Snake game

Phase 2

Mohamed Nasser	20200480
Lilian Stephanos	20200404
Natalie Monged	20200586
Riyad Abdelmenam	20190212

Stage#1:

To apply A* on the snack game, we need the following information and assumptions:

- **Game board representation:** We need a way to represent the game board and the current state of the game. This can be done using a matrix or a graph, where each node represents a state of the game.
- **Starting position:** We need to know the starting position of the snake, which is usually located at a random position on the game board.
- **Goal state:** We need to define the goal state, which is when the snake has eaten all the food on the game board. This will help us determine when to stop the search.
- **Valid moves:** We need to know the valid moves the snake can make, which are typically left, right, up, and down. We also need to ensure that the snake doesn't move into a wall or its own body.
- **Heuristic function:** We need a heuristic function to estimate the distance between the current state and the goal state. One commonly used heuristic is the Manhattan distance, which is the sum of the horizontal and vertical distances between the current state and the goal state.
- **Cost function:** We need to define a cost function to calculate the cost of moving from one state to another. In the snake game, we can use the number of steps taken to reach the new state as the cost.

Assumptions:

- The snake can only move in one direction at a time, and can't move diagonally.
- The snake can only eat food that is adjacent to its head.
- The snake will always move towards the nearest food item, based on the heuristic function.
- The game board is finite and has boundaries, and the snake cannot move outside of these boundaries.
- The game board is always solvable, and there is a path to the goal state.

General initial State:

-the length of the snake is 1

-Score : 0

General goal State:

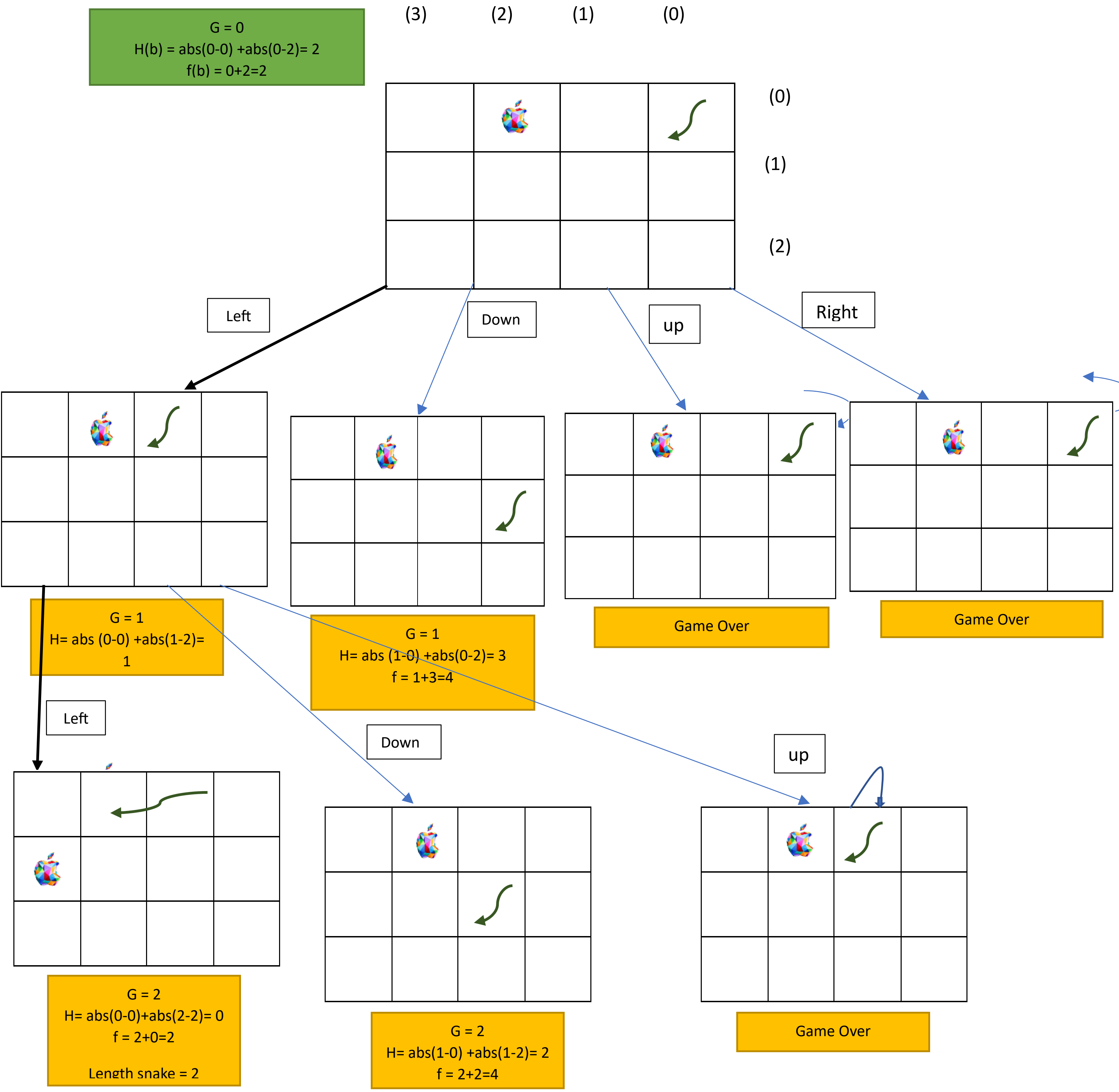
-Maximizing the score and by doing this we will maximize the length of the snake.

