1 Introduction

Consider a collection of containers of positive integral size, $c_1 \leq c_2 \leq \ldots \leq c_n$), with $c_i \geq 0$, and a process which at each step of the process fills s distinct containers with an item of size 1. We denote such a problem with the tuple $(\{c_i\}, n, s)$. Let t be the number of steps taken before there are fewer than s containers with non-zero capacity. We wish to develop an upper bound, $t_{\max}$ for t, and show that the algorithm in section 3 achieves that bound.

We denote the remaining capacity of containers at the "current" step as c_i, and at the following step as c'_i. Values at an arbitrary step x are denoted $c_i^{(x)}$.

2 Upper bounds on the number of steps

A trivial upper bound, t_0, on t can be found by observing that each step of the process reduces the total capacity by s, and so

$$t_0 = \left\lfloor \frac{\sum c_i}{s} \right\rfloor \tag{1}$$

is an upper bound on t.

Now, consider a very unbalanced set of containers, where a few of them (q, where $q < s$) are much larger than the rest. If we place an item into each of the the q large containers on every step, then the remaining $s - q$ items must be distributed within the $n - q$ small containers. If the large containers are large enough, we will fill all of the small containers first. We therefore have a reduced problem of placing $s - q$ items at a time into $n - q$ containers, with a corresponding trivial bound:

$$t_q = \left\lfloor \frac{\sum_{q<i\leq n} c_i}{s-q} \right\rfloor \tag{2}$$

This bound holds only if the "small" containers in the partition are sufficiently small. What counts as sufficiently small? We are placing items in all of the "large" containers at each step, so each of those containers must be larger than t_q. Since the c_i are ordered, this means that for t_q to be a bound on the number of steps, the smallest "large" container is c_q, and therefore t_q is a bound if

$$c_q \geq t_q. \tag{3}$$

An upper bound on the number of steps possible is therefore the minimum of the trivial bound and the valid q-bounds:

$$t_{\max} = \min\left(t_0, \min_{q:q<s, c_q \geq t_q} t_q\right). \tag{4}$$

Any process which achieves this bound will have placed $st_{\max}$ items into the containers, and if the bounding constraint is t_q (for $0 \leq q < s$), then every container c_i with $1 \leq i \leq q$ will have $t_{\max}$ items in it. The other containers, with $q + 1 \leq i \leq n$, will be filled with $(s - q)t_q$ items between them, leaving $(\sum_{q<i\leq n} c_i) - (s - q)t_q$ capacity unfilled.

3 Optimal algorithm

Take the s containers with the largest free capacity. Add one item to each of those. Repeat until there are fewer than s containers with available capacity. Believed to be optimal, but not proven.