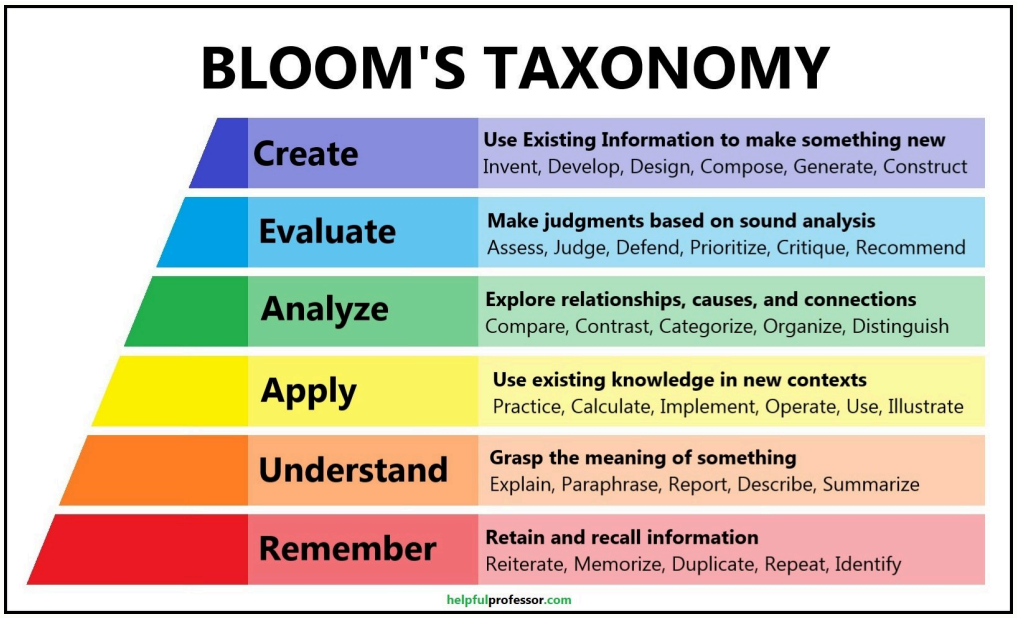
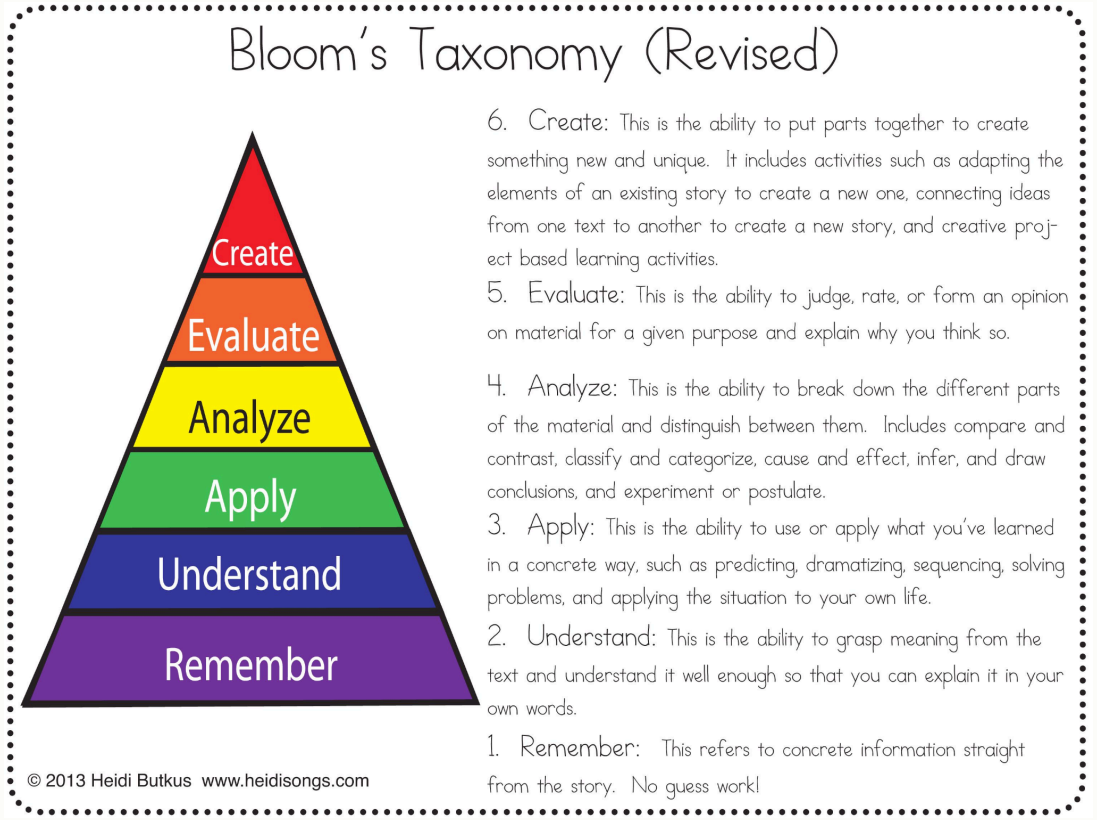
Problem Solving Framework

