Supporting Information

Novikoff et al. 10.1073/pnas.1109863109

SI Text

The Recap Method. In *The Recap Method* in the paper, we described a schedule in terms of a depth-first traversal of a full binary tree, and claimed that it conformed to the spacing constraints

$$a_k = 2^k$$
$$b_k = 2^{k-1}(k+1).$$

We also claimed that for this schedule, which we refer to as the recap schedule, the number of time steps before n distinct educational units have been introduced, denoted in the paper as t_n, grows as $\Theta(n \log n)$, and that for a certain class of functions $r(n)$ we can explicitly construct schedules for which t_n grows as $\Theta(n \cdot r^{-1}(n))$.

First we prove the results about the original recap schedule in *The Recap Schedule*, and then we generalize these results in *The Generalized Recap Schedule*. In both subsections we use notation slightly different from the paper by subtracting one from all the indices of the educational units, so that the lowest index is zero. This notational change will make calculations easier and will allow for a cleaner generalization later on.

The recap schedule. We begin by reiterating how to construct the recap schedule. To find the first $(k+1)2^k$ entries of the schedule, consider a depth-first postorder traversal of a full binary tree of height k with 2^k leaves labeled $u_0, u_1, \ldots, u_{2^k-1}$ from left to right (see Fig. S1). Begin with an empty sequence. Every time a leaf is visited, append the sequence with the corresponding educational unit. Every time a nonleaf node is visited (after both children have been visited), append the sequence with the units corresponding to all of the descendant leaves, in left-to-right order. To be clear, we mean for the leaves to have height zero, their parents to have height one, etc.

Thus, using $k = 2$, we have that the first 12 entries of the schedule are

$$u_0, u_1, u_0, u_1, u_2, u_3, u_2, u_3, u_0, u_1, u_2, u_3.$$

It should be noted that, by the properties of depth-first postorder traversal, this description defines a unique sequence, because the first $(k+1)2^k$ elements of the sequence are the same regardless of whether one considers a tree of height k or one of height greater than k. Thus in the discussion and proofs below we simply assume that the tree being discussed is always of sufficient height to include all of the relevant nodes.

The following lemma, which should be clear from the diagram in Fig. S1, is justified by the fact that a depth-first postorder traversal of a tree will visit, in order of increasing height, each node on the path from any given leaf to the root. The lemma follows from this fact and from basic properties of a full binary tree.

Lemma 1. *(The Recap Lemma). In the construction of the recap schedule, the left-most node at height k corresponds to the* $(k+1)^{\mathrm{st}}$ *occurrences of units* $u_0, u_1, \ldots, u_{2^k-1}$*, and the sibling of that node corresponds to the* $(k+1)^{\mathrm{st}}$ *occurrences of units* $u_{2^k}, \ldots, u_{2^{k+1}-1}$.

To make the discussion below more concise, we introduce some notation. Let $T_i(k)$ be the index of the kth occurrence of unit u_i in the sequence. Note that $t_n = T_n(1)$ by this definition. Thus the recap lemma states that in the recap schedule, the left-most node at height k corresponds to $T_i(k+1)$ for $i \in [0, 2^k - 1]$, and the sibling of that node corresponds to $T_i(k+1)$ for $i \in [2^k, 2^{k+1} - 1]$.

We are now ready to prove the statements about the recap schedule from the paper.

Theorem 1. *(Asymptotics of the Introduction Time Function.) In the recap schedule,* $t_n = T_n(1)$ *grows as* $\Theta(n \log n)$.

Proof: By the recap lemma and properties of depth-first postorder traversal, at time step $T_{2^k}(1)$ units $u_0, u_1, \ldots, u_{2^k-1}$ have each occurred exactly $k+1$ times, and nothing else has occurred at all. Therefore,

$$T_{2^k}(1) = 2^k \cdot (k+1),$$

and so

$$T_n(1) = n \cdot (\log_2 n + 1)$$

for n of the form $n = 2^k$, which establishes that $T_n(1)$ grows as $\Theta(n \log n)$ when considered as a function of integers of the form $n = 2^k$.

Because $T_n(1)$ increases monotonically in n, it follows that that $T_n(1)$ grows as $\Theta(n \log n)$ when considered as a function of all positive integers.

Theorem 2. *(Bounds on the Introduction Time Function.) In the recap schedule,*

$$T_n(1) \le n \cdot (\lfloor \log_2 n \rfloor + 1)$$

and

$$\frac{1}{2} \cdot n \cdot (\lfloor \log_2 n \rfloor + 1) \le T_n(1)$$

for all n.

