Day 1: BestFour Landing at Greedy-ent Mart

CalliLens Lens the Magical Lens: Mission on the Land of Apple Pi	**1**
Landing at Greedy-ent Mart	**9**
Decoding the Nature: Golden Spirals, Fibonacci and Recursion	**13**
Selling CalliLens: Coin Change Challenge and Greedy Approach	**25**
Coconut Nut and Macadamia Nut: Fractional Knapsack Problem	**31**

CalliLens Lens the Magical Lens: Mission on the Land of Apple Pi

"Look!" Dark Knight shouts as he gallops to his clan, "I found this in Grandpa's treasure box."

"What is it?" asks the BestFour.

Shining beneath the warmth of the sun, Dark Knight holds out a glistening and immaculate lens. However, right before the rest decide to lay a hand on it, a line of fine print catches Banana Split's eyes: *CalliLens welcomes you to the crystal world.*

"CalliLens?" Bubble Gum murmurs, "I heard of HoloLens by Microsoft. Their transparent lens shows you the augmented reality. If I'm hungry, a heap of alfalfa hay in front of my nose will display the number of calories it has; and when I'm thirsty, a small stream of river along my hooves will display the mineral ingredients."

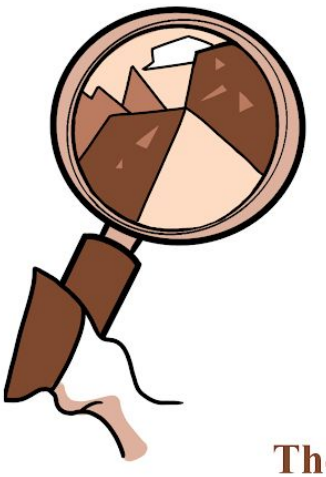
“Alright,” Banana Split chimes in, “if *holo* means a three-dimensional image, then what does *Calli* mean?”

Everyone turns to look at each other blankly before falling silent again.

“Hmm…” Banana Split continues, “if calligraphy is a style of beautiful writing, and *Calli* means beautiful in Greek, then CalliLens must be—”

“It must mean that we’re going to see a beautiful world through these CalliLens!” Bubble Gum exclaims. “That’s wonderful.”

Crystal Clear and Abstract World



The roads are straight and the stones are smooth.

The cookies are crispy and cakes are fluffy.

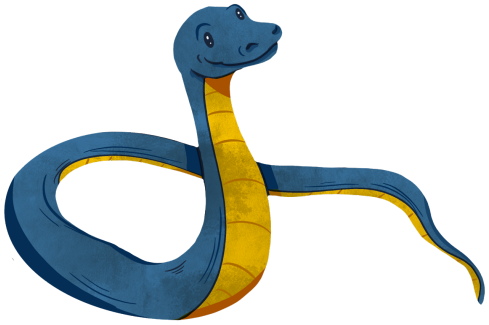
The dreams are vast and our friendship is warm.

"*We get it!*" the BestFour shout together, "*Beautiful. We can see an abstract world out of the unnecessarily chaotic reality!*"

Everyone gasps and turns towards Mighty Python. “What is this mysterious CalliLens?”

Lifting his head and gazing towards the horizon, Mighty Python begins to explain:

The CalliLens has a remarkable ability. When you are agitated by a monster problem, wear this gadget, and you will see.



1.) The monster problem will shrink and decompose into several miniature sub-problems that are easier to manage.

2.) The patterns among the closely related problems will jump out, and the irrelevant details will fade away.

3.) The instructions on how to attack the monster will emerge in front of your eyes.

CalliLens is a symbolic representation of Computational Thinking: a set of problem-solving methods to abstract the problems and identify algorithms as their solutions in a computer-understandable way.

Computational Thinking Process

“Who here has a monster problem?” Mighty Python asks while waving the CalliLens in the air.

“Me,” Banana Split replies, “Mom asked me to clean my room today. Where should I start—”

“Remember...the CalliLens will tell you.”

“Oh right...it shows that I should gather my clean clothes in a pile, toss the dirty ones into the laundry bin, and place my books onto the shelf. This is easier than I anticipated!”

“Yep, now let me tell you something.” Mighty Python continues, “long, long ago, the Land of Apple Pi was thriving and transforming with humans’ inventions generation after generation. They saved people’s lives, lightened them in the darkness, and elevated the power of controlling their own destiny.

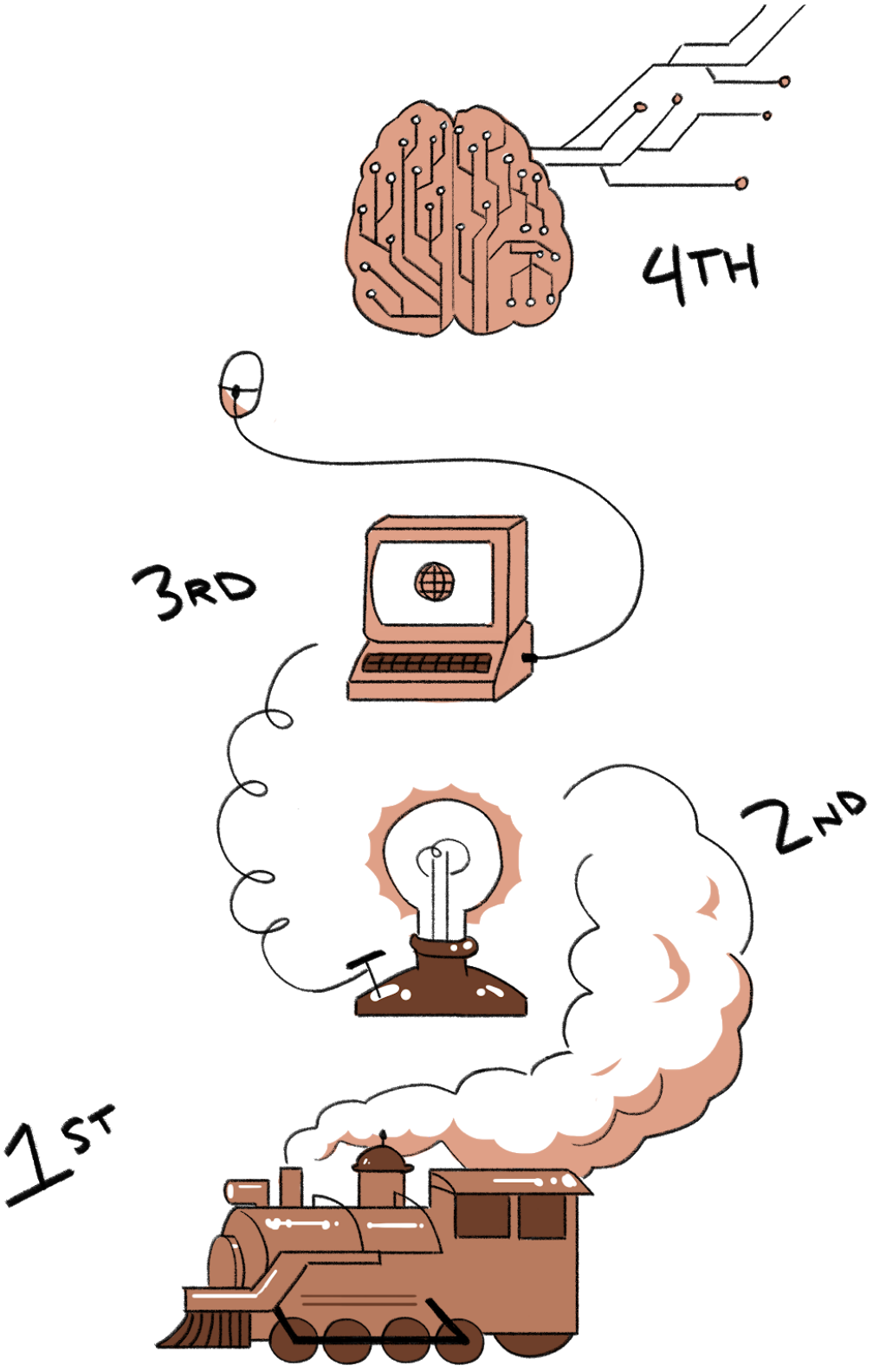


Figure 1.1 Four Industrial Revolutions: inventions that make human beings thrive

However, some of the inventions showed two sides of the coin: nuclear reaction as a power source but also as a weapon, dynamite to blast rocks in mining but also to blow people up! Some inventions were even purely driven by people's greed for possession and greed to win. This led the world to start twisting and wrinkling."

