

Name: Agoorukasetty Adithya

Email: adithyaak18@gmail.com

Question:

Airplane Seating Algorithm

Write a program that helps seat audiences in a flight based on the following input and rules.

Rules for seating

- 1. Always seat passengers starting from the front row to back, starting from the left to the right**
- 2. Fill aisle seats first followed by window seats followed by center seats (any order in center seats)**

Input to the program will be

- 1. A 2D array that represents the columns and rows - Ex. `[[3,4], [4,5], [2,3], [3,4]]`**
- 2. Number of passengers waiting in the queue.**

Python code:

```
arr = [[3, 2], [4, 3], [2, 3], [3, 4]]
n = len(arr)
count = 1
k = 30
p = 0
q = -1
aisle = []
aisle.append(arr[0][0] - 1)
sum = arr[0][0] - 1
i = 1
while (i < n - 1) :
    if (arr[i][0] != 2) :
        sum += 1
        aisle.append(sum)
        sum += arr[i][0] - 1
        aisle.append(sum)
    else :
        sum += 1
        aisle.append(sum)
        sum += 1
        aisle.append(sum)
    i += 1
aisle.append(sum + 1)
i = 0
while (i < n) :
    p += arr[i][0]
    if (arr[i][1] > q) :
        q = arr[i][1]
    i += 1
window = []
window.append(0)
window.append(p - 1)
center = []
i = 0
while (i < p) :
    if (not i in aisle and not i in window) :
        center.append(i)
    i += 1
arr2 = [[0] * (p) for _ in range(q)]
i = 0
while (i < q) :
    j = 0
    while (j < p) :
        arr2[i][j] = 0
        j += 1
    i += 1
sum2 = -1
```

```

limit = []
limit.append(0)
i = 0
while (i < n) :
    sum2 += arr[i][0]
    limit.append(sum2)
    i += 1
map = dict()
map[0] = arr[0][1]
i = 0
while (i < n) :
    j = limit[i] + 1
    while (j <= limit[i + 1]) :
        map[j] = arr[i][1]
        j += 1
    i += 1

i = 0
while (i < q) :
    j = 0
    while (j < p) :
        if (j in aisle and i < map.get(j)) :
            arr2[i][j] = count
            count += 1
        j += 1
    i += 1
i = 0
while (i < q) :
    j = 0
    while (j < p) :
        if (j in window and i < map.get(j)) :
            arr2[i][j] = count
            count += 1
        j += 1
    i += 1
i = 0
while (i < q) :
    j = 0
    while (j < p) :
        if (j in center and i < map.get(j)) :
            arr2[i][j] = count
            count += 1
        j += 1
    i += 1
i = 0
while (i < q) :
    j = 0

```

```

while (i < q) :
    j = 0
    while (j < p) :
        if (arr2[i][j] != 0 and arr2[i][j] <= 30) :
            print(str(arr2[i][j]) + " ", end = "")
        elif(arr2[i][j]>30):
            print("XX ", end = "")
        else:
            print(" ",end="")
        j += 1
    print(" ")
    i += 1

```

Output:

```

----- RESTART: C:\Users\adithya\Desktop\te.
19 25 1 2 26 27 3 4 5 6 28 20
21 29 7 8 30 XX 9 10 11 12 XX 22
      13 XX XX 14 15 16 17 XX 23
                        18 XX 24
>>>

```

Ruby Code:

```
1  arr = [[3, 2], [4, 3], [2, 3], [3, 4]]
2  n = arr.length
3  count = 1
4  k = 30
5  p = 0
6  q = -1
7  aisle = []
8  aisle.append(arr[0][0] - 1)
9  sum = arr[0][0] - 1
10 i = 1
11 while (i < n - 1)
12   if (arr[i][0] != 2)
13     sum += 1
14     aisle.append(sum)
15     sum += arr[i][0] - 1
16     aisle.append(sum)
17
18   else
19     sum += 1
20     aisle.append(sum)
21     sum += 1
22     aisle.append(sum)
23   end
24   i += 1
25 end
26 aisle.append(sum + 1)
27 i = 0
28 while (i < n)
29   p += arr[i][0]
30   if (arr[i][1] > q)
31     q = arr[i][1]
32   end
33   i += 1
34 end
```

```

38 window = []
39 window.append(0)
40 window.append(p - 1)
41 center = []
42 i = 0
43
44 while (i < p)
45     if (!aisle.include?(i) and !window.include?(i))
46         center.append(i)
47     end
48     i += 1
49 end
50 arr2= Array.new(q) { Array.new(p) { 0 } }
51 i = 0
52 while (i < q)
53     j = 0
54     while (j < p)
55         arr2[i][j] = 0
56         j += 1
57     end
58     i += 1
59 end
60 sum2 = -1
61 limit = []
62 limit.append(0)
63 i = 0
64 while (i < n)
65     sum2 += arr[i][0]
66     limit.append(sum2)
67     i += 1
68 end
69 i=0
70 map = Hash.new
71 map[0] = arr[0][1]
72
73 while (i < n)
74     j = limit[i] + 1

```

```

74     j = limit[i] + 1
75     while (j <= limit[i + 1])
76     |   |   map[j] = arr[i][1]
77     |   |   j += 1
78     |   |   end
79     |   |   i += 1
80     end
81     i = 0
82     while (i < q)
83     |   |   j = 0
84     |   |   while (j < p)
85     |   |   |   |   if (aisle.include?(j) and i < map[j])
86     |   |   |   |   |   |   arr2[i][j] = count
87     |   |   |   |   |   |   count += 1
88     |   |   |   |   |   |   end
89     |   |   |   |   |   |   j += 1
90     |   |   |   |   |   |   end
91     |   |   |   |   |   |   i += 1
92     |   |   |   |   |   |   end
93     |   |   |   |   |   |   i = 0
94
95     while (i < q)
96     |   |   j = 0
97     |   |   while (j < p)
98     |   |   |   |   if (window.include?(j) and i < map[j])
99     |   |   |   |   |   |   arr2[i][j] = count
100    |   |   |   |   |   |   count += 1
101    |   |   |   |   |   |   end
102    |   |   |   |   |   |   j += 1
103    |   |   |   |   |   |   end
104    |   |   |   |   |   |   i += 1
105    |   |   |   |   |   |   end
106    |   |   |   |   |   |   i = 0
107
108    while (i < q)
109    |   |   j = 0
110    |   |   while (j < p)

```

```

102         j += 1
103     end
104     i += 1
105 end
106 i = 0
107
108 while (i < q)
109     j = 0
110     while (j < p)
111         if (center.include?(j) and i < map[j])
112             arr2[i][j] = count
113             count += 1
114         end
115         j += 1
116     end
117     i += 1
118 end
119 i = 0
120
121 while (i < q)
122     j = 0
123     while (j < p)
124         if (arr2[i][j] != 0 and arr2[i][j] <= 30)
125             print((arr2[i][j]))
126             print(" ")
127
128             elsif(arr2[i][j]>30)
129                 print("xx ")
130
131             else
132                 print("  ")
133             end
134         j += 1
135     end
136     puts(" ")
137     i += 1
138 end

```


Output of the Ruby Code:

```
C:\Users\adithya\Desktop\test>ruby volo2.rb
19 25 1 2 26 27 3 4 5 6 28 20
21 29 7 8 30 XX 9 10 11 12 XX 22
      13 XX XX 14 15 16 17 XX 23
                        18 XX 24
```

The test case in which only two columns are present and this leads to both the columns being the aisle seats is also present.