Proof: In general, by time step $T_n(1)$, only units $u_0, u_1, \ldots, u_{n-1}$ have already occurred at all, by the properties of depth-first postorder traversal, and each at most $\lfloor \log_2 n \rfloor + 1$ times, by the recap lemma. Therefore,

$$T_n(1) \le n \cdot (\lfloor \log_2 n \rfloor + 1).$$

Furthermore, by time step $T_n(1)$, all units with index less than $\frac{1}{2}n$ have occurred exactly $\lfloor \log_2 n \rfloor + 1$ times, again by the properties of depth-first postorder traversal and the recap lemma. Therefore,

$$\frac{1}{2} \cdot n \cdot (\lfloor \log_2 n \rfloor + 1) \le T_n(1).$$

Theorem 3. *(Adherence to Spacing Constraints.) The recap schedule adheres to the spacing constraints*

$$a_k = 2^k$$
$$b_k = 2^{k-1}(k+1).$$

Proof: Our goal is to show that

$$a_k \leq T_i(k+1) - T_i(k) \leq b_k,$$

for all i, k. We will establish these bounds by calculating the minimum and maximum possible values of $T_i(k+1) - T_i(k)$.

Because the tree is a full binary tree, we have that for any k all the subtrees with roots at height k are identical. Therefore all values of $T_i(k+1) - T_i(k)$ which occur in the context of any given subtree at height k must occur in the context of the subtree rooted at the left-most node at height k. Thus, for any k, we only need to consider $i < 2^k$ in order to find the minimum and maximum values of

$$T_i(k+1) - T_i(k).$$

By construction

$$T_j(k+1) - T_i(k+1) = j - i$$

whenever $i < j < 2^k$. Also, because $T_i(k)$ is monotonic in i,

$$T_j(k) - T_i(k) \geq j - i$$

whenever $i < j$. Therefore, for all i and j such that $i < j < 2^k$ we have that

$$T_j(k+1) - T_j(k) \leq T_i(k+1) - T_i(k).$$

Thus, the maximum value of $T_i(k+1) - T_i(k)$ must occur when $i = 0$ and the minimum value must occur when $i = 2^k - 1$.

Thus if we can show that

$$T_{2^k-1}(k+1) - T_{2^k-1}(k) \geq 2^k$$

and that

$$T_0(k+1) - T_0(k) \leq 2^{k-1}(k+1)$$

for all k, then we will be done. We will in fact show that we have equality in both cases; we will show that

$$T_{2^k-1}(k+1) - T_{2^k-1}(k) = 2^k$$

and

$$T_0(k+1) - T_0(k) = 2^{k-1}(k+1)$$

for all k.

By construction, the last entry of the schedule due to the left-most node at height k corresponds to $T_{2^k-1}(k+1)$, and the last entry of the schedule due to the right child of that node corresponds to $T_{2^k-1}(k)$. Because in depth-first postorder traversal each node is visited immediately after its right child, we have that

$$T_{2^k-1}(k+1) - T_{2^k-1}(k) = 2^k,$$

corresponding to the 2^k entries of the schedule due to the left-most node at height k.

Meanwhile, $T_0(k+1)$ and $T_0(k)$ refer to the first entry of the schedule due to the left-most nodes at heights k and $k-1$, respectively. Thus $T_0(k+1) - T_0(k)$ will be equal to the number of entries in the schedule due to the left-most node at height $k-1$, plus the number of entries due to the subtree whose root is the right sibling of the left-most node at height $k-1$.

The first quantity is 2^{k-1}, by construction. As for the second quantity, because the subtree in question corresponds to k occurrences of 2^{k-1} units, we have that the second quantity is equal to $k \cdot 2^{k-1}$. Thus

$$\begin{aligned} T_0(k+1) - T_0(k) &= 2^{k-1} + k \cdot 2^{k-1} \\ &= 2^{k-1}(k+1). \end{aligned}$$

Corollary 1. *(Window Growth.) The minimal window length required for the general recap schedule, $b_k - a_k$, grows as $\Theta(k \cdot 2^k)$.*

Proof: This proof follows from the bounds above, because

$$\begin{aligned} b_k - a_k &= 2^{k-1}(k+1) - 2^k \\ &= 2^{k-1}(k+1-2) \\ &= \frac{1}{2} \cdot 2^k \cdot (k-1). \end{aligned}$$

The generalized recap schedule. We now move on to generalizing these results by considering a class of schedules which we call the "generalized recap schedule." To that end, we consider a class of trees more general than the full binary tree; we consider trees where at any given height all of the nodes have the same number of children, but where this number is not necessarily two for every height (as it is in a full binary tree).

