631. Design Excel Sum Formula

Leetcode Link

Problem Explanation

The problem is about creating a simplified Excel spreadsheet using a 2D array as the backend data structure. The height, `H`, and the width, `W`, of the Excel is given at the time of construction where `H` is a positive integer, and `W` is an uppercase letter denoting the maximum width of the Excel sheet. The user should be able to set a value to any cell in the spreadsheet by specifying the row and column. Along with that, the user should be able to get the value stored in any cell. Additionally, the program should support the capability to calculate the sum of a list of cells or a range of cells and store the sum in a particular cell.

Walkthrough of an Example

Let's go through a scenario to understand how a solution might look like.

If we create an Excel using `Excel(3, 'C')`, it means we have an Excel spreadsheet with 3 rows and 3 columns, and all the cells have an initial value of 0.

```
1
2
3    A B C
4  1 0 0 0
5  2 0 0 0
6  3 0 0 0
```

If we execute `Set(1, "A", 2)`, it means we are setting the cell at row 1, column 'A' to 2.

```
1
2
3    A B C
4  1 2 0 0
5  2 0 0 0
6  3 0 0 0
```

Solution Explanation

The approach to solve this problem is by using a 2D array to simulate the Excel spreadsheet. Each cell in this 2D array is an object of type `Cell`. Each `Cell` object stores the value of the cell and an unordered map of cells that contribute to its value. If a sum operation is being performed, then the `posCount` map in the cell where the sum is being stored, will keep track of each cell's position that contributes to its value, along with the corresponding contribution count.

For example, in the initial scenario where Excel of size `(3, 'C')` was created, the 2D array would be initialized to all zeros:

```
1
2
3 [
4   [0, 0, 0],
5   [0, 0, 0],
6   [0, 0, 0]
7 ]
```

Python Solution

```
1
2 python
3 class Cell:
4     def __init__(self):
5         self.val = 0
6         self.posCount = {}
7
8 class Excel:
9
10     def __init__(self, H: int, W: str):
11         self.width = ord(W) - ord('A') + 1
12         self.sheet = [[Cell() for _ in range(self.width)] for _ in range(H)]
13
14     def set(self, r: int, c: str, v: int) -> None:
15         self.sheet[r-1][ord(c)-ord('A')].val = v
16         self.sheet[r-1][ord(c)-ord('A')].posCount = {}
17
18     def get(self, r: int, c: str) -> int:
19         cell = self.sheet[r-1][ord(c)-ord('A')]
20         return cell.val if not cell.posCount else self.sum(r, c, [f"{k[0]}{k[1]}" for k in cell.posCount.keys()])
21
22     def sum(self, r: int, c: str, strs: List[str]) -> int:
23         self.sheet[r-1][ord(c)-ord('A')].posCount = {}
24         total = 0
25         for st in strs:
26             if ':' not in st:
27                 val = self.get(int(st[1:]), st[0])
28                 self.sheet[r-1][ord(c)-ord('A')].posCount[(st[0], int(st[1:]))] = 1
29                 total += val
30             else:
31                 st_split = st.split(':')
32                 r1, c1, r2, c2 = int(st_split[0][1:]),st_split[0][0], int(st_split[1][1:]), st_split[1][0]
33                 for row in range(r1, r2+1):
34                     for col in range(ord(c1)-ord('A'), ord(c2)-ord('A')+1):
35                         total += self.sheet[row][col].val
36                         self.sheet[r-1][ord(c)-ord('A')].posCount[(chr(ord('A') + col), row+1)] = 1
37         self.sheet[r-1][ord(c)-ord('A')].val = total
38         return total
39
```

In this code we use a `Cell` class which will store the value of the cell along with a map `posCount` of cells that contribute to its value. In the constructor we initialize `width` and `sheet`. In the `set` method we are setting a specific cell's value and resetting its `posCount`. For the `get` method, if a cell's value is being calculated from other cells, we have to calculate it, otherwise return the value of the cell directly. The `sum` method updates the `posCount` for a cell whose value will be computed by the sum of other cells or ranges of cells. It returns this sum.

