Greedy In Farmer's Market

**Mission to Travel to the Past** **1**

Day 1 Adventure: BestFour in Atrophy Land **3**
- Abstraction of the Real World Through CalliLens 4
  - Magic Spirals 4
  - Decode the Nature 12
- Pattern Recognition and Solution Through CalliLens 14
- Introduction to Recursion 15
- Mighty Python 17
  - Iterative Approach 18
  - Recursive Approach 18

**Greedy Method** **20**
- Coin Change Problem 20
  - Diagram Illustration 22
- Feasible Solution vs. Optimal Solution 24
- Greedy Approach to Solve Optimization Problems 24
  - Abstraction 27
- Fractional Knapsack Algorithm 27
  - Diagram Illustration 27
  - Mighty Python 29

# Mission to Travel to the Past

"Look what I found in Grandpa's treasure box?". Dark Knight gallops to his clan with two pairs of glistening glasses.

Hurtling to Dark Knight, the BestFour starts a close examination on the glasses. A line of fine print jumps to Banana Split's eyes, "CalliLens welcomes you to the crystal world."

"CalliLens?" Bubble Gum murmurs, "I heard of HoloLens produced by Microsoft on that Zero-One Land. Their transparent lenses give you an augmented reality experience. When I am hungry, alfalfa hay rises in front of my nose. When I am thirsty, a river runs by my hooves. "

"Holo means three-dimensional image. Then what is this Calli for?" Banana Split scratches his head, "Calligraphy means beautiful writing. Calli means beautiful in Greek. Ha, I get it! We are going to see a beautiful world through these CalliLens."

"Give to me, " Dark Knight puts on the lens, "the road is straight, the stone is firm."

"Give to me, " Banana Split puts on the lens, "the bread is soft, the cake is yum."

"Give to me, " Bubble Gum puts on the lens, "the dream is big, the reality is warm."

"What are these mysterious CalliLens?" they turn to Mighty Python, the immigrant from the Zero-One Land. Mighty Python lifts his head and gazes towards the horizon and starts explaining:

CalliLens has the magic power. When you are scared by a monster problem, wear this gadget, you will see

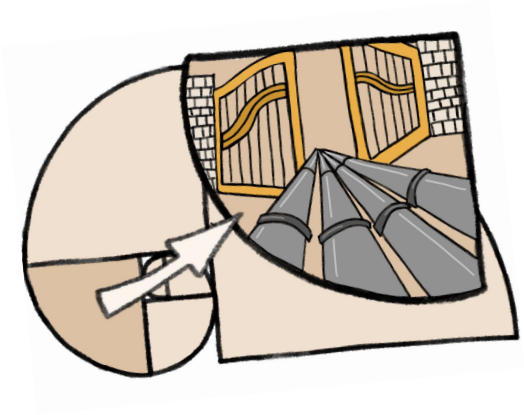
1) The monster is shrinked and decomposed to some miniature problems. You will be able to manage each miniature very well.
2) Spot the patterns or commonalities among the closely related problems and ignore the details that do not matter.
3) The step by step instruction on how to attack the monster will emerge in front of your eyes.

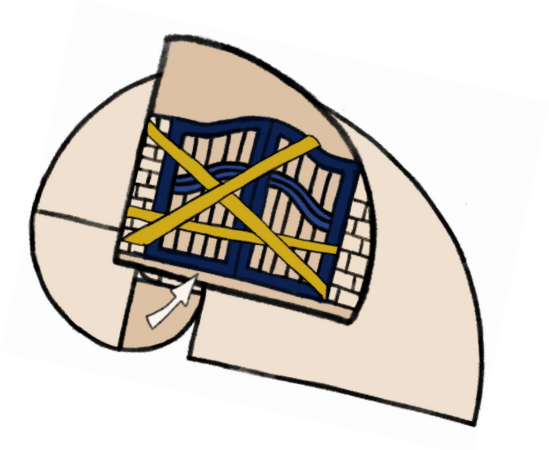
"Who is having a monster problem to attack?" Mighty Python waves the CalliLens in the air.

"Me!" Banana Split hustles, "Mom asked me to clean my room today. That is a monster task! Where to start?"

"Oh, CalliLens shows that go and put clean clothes in a pile, dirty clothes in the laundry bin, and books on the shelf." Bubble Gum feels better, "I can manage these three small tasks. "

Mighty Python goes on:
Around 100 years ago, the world was full of inventions and people started "transforming" the planet with their intelligence. However, some of the inventions were driven by people's greed to possession, greed to wealth, greed to win. They accelerated the aging process for human beings and the planet. There is only one chance to reverse the aging (everyone is holding their breath)...





- Bring back the CalliLens to the past to traverse through the chiaos
- Seal the Gate of Traceback, the source of the toxic inventions
- Build pipeline from the Gate of Summit, the source of righteous inventions

"The coming week is a holiday. Let us secretly travel back and reverse the aging!" announced Dark Knight.

"Grandpa, we are going camping, just regular camping in the woods..." The BestFour waved good-bye to the family and off on the road (to be precise: time vessel).

## Day 1 Adventure: BestFour in Atrophy Land

Hovering on top of the Atrophy Land, the BestFour saw the little ants crawling along some curved lines. The lines are intertwined.

One word to describe "Entropy"!

Mysterious, restless...

Following a noice cloud, they landed at a market place.

There are so many mysterious and attempting foods. Banana Split reaches out for a plate of nuts.

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

“Grandpa showed us once in his treasure box,” Dark Knight recalls, “ coins and bills are physical currency used for goods exchange.”

Banana Split zooms close to the cashier’s box, “well, can I pay with my fingerprint, with facial recognition, retina recognition, with any sort of biometric identification? We don’t use physical currency anymore.”

“What?! “ the cashier stares at the “aliens”.

“I have an idea: we can sell what we have to exchange the physical currency here.” Dark Knight announces.

“We brought two pairs of CalliLenses. What about selling one?” Bubble Gum takes off her lens.

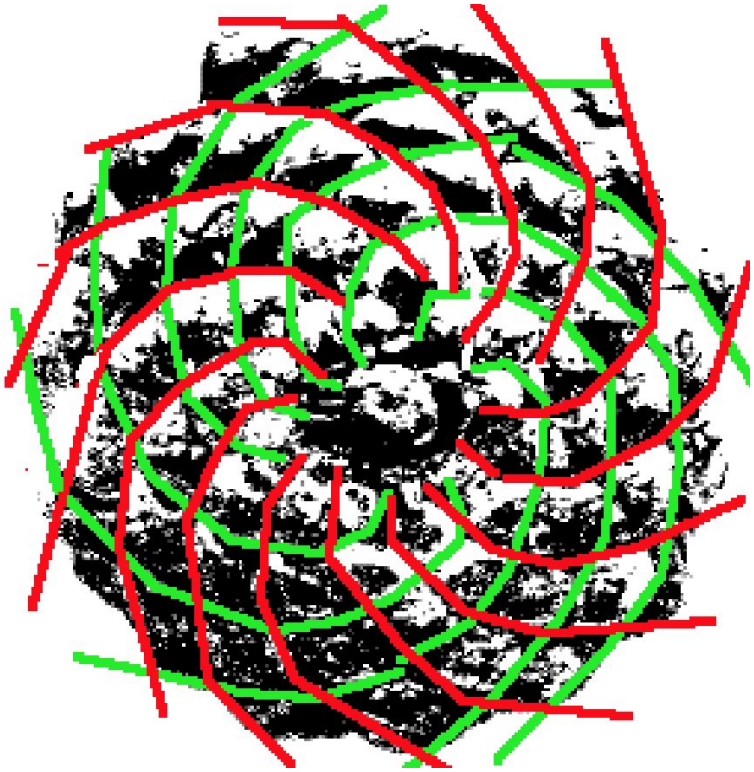
## Abstraction of the Real World Through CalliLens

### Magic Spirals

A farmer comes by and tries the lens on.