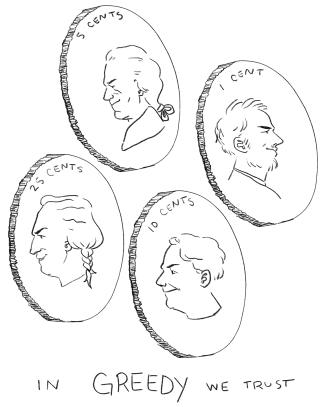
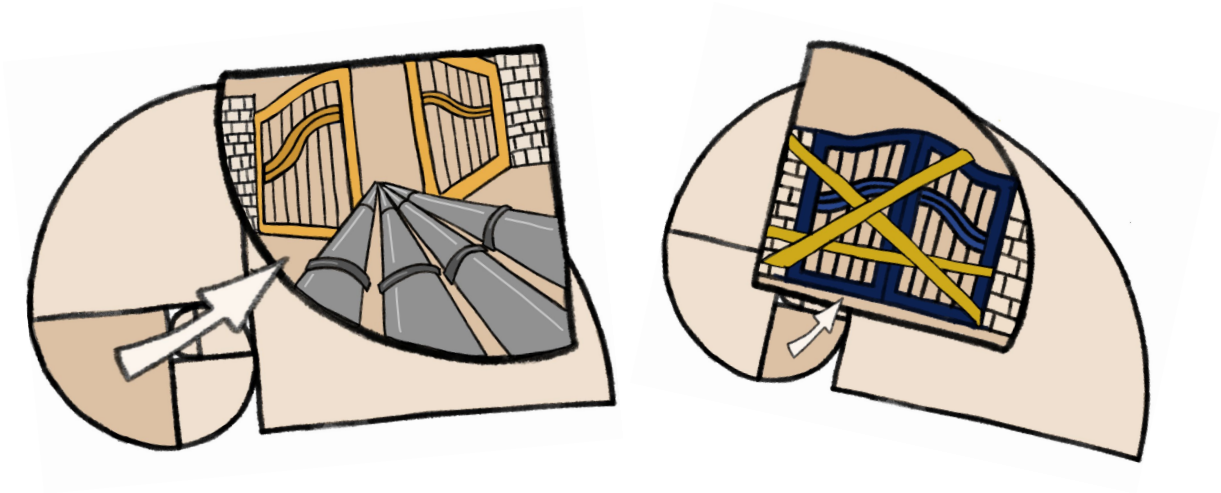


Figure 1.2 Toxic inventions that age the Earth: atomic bombs, deadly gasses, spam email, etc.

"Now what?" Everyone asks with their eyes wide open.

“There is only one chance to reverse the Earth’s aging, and that is we first have to take the CalliLens back to the Land of Apple Pi. It will help navigate us through the chaos. Then we need to build pipelines to flow the righteous inventions from the Gate of Summit to every city and town. Lastly, the Gate of Traceback must be sealed, for it is the source of the toxic inventions.”



In God We Trust

“Next week is a holiday. Let’s secretly travel back to the Land of Apple Pi and reverse the Earth’s aging!” Dark Knight proposes.

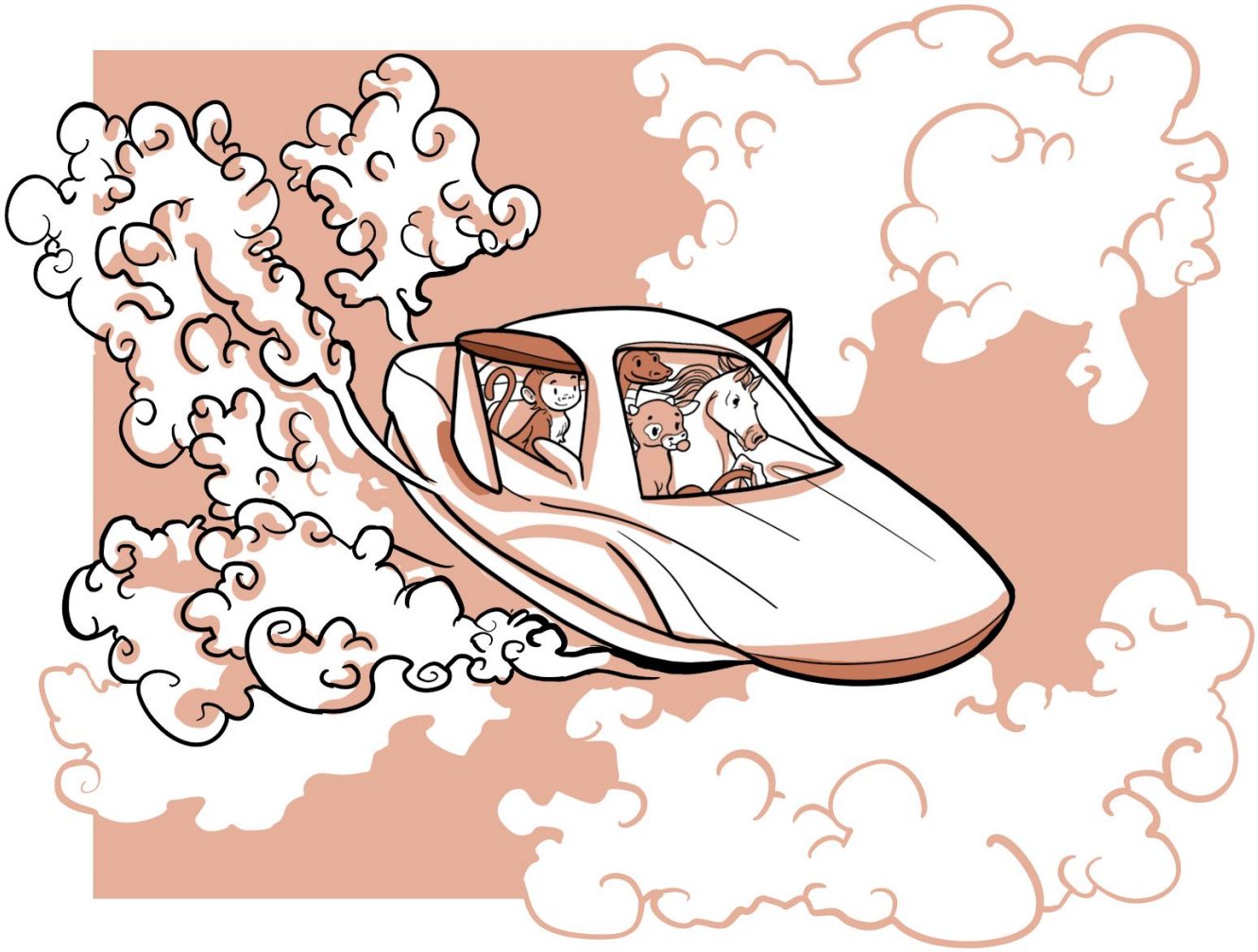
“CalliLens lens is the magical lens.” Recites Banana Split, “If you wear it often, it’ll solve your question. I’ll just borrow the other one from Grandma.”

