## How I made the spaced repetition algorithm

As discussed in the second chapter, Anki's algorithm, which Kram's one is a derivate, suffers from many flaws that negatively impact the review efficiency.

First, it fails at predicting the probability (the term is important, as it is unlikely that any algorithm will ever be able to predict with certainty the exact moment even information is forgotten) of the moment when information splits out of the memory. By applying radical changes to the ease factor as early as the first review day, a card might suffer from different outcomes that negatively impact a user's learning experience.

Hitting the *again* or *hard* button repeatedly, the card falls into what is known as the ease hell. A state where the cards keep showing more often than necessary, even if information finally ends up being retained by the learner, thus resulting in a loss of time efficiency for him, a resource that should be redirected towards learning new content.

Hitting the *normal* button is the least evil solution. Cards deemed as easy end up appearing more often than necessary, but since they might account for at most 15% of the cards according to the author's own experience, this is a negligible issue compared to those seen as hard. The harder cards will show up as often as the *easy* ones are shown scarcely, but this time, because the inherent difficulty is greater and they show up before it is required, impacting the quality of the review, pushing the inevitable failure to a date further in time, way after needed. Finally, it leaves the card in between, which depending on the user's probability of remembering them or not, will anyhow filter them in one category or another. Hitting the *easy* button might conduct the user to misjudge the difficulty of a card, and yet again, next time the card shows up, the probability for the user to remember the card will be low.



Figure 42 the ease hell representation

The main issue with this system is that it treats memory as a binary system; one either fails or manages to remember information. Memory is more of a grey area of memory fluke. For example, a user perceiving a card as easy at first might not be able to recall its information the next time he sees it. This biased perception then creates more harm to the user than benefits.

Thus, Anki's algorithm performs poorly at predicting the probability of when a user will forget the content of a card. Furthermore, the application gains again inefficiency by offering the user to give feedback over four choices. As described by Eshapard (Eshapard, 2017), the mental strain of giving multiple choices results in fatigue that falsifies feedback.

In a nutshell, the Kram algorithm must deal with two issues that Anki failed to tackle. First, how can it mathematically provide information at the right time, namely before the information is forgotten, as to trick the mind to process information as important data? The problem when facing probability, it is often, if not essential to acquire a large pool of data to construct an efficient model that reduces the error margin, as Duolingo does with its HLR model (see chapter 2). Secondly, how to ensure a good balance between spending time on reviews and the time in between so that learners suffer from the least possible pain of reviewing the cards but manage to recall as much information as possible to keep the motivation going in the long run.

Henceforth, Kram comes with a novel solution to tackle those issues at once. Instead of adjusting the ease factor right from the beginning, the algorithm promotes failure from the client's side to offer him the possibility to build solid foundations for his memory by filtering cards by their inherent difficulty pushing the easy cards out of the cue as soon as possible. Because Kram tries to achieve database storage and calculation efficiency for flawless navigation, time constraints limit the number of functionalities being implemented. It disregards the possibility of using users' history and carries on with the predictions once the cards reach a fixed time threshold: 30 days. In addition, the UX was rethought to ease users' mental charge so that they would choose either between fail and pass choices, forcing them to admit failure and move on as quickly as possible to the next step.

Kram's algorithm philosophy could be resumed as a tool that instead of optimizing every single aspect of the learning curve, looks to gain profile flexibility. In other words, it tries to blend in with the people learning patterns. This flexibility is achieved by a double cueing system that aims at filtering tough elements from easier ones, the latter staying, if necessary, in the learning cue to toughen the memory.

Unlike Anki which graduates a card as matured after the first day (meaning that the ease factor gets updated right away), Kram stretches the time required for a card to mature and graduate from the learning cue and effectively gives proper training. Unlike its predecessor, Kram gets rid during the learning cue of any boosts or penalties, not even touching the ease factor so that time is used efficiently.

When a card is added to the cue, the user will see it at least twice that day, one at the beginning of the session, then again after 15 minutes. If successful, the card is pushed to the day after. If successful again, the interval is pushed to 48 hours. Assuming the user will be successful each time, he will see the card after seven days, fourteen days and thirty days. This cueing has two objectives. First, the gap between two and seven days is so large that it gives enough time for the mind to rest and forget what needs to be forgotten, thus making the first filter between easy and hard cards. Would a card reach the 14 days threshold, it might be safe to assume then that the information is successfully remembered. It also ensures that there is a good difficulty balance. The easy cards pushed to fourteen days are perceived positively, thus creating a sense of accomplishment, feeding motivation.

If a user was to fail to remember a card during the learning cue, that card goes back to day one and the cue iterates yet again. The main point is then to offer no penalties that will determine a card's lifecycle too early in the process and offer users to be ready to face the matured cue whose lifecycle gets might get exponentially longer.

Assuming the student reviewed a card with no failure, after 54 days the card enters the second cue. It is assumed that this is an unlikely scenario for most cards as they will require more than 54 days based on the author's own experience. At this point, the algorithm starts updating the ease factor according to rules specific to this cue.

After thirty days, the interval is calculated as such: I' = I \* EF, where I is the new interval. But beforehand, it is the ease factor that gets updated or not. Each card starts with an initial 250% ease factor, as recommended pas Piotr Wozniak. Now that the card is entered in the matured cue, intervals get increasingly linger, but the probability for a user to forget information is still there. Yet, reverting a card to day one even though the user has the feeling that he would remember it the next time he sees it would be counterproductive, a time loss and mental pain for the user. Thus, instead of reverting the card to day one, the new interval is 20% to 15% less than it would have been if successful. The worst-case scenario is easily predictable (the values are rounded up):

|  |  |  |
| --- | --- | --- |
| Failed attempt | Calculus | Results |
| One | 30 \* 215% | 65 |
| Two | 65 \* 175% | 114 |
| Three | 114 \* 145% | 166 |
| Four | 166 \* 130% | 216 |

Table 1 – Results of the worst-case scenario in matured cue

The ease factor is blocked at a minimum of 130% so that the intervals are not too close to the previous one, as recommended by Piotr Wozniak. If a card fails its duty five times in a row, then it is marked a leech and will not repeat into the deck. Such harsh penalties are supplementary incentives for users to be proactive in their card management by filtering bad cards out of the deck. After reaching this stage, the user has three choices, either put back the card in the learning cue, update it, or delete it. Yet, because memory is not an exact science, the algorithm puts on a safeguard. If a card failed thrice, but on the fourth attempt the user remembered the card correctly, the failure counter decreases by one, pushing back the threshold further back. On the other hand, the best-case scenario is made so that cards with no apparent difficulty are pushed to very long intervals as such:

|  |  |  |
| --- | --- | --- |
| Failed attempt | Calculus | Results |
| One | 30 \* 250% | 75 |
| Two | 75 \* 250% | 188 |
| Three | 188 \* 270% | 508 |
| Four | 508 \* 270% | 1371 |
| Five | 1371 \* 295% | 3975 |

Table 2 – Results of the best-case scenario in matured cue

The maximum threshold is set to 10 years so that cards are not lost in eternity. Moreover, the ease factor is not increased systematically unlike when the failure button is hit, to avoid an exponential growth that would decorrelate from the forgetting curve. For an ease factor to gain 20 points, the learner must remember twice correctly the card.

Cards in between will not follow such a direct pattern, but rather fall in between.

Thus, the algorithm is done so that the cards not requiring any studies are buried at the bottom of the cue and those needing more attention reappear more often.