-Maximum score: 11

-Maximum length: 12

abs (inital.x - goal.x) + abs (inital.y - goal.y)

f(b) = c(b) + H(b)



G = 0
H(b) = abs(0-0) +abs(0-2)= 2
f(b) = 0+2=2

G = 1
H= abs (0-0) +abs(1-2)=
1

G = 1
H= abs (1-0) +abs(0-2)= 3
f = 1+3=4

Game Over

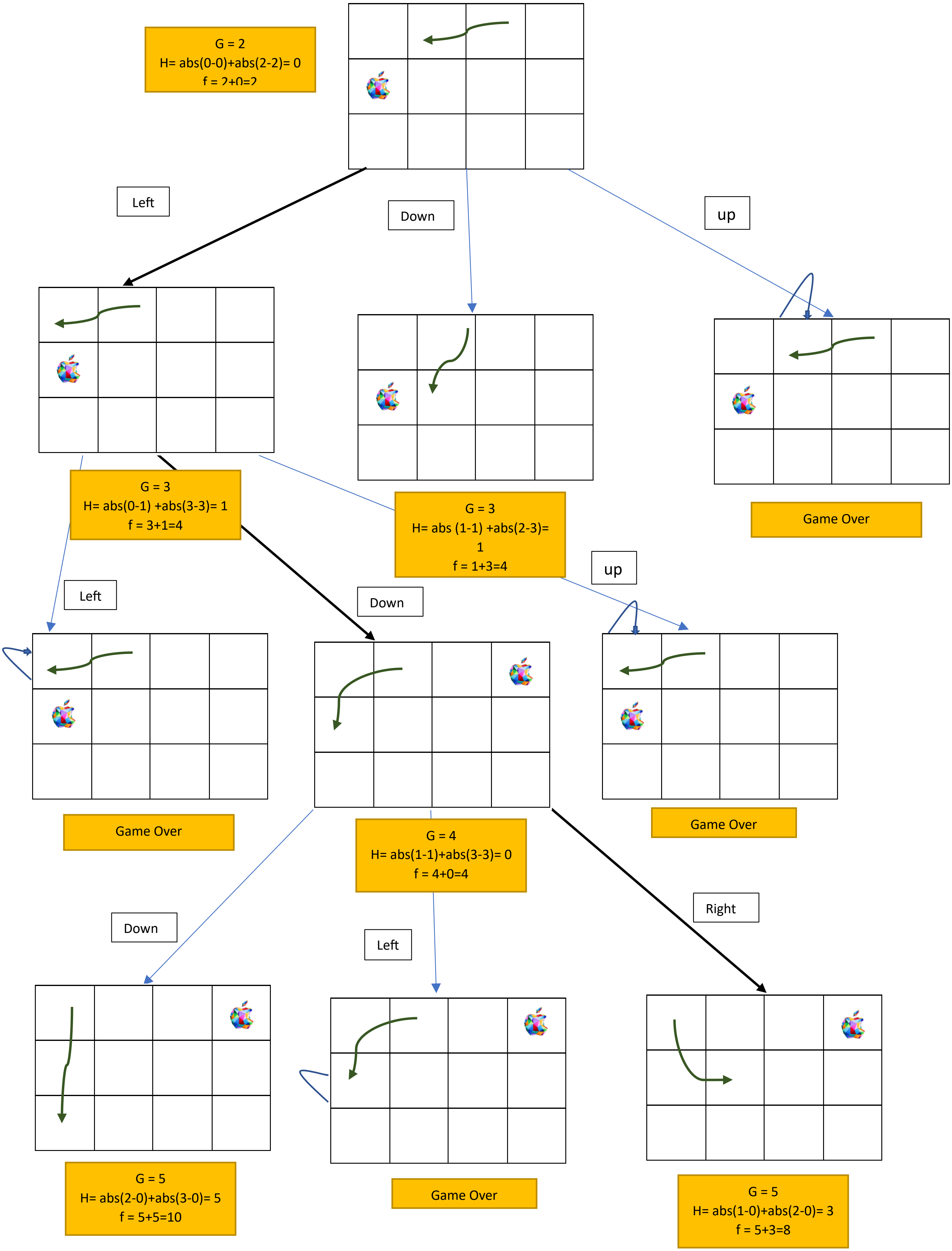
Game Over

G = 2
H= abs(0-0)+abs(2-2)= 0
f = 2+0=2

Length snake = 2

G = 2
H= abs(1-0) +abs(1-2)= 2
f = 2+2=4

Game Over



G = 2
H= abs(0-0)+abs(2-2)= 0
f = 2+0=2

Left

Down

up

G = 3