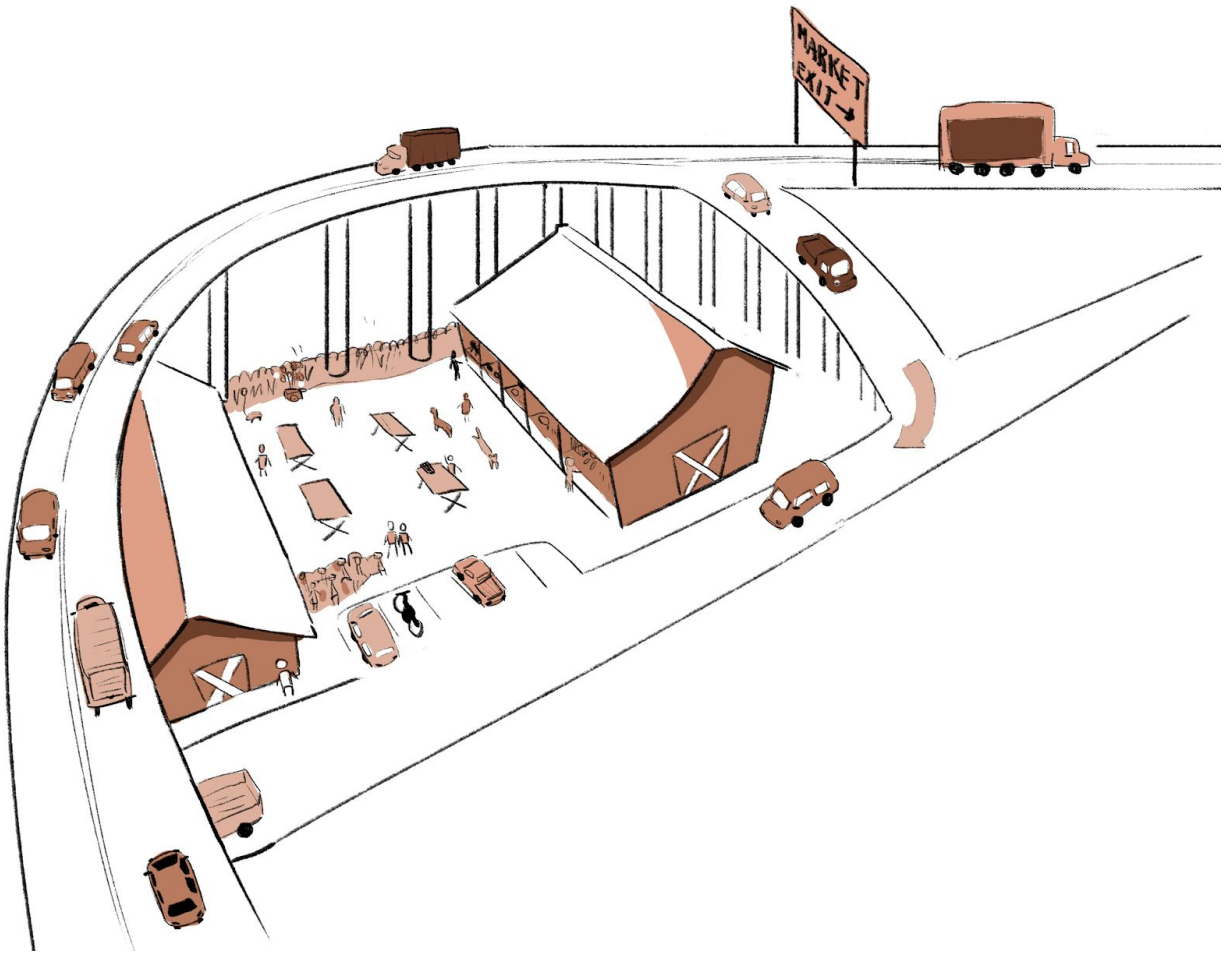
“Ok, sounds great.”

“Nonni, we’re going camping, just typical camping in the woods—” The BestFour waves and calls out as they are off to the *road*.

Landing at Greedy-ent Mart

Hovering on top of a peculiar and restless land, the BestFour notices tiny, little ants crawling along several intertwined lines.





They land at a market place after following a hazy cloud.

Banana Split, however, could not help but to reach out for a tempting plate of nuts.

“Wait, young man, don’t you know that you should pay first?” The cashier growls as he points to a box of coins and bills collected from the customers.

Banana Split peers at the box for a split second before getting interrupted.

"Once," Bubble Gum recalls, "Grandpa showed us the coins and bills in his treasure box. Remember? They were used as physical currency for goods exchanged on the Land of Apple Pi."

"Oh, you're talking 'bout this right?" Banana Split zooms in close to the cashier's box, "well, can I pay with my fingerprint, facial recognition, retina recognition, or with any other sort of biometric identification? We don't carry physical currency anymore."

"W-What?!?" The cashier's jaw drops. "What are you guys? Aliens?"

“Ah no, unfortunately, but I have an idea: we can sell what we have to exchange the physical currency here.” Dark Knight announces.

“Glad we brought two CalliLenses. What about selling one?” Banana Split asks as he rummages through his pocket.

Decoding the Nature: Golden Spirals, Fibonacci and Recursion

“Allow me to do the honors of trying out this masterpiece if you insist.” A nearby farmer proudly states. “Oh my me! What happened to my Romanesco broccoli? What are these 5 spiral lines doing here? And why is it formed by the pinnacles as it curls to one side? Why are there 8 curling to the other side?“

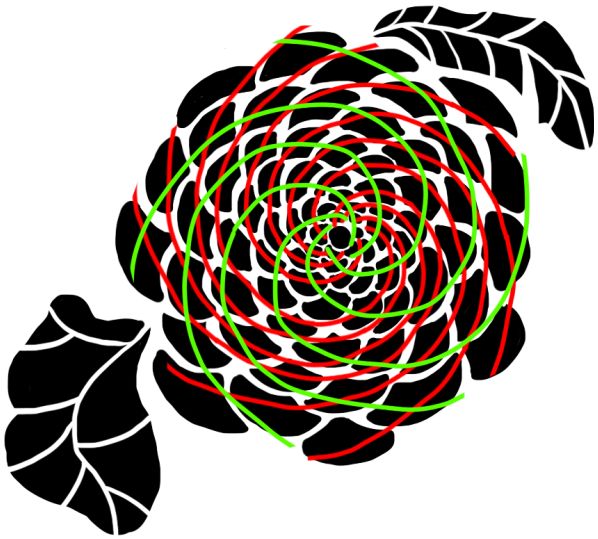
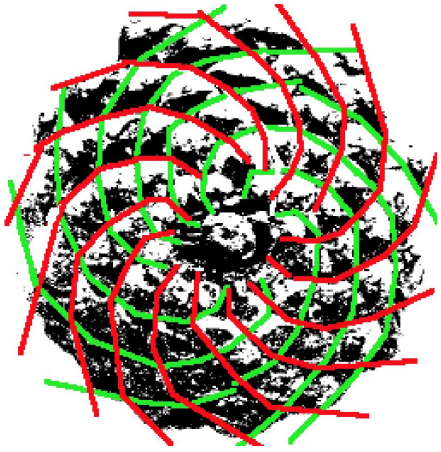


Figure 1.3 Spirals on romanesco broccoli



“What do you mean?” Another farmer asks after overhearing the conversation. “I-...my dear pinecone. It has 8 green spirals and 13 red spirals drawn along it!”

Figure 1.4 Spirals on pine cone [1]

“Oh, 5, 8, and 13! All these spiral numbers are from the Fibonacci sequence.” Banana Split jumps in, “In fact, based on a statistical study in Norway, 95% of pine cones demonstrate Fibonacci spirals.” [1]

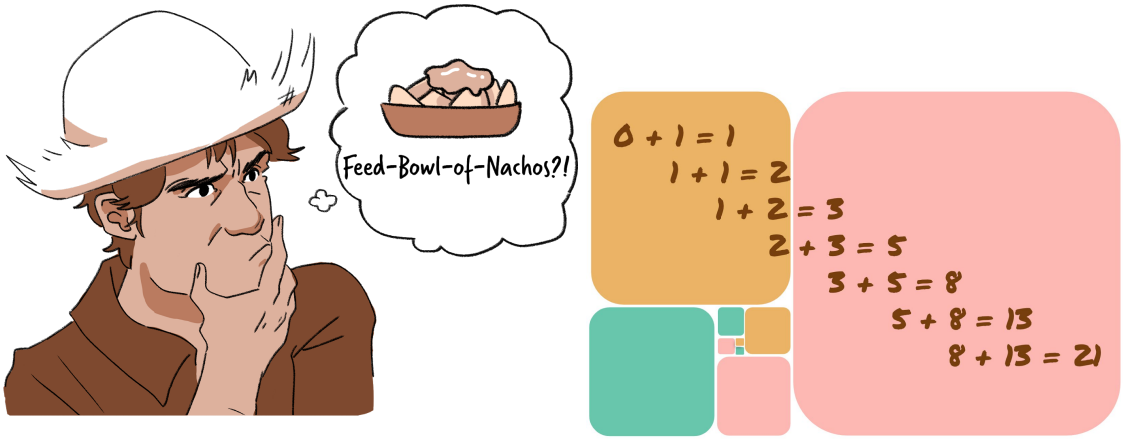


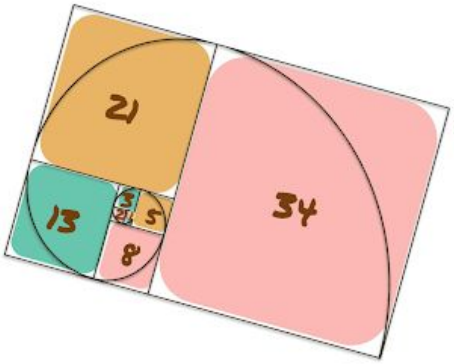
Figure 1.5 Fibonacci computation

Fibonacci

A series of numbers in which each number is the sum of the previous two
Re-discovered by Leonardo Fibonacci, an Italian mathematician born somewhere around 1170

Golden Ratio

The ratios of Fibonacci numbers F_n/F_{n-1} approaches the factor 1.618 as n approaches infinity. This magic factor is called φ (Phi) or Golden Ratio, which was first proved by Scottish mathematician Robert Simson in 1753.