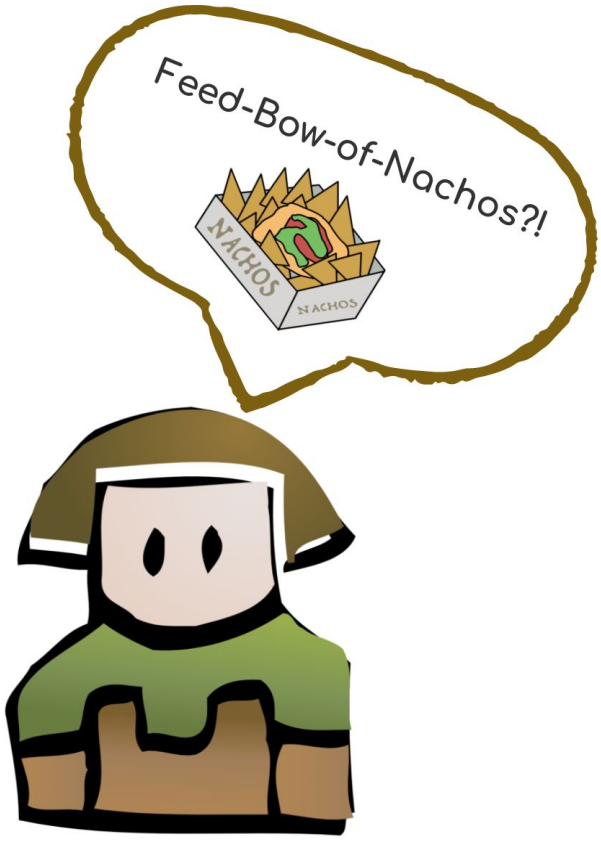
“What happened to my Romanesco broccoli? I see 5 spiral lines formed by its pinnacles curling to one side, and 8 spiral lines curling to the other side. “

"Let me try the magic lens." a neighbouring farmer overhears the conversation and comes by.
"OMG, this is my pine cone. It has 8 green spirals and 13 red spirals, Fibonacci number!"