To construct a generalized recap schedule, begin with any sequence of positive integers

$$\{q(i)\},$$

such that $q(i) \geq 2$ for all i. Then define a sequence

$$\{r(i)\}$$

by setting $r(0) = 1$ and letting

$$r(i) = \prod_{j=1}^{i} q(j)$$

for $i \geq 1$.

Now, to find the first $(k+1)r(k)$ entries of the schedule, consider a depth-first postorder traversal of a tree of height k with $r(k)$ leaves labeled $u_0, u_1, \ldots, u_{r(k)-1}$ from left to right, and such that all the nodes at height j have exactly $q(j)$ children. Begin with an empty sequence. As before, every time a leaf is visited, append the sequence with the corresponding educational unit. Every time a nonleaf node is visited (after all of its children have been visited), append the sequence with the units corresponding to all of the descendant leaves, in left-to-right order. Again, we mean for the leaves to have height zero, their parents to have height one, etc.

Thus, for example, using $q(i) \equiv 2$, we simply have the original recap schedule, whereas using

$$\{q(i)\} = 3, 2, \ldots,$$

as in the diagram in Fig. S2, we have that the first 18 entries of the schedule are

$$\begin{gathered} u_0, u_1, u_2, u_0, u_1, u_2, \\ u_3, u_4, u_5, u_3, u_4, u_5, \\ u_0, u_1, u_2, u_3, u_4, u_5. \end{gathered}$$

Again it should be noted that, by the properties of depth-first postorder traversal, this description defines a unique sequence, because the first $(k+1)r(k)$ elements of the sequence are the same regardless of whether one considers a tree of height k or one of height greater than k. Thus in the discussion and proofs below, we simply assume that the tree being discussed is always of sufficient height to include all the relevant nodes.

We would like to extend r^{-1} so that it has an inverse defined for all positive integers. Where r^{-1} is not naturally defined [i.e., for positive integers n such that $r(i) \neq n$ for any i], we define $r^{-1}(n)$ to simply be $r^{-1}(m)$ where m is the largest number less than n such that $r(i) = m$ for some i. [Thus, for example, if $q(k) \equiv 2$, then we have that $r(k) = 2^k$ and $r^{-1}(n) = \lfloor \log_2 n \rfloor$.]

We note that $r^{-1}(n) + 1$ can be interpreted as the height of the lowest ancestor of leaf u_n that is the left-most node at that height. Thus, by the properties of depth-first postorder traversal, when leaf u_n is visited, only nodes of height less than or equal to $r^{-1}(n)$ have already been visited.

The generalization of the recap lemma is evident from the diagram in Fig. S2.

Lemma 2. *(The Recap Lemma—Generalized.) The left-most node at height k corresponds to the* $(k+1)^{\text{st}}$ *occurrences of units*

$$u_0, u_1, \ldots, u_{r(k)-1},$$

and in general the jth node at height k, counting from left to right, corresponds to the $(k+1)^{\text{st}}$ *occurrences of units*

$$u_{(j-1)r(k)}, \ldots, u_{jr(k)-1}.$$

In other words, the left-most node at height k corresponds to $T_i(k+1)$ *for* $i \in [0, r(k)-1]$*, and the jth node at height k corresponds to* $T_i(k+1)$ *for*

$$i \in [(j-1)r(k), jr(k)-1].$$

With this lemma in hand, we prove the main results about the generalized recap schedule. The structure of all of the proofs mirrors the structure of analogous proofs in *The Recap Schedule*.

Theorem 4. *(Asymptotics of the Introduction Time Function—Generalized.) In the generalized recap schedule,* $T_n(1)$ *grows as* $\Theta(n \cdot r^{-1}(n))$.

Proof: By the properties of depth-first postorder traversal and the recap lemma, at time step $T_{r(k)}(1)$, units $u_0, u_1, \ldots, u_{r(k)-1}$ have each occurred exactly $k+1$ times, and nothing else has occurred at all. Therefore,

$$T_{r(k)}(1) = r(k) \cdot (k+1),$$

and so

$$T_n(1) = n \cdot [r^{-1}(n) + 1]$$

for n of the form $n = r(k)$ for some positive integer k. Thus $T_n(1)$ grows as $\Theta(n \cdot r^{-1}(n))$ when considered as a function over integers of the form $n = r(k)$.

Because $T_n(1)$ increases monotonically in n, it follows that $T_n(1)$ grows as $\Theta(n \cdot r^{-1}(n))$ when considered as a function of all positive integers, so long as $(n+1) \cdot r^{-1}(n+1)$ grows as $\Theta(n \cdot r^{-1}(n))$. This last statement is true because $r^{-1}(n)$ grows at most logarithmically [because, by construction, $r(k) \geq 2^k$ for all k], and so we are done.