H= abs(0-1) +abs(3-3)= 1
f = 3+1=4

G = 3
H= abs (1-1) +abs(2-3)= 1
f = 1+3=4

Game Over

up

Left

Down

Game Over

Game Over

G = 4
H= abs(1-1)+abs(3-3)= 0
f = 4+0=4

Down

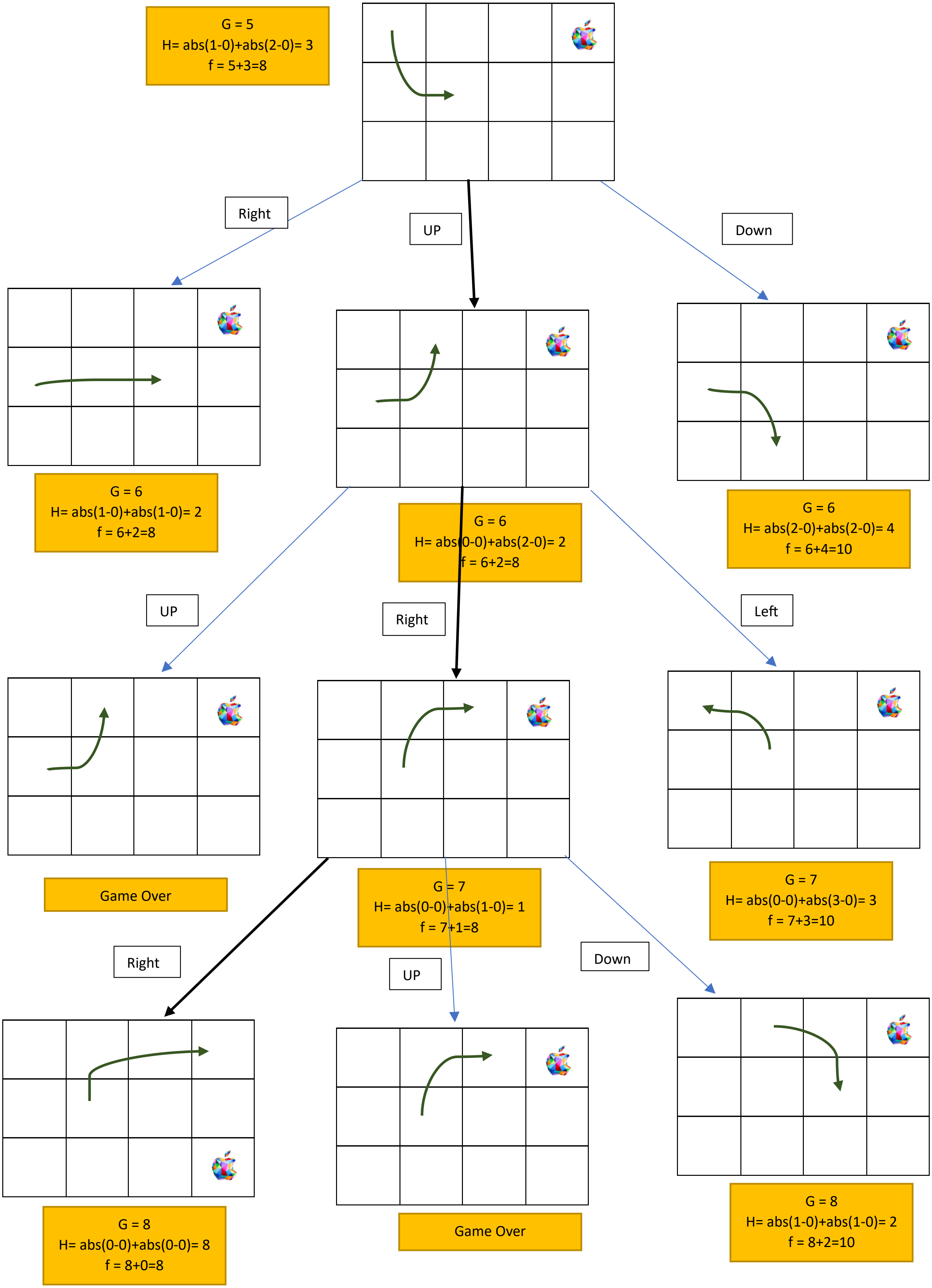
Left

Right

G = 5
H= abs(2-0)+abs(3-0)= 5
f = 5+5=10

Game Over

G = 5
H= abs(1-0)+abs(2-0)= 3
f = 5+3=8



G = 5
H= abs(1-0)+abs(2-0)= 3
f = 5+3=8

Right

UP

Down

G = 6
H= abs(1-0)+abs(1-0)= 2
f = 6+2=8

G = 6
H= abs(0-0)+abs(2-0)= 2
f = 6+2=8

G = 6
H= abs(2-0)+abs(2-0)= 4