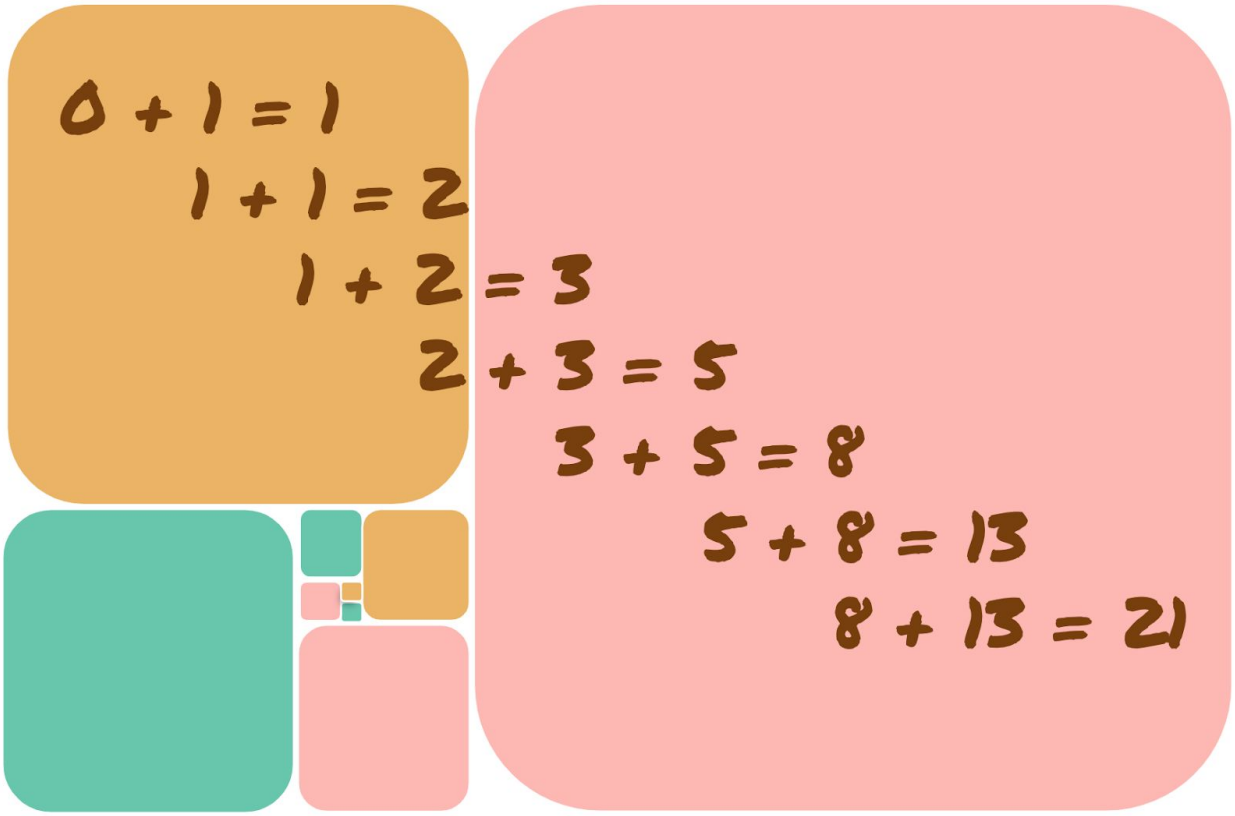


http://www.mit.edu/~levitov/FibonacciPhyllotaxis.pdf

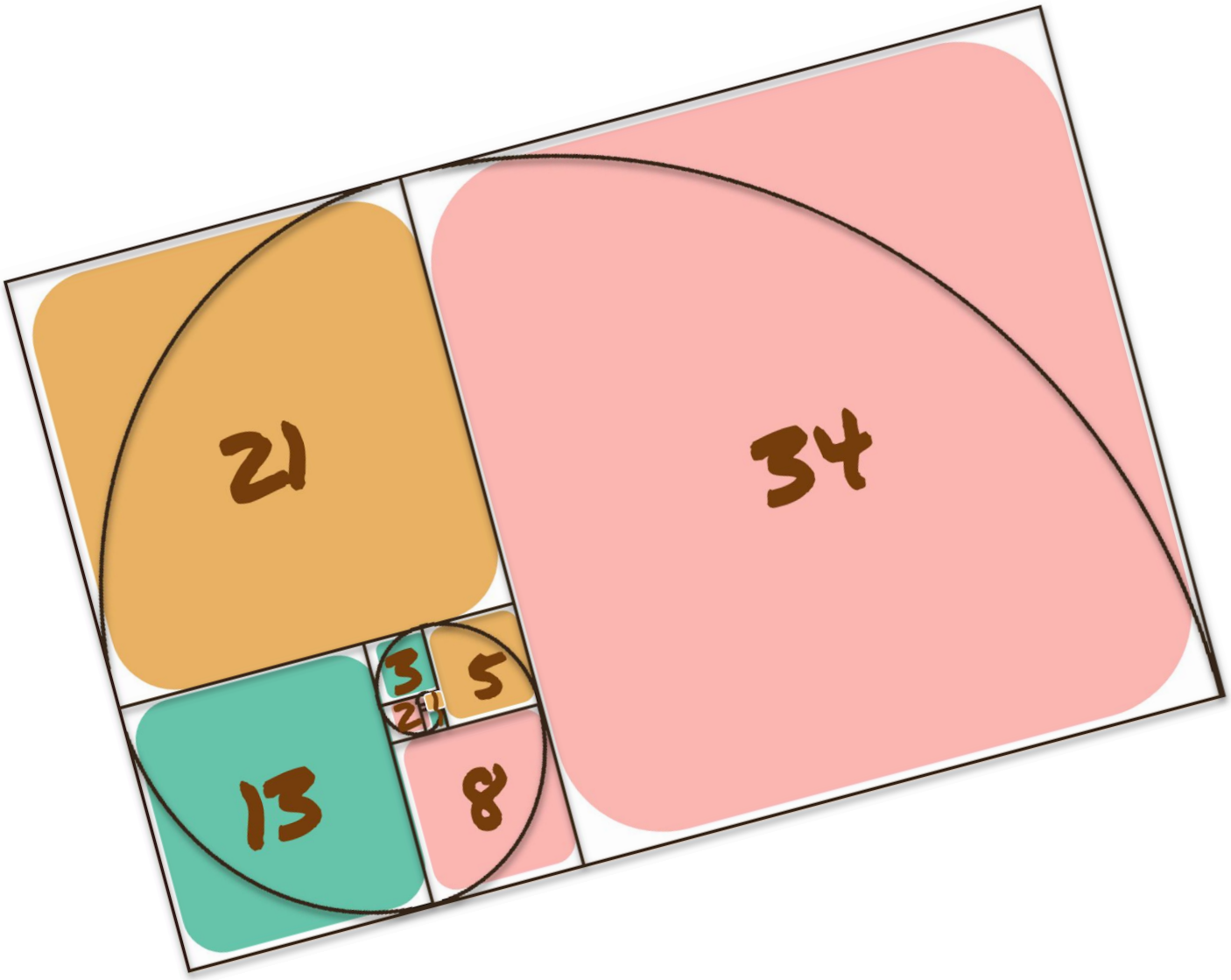
"Oh, 5, 8 and 13 are numbers from the Fibonacci sequence." Banana Split jumped in.




“Fibonacci is a series of numbers in which each number is the sum of the two previous numbers.”




If you puzzle the squares with the Fibonacci side length together, it forms a spiral.




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 (Wells 1986, p. 62). [2]

[2] Wells, D. *The Penguin Dictionary of Curious and Interesting Numbers.* Middlesex, England: Penguin Books, pp. 61-67, 1986.

Therefore, this special spiral is called Golden Spiral. It gets wider (i.e. further from its origin) by a factor of 1.618 for every quarter turn it makes.

Reality Check

**Fibonacci Numbers?** Leonardo Fibonacci, the man responsible for re-discovering the amazing Fibonacci Numbers, was an Italian mathematician who was born somewhere around 1170. In 1202 he published a book called 'Liber Abaci' (Book of Calculation).

**He was also the man who gave us our decimal numbering system.** He figured out that it was far more efficient than the Roman Numeral System, which did not even contain a '0' - zero.

https://www.all-my-favourite-flower-names.com/fibonacci-numbers.html

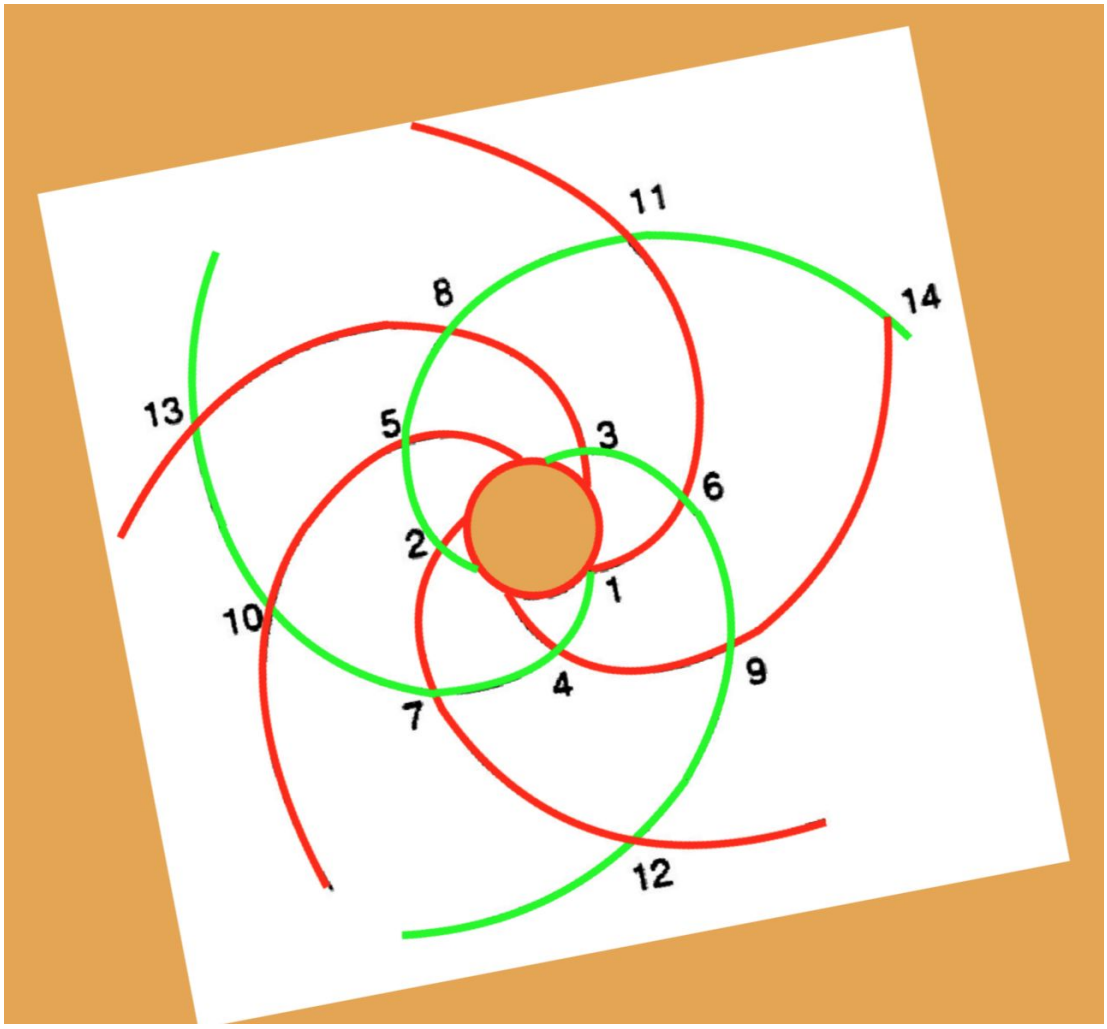
(OEIS A000045)

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

Reality Check:
Besides Finobanni, there are Lucas numbers which are also special: 1, 3, 4, 7, 11, 18, 29, 47, 76…

Statistics for cones of pine-trees (Norway):
95% Fibonacci
4% Lucas
1% deficient
http://www.mit.edu/~levitov/FibonacciPhyllotaxis.pdf

"Let me try." the neighbouring gardener is tuned into this conversation.
"OMG, this is my flower."