Golden Spiral

Therefore, this special spiral is called Golden Spiral. It gets wider by a factor of 1.618 for every quarter turn it makes.

Figure 1.6 Fibonacci number sequence, golden ratio and golden spiral

“Fibonacci numbers are very common in nature,” Bubble Gum adds.

“Well...”

“OH MY GOSH, is this even my flower?” Asks a nearby farmer.

In this representation, the smaller the numbers, the younger the petals.

3 green clockwise spirals and 5 counter-clockwise red spirals are shown.

Again, 3 and 5 are Fibonacci numbers! In addition, the spirals are golden spirals.

The adjacent petals are always apart by a constant angle, 137.5°! 137.5° between petal 1 and 2, same 137.5° between petal 2 and 3, and so forth.

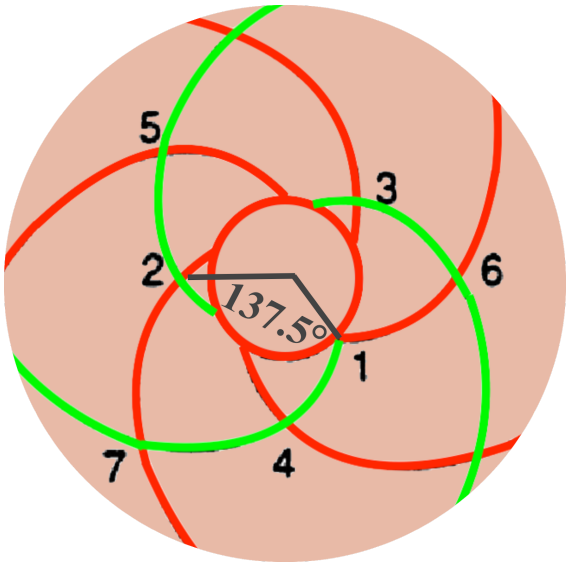


Figure 1.7 Spirals on flower petals

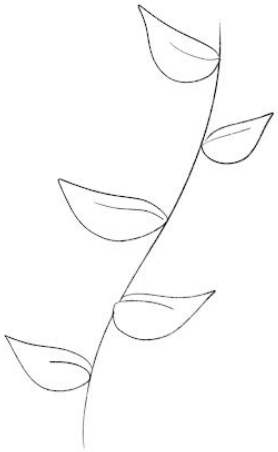
A crowd has formed around the BestFour by now.

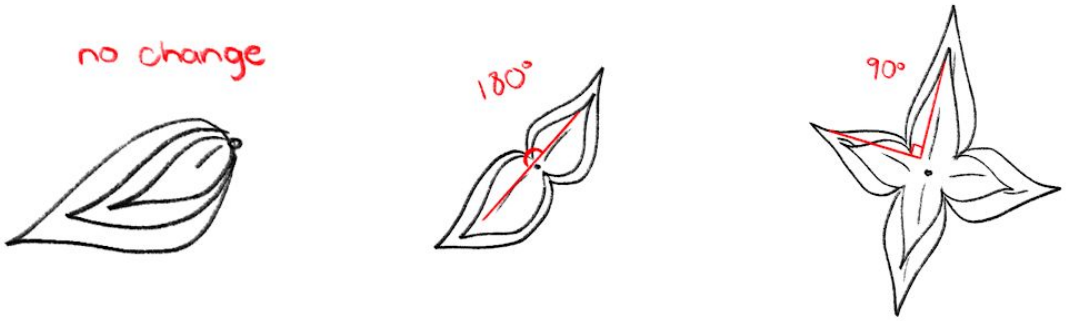
“Look at the branches on the tree!” someone shouts, “The branches follow the same 137.5° angle to diverge from each other!” [4]

You might ask: why do plants follow this magical angle and hence form Fibonacci number of spirals (3, 5, 8, 13, etc.) for their growth? In fact, this was the question raised many times in history.

To decode nature’s mystery, let’s experiment with some hypothesis:

Leaves overlap and get little sunlight when they separate apart with these angels...





Or 60? 30? 10? 360/n degrees?

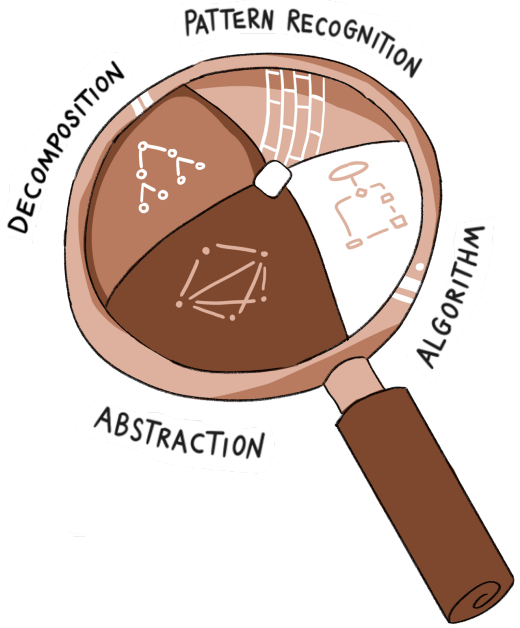
Is there a magic angle so that leaves will never overlap?

Golden angle Ψ ≈ 137.5077°

This is the exact angle we observed on pine cones, flowers and tree branches!

Figure 1.8 Golden angle to prevent leaves from overlapping

Ha, nature loves Fibonacci and the Golden Angle 137.5°!



"Then, how do we compute the Fibonacci number sequence?" One of the farmers asks.

Bubble Gum points to the lens. "It will tell."

Wow…

- Say function fib(n) is used to compute the nth Fibonacci number.
- Per Fibonacci definition, the function calls itself to add the previous two numbers together fib(n-1) + fib(n-2).
- When we reach the first two numbers in the Fibonacci sequence (0 and 1), the function will stop the calling chain and return 0 and 1 respectively.

Before getting down to coding, we'll draw a picture of the algorithm. This picture is called **Flowchart**. It helps us sort through and visualize the thinking process. We may realize some defects during the drawing process, and hence have a chance to fix them before coding.

Codephile

If coding is coffee, the flowchart will be the coffee mate!

Flowchart Illustration [6][7]:
Rectangle: activity or operation of the algorithm

Diamond: decision, conditional operation to show which path to take next
Rhomboid: input/output of data, e.g. take user input, display result on screen
Lines and arrows: order and relationship of the steps
Rounded Rectangle: start and end of the process

We read flowcharts from top to bottom and left to right.

