Education of a model student

Timothy P. Novikoff[a], Jon M. Kleinberg[b,1], and Steven H. Strogatz[a]

[a]Center for Applied Mathematics, and [b]Department of Computer Science, Cornell University, Ithaca, NY 14853

Edited by Charles S. Peskin, New York University, New York, NY, and approved December 12, 2011 (received for review July 3, 2011)

A dilemma faced by teachers, and increasingly by designers of educational software, is the trade-off between teaching new material and reviewing what has already been taught. Complicating matters, review is useful only if it is neither too soon nor too late. Moreover, different students need to review at different rates. We present a mathematical model that captures these issues in idealized form. The student's needs are modeled as constraints on the schedule according to which educational material and review are spaced over time. Our results include algorithms to construct schedules that adhere to various spacing constraints, and bounds on the rate at which new material can be introduced under these schedules.

asymptotic analysis | scheduling

In his 2009 speech to the National Academy of Sciences, President Barack Obama exhorted the audience to imagine some of the things that could be made possible by a commitment to scientific research, including the invention of "learning software as effective as a private tutor" (1). This paper is a modest step in that direction.

An important challenge for the design of educational software is to incorporate the results of empirical research on how people learn. Such research endeavors to provide principles for how to choose what is taught, how to present it, and how to sequence the material. Ultimately, educational software will require mechanisms for managing the constraints that arise when these principles are applied in different settings.

Here we develop and analyze an idealized mathematical model for incorporating a fundamental class of such constraints into educational software—constraints arising from the importance of timing and review in the presentation of new material. For example, software for building vocabulary must determine when to introduce new words that the student has not yet learned, and when to review words whose definitions the student has successfully recalled in the past. The issue is similar to that faced by a high school math teacher deciding how often to schedule lessons that involve trigonometry, or by a piano teacher who needs to decide how much time a student should devote to practicing scales while also learning to play new pieces.

The study of the importance of timing with respect to review dates to at least 1885 (2). The notion that it is better to spread studying over time instead of doing it all at once is called the spacing effect. In 1988, Dempster noted that "the spacing effect is one of the most studied phenomena in the 100-year history of learning research" (3). As Balota et al. point out in their review (4), "spacing effects occur across domains (e.g., learning perceptual motor tasks vs. learning lists of words), across species (e.g., rats, pigeons, and humans), across age groups and individuals with different memory impairments, and across retention intervals of seconds to months." See refs. 5 and 6 for reviews. For further results and background, see refs. 7–9. For work on exploiting the spacing effect to build vocabulary (in humans), see refs. 10–13.

Review is an important part of the learning process, but the extent to which review is needed varies by student. Some students need to review early and often, whereas others can learn a lot without any review at all. Personalized educational software of the future could fit a model to the user and then schedule review in a way that is tailored to the model.

With such educational software in mind, we envision a system in which the software designer can specify a schedule for the introduction of new material, together with a schedule by which the review of existing material is spaced over time. What we find, however, is that the resulting scheduling problems are mathematically subtle: Existing techniques do not handle scheduling problems with spacing constraints of this type.

Our main contribution is to develop an approach for reasoning about the feasibility of schedules under spacing constraints. We begin by introducing a stylized mathematical model for the constraints that arise from the spacing effect, and then consider the design of schedules that incorporate these constraints.

Models

Roughly speaking, we model the introduction and review of material as a sequence of abstract educational units, and we model the needs of the student using two sequences, $\{a_k\}$ and $\{b_k\}$: After an educational unit has been introduced, the "ideal" time for the student to see it for the $(k+1)^{\text{st}}$ time is between a_k and b_k time steps after seeing it for the kth time. We also model various educational outcomes that the designer of educational software may seek to achieve. The details are given below.

The Educational Process. We imagine the underlying educational software as implementing a process that presents a sequence of abstract educational units which could represent facts such as the definitions of vocabulary words, concepts such as trigonometric identities, or skills such as playing a scale on the piano. For example, the sequence $u_1, u_2, u_3, u_1, u_4, \ldots$, indicates that educational unit u_1 was introduced at the first time step and reviewed at the fourth time step. This sequence defines the schedule according to which the units will be presented. We also allow our schedules to contain blanks, or time steps in which no educational unit is presented; thus, an arbitrary schedule will have each entry equal to either an educational unit or a blank.

This model is highly idealized. It ignores possible relationships between units, such as the etymological (and potentially pedagogically useful) connection between the vocabulary words "neophyte" and "neologism," for example, or the dependence of trigonometry on more basic concepts in geometry. It also treats all units as equal. Thus it does not capture, for example, that an experienced pianist may benefit more from practicing a scale than practicing Twinkle Twinkle Little Star.

The real-life educational process is nuanced, complex, and context dependent. Future work in building models for educational software may introduce more complexity to this model, or simply reduce scope and model more specific situations. Here,

Author contributions: T.P.N., J.M.K., and S.H.S. designed research, performed research, analyzed data, and wrote the paper.

Conflict of interest statement: T.P.N. is the owner of a small, wholly owned company, Flash of Genius, LLC, that sells educational software.

This article is a PNAS Direct Submission.

Freely available online through the PNAS open access option.

[1]To whom correspondence should be addressed. E-mail: kleinber@cs.cornell.edu.

This article contains supporting information online at www.pnas.org/lookup/suppl/doi:10.1073/pnas.1109863109/-/DCSupplemental.

in the interest of generality as well as mathematical tractability, we use this very simple model of the general educational process in order to create a formalism that captures spacing constraints and their role in reviewing material after it has been introduced.