f = 6+4=10

UP

Right

Left

Game Over

G = 7
H= abs(0-0)+abs(1-0)= 1
f = 7+1=8

G = 7
H= abs(0-0)+abs(3-0)= 3
f = 7+3=10

Right

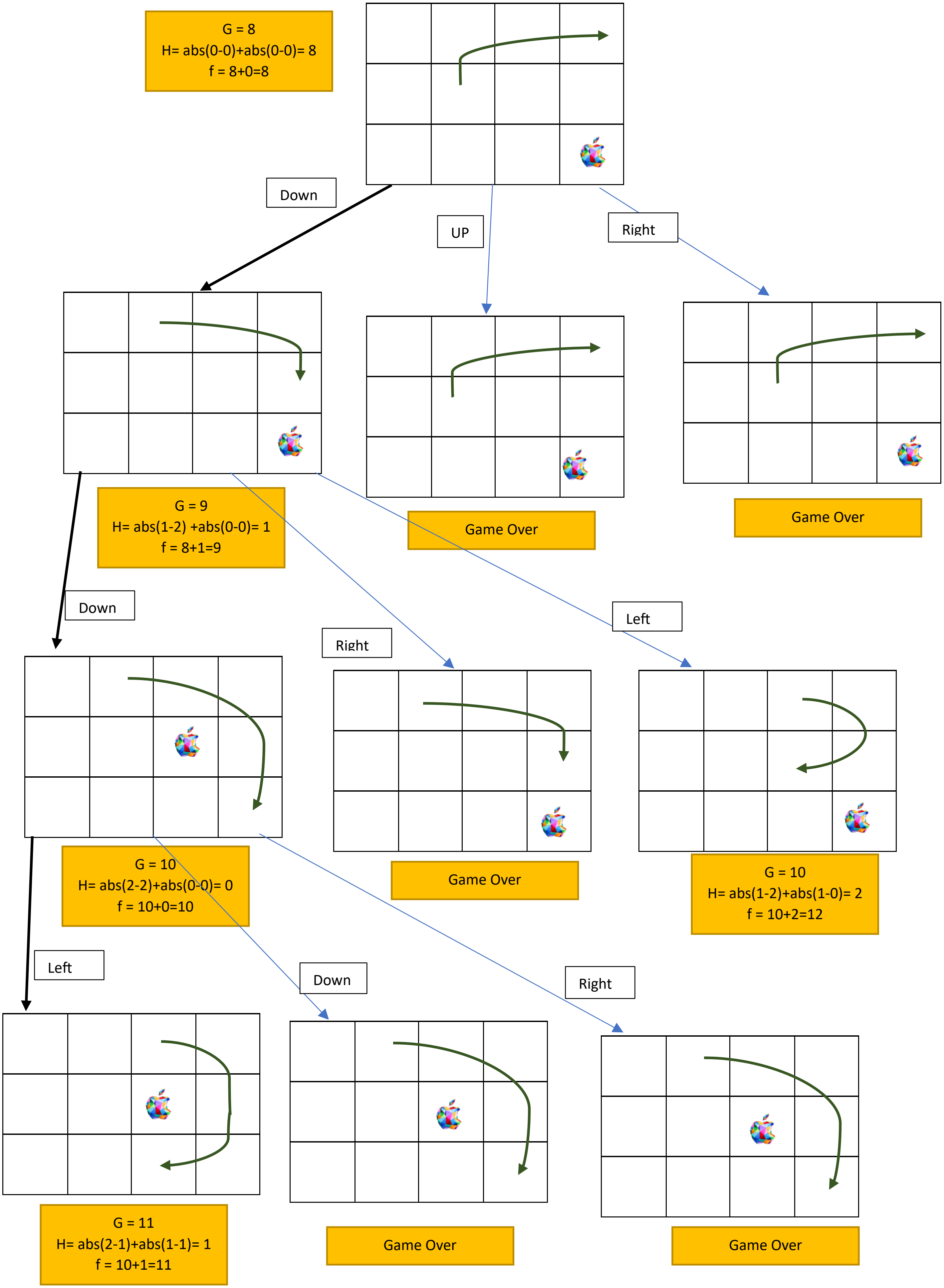
UP

Down

G = 8
H= abs(0-0)+abs(0-0)= 8
f = 8+0=8

Game Over

G = 8
H= abs(1-0)+abs(1-0)= 2
f = 8+2=10



G = 8
H= abs(0-0)+abs(0-0)= 8
f = 8+0=8

G = 9
H= abs(1-2) +abs(0-0)= 1
f = 8+1=9

Game Over

Game Over

G = 10
H= abs(2-2)+abs(0-0)= 0
f = 10+0=10

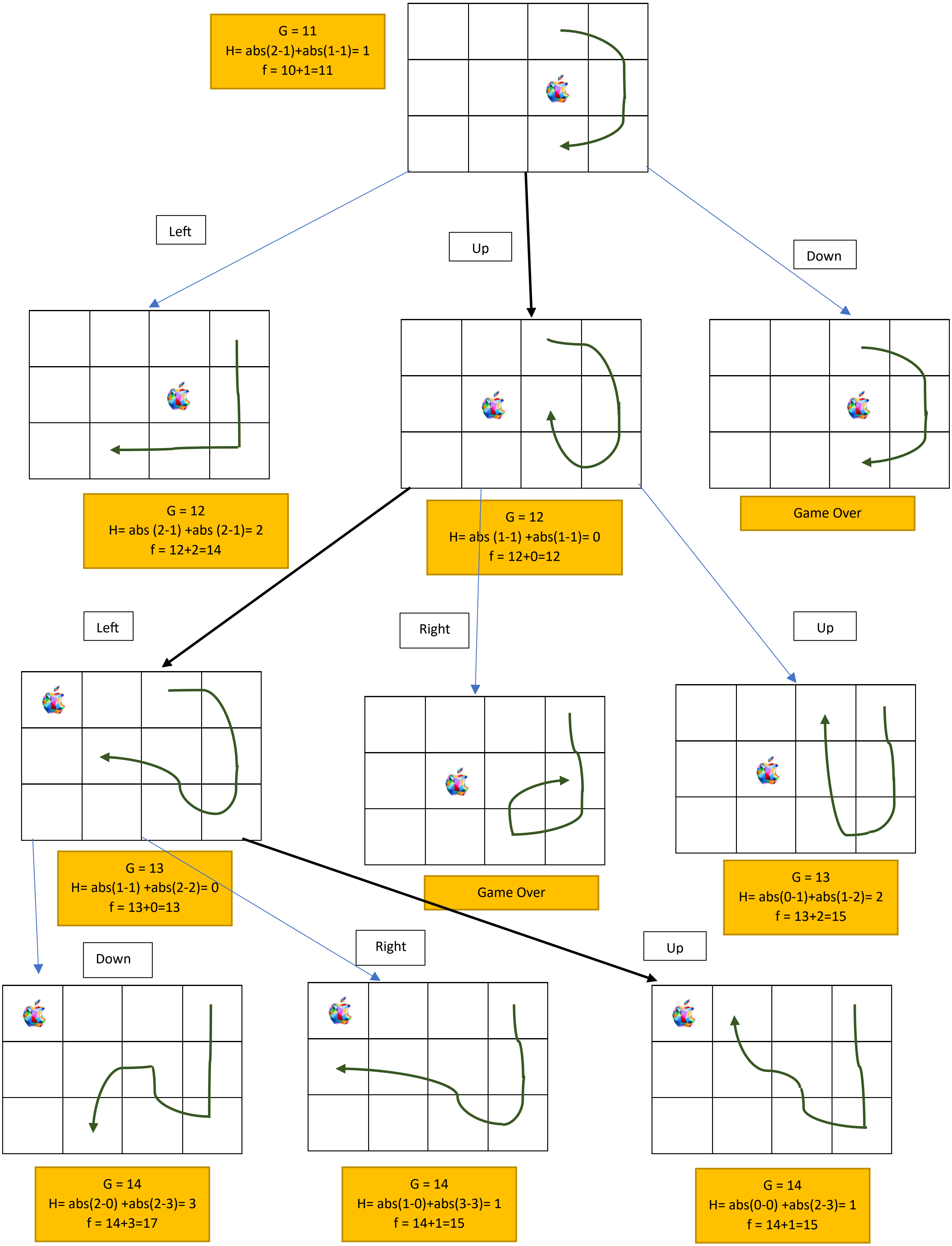
Game Over