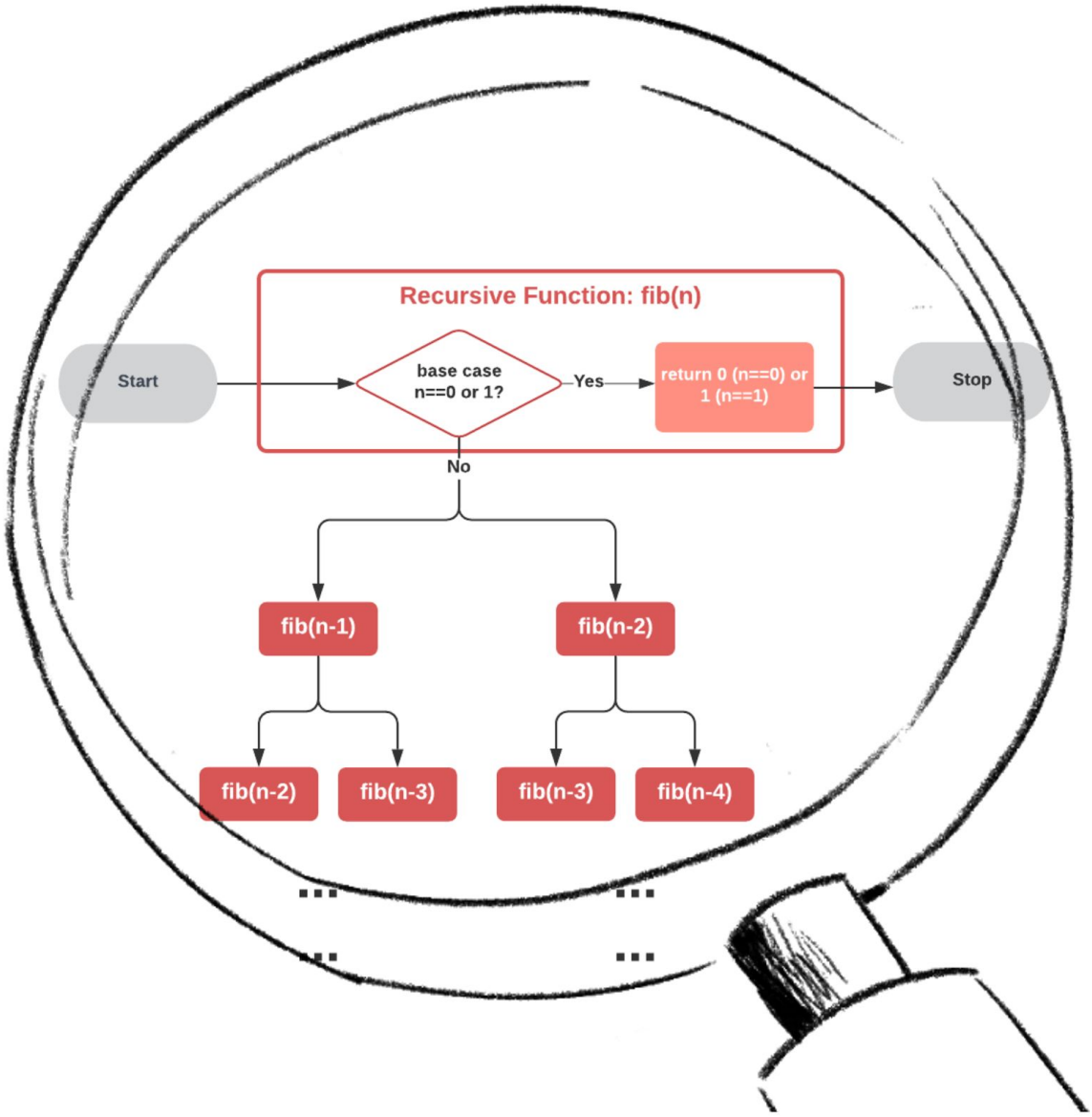
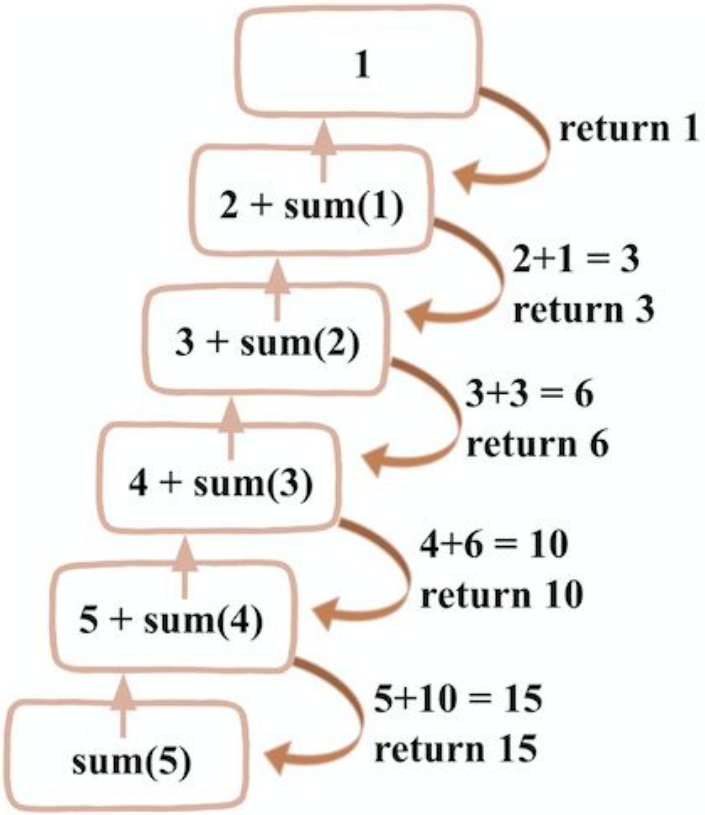


Figure 1.9 Flowchart: recursive function to solve Fibonacci number sequence

Recursion

A function that calls itself and stops the chain at the <u>base case</u>



sum() starts from the bottom of the stack, winding up to decompose until base case, and unwinding down to compute the result.

Figure 1.10 Recursive function definition, recursive approach to solve sum()

As a comparison, let's present the regular iterative approach using a loop below.

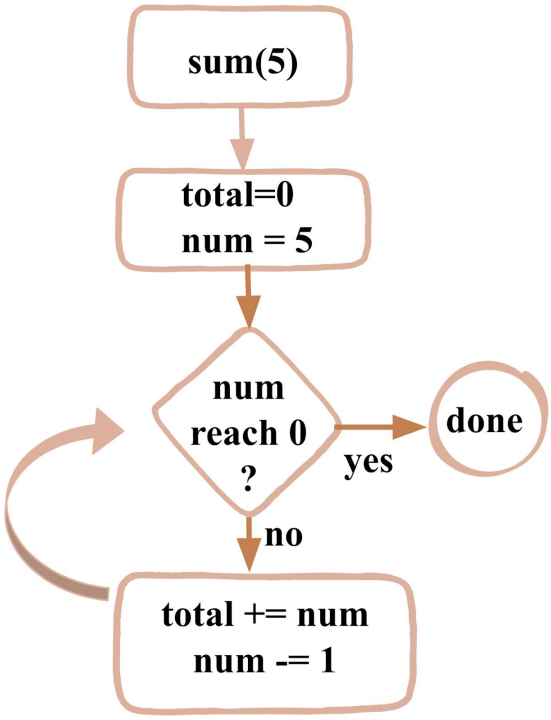
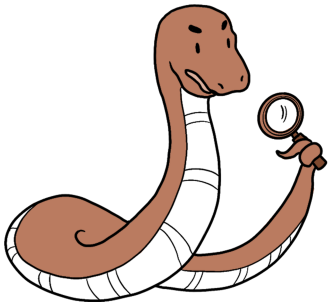


Figure 1.11 Iterative approach to solve sum()

Connecting with Reality

The first person who brought up the "definition by recursion" is Dedekind in his essay *Was sind und was sollen die Zahlen* (1888). He recursively defined addition, multiplication, and exponentiation [8]



Now let's look at the Python solution using a recursive and iterative approach respectively.

Recursive Approach:

```
# Apple Pi Inc.
# Algorithmic Thinking
# Fibonacci Numbers
# Recursive Approach

# recursive function
def fibonacci(n):
    # base case to stop the recursion call
    if (n==0):
        return 0
    if (n==1):
        return 1

    # compute fib number
    return fibonacci(n-1) + fibonacci(n-2)


for i in range(10):
    print(fibonacci(i), end=" ")
```

All seem good. However, let's say when we compute the 5th Fibonacci number, we first need to compute the 4th and 3rd numbers. Then by the time we compute the 4th number, we need to compute the 3rd number again. There are repetitive computations in this recursion process. Is there a way to remove the repetition?

An iterative approach will not have repetitive computation!

```
# Apple Pi Inc.
# Algorithmic Thinking
# Fibonacci Numbers
# Iterative Approach

# iteration while loop
def fibonacci(n):
    # two initial numbers
    # in Fibonacci sequence
    a = 0
    b = 1

    i = 0
    while i < n:
        next_num = a + b
        a = b
        b = next_num
        i += 1

    return a

for i in range(10):
    print(fibonacci(i), end=" ")
```

Yes, the iterative approach does not have repetition. Still, Bubble Gum's mind is tinkering with the recursion. Is there a way to optimize it?

Selling CalliLens: Coin Change Challenge and Greedy Approach

By this time, CalliLens is getting so popular at Greedy-ent Mart. We can sell it for a good deal so that we'll have cash to buy food and other necessities!

"$100", "$200", "300", "320"! The BestFour's eyes begin to sparkle with gold.