Theorem 5. *(Bounds on the Introduction Time Function—Generalized.) In the generalized recap schedule,*

$$T_n(1) \leq n \cdot [r^{-1}(n) + 1]$$

and

$$\frac{1}{2} \cdot n \cdot [r^{-1}(n) + 1] \leq T_n(1)$$

for all n.

Proof: In general, by time step $T_n(1)$ only units $u_0, u_1, \ldots, u_{n-1}$ have already occurred at all, by the properties of depth-first postorder traversal, and each at most $r^{-1}(n) + 1$ times. Therefore,

$$T_n(1) \leq n \cdot [r^{-1}(n) + 1].$$

Furthermore, by time step $T_n(1)$, all units with index less than $\frac{1}{2}n$ have occurred exactly $r^{-1}(n) + 1$ times. To see why, consider an arbitrary n and let j represent the left-to-right index of the ancestor of leaf u_n that is at height $r^{-1}(n)$. [Thus, if the ancestor of leaf u_n at height $r^{-1}(n)$ is immediately to the right of the left-most node at that height, then $j = 2$, whereas if it is the right-most sibling of the left-most node at that height, then $j = q(r^{-1}(n) + 1)$. Note that $j \geq 2$ because, as noted earlier, $r^{-1}(n) + 1$ is the height of the lowest ancestor of u_n that is the left-most node at that height.]

By the properties of depth-first postorder traversal, when leaf u_n is visited, all $j-1$ nodes at height $r^{-1}(n)$ to the left of the ancestor of u_n at that height will have been visited already, as will all of the descendants of these $j-1$ nodes. Such leaves will have indices zero through

$$(j-1) \cdot r(r^{-1}(n)) - 1.$$

[Note that by construction, $r(r^{-1}(n))$ is not generally equal to n, but rather to the greatest number m less than n such that $r(k) = m$ for some k.] Thus, at $T_n(1)$, we have that all units with index less than

$$(j-1) \cdot r(r^{-1}(n))$$

have been seen $r^{-1}(n) + 1$ times. Because

$$n < j \cdot r(r^{-1}(n))$$

and $j \geq 2$, it follows that at least $\frac{1}{2} \cdot n$ units have been seen at least $r^{-1}(n) + 1$ times by $T_n(1)$. Thus

$$\frac{1}{2} \cdot n \cdot [r^{-1}(n) + 1] \leq T_n(1).$$

Theorem 6. *(Adherence to Spacing Constraints—Generalized.) The recap schedule adheres to the spacing constraints*

$$a_k = r(k)$$
$$b_k = r(k-1) \cdot (k+1).$$

Proof: Our goal is to show that

$$r(k) \leq T_i(k+1) - T_i(k) \leq r(k-1) \cdot (k+1),$$

for all i, k. We will establish these bounds by calculating the minimum and maximum possible values of $T_i(k+1) - T_i(k)$.

For any k, all the subtrees with roots at height k are identical except for a shift in the labels on the leaves, by construction. Therefore all values of $T_i(k+1) - T_i(k)$ that occur in the context of any given subtree at height k must occur in the context of the subtree rooted at the left-most node at height k. This subtree corresponds to units $u_0, \ldots, u_{r(k)-1}$. Thus, for any k, we only need to consider $i < r(k)$ in order to find the minimum and maximum values of

$$T_i(k+1) - T_i(k).$$

By construction

$$T_j(k+1) - T_i(k+1) = j - i$$

whenever $i < j < r(k)$. Also, because $T_i(k)$ is monotonic in i,

$$T_j(k) - T_i(k) \geq j - i$$

whenever $i < j$. Therefore, for all i and j, such that $i < j < r(k)$, we have that

$$T_j(k+1) - T_j(k) \leq T_i(k+1) - T_i(k).$$

Thus, the maximum value of $T_i(k+1) - T_i(k)$ must occur when $i = 0$ and the minimum value must occur when $i = r(k) - 1$.

Thus if we can show that

$$r(k) \leq T_{r(k)-1}(k+1) - T_{r(k)-1}(k)$$

and that

$$T_0(k+1) - T_0(k) \leq r(k-1) \cdot (k+1)$$

for all k, then we will be done. We will in fact show that we have equality in both cases; we will show that

$$T_{r(k)-1}(k+1) - T_{r(k)-1}(k) = r(k)$$

and

$$T_0(k+1) - T_0(k) = r(k-1) \cdot (k+1)$$

for all k.