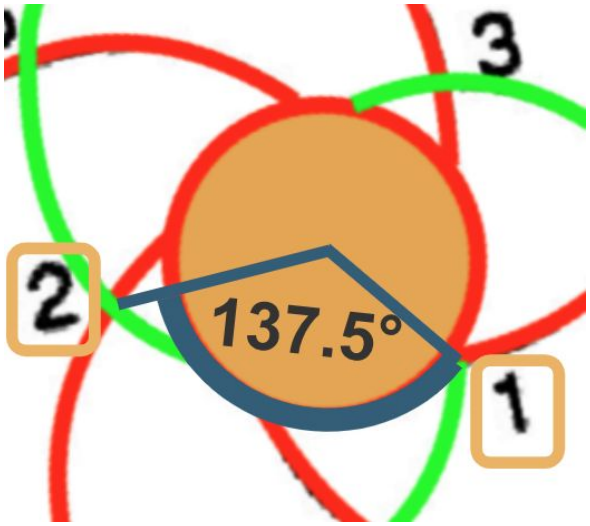



Smaller numbers represent younger petals.

3 green clockwise spirals and 5 counter-clockwise red spirals. Again, 3 and 5 are Fibonacci numbers! And the spirals are golden spirals!

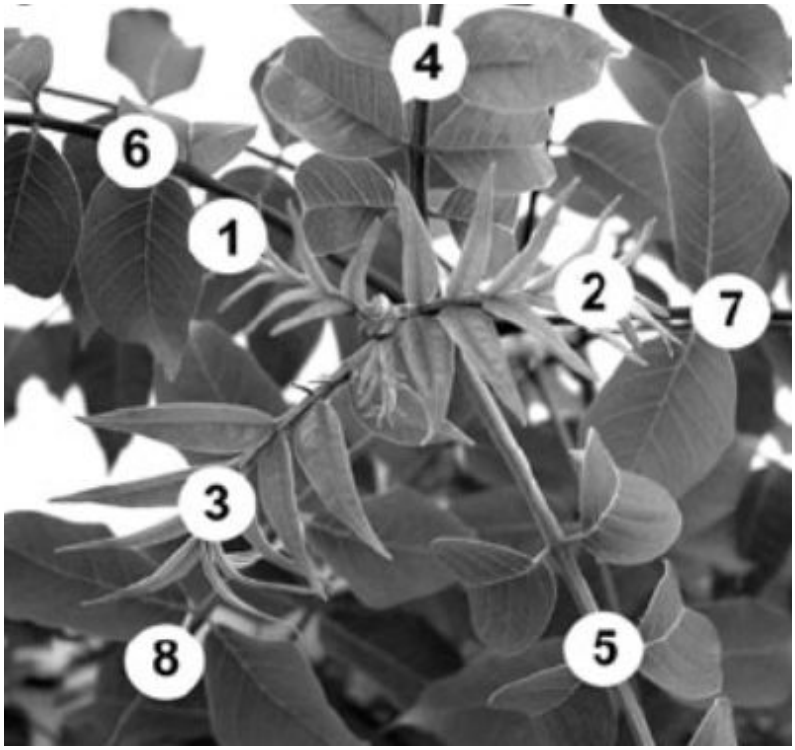
http://www.mit.edu/~levitov/FibonacciPhyllotaxis.pdf

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.

The crowd has formed around the BestFour by now.
"Look at the tree!" someone shouted.




Branches of one Ailanthus altissima tree. Smaller numbers represent younger branches

https://www.researchgate.net/publication/7663446_Regulation_of_phyllotaxis
The branches follow the same 137.5° angle to diverge from each other.

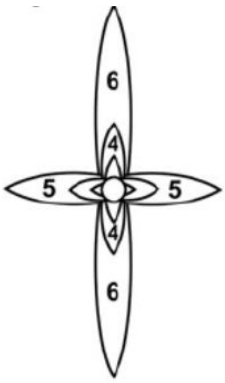
## Decode the Nature

You might ask: why do plants follow this magic 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 the nature's mystery, let us experiment some hypotheses:
What if the leaves grow at the same angle?

What if in 180 degree angle?


What if 90 degree angle?



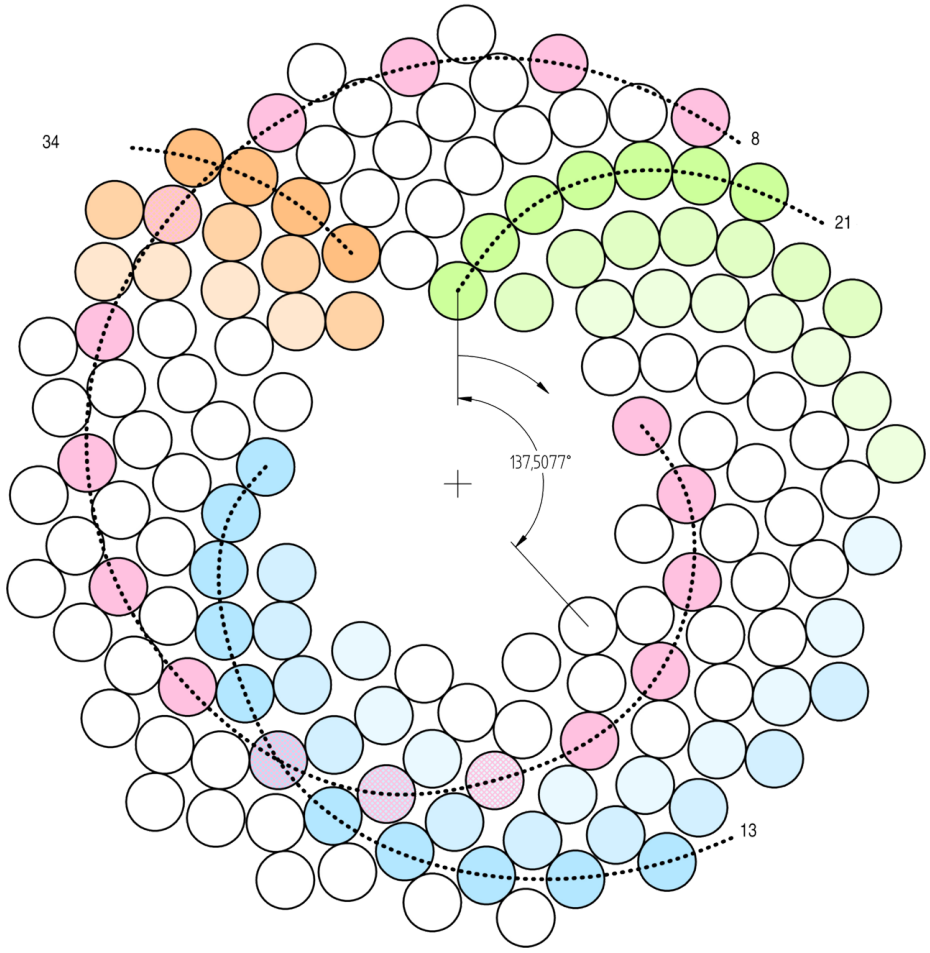

60? 30? 10? 360/n degrees?