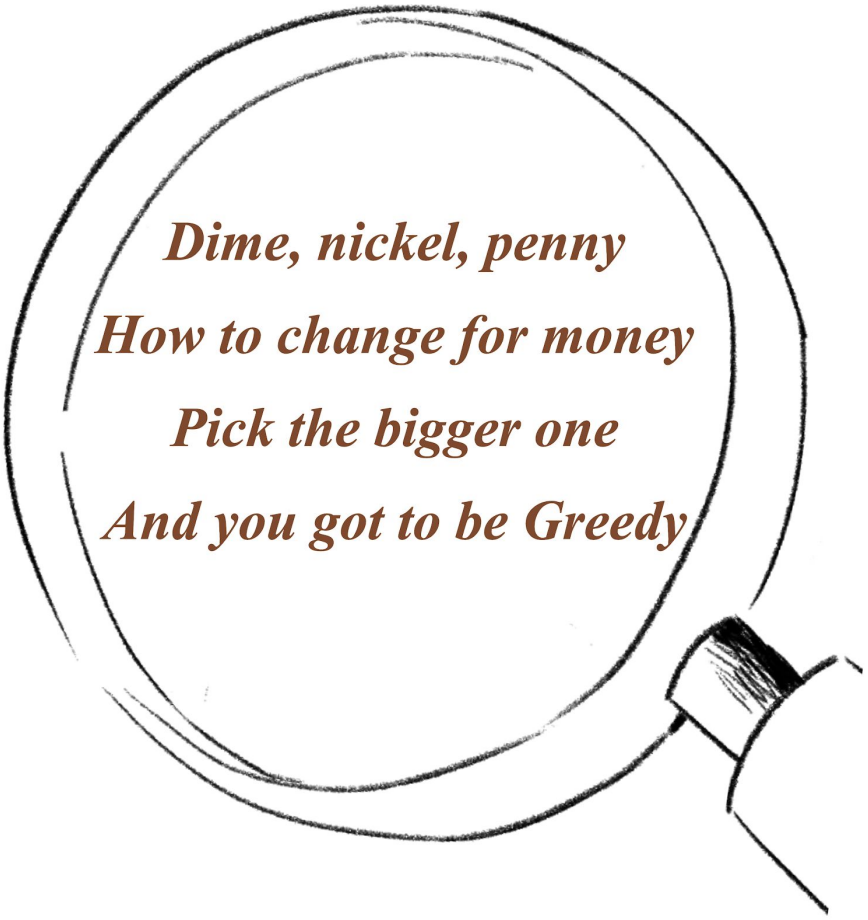
"Ok, ok, why don't we average the four prices and sell it for $230?" Dark Knight suggests the final call.

A farmer in front of the line immediately takes out a huge sack of money and begins piling up the pennies.

"Stop, stop," Bubble Gum urges, "this will be too heavy for us to carry. Please give us the least number of bills and coins!"

The farmer shuffles the paper bills and coins back, "Well..." He forwards in his hands, "This way, oh, that way is better… and yet, a better way…"

Banana Split hints to him. "Now you are empowered with the CalliLens. Why don't you look through it for the solution?"


Dime, nickel, penny
How to change for money
Pick the bigger one
And you got to be Greedy

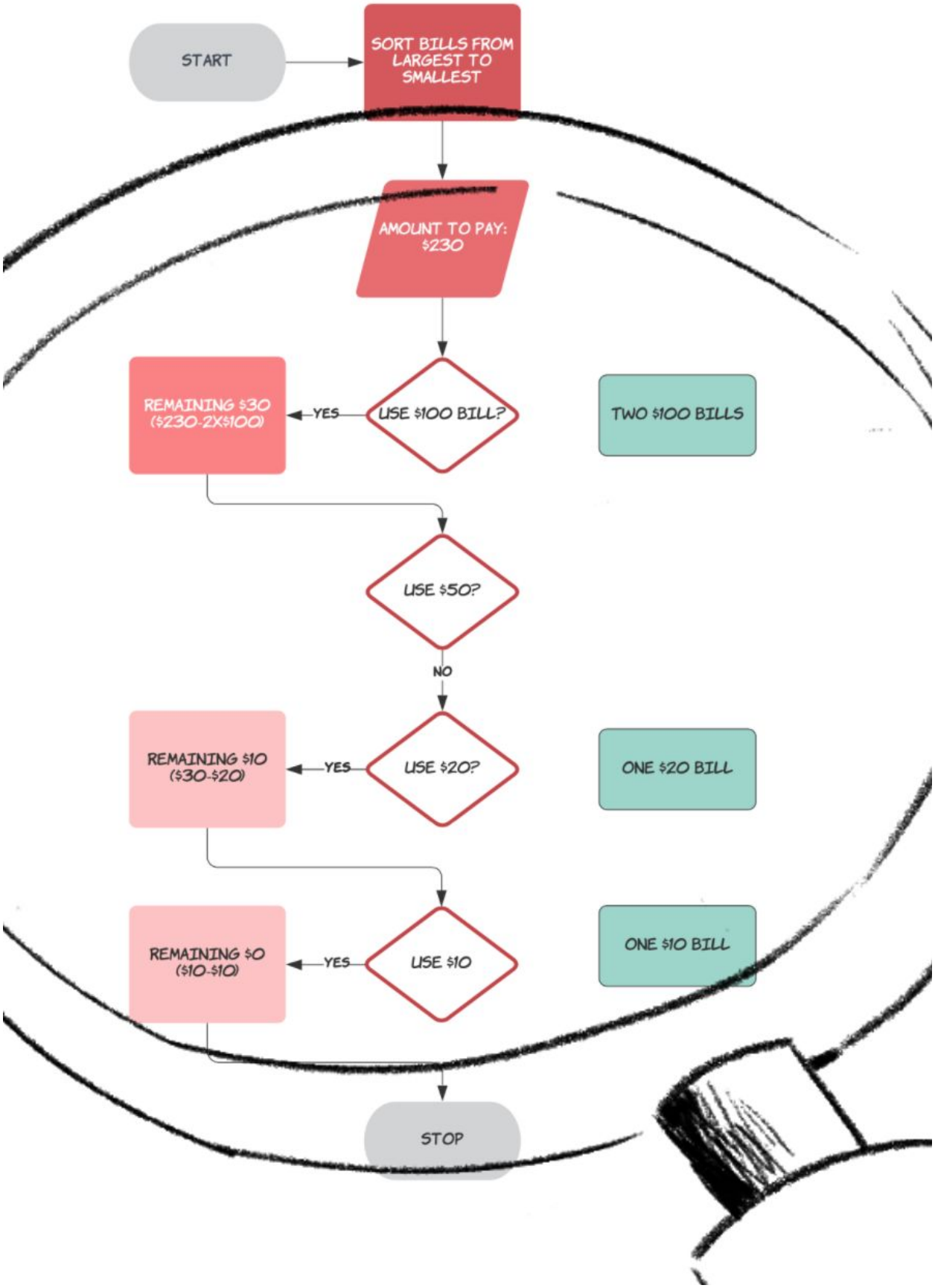

START
SORT BILLS FROM LARGEST TO SMALLEST
AMOUNT TO PAY: $230
USE $100 BILL?
YES
REMAINING $30 ($230-2X$100)
TWO $100 BILLS
USE $50?
NO
USE $20?
YES
REMAINING $10 ($30-$20)
ONE $20 BILL
USE $10
YES
REMAINING $0 ($10-$10)
ONE $10 BILL
STOP

