Code Snippet Repository

Anh Tran - LeetCode

June 30, 2025

1 Python Code Snippets

1.1 Max Area of Island

Breadth first searchs

```python
from collections import deque

class Solution:
    def maxAreaOfIsland(self, grid: List[List[int]]) -> int
        :
        m = len(grid)
        n = len(grid[0])
        count = 0
        self.max_area = -float('inf')

        def bfs(i,j):
            queue = deque()
            queue.append((i,j))
            grid[i][j] = 0 # marked as visited
            area = 1

            while queue:
                ii, jj = queue.popleft()
                for di, dj in [(-1, 0), (1, 0), (0, 1), (0,
                    -1)]:
                    ni, nj = ii + di, jj + dj
                    if 0 <= ni < m and 0 <= nj < n and grid
                        [ni][nj] == 1:
                        queue.append((ni, nj))
                        grid[ni][nj] = 0 # marked as
                            visited
                        area += 1
            self.max_area = max(self.max_area, area)


        for i in range(m):
            for j in range(n):
                if grid[i][j] == 1:
                    bfs(i,j)
                    count += 1
        return max(self.max_area, 0)
```

1.2 Koko eating bananas

Binary Search

```python
import math
class Solution:
    def minEatingSpeed(self, piles: List[int], h: int) ->
        int:
        left, right = 1, max(piles)
        while left < right:
            mid = (left + right) // 2
            # Calculate round up ceil() number of hours
            # hours = sum((pile + mid - 1) // mid for pile
                in piles)
            hours = sum(math.ceil(pile / mid) for pile in
                piles)
            if hours > h:
                left = mid + 1
            else:
                right = mid
        return left
```

1.3 Two-sum

Two Sum

```python
import math
class Solution:
    def minEatingSpeed(self, piles: List[int], h: int) ->
        int:
        left, right = 1, max(piles)
        while left < right:
            mid = (left + right) // 2
            # Calculate round up ceil() number of hours
            # hours = sum((pile + mid - 1) // mid for pile
                in piles)
            hours = sum(math.ceil(pile / mid) for pile in
                piles)
            if hours > h:
                left = mid + 1
            else:
                right = mid
        return left
```