Spacing Constraints. A large body of empirical work in learning research has studied the expanding nature of optimal review. For example, when students first learn a new vocabulary word, they must review it soon or else they likely lose the ability to correctly recall the definition. If they do review it before forgetting it, they will then generally be able to go longer than before without needing review. By repeatedly reviewing, the student "builds up" the ability to go longer and longer without seeing the word while maintaining the ability to recall its definition. However, reviewing a word too soon after studying it can reduce the benefit of the review. These are the principles of the well-established theory of expanded retrieval; see ref. 4 for a review specifically on expanded retrieval. More generally, see refs. 3–13.

We want a simple formalism that captures the need to review an educational unit on a schedule that "expands" the spacing between successive viewings. We wish to leave the exact rate of expansion under the control of the software designer, motivated by the goal of creating different schedules for different students. We thus imagine that the designer of the software specifies a set of spacing constraints, consisting of an infinite sequence of ordered pairs (a_1,b_1), (a_2,b_2),…,(a_k,b_k),…, where $a_k \leq b_k$ are positive integers for all k. Intuitively, the idea is that, for each educational unit u_i in the schedule, the designer wants the gap in the schedule between the kth and $(k+1)^{\text{st}}$ occurrences of u_i to have length in the closed interval $[a_k,b_k]$. The fact that a student can go longer between occurrences as they gain familiarity with the educational unit is represented by the assumption that the numbers a_k and b_k are weakly increasing: We impose the requirement that $a_k \leq a_{k+1}$ and $b_k \leq b_{k+1}$ for all k.

Thus, our key definition is that a schedule satisfies a set of spacing constraints if, for each u_i in the schedule, the $(k+1)^{\text{st}}$ occurrence of u_i in the schedule comes between a_k and b_k positions (inclusive) after the kth occurrence. Roughly speaking, the numbers b_k model how long the student can retain learned material. The numbers a_k model how long a student should wait before review is beneficial, capturing the notion that there is an ideal time to review. If review is done too early, it is less beneficial. If it is done too late, the student will forget the material in the interim.

This model is, by design, a simplification of spacing constraints and their role in learning. A more nuanced model might allow for "blurry" boundaries for the intervals $[a_k,b_k]$, in which there is a numerical penalty for missing the interval by a small amount. Another refinement would be to allow the model to discriminate between educational units to reflect that some things are easier to learn than others. However, our simple model captures the essence of the phenomenon we are investigating, and will allow for mathematical analysis that elucidates the mechanics of scheduling review in an optimal way.

Educational Goals. We consider two natural goals for the designer of the educational software. The mission of the software could be to teach students in such a way that they grow their knowledge without bound, never forgetting anything along the way; a sort of "lifelong learning" approach to education. Alternatively, the mission could simply be to get students familiar with a certain set of educational units by a particular point in time, regardless of whether they are destined to forget what they learned quickly thereafter; something like the studying technique known as "cramming." We address both goals in this paper.

We model the first goal by saying that a schedule exhibits infinite perfect learning with respect to some spacing constraints if (*i*) it satisfies the spacing constraints, and (*ii*) it contains infinitely many educational units, each of which occurs infinitely often. Thus if the constraints represented the needs of a student, then with such a schedule the student would, over the course of the infinite sequence, learn an infinite set of educational units without ever forgetting anything.

For the second goal we consider a finite sequence, representing the presentation of material up to a test or performance. For a positive integer n, we say that the sequence is a cramming sequence, and exhibits bounded learning of order n, if (*i*) it satisfies the spacing constraints, and (*ii*) it contains at least n distinct educational units such that, if a 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. Condition (*ii*) captures the requirement that the student should still be able to recall all of the n educational units at the end of the sequence, which was the whole point of cramming. Note that, although the spacing constraints have been respected up to the end of the finite sequence, there is no guarantee that the sequence could be extended while continuing to satisfy the spacing constraints.

With respect to infinite perfect learning, we will be interested in the rate at which the student would learn if taught according to the schedule. To this end, we define the introduction time function: For a given schedule of educational units, let t_n denote the position in the schedule of the first occurrence of the nth distinct educational unit. Thus the slower the growth of t_n, the faster new educational units are being introduced.

We will be considering questions of the following nature. Given spacing constraints $\{(a_k,b_k)\}$, does there exist a schedule that exhibits infinite perfect learning, or bounded learning, with respect to the constraints? If so, how can we construct such schedules? And when such a schedule exists, what rate of learning (as measured by the sequence $t_1,t_2,\ldots$) is achievable? These are fundamental problems that would be faced by an educational software designer seeking to incorporate spacing constraints in the design of the underlying algorithms. As we will see, despite the simply stated formulation of these questions, the combinatorial challenges that they lead to quickly become quite intricate.

Results

The spacing constraints are described by two infinite sequences of parameters, $\{a_k\}$ and $\{b_k\}$. In this section, we describe how choices for these parameters affect the rate at which new educational units can be introduced into the schedule and how schedules can be tailored to particular parameter regimes and educational goals.

Overview of Results. We begin by examining the first main issue of this paper: the trade-off between (*i*) the rate at which b_k grows as a function of k and (*ii*) the rate at which t_n grows as a function of n. Informally, if b_k grows relatively slowly, then a lot of time must be spent on review rather than on introducing new units, and hence t_n must grow more quickly, corresponding to slower learning. It is clear that $t_n \geq n$ for any schedule, because even without review we can only introduce one educational unit per time step. With these considerations in mind, we investigate the following pair of questions that explore the two sides of the trade-off. First, is there a set of spacing constraints for which t_n is close to this trivial bound, growing nearly linearly in n? Second, as we require b_k to grow slower as a function of k, it becomes more difficult to achieve infinite perfect learning. Is there a set of spacing constraints for which infinite perfect learning is possible, and for which b_k grows as a polynomial function of k? We answer both of these questions in the affirmative.