Bloom’s Taxonomy





Рассказывает нам про уровни экспертности. Очевидно, что то, что снизу занимает намного меньше времени, дает общее понимание и навыки, но не делает вас экспертом. Приближаюсь к верхушке, вы тратите года жизни, но приобретаете экспертность.

Для ежедневного применения на работе основных алгоритмов, базовых конструкций и паттернов программирования, нам необходимо пройти несколько этапов и остановиться на нужном нам уровне. Мы хотим применить принцип Парето и получить максимум профита за меньшие усилия.

Давайте разберем их.

1. Remember - вы помните о каких-то названиях, структурах данных, алгоритмах. Что они существуют. Видите их в тексте и термины вам знакомы.
2. Understand - вы начинаете понимать, что это и где используется в общем. Пока что вы не особо понимаете, как это использовать.
3. Apply - теперь вы можете применять это в реальной жизни, решать свои проблемы с помощью этого инструмента. Понимаете, как и где применить. Возможно, уже даже не задумываетесь об этом.
4. Analyze - вы умеете анализировать, экспериментировать, выделить какие-то сущности и части, категоризировать и тп
5. Evaluate - у вас есть свое собственное экспертное мнение по данной теме, которая сформировалось через большой опыт, а не потому что вы пару дней прочитали про конкретную тему и теперь у вас есть “экспертное мнение”. Это координальная разница между экспертностью и “я так думаю”.
6. Create - вы способны брать свои инструменты и создавать нечто новое с помощью них, вы творите и меняете мир.

Для ежедневных обычных задач, нам скорее всего, понадобится 3-й уровень Apply. Здесь у вас есть автоматические навыки, вы не тратите на это время, понимаете, как оптимально написать и основная ментальная энергия будет идти на решение бизнес задач.

На всех уровнях ниже, огромная часть вашей ментальной энергии идет на гугление, понимание, как этот код написать, какой алгоритм выбрать. Или может быть структуру данных? Вы гуглите, думаете, вспоминаете, где вы что видели, что вообще загуглить. Бизнес задача пока стоит в сторонке и ждет вас.

Также, на втором уровне может жить одна из ошибок мышления, когда человек прочитал в вики определение и запомнил его. Он ошибочно считает, что понимает тему и все говорит “это легко”, “это база”, “что тут непонятного” и т.д. Но, сам не может ответить на уточняющие вопросы типа “а как это работает? Расскажи, раз так просто?”. Часто можно тут увидеть разного рода защитные реакции типа “я не обязан ничего тебе рассказывать”, “тебя в гугле забанили?”, “у меня там все нормально, я уверен в себе”, “нет времени тебе объяснять” и другие. Подсознание внушило себе, что ты все знаешь и разрушить это чувство - разрушить свою психику и все представление о себе.

Нужно делать все, чтобы там не оказаться. Знать, что ты знаешь и что не знаешь.

Как проходить интервью?