Eventually, leaves will repeat the same cycle and overlap.

Is there a magic angle so that leaves will not overlap and get maximum sunlight?

Golden angle Ψ ≈ 137.5077°

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




Mathematical Abstraction

Arrangement of equal-sized circles at a distance of the golden angle Ψ ≈ 137.5077° with colored Fibonacci spirals

https://commons.wikimedia.org/wiki/File%3AFibonacci_numbers.jpg

Realty Check

Leonardo Da Vinci in his notebook 1503 stated, "Nature has arranged the leaves of the latest branches of many plants so that the sixth is always above the first, and so it follows in succession if the rule is not impeded."
http://www.mit.edu/~levitov/FibonacciPhyllotaxis.pdf

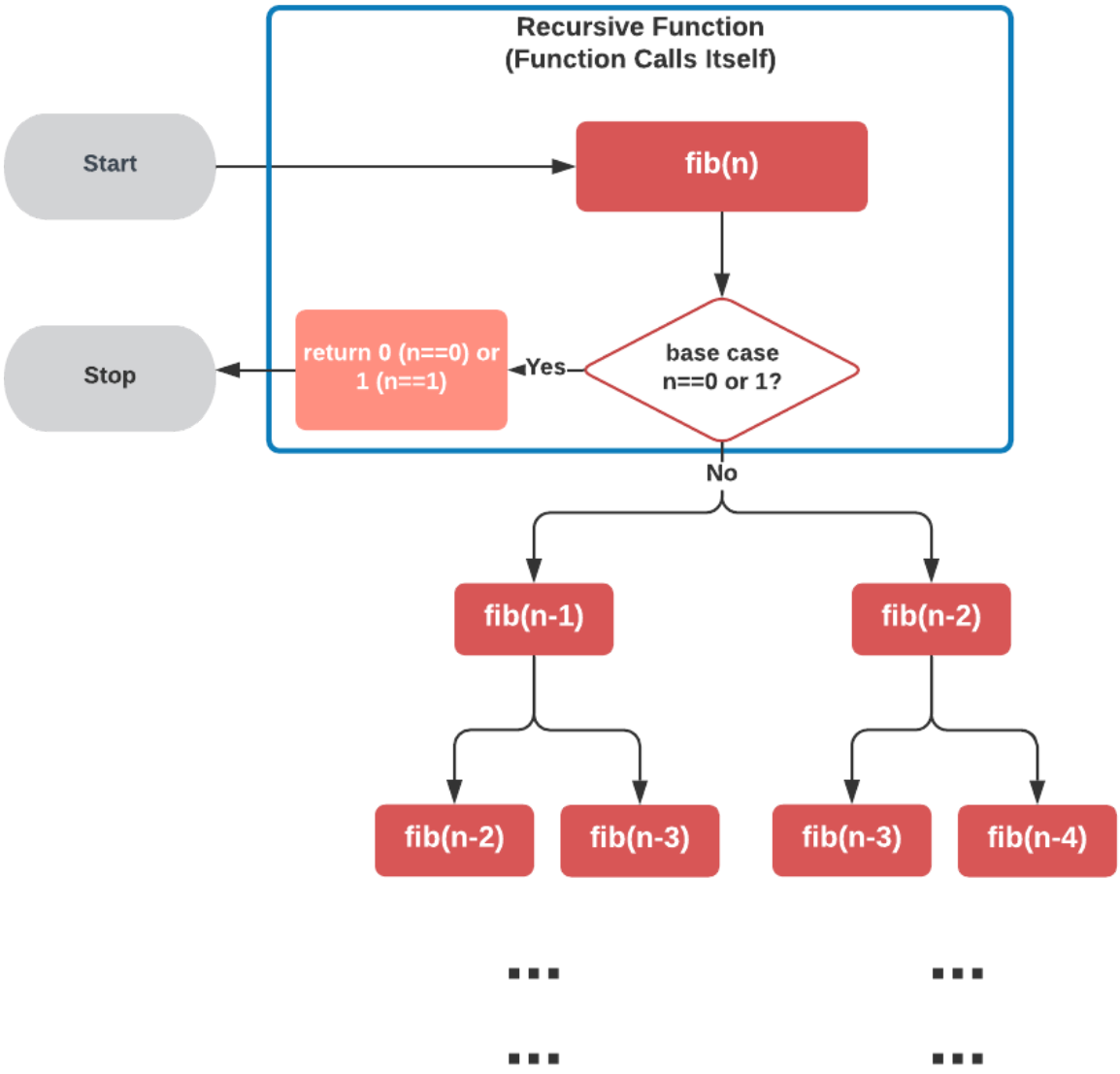
The total angle between the first and the sixth leaves is 5 x 137.5° = 687.5°. It is close to the end of the 2nd circle (360° x 2 = 720°), so the sixth leaf is not totally overlapping with the 1st leaf, so it can get sufficient sunlight!

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

## Pattern Recognition and Solution Through CalliLens

"Then, how to find the Fibonacci number sequence?" one of the farmers asks. Bubble Gum points to the lens, "it will tell…"

Wow…




0, 1, 1, 2, 3, 5, 8, 13, etc.

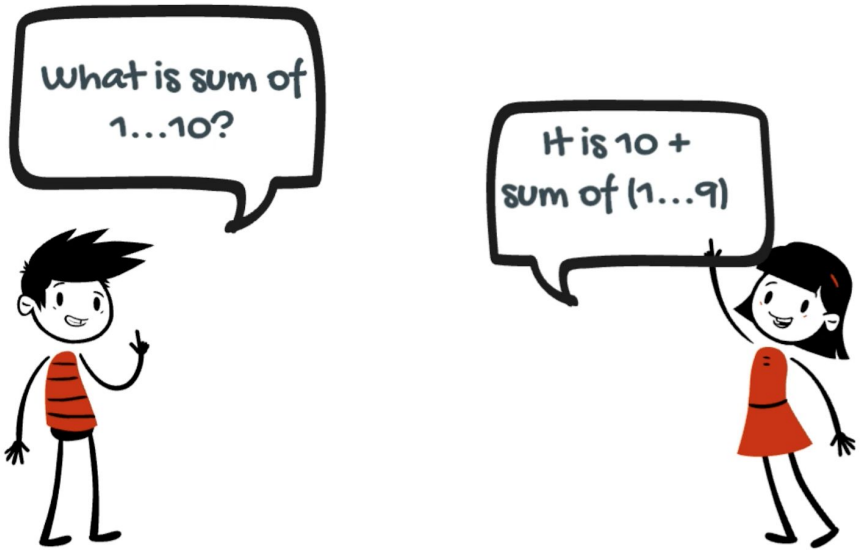
- Function fib(n) is called to compute the nth Fibonacci number.
- The first two numbers are predetermined, so the function returns 0 and 1 respectively.
- When n >=2, the function fib(n) computes the number by adding the previous two together.
- Well, how to find out the previous numbers, the answer is to call the same function fib(n-1) and fib(n-2)

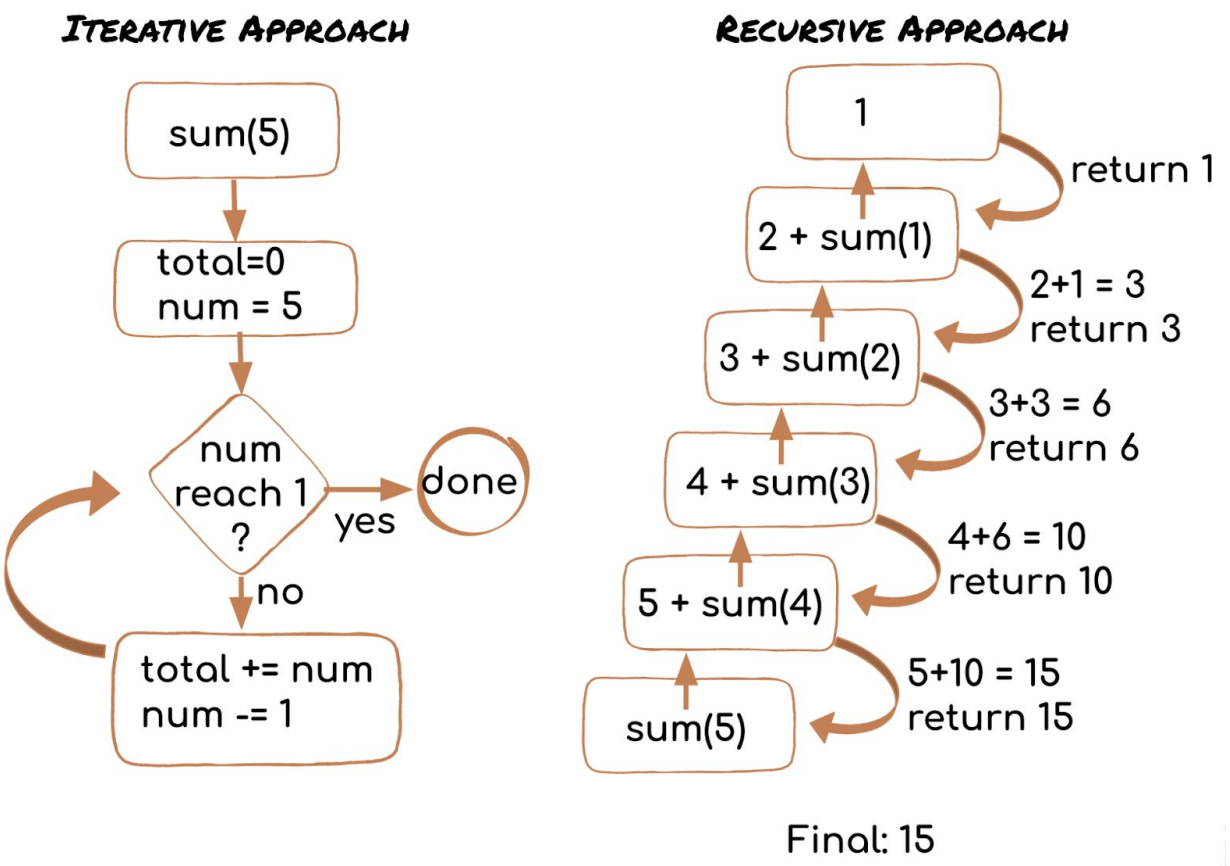
"A function that calls itself is called a Recursive Function." Banana Split added.

This is  truly a magic lens! It reveals abstraction of the real world, and also represents the steps to solve the problem.

## Introduction to Recursion

Before we compute the Fibonacci number sequence, let us understand recursive function with a simpler example: the sum of consecutive numbers.






As a comparison, we present the traditional iterative approach using a loop in the left of the diagram.

Whereas for the recursive approach in the right of the diagram, we start from the bottom of the stack. The winding up process is to decompose the problem to simpler and simpler cases. The unwinding down process is to compute the result, like a snowball rolling down the hill.

The operation of sum() is used within its own operation until it reaches a certain condition and stops the chain. Such a process is called **recursion**.
A function calling itself is a **recursive function**. The condition to stop the chain is called **base case**. In the above example, the base case is when the number reaches 1.

Reality Check

The first person brought up "definition by recursion" is Dedekind in his essay *Was sind und was sollen die Zahlen* (1888). He recursively defined addition, multiplication, and exponentiation [1]

Think about human beings growing generation after generation, and advancing technologies along the way: the steam machines in the 1st industrial revolution, electricity in the 2nd, computing in the 3rd, and currently artificial intelligence and robotics in the 4th industrial revolution...Will it stop at some point? Time will tell.

One thing for sure is that the toxic inventions will stop soon! Right, that is our mission!

## Mighty Python

Now let us look at the Python solution using an iterative and recursive approach respectively.

Iterative Approach

```
# 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=" ")
```

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=" ")
```

Introduction to Pygame

# Greedy Method

## Coin Change Problem


By this time, CalliLens is so popular on this market. We can sell a good deal!

"$100", "$200", "300", "320"...
"Ok, ok, why don't we sell it for the average of these four prices: $230?" Dark Knight makes the final offer.

The farmer in front of the line immediately takes out his money sag and starts piling up the pennies (1 cent coin).

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

"Well..." the farmer is shuffling the paper bills and coins back and forward in his hands, "this way, oh, that way is better... and yet a better way..."

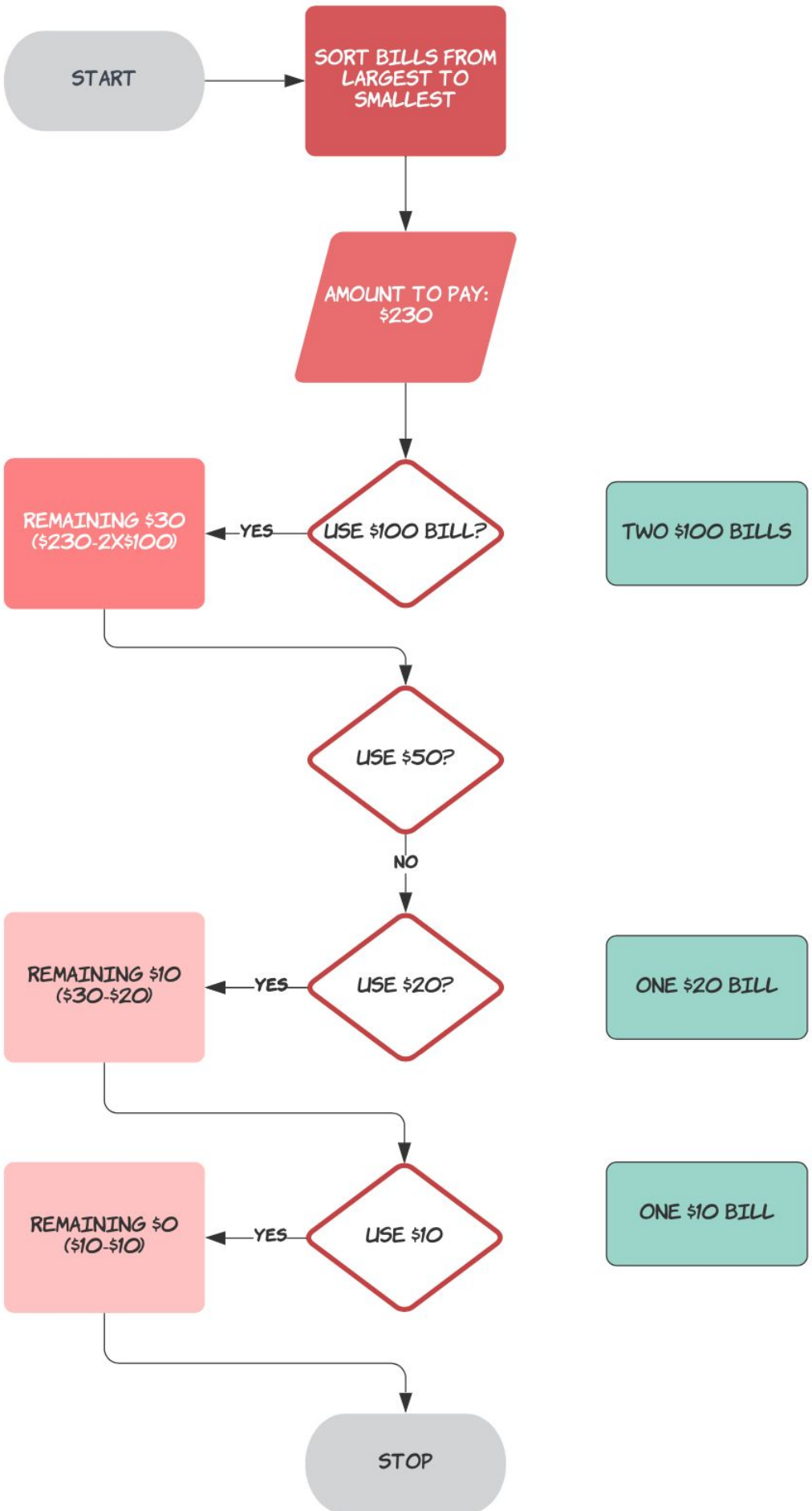
"Now you are empowered with the CalliLens. Why don't you look through it?" Banana Split hints him.

"Alright, I definitely need to be greedy and try to use the largest possible bill, and then 2nd largest..."

**Greedy! Greedy! Greedy! Greedy! Greedy! Greedy!**

**You're darn right. This is indeed called Greedy Approach!**

Diagram Illustration


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

We can generalize this problem as Coin Change Problem: given a list of coin types, what is the minimum number of coins that we must use to represent a certain amount? Assume we have unlimited supply of coins of each type.

## Feasible Solution vs. Optimal Solution

As we can see there are so many ways to pay $230:
One $100
Two $50
Three $10
Or
One $100
One $50
Four $20
Or
….
All these are feasible solutions because they all make up to $230.

But there is only one optimal solution:
Two $100
One $20
One $10

The problem requiring optimal solution is called Optimization Problem. It requires either minimum or maximum result, e.g. minimum number of bills, shortest path, maximum value, minimum travel time, etc.

## Greedy Approach to Solve Optimization Problems

Greedy is one of the approaches to solve **optimization** problems. This approach will work if the optimization problem has the following two properties:

- Optimal sub-structures

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.
The optimal solution to the local context (or called sub-problems) is part of the optimal solution to the overall problem.

- Greedy property

While it moves forward to the next decision steps, it will never have to reconsider the previous choices.

If the optimization problem does not have above properties, we need to consider other approaches such as Dynamic Programming (DP) (later in the book).
Greedy is an umbrella term for many algorithms.

Algorithm is a fancy term to describe a list of steps to solve a category of problems.

- Algorithms to walk through graphs, like Dijkstra's algorithm to shorten the span of routes from one location to another (later covered in this book).
- Prim's and Kruskal's algorithms to minimize path costs across multiple locations (later covered in this book).
- Network protocols such as the open-shortest-path-first (OSPF) and other network packet switching protocols to minimize time spent on a network.

"Now we have enough cash. I can't wait for the food quest to satisfy my empty stomach!" Banana Split is about to run to the foot stalls.

"Remember, you have limited stomach size, you got to be careful about your choices?" Dark Knight chases him behind.

"You have a point. Let me quickly survey the market and list my favorite food. I want to get the highest satisfaction with my stomach capacity." Bubble Gum says.

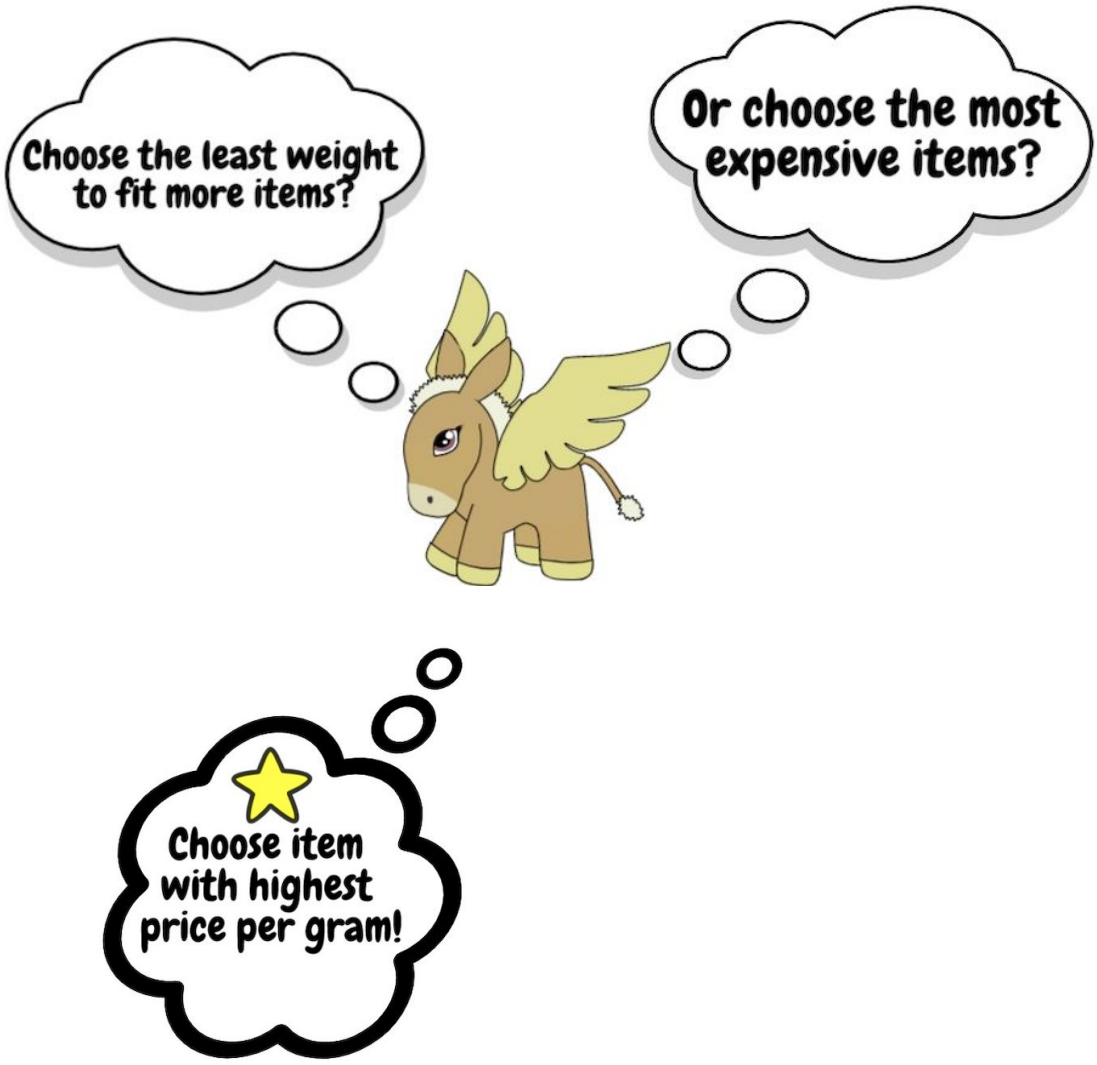
Stomach Capacity: 100g
Can take fractional up to 1 serving per type

| 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 |

[ToDo: quote Coconut song]

Banana Split can not fit in all his favorite nuts in his stomach, he will have to choose to maximize the satisfaction value. Maximization problem is an optimization problem. We will think of Greedy approach.

Can we rank the items and try one by one, just like what we did for Bill (Coin) Change when selling the CalliLens?
[ToDo: to change the drawing]




This is different from the coin change problem. Here we have weight limitation. This is a constraint.

https://www.codesdope.com/course/algorithms-knapsack-problem/

What is your idea to help Banana Split?

What? Wear CalliLens and it will tell. Did I hear you right? Smart! CalliLens will show the abstraction of the problem, patterns and the solution steps (i.e. algorithm).

[ToDo: to introduce the definition of algorithm when selling CalliLens]

## Abstraction

# Fractional Knapsack Algorithm

The abstraction of this Banana Split filling his stomach problem is a typical Fractional Knapsack problem: given a set of items with specific value and weights, the goal is to maximize the value in a knapsack (in this case total satisfaction value) within the weight constraint (Banana Split's stomach).

Since Banana Split can eat fractional serving of different nuts (e.g. half a serving of Salted Sunflower Seed), this type of Knapsack problem is called Fractional Knapsack Problem (or Continuous Knapsack Problem).

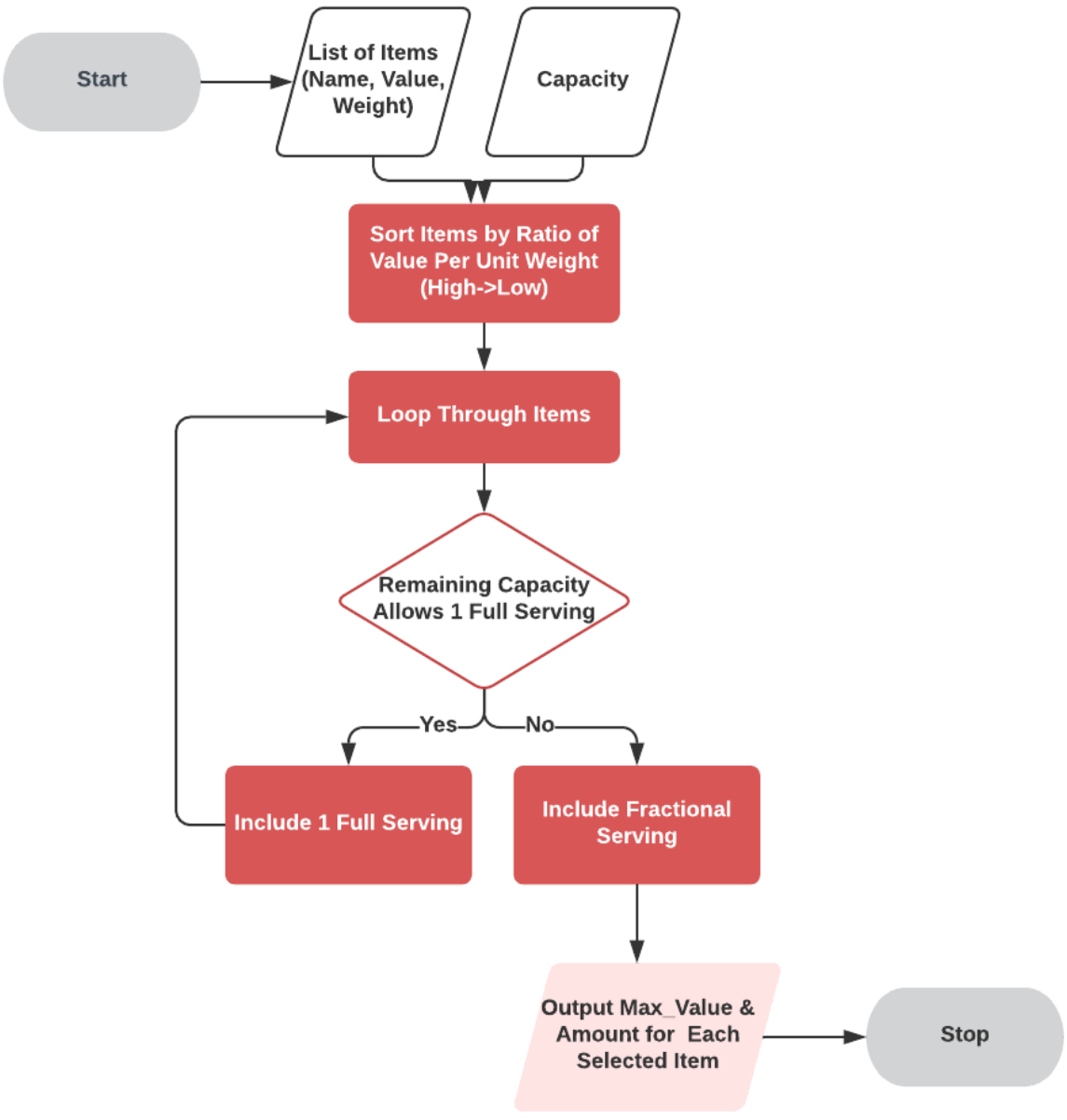
## Diagram Illustration

Greedy Solution Steps:

1. Sort items by value per unit weight (i.e. ratio of value/weight) from high to low
2. Greedily take high ratio items as long as it fits capacity
3. If can't take full serving, take fractional serving
4. Done!

| 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 |



factional_knapsack_greedy.py at…

```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

```

```python
# 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! Stomach is happy, Banana Split is happy.

"Now we need to find the map to go to the Gate of Summit and the Gate of Traceback!"

"Sir, do you know where most kinds of maps are stored?" Dark Knight asks a favor from the owner of the nearby newspaper stand.

"Centermount Academy." he points in that direction.

"Vroom…" the Vessel is flashing by the coconut stand, the peanut stand, the Hawaii nut stand, the xxx stand, yy stand, z stand…

"You're going to get a traffic speeding ticket!!!" the x stand owner is shouting,
yy stand owner: for sure
zzz stand owner: certainly

[Todo: wind recursion image]