In *The Recap Method*, we describe a set of spacing constraints for which a schedule can be constructed that exhibits infinite perfect learning, and where the rate of learning is relatively quick. In the schedule we construct, t_n grows as $\Theta(n \log n)$, and in fact $t_n \leq n \cdot (\lfloor \log_2 n \rfloor + 1)$. Recalling that we must have $t_n \geq n$ for all

n for any schedule, we see that the recap schedule achieves infinite perfect learning with only a modest increase on this bound—that is, with a relatively small amount of review. In the *SI Text*, we show how to construct a schedule where t_n grows at a rate that, in some sense, can be as close to linear as desired. In *Superlinearity of the Introduction Time Function*, we show that there can be no schedule whatsoever such that t_n grows as $O(n)$.

In *The Slow Flashcard Schedule*, we show a set of spacing constraints for which a_k and b_k grow polynomially in k and for which infinite perfect learning is possible. With these spacing constraints based on much smaller a_k and b_k, the schedule we construct has a slower rate of learning; t_n is bounded below by $\Omega(n^2)$ and bounded above by $O(n^3)$, in contrast to the schedule from *The Recap Method* for which the introduction time function grows as $\Theta(n \log n)$. The gap between the quadratic lower bound and the cubic upper bound is an interesting open question; we give numerical evidence that in fact the lower bound may be tight, and that t_n grows as $O(n^2)$. Much of this analysis is done in the *SI Text*.

Following these results, we turn to the second main issue in this paper, which is to identify general possibility and impossibility results for satisfying classes of spacing constraints. We first show, in *Flexible Students*, that the difficulty in achieving infinite perfect learning stems, in a sense, from the fact that the numbers a_k are growing: Specifically, we show that for *any* spacing constraints in which $a_k = 1$ and $b_k \to \infty$, it is possible to construct a schedule that exhibits infinite perfect learning. The construction introduced here demonstrates a general method that can be adapted to many sets of spacing constraints with $a_k > 1$ as well.

Thus far, we have only considered spacing constraints that allow for the construction of schedules that exhibit infinite perfect learning. In *The Finicky Slow Student*, we show that there exist spacing constraints for which no schedule can exhibit infinite perfect learning. In particular, we build this impossibility result from an extreme case, where $a_k = b_k = f(k)$ and $f(k)$ is a function that grows slowly in k. These constraints represent a setting in which the student insists on reviewing material on an extremely precise and plodding schedule, with no room for error.

The difficulty in constructing a schedule for such a set of constraints is that as the knowledge base—the number of educational units introduced—grows, so does the need to review, and so the potential for scheduling conflicts increases. The slower the student [the slower the growth of $f(k)$], the fewer educational units can be put on the back burner, so to speak, while the student focuses on new units. The more finicky the student (the smaller the windows $b_k - a_k$), the less wiggle room there is in scheduling review.

All of these points match with intuition. Students who don't need to review much and aren't too picky about when the review needs to happen can be taught a lot, and fast. But students who need a lot of review and who only derive benefit from very well-timed review will be more difficult to teach. The educational mantra is "every child can learn," but designers of personalized educational software may find that scheduling the educational process for some students is, at the least, more difficult for some than for others.

In *Cramming*, we show that every student can cram. More precisely, we show that, for any set of spacing constraints and any n, it is possible to construct a finite sequence that exhibits bounded learning of order n. Consistent with the discussion that accompanied the definition of bounded learning earlier, the construction assures nothing about whether the sequence can be extended beyond this moment of "expertise" at its end.

Finally, also in *Cramming*, we explore the question of how much can be crammed in a given amount of time. Given a set of spacing constraints and a finite number T, we derive a nontrivial upper bound on those n for which bounded learning of order n is possible using only T time steps.

The Recap Method. Here we explore spacing constraints that allow for infinite perfect learning with a rapid learning rate—that is, where the introduction time function, t_n, grows slowly.

Consider the spacing constraints $a_k = 2^k$ and $b_k = 2^{k-1}(k+1)$. A schedule that allows for infinite perfect learning with respect to these spacing constraints can be described as follows. To find the first $(k+1) \cdot 2^k$ entries of the schedule, consider a depth-first postorder traversal of a full binary tree of depth k with 2^k leaves labeled $u_1, u_2, \ldots, u_{2^k}$ from left to right. (A depth-first postorder traversal of a tree is a particular order for visiting the nodes of a tree, defined as follows. Starting at the root v of the tree, the depth-first postorder traversal is first applied recursively to each subtree below v one at a time; after all these traversals are done, then the root v is declared to be visited.) 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, append the sequence with the units corresponding to all of the descendant leaves, in left-to-right order. The resulting sequence gives the first $(k+1) \cdot 2^k$ entries in the schedule.

Thus, using $k = 2$ we have that the first 12 entries of the schedule are $u_1, u_2, u_1, u_2, u_3, u_4, u_3, u_4, u_1, u_2, u_3, u_4$. We call this schedule "the recap schedule" because it incorporates periodic review of everything that has been learned so far, like a teacher saying "okay, let's recap."

In this schedule, the number of time steps between the kth and $(k+1)^{\text{st}}$ occurrence of any particular unit is always between 2^k and $2^{k-1}(k+1)$, with both bounds actually achieved for each k. This fact—along with the fact that infinitely many units occur infinitely often due to properties of depth-first traversals—establishes that the recap schedule allows for infinite perfect learning with respect to the given spacing constraints. Calculations which establish these facts are shown in the *SI Text*.