- Задавать вопросы. Собирать информацию
- Уточнять
- Ограничения
- Спрашивать примеры и как можно больше
- Зачем и как будет применяться?
- Можно ли сделать в виде приближения, а не четкого и красивого кода?
- Достаточно ли правильного направления мышления?
- Код отрефакторим потом, если будет время? Это норм?
- Можно ли просто решить в лоб bruteforce solution или надо сразу искать супер оптимальное решение? Вдруг не успеем?
- Попросить подсказку. Это не допрос и не загадки, а мышление и общение.
- Не кодить, пока не попросят и пока нет идеи, что вы делаете
- Предлагать подходы и идеи. Спрашивать, что человек думает про них
- Набросать высокоуровневый псевдо код. Некоторые вещи могут потребовать много времени и дебаггинга, поэтому лучше их выделить в псевдо функции. Например, читаем инпут, потом парсим, создаем айтемы, фильтруем айтомы, находим самый лучший айтем, возвратить результат. Перед этим КАК-ТО модифицировать (пока не знаем как, просто напишем ДЕЙСТВИЕ ТАКОЕ-ТО). Ничего не реализовываем. Используем метод black box. Спросить, что человек думает. Ок или нет? Есть смысл или нет?
- Если время осталось, реализовать самые важные функции с общим алгоритмом. Вспомогательные функции - если успеешь.
- Интервьюер работает вместе с тобой. Это не допрос. Не только он тебя проверяет, но и ты проверяешь с кем будешь работать. Как он помогает тебе, общается, ведет себя, адекватность. Грубый он или токсичный. Токсичные люди будут так потом и работать в форме допроса, где он старослужащий и самый умный, а ты говно и должен подчиняться. Смотрим, что вам идут на встречу, подсказывают, шутят, все на позитиве.

Ментальная гигиена

1. Относитесь как к игре, отпустите напряжение. Накидывайте примеры, меняйте условия, решайте частные случаи, и другие методы из списка ниже. Это значительно снижает уровень стресса и создает интерес. Ваш персонаж в рпг и вы его прокачиваете по-немногу. Ищите полезные предметы, которые можно заюзать позже (паттерны, техники) и тд.
2. Решите ли вы или нет - это не показатель вашей глупости и умности. Отпустите нарциссические установки “быть лучше всех”, “не потерять свое лицо”, “все узнают, что я дерьмо” и т.д. “Я не смог, значит я тупой” - тоже не верно, ты просто не имеешь достаточно опыта и паттернов или просто не знаешь алгоритм. Или просто сегодня машинка хуже работает. Это прямо полностью нормальная история.
3. Процесс, а не результат. Частая ошибка, ведущая к сильному стрессу, паническим атакам и неуверенности в себе - желание показать себя другим и самому себе через результат. Мы полностью фокусируемся на процессе и получаем от него удовольствие. Не “Я должен сейчас решить задачу, иначе все пропало”, а “Я сейчас сяду и порешаю задачки, поразминаю свои мозги немного. Классно проведу время вместо пива на диване. Если забыл - повторю. А почему забыл?”.
4. Не сравнивать себя с другими. У каждого свой путь, способности, время. Все люди очень разные. Можно найти кумира и хотеть быть таким же. Но ни в коем случае нельзя завидовать и угнетать себя “он лучше меня, я никогда не смогу так, я просто тупой”. Это войдет в привычку и вы обречете себя на вечные страдания, зависть, неуверенность и токсичность.
5. Из предыдущего пункта вытекает, что если задача не решается и идей нет - переключитесь на что-то другое. Такое повторяется снова - что-то просто не знаете и стоит посмотреть решение. Когда смотреть - каждый сам решает. Но, лучше погенерить идеи сначала и активно включиться в работу. Стоит делать несколько заходов и у вас есть время или минут 20-30 порешать и потом смотреть решение - зависит от потребностей.
6. Найдите себе мотивацию - Зачем это делаете? Чего вы достигнете? Станете ли вы лучше? Какой навык вы качаете? Заинтересуйте себя сами. Выработайте привычку решать по задаче в день. За год - 365. Ну как-бы норм.

Focused & diffused modes

Вы не можете долго находиться в режиме фокуса и концентрации. За каждым напряжением должно быть расслабление. Поэтому регулярно нужно менять род деятельности и менять контекст. Погулять пешком, медитация, сон, почитать книгу, что-то из другой области, что не требует концентрации и не сильно связано с задачами. Так вы дадите возможность мозгу отдохнуть и найти нейронные связи между проблемой и имеющими знаниями и опытом. Связать их через ассоциации, аналогии и другие механизмы. Если вы все время в фокусе, вы теряете гибкость мышления. Это может быть 1 час работы и 20 минут отдыха. Или 45 минут работы и 10-15 минут отдыха и т.д. Учеба и тренировки - это постоянные смены деятельности, усилие и расслабление, focused и diffused modes.