By construction, the last entry of the schedule due to the left-most node at height k corresponds to $T_{r(k)-1}(k+1)$, and the last entry of the schedule due to the right-most child of that node corresponds to $T_{r(k)-1}(k)$. Because in a postorder depth-first traversal each node is visited immediately after its right-most child, we have that

$$T_{r(k)-1}(k+1) - T_{r(k)-1}(k) = r(k),$$

corresponding to the $r(k)$ entries of the schedule due to the left-most node at height k.

Meanwhile, $T_0(k+1)$ and $T_0(k)$ refer to the first entry of the schedule due to the left-most nodes at heights k and $k-1$, respectively. Thus $T_0(k+1) - T_0(k)$ will equal the number of entries in the schedule due to the left-most node at height $k-1$, plus the number of entries due to all the subtrees whose roots are siblings of the left-most node at height $k-1$.

The first quantity is $r(k-1)$, by construction. As for the second quantity, because the subtrees in question each correspond to k occurrences of $r(k-1)$ units, we have that the second quantity is equal to $k \cdot r(k-1)$. Thus

$$\begin{aligned} T_0(k+1) - T_0(k) &= r(k-1) + k \cdot r(k-1) \\ &= r(k-1) \cdot (k+1). \end{aligned}$$

The Slow Flashcard Schedule. Here we examine the slow flashcard system in detail. In particular, we show that the slow flashcard schedule adheres to the spacing constraints

$$\begin{aligned} a_k &= k \\ b_k &= k^2. \end{aligned}$$

We also present evidence which suggests that the slow flashcard schedule even adheres to the more stringent constraints

$$\begin{aligned} a_k &= k \\ b_k &= 2k. \end{aligned}$$

We also show that for the slow flashcard schedule, t_n is bounded below by $\Omega(n^2)$ and bounded above by $O(n^3)$, and we present evidence that in fact t_n grows as $\Theta(n^2)$.

We begin by reexamining the construction. We consider an infinite deck of flashcards, indexed by positions 1, 2, 3,… We call position 1 the top or the front of the deck, we say that a flashcard in position i is behind another flashcard in position j if and only if $i > j$. Otherwise, it is in front of the other flashcard. Each flashcard corresponds to an educational unit u_i, and at the beginning of the construction, flashcard u_1 is in position 1, flashcard u_2 is in position 2, etc.

We construct the schedule as follows. At a given time step t, suppose that the flashcard at the top of deck corresponds to educational unit u_i, and that u_i has appeared in the sequence $k-1$ times so far. Then we include u_i in the sequence at time step t (resulting in its kth occurrence), and we move the flashcard containing u_i to position $k+1$ in the deck of flash cards.

Thus the configurations of the deck in the first few time steps are as follows:

$$\begin{array}{l} u_1, u_2, u_3, u_4, u_5, u_6, \ldots \\ u_2, u_1, u_3, u_4, u_5, u_6, \ldots \\ u_1, u_2, u_3, u_4, u_5, u_6, \ldots \\ u_2, u_3, u_1, u_4, u_5, u_6, \ldots \\ u_3, u_1, u_2, u_4, u_5, u_6, \ldots \\ u_1, u_3, u_2, u_4, u_5, u_6, \ldots \\ u_3, u_2, u_4, u_1, u_5, u_6, \ldots \\ u_2, u_4, u_3, u_1, u_5, u_6, \ldots \\ u_4, u_3, u_1, u_2, u_5, u_6, \ldots \\ u_3, u_4, u_1, u_2, u_5, u_6, \ldots \end{array}$$

resulting in the schedule

$$u_1, u_2, u_1, u_2, u_3, u_1, u_3, u_2, u_4, u_3, \ldots,$$

which simply corresponds to the units at the top of the deck (the left entries in the sequences above) at each time step.

As in the last section, we let $T_i(k)$ be the time step of the kth occurrence of unit u_i in the schedule. Thus, for example, here we have that $T_2(3) = 8$, $T_4(1) = 9$, and $T_3(3) = 10$.

Note that, by construction, at every time step, each flashcard except for the one being reinserted either maintains its position or moves up in the deck, decreasing its position by one. The former happens if the presented flashcard is reinserted in front of the flashcard in question, and the latter happens if the presented flashcard is reinserted behind the flashcard in question. We call this the "slow marching property," because informally it says that once a flashcard is inserted into position n, it will "slowly march" to the front of the deck, moving up at a rate of at most one position per time step.

Note also that if u_b is behind u_a in the deck at time step t, then u_b will also be behind u_a at time step $t+1$, unless u_a is in position 1 at time step t, and is reinserted behind u_b at the end of time step t. We call this the "no-passing property."