Further calculations, also shown in the *SI Text*, establish that t_n grows as $\Theta(n \log n)$ in this schedule. More precisely,

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

By generalizing the construction of the schedule, using a more general class of trees, we can show that, for a large class of functions $r(n)$, schedules can be constructed that exhibit infinite perfect learning for which t_n grows as $\Theta(n \cdot r^{-1}(n))$. The class includes functions $r(n)$ that grow arbitrarily fast, and so in that sense we can create schedules for which t_n grows at a rate that is as close to linear as desired. The downside of these schedules is that they require increasingly lax spacing constraints as the growth rate of t_n approaches linearity: The schedules that we construct for which t_n grows as $\Theta(n \cdot r^{-1}(n))$ require b_k, as well as $b_k - a_k$, to grow as $\Theta(k \cdot r(k))$. Calculations which establish these facts, too, are shown in the *SI Text*.

Superlinearity of the Introduction Time Function. Although our constructions are able to achieve introduction times t_n that grow arbitrarily close to linearly in n, we can also show that an actual linear rate of growth is not achievable: For schedules that exhibit infinite perfect learning, the introduction time function t_n must be superlinear. More precisely, we show that, for any schedule that exhibits infinite perfect learning with respect to any spacing constraints $\{(a_k, b_k)\}$, there cannot be a constant c such that $t_n \leq cn$ for all n.

To prove this statement, we consider an arbitrary set of spacing constraints $\{(a_k, b_k)\}$ and an arbitrary schedule that exhibits infinite perfect learning with respect to these constraints, and assume for the sake of contradiction that there is a constant c such that $t_n \leq cn$ for all n. Let $\hat{b}_k = \sum_{j=1}^{k} b_j$ and let n_0 be any integer such that $n_0 > \hat{b}_{c+1}$. By our assumption, at least n_0 educational units have been introduced by the time step cn_0. In general, for any

schedule that exhibits infinite perfect learning, any unit that has been introduced by time step t will have occurred k times by time step $t + \hat{b}_k$, by the definition of $\hat{b}_k$. Thus at least n_0 units will have occurred $c + 1$ times by time step $cn_0 + \hat{b}_{c+1}$. So $cn_0 + \hat{b}_{c+1} \geq (c + 1)n_0$ because each time step corresponds to at most one educational unit. Subtracting cn_0 from both sides, we have that $\hat{b}_{c+1} \geq n_0$, which contradicts our choice of n_0. Thus, there cannot be a constant c such that $t_n \leq cn$ for all n.

The Slow Flashcard Schedule. One can describe the construction of the recap schedule without using depth-first traversals or full binary trees, but rather using a deck of flashcards. Doing so will shed some light on the recap schedule, and will also serve as a useful jumping-off point for discussing the very different schedule which is the focus of this section. So we begin our discussion here by revisiting the recap schedule.

Imagine a deck of flashcards, with the kth card corresponding to an educational unit u_k. Thus the top card corresponds to u_1, the next card corresponds to u_2, etc. Then we construct a schedule as follows. At every step, we present to the student the educational unit corresponding to the top card, and then reinsert the card into position 2^k, where k is the number of times we have presented the educational unit corresponding to that card, up to and including this latest time step. Thus first we present u_1, then we remove it from the deck and reinsert it into position 2 in the deck. Then we present u_2, then remove it and reinsert it into position 2 in the deck. Then u_1 is again on top and so we present it for a second time and then remove it and reinsert it into position $2^2 = 4$. This process produces the recap schedule.

In this section, we consider a schedule that is much more difficult to describe explicitly than the recap schedule, but whose construction can similarly be described in terms of a deck of flashcards. Instead of reinserting into position 2^k as above, though, we reinsert into position $k + 1$. Carefully applying this rule shows the first few entries of the schedule to be $u_1,u_2,u_1,u_2,u_3,u_1,u_3,u_2,u_4,u_3$. Of all schedules that can be constructed with a similar flashcard-reinsertion scheme using some strictly increasing reinsertion function $r(k)$ with $r(1) > 1$, it is this schedule, constructed using $r(k) = k + 1$, which progresses through the deck the slowest. For this reason, we call this schedule "the slow flashcard schedule."

In the *SI Text*, we show that the slow flashcard schedule exhibits infinite perfect learning with respect to the spacing constraints with $(a_k,b_k) = (k,k^2)$. Thus the slow flashcard schedule provides a dramatic alternative to the recap schedule. Whereas b_k and $b_k - a_k$ both grew exponentially in k in the recap schedule, here they grow quadratically and yet still allow for infinite perfect learning.

Numerical simulations shown in the *SI Text* suggest that this schedule in fact exhibits infinite perfect learning even with respect to the much tighter spacing constraints with $(a_k,b_k) = (k,2k)$. If that is the case, the contrast with the recap schedule would be even more stark.

The trade-off for this slow growth in b_k is the speed at which the knowledge base grows. Whereas t_n, the time needed for the knowledge base to achieve size n, grew as $\Theta(n \log n)$ in the recap schedule, here it is bounded below by $\Omega(n^2)$. A proof of this fact can be found in the *SI Text*, along with numerical evidence that it in fact grows as $\Theta(n^2)$.

The spacing constraints in *The Recap Method* and *The Slow Flashcard Schedule* are tailored to allow for existence proofs that certain bounds on t_n, b_k, and $b_k - a_k$ can be achieved in the context of infinite perfect learning. The methods used to describe the schedules, though, suggest general principles for how to construct schedules with desirable properties. Moreover, we note that the schedules constructed are relevant to any set of spacing constraints that are more relaxed than the given ones: If a schedule exhibits infinite perfect learning with respect to spacing constraints $\{(a_k,b_k)\}$, then it also exhibits infinite perfect learning with respect to $\{(a'_k,b'_k)\}$ when $a'_k \leq a_k$ and $b'_k \geq b_k$ for all k.

In the next section, *Flexible Students*, we begin with a general class of students and build schedules tailored to each individual student in the class.