Figure 1.12 Flowchart: Greedy Approach to solve Coin Change Problem

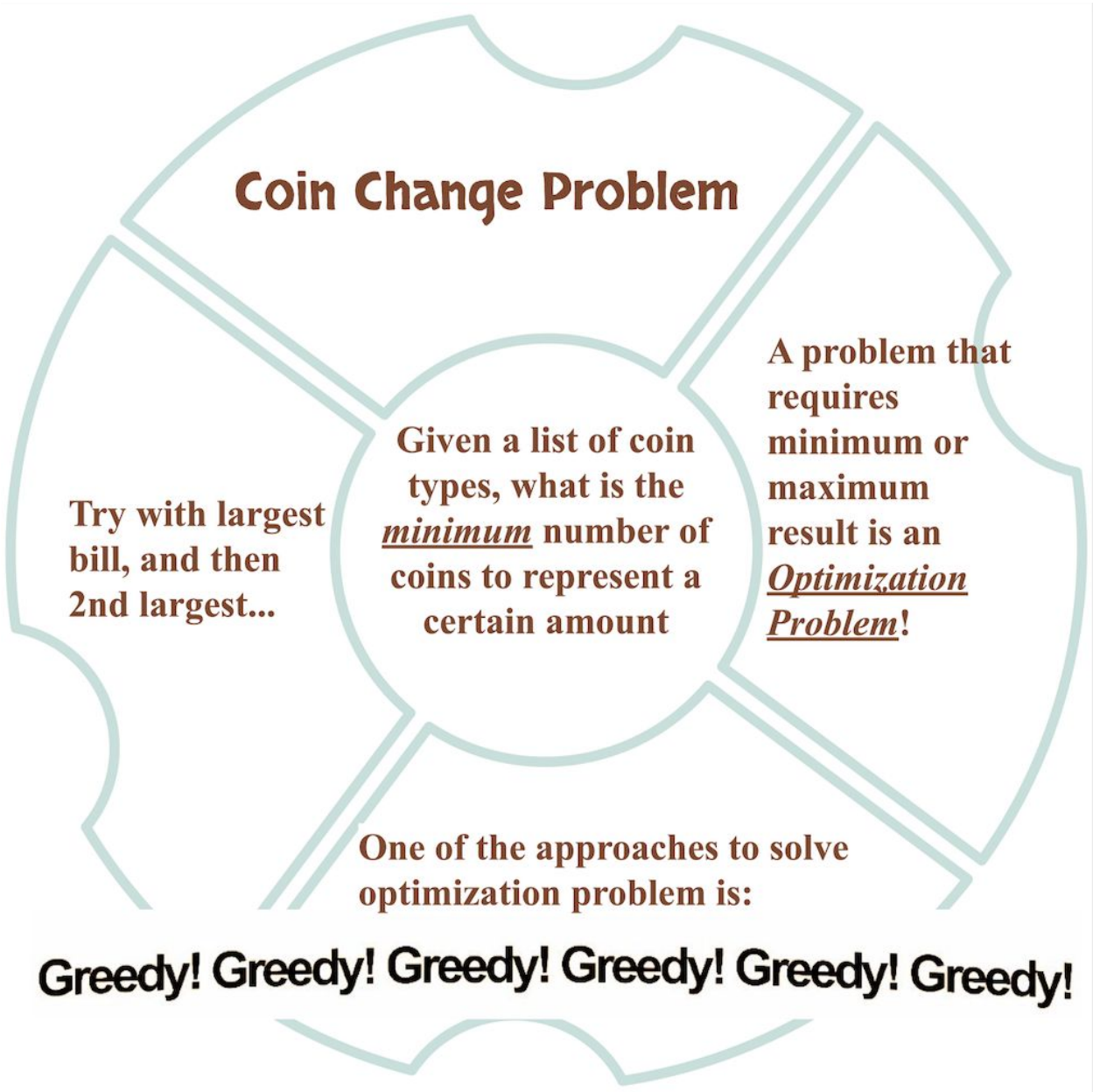


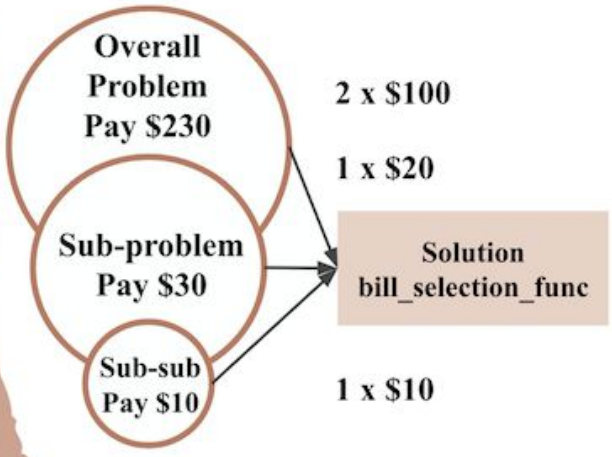
Figure 1.13 Coin Change Problem definition and characteristics

Greedy Approach

One of the approaches to solve optimization problems. Greedy works for the problems with these two properties:

Optimal Sub-Structure

The optimal solution to the local context (i.e. sub-problems) is part of the optimal solution to the overa problem.



Greedy Property

Greedy is 'short sighted'. It makes optimal choice at each step based on the local context with the hope of eventually reaching the globally optimal solution. It never reconsiders the previous choices.

Figure 1.14 Greedy Approach and its two properties

On the other hand, if the optimization problem does not have the above properties, we need to consider other approaches such as Dynamic Programming (DP) (please refer to Day 2 Adventure in the book).

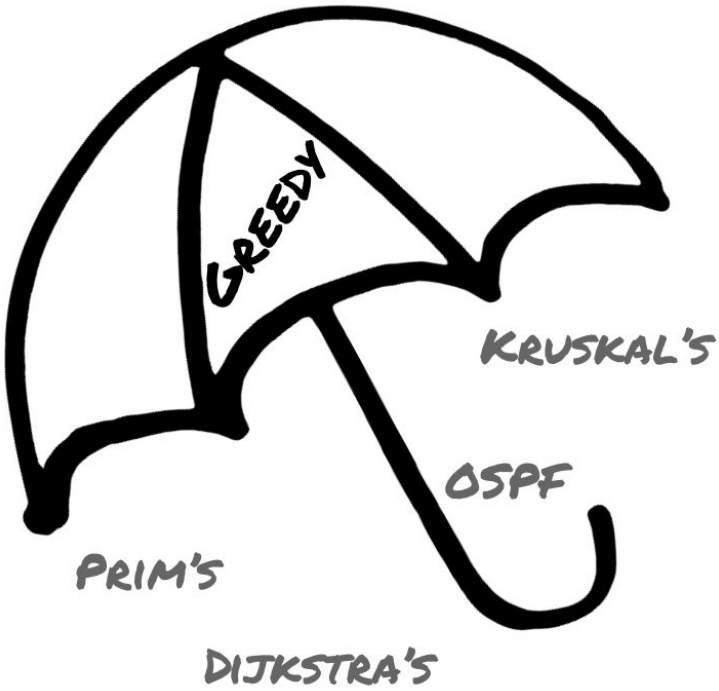


Figure 1.15 Greedy is an umbrella term for many algorithms

- Algorithms to walk through graphs, like Prim's and Kruskal's algorithms to minimize path costs across multiple locations (Day 3 Adventure in the book).

- Dijkstra's algorithm to find the shortest path from a given source location (Day 4 Adventure in the book).

- Network protocols such as the open-shortest-path-first (OSPF) and other network packet switching protocols to minimize time spent on a network.

Coconut Nut and Macadamia Nut: Fractional Knapsack Problem

"Now we have enough cash from selling CalliLens. I can't wait for the food quest to satisfy my empty stomach!" Banana Split states as he is about to run to the food stands.

"Don't bloat your stomach!" Dark Knight calls from behind.

"How can I get the highest satisfaction with my stomach?"

Stomach Capacity: 100g

She can take a portion or one full serving

Table 1.1

Fractional Knapsack Problem: nuts value and weights

Food Type	Honey Roasted Peanuts	Salted Sunflower Seed	Coconut Water	Chocolate Hawaii Macadamia Nuts
Satisfaction Value	90	45	140	90
Grams Per Serving	30g	30g	80g	20g

Can you help Banana Split?

What? See through the CalliLens and it will tell?

You are smart!

The abstraction of this "Banana Split filling his stomach" problem is a typical Fractional Knapsack problem. In this case, we try to maximize the total satisfaction value. The knapsack constraint is the stomach's capacity.