Now we move on to proving the main results of this section.

Theorem 7. *(Adherence to Spacing Constraints.) The slow flashcard schedule adheres to the spacing constraints*

$$a_k = k$$
$$b_k = k^2.$$

Proof: To prove the theorem, we need to show that

$$n \le T_i(n+1) - T_i(n) \le n^2$$

for all i and n. The left inequality follows from the slow marching property as follows. At $T_i(n)$, the flashcard u_i has been reinserted into position $n+1$ of the deck, by construction. Thus it will be at least n time steps until it is in position 1 by the slow marching property; it will be at least n time steps until $T_i(n+1)$. So

$$T_i(n+1) \ge T_i(n) + n,$$

and from this we get the left inequality.

For the right inequality, again consider the time step $T_i(n)$, where u_i is presented for the nth time and then removed from the deck and reinserted into position $n+1$. Immediately after $T_i(n)$, flashcard u_i is in position $n+1$. Because there is no passing, the part of the schedule in between $T_i(n)$ and $T_i(n+1)$ will consist only of the flashcards that are in front of u_i at $T_i(n)$.

Each time one of these flashcards is presented, it may be reinserted either in front of or behind u_i. Once it has been reinserted behind, it will not be shown again until at least $T_i(n+1)$, again by the no-passing property. Meanwhile, each time it is reinserted, it is reinserted further back in the deck than the previous time it was reinserted, by construction. Thus any flashcard can be presented/reinserted at most n times in between $T_i(n)$ and $T_i(n+1)$, one time for every position less than $n+1$ into which it could be reinserted. So in between $T_i(n)$ and $T_i(n+1)$, the possible reinsertions are limited to each of the n flashcards that are in positions 1, 2,…, n at $T_i(n)$, each being reinserted at most n times. Thus

$$T_i(n+1) - T_i(n) \le n^2.$$

In fact, because flashcards cannot be reinserted into position 1, we have

$$T_i(n+1) - T_i(n) \le n(n-1).$$

In any case, our proof is done.

Theorem 8. *(Asymptotics of the Introduction Time Function.) In the slow flashcard schedule, $T_n(1)$ grows as $\Omega(n^2)$ and $T_n(1)$ grows as $O(n^3)$.*

Proof: We prove this by first showing that

$$T_1(n-1) < T_n(1) < T_1(n)$$

and then showing that $T_1(n)$ grows as $\Omega(n^2)$ and $T_1(n)$ grows as $O(n^3)$.

First, note that

$$T_1(n) < T_i(n)$$

for $i > 1$, for all n. Thus the first flashcard to be inserted into any given position will be the one corresponding to u_1. Thus for any n, flashcard u_n, which began in position n, will remain in position n until flashcard u_1 is reinserted into position n, at $T_1(n-1)$. Only after that can u_n make its way to the front of the deck and be presented for the first time. Thus,

$$T_1(n-1) < T_n(1).$$

At time $T_1(n-1)+1$, flashcard u_1 is right behind flashcard u_n. By the no-passing property, then, we get that

$$T_n(1) < T_1(n).$$

Thus we have that

$$T_1(n-1) < T_n(1) < T_1(n).$$

Now note that, from the theorem above, we have that

$$n \le T_i(n+1) - T_i(n) \le n^2.$$

Thus $T_1(n+1) - T_1(n)$ grows as $\Omega(n)$ and as $O(n^2)$, and so $T_1(n)$ grows as $\Omega(n^2)$ and as $O(n^3)$.

We believe that both results above can be strengthened, and so we finish with two conjectures.

Conjecture 1. *For the slow flashcard schedule, $T_n(1)$ grows as $O(n^2)$, which would imply $T_n(1)$ grows as $\Theta(n^2)$.*

This conjecture is true if and only if $T_{n+1}(1) - T_n(1)$ grows as $O(n)$, and so as evidence for this conjecture, we plot in Fig. S3 $T_{n+1}(1) - T_n(1)$ against n.

Conjecture 2. *The slow flashcard schedule would exhibit infinite perfect learning with respect to spacing constraints with $a_k = k$ and $b_k = 2k$.*

This would be true if and only if

$$n \le T_i(n+1) - T_i(n) \le 2n$$

for all i and n. So as evidence for this conjecture, we plot in Fig. S4 $T_i(n+1) - T_i(n)$ for all i.

Cramming. Here we establish bounds on how much can be crammed in a limited amount of time. Assume that spacing constraints $\{a_k\}$ and $\{b_k\}$ are given, as well as a positive integer T, and suppose there is a cramming sequence of length T that exhibits bounded learning of order n with respect to the given spacing constraints. We will derive an upper bound on n.