В focused mode вы сконцентрированы и сильно думаете над проблемой. В diffused mode вы стараетесь не думать ни о чем специфическом и позволяете мозгу расслабиться и думать о чем он хочет. Вы как-бы отпускаете контроль. Часто этот режим связывают как раз с креативностью и способность мозга связывать множество идей и мыслей, областей, который на первый взгляд могут быть не связаны.

Меняем контекст регулярно. Цикл работа - расслабление. Фокусное внимание и рассеянное. Логика и творчество.

Fast Brain and Slow Brain

Интуиция и способность видеть паттерны развивается с опытом, это невозможно сделать сразу, как и накачать мышцы. Нужны усилия и отдых и сон между ними. Интуиция работает на паттернах, ощущениях, подсознательной и неосознанной связи с предыдущим опытом. Она часто носит названия Fast Brain. Вы видите что-то и сразу понимаете, чо это - человек, животное, токсичность, что-то красивое, binary search. Оно работает на опыте, который у вас уже есть.

Другой вид мышления носит название Slow Brain. Он больше относится к осознанному мышлению. Мы делаем усилия, думаем, размышляем над проблемой, перебираем варианты и тд. Ведет к ситуации, когда решение внезапно приходит вам в голову.

Таким образом мы развиваем первый тип, получая опыт. И второй тип - используя сознательное мышление, логику и критический анализ. Что ведет к появлению нейронных связей и установок, которые затем переходят на подсознательный уровень и становятся интуицией - первым типом.

Развиваем оба мышления. Быстро сканим интуицией. Потом внимательно обдумываем. Повторяем по кругу.

Mental Picture

Пробуйте развивать свое воображение и картинку в голове. Учитесь управлять большими частями и деталями, оживлять, преобразовывать, уменьшать и увеличивать, придавать разные формы. Так вы сможете перебирать какие-то идеи в голове. Для людей, у которых плохо с воображением или имеющих Aphantasia, стоит все делать на бумаге - постоянно рисовать или брать готовые рисунки.

Развиваем воображение. Мнемотехника. Дворец памяти.

Сбор информации

- Прочитайте внимательно
- Поймите проблему
- Перескажите своими словами
- Какие данные у вас есть?
- Что нужно найти?
- Какие ограничения?
- Нарисуйте проблему
- Нужно ли найти ещё данные?
- Уточняйте. Так много и часто, как возможно, пока не наступит момент, что вопросов больше нет. (Как? Почему? Где? Зачем? А здесь? Что, если?)
- Ищите примеры (простые, сложнее, сложные)
- Где применяется?
- Что решает проблема?
- Кто связан с ней?
- Можно ли менять условия?
- Чего мы не знаем?
- Валидные ли условия?
- Нет ли противоречий?
- Разделите условия на составные части

Работайте над гипотезами

- Какие подобные проблемы мы уже решали? Попробовать увидеть паттерн, дать поработать интуиции, а потом ее проверить
- Какие проблемы мы видели, но в другой форме?
- Переберите все возможные техники и алгоритмы, которые знаете
- Посмотрите на проблему с разных углов, точек зрения, областей.
- Декомпозиция. Может проблема состоит из комбинации других? Попробовать выделить подпроблемы насколько это возможно. Они должны быть проще, чем текущая. Они также разбиваются на более простые подпроблемы и тд, пока не наступит этап, когда каждая из маленьких проблем имеет решение.
- Решаем в лоб самым тупым способом - brute force. Есть ли медленное решение, на котором можно проверить более быстрое? Переберем все значения. Может дать большее понимание задачи и уверенность, что какое-то решение уже есть.
- Нарисуйте на бумаге варианты идей
- Попытка угадать алгоритм, как черный ящик, используя гипотезу - “А что, если оно вот так работает?” (неизвестно, как именно досконально, но поведение и результат вот такой). Потом начинаем ее проверять - прямолинейно или в обратном порядке, или от противного (если бы она не работала, то у нас была бы такая ситуация, при которой такие-то данные, условия, действия и тд, решим когда все плохо и потом посмотрим, что останется для ситуации, когда все хорошо).
- Уберите все невозможные варианты из решения. Запишите их отдельно. Иногда, самое невозможное и есть путь к решению.
- Найди все corner (extreme) cases вашей проблемы. Попробуйте решить проблему для них. Возможно, это даст понимание, решение или путь к нему.
- Используйте инверсию (Что, если все наоборот? Как сделать так, чтобы такие условия точно не сработали? Как сделать наоборот хуже? Если я хочу помочь Индии, то какие есть способы, чтобы навредить Индии?). Этот способ очень любил Charlie Munger.
- Используйте backwards thinking. Думаем не вперед, как в проблеме, а назад. Как-будто все действия уже случились и наше конечное состояние на самом деле начальное, а начальное конечное. Мы идем не вперед, а назад и пытаемся найти закономерности или идеи.
- Решите самую простую версию по данным. Все числа равны нулю, только один или два элемента, ноль элементов, только одна-две вершины в графе, только одна строка в матрице и тд.
- Решите самую простую версию по условиям. Уберите все условия и ограничения. Можно ли решить задачу теперь?
- Измените упрощенную версию. Постепенно добавляйте в упрощенную версию больше значений, параметров или условий. Приближает ли к решению?
- Представьте ситуацию, когда все условия соблюдены.