Flexible Students. What if the student did not need to wait at all in order to derive benefit from studying? In other words, what if $a_k = 1$ for all k? In this case, we can use a technique for constructing schedules that we call "hold-build": sequencing the educational units that are known to the student in a "holding pattern" so that they meet the spacing constraints, while showing new educational units in quick repetition (thereby "building" them up). The only assumptions that are needed for the construction, besides $a_k \equiv 1$, are that $b_1 \geq 2$ and that $b_k \to \infty$. (Note that b_k is already required to be weakly increasing.)

We define the sequence HB_m to be the infinite sequence that starts with u_m, contains u_m in every other entry, and cycles through units $u_1,\ldots,u_{m-1}$ in the remaining entries. So, for example,

$$HB_2 = u_2,u_1,u_2,u_1,u_2,u_1,\ldots$$
$$HB_3 = u_3,u_1,u_3,u_2,u_3,u_1,u_3,u_2,u_3,u_1,\ldots$$
$$HB_4 = u_4,u_1,u_4,u_2,u_4,u_3,u_4,u_1,u_4,u_2,u_4,\ldots$$

Now, consider an arbitrary set of spacing constraints such that $a_k \equiv 1$, $b_1 \geq 2$, and $b_k \to \infty$. For ease of discussion, we say that an educational unit is at level k when it has been shown exactly k times. We construct the schedule that assures infinite perfect learning as follows.

First, present unit u_1. Then present units according to HB_2. So far so good; so long as units are presented according to HB_2, we know the spacing constraints will be met, because $b_k \geq 2$ for all k. Meanwhile, the levels for u_1 and u_2 can be built up without bound.

Continue HB_2 as long as necessary until it is feasible to move on to HB_3. In other words, show enough of HB_2 so that if the schedule up to that point were followed by an indefinite run of HB_3, then the spacing constraints would be met. This condition is guaranteed to be true after a finite number of time steps because $b_k \to \infty$ and HB_2 is periodic.

Thus, once enough of HB_2 has been shown, we show as much of HB_3 as necessary until we can afford to move on to HB_4. Then we show as much of HB_4 as necessary until we can afford to move on to HB_5, and we continue building the schedule like this indefinitely.

The schedule formed by concatenating hold-build patterns in this way assures infinite perfect learning, and it applies to any set of spacing constraints with $a_k \equiv 1$, $b_1 \geq 2$, and $b_k \to \infty$. Thus we actually have a class of spacing constraints where infinite perfect learning is possible and yet b_k can grow arbitrarily slowly. The trade-off for an exceedingly slow-growing b_k will again be a fast-growing t_n, corresponding to a slow rate of learning. The exact rate will depend on the exact rate of growth of b_k.

To give a concrete example of this hold-build construction and an accompanying calculation of t_n, we can consider the simple case where $b_k = k + 1$. (Note that, because we only require that b_k be weakly increasing, there are much slower-growing choices for b_k than this one.) Then, by carrying out the construction above, we have that the sequence is

$$u_1,\underbrace{u_2,u_1,u_2,u_1,u_2}_{HB_2},\underbrace{u_3,u_1,u_3,u_2,u_3,u_1,u_3,u_2,u_3}_{HB_3},\underbrace{u_4,u_1,\ldots}_{HB_4}$$

with $4k-3$ time steps spent in HB_k for every k. Thus, because u_i will be introduced one time step after finishing the HB_{i-1} part of the schedule, we have that

$$t_n = 1 + \left(\sum_{k=2}^{n-1}(4k-3)\right) + 1 = 2n^2 - 5n + 4.$$

The idea behind the hold-build construction, namely the method of putting some units in a holding pattern while others are being "built up," could readily be used to construct schedules assuring infinite perfect learning for many sets of spacing constraints with $a_k > 1$ as well (or spacing constraints with $b_1 = 1$, for that matter). It is a tool that can be used to tailor educational processes to model students in general.

The Finicky Slow Student. We now give a set of spacing constraints $\{(a_k,b_k)\}$ for which no schedule can exhibit infinite perfect learning. They are simply the constraints defined by $a_k = b_k = k$. We call this set of constraints "the finicky slow student" because $b_k - a_k$ is so small, and because b_k grows so slowly as a function of k. We show that no schedule can exhibit infinite perfect learning with respect to the finicky slow student.

Suppose, for the sake of contradiction, that there were a schedule that exhibited infinite perfect learning with respect to the finicky slow student. We say that a schedule incorporates educational unit u_i if the unit occurs infinitely many times, and if the sequence of occurrences satisfies the spacing constraints. Thus, given the particulars of the finicky slow student, if a unit u_i is incorporated, and if it first occurs at step τ, then it must also occur at steps $\tau+1$, $\tau+3$, $\tau+6$, $\tau+10$,….

By assumption, the schedule incorporates infinitely many educational units. We can assume, without loss of generality, that educational unit u_1 is incorporated and that its first occurrence is at time step $\tau_0 = 0$. (Letting time start at zero here allows for cleaner calculations.) Then we know that u_1 also occurs at precisely the steps $\tau_1, \tau_2, \tau_3,\ldots$, where $\tau_i = \sum_{k=1}^{i} a_k$. It will be sufficient to show that no other unit can be incorporated without creating a scheduling conflict—in other words, without needing to eventually be scheduled at a step of the form τ_i.

Suppose another unit, call it u_2, were incorporated, with its first occurrence at step s_0. Then we know that u_2 must also occur at precisely the steps $s_1, s_2, s_3,\ldots$, where $s_i = s_0 + \sum_{k=1}^{i} a_k$.

We show that there must be some step common to both sequences $\{s_i\}$ and $\{\tau_i\}$. Thus we will have a contradiction because, at most, one educational unit can appear in each entry of the schedule.