G = 10
H= abs(1-2)+abs(1-0)= 2
f = 10+2=12

G = 11
H= abs(2-1)+abs(1-1)= 1
f = 10+1=11

Game Over

Game Over



G = 11
H= abs(2-1)+abs(1-1)= 1
f = 10+1=11

G = 12
H= abs (2-1) +abs (2-1)= 2
f = 12+2=14

G = 12
H= abs (1-1) +abs(1-1)= 0
f = 12+0=12

Game Over

G = 13
H= abs(1-1) +abs(2-2)= 0
f = 13+0=13

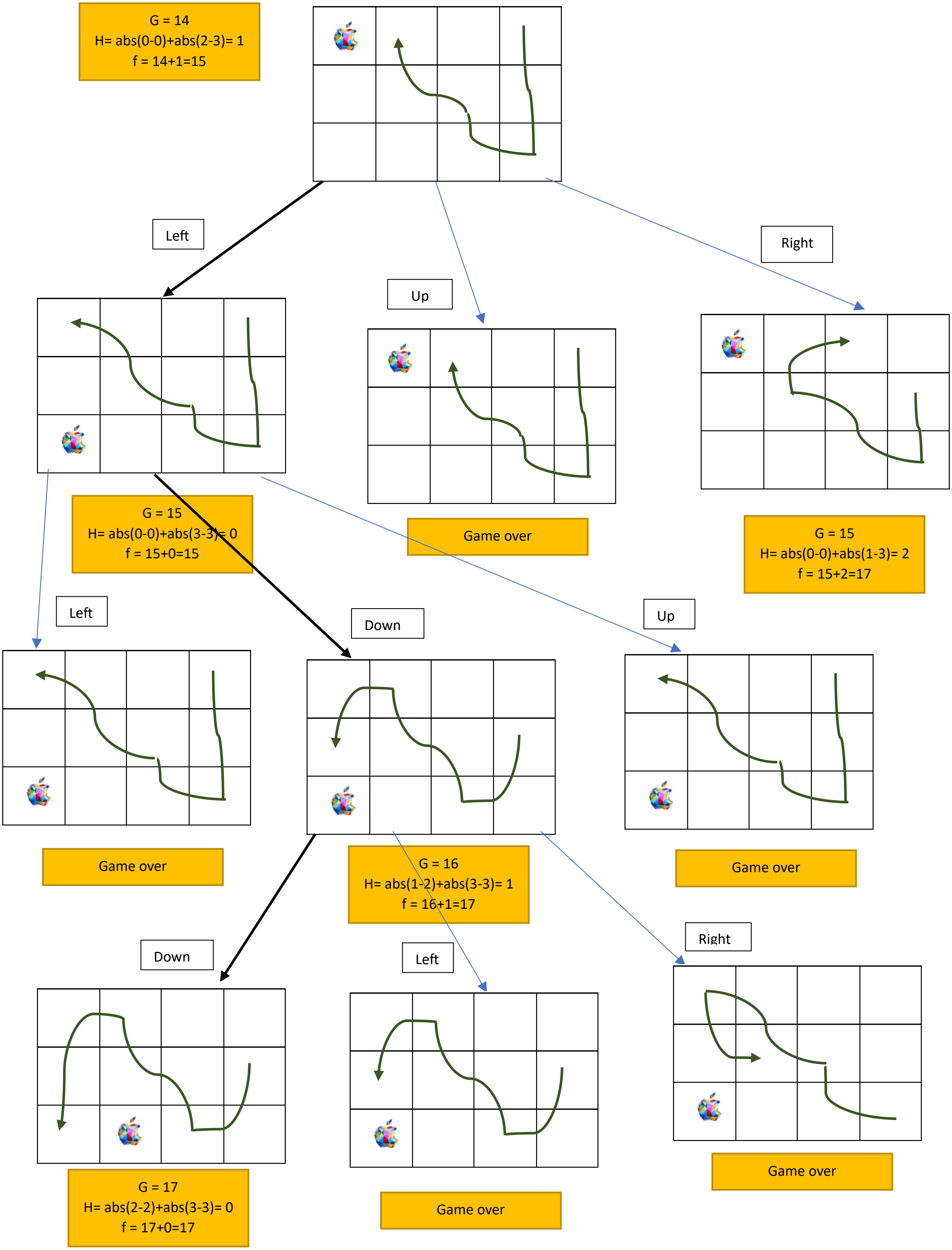
Game Over

G = 13
H= abs(0-1)+abs(1-2)= 2
f = 13+2=15

G = 14
H= abs(2-0) +abs(2-3)= 3
f = 14+3=17

G = 14
H= abs(1-0)+abs(3-3)= 1
f = 14+1=15