By the definition of bounded learning of order n, (*i*) the sequence adheres to the spacing constraints, and (*ii*) the sequence contains at least n distinct educational units such that, if the unit occurs a total of k times in the sequence, then its last occurrence is within b_k positions of the end of the sequence. (To be clear, this is to be interpreted to mean that the last element in the sequence is defined to be one position from the end of the sequence, not zero.)

Assume, without loss of generality, that these n units are labeled in reverse order of their last occurrences in the sequence. Thus unit u_1 is the last unit to appear in the sequence. Unit u_2 occurs for the last time before unit u_1 occurs for the last time, and so u_2 occurs for the last time at time step $t = T - 1$ at the latest. In general, for each i, unit u_i must appear for the last time at time step $t \le T - i + 1$ at the latest—that is, at least i time steps from the end of the sequence.

Let $m(i)$ denote the smallest number k such that $b_k \ge i$. Then, for every i, unit u_i must occur at least $m(i)$ times in the sequence, because otherwise the sequence would not satisfy part (*ii*) of the definition of bounded learning.

Because each of the n units must occur at least $m(i)$ times in the sequence, where i represents the label of the educational unit, and because each time step can afford at most one occurrence of one educational unit, we have that

$$\sum_{i=1}^{n} m(i) \le T.$$

This represents an upper bound on n, because n must be such that this inequality holds true. [Note that the function $m(i)$ depends implicitly on the numbers in $\{b_k\}$.]

Now consider just unit u_n. When it occurs last, it is for at least the $m(n)$th time. Because the spacing constraints must have been adhered to with respect to u_n, it follows that the $m(n)$th occurrence of u_n must occur after a minimum of

$$\sum_{j=1}^{m(n)-1} a_j$$

time steps. And because it can occur no later than n time steps from the end of the sequence (that is, at time step $t = T - n + 1$), we have another statement on the minimum possible length of the sequence. Namely,

$$\left(\sum_{j=1}^{m(n)-1} a_j\right) + n \le T.$$

Thus we have two inequalities, each of which represents an upper bound on n. In the language of scheduling theory, the first inequality represents a "volume bound," assuring that there is enough time for every unit to be seen as many times as it needs to be seen, and the second inequality represents a "path bound," assuring that the sequence is long enough to allow for even the unit which requires the longest time from the first occurrence to the end of the sequence.

Together the bounds incorporate the spacing constraints as well as the given amount of time. Nevertheless, for a given set of spacing constraints and a given T, the actual maximal n (that is, the maximal n such that a sequence of length T can exhibit bounded learning of order n with respect to the given spacing constraints) could be lower than the lower of these two upper bounds. This is because the bounds do not address the actual construction of cramming sequences, which appears in general to be a difficult scheduling problem that hinges on the particulars of the spacing constraints. How to design general and efficient algorithms for constructing sequences which provably maximize cramming, so to speak, remains an open problem.

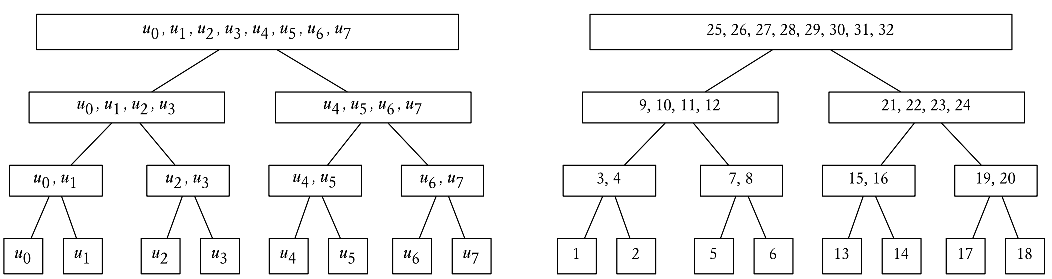

Fig. S1. The full binary tree on the left has each node labeled with the corresponding educational units in the construction of the recap schedule. The tree on the right is identical, except the nodes are labeled with the corresponding time steps. The corresponding schedule, up to and including the left-most node at height $k = 2$, is $u_0, u_1, u_0, u_1, u_2, u_3, u_2, u_3, u_0, u_1, u_2, u_3, \ldots$