We begin by noting that $s_i - \tau_i = s_0$ for all i, and that $s_{i+1} - s_i = \tau_{i+1} - \tau_i = a_{i+1}$ for all i. Now choose k large enough so that $\tau_{k+1} - \tau_k > s_0$. Then $\tau_{k+1} > \tau_k + s_0 = s_k$. Thus for sufficiently large k, we have that $\tau_{k+1} > s_k$. Now let m be the smallest number such that $\tau_{m+1} > s_m$. We know $m \geq 1$, because $\tau_0 = 0$ and $\tau_1 = 1$ by construction. We claim that $\tau_m = s_{m-1}$.

If $\tau_m > s_{m-1}$, then m would not be the smallest number such that $\tau_{m+1} > s_m$ (because then $m-1$ would also qualify), so $\tau_m \not> s_{m-1}$.

If $\tau_m < s_{m-1}$, then we have that $\tau_m < s_{m-1} < s_m < \tau_{m+1}$, which implies that $s_m - s_{m-1} \leq \tau_{m+1} - \tau_m - 2$, because all s_i and τ_i are integer valued. Thus $a_m \leq a_{m+1} - 2$, so $a_{m+1} - a_m \geq 2$, which is not possible because $a_{k+1} - a_k = 1$ for all k. So $\tau_m \not< s_{m-1}$.

Thus we have that $\tau_m = s_{m-1}$, which is a contradiction, of course, because only one educational unit can be scheduled for any given time step. Thus no schedule can exhibit infinite perfect learning with respect to the finicky slow student; in fact, the finicky slow student does not even allow for the incorporation of more than one educational unit.

This proof holds not only for the spacing constraints $a_k = b_k = k$, but for any spacing constraints such that $a_k = b_k = f(k)$, where $f(k)$ is an integer sequence such that $f(1) = 1$, $f(k+1) - f(k) \in \{0,1\}$, and $f(k) \to \infty$. The exact choice doesn't matter; the finickiness ($a_k = b_k$) and the slowness [$f(1) = 1$ and $f(k+1) - f(k) \in \{0,1\}$] are sufficient to carry out the proof as written, but with the final argument using $a_{k+1} - a_k \leq 1$ instead of $a_{k+1} - a_k = 1$.

Cramming. The focus up until now has been on infinite perfect learning, but there could be less ambitious goals for a student. We turn our attention now to cramming. At the end of this section, we address the question of how much cramming can be done in a given amount of time. We begin here with a positive result, showing that for every positive integer n and every set of spacing constraints with $b_k \to \infty$, there exists a sequence that achieves bounded learning of order n.

We consider an arbitrary set of spacing constraints with $b_k \to \infty$ and proceed by induction on n. It is clear that bounded learning is possible for $n = 1$; the sequence consisting simply of u_1 satisfies the definition.

Now, let S_n be a sequence of length T_n that achieves bounded learning of order n. To complete the induction, we construct a new sequence, S_{n+1} of length T_{n+1}, that achieves bounded learning of order $n+1$.

Recall from *Flexible Students* that the level of an educational unit at time t in a sequence is the number of times it has appeared prior to time t. The basic idea behind the construction of S_{n+1} is to start by building up the level of u_1 until it is at a level m such that $b_m > T_n$. Then we use the next T_n steps to present units u_2, $u_3,\ldots,u_{n+1}$ according to the sequence S_n. When that is done, the time limit of b_m has still not been reached for u_1, and hence the sequence satisfies the definition of bounded learning of order $n+1$.

Formally, let m be the smallest integer such that $b_m > T_n$. Then present unit u_1 at time $t = 0$ and at times $t = \sum_{i=1}^{j} a_i$ for $j = 1,2,\ldots,m-1$. Present a blank in the sequence at every other time step in between. Then, starting at time $t = 1 + \sum_{i=1}^{m-1} a_i$, present units $u_2,u_3,\ldots,u_{n+1}$ according to the T_n elements of the sequence S_n. This sequence, through time $T_{n+1} = T_n + \sum_{i=1}^{m-1} a_i$, is our new sequence S_{n+1}. By construction it satisfies the conditions of bounded learning of order $n+1$. By induction, then, bounded learning of order n is possible for all positive integers n for any set of spacing constraints with $b_k \to \infty$.

In the construction above, it is entirely possible that one or more units would begin to violate the spacing constraints even one time step later. Little is assured other than the educational units having met the scheduling constraints up to a certain time step. We call this sort of construction cramming because it presents the material with a particular target time in view and without regard to the scheduling of material after this target time, like a student cramming for a final exam who doesn't worry about how much will be retained after the test.

Condition (*ii*) of our definition of bounded learning models the notion of studying up to a point in time and then being able to remember everything that was studied for at least one more time step, as if there were a quiz lasting one time step which would occur in the time step immediately following the cramming sequence. We could similarly model the notion of a quiz that lasts d time steps by requiring that if a unit's last occurrence is s time steps from the end of the sequence, and the unit occurs a total of k times in the sequence, then $s+d$ must be less than or equal to b_k. We note that our results regarding cramming sequences could be adapted to such an alternative model.

We turn now to the issue of how much can be crammed in a given amount of time. Given a set of spacing constraints $\{(a_k,b_k)\}$, and a positive integer T, we can put an upper bound on the numbers n for which bounded learning of order n is possible in T time steps. If we let $m(i)$ denote the smallest integer k such that $b_k \geq i$, then it can be shown that n must satisfy

$\sum_{i=1}^{n} m(i) \leq T$ and $\left(\sum_{j=1}^{m(n)-1} a_j\right) + n \leq T$. Each inequality implicitly bounds n from above. These bounds reflect the constraints imposed by the limited number of time steps available, and the notions that $\{a_k\}$ and $\{b_k\}$ represent limitations on how fast the level of an educational unit can be built up and how long educational units can be remembered. Details can be found in the *SI Text*.