- Найдите такую упрощенную версию, которая может быть сведена к более сложной текущей версии, но которую можно решить.
- Рассмотрите более сложную версию (парадоксально, но может помочь).
- Изменить условия. Пробовать решить за O(n), O(nlogn), O(n^2), O(n^3) и т.д.
- Возможно для решения уже есть формула. Последовательности и тп.
- Сколько состояний у нас есть? Что меняется в процессе работы алгоритма? Что остается неизменным?
- Моделируйте, добавляя новые и убирая текущие свойства проблемы.
- Представьте, что решение уже готово. Можем ли мы как-то это использовать? Разложить еще на составляющее и тд? Выделите его в функцию или объект, который вернет нужный ответ или частичный ответ. Или использовать, как заглушку (Get Filtered Items - как-то там вернет то, что нам надо для решения).
- Итеративное мышление. (Быстро и жадно, сохраняя какой-то инвариант и накапливая решение).
- Декларативное решение, рекурсия, divide and conquer. (Более емко, нет изменяемых данных и т.д.). Пусть задача решена для текущего случая или отрезка - как нам перейти в следующее состояние? Пусть задача решена для левого и правого отрезка - как нам их объединить в текущем отрезке?
- Используйте аналогию (аналогичная проблема, которую решили раньше).
- Используйте ассоциативное мышление (Найти какие-то связанные задачи, объединяя с текущей по какой-то идее, признаку или свойству).
- Используйте индукцию (Выведите общее решение из наблюдения и комбинации простых вариантов решения и частных случаев).
- Решите частные случаи. Specialization.
- Решите в общем виде. Generalization (generic решение для всех похожих проблем). Для этого нужно искать закономерности и паттерны. Чем все частные решения похожи, что можно выделить в общее? Теперь общее может принимать и обрабатывать любые частные случаи.
- Inventor's paradox. Чтобы решить частную проблему, мы можем попытаться быть намного более амбициозными и решить намного более сложную или общую проблему, больше, чем текущая. И решение текущей будет получено из данного решения. Общее решение может иметь проще алгоритм и более чистый дизайн, и занимать меньше времени для решения.
- Можно ли изменить данные?
- Можно ли рекомбинировать проблему и элементы? Поставить их иначе, разный порядок, рисунок, паттерн и тд.
- Можно ли трансформировать проблему? Измените вашу проблему так, чтобы создать новую проблему, решение которой поможет вам найти решение текущей.
- Можно ли трансформировать похожую проблему?
- Вернуться назад и проверить, все ли мы использовали из условий?
- Визуализируйте. Представьте, как оно работает. Оживите вещи в задаче. Посмотрите на них мысленным взором. Манипулируйте ими в вашем воображении.

