v_l \leftarrow C \) and \( \pi_l \leftarrow 0 \).

4. For all \( k \in S \) (only nonempty because of numerical instability, so this means \( u_k = C \) ), set \( v_k \leftarrow C \) and \( \pi_k \leftarrow 0 \).

References