Despite the negative connotations associated with cramming, the basic idea can actually be useful in a number of settings in real life. A traveler may only care to learn a language enough to travel in a foreign country just once, for example, or a performer may only need to have a certain skill set on the day of a performance and not necessarily after that. Perhaps educational software of the future will have tunable parameters that allow the student (or teacher or parent) to set the goal of the educational process. This way the software may not only adapt to the natural abilities of the students, but also to their personal goals.

Conclusion

The possibilities for future work seem limitless. A more complete theory of infinite perfect learning could be one goal. Such a theory would include more techniques for constructing educational processes tailored to students, and a more complete theory relating a_k and b_k to the maximum rate at which the model student can accrue knowledge.

A major goal should be a truly adaptive educational process: one that adapts to the student in real time. For example, in this paper we model the educational process as a sequence designed to satisfy a set of constraints fixed in advance, but an alternate approach would be to test the student's knowledge throughout the process, and for the schedule to be controlled by an online algorithm that chooses the next unit based on the answers the student has given. Such a system would model the process of a teacher observing student progress before deciding what to teach next.

Modeling this situation would be an exciting challenge. The interaction between the two online algorithms, one modeling the student and the other modeling the teacher, promises to be complex and fascinating, and hopefully enlightening and useful to future designers and engineers of educational software. There is much opportunity here for mathematical modeling, theoretical calculations, and numerical simulations that shed light on what makes an effective teacher and how educational software can adapt in real time to user behavior.

Another area for future work is the design and analysis of models that are tailored to specific subjects. Perhaps a model involving a network of educational units could be used to investigate the phenomenon that it is often easier to learn a set of facts that somehow "reinforce" each other than a set of unrelated facts. Introducing relationships between educational units calls for new models of the student's reception of the units, which in turn call for different educational processes.

Yet another avenue of research is empirical work. The techniques and intuitions gained from theoretical work should be put to use to create actual educational software. Then data from real students can be collected and the process of using the data to validate and refine the models can begin.

Finally, the mathematics of managing spacing constraints in sequences could find additional applications beyond those considered above, for example, to task management in parallel processing or the study of multitasking in humans.

The models presented in this paper are simple and theoretical. Designers of educational software will likely need to implement models and algorithms that are more complex and tailored to the educational content being delivered. It is our hope that work on simple theoretical models will provide the foundations of intuition for designers of educational software, in much the same way that algorithmic game theory does for engineers who work in online ad auctions and other related fields.

With the current boom in educational software—not to mention the humanoid robot teacher industry (14)—it is clear that the time has come to develop a theory of algorithmic education.

ACKNOWLEDGMENTS. Research supported in part by an National Science Foundation (NSF) Graduate Research Fellowship, The John D. and Catherine T. MacArthur Foundation, a Google Research Grant, a Yahoo! Research Alliance Grant, and NSF Grants CCF-0325453, BCS-0537606, IIS-0705774, and CISE-0835706.

1. Obama B (April 27, 2009) Remarks by the President at the National Academy of Sciences Annual Meeting. (Natl Acad Sci, Washington, DC), http://www.whitehouse.gov/the_press_office/Remarks-by-the-President-at-the-National-Academy-of-Sciences-Annual-Meeting/.
2. Ebbinghaus H (1885) *Memory: A Contribution to Experimental Psychology (Translated from German)trans Bussenius RH (1913)* (Teachers College at Columbia University, New York).
3. Dempster F (1988) The spacing effect: A case study in the failure to apply the results of psychological research. *Am Psychol* 43:627–634.
4. Balota DA, Ducheck JM, Logan JM (2007) *Foundation of Remembering: Essays in Honor of Henry L. Roediger III*, ed JS Nairne (Psychology Press, New York), pp 83–105.
5. Cepeda NJ, Pashler H, Vul E, Wixted JT, Rohrer D (2006) Distributed practice in verbal recall tasks: A review and quantitative synthesis. *Psychol Bull* 132:354–380.
6. Crowder RG (1976) *Principles of Learning and Memory* (Psychology Press, New York).
7. Roediger HI, III, Karpicke JD (2011) *Successful Remembering and Successful Forgetting: A Festschrift in Honor of Robert A. Bjork*, ed AS Benjamin (Psychology Press, New York), pp 23–48.
8. Melton AW (1970) Situation with respect to spacing of repetitions and memory. *J Verb Learn Verb Behav* 9:596–606.
9. Bahrick HP, Bahrick LE, Bahrick AS, Bahrick AP (1993) Maintenance of foreign language vocabulary and the spacing effect. *Psychol Sci* 4:316–321.
10. Landauer T, Bjork R (1978) *Practical Aspects of Memory* (Academic, New York), pp 625–632.
11. Pimsleur P (1967) A memory schedule. *Mod Lang J* 51:73–75.
12. Wozniak PA, Gorzelanczyk EJ (1994) Optimization of repetition spacing in the practice of learning. *Acta Neurobiol Exp* 54:59–62.
13. Wolf G (2005) Want to remember everything you'll ever learn? Surrender to this algorithm. *Wired*(16.05).
14. Hyun E, Kim S, Jang S, Park S (2008) Comparative study of effects of language instruction program using intelligence robot and multimedia on linguistic ability of young children. *Robot and Human Interactive Communication* 187–192.

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* $(k+1)^{\text{st}}$ *occurrences of units* $u_0, u_1,\ldots,u_{2^k-1}$, *and the sibling of that node corresponds to the* $(k+1)^{\text{st}}$ *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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 $t \leq T - i + 1$ 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 $b_k \geq i$. 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

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

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,

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

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.