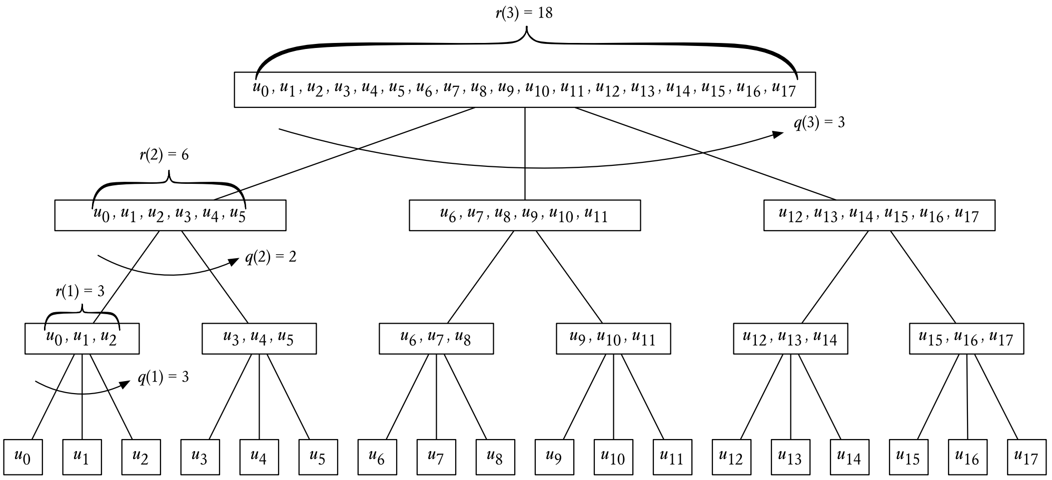

Fig. S2. A tree made using $q(1) = 3$, $q(2) = 2$, and $q(3) = 3$, with each node labeled with the corresponding educational units in the construction of the general recap schedule. The corresponding schedule, up to and including the left-most node at height $k = 2$, is $u_0, u_1, u_2, u_0, u_1, u_2, u_3, u_4, u_5, u_3, u_4, u_5, u_0, u_1, u_2, u_3, u_4, u_5 \ldots$

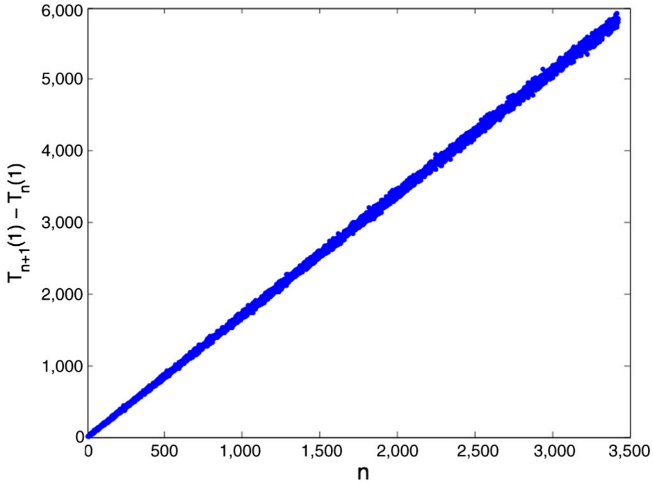

Fig. S3. This figure shows $T_{n+1}(1) - T_n(1)$ plotted against n. The data are taken from the first 1 million time steps of the slow flashcard schedule. A linear regression gives a line with slope 1.7, with a correlation coefficient of $r > 0.9997$.

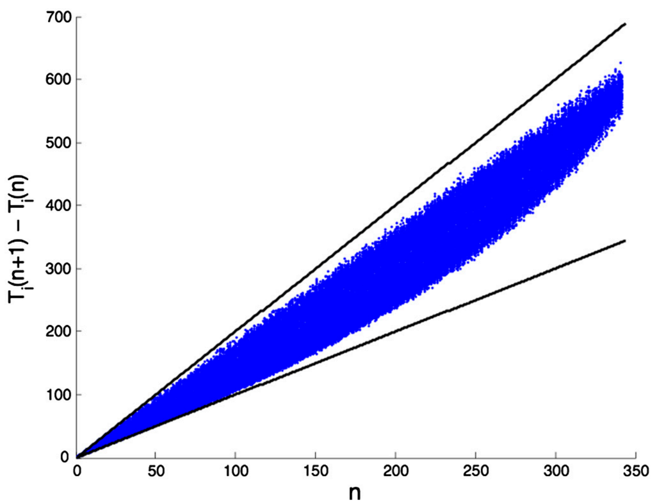

Fig. S4. This figure shows $T_i(n+1) - T_i(n)$ plotted against n, for all i for which data were collected. The data are taken from the first 100,000 time steps of the slow flashcard schedule. Also shown are the lines going through the origin with slopes 1 and 2. All data points lie between the two lines.