JavaScript Solution

```
1
2 javascript
3 class Cell {
4     constructor() {
5         this.val = 0;
6         this.posCount = {};
7     }
8 }
9
10 class Excel {
11     constructor(H, W) {
12         this.width = W.charCodeAt(0) - 'A'.charCodeAt(0) + 1;
13         this.sheet = new Array(H).fill(0).map(() => new Array(this.width).fill(new Cell()));
14     }
15
16     set(r, c, v) {
17         this.sheet[r][c.charCodeAt(0) - 'A'.charCodeAt(0)] = new Cell();
18         this.sheet[r][c.charCodeAt(0) - 'A'.charCodeAt(0)].val = v;
19     }
20
21     get(r, c) {
22         let cell = this.sheet[r][c.charCodeAt(0) - 'A'.charCodeAt(0)];
23         if (!cell.posCount) {
24             return cell.val;
25         } else {
26             return this.sum(r, c, Object.keys(cell.posCount));
27         }
28     }
29
30     sum(r, c, strs) {
31         this.sheet[r][c.charCodeAt(0) - 'A'.charCodeAt(0)] = new Cell();
32         let total = 0;
33         for (let str of strs) {
34             if (!str.includes(":")) {
35                 let val = this.get(Number(str.slice(1)), str[0]);
36                 this.sheet[r][c.charCodeAt(0) - 'A'.charCodeAt(0)].posCount[str] = 1;
37                 total += val;
38             } else {
39                 let [start, end] = str.split(":");
40                 let [r1, c1, r2, c2] = [Number(start.slice(1)), start[0], Number(end.slice(1)), end[0]];
41                 for (let row = r1; row <= r2; row++) {
42                     for (let col = c1.charCodeAt(0) - 'A'.charCodeAt(0); col <= c2.charCodeAt(0) - 'A'.charCodeAt(0); col++) {
43                         total += this.sheet[row][col].val;
44                         let cellPos = `${String.fromCharCode(65 + col)}${row+1}`;
45                         this.sheet[r][c.charCodeAt(0) - 'A'.charCodeAt(0)].posCount[cellPos] = 1;
46                     }
47                 }
48             }
49         }
50         this.sheet[r][c.charCodeAt(0) - 'A'.charCodeAt(0)].val = total;
51         return total;
52     }
53 }
```

Another similar approach used except in JavaScript. Notice that the conversion between char and ASCII value is done using `charCodeAt(0)`.

Java Solution

```
1
2 java
3 class Excel {
4     class Cell {
5         int val = 0;
6         Map<String, Integer> cells = new HashMap<>();
7     }
8
9     Cell[][] table;
10
11     public Excel(int H, char W) {
12         table = new Cell[H+1][W-'A'+1];
13         for(int i=0; i <= H; i++){
14             for(int j=0; j <= W-'A'; j++){
15                 table[i][j] = new Cell();
16             }
17         }
18     }
19
20     public void set(int r, String c, int v) {
21         Cell cell = table[r][c.charAt(0)-'A'];
22         cell.val = v;
23         cell.cells.clear();
24     }
25
26     public int get(int r, String c) {
27         if(table[r][c.charAt(0)-'A'].cells.isEmpty()){
28             return table[r][c.charAt(0)-'A'].val;
29         } else {
30             return sum(r, c, new String[]{c});
31         }
32     }
33
34     public int sum(int r, String c, String[] strs) {
35         Cell cell = table[r][c.charAt(0)-'A'];
36         cell.cells.clear();
37         int total = 0;
38         for(String s : strs){
39             if(s.indexOf(":") < 0){
40                 total += get(Integer.parseInt(s.substring(1)), s.substring(0,1));
41                 cell.cells.put(s, cell.cells.getOrDefault(s, 0) + 1);
42             } else {
43                 String[] range = s.split(":");
44                 int rowStart = Integer.parseInt(range[0].substring(1));
45                 char colStart = range[0].charAt(0);
46                 int rowEnd = Integer.parseInt(range[1].substring(1));
47                 char colEnd = range[1].charAt(0);
48                 for(int i=rowStart; i<=rowEnd; i++){
49                     for(char j=colStart; j<=colEnd; j++){
50                         total += get(i, String.valueOf(j));
51                         String key = j + String.valueOf(i);
52                         cell.cells.put(key, cell.cells.getOrDefault(key, 0) + 1);
53                     }
54                 }
55             }
56         }
57         cell.val = total;
58         return total;
59     }
60 }
```

The Java solution uses a `Cell` class similar to the previous solutions, with a `val` attribute to store the value of the cell and a `cells` HashMap to store any cells that contribute to its value. The `set` and `get` methods are straightforward, while the `sum` method involves more complex logic to deal with ranges. If a range is provided, the range is parsed and a nested loop is used to loop through the cells in the range, while updating the total and the cell HashMap as needed. The final total sum is then stored in the cell and returned.