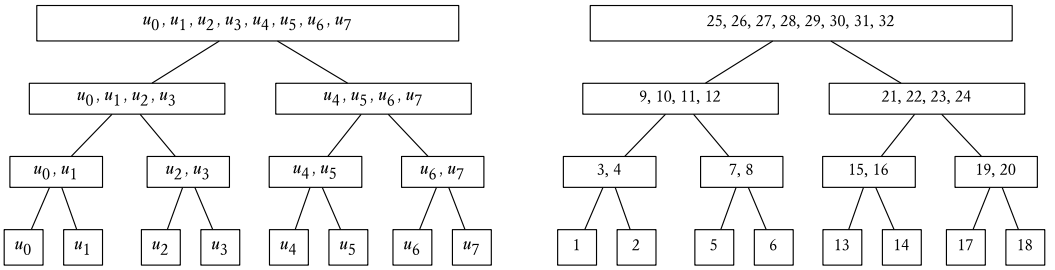

Fig. S1. The full binary tree on the left has each node labeled with the corresponding educational units in the construction of the recap schedule. The tree on the right is identical, except the nodes are labeled with the corresponding time steps. The corresponding schedule, up to and including the left-most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

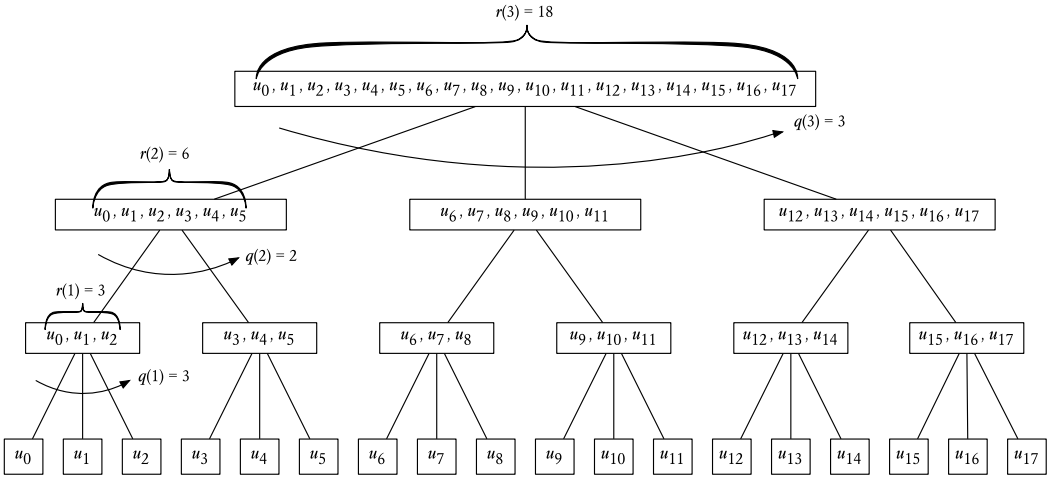

Fig. S2. A tree made using $q(1) = 3$, $q(2) = 2$, and $q(3) = 3$, with each node labeled with the corresponding educational units in the construction of the general recap schedule. The corresponding schedule, up to and including the left-most node at height $k = 2$, is $u_0,u_1,u_2,u_0,u_1,u_2,u_3,u_4,u_5,u_3,u_4,u_5,u_0,u_1,u_2,u_3,u_4,u_5\ldots$

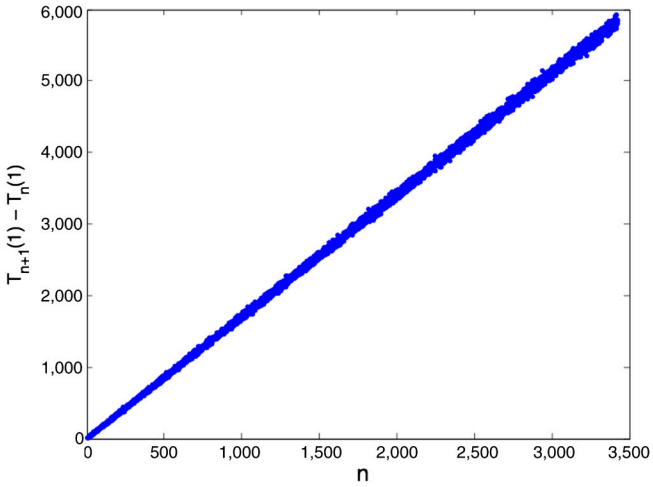

Fig. S3. This figure shows $T_{n+1}(1) - T_n(1)$ plotted against n. The data are taken from the first 1 million time steps of the slow flashcard schedule. A linear regression gives a line with slope 1.7, with a correlation coefficient of $r > 0.9997$.

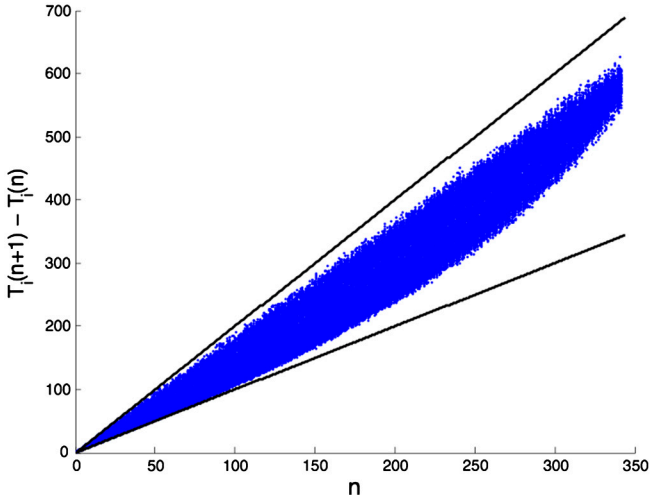

Fig. S4. This figure shows $T_i(n+1) - T_i(n)$ plotted against n, for all i for which data were collected. The data are taken from the first 100,000 time steps of the slow flashcard schedule. Also shown are the lines going through the origin with slopes 1 and 2. All data points lie between the two lines.