- Важен ли порядок? Если нет, мы можем переставлять, сортировать, как нам удобно. Какой алгоритм можно применить, если числа сортированы? Что, если они в определенном порядке?
- Какие есть инварианты? То, что всегда должно быть валидно после каждой итерации? То, что можно поддерживать в ходе алгоритма в валидном состоянии и использовать для получения следующего частичного или полного ответа. Что ведет нас к ответу.
- Гуглите идеи.
- Спрашивайте у других.
- Найдите похожую проблему или отдаленно похожую и решите ее (Атакуйте все проблемы вокруг, пытаясь приблизиться к данной).
- Придумайте похожую проблему, если не помните ни одной.
- Какая связь между данными и неизвестным?
- Brainstorming. Генерируйте идеи. Любой кринж, уникальность или неочевидность.
- Reduction to an absurdity. Если это было бы так, то свиньи умели бы летать. Если земля была бы плоской, у нее был бы край и с него можно было бы упасть. Мы берем утверждение и развиваем его, доводя до абсурда - ложного или невозможного заключения, которое нивелирует изначальное утверждение.
- Proof by contradiction. Делаем отрицание исходного утверждения и доказываем его. Если оно приводит к противоречию с исходным утверждением, значит исходное была правдой. Если свойства объекта существуют, то докажем, что ни у какого объекта их не существует.
- Создайте всевозможные тестовые данные для проверки.
- Используйте автоматические fuzzy тесты.
- Фиксируйте промежуточные решения тестами и отдельными блоками. Сохраните себе то, что точно работает.
- Перечитать условия еще раз.
- Используйте симметрию. Попробуйте найти симметричные детали. Нарисовать вторую половину, какой она могла бы быть и тд.
- Рассмотреть разные области знаний. Искать взаимосвязи между разными науками, областями, разделами. Искать связи между несвязанными вещами.
- Попробовать смоделировать - отбросить составные части, которыми можно пожертвовать для того, чтобы модель реального мира была проще и доступней на бумаге.
- Отложить задачу на потом и сменить контекст. Вернуться позже сегодня или завтра-послезавтра. Поделать что-то не связанное с этой задачей или вообще этой деятельностью.

Пробуйте каждую гипотезу

- Одну за один подход.
- Вовремя понять, если она не работает.
- Не бояться оставить ее, если она не работает и искать другую.
- Фиксируйте этапы письменно.
- Если нужно, be agile, и откатитесь на пару этапов назад.
- Не долбитесь головой об стену, если нет прогресса.

Обдумайте решение

- Как у вас получилось решить проблему?
- Как еще можно решить?
- Где можно применить в будущем?
- Какие есть похожие проблемы?
- Как можно было сделать это лучше?
- Как можно сделать красивее?
- Проверьте каждый шаг.
- Что мешало вам решить проблему?
- Как избежать этих помех в будущем?
- Как делать это быстрее в будущем?
- О чем вы думали?
- Что вас отвлекало?
- Мы нашли новый паттерн? Стоит запомнить?

Black Box

- Используйте готовые решения, структуры данных и алгоритмы для решения.
- Если нужно или интересно, можно их изучить потом.
- Как и когда уметь их применять - отдельный навык.
- Соберите себе решение из нескольких черных ящиков. Думайте про основной алгоритм, используя вспомогательные инструменты, не тратя энергию, внимание и время на то, как они устроены.
- Чтобы применять технологию или алгоритм, необязательно знать его устройство. Достаточно знать, где его использовать и что он дает на выходе. Каждый человек делает это в 90% своей коммуникации с внешним миром. Почти все инструменты, машины, предметы, явления природы моделируются человеческим мозгом как некоторая модель, отбрасывая детали. Это защитный механизм мозга, который не дает ему быть перегруженным деталями, но позволяет эффективно пользоваться всеми нужными инструментами. Мы успешно живем, применяем их, не зная их устройства. Это здоровый подход без упоротости в перфекционизм или внутренние комплексы и обиды, когда подсознание тебе говорит, что ты должен знать абсолютно все, иначе все поймут, кто ты, тебя наконец раскроют, что ты ничего не знаешь.

References

https://en.wikipedia.org/wiki/Impostor_syndrome
https://en.wikipedia.org/wiki/Dunning–Kruger_effect
https://en.wikipedia.org/wiki/Reductio_ad_absurdum
https://en.wikipedia.org/wiki/Proof_by_contradiction
https://en.wikipedia.org/wiki/Pareto_principle
https://en.wikipedia.org/wiki/Black_box
https://en.wikipedia.org/wiki/How_to_Solve_It
https://en.wikipedia.org/wiki/Inventor%27s_paradox
https://en.wikipedia.org/wiki/Aphantasia
Book: “Learning How to Learn: How to Succeed in School Without Spending All Your Time Studying; A Guide for Kids and Teens”, Barbara Oakley PhD
Book: “Thinking, Fast and Slow”, Daniel Kahneman