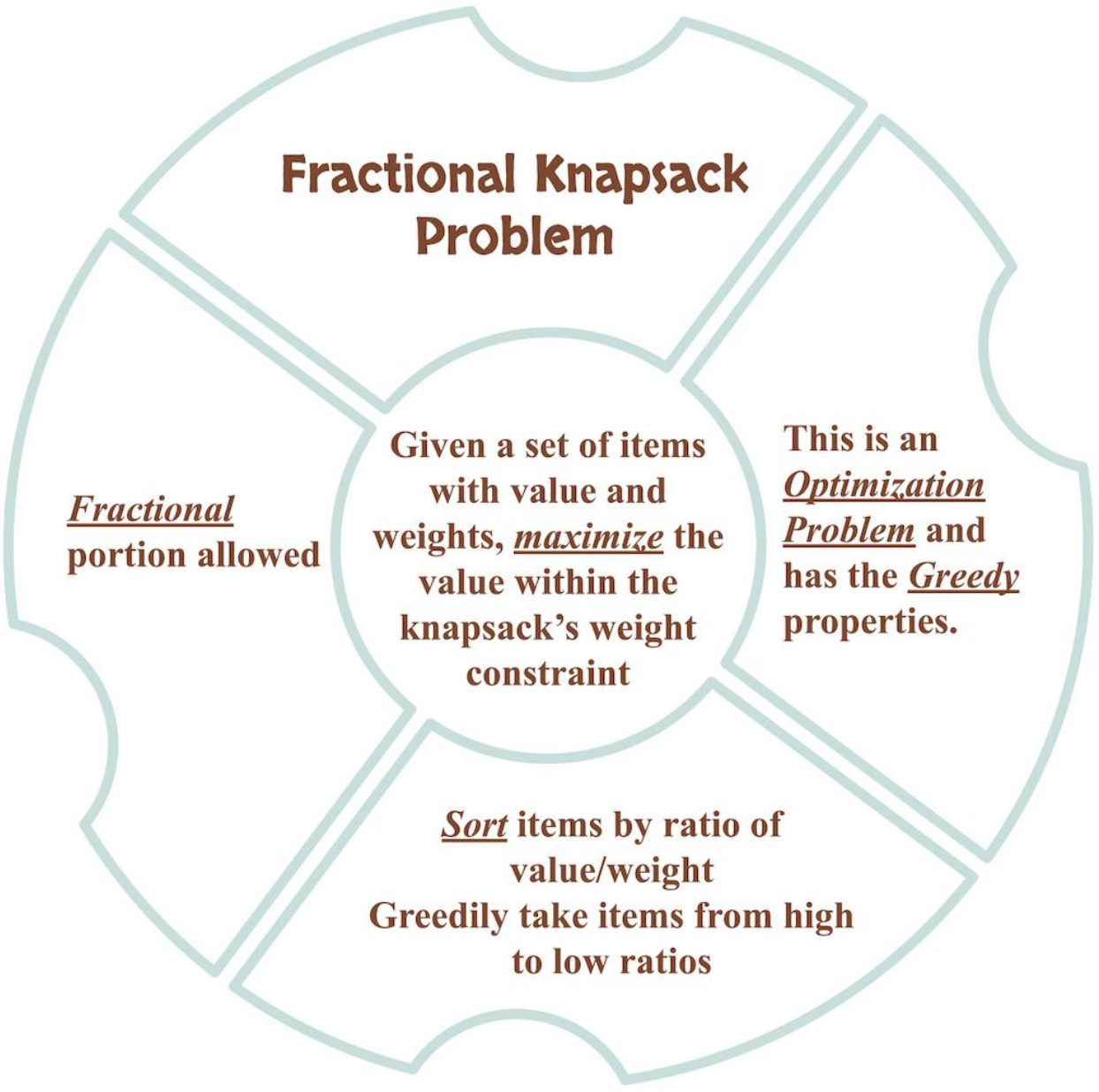


Figure 1.16 Fractional Knapsack Problem definition and characteristics

Table 1.2

Fractional Knapsack Problem: sort items by ratio of value/weight

Food Type	Honey Roasted Peanuts	Salted Sunflower Seed	Coconut Water	Chocolate Hawaii Macadamia Nuts
Satisfaction Value	90	45	140	90
Grams Per Serving	30g	30g	80g	20g
Satisfaction Per Gram	**3**	**1.5**	**1.75**	**4.5**

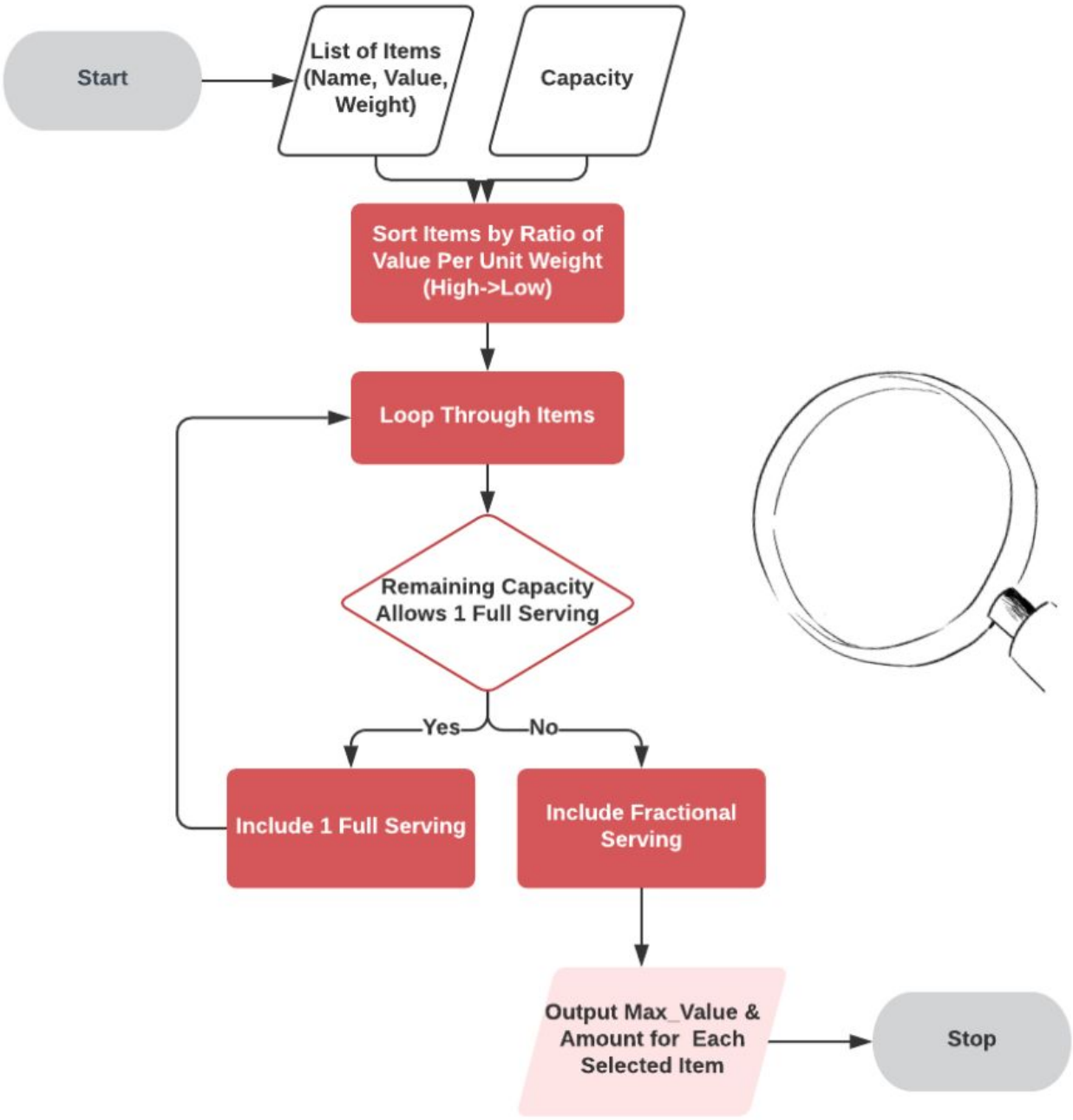


Figure 1.17 Flowchart: Greedy Approach to solve Fractional Knapsack Problem

Mighty Python

fractional_knapsack_greedy.py

```python
# Apple Pi Inc.
# Algorithmic Thinking
# Fractional Knapsack Problem
# Greedy Approach

class Item:
    def __init__(self, name: str, value: int, weight: int):
        self.name = name
        self.value = value
        self.weight = weight

# return (max_value, fractions)
#   max_value: the maximum value of selected items within capacity
#   fractions: dictionary showing selected items and the serving fractions (0,1]
# argument items: choices
# argument capacity: the maximum weight
def fractional_knapsack(items, capacity):

    # reverse sort the items by value/weight ratio (high ratio on top)
    # lambda represent a small function without name (called anonymous function)
    # format 'lambda arguments: expression'
    # here, the argument is each item in the items list
    # the expression is the ratio
    # use this lambda function to sort items by ratio in reverse order
    items.sort(key=lambda item: item.value/item.weight, reverse=True)

    max_value = 0
    fractional_amt = {}

    # Greedy approach to take high value/weight ratio items by order
    for item in items:

        if item.weight <= capacity:
            # include the whole serving
            fractional_amt[item] = 1
            max_value += item.value
            capacity -= item.weight
        else:
            # include partial serving
            fraction_to_add = capacity/item.weight
            fractional_amt[item] = fraction_to_add
            max_value += item.value * fraction_to_add
            break

    return max_value, fractional_amt

```

```
# initialize item list and capacity
items = [Item('Honey Roasted Peanuts', 90, 30),
         Item('Salted Sunflower Seed', 45, 30),
         Item('Coconut Water', 140, 80),
         Item('Chocolate Hawaii Macadamia Nuts', 90, 20)]
capacity = 100

# call knapsack function
max_value, fractions = fractional_knapsack(items, capacity)

# output result
print('The maximum value of items that can be taken within capacity:', max_value)
for item in fractions:
    print("item: " + str(item.name) + ', ', end='')
    print("amount: " + str(fractions[item]))
```

Bingo! The stomach is happy and so is Banana Split.

The sky begins to turn black, and the stars are now out. The night is approaching. The BestFour settles in the barn at Greedy-ent Mart.

chirp chirp chirp

“Hurry, wake up!” Dark Knight exclaims, “Today we need to find the map so that we can locate the Gate of Summit and the Gate of Traceback!”

“Sir, do you know where to get the map of the Land of Apple Pi?” He asks a favor from the owner of the nearby newspaper stand.

Pointing to the open land between the mountains, the owner replies. “Centermount Academy.”


HOURS
M – W – F –
T – TR – –
You're going to get a speeding ticket!
certainly!
?