G = 14
H= abs(0-0) +abs(2-3)= 1
f = 14+1=15



G = 14
H= abs(0-0)+abs(2-3)= 1
f = 14+1=15

Left

Up

Right

G = 15
H= abs(0-0)+abs(3-3)= 0
f = 15+0=15

Game over

G = 15
H= abs(0-0)+abs(1-3)= 2
f = 15+2=17

Left

Down

Up

Game over

G = 16
H= abs(1-2)+abs(3-3)= 1
f = 16+1=17

Game over

Down

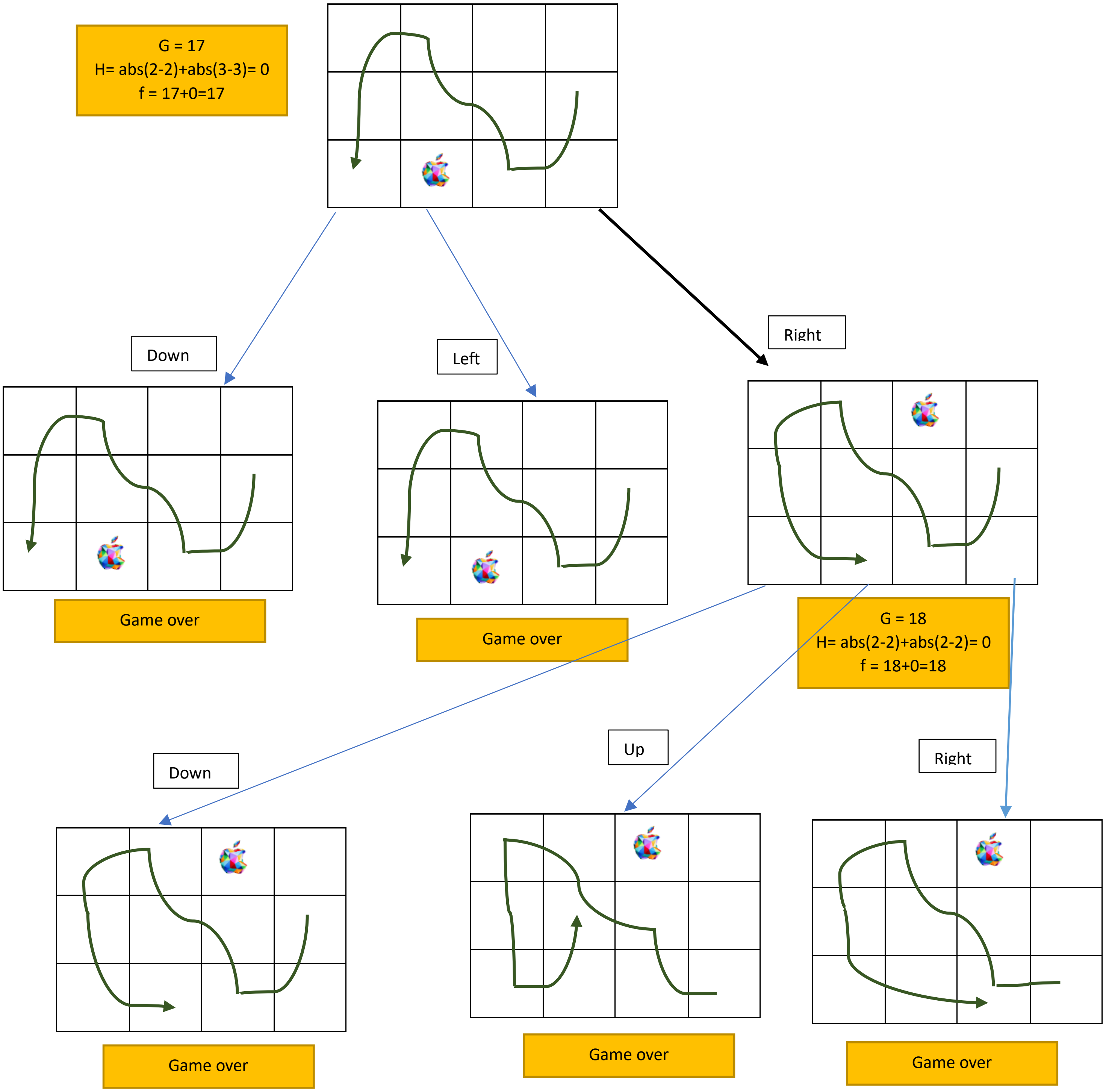
Left

Right

G = 17
H= abs(2-2)+abs(3-3)= 0
f = 17+0=17

Game over

Game over



G = 17
H= abs(2-2)+abs(3-3)= 0
f = 17+0=17

Down

Left

Right

Game over

Game over

G = 18
H= abs(2-2)+abs(2-2)= 0
f = 18+0=18

Down

Up

Right

Game over

Game over

Game over

<u>“STOP”</u>

Stage#2 (Implementation):

does the A* search is the best algorithm for snake game ?

The A* search algorithm can be a good choice for implementing the pathfinding logic in a snake game, as it can efficiently find the shortest path from the snake's current position to the food. However, it may not be the only or the best algorithm for a snake game, as it depends on the specific requirements and design of the game. For example, if the game involves a larger number of obstacles or dynamic obstacles, other algorithms such as Dijkstra's algorithm or swarm intelligence algorithms may be more suitable. Additionally, there may be other factors to consider, such as the size of the game board and the level of complexity desired in the game. Ultimately, the choice of algorithm depends on the specific needs